UiO : Department of Mathematics
University of Oslo

Invertible and Pseudo-Invertible Encoders

An Approach to Inverse Problems with Neural Networks

Marius Aasan

Master's Thesis, Autumn 2021

This master's thesis is submitted under the master's programme *Data Science*, with programme option *Statistics and Machine Learning*, at the Department of Mathematics, University of Oslo. The scope of the thesis is 60 credits.

The front page depicts a section of the root system of the exceptional Lie group E_8, projected into the plane. Lie groups were invented by the Norwegian mathematician Sophus Lie (1842–1899) to express symmetries in differential equations and today they play a central role in various parts of mathematics.

Abstract

While neural networks have been demonstrated to be highly successful in mathematical and statistical modelling of a comprehensive selection of problems, their application to inverse problems is not without complications. Recent works have shown that neural networks are especially prone to stability issues – both in a classical sense, and in the context of so called *adverserial attacks* which has come to be regarded as the most pervasive source of instability in modern neural network models.

Concomitantly, methods of constructing *invertible neural networks* with diffeomorphic layer structures with *normalizing flows* have been proposed as an interesting method for approaching inverse problems by probabilistic augmentation of latent variable models to induce full-rank in a conditional setting. However, these models can often be prohibitively expensive in terms of computational efficiency and memory usage while displaying sufficiently different architectures as to not be trivially extendable to tools for commonly defined feed-forward neural networks.

In this thesis, we motivate the theory of inverse problems via integral equations and spectral theory and discuss the connection of statistical learning theory to neural networks with special focus on encoder-decoder models. Furthermore, in the context of neural networks, we will discuss the underlying theory of epistemic and aleatoric uncertainty, discuss the role of probabilistic modelling, and evaluate the idea of latent probabilistic completion as a remedial method for undercomplete modelling tasks.

Our contribution to the subject of invertible neural networks can be summarized as follows. We propose a relatively simple architectural modification of existing encoder-decoder models using both implicit and explicit orthogonal constraints using Riemannian manifold learning and resolvent operators to construct a wider class of invertible neural networks which are compatible with classic feed-forward architectures. We show that these models provide both a significant decrease in the parameter space compared to standard encoder-decoder networks, as well as theoretical guarantees of robustness and stability without significant loss to model performance. We apply these architectures in combination with existing variational Bayesian methods for a generative approach to underdetermined inverse problems. To this end, we introduce a class of piecewise diffeomorphic activation functions and a bijective Gaussian to Dirichlet transformation for latent variables as an alternative to the canonical Softmax transformation, and propose the application of simple conditional additive coupling layers to improve conditioning in generative models.

Acknowledgements

Firstly, I would like to thank my family. My parents Paul and Sigrun as well as my sister Camilla are always supportive of me, whatever I choose to do in my life – no matter how unconventional my choices are. They will always have my unending appreciation and love. Secondly, my girlfriend Nina and my close friends Keith, Kristoffer, Reza, and Jo have been there for me throughout both joyous and difficult times. Without you, I would not be where I am today. Thirdly, I am very much indebted to my fellow students, which are probably too numerous to mention, but I will give it a shot.

- The *kernel* of my undergraduate mathematics studies; Åsmund Danielsen Kvitvang, Simen Westbye Moe, Fredrik Jabeer Mahal Nordeng, Are Aamot, Lars Peder Fjellberg, Marius Havgar, Edvard Aksnes, Ola Sande, Yvonne Neraal, Eva Steine Dahl, Max Magnus Nils Rafstedt, Simon Foldvik, Lars Henry Berge Olsen, Lars Gabrielsen, Elisabeth Hagen, Fride Josefine Emberland Straum, Ingvild Hjortnæs Larsen, Teah Kaasa McLean and many more.

- My upperclassmen and mentors; Ivar Haugaløkken Stangeby, Camilla Lingjærde, Tale Dudde, Kristine Heimdal, Henrik Aasen Kjeldsberg, Luca Gazdag, Magnus Vodrup, Kristoffer Ulvik Høisæther, and of course Paul Aleksander Maugesten. Thank you for providing a strong academic and social foundation to me and the rest of the students at MAMI/MIT.

- I would also very much like to thank my data science colleagues, especially Erik Lien Bolager as well as Fabian Bull and Ingebjørg Sævareid for being great partners in our early work on our masters degrees, as well as good friends and comrades.

Last – but by no means least – I am greatly indebted to my academic tutors and advisors throughout my studies. My supervisor Odd Kolbjørnsen has been nothing but supportive and encouraging throughout my work on my thesis and has provided me with good feedback, interesting discussions, and challenges. I would also like to thank my supervisor for my bachelor thesis – Tom Louis Lindstrøm – as well as my co-advisors from the Bayesian neural network group; Geir Olve Storvik and Anne Schistad Solberg. They have always had an open door with me and I have felt supported and welcomed by them. I would also like to express gratitude to Vegard Antun, whose work and helpful discussion has been especially helpful in the development of my thesis.

Contents

CHAPTER 1

Preliminaries

1.1 Introduction

An inverse problem is a product of a seemingly reasonable inquiry. Assume we have a mathematical model which provides reliable outputs for some mapping. Furthermore, assume that we are provided a set of measured outputs for this particular mapping, and for which we are uncertain of what inputs are generated through these specific outputs. Is it then possible to compute the causal inputs using the effective outputs, or in other words; is it possible to invert the mapping to give reasonable estimates of the input values?

Connecting effects to causes has certainly satiated minds as far back as rational thought itself, and prominent mathematicians have long been utilizing the power of inversion methods. Carl Gustav Jacobi is attributed the quote *'man muss immer umkehren'* or *'invert, always invert'* [Van16]. His preferred approach to a new problem was to reformulate an inverse representation, which he stated would often lead to the solution coming to him in a more effortless manner. The advent of such important disciplines as quantum mechanics and the general theory of nonlinear systems succoured the need for new paradigms in mathematics. The most important of these methods was pioneered by Andrey Tikhonov [Tik43], whose research directly ushered the theory of regularization for robust solutions of inverse problems. At the same time, the Bayesian approach to statistical modelling provides a probabilistic framework for robust modelling of inverse problems.

Inverse problems find a natural formulation in image reconstruction tasks. Until recently, models for data-driven image analysis have been computationally expensive and relied heavily on specialized hand-crafted algorithms for each individual problem. The current resurgence of neural network models, originally conceived by McCullough and Pitts [MP43], have completely revolutionized the field of image analysis. Much of the success of neural network models stem from the development of Convolutional Neural Networks (CNNs), whose origins go back to the neocognitron [Fuk79], further developed in [LeC+89]. Even more recently, the application of *invertible neural networks* has been proposed as a particularly interesting approach to modelling inverse problems [Ard+18]. This thesis is an investigation of the intersection of inverse problems in the context of image reconstruction and invertible neural networks.

1.2 Project Description and Goals

The project description of the master thesis outlined four main goals of the thesis. The focus of the thesis is

1. to exposition inverse problems and the related mathematical framework,
2. to exposition statistical learning and neural networks in the context of inverse problems,
3. to examine current methods for modelling inverse problems with neural networks,
4. to investigate potentially new models for solving inverse problems, and comparing these with comparative models.

In Part I, we discuss the general fundamental theoretical background of inverse problems, introduce statistical learning and data-driven methods for solving inverse problems, and provide an exposition of neural networks with particular focus on encoder-decoder models. In Part II, we discuss methods for solving inverse problems in the context of neural networks, and present and motivate our approach of *invertible- and pseudo-invertible encoders*. In Part III, we present experimental results, discuss the implications, strengths and weaknesses, and provide relevant discussion on further research opportunities for this class of models.

1.3 Implementations and Software

We exclusively used the Python programming language using PyTorch (v1.1.0 through v1.10.0) [Pas+19] for our implementations and experiments. The PyTorch framework provides tools and modules for constructing neural networks, although most of our proposed models cannot be implemented using the standard tools provided in the framework. There was therefore a need for custom classes and modules for neural networks with methods for adjoint- and inverse operators. We will briefly discuss our implementation and methods.

We programmed custom modules for adjoint and invertible operators for dense, convolutional, as well as seperable patch layers (Definition 4.3.3). Additionally, special modules were programmed for orthogonal layer structures with specific methods for updating their weight and base operators via Riemannian gradient descent, as well as the resolvent layers (see Section 6.4). To easily combine these modules for invertible networks, we also developed specific tools for inverse splitting, permutation, and composition of sequential blocks with specific methods for adjoint and inverse computation. Furthermore, all proposed activation functions (see Sections 6.2 and 6.3) were programmed as modular layer structures, and we implemented specific tools for computing relevant metrics and objectives.

Most of the challenges we faced during the programming of these structures were in relation to convolutional layer modules. When we started our work, the version of PyTorch we used (v.1.1.0) did not have specific modules for circular boundary conditions, which was essential for our purposes. Furthermore, the built-in tools for the Fourier transform were limited and did not allow for gradient computations. We initially implemented convolution operators explicitly by constructing sparse circulant Toeplitz matrices, which were computationally expensive. Furthermore, sparse matrix representations could not be successively applied to built-in optimization methods. As we progressed, the PyTorch framework received updates which featured circular boundary conditions for convolution as well as gradient computation for the Fourier transform (v.1.7.0). However, circular boundary conditions with transposed convolution operators are still not implemented. As these are coded in low-level implementations in C using the CUDA framework [NVF20], we did not have the resources to construct specialized code for this purpose. These difficulties limited the scope of our experiments on convolutional invertible encoders.

Furthermore, the computation of network Jacobians in PyTorch has certain stability issues, which manifests as errors if the computational graph for backpropigation becomes too complex, which is the case for invertible encoder networks, as the same parameters feature multiple times in a graph. This leads to issues in computing the relative condition numbers via the Jacobian in certain models. Another issue is that our implementations of these computations are quite memory expensive, sometimes requiring 100+ GB of RAM, so any researcher who wants to individually verify our results should be aware of the computational overhead involved.

The source code of our repository is available on GitHub[1], and our experiments were conducted using the Jupyter Notebook format with fixed seeding of the random number generator to promote reproducibility. Note that seeding is often dependent on the underlying operating system as well as CPU/GPU. Our experiments were conducted on the ML-Nodes at UiO with RTX2080Ti GPUs and Intel Xeon CPUs.

1.4 Summary of Contributions

- Our definition of the so-called seperable patch layers (Definition 4.3.3) are directly inspired by the model proposed in [Tol+21]. Our contribution is limited to the generalized formulation as composable single network layers.
- The conditional additive coupling layers from Definition 4.3.6, which are constructed via convex combinations to act as a conditional bias term for stochastic sampling layers is an original contribution.
- The results from Proposition 4.4.4 and Definition 4.4.3 for estimating sensitivity imbalance based on the notion of a sensitivity equilibrium between forward and inverse compact operators are original contributions.
- Invertible encoder networks (Corollary 6.1.7) and the related results on invertible and pseudo-invertible encoder networks (Definitions 6.4.1 and 6.1.2, Propositions 6.1.5 and 6.1.6, and Lemma 6.1.3) are – to the best of our knowledge – a new approach to invertible neural networks.
- All bi-Lipschitzian piecewise diffeomorphic activation functions proposed in Section 6.2 are original contributions.

[1] https://github.com/PolterZeit/invertible_encoders

- Our proposed invertible Dirichlet-Softmax transform from Proposition 6.3.4 is an original contribution.
- The theory of Riemannian manifold learning is based on [AMS08; Cas19], and our contribution is limited to the software implementation for manifold learning on $\mathcal{SO}(n), \mathrm{St}(n, k)$ and the applications of these methods for constructing invertible neural networks.
- The results on invertible resolvent operators in neural networks (Propositions 6.4.15 and 6.4.16) are original contributions.
- Lastly, we note that all our custom modules and software for applying invertible encoders in neural networks in PyTorch are original contributions.

1.5 Prerequisites and Fundamentals

When writing a thesis in a field as broad as Data Science, one needs to consider the variation in the mathematical background of possible readers. Thus, certain definitions and results that may be familiar to some might be unknown to others. To accommodate for this, we outline some fundamental results in Appendix A to properly define the mathematical tools behind the argumentation in the text, and to adhere to a level of rigour.

However, anyone who finds themselves sifting through a summary of mathematical definitions is almost certainly going to find the reading experience tedious, so the the appendix is accompanied by a summary of the most relevant results, as well as a list of notational conventions used in this thesis with references to relevant definitions and results. Interested readers will find a more comprehensive discussion on these topics in [AMS08; Bil95; Hal15; Lin17; MW99; Rud87; RY08; Sch95].

Selected Fundamental Results

Hilbert Space	(Definition A.1.15).
Compact Operator	(Definition A.2.9).
Adjoint Operator	(Definition A.2.7).
Unitary Operator	(Definition A.2.12).
σ-algebra	(Definition A.3.2).
Measurable Space	(Definition A.3.3).
Measure Space	(Definition A.3.6).
L^p space	(Definition A.3.15).
Probability Space	(Definition A.4.2).
Probability Distribution	(Definition A.4.4).
Fourier Transform	(Definition A.5.1).
Plancherel's Theorem and the Fourier Operator	(Theorem A.5.2).
Lie Groups and Manifolds	(Theorem 6.4.6)
Lie Algebras and Tangent Spaces	(Theorem 6.4.7).

1.6 Notation

$\bullet$	Placeholder for a variable in an expression.	
$\mathbb{K}$	A field, either $\mathbb{R}$ or $\mathbb{C}$.	
$\overline{\mathbb{K}}$	The closure of the field $\mathbb{K}$, e.g. $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$.	
A^c	The complement of a set A.	
$A_{\geq k}$	The set $\{a \in A : a \geq k\}$.	
$\mathbb{B}(x, \epsilon)$	A ball of radius epsilon, centered at x.	
$\mathcal{C}^n(A)$	Space of functions of nth order continuity on A.	
$f \circ g$	The function composition of functions f, g.	
$f \propto g$	f is proportional to g up to a constant c, $f(x) = cg(x)$	
$\overline{x}$	The complex conjugate of $x \in \mathbb{C}$.	
$\mathcal{J}_W(x)$	The Jacobian of W evaluated at x.	
$\operatorname{Tr}(W)$	The trace of W.	
$x \odot y$	The pointwise Hadamard product $(x_i y_i)_{i=1}^n$.	
$x^{\odot k}$	The pointwise exponentiation $(x_i^k)_{i=1}^n$.	
$\varepsilon \sim \mathbb{P}$	ε is distributed as $\mathbb{P}$.	
$\varepsilon \dot{\sim} \mathbb{P}$	ε is approximately distributed as $\mathbb{P}$.	
$x^{\odot k}$	The pointwise exponentiation $(x_i^k)_{i=1}^n$.	
$\mathcal{I}$	An index set.	(Definition A.1.1).
$\mathbb{I}_A(x)$	The indicator function.	(Definition A.1.4)
$d(x, y)$	A metric.	(Definition A.1.6)
$\|x\|$	A norm.	(Definition A.1.8).
$\langle x, y \rangle$	An inner product.	(Definition A.1.13)
$x \perp y$	Orthogonal relation.	(Definition A.1.16)
$A^\perp$	Orthogonal complement.	(Definition A.1.17)
W^*	Adjoint operator.	(Theorem A.2.5)
$\mathscr{B}$	A basis.	(Definition A.2.11)
$\mathscr{P}(A)$	The power set of A.	(Definition A.3.1)
ν-a.e.	Almost everywhere w.r.t. a measure ν.	(Definition A.3.7)
$\mathcal{M}(A)$	Space of measurable functions on A.	(Definition A.3.8)
$\mathcal{B}$	Borel σ-algebra.	(Definition A.3.10)
$\|x\|^p$	The L^p norm.	(Definition A.3.14)
$L^p(A)$	The L^p space on A.	(Definition A.3.15)
$\delta_x(A)$	The Dirac measure.	(Definition A.3.17)
$\delta(x)$	The Dirac-δ function.	(Remark A.3.18)
δ_{ij}	The Kronecker-δ.	(Remark A.3.18)
$\#(A)$	The counting measure.	(Definition A.3.19)
$\mathbb{K}^{\mathbb{N}}$	Sequence space.	(Definition A.3.21)
$f_*\nu$	A pushforward measure w.r.t. f, ν.	(Definition A.4.1)
P	A probability measure.	(Definition A.4.2)
$\mathrm{X}, \mathrm{Y}, \mathrm{Z}$	Random elements or variables.	(Definition A.4.3)
$\mathbb{P}$	A probability distribution.	(Definition A.4.4)
f_{X}	A probability density function.	(Definition A.4.5)
F_{X}	A cumulative distribution function.	(Definition A.4.6)
$\mathbb{E}[\mathrm{X}]$	The expected value of X.	(Definition A.4.9)
$\tilde{x}$	The Fourier transform of x.	(Definition A.5.1)
$\mathcal{F}$	The Fourier operator.	(Theorem A.5.2)
$U \otimes V$	Kronecker tensor product of U and V.	(Definition A.7.8)

1.7 Abbreviations, Nomenclature, and Data

SVE/SVD	Singular value expansion / decomposition.
TSVD	Truncated singular value decomposition.
MLE	Maximum likelihood estimate.
MAP	Maximum a posteriori estimate.
ELBO	Evidence lower bound.
RELU/BIRELU	Regular and bi-Lipschitz rectified linear unit activation.
ELU/BIELU	Regular and bi-Lipschitz exponential linear unit activation.
CELU/BICELU	Regular and bi-Lipschitz continuous exp. linear unit activation.
ADAM	Moment based optimization algorithm for gradient descent.
AE	Autoencoder, or general encoder-decoder network.
VAE	Variational autoencoder network.
PIE	Pseudo-invertible encoder network.
IE	Invertible encoder network.
IRE	Invertible resolvent encoder network.
CNJ	Relative condition number by Jacobian.
CNA	Relative condition number by adverserial perturbation.

Abbr.	**Name**	**Function**
RE	Mean relative error	$\frac{1}{n}\sum_i \frac{\lVert y_i - \hat{y}_i \rVert}{\lVert y_i \rVert}$
MSE	Mean squared error	$\frac{1}{n}\sum_i (y_i - \hat{y}_i)^2$
ABS	Mean absolute error	$\frac{1}{n}\sum_i \lvert y_i - \hat{y}_i \rvert$
LHC	Mean log.hyperbolic cosine	$\frac{1}{n\beta}\sum_i \ln\cosh\left(\beta(y_i - \hat{y}_i)\right)$
PSNR	Peak signal-to-noise ratio	$10\log_{10}\frac{\max(\mathcal{Y})^2}{C_{\mathrm{MSE}}(y,\hat{y})}$
SSIM	Structural Similarity	$\frac{1}{n}\sum_i \sum_w \frac{(2\mu_y^w \mu_{\hat{y}}^w + \epsilon)(2\sigma_{y\hat{y}}^w + \delta)}{(\mu_y^w + \mu_{\hat{y}}^w + \epsilon)(\sigma_y^w + \sigma_{\hat{y}}^w + \delta)}$
BCE	Mean binary cross-entropy	$\frac{1}{n}\sum_i \sum_j -y_{ij}\ln\hat{y}_{ij}$
ACC	Accuracy	$\frac{1}{n}\sum_i \mathbb{I}(\arg\max_{y_i} = \arg\max_{\hat{y}_i})$
AC5	Top-5 Accuracy	$\frac{1}{n}\sum_i \arg\max_{a,b\subset y_i,\hat{y}_i} \sum_{j=1}^{5} \mathbb{I}(a_j = b_j)$
REID	Relative error to identity matrix	$\frac{\lVert I - W \rVert_{\mathrm{Fro}}}{\lVert I \rVert_{\mathrm{Fro}}}$

Table 1.5: Overview of applied empirical risk functions and metrics.

Source	Abbr.	No.Train	No.Val.	Classes	Spat.Dim	Chan.	Scaling	Blur
CIFAR100	CIFAR	50000	10000	100	32×32	1	1:1	None
CIFAR100	CIFAR$_{\text{Avg.}}$	50000	10000	100	32×32	1	1:1	Avg. 3×3
EMNIST	EMNIST	112800	18800	47	28×28	1	1:1	None
MS-COCO	COCO	96336	4002	80	384×384	1	1:1	None
MS-COCO	COCO$_{16:1}$	96336	4002	80	96×96	1	16:1	None

Table 1.6: Overview of all datasets used in our experiments and examples. More details can be found in [Coh+17; KH09; Lin+14] as well as Section 7.1.

PART I

Theoretical Background

CHAPTER 2

Inverse Problems

In this chapter we provide an outline of the basic framework associated with inverse problems posed as integral equations, and discuss what it means for a problem to be inherently ill-posed. We discuss the mathematical tools used in the context of these problems and show their relation to spectral theory and harmonic analysis. Much of the foundations for this chapter is built on and around the work in [Gro93; Han10; HNO06; Kol02].

2.1 Ill-Posedness

Semantically, the notion of an *inverse* requires some initial structure for such a concept to be meaningful. We encounter these concepts in mathematics via relations and mappings where an inverse maps elements from the co-domain of some initial map to elements or sets of the domain. An inverse problem similarly requires an initial problem which maps some unobserved cause to an observed, measured effect. We call such an initial problem a *forward problem* or alternatively a *direct problem.* Given normed spaces $\mathcal{X}, \mathcal{Y}$ as well as an unknown probability distribution $\mathbb{P}$, a *forward problem* consists of the following components.

- An *input element* or *input state* $x \in \mathcal{X}$.
- An *output element* or *output state* $y \in \mathcal{Y}$.
- A *mapping* $\Phi : \mathcal{X} \to \mathcal{Y}$.
- A *noise component* $\varepsilon \sim \mathbb{P}$.

The general form of a *forward problem* is then given by

$$\Phi(x) + \varepsilon = y. \tag{2.1}$$

The noise component ε is the result of general uncertainty related to the output state being observed via a measurement process. We consider this noise inherent to the measurement process, and thus characteristic of the forward problem. Given a forward problem, there are two natural formulations of an *inverse problem.* The first and most common is the task of approximating an input state x using the measured output state y under affection of the noise ε, requiring assumptions on the form of the mapping Φ. In the nomenclature of this thesis, we refer to these problems as *causal* inverse problems.

The second formulation originates from approximating an inverse operator $\Phi^{-1} : \mathcal{Y} \to \mathcal{X}$ which may or may not exist given the forward problem, using observations of inputs and outputs in a *data driven approach.* These are problems which we will refer to as inverse problems of *model identification*, and problems on this form is at the center of this thesis. This approach differs from Equation (2.1) as we instead look for an optimal parametrization for an estimate of Φ^{-1} given the forward problem

$$\Phi(x;\theta) + \varepsilon = y, \tag{2.2}$$

where we let θ be an element in a parameter space Θ encode the set of relevant variables for the model, and refer to θ as a *parameter state* of Φ.

The above exposition may strike the reader as too fleeting to be of much use in a mathematical context. It seems that in most available mathematical literature on the subject of inverse problems there is a certain reluctance towards providing a formal definition – in fact there does not seem to be any acknowledged mathematical definition for inverse problems at all. Instead, the conventions and terminology for inverse problems have been directly adopted from the physical sciences. Thus, it might not be sufficiently clear why inverse problems should be any more challenging to solve than a forward problem. The fundamental difference between a forward problem and its corresponding inverse problem coincides with the pivotal classification of mathematical problems formulated by Jacques Hadamard [Had48].

Definition 2.1.1 (Well-posedness)**.** The problem $\Phi(x) = y$ is called a *well-posed* problem if

(i) *(Existence)* for all $y \in \mathcal{Y}$ there exists $x \in \mathcal{X}$ s.t. $\Phi(x) = y$,

(ii) *(Uniqueness)* for all $y \in \mathcal{Y}$ there is at most one $x \in \mathcal{X}$ s.t. $\Phi(x) = y$,

(iii) *(Stability)* for all $(x_n)_{n\in\mathbb{N}} \subset \mathcal{X}$, $\lim_{n\to\infty} \Phi(x_n) = \Phi(x)$ implies $\lim_{n\to\infty} x_n = x$.

Any problem that fails to meet any of the aforementioned criteria is consequently defined as *ill-posed.*

While this set of criteria are undeniably reasonable in order to constructively *solve* a problem, this definition exposes an adherence to a belief that any physical phenomena universally had well-posed mathematical equivalents. Today however, we know that this is not necessarily the case. Generally, most inverse problems fail to meet one or more of the aforementioned criteria, and we consider stability (Property 2.1.1.iii) to be especially challenging.

Mathematically, this is what we generally consider to be the defining characteristic of inverse problems, namely, that they are *inherently ill-posed*, while a forward problem is on the other hand assumed to be well-posed. This biformity makes them particularly challenging and coincidentally mathematically interesting.

2.2 Integral Equations

We have established that the motivation behind our understanding of inverse problems originates from physical models, thus many inverse problems are naturally phrased in the setting of continuous Euclidean space, and in this domain the natural manifestation of inverse problems are equations known as *integral equations.* The first recorded inverse problem that applied integral equations is the *Tautochrone problem* by Abel in 1823, using *Abel's integral equation* [Abe81]. This was later developed into a comprehensive theory by Fredholm [Fre03], now eponymously known as *Fredholm Theory.*

Definition 2.2.1 (Fredholm Equation, First Kind)**.** Let $\mathcal{S}, \mathcal{T}$ be open, connected subsets of $\mathbb{K}$ and let $w\colon \mathcal{S} \times \mathcal{T} \to \mathbb{K}$. Furthermore, let $x\colon \mathcal{S} \to \mathbb{K}, y\colon \mathcal{T} \to \mathbb{K}$, and assume that y is known, and x is unknown. Then

$$y(t) = \int_{\mathcal{S}} w(s,t)x(s)\ ds \tag{2.3}$$

is called a *Fredholm equation of the first kind.*

The function $w(s,t)$ is often referred to as a *kernel function.* The forward problem of a Fredholm equation can thus be stated as solving Equation (2.3) for y given x, and the inverse problem is contrarily determining x given y. In this context, it is evident that the solution to the forward problem is a matter of computation, while the inverse problem requires a more comprehensive approach.

In Definition A.2.1 we define the *linear operator.* By restricting the kernel, input and output functions to be square-integrable, a Fredholm equation can be expressed as a linear operator called the *Hilbert-Schmidt integral operator.*

Definition 2.2.2 (Hilbert-Schmidt Integral Operator)**.** Let $\mathcal{S}, \mathcal{T}$ be open, connected subsets of $\mathbb{K}$ and let $w \in L^2(\mathcal{S} \times \mathcal{T})$. Furthermore, let $x \in L^2(\mathcal{S})$ and $y \in L^2(\mathcal{T})$. The *Hilbert-Schmidt integral operator* W is then given by

$$(Wx)(t) = \int_{\mathcal{S}} w(s,t)x(s)\ ds. \tag{2.4}$$

The operator W from Definition 2.2.2 is a special case of a wider class of *Hilbert-Schmidt operators.*

Definition 2.2.3 (Hilbert-Schmidt Operator Norm)**.** Let $\mathcal{I} \subseteq \mathbb{N}$ be an index set. Given an orthonormal basis $\mathscr{B} = (v_n)_{n\in\mathcal{I}}$ (Definition A.2.11) for a Hilbert space $\mathcal{X}$, if for any bounded operator $W \in B(\mathcal{X})$ (Definition A.2.2) we have

$$\|W\|_{\mathrm{HS}} = \sum_{n\in\mathcal{I}} \|Wv_n\|^2 < \infty, \tag{2.5}$$

then W is a *Hilbert-Schmidt operator*, and we call $\|\bullet\|_{\mathrm{HS}}$ the *Hilbert-Schmidt norm.*

This class of operators form a linear subspace of $B(\mathcal{X})$ – the space of all bounded operators on $\mathcal{X}$ (Definition A.2.2). Moreover, this subspace is a Hilbert space [Con90, p.267], and Hilbert-Schmidt operators share the property of all being *compact* (Definition A.2.9 and Theorem A.7.6). The following important theorem allows us to effectively treat compact operators as infinite dimensional extensions of matrices.

Theorem 2.2.4 (Spectral Theorem for Compact Self-adjoint Operators)**.** *Let $\mathcal{X}$ be a Hilbert space and let $W : \mathcal{X} \to \mathcal{X}$ be a self-adjoint operator. Then W has an orthonormal basis of eigenvectors or eigenfunctions corresponding to real eigenvalues.*

A proof is provided in [MW99, pp.517–518]. Theorem 2.2.4 allows the application of spectral theory in functional analysis. As we will see, spectral theory is a powerful tool for solving inverse problems.

In addition to *homogeneous* integral equations introduced in Definition 2.2.1, we also have *inhomogeneous* integral equations, often referred to as equations *of the second kind.*

Definition 2.2.5 (Fredholm Equation, Second Kind)**.** Let $\mathcal{S}, \mathcal{T}$ be open, connected subsets of $\mathbb{K}$ and let $w \colon \mathcal{S} \times \mathcal{T} \to \mathbb{K}$. Furthermore, let $x \colon \mathbb{K} \to \mathbb{K}, y \colon \mathcal{T} \to \mathbb{K}$, $\lambda \in \mathbb{K} \setminus \{0\}$, and assume that y is known, and x is unknown. Then

$$y(t) = x(t) - \lambda^{-1} \int_{\mathcal{S}} w(s,t)x(s)\ ds \tag{2.6}$$

is called a *Fredholm equation of the second kind.*

The kernel $w(s,t)$ for an inhomogeneous Fredholm equation induces a Hilbert-Schmidt integral operator, and can be expressed as a linear operator equation on the form $y = (I - \lambda^{-1}W)x$. It turns out that the solution of an inhomogeneous integral equation is closely related to the spectrum of the operator W. We discuss this further in Section 2.3.

If the kernel of a homogeneous integral equation can be expressed as $w(t-s)$, the resulting equation defines a *convolution.* Convolution operators see a variety of applications in physics, statistics, and signal processing. They also happen to play an important role in modern neural network architectures.

Definition 2.2.6 (Convolution)**.** A Fredholm equation is a *convolution* if the kernel can be expressed as $w(s,t) = w(t-s)$, and we denote it by

$$(w * x)(t) = \int_{S} w(t-s)x(s)\ ds. \tag{2.7}$$

A convolution can be viewed either as a binary operator acting on two elements from distinct univariate function spaces, or an integral operator as per Definition 2.2.2. When a forward problem is given by convolution operators, the inverse problem becomes a *deconvolution problem.*

The *convolution theorem* (Theorem A.7.7) makes deconvolution problems with a known kernel more or less straightforward to solve in Fourier space. Consider the convolution given by

$$\tilde{y}(\xi) = \tilde{w}(\xi) \cdot \tilde{x}(\xi), \tag{2.8}$$

which can be rewritten as

$$\tilde{x}(\xi) = \tilde{y}(\xi)/\tilde{w}(\xi). \tag{2.9}$$

This greatly simplifies the solution, as the input signal can then simply be recovered by $\hat{x} = \mathcal{F}^{-1}(\tilde{y}/\tilde{w})$. In the following example, we will see how issues with ill-posedness generally prevent us from simply applying this method in practice.

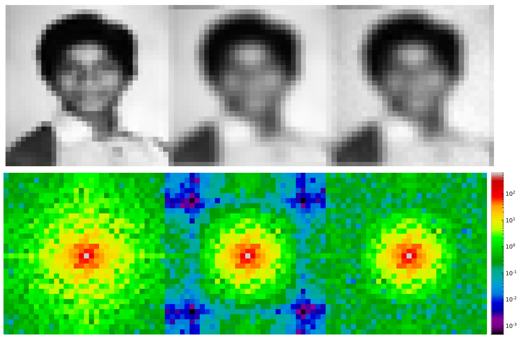

Figure 2.1: Illustration of deblurring problem. Original image (left), blurred image (middle) and noisy blurred image (right) in spatial domain (above) and frequency domain (below).

Example 2.2.7 (Deblurring in Fourier Space)**.** A common inverse problem in image processing is reconstructing an original image from a low-pass filtered image using deconvolution, called a *deblurring problem*. Deblurring is a inverse problem of model identification with a compact operator, where we assume a forward problem with a parametrized operator W_θ of the form

$$y = W_\theta x + \varepsilon$$

and we want to estimate

$$\hat{\theta} = \arg\min_\theta \|W_\theta^{-1} y - x\|.$$

For this task, we apply an average smoothing filter to images from the CIFAR [KH09] dataset (see Table 1.6 and Section 7.1 for details). For digital images, the convolution kernel is a discrete 3×3 moving average filter given by $\theta_{ij} = 1/9$ for $i, j \in \{1, 2, 3\}$. In addition, we apply a minor level of Gaussian noise $\varepsilon \sim \mathcal{N}(0, 0.01^2)$, resulting in the observed images in Figure 2.1. We compute the filter $\tilde{w}(\xi)$ by

$$\tilde{w}(\xi) = \tilde{y}(\xi)/\tilde{x}(\xi)$$

which we then apply to the image point wise in the Fourier domain using Equation (2.9). The results can be observed in Figure 2.2.

To evaluate the reconstructions, we apply the commonly used peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) metrics (Table 1.5). The the reconstruction of the blurred image without noise (left) achieves a PSNR of 141.95 and a SSIM of 1.0, indicating a more or less perfect reconstruction of the original image. However, the noisy image only achieves a PSNR of 5.64 and a SSIM of 0.27. We conclude that even with full knowledge of the kernel used in the convolution, an almost imperceptible amount of added noise will result in poor reconstruction with this approach. This demonstrates the inherent ill-posedness of deconvolution problems.

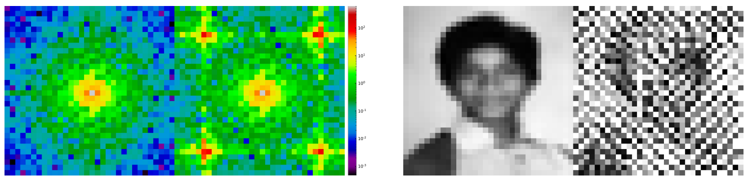

Figure 2.2: Reconstructed images in Fourier domain (left) and spatial domain (right) from deblurring problem. The deconvolution with the applied blur kernel results in a barely recognizable reconstruction for the noisy image.

If we now consider convolutions with square integrable kernel functions in terms of Theorem 2.2.4, i.e. operators of the form

$$(Wx)(t) = \int_S w(t-s)x(s)\ ds$$

and derive the eigenfunctions, we get an interesting result.

Proposition 2.2.8. *The eigenfunctions of the convolution operator are given by*

$$v(t) = e^{2\pi it\cdot\xi}.$$

Proof. Let $v(s) = e^{2\pi is\cdot\xi}$. A consequence of Theorem A.7.7 is that convolution is commutative. We thus have $w * v = v * w$, so we can write

$$\begin{aligned}(Wv)(t) &= \int_S w(s)e^{2\pi i(t-s)\cdot\xi}\ ds \\ &= e^{2\pi it\cdot\xi}\int_S w(s)e^{-2\pi is\cdot\xi}\ ds \\ &= \tilde{w}(\xi)v(t).\end{aligned}$$

Evaluating $\tilde{w}(\xi)$ for a fixed frequency ξ yields a scalar, which we denote as $\tilde{w}(\xi) = \lambda_\xi \in \mathbb{C}$. We consequently have $\lambda_\xi v(t)$, so the eigenfunctions of W are on the form $e^{2\pi i\xi s}$. ■

The family of functions $v(t)$ are the *complex exponentials* which form an orthonormal basis for Fourier space corresponding with specific frequencies, and these functions are fundamental in the field of harmonic analysis. We can interpret the result of Proposition 2.2.8 as the Fourier transform acting as a diagonalization of convolution operators. In the next section, we discuss how this result can be generalized to solve inverse problems for other integral equations with kernels yielding compact operators.

2.3 Expansion Methods and Spectral Theory

In this section, we connect the generalized Fourier series and singular value expansion, and show how these tools are applied to solve homogeneous linear integral equations. We also discuss some fundamental spectral theory on Banach spaces, and show how these are useful for solving inhomogeneous integral equations.

Recall that the most common application of the Fourier transform (Definition A.5.1) is *Fourier series expansion*, the premise for which can be derived from the properties of the sequence

$$\mathscr{B}_{\mathcal{F}} = \left(v_n : v_n(t) = \frac{e^{int}}{\sqrt{2\pi}} \right)_{n\in\mathbb{Z}}.$$

Given the canonical L^2-norm and inner product (Lemma A.1.14 and Definition A.1.13), we have $\|v_n\| = 1$ with $\langle v_n, v_m \rangle = \delta_{nm}$ denoting the Kronecker delta (Remark A.3.18). Then $\mathscr{B}_{\mathcal{F}}$ is an orthonormal basis (Definition A.2.11) for $L^2([-\pi,\pi])$, or equivalently an orthonormal basis for $L^2(\mathbb{R}/2\pi\mathbb{Z})$, the space of functions with periodicity 2π which are square integrable over their period.

Definition 2.3.1 (Fourier Coefficients)**.** Let $x \in L^2(\mathbb{R}/2\pi\mathbb{Z})$. Let $(c_n)_{n\in\mathbb{Z}}$ be a sequence with elements given by

$$c_n = \frac{\langle x, e^{-int} \rangle}{\sqrt{2\pi}}. \tag{2.10}$$

Then $(c_n)_{n\in\mathbb{Z}}$ are the *Fourier coefficients* of x.

Definition 2.3.2 (Fourier Series)**.** Let $x \in L^2(\mathbb{R}/2\pi\mathbb{Z})$ with Fourier coefficients $(c_n)_{n\in\mathbb{Z}}$. Then

$$x(t) = \frac{1}{\sqrt{2\pi}} \sum_{n\in\mathbb{Z}} c_n e^{int} \tag{2.11}$$

is called the *Fourier series expansion* of x.

Thus, we can decompose the reconstruction of a Fourier series expansion into two steps; a *analysis* step where we determine the Fourier coefficients, and a *synthesis* step where we reconstruct the function by the Fourier series. Appropriate scaling and shifting can be applied to functions on domains outside the periodic interval $[-\pi,\pi]$.

Proposition 2.2.8 tells us that the Fourier transform acts as a diagonalization of convolution operators, in the sense that it decomposes the operator into an orthogonal basis of Fourier coefficients. The series expansion of Fourier series can be further generalized to alternate bases. We call these expansion methods a *generalized Fourier series.*

Definition 2.3.3 (Generalized Fourier Series)**.** Let $\langle \bullet, \bullet \rangle_*$ be some inner product with an induced norm $\|\bullet\|_*$. Let $\mathcal{S}$ be a connected subset of $\mathbb{K}$ and let $\mathscr{B} = (v_n)_{n\in\mathbb{N}}$ be an orthogonal basis for $L^2(\mathcal{S})$. For $x \in L^2(\mathcal{S})$, let $(c_n)_{n\in\mathbb{Z}}$ be a sequence with elements given by

$$c_n = \frac{\langle x, v_n \rangle_*}{\|v_n\|_*^2}. \tag{2.12}$$

Then we call $(c_n)_{n\in\mathbb{Z}}$ the *generalized Fourier coefficients*, and we call

$$x = \sum_{n\in\mathbb{Z}} c_n v_n \tag{2.13}$$

the *generalized Fourier series expansion* of x.

The generalized Fourier transform is useful precisely because it allows for a more flexible choice of basis and inner product space.

From the spectral theorem (Theorem 2.2.4) we know that we can construct a countable orthonormal basis for self-adjoint operators, which can be extended to compact operators by considering the self-adjoint W^*W. This allows to diagonalize a compact operator by generalizing the spectral decomposition in terms of *singular values.*

Definition 2.3.4 (Singular Values)**.** Let $\mathcal{X}, \mathcal{Y}$ be Hilbert spaces, and let $W\colon \mathcal{X} \to \mathcal{Y}$ be a compact operator with adjoint operator $W^*\colon \mathcal{Y} \to \mathcal{X}$. Then the square roots of the eigenvalues $\varsigma_j = \sqrt{\lambda_j} \in \mathbb{R}_{\geq 0}$ for $j \in \mathbb{N}$ of the self-adjoint operator $W^*W\colon \mathcal{X} \to \mathcal{X}$ are called *singular values* of W.

Note that the singular values are necessarily nonnegative as W^*W is a positive operator (Definition A.2.8) by $\langle x, W^*Wx\rangle = \langle Wx, Wx\rangle \geq 0$, so its eigenvalues must be nonnegative. Singular values are commonly encountered in linear algebra via singular value decomposition of a matrix. It turns out that the singular value decomposition for finite rank operators can be generalized by the *singular value expansion* in functional analysis.

Corollary 2.3.5 (Singular Value Expansion)**.** *Let $\mathcal{X}, \mathcal{Y}$ be Hilbert spaces, let $W\colon \mathcal{X} \to \mathcal{Y}$ be a compact linear operator with adjoint operator $W^*\colon \mathcal{Y} \to \mathcal{X}$. Let $\varsigma_1 \geq \varsigma_2 \geq \cdots > 0$ be an ordered sequence of singular values of W. Then there exist orthonormal systems $(v_n)_{n\in\mathbb{N}} \subset \mathcal{Y}$ and $(u_n)_{n\in\mathbb{N}} \subset \mathcal{X}$ where*

$$Wv_n = \varsigma_n u_n \quad \text{and} \quad W^* u_n = \varsigma_n v_n \tag{2.14}$$

for all $n \in \mathbb{N}$. The system $(\varsigma_n, u_n, v_n)_{n\in\mathbb{N}}$ is called a singular system *for W. The* singular value expansion (SVE) *of x, y w.r.t. W is given by*

$$x(s) = \sum_{n\in\mathbb{N}} \langle v_n, x\rangle v_n(s) \tag{2.15}$$

$$y(t) = \sum_{n\in\mathbb{N}} \langle u_n, y\rangle u_n(t) \tag{2.16}$$

Proof. From Theorem 2.2.4 we know that $W^*W : \mathcal{X} \to \mathcal{X}$ has an orthonormal basis of eigenfunctions $(v_n)_{n\in\mathbb{N}}$ corresponding to eigenvalues $(\varsigma_n^2)_{n\in\mathbb{N}}$. By definition, we then have

$$W^*Wv_n = \varsigma_n^2 v_n. \tag{2.17}$$

Let $u_n = \frac{1}{\varsigma_n} W v_n$. Then $W^*u_n = \varsigma_n v_n$. Then $(u_n)_{n\in\mathbb{N}}$ is orthonormal as

$$\begin{aligned}
\langle u_m, u_n\rangle &= \frac{1}{\varsigma_m\varsigma_n}\langle Wv_m, Wv_n\rangle && (2.18)\\
&= \frac{1}{\varsigma_m\varsigma_n}\langle W^*Wv_m, v_n\rangle && (2.19)\\
&= \frac{\varsigma_m^2}{\varsigma_m\varsigma_n}\langle v_m, v_n\rangle && (2.20)\\
&= \begin{cases} 1, & \text{if } n = m;\\ 0, & \text{otherwise.}\end{cases} && (2.21)
\end{aligned}$$

Similarly, $WW^* : \mathcal{Y} \to \mathcal{Y}$ has an orthonormal basis of eigenfunctions $(u_n)_{n\in\mathbb{N}}$ corresponding to the same set of eigenvalues. Then Equations (2.15) and (2.16) follow from Definition A.2.11. ■

As such, the singular value expansion can be considered a generalized Fourier transform with respect to a compact operator W. Equation (2.14) is sometimes referred to as the *fundamental relation*, and when combined with Equation (2.16), we can construct an explicit solution for the linear system

$$\begin{aligned}
Wx &= y && (2.22)\\
&= \sum_{n\in\mathbb{N}} \langle u_n, y\rangle u_n && (2.23)\\
&= W \sum_{n\in\mathbb{N}} \frac{\langle u_n, y\rangle}{\varsigma_n} v_n && (2.24)\\
\hat{x} &= \sum_{n\in\mathbb{N}} \frac{\langle u_n, y\rangle}{\varsigma_n} v_n. && (2.25)
\end{aligned}$$

Equation (2.25) is in fact an infinite dimensional representation of a least squares approximation $\hat{x} = \arg\min_{x \in \mathcal{X}} \|y - Wx\|_2^2$ via the *pseudo-inverse* of W.

Theorem 2.3.6 (Moore-Penrose Pseudoinverse)**.** *Let $\mathcal{X}, \mathcal{Y}$ be Hilbert spaces, and let $W \colon \mathcal{X} \to \mathcal{Y}$ be a compact linear operator. Then there exists a unique $W^\dagger : \mathcal{Y} \to \mathcal{X}$ such that*

(i) $WW^\dagger W = W$,

(ii) $W^\dagger W W^\dagger = W^\dagger$,

(iii) $(WW^\dagger)^* = WW^\dagger$,

(iv) $(W^\dagger W)^* = W^\dagger W$,

called the Moore-Penrose generalized inverse *or* pseudo-inverse *of W.*

An exposition of the full theorem along with proofs for finite rank operators is provided in the original paper by Sir Roger Penrose [Pen55], while an extension to general operators is given in [Beu65].

For a causal inverse problem where the system Φ is fully given by a compact operator W, we can construct a least squares solution by $W^\dagger y = \hat{x}$. If W has full rank we can construct this solution algebraically by taking the right-inverse $W^\dagger = (W^* W)^{-1} W^*$. The pseudo-inverse of finite rank operators can be expressed by singular value decomposition (SVD) [GBC16, pp.45–46] yielding

$$W^\dagger = V \Sigma^\dagger U^* \tag{2.26}$$

which is exactly the form of Equation (2.25), so the solution approximated by singular value expansion is a least squares approximation. We have yet to address the effect of the random element ε (Definition A.4.3). In fact, the singular values directly affect the sensitivity of the approximate solutions.

Proposition 2.3.7 (Variance of SVE solutions)**.** *Let $y = Wx + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. The variance of an* SVE *reconstruction $\hat{x}$ (Equation* (2.25)*) is then given by*

$$\operatorname{Var}[\hat{x}] = \sum_{n \in \mathbb{N}} \left(\frac{\sigma}{\varsigma_n} \right)^2 . \tag{2.27}$$

Proof. Let $\varepsilon(\omega, t) \sim \mathcal{N}(0, \sigma^2)$ be the random element of $y(t)$. Then

$$\begin{aligned}
\hat{x} &= \sum_{n\in\mathbb{N}} \frac{\langle u_n, y - \varepsilon\rangle}{\varsigma_n} v_n \\
&= \sum_{n\in\mathbb{N}} \frac{\langle u_n, y\rangle - \langle u_n, \varepsilon\rangle}{\varsigma_n} v_n.
\end{aligned}$$

Excluding the non-stochastic elements, the variance can be written as

$$\begin{aligned}
\mathrm{Var}[\hat{x}] &= \mathrm{Var}\left[\sum_{n\in\mathbb{N}} \frac{\langle u_n, \varepsilon\rangle}{\varsigma_n}\right] \\
&= \left(\frac{1}{\varsigma_n}\right)^2 \mathrm{Var}\left[\sum_{n\in\mathbb{N}} \langle u_n, \varepsilon\rangle\right],
\end{aligned}$$

where we interpret $\langle u_n, \varepsilon\rangle$ as a stochastic functional $\langle u_n, \varepsilon_\omega\rangle\ |_{\omega=\omega'}$ for some possible realization ω' over $\mathcal{T}$. Given $\mathbb{E}[\varepsilon_\omega] = 0$ we have $\mathrm{Var}[\varepsilon_\omega] = \mathbb{E}[\varepsilon_\omega^2]$, so by linearity of expectation we express the variance component wise as

$$\begin{aligned}
\mathrm{Var}[\langle u_n, \varepsilon\rangle] &= \mathrm{Var}\left[\int_{\mathcal{T}} u_n(t)\varepsilon_\omega(t)\ dt\right] && (2.28)\\
&= \int_\Omega \left[\int_{\mathcal{T}} u_n(t)\varepsilon_\omega(t)\ dt\right]^2 f_\varepsilon(\omega)\ dP(\omega) && (2.29)\\
&= \int_{\mathcal{T}'}\int_{\mathcal{T}} \left[\int_\Omega \varepsilon_\omega(t)\varepsilon_\omega(t') f_\varepsilon(\omega)\ dP(\omega)\right] u_n(t)u_n(t')\ dt\ dt' && (2.30)\\
&= \int_{\mathcal{T}'}\int_{\mathcal{T}} \mathrm{Cov}[\varepsilon_\omega(t), \varepsilon_\omega(t')] u_n(t)u_n(t')\ dt\ dt' && (2.31)\\
&= \sigma^2 \int_{\mathcal{T}'}\int_{\mathcal{T}} \delta(t-t') u_n(t)u_n(t')\ dt\ dt' && (2.32)\\
&= \sigma^2 \|u_n\|^2 && (2.33)\\
&= \sigma^2, && (2.34)
\end{aligned}$$

where f_ε is the pdf. of ε (Definition A.4.5) and δ is the Dirac delta function (Remark A.3.18). This yields the final expression in Equation (2.27). ∎

The variance of the reconstructions can consequently become arbitrarily magnified for singular values of very low magnitude. For the purposes of solving inverse problems, if we consider $\varsigma_n \to 0$ as $n \to \infty$ then clearly $\hat{x} \to \infty$. By the Riemann-Lesbegue lemma (Lemma A.7.3) and Proposition 2.2.8 where we showed that the eigenfunctions of convolution operators are complex exponentials, we know this is exactly the case for deconvolution. This shows that inverse problems of deconvolution are inherently unstable, and thus ill-posed.

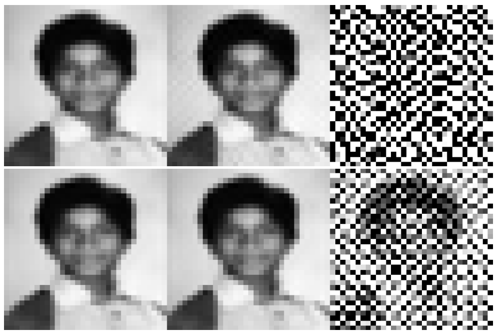

Figure 2.3: Reconstructions with separable filters (above) and linear least squares (below). The reconstructions from the blurred images (middle) are practically identical to the original images (left), while the reconstructions from images affected by noise (right) are more or less unrecognizable.

Example 2.3.8 (Deblurring with Least Squares)**.** To practically illustrate the instability of least squares solutions, we revisit our example from Example 2.2.7. Recall that a single channel digital image can be represented as a matrix of pixel values $X \in \mathbb{R}^{m\times n}$ where m, n are the height and width of the image, respectively. An image blurred with a *separable filter* [HNO06] is the result of applying operators $W_c \in \mathbb{R}^{n\times n}, W_r \in \mathbb{R}^{m\times m}$, yielding a blurred image

$$Y = W_c X W_r^*. \tag{2.35}$$

In our example – a moving average 3×3 kernel – the filter is not only separable, but also symmetric, i.e., $W_c = W_r^* = W_s$. In this case, a closed form estimate for W_s can be constructed by

$$\hat{W}_s = X^{-\frac{1}{2}}(X^{\frac{1}{2}} Y X^{\frac{1}{2}})^{\frac{1}{2}} X^{-\frac{1}{2}}. \tag{2.36}$$

A more common method is to pose the problem as an ordinary linear least squares problem. A separable filter can be rewritten as a linear operator by

$$\begin{aligned} Y &= W_c X W_r^* && (2.37)\\ \mathrm{vec}(Y) &= (W_r \otimes W_c)\mathrm{vec}(X) && (2.38)\\ y &= Wx && (2.39) \end{aligned}$$

where $\mathrm{vec}(X)$ is the lexicographical vectorization of the matrix X, and $\bullet \otimes \bullet$ is the *Kronecker tensor product* (Definition A.7.8). Estimating the linear operator W has the additional benefit of no assumptions of separability, and pseudoinverse can be applied for reconstruction.

The reconstructed images from both methods can be seen in Figure 2.3. The estimate computed by Equation (2.36) produces a reconstruction of the noisy image with a PSNR of -9.2 and a SSIM of 0.04, while the linear least squares method fares marginally better with a PSNR of 4.11 and a SSIM of 0.21, however both methods perform worse than deconvolution in Fourier space. Clearly, the inverse operators estimated using least squares methods suffer from considerable instability.

Thus far, we have focused on methods for approximating homogeneous equations with compact linear operators using least squares approximations and series expansions. In Section 2.2 we also introduced the inhomogeneous equations (Definition 2.2.5) on the form

$$y = (I - \lambda^{-1} W)x. \tag{2.40}$$

These equations are central in the resolvent formalism first applied by Fredholm [Fre03] where he introduced the Fredholm alternative. a set of methods in functional analysis which applies techniques from complex analysis to spectral theory on operators in Banach spaces.

Suppose we want to algebraically extend the properties of univariate calculus to applications on operators $W \in B(\mathcal{X})$. The canonical example is the polynomial extension where we consider

$$p_n(z) = \sum_{0 \leq i \leq n} c_i z^i, \tag{2.41}$$

for some $z \in \mathbb{K}$. The idea is that we can simply replace z with an operator W and retain properties of the polynomial ring. From here, it is natural to consider a functional extension to the set of holomorphic functions $f \in \mathrm{Hol}(\mathbb{C})$. These functions have a unique representation for the power series

$$f(x) = p_\infty(x) = \sum_{k \in \mathbb{Z}_{\geq 0}} c_i x^k, \tag{2.42}$$

as any $f \in \mathrm{Hol}(\mathbb{C})$ is necessarily absolutely convergent. As the rules for convergence are the same for operators and numbers with respect to their norms, the properties are retained for bounded operators, justifying

$$p_\infty(W) = \sum_{k \in \mathbb{Z}_{\geq 0}} c_i W^k \tag{2.43}$$

where we use the convention $W^0 = I$. The so called *holomorphic functional calculus* is a homomorphism (Definition A.6.4) between $\mathbb{C}$ and local neighbourhoods of the spectra of functional Banach spaces. The homomorphism requires a well-defined multiplicative operator on power series of operators in the form of the *Cauchy product* (Definition A.7.9). From here, it is necessary to invoke concepts from spectral theory.

Definition 2.3.9 (Resolvent Sets and Operators)**.** Let $W \in B(\mathcal{X})$ over $\mathbb{K}$. Let $\rho(W)$ be a set given by

$$\rho(W) = \{\lambda \in \mathbb{K} : (\lambda I - W) \text{ is bijective}\}. \tag{2.44}$$

Furthermore let $R(\bullet, W) : \rho(W) \to B(\mathcal{X})$ be the mapping given by

$$R(\lambda, W) = (\lambda I - W)^{-1}. \tag{2.45}$$

Then $\rho(W)$ is called the *resolvent set* of W, and the operators $R(\lambda, W)$ are called *resolvent operators* of W.

Note that the formulations of $(\lambda I - W)$ and $(I - \lambda^{-1} W)$ are equivalent in terms of determining the spectrum. By definition, the resolvent set $\rho(W)$ is the open set compliment of the closed spectrum $\varsigma(W)$, as $R(\lambda, W)$ is not defined if $\lambda \in \varsigma(W)$. This allows a method of defining invertable operators using the holomorphic functional extension of geometric series.

Theorem 2.3.10 (Neumann Series)**.** *Let $\mathcal{X}$ be Banach and let $W \in B(\mathcal{X})$ with $\|W\| < 1$. Then $I - W$ is invertible and we have*

$$(I - W)^{-1} = \sum_{k \in \mathbb{Z}_{\geq 0}} W^k. \tag{2.46}$$

Hence $W_n = \sum_{k \leq n} W^k \to (I - W)^{-1}$ as $n \to \infty$, and we call W_n a Neumann series *for the operator W.*

Proof. Firstly, we show that the operator norm is submultiplicative. We have

$$\|UV\| = \max_{x \neq 0} \frac{\|UVx\|}{\|x\|} \tag{2.47}$$

$$= \max_{Vx \neq 0} \frac{\|UVx\|}{\|Vx\|} \frac{\|Vx\|}{\|x\|} \tag{2.48}$$

$$\leq \max_{x \neq 0} \frac{\|Ux\|}{\|x\|} \max_{x \neq 0} \frac{\|Vx\|}{\|x\|} \tag{2.49}$$

$$= \|U\|\|V\|. \tag{2.50}$$

Then $\sum_{k \leq n} \|W^k\| \leq \sum_{k \leq n} \|W\|^k$. As $\|W\| < 1$ the Neumann series is geometric, and thus also Cauchy with respect to the operator norm. It follows that if $\mathcal{X}$ is Banach, it is necessarily complete, so $\lim_{n \to \infty} W_n \in B(\mathcal{X})$ and we have

$$(I - W) \sum_{k \in \mathbb{Z}_{\geq 0}} W^k = \sum_{k \in \mathbb{Z}_{\geq 0}} W^k - W^{k+1} \tag{2.51}$$

$$= I + \sum_{k \in \mathbb{Z}_{\geq 1}} W^k - W^k \tag{2.52}$$

$$= I. \tag{2.53}$$

As the argument is symmetrical for right multiplication, we necessarily have $\sum_{k \in \mathbb{Z}_{\geq 0}} W^k = (I - W)^{-1}$ as the theorem states. ■

Corollary 2.3.11 (Extension of Neumann Series)**.** *Let $\mathcal{X}$ be a Banach space and let $U, V \in B(\mathcal{X})$. Let U be invertible and let $\|V\| < 1/\|U^{-1}\|$. Then $U + V$ is invertible with*

$$(U + V)^{-1} = (I + U^{-1} V)^{-1} U^{-1} \tag{2.54}$$

$$= U^{-1} (I + V U^{-1})^{-1}. \tag{2.55}$$

Proof. From $U + V = U(I + U^{-1} V)$ we have that $\|U^{-1} V\| < 1$, so from Theorem 2.3.10 and the fact that U is invertible, the result trivially holds. ■

Theorem 2.3.10 and Corollary 2.3.11 make it clear that the resolvent operators $R(\lambda, W)$ are extremely useful for solving inverse problems. In fact, this was the precise motivation for the introduction of Fredholm theory and spectral theory in general. For Fredholm integral equations of the second kind, solutions can be approximated by constructing a *Liouville-Neumann* series.

Corollary 2.3.12 (Liouville-Neumann Series)**.** *Let $(I - \lambda^{-1}W)x = y$ be an inhomogeneous integral equation with $\|W\| < |\lambda|$. Then the solution is uniquely given by*

$$x = \sum_{k\in\mathbb{Z}_{\geq 0}} \frac{W^k y}{\lambda^k}, \tag{2.56}$$

called the Liouville-Neumann series*.*

Corollary 2.3.12 follows as a direct consequence of Theorem 2.3.10 and Corollary 2.3.11, demonstrating how inverse problems can be approached by applying resolvent operators in series expansion.

Lastly, we would like to demonstrate an often overlooked property of the spectra of linear operators – the fact that the spectrum $\varsigma(W)$ is *upper semi-continuous*.

Theorem 2.3.13 (Upper Semi-Continuity of Linear Operator Spectra)**.** *Let $V \in B(\mathcal{X})$ and let $O \in \mathbb{K}$ be an open set with $\varsigma(V) \subseteq O$. Then there exists a $\delta > 0$ such that $\varsigma(U) \subseteq O$ for every $U \in B(\mathcal{X})$ with $\|U - V\| < \delta$.*

Proof. Firstly, we note that since $\lambda \in O^c$ we necessarily have $\lambda \in \rho(V)$, thus $R(\lambda, V) \in B(\mathcal{X})$ is well defined. Using Theorem 2.3.10, we assume that $\|W\| = \|(\lambda I - V)^{-1}(\lambda I - U)\| < 1$ and consider

$$\begin{aligned}
\|I - (\lambda I - V)^{-1}(\lambda I - U)\| &= \|R(\lambda, V)\big[(\lambda I - V) - (\lambda I - U)\big]\| && (2.57)\\
&= \|R(\lambda, V)\big[\lambda I - V - \lambda I + U\big]\| && (2.58)\\
&\leq \|R(\lambda, V)\|\|U - V\| < 1. && (2.59)
\end{aligned}$$

Then setting $\|R(\lambda, V)\|^{-1} = \delta$ results in $(\lambda I - U)$ being invertible by Corollary 2.3.11, so $\lambda \in \rho(U)$. Thus $\varsigma(U) \subseteq O$ and $\|U - V\| < \delta$ as we wanted. ∎

Theorem 2.3.13 ensures that the spectrum of an operator is well-behaved and has semi-continuous properties. However, this semi-continuity is not restricted from below, which could be a source of instability symptomatic of inverse problems. In any case, resolvent operators are a powerful method for solving inverse problems.

2.4 Discretization and Projection

Thus far, we have only considered inverse problems in the context of integral operators (Section 2.2) acting on function spaces defined over continuous domains. It is clear that solving these equations numerically on a computer requires some discrete approximation. To this end, we apply *discretization* to function spaces in some appropriate finite dimensional vector space via *projection operators.*

Definition 2.4.1 (Projection operator)**.** Let $\mathcal{Z}$ be a normed space over $\mathbb{K}$, and let $A \subset \mathcal{Z}$ be a closed subspace. Let $Q : \mathcal{Z} \to \mathcal{Z}$ be an operator such that

(i) $Qz \in A$ for all $z \in \mathcal{Z}$,

(ii) $Qz = z$ for all $z \in A$.

Then Q is a projection operator onto the subspace A.

Property 2.4.1.ii is also referred to as the *idempotent property*, implying that $Qz = Q^2 z$. In some literature this is considered the defining property of projection operators. On the other hand, Property 2.4.1.i implies that the range of Q is closed if Q is continuous. A continuous projection can therefore be decomposed into two closed, orthogonal subspaces. This is especially important for projection operators in Hilbert spaces.

Theorem 2.4.2 (Projection Theorem)**.** *Let $\mathcal{Z}$ be a Hilbert space, and let $A \subset \mathcal{Z}$ be a closed nontrivial subspace. Then there exists a unique element $z' \in A$ such that*

$$\|z - z'\| = \inf_{a \in A} \|z - a\| \tag{2.60}$$

if and only if $(z - z') \in A^\perp$.

A proof for Theorem 2.4.2 is provided in [Rud87, pp.79–80]. If we now let $(A_n)_{n \in \mathbb{N}}$ be a nested sequence of monotonically increasing subspaces, i.e. $A_n \subset A_{n+1} \subset \cdots \subset \mathcal{Z}$ with associated projection operators Q_n such that

$$\lim_{n \to \infty} \|Q_n z - z\| = 0. \tag{2.61}$$

Then we have that

$$\lim_{n \to \infty} \inf_{z' \in A_n} \|z - z'\| = 0, \tag{2.62}$$

so $\mathcal{Z}$ can be approximated by a sequence of subspaces as *multi-resolution analysis.* Selecting a finite subset of such a sequence by choice or truncation yields a discretization. Furthermore by making any such selection, we can encode a priori information about the desired solution of a given inverse problem.

A relevant example of how such a process is applied can be made by way of the generalized Fourier series expansion (Definition 2.3.3). By selecting a subset of coefficients using a finite index set $\mathcal{I} \subset \mathbb{Z}$, the sequence of coefficients $(c_i)_{i \in \mathcal{I}} \subset (c_n)_{n \in \mathbb{Z}}$ can be selected to omit undesired frequencies. Thus, a priori knowledge of the frequency band of the noise variable ε can be applied to effectively *filter* unwanted frequencies. Such methods are a form of *spectral*

K	PSNR	PSNR (Noise)	SSIM	SSIM (Noise)
750	34.876	21.462	0.995	0.887
500	30.035	26.974	0.983	0.962
250	26.273	26.097	0.962	0.959

Table 2.1: Results of deblurring example with TSVD

.

filtering, which dampens certain spectra of the full system to avoid reconstruction errors that stem from the inherent noise. By Riemann-Lebesgue (Lemma A.7.3) the high frequency coefficients tend to zero, so the truncation includes mostly high-frequency components, thus truncation generally has a smoothing effect, acting similarly to low-pass filters.

In practice, truncation is not necessarily the most effective method for removing noise, and much of the work in the field of signal processing is concerned with the design of effective filters for such purposes. However, in the case of a singular system $(\varsigma_n, v_n, u_n)_{n\in\mathbb{N}}$ the values are in descending order, and a simple truncation of the singular values at some finite K can prove very effective.

Example 2.4.3 (Deblurring with TSVD)**.** In Proposition 2.3.7 we showed that the variance of solutions computed by the singular value expansion is inversely proportional to the singular values, showing how a truncation of low magnitude values can be effective for reducing the reconstruction error caused by the noise component ε. To demonstrate this in practice, we can apply a truncation of the singular values in the computed pseudo-inverse in our least squares problem from Example 2.3.8. The solutions are computed by

$$\hat{x}_K = \sum_{1\leq n\leq K} \frac{\langle u_n, y\rangle}{\varsigma_n} v_n \tag{2.63}$$

$$= V\Sigma_K^{\dagger} U^* y \tag{2.64}$$

$$= W_K^{\dagger} y, \tag{2.65}$$

and we call $\hat{x}_K$ the *truncated singular value* solution (TSVD) of order K. Truncation of expansion methods and generalized Fourier expansions can effectively be considered a discretization of the original space. We compute the TSVD solutions for the images in our deblurring problem. The images are of dimension 32×32 so $\dim(y) = 1024$. We perform truncation for $K = (750, 500, 250)$. The results can be observed in Table 2.1 and Figure 2.4. The effect of the truncation is evident in the noisy images on the far right, however for the nonnoisy images in the middle, we observe a decrease in quality when compared to the original images on the far left. This demonstrates a trade-off inherent in these methods; we trade accuracy for better robustness to errors in our reconstructions. As such, the TSVD demonstrate how selective discretization methods can act as a form of *regularization*. We will discuss regularization methods in more depth in Section 3.5.

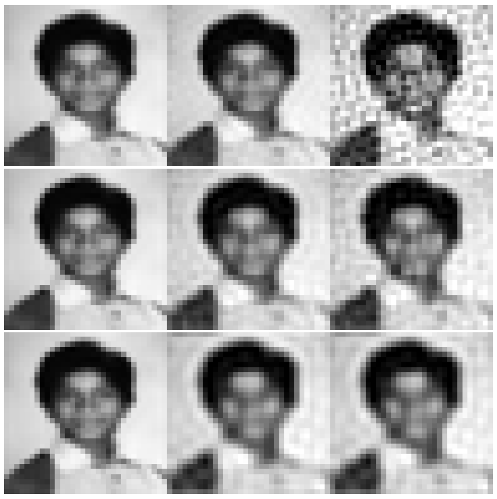

Figure 2.4: Reconstructions with TSVD for $K_1 = 750$ (above), $K_2 = 500$ (middle) and $K_3 = 250$ (below). Decreasing K leads to better robustness for the noisy image (right) but a degradation in reconstruction quality for the non-noisy images (middle).

While truncated series expansion can be effective, we often rely on basic general methods for discrete signal representation on a computer. A common discretization method for a connected subset $\mathcal{S} \subset \mathbb{K}^d$ is obtained by deciding on some strictly increasing set of abscissae $(s_i)_{i\in\mathcal{I}} \subset \mathcal{S}$ where $s_1 < s_2 < \cdots < s_m$. A natural way of selecting these abscissae is by applying a fixed increment in the domain given by $\mathit{\Delta}_s$, such that

$$s_i + \mathit{\Delta}_s = s_{i+1}.$$

Note that for a multidimensional x, this imposed ordering can be selected to be lexicographical without loss of generality, as in the case with vectorization $x = \mathrm{vec}(X)$. Repeating this process over the domain T with M, we can rewrite the integral equation in matrix form

$$y_i = y(t_i) = \sum_{j=1}^{m} \mathit{\Delta}_s w(s_j, t_i) x(s_j) \tag{2.66}$$

$$= W_i x, \;\; W \in \mathbb{R}^{m\times n}, x \in \mathbb{R}^n \tag{2.67}$$

which corresponds to Riemann sums over rectangular partitions. Equivalently, we can instead interpret this as a projection from the general L^p space onto a dense subspace of simple functions, similar to Lebesgue integration. This interpretation has the additional benefit of corresponding to how we introduced discretization as a projection into finite dimensional subspaces. No matter how this method is interpreted, it can readily be extended to more advanced numerical integration methods, as more refined methods for choosing abscissae may be desirable for certain problems. The fixed increment Δ_s is usually chosen to yield a desired dimensionality or *resolution* of the discretized signal.

CHAPTER 3

Modelling and Learning

In Chapter 2 we showed an how the inherent uncertainty of inverse problems affects the reconstructed solutions. Given their inherent stochastic nature, the application of statistical methods seems perspicaciously appropriate. In this chapter we introduce statistical modelling and learning theory to demonstrate how these tools are directly applicable to construct data-driven solutions for inverse problems. Most of the underlying probability theory is based on [Bil95; Cox04; Kal02; Sch95]. Fundamental definitions and theorems are outlined in Appendix A.4.

3.1 Statistical Modelling and Probability

Courses in applied statistical modelling commonly circumvent rigorous definitions grounded in measure theory and probability theory in favor of a more practical approach. However, for a holistic understanding of the core concepts in statistical modelling, we find it useful to outline a rudimentary set of definitions in these fields. We begin by delineating a set of useful definitions for dealing with observations and data.

Proposition 3.1.1 (Observable Space)**.** *Let $(\Omega, \mathcal{E}, P)$ be a probability space, and let $\mathrm{X} : \Omega \to \mathcal{X}$ be a random element. Let $\mathcal{B}$ be the Borel σ-algebra on $\mathcal{X}$. Then $(\mathcal{X}, \mathcal{B}, \mathbb{P})$ is a probability space under the image of* X*, and we call $(\mathcal{X}, \mathcal{B})$ an* observable space *under* X*.*

Proof. This follows directly from Definitions A.3.2, A.4.2 and A.4.4. $\mathcal{B}$ is a Borel σ-algebra on $\mathcal{X}$, and as such, $(\mathcal{X}, \mathcal{B})$ is a measurable space. $\mathbb{P}$ is a probability measure as the pre-image is $\mathrm{X}^{-1}(\mathcal{X}) = \Omega$ (Definition A.1.3), i.e. $\mathbb{P}(\mathcal{X}) = \mathrm{X}_* P(\mathcal{X}) = P(\Omega) = 1$. Thus $(\mathcal{X}, \mathcal{B}, \mathbb{P})$ is a probability space. ∎

Note that we exclude the probability distribution $\mathbb{P}$ from the definition of an observable space, as it is not directly observable, and thus unknown to us by simple observation. As such, we consider an observable space as induced from; and not itself a probability space. A given number of available elements from an observable space forms a *dataset*.

Definition 3.1.2 (Dataset)**.** Let $(\mathcal{X}, \mathcal{B})$ be an observable space under $\mathrm{X} : \Omega \to \mathcal{X}$, and let $\mathcal{I}$ be a finite index set. Then any sequence $\mathcal{D} = (x_i \in \mathcal{X})_{i \in \mathcal{I}}$ is called a *dataset.*

Generally, we consider $\mathcal{X} \subseteq \mathbb{R}^d$ and, unless explicitly specified, do not consider the order imposed by the index set $\mathcal{I}$ to carry any particular meaning other than that $\mathcal{D}$ is countable. Note that even if X is a d-dimensional random variable, this does not mean its elements are identically distributed. For our purposes, we take the liberty of considering a multivariate random variable as actualizations of d distinct random variables, and the multivariate convention is simply a useful conceptualization. This is a highly simplified approach, but is useful for the purposes of exposition for providing a top-down overview of statistical modelling.

Proposition 3.1.1 and Definition 3.1.2 will prove useful in constructing the necessary definitions for what we mean by statistical modelling. In addition to these, we will require definitions for what we mean when we talk about families of probability distributions.

Definition 3.1.3 (Parametric Family of Probability Distributions)**.** Let $(\mathcal{X}, \mathcal{B})$ be an observable space. Let Θ be a parameter space, and let $\mathcal{P}$ be a set given by $\mathcal{P} = \{\mathbb{P}_\theta : \mathbb{P}_\theta \text{ is a probability measure on } (\mathcal{X}, \mathcal{B})\}_{\theta \in \Theta}$. Then $\mathcal{P}$ is a *family of parametrized probability distributions.*

If there exists a bijective mapping $\theta \mapsto \mathbb{P}_\theta$ we say that $\mathcal{P}$ is *identifiable.* The parametrization $\theta \mapsto \mathbb{P}_\theta$ is not strictly required for $\mathcal{P}$ to act as a family of distributions. It is also worth mentioning that the bijectivity of $\theta \mapsto \mathbb{P}_\theta$ ensures that for every $\theta \neq \theta'$ we have $\mathbb{P}_\theta \neq \mathbb{P}_{\theta'}$, such that each probability distribution is identifiable by its parametrization. Definition 3.1.3 can perhaps best be illustrated with the following canonical example.

Example 3.1.4 (Family of Independent Multivariate Gaussians)**.** Consider the parametric family of probability distributions given by $\mathcal{P} = \{\mathcal{N}_\theta : \theta = (\mu, \sigma^2 I)\}$ with $\mu \in \mathbb{R}^d, \sigma^2 \in \mathbb{R}_{>0}$. This family yields a parameter space $\Theta = \mathbb{R}^d \times \mathbb{R}_{>0}$. As each parametrization yields a unique probability distribution, the mapping is necessarily injective. Furthermore, as the parameter space spans $\mathbb{R}^d \times \mathbb{R}_{>0}$, the mapping is surjective, and thus also bijective. We can conclude that $\mathcal{P}$ is an identifiable parametric family of probability distributions.

With this clarification, we are now in possession of all necessary components to provide a rigorous definition for what we mean by a *statistical model* [McC02].

Definition 3.1.5 (Statistical Model)**.** Let $(\mathcal{X}, \mathcal{B})$ be an observable space, and let $\mathcal{P}$ be a family of probability distributions on $(\mathcal{X}, \mathcal{B})$. Then the pair $(\mathcal{X}, \mathcal{P})$ is a *statistical model.*

Given Definition 3.1.5, it becomes clear how statistical modelling is concerned with defining a suitable $\mathcal{P}$ with respect to an observable space $(\mathcal{X}, \mathcal{B})$. As previously mentioned, an observable space is only available to us via a fixed number of observations from a dataset $\mathcal{D}$, which allows us to infer a distribution based on assumptions and observations of the underlying structure. The most basic statistical models can be constructed by constricting $\mathcal{P}$ to only contain the *empirical distribution.*

Example 3.1.6 (Empirical Distribution)**.** Let $(\mathcal{X}, \mathcal{B})$ be an observable space and let $\mathcal{D}$ be a dataset on $\mathcal{X}$ with $\#(\mathcal{D}) = n$, with $\#$ being the counting measure (Definition A.3.19). Let $\hat{\mathbb{P}} : \mathcal{B} \to [0, 1]$ be such that for all $B \in \mathcal{B}$ we have

$$\hat{\mathbb{P}}(B) = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}(B). \tag{3.1}$$

where δ_x is the Dirac measure (Definition A.3.17). Then $\hat{\mathbb{P}}$ is an *empirical probability distribution* on $\mathcal{X}$ given the dataset $\mathcal{D}$.

While the set $\hat{\mathcal{P}} = \{\hat{\mathbb{P}}\}$ is a singleton set, only containing an empirical distribution defined by the observations in $\mathcal{D}$, we still consider $\hat{\mathcal{P}}$ to be a 'family', and thus regard $(\mathcal{X}, \hat{\mathcal{P}})$ to fulfill the requirements of Definition 3.1.5. The fundamental lack of assumptions on $\mathcal{X}$ means the empirical distribution is inherently unbiased, but does rely on large amounts of observations to provide a accurate estimates of the true distribution.

It is worth clarifying the difference between *parametric* and *non-parametric* statistical models; a classical discernment of statistical models with respect to the dimensionality of the induced parameter space.

Definition 3.1.7 (Parametric Models)**.** Let $(\mathcal{X}, \mathcal{P})$ be a statistical model. We say a model is

(i) *parametric* if $\dim(\Theta) < \infty$,

(ii) *nonparametric* otherwise.

A common way of interpreting Property 3.1.7.ii is to consider the underlying model structure of a nonparametric model as not being clearly defined in advance, so that the dimensionality of the parameter space can grow arbitrarily. This provides greater flexibility and fewer assumptions as we discussed in the case of the empirical distribution; generally considered to be a nonparametric model. On the other hand, parametric models allow us to fit a model using a fixed number of parameters, which are then estimated to yield the most appropriate distribution given the observed data. The most common approach for parameter estimation is via optimization of a *likelihood function* [GH13, p.9].

Definition 3.1.8 (Likelihood Function)**.** Let $(\mathcal{X}, \mathcal{P})$ be a statistical model parametrized by $\theta \in \Theta$. Let $\mathcal{D}$ be a dataset on $\mathcal{X}$. Then $\mathscr{L} : \Theta \times \mathcal{X} \to [0, 1]$ given by

$$\mathscr{L}(\theta; \mathcal{D}) \propto \mathbb{P}_\theta(\mathcal{D}) \tag{3.2}$$

is called the *likelihood* of the parameter θ.

Very often in practical likelihood estimation, we assume that $x_i \in \mathcal{D}$ are *independent and identically distributed* (iid.) observations for which – in the case of continuous probability distributions – we arrive at

$$\mathscr{L}(\theta; \mathcal{D}) \propto \prod_{i \in \mathcal{I}} f_{\mathrm{X}}(x_i; \theta). \tag{3.3}$$

For computational convenience, we often apply a logarithmic transformation to the likelihood function. This allows us to change the product to a sum, yielding the *log-likelihood function* given by

$$\log \mathscr{L}(\theta; \mathcal{D}) \propto \sum_{i \in \mathcal{I}} \log f_{\mathrm{X}}(x_i; \theta), \tag{3.4}$$

which is commonly more easy to deal with in an optimization setting. A *maximum likelihood estimate* is the point in the parameter space Θ given by

$$\theta_{\mathrm{MLE}} = \arg\max_{\theta} \mathscr{L}(\theta; \mathcal{D}), \tag{3.5}$$

and by the strict monotonicity of the logarithm, this is equivalent to

$$\theta_{\mathrm{MLE}} = \arg\max_{\theta} \log \mathscr{L}(\theta; \mathcal{D}). \tag{3.6}$$

Often the goal of statistical modelling is to establish relations between variables by investigating conditional structures between observations - which requires a clearer definition of *conditional probability.* The concept of conditional probability is commonly introduced by way of *conditional expectation.*

Definition 3.1.9 (Conditional Expectation)**.** Let $(\Omega, \mathcal{E}, P)$ be a probability space, and let $\mathrm{Y} : \Omega \to \mathbb{R}^d, \mathrm{Y} \in L^1(\Omega, \mathcal{E}, P)$. Let $\mathcal{F} \subseteq \mathcal{E}$ be a sub σ-algebra. The *conditional expectation* of Y is a $\mathcal{F}$-measurable random variable such that

$$\int_{F \in \mathcal{F}} \mathbb{E}[\mathrm{Y} \mid \mathcal{F}] \; dP = \int_{F \in \mathcal{F}} \mathrm{Y} \; dP = \mathbb{E}[\mathbb{I}_{F \in \mathcal{F}} \mathrm{Y}]. \tag{3.7}$$

While the conditional expectation and conditional probability are assumed well known to the reader, Definition 3.1.9 can at first glance seem unfamiliar. To relate this to practice, let $\mathrm{X} : \Omega \to \mathbb{R}^p$ be a random variable, and let $\mathcal{F} = \{\mathrm{X}^{-1}(B) : B \in \mathcal{B}(\mathbb{R}^p)\}$. Then $\mathcal{F}$ is the σ-algebra generated by X, which is indeed a sub σ-algebra of $\mathcal{E}$, and we write $\mathbb{E}[\mathrm{Y} \mid \mathcal{F}] = \mathbb{E}[\mathrm{Y} \mid \mathrm{X}]$. This should align with practical probabilistic intuition and can be used to formalize the definition of regular conditional probability as a parametrized family of probability distributions.

Definition 3.1.10. Let $(\Omega, \mathcal{E}, P)$ be a probability space, and let $\mathcal{F} \subset \mathcal{E}$ be a sub σ-algebra. The conditional probability of $E \in \mathcal{E}$ with respect to $\mathcal{F}$ is given by

$$P(E \mid \mathcal{F}) = \mathbb{E}[\mathbb{I}_E \mid \mathcal{F}]. \tag{3.8}$$

We note that both Definitions 3.1.9 and 3.1.10 require certain theorems to ensure uniqueness and existence of conditional probability, which interested readers can verify in [Bil95; Sch95].

Conditional probability is a very powerful tool which establishes a foundation for the probabilistic framework known as *Bayesian inference*. For instance, the maximum likelihood estimator can be more naturally formulated in a conditional probability setting by considering

$$\theta_{\text{MLE}} = \arg\max_{\theta} \mathscr{L}(\theta; \mathcal{D}) \propto \mathbb{P}(\mathcal{D} \mid \theta). \tag{3.9}$$

This formulation implies that Θ has an associated probability distribution, and the parameters are random variables $\vartheta : \Omega \to \Theta$, inducing a observable space $(\Theta, \mathcal{T})$. In this context, the probability distribution $\mathbb{P}_\vartheta$ is called the *prior distribution* and the probability distribution $\mathbb{P}_{\mathrm{X}}$ is called the *marginal distribution*. The interpretation of parameters as random variables is natural in Bayesian inference, and is formalized in *Bayes' theorem*.

Theorem 3.1.11 (Bayes)**.** *Let* $(\Omega, \mathcal{E}, P)$ *be a probability space. Let* $\mathrm{X} : \Omega \to \mathcal{X}$ *such that* $(\mathcal{X}, \mathcal{B}_{\mathcal{X}})$ *is an associated observable space. Let* $\vartheta : \Omega \to \Theta$ *be a random variable such that* $(\Theta, \mathcal{B}_\Theta)$ *is an associated observable space. Then the pdf. of the conditional probability distribution* $\mathbb{P}_{\vartheta \mid \mathrm{X}}$ *is given by*

$$f_{\vartheta \mid \mathrm{X}}(\theta \mid x) = \frac{f_{\mathrm{X} \mid \vartheta}(x \mid \theta) f_\vartheta(\theta)}{f_{\mathrm{X}}(x)} \tag{3.10}$$

and we call the distribution $\mathbb{P}_{\vartheta \mid \mathrm{X}}$ *the* posterior distribution*.*

Our exposition of Theorem 3.1.11 is a simplification from [Sch95, Theorem 1.31], which formalizes the theorem in the probability theoretic setting. When applying Bayesian inference, we usually start with some initial belief on the parameters for $\mathcal{P}$ encoded via the prior distribution. This belief is subsequently updated by the evidence – the observed data we have at hand – to form a posterior distribution. The advantage of the Bayesian framework is that we effectively model all relevant uncertainties, while effectively encoding a priori information in the form of domain knowledge or particular knowledge of the task at hand in the choice of prior distribution. The disadvantage with this approach is that the choice of prior distributions can significantly alter the outcome, thus the objective validity of the model is diminished. Furthermore, the marginal distribution $\mathbb{P}_{\mathrm{X}}$ is generally intractable, and often only available through approximation.

While Bayesian inference provides the modeller with estimated probability distributions over all parameters, point estimates can be obtained by constructing a *maximum a posteriori* estimate (MAP) given by

$$\theta_{\text{MAP}} = \arg\max_{\theta} f_{\vartheta \mid \mathrm{X}}(\theta \mid x) \propto f_{\mathrm{X} \mid \vartheta}(x \mid \theta) f_\vartheta(\theta). \tag{3.11}$$

3.2 Statistical Learning and Hypothesis Spaces

In the previous section, we introduced the classical definition of statistical modelling and provided a general outline of a more comprehensive probability theory. This section is intended to generalize the concept of statistical models into a more comprehensive theory of *statistical learning theory* [HTF09], which lays the foundation for machine learning and provides us with a taxonomy of model categories which is useful for further discussion. In a statistical learning context, we are often interested in estimating an unknown data generation mechanism by

$$y = \Phi(x;\theta) + \varepsilon, \tag{3.12}$$

where $(x, y)_{i\in\mathcal{I}}$ are observations from some observable space $(\mathcal{X} \times \mathcal{Y}, \mathcal{B})$ over the product space $\mathcal{X} \times \mathcal{Y}$ generated by X, Y called *independent* and *dependent* variables. The most comprehensive approach is to construct a classical statistical model for Equation (3.12) by reformulating the problem as

$$\mathrm{Y} \mid \mathrm{X} \sim \mathbb{P}_\theta, \tag{3.13}$$

which can either be estimated via classical or Bayesian approaches. Very often, the purpose of the model is to provide a point estimate for the dependent variable Y conditioned on the independent variable X via the conditional expectation $\mathbb{E}_\theta[\mathrm{Y} \mid \mathrm{X} = x]$. When this model explicitly relies on estimating probability distributions $\mathbb{P}_\theta$ – as is the case for Definition 3.1.5 – we refer to the model as *probabilistic*.

As mentioned in the previous section, Bayesian inference and modelling [Gel+14] provides the most comprehensive methodology for constructing probabilistic models by pursuing an exhaustive determination of all uncertainties related to the model, including probability distributions on the parameter space Θ. This approach yields highly robust, inferable models, but can become prohibitively costly in terms of computational resources – especially for high-dimensional data. High-dimensional data also exacerbates the importance of a choice of prior distributions, which – as previously mentioned – is often influential on the resulting posterior distribution.

The alternative to a probabilistic approach is to construct *deterministic models*, sometimes referred to as *distribution-free models*. A deterministic model will to some extent disregard elements of underlying probability distributions and sacrifice measures of uncertainty for computational efficiency or better point estimates. The lack of uncertainty measures and inference capabilities in distribution-free prediction models have been highlighted by some as the conceptual divide between statistics and machine learning [BAK18; Fri98] and raises a pertinent question; is there a place for discriminative models in statistics, without explicit probabilistic modelling? One could always make a case for the fact that some non-parametric prediction models – e.g. *k-nearest neighbours* (kNN) [FH89] or *decision trees* [Bre+84] – make few to no assumptions on the underlying probability distribution and instead constructs discrete partitions of the solution space. Given that this does not provide an explicit probability distribution, such a partitioning does not align with the classic definition of a statistical model, given in Definition 3.1.5. However, the partitioning of an observable space is intimately related to the construction of histograms and empirical probability distributions (Example 3.1.6) – which is undeniably central to classical statistical theory. With this in mind, there is certainly some merit in expanding the definition of statistical models to include distribution-free models – particularly in the context of machine learning tasks. Indeed, many of the recent groundbreaking approaches to data driven methods for mathematical modelling has come from the paradigm of machine learning, which has instigated the interest of many classically trained statisticians and has served to broaden both fields. One motivation for the generalized approach of statistical learning methods is to accommodate for both deterministic and probabilistic models defined on observable spaces. To unify both approaches, we generalize the definition of a statistical model by considering relevant *hypothesis spaces* [Blo10; HW21].

Definition 3.2.1 (Parametric Hypothesis Space)**.** Let $(\mathcal{X}, \mathcal{B})$ be an observable space and let $\mathcal{Z}$ be some space of interest. Let Θ be a parameter space and let $\mathscr{H}$ be a set given by $\mathscr{H} = \{h_\theta : \mathcal{X} \to \mathcal{Z}\}_{\theta \in \Theta}$. Then $\mathscr{H}$ is a *parametrized hypothesis space* on $\mathcal{X}$.

Similar to Definition 3.1.5 the parametrization is not a strict requirement, and if $\theta \mapsto h_\theta$ is bijective, we say $\mathscr{H}$ is identifiable. Note that any mention of the space $\mathcal{Z}$ is left intentionally ambiguous. This allows sufficient flexibility for the hypothesis space to allow for either point estimates (i.e., $\mathcal{Z} \subseteq \mathcal{X}$), density estimates (i.e., $h_\theta \approx f_{\mathrm{X}}$), or other estimates of interest. In other words, considering a hypothesis space $\mathscr{H}$ instead of a family of probability distributions $\mathcal{P}$ allows more flexibility, allowing us to extend Definition 3.1.5 to the more general class of *learning models*.

Definition 3.2.2 (Learning Model)**.** Let $(\mathcal{X}, \mathcal{B})$ be an observable space, and let $\mathscr{H}$ be a hypothesis space over $\mathcal{X}$. Then $(\mathcal{X}, \mathscr{H})$ is called a *learning model*.

Definition 3.2.2 allows us to consider probabilistic and deterministic models in a more unified context. Note that we do not require the observable space to be generated by a dependent and independent variable explicitly. Instead we call models that are estimated with a dependent variable for each independent variable for *supervised learning* models. Conversely, models that do not use a dependent variable are called *unsupervised learning* models. A *semi-supervised* learning model can be considered a hybrid model, where we have a limited number of observations featuring dependent variables.

The distinction between deterministic models and probabilistic models can be extended to either two or three distinct classes, depending on the literature. According to [Jeb03, Chapter 2], we differentiate between fully *discriminative* models, *conditional models*, and *generative models*. We will use this taxonomy as a baseline for the nomenclature of this thesis.

Definition 3.2.3 (Discriminative Learning Model)**.** Let $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$ be a parametrized learning model. Let $h_\theta : \mathcal{X} \to \mathcal{Y}$ be an approximation of the conditional expectation $\mathbb{E}[\mathrm{Y} \mid \mathrm{X} = x]$. Then $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$ is called a *discriminative learning model*.

A discriminative model provides non-probabilistic statistical point estimates, which can be made without any explicit probability distribution. If we utilize a probabilistic approach with explicit dependence on a parametrized family of probability distributions, we instead call such models *conditional*.

Definition 3.2.4 (Conditional Learning Model)**.** Let $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$ be a parametrized learning model. Let h_θ be an approximation of the conditional density of the random variable $\mathrm{X} \mid \mathrm{Y}$ such that $f_{\mathrm{X}|\mathrm{Y}} \approx h_\theta$. Then $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$ is called a *conditional learning model*.

We note that by the alternative nomenclature of [NJ02] both models outlined in Definitions 3.2.3 and 3.2.4 are termed discriminative. The goal of discriminative modelling is commonly to derive a high quality point estimate for the expectation of the dependent random variable Y conditioned on the observed random variable X.

A more comprehensive probabilistic approach can be applied to construct models that better approximate the underlying data generation process and can be used to effectively *synthesize* data via sampling via a estimated joint probability distribution. We call such models *generative*.

Definition 3.2.5 (Generative Model)**.** Let $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$ be a parametrized learning model. Let h_θ be an approximation of the joint distribution over the random variables X, Y such that $f_{\mathrm{XY}} \approx h_\theta$. Then $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$ is called a *generative model*.

Examples of generative models include Bayesian networks [Ben08], mixture models [Gel+14, pp.519–543] and hidden Markov models [GH13, p.124].

Approaching the modelling process generatively has some rather unique benefits. By modelling the joint probability distribution, we effectively have a more complete probabilistic model, which can be designed to be more robust and flexible to either outliers, missing, or corrupt data [GDB19; Yin18]. This makes it especially stronger for online learning tasks, and allows the model to be effectively retrained to include more classes. Such an approach is often not possible with discriminative models, and generally prohibitive with conditional models.

Additionally, generative models can offer much in terms of inference, especially if they are constructed as *probabilistic graphical models* [DFO20, Section 8.5] which are designed to model the conditional dependencies of all variables in a problem. These dependencies require an explicit definition in return for inferable relationships for multilevel hierarchical conditional structures in the data. Furthermore, the synthesis aspect of generative modelling allows a modeller to sample or interpolate to generate data, which can be directly applied in inference tasks.

All of this makes generative modelling an attractive prospect, but as [Jeb03, Chapter 2] points out, this flexibility comes at a cost. There is generally a lack of model accuracy when applied directly to discriminative tasks. In certain predictive applications, this gap in predictive power between discriminative and generative modelling is too wide for effective deployment. In addition, as many generative models are modelled with probabilistic methods, they become prohibitively expensive in terms of computational resources, while typically requiring a much higher number of observations to approximate the underlying probability distribution adequately.

3.3 Modelling Linear Inverse Problems

In this section, we apply the concepts from previous sections and chapters to outline the general modelling process in the context of inverse problems. From Section 2.3 we recall the singular value expansion of Corollary 2.3.5 as a method for solving inverse problems given integral operators and mentioned that compact operators and Hilbert-Schmidt operators can be considered extensions of matrices. Furthermore, in Section 2.4, we described how continuous problems can be solved numerically by projection onto a discrete space, and in Sections 3.1 and 3.2 we discussed the applications of statistical modelling and machine learning. We now want to explicitly connect the concept of integral equations in function spaces to finite-dimensional cases to construct computational learning models. In this chapter, we will mainly consider *linear* inverse problems.

Definition 3.3.1 (Linear Problems)**.** Let $\mathcal{X}, \mathcal{Y}$ be Hilbert spaces, and let $\Phi : \mathcal{X} \to \mathcal{Y}$ be given by a linear operator (Definition A.2.1). Then $\Phi(x) + \varepsilon = y$ is a *linear forward problem*, and we subsequently refer to the associated inverse problem as a *linear inverse problem*.

From Section 2.2 we know that a Fredholm integral equation induces a compact linear Hilbert-Schmidt operator. We can thus consider a continuous signal as a discretized operator via an appropriate projection to construct a linear discrete problem. We briefly discussed the basic discretization process in the end of Section 2.4, and we now demonstrate how this can be applied in practice.

Example 3.3.2 (Discretization of Linear Forward Problem)**.** Assume we have an integral equation given a random variable $\mathrm{X} \in L^2(\Omega \times \mathbb{R})$ given by

$$\mathrm{Y}(\omega, t) = \int_{\mathcal{S}} w(s, t)\mathrm{X}(\omega, s)\; ds. \tag{3.14}$$

Now assume some discretization process over $\mathcal{S}, \mathcal{T}$ such that

$$\begin{aligned} \mathrm{Y}_k(\omega) &= \mathrm{Y}(\omega, t_k) && (3.15)\\ &= \sum_{j=1}^{S} w(s_j, t_k)\mathrm{X}(\omega, s_j) && (3.16)\\ &= \sum_{j=1}^{S} W_{jk}\mathrm{X}_j(\omega). && (3.17) \end{aligned}$$

This gives us two multivariate random variables, $\mathrm{Y}^{(T)} : \Omega \to \mathcal{Y} \subseteq \mathbb{R}^T$, and $\mathrm{X}^{(S)} : \Omega \to \mathcal{X} \subseteq \mathbb{R}^S$. Consider a parametrized hypothesis space $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$ where $h_\theta(x) = W_\theta x$ for a parametrized linear operator, as well as a dataset $\mathcal{D} = \Big(\big(\mathrm{X}^{(S)}(\omega_i), \mathrm{Y}^{(T)}(\omega_i)\big) : \omega_i \in \Omega\Big)_{i \in \mathcal{I}}$. The problem can be formulated as

$$\begin{aligned} y &= h_\theta(x) + \varepsilon = W_\theta x + \varepsilon && (3.18)\\ &= \Phi(x, \theta) + \varepsilon, && (3.19) \end{aligned}$$

where $\theta \mapsto h_\theta$ is a discrete parametrization of the learning model for the unknown functional operator W associated with Φ.

Remark 3.3.3. Applied statistics often adapt the convention of considering independent variables in a *design matrix* X, which is considered an operator for a discrete causal linear inverse problem

$$\begin{aligned} y &= X\theta + \varepsilon && (3.20)\\ &= \Phi(\theta) + \varepsilon, && (3.21) \end{aligned}$$

as opposed to a model identification problem of the same form

$$\begin{aligned} y_i &= W_\theta x_i + \varepsilon && (3.22)\\ &= \Phi(x;\theta) + \varepsilon. && (3.23) \end{aligned}$$

The causal formulation poses regression as an inverse problem where we are interested in determining the parametrization θ which generates the outputs y_i. On the other hand, the model identification problem poses this as a forward problem, and the implication is that we are interested in first determining a parametrization for an invertible operator $\theta \mapsto W_\theta$ which we want to use to reconstruct x_i given y_i. Both formulations are inverse problems in their own right, however we will primarily focus on model identification problems in this thesis.

Note that we will apply lowercase notation for both independent and dependent observed variables to differentiate between operators and observations. Primarily, we generalize an operator $W_\theta : \mathcal{X} \to \mathcal{Y}$ over an N dimensional dataset by considering $x \in \mathcal{X} \subseteq \mathbb{R}^{S\times N}, y \in \mathcal{Y} \subseteq \mathbb{R}^{T\times N}$. Then $W_\theta \in \mathbb{R}^{T\times S}$ such that $y = W_\theta x + \varepsilon$ for some $\varepsilon \in \mathbb{R}^{T\times N}$. Subsequently, when we index the data by x_i or y_i, we refer to the index $i \in \mathcal{I}$ of the dataset $\mathcal{D}$, and not necessarily the row of the matrices. We should also mention that least-squares problems generally include a bias term for translation away from the origin, thus the problems are often given in the form $y = W_{\theta_{1:d}} x + \theta_0 + \varepsilon$. Without loss of generality, we adopt the convention of assuming an *augmented affine transformation* (Definition A.7.18).

In the model identification problem, the first objective is to estimate W_θ using observations from $\mathcal{D}$. This problem is ill-posed, as the uncertainty of Y captured via the noise component ε implies $y \notin \text{range}(W)$ almost surely. Any such problem can be solved by finding a *least squares approximation* over the parameter space Θ, given by

$$\hat{\theta} = \arg\min_\theta \|y - W_\theta x\|_2^2, \tag{3.24}$$

which corresponds to a *linear regression problem.* At first glance, this method seems comparatively different from likelihood estimation (Definition 3.1.8), but for a centered symmetric homoscedastic noise component these two methods coincide. We demonstrate this by a familiar example.

Example 3.3.4 (Estimation in Linear Regression)**.** Let $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$ be a parametrized hypothesis space over a dataset $\mathcal{D}$ with n observations. Assume $y = h_\theta(x) + \varepsilon \in \mathbb{R}^n$, and assume $\varepsilon^\intercal \sim \mathcal{N}(0, \sigma^2 I)$. Let $h_\theta(x) = \theta^\intercal x \in \mathbb{R}^{1\times S}$ so we have $y^\intercal \sim \mathcal{N}(\theta^\intercal x, \sigma^2 I)$. Then we can estimate the operator using the likelihood function

$$\hat{\theta} = \arg\max_\theta \mathscr{L}(\theta \mid \mathcal{D}) = \arg\max_\theta f_{\mathrm{Y}}(\theta^\intercal x, \sigma^2 I) \tag{3.25}$$

$$= \arg\max_\theta \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \langle \theta, x_i\rangle)^2}{2\sigma^2}\right) \tag{3.26}$$

$$= \arg\max_\theta \sum_{i=1}^n -2\log s - \log(2\pi) - \frac{(y_i - \langle \theta, x_i\rangle)^2}{2\sigma^2}. \tag{3.27}$$

This corresponds to minimizing the least-squares objective

$$\arg\max_\theta \mathscr{L}(\theta \mid \mathcal{D}) = \arg\max_\theta \|y - \theta^\intercal x\|_2^2 \tag{3.28}$$

$$= \arg\max_\theta \|y - W_\theta x\|_2^2 \tag{3.29}$$

Conceptually, Example 3.3.4 exposes a setting where likelihood estimation and the distribution-free approach using least squares are equivalent. Both methods minimize a *risk functional* [Jeb03; Vap92] which is quantified by an *objective function*, alternatively called a *loss* or *cost* function which quantifies some notion of discrepancy or induced metric (Definition A.1.6) between a desired response and the response of a given learning model. The objective function can be selected specifically for a given learning task, but in a supervised learning task for some hypothesis space $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$, we often work with an objective function $C : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ generally selected to be convex (Definition A.7.19) and satisfying

$$y = \arg\min_{h_\theta(x)} C\big(h_\theta(x), y\big), \tag{3.30}$$

that is to say, C has a minima at $h_\theta(x) = y$. Evaluating a risk functional generally requires integration over the joint density f_{XY}, which we do not have access to. In practice, we instead approximate the joint distribution using an empirical distribution $\hat{\mathbb{P}}_{\mathrm{X,Y}}$ (see Example 3.1.6), and we instead refer to this as minimizing an *empirical risk function.*

In Section 2.3 we discussed how least squares solutions can be expressed by way the pseudoinverse (Theorem 2.3.6) equivalently to the reconstruction by singular value expansion from Equation (2.25). This algebraic formulation provides us with another useful interpretation. By the projection theorem (Theorem 2.4.2), we can consider the operator

$$H = WW^\dagger \tag{3.31}$$

called the *hat operator* or *hat matrix*, which is an orthogonal projection onto the column space of W - sometimes called the *model space* - such that

$$(y - Hy) = (I - H)y = Ry \tag{3.32}$$

where Ry are the residuals, orthogonal to the approximation Hy. Then for a linear model W we have $\varepsilon \in \mathrm{range}(R)$, i.e. the noise is an element in the range of R, and $\operatorname{range}(R) \perp \operatorname{range}(H)$. An equivalent probabilistic definition can be constructed via the conditional expectation (Definition 3.1.9).

Observation 3.3.5. *Let $(\Omega, \mathcal{E}, P)$ be an probability space, and let $\mathcal{F} \subset \mathcal{E}$ be a sub σ-algebra. Then $\mathbb{E}[\bullet \mid \mathcal{F}]$ is a projection $L^2(\Omega, \mathcal{E}, P) \mapsto L^2(\Omega, \mathcal{F}, P)$.*

Proof. For $\mathbb{E}[\bullet \mid \mathcal{F}]$ to be a projection we need to show that Property 2.4.1.i (closed) and Property 2.4.1.ii (idempotent) holds. Firstly, let $\mathrm{Y} \in L^2(\Omega, \mathcal{E}, P)$. Then clearly $\mathbb{E}[\mathrm{Y} \mid \mathcal{F}] \in L^2(\Omega, \mathcal{F}, P)$, so Property 2.4.1.i must hold. Furthermore, for $\mathrm{Y} \in L^2(\Omega, \mathcal{F}, P)$ we have $\mathbb{E}[\mathrm{Y} \mid \mathcal{F}] = \mathrm{Y}$, thus Property 2.4.1.ii also holds, and so $\mathbb{E}[\bullet \mid \mathcal{F}]$ is a projection. ■

Observation 3.3.5 in conjunction with Theorem 2.4.2 thus tells us that given $\mathcal{F}$ being the sub σ-algebra generated by X, the conditional expectation of $\mathrm{Y} \mid \mathrm{X}$ is such that $(\mathrm{Y} - \mathbb{E}[\mathrm{Y} \mid \mathrm{X}]) \perp L^2(\Omega, \mathcal{F}, P)$. This is exactly the same as we saw in the linear case, where $\operatorname{range}(R) \perp \operatorname{range}(H)$. Intuitively, the conditional expectation minimizes the prediction error given the sub σ-algebra generated by X w.r.t. the norm induced by $L^2(\Omega, \mathcal{E}, P)$, and the hat matrix H equivalently minimizes the prediction error in a linear least squares model. In the case of Example 3.3.4, these methods coincide.

3.4 Modelling Non-Linear Inverse Problems

Thus far, we have focused on inverse problems whose forward problem can be expressed by a linear operator, which we refer to as linear inverse problems (Definition 3.3.1). Any inverse problem which does not meet this requirement is called a *nonlinear inverse problem.* In this section we will briefly outline how we can construct models for dealing with such problems. The most common approach for non-linear modelling is by *data transformation* [BK19, pp.122–124] via a *linear predictor.*

Definition 3.4.1 (Linear Predictor)**.** Let $(\mathcal{X} \times \mathcal{Y}, \mathcal{B})$ be an observable space, and let $\mathcal{H}$ be a Hilbert space with $\dim(\mathcal{H}) = \dim(\mathcal{Y})$. Let $W_\theta \colon \mathcal{X} \to \mathcal{H}$ be a compact linear operator with parametrization $\theta \mapsto W_\theta$. Then

$$\eta_i = W_\theta x_i \tag{3.33}$$

is called a *linear predictor.*

The purpose of the linear predictor is to establish a linear map between the input data points $x \in \mathcal{X}$ and the linear predictor $\eta \in \mathcal{H}$ with a goal of finding some explicit mapping $\gamma : \mathcal{H} \to \mathcal{Y}$. In the context of *generalized linear models* [NW72], these functions are referred to as *link functions.*

Definition 3.4.2 (Link Function)**.** Let $\eta_i : \mathcal{X} \to \mathcal{H}$ be a linear predictor and let $g : \mathcal{Y} \to \mathcal{H}$ be a bijective map. Let $\mu_i = \mathbb{E}[\mathrm{Y}_i \mid \mathrm{X}_i = x_i]$. Then

$$\eta_i = g(\mu_i) \tag{3.34}$$

is a *link function.*

Generalized linear models are constructed explicitly for modelling random variables in the *exponential distribution family* - a very common parametrized family of probability distributions (Definition 3.1.3). The most straightforward example of a generalized linear model is by the identity link $g = g^{-1} = \mathrm{id}$, which amounts to an ordinary linear regression model (Example 3.3.4). A more interesting link function is applied in *logistic regression* which uses the *logit-link* function given by

$$\mathrm{logit}(p_i) = \log\left(\frac{p_i}{1-p_i}\right), \tag{3.35}$$

which yields the log-odds of a probability $p_i = P(\mathrm{Y}_i = 1)$. As the link function is bijective, the map $\mathrm{logit} : [0,1] \to \mathbb{R}$ is bijective, and the inverse map $\mathrm{logit}^{-1} : \mathbb{R} \to [0,1]$ is given by

$$\mathrm{logit}^{-1}(\eta_i) = \mathrm{logistic}(\eta_i) = \frac{1}{1+\exp(-\eta_i)}. \tag{3.36}$$

The inverse of the logit-link function is called the *logistic* function, sometimes referred to as the *sigmoid* function, especially in the context of neural networks [GBC16, p.67]. Logistic regression is the canonical method for discrete classification tasks and is assumed known to the reader. For more details on generalized linear models we refer to [Agr15].

When constructing a generalized linear model, we utilize a linear predictor and some appropriate nonlinear transformations. This idea can be generalized to model observations that are not necessarily restricted to the exponential distribution family by some inferred function $\gamma : \mathcal{H} \to \mathcal{Y}$ allowing us to apply linear estimation methods by considering a parametrized hypothesis space $\mathscr{H}$ of models $h_\theta = \gamma(\eta_i)$. We call such methods *linearized transformations*, . A more general method for linearization of features which act non-linearly on $\mathcal{Y}$ is *linear basis expansion*, which appends features with some linear combination of functions on $\mathcal{X}$.

Definition 3.4.3 (Linear Basis Expansion)**.** Let $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$ be a parametrized learning model, and let $h_\theta : \mathcal{X} \to \mathcal{Y}$ be given by

$$h_\theta(x_i) = \sum_j \theta_j \gamma_j(x_i). \tag{3.37}$$

where $\gamma_j : \mathbb{R} \to \mathbb{R}$. Then h_θ is a *linear basis expansion.*

Linear basis expansion allows us to enlarge or transform the input space by augmenting the dependent observations x_i with derived features. This can take the form of indicator functions, polynomials, splines, and other nonlinear transformations. Linear basis expansion allows us to model more complicated solution spaces with linear methods, but suffers from interpolation problems in terms of locality and globality [HTF09, Chapter 5] – as well as the overfitting issues common to high variance models. Deciding on a useful function space for expansion is not trivial and can lead to dimensionality issues, particularly in the case of polynomial expansion methods.

3.5 Regularization

In the previous chapters, we looked at methods for modelling linear and nonlinear inverse problems via maximum likelihood or empirical risk minimization using least squares methods, and how these are linked via the projection theorem. These methods effectively deal with the ill-posedness related to existence (Property 2.1.1.i). In this section, we will address the remaining causes of ill-posedness; stability (Property 2.1.1.iii) and uniqueness (Property 2.1.1.ii).

Recall the result of Proposition 2.3.7, and how this demonstrated that the decay of singular values of an operator amplify the variance of the inherent noise component, causing instability (Property 2.1.1.iii). Moreover, in a discrete setting, any underdetermined system will necessarily contain zeroes in its singular value expansion as a result of ambiguity (Property 2.1.1.ii). In underdetermined systems we thus need to overcome a fundamental lack of information in the reconstruction process of an inverse problem. Such problems are sometimes referred to as $p \gg n$ problems [HTF09, Chapter 18], where p is the number of independent variables and n is the number of observations in the dataset $\mathcal{D}$. This is not necessarily constricted to the number of observations, but can also be manifested directly related to the dimensionality of the spaces $\mathcal{X}$ and $\mathcal{Y}$. If $\dim(\mathcal{Y}) \gg \dim(\mathcal{X})$ we have an underdetermined system for a forward problem, and in the case of $\dim(\mathcal{X}) \gg \dim(\mathcal{Y})$ we have an underdetermined system for an inverse problem. This means that for underdetermined systems, we can at best limit our solution space to a subspace (Theorem 2.4.2) and require additional information to determine a unique solution.

Stability and ambiguity represent potential issues when constructing learning models and manifest in the spectrum of the operator. While technically stable, we say such systems are *ill-conditioned.* The *conditioning* of a system or function is closely related to its *Lipschitz constant.*

Definition 3.5.1 (Lipschitz Continuity)**.** Let $\mathcal{X}, \mathcal{Y}$ be metric spaces with metrics $d_{\mathcal{X}}, d_{\mathcal{Y}}$ respectively. Let $\Phi : \mathcal{X} \to \mathcal{Y}$. Then if for all $x_1, x_2 \in \mathcal{X}$ we have

$$d_{\mathcal{Y}}\big(\Phi(x_1), \Phi(x_2)\big) \leq k d_{\mathcal{X}}(x_1, x_2) \tag{3.38}$$

we say Φ is *Lipschitz continuous*, and we call k the *Lipschitz constant* of Φ.

While the Lipschitz constant of a system is commonly defined in terms of metric spaces, for general Hilbert or Banach spaces, this readily extends to the norm by virtue of Lemma A.1.9. Furthermore, the idea of Lipschitz continuity for operators can be generalized for normed spaces by Definitions A.2.2 and A.2.3. This provides us with the general definition of *condition numbers* [Ric66].

Definition 3.5.2 (Relative Condition Number)**.** Let $\mathcal{X}, \mathcal{Y}$ be normed spaces. The *relative condition number* of a system $\Phi : \mathcal{X} \to \mathcal{Y}$ given input $x \in \mathcal{X}$ and perturbation δ is given by

$$\kappa(\Phi) = \lim_{\epsilon \to 0} \sup_{\|\delta x\|_{\mathcal{X}} \leq \epsilon} \frac{\|\Phi(x + \delta x) - \Phi(x)\|_{\mathcal{Y}}}{\|\Phi(x)\|_{\mathcal{Y}}} \cdot \frac{\|x\|_{\mathcal{X}}}{\|\delta x\|_{\mathcal{X}}}. \tag{3.39}$$

If the given condition number is small, we say Φ is *well-conditioned*, while a problem with a high condition number is said to be *ill-conditioned.*

Definition 3.5.2 is defined for a general multivariate system, and quantifies how sensitive the problem is to small perturbations in the data - thus a measure of general stability. In terms of compact linear operators, [Tur48] show that this simplifies to

$$\kappa(W) = \|W\|_{\text{op}} \|W^{\text{-}1}\|_{\text{op}}. \tag{3.40}$$

Of particular interest is the operator norm induced by the ℓ^2 norm (Remark A.3.22), as they relate to Hilbert spaces (Theorem A.3.23) and the singular value expansion. This can be seen by considering

$$\|W\|_2 = \sup_{x \neq 0} \frac{\|Wx\|_2}{\|x\|_2} = \sup_{\|\bar{x}\|_2 = 1} \|W\bar{x}\|_2 = \varsigma_1 \tag{3.41}$$

and noting that $\|W^{\text{-}1}\|_2 = 1/\varsigma_N$, the condition number for a compact operator on Hilbert spaces is thus given by

$$\kappa(W) = \frac{\varsigma_1}{\varsigma_N}. \tag{3.42}$$

which is in line with previous results from Proposition 2.3.7 and Theorem 2.3.6.

Having demonstrated how ill-conditioning can a source of instability, we would ideally want to enforce some notion of regularity to effectively counter this issue. This can be achieved by adding additional *a priori* information to our problem definition, often in the form of smoothness constraints. This process is known as *regularization*.

Definition 3.5.3 (Regularized Objective Function)**.** Let $\Phi(x;\theta) + \varepsilon = y$ be an inverse problem of model identification. Let $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$ be a parametrized learning model and let $C : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be some predefined objective function. A *regularized objective function* is given by

$$C_J\big(y, h_\theta(x); \alpha\big) = C\big(y, h_\theta(x)\big) + \alpha J(\theta). \tag{3.43}$$

We call $J(\theta)$ a *regularized penalty term* which applies additional constraints encoding a priori knowledge about the reconstruction process. We usually want to constrain the values of the solution in some way, commonly by penalizing the norm. The parameter $\alpha \in \mathbb{R}$ controls how much regularization is applied to the solution, and must usually be tuned for each individual problem. A common method of regularization is to impose penalties on the norm of the operator W_θ. These methods are known as *L^p regularization*.

Definition 3.5.4 (L^p regularization)**.** Let the regularizer $J(\theta)$ be given by

$$J(\theta) = \|\theta\|_p^p. \tag{3.44}$$

Then $J(\theta)$ is called a L^p regularization objective.

Both quintessential regularization methods; *Tikhonov* [Tik43] and *Lasso* [Tib96] – correspond to L^p regularization with $p = 2$ and $p = 1$ respectively. Any choice of $p \leq 1$ promotes sparsity in the solution set, and the optimization is convex for $p \geq 1$. This is why Lasso is the regularization method of choice for sparsity.

In Bayesian inference, the regularization is imposed by the choice of *prior distributions* which are then updated to form *posterior distributions* using available data. The Bayesian paradigm is arguably a more intuitive way of conceptualizing the process of regularization, as it directly seeks to quantify the availability of information, and effectively enhances estimation by deducing sensible prior distributions. As an example, a Bayesian interpretation for Lasso corresponds with the priors for θ being concentrated around the origin similar to a Laplacian prior [Gel+14, pp.368–369].

To show how regularization directly affects the condition numbers, consider an L^2-regularized solution for a causal problem $Wx + \varepsilon = y$. Applying the singular value decomposition, we have

$$\begin{aligned}
\hat{x} &= (W^*W + \alpha I)^{-1} W^* y && (3.45)\\
&= (V\Sigma U^* U \Sigma V^* + \alpha I)^{-1} V\Sigma U^* y && (3.46)\\
&= (V\Sigma^2 V^* + \alpha V I V^*)^{-1} V \Sigma U^* y && (3.47)\\
&= V(\Sigma^2 + \alpha I)^{-1} \Sigma U^* y && (3.48)\\
&= V D_\varsigma U^* y, && (3.49)
\end{aligned}$$

where D_ς is an augmented matrix with diagonal elements $\left(\varsigma_i/(\varsigma_i^2+\alpha)\right)_i$. This is in effect a modified pseudoinverse which directly lessens the impact of singular values of low or high magnitude. Tikhonov regularization thus has a similar effect to TSVD, but applies direct smoothing rather than truncation.

Example 3.5.5 (Upscaling with Tikhonov Regularization)**.** In our previous examples, we demonstrated methods for solving inverse problems in a deblurring setting. In this example, we instead look at the related problem of *image upscaling*, where we effectively want to reconstruct a high resolution image given from a downsampled image. To this end, we apply regularization via constructing a modified pseudoinverse using SVD, as outlined in Equation (3.49). We also compare the method to the TSVD approach, outlined in Example 2.4.3.

Method	K/α	PSNR	PSNR (Noise)	SSIM	SSIM (Noise)
TSVD	**9000**	34.544	22.198	0.864	0.353
	7000	28.647	26.679	0.645	0.550
	5000	27.469	26.787	0.591	0.557
$\mathbf{L^2}$	**1.0**	31.602	27.841	0.783	0.614
	5.0	28.613	28.195	0.696	0.672
	10.0	27.313	27.199	0.679	0.671

Table 3.1: Results of upscaling examples with TSVD and Tikhonov regularization. The best results for each method are highlighted in gray.

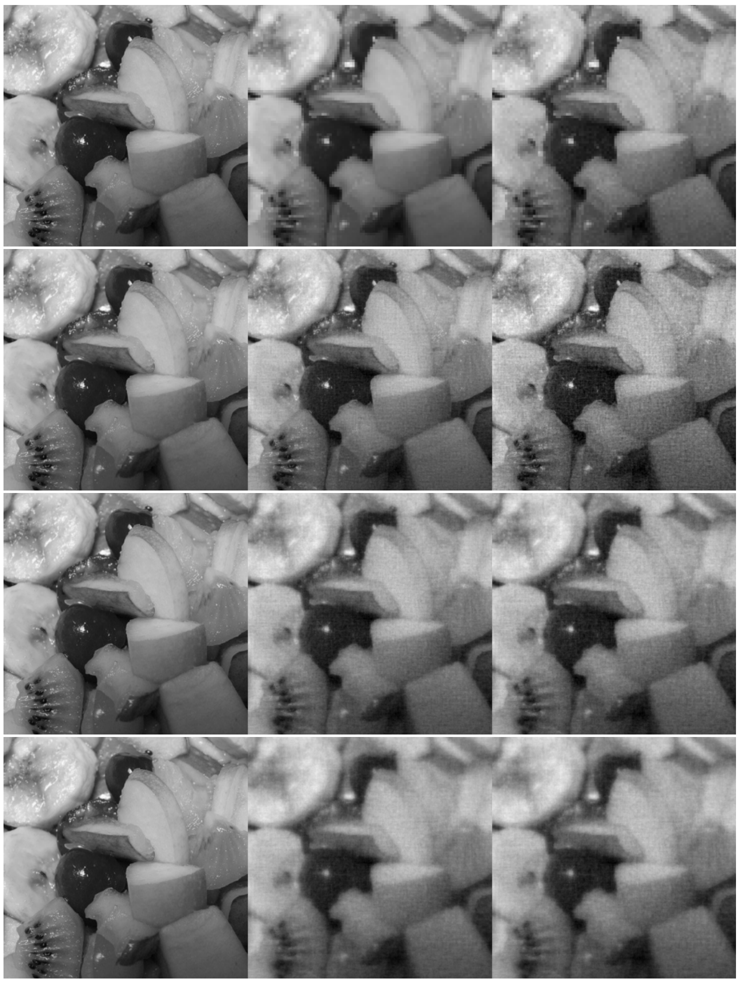
Figure 3.1: Reconstructions with Tikhonov regularization for $\alpha_1 = 1.0$ (second row), $\alpha_2 = 5.0$ (third row) and $\alpha_3 = 10.0$ (bottom row). The top row features the original image (left), the downsampled image (middle) and a noisy downsampled image (right).

For our example images, we use centered grayscale images taken from the COCO dataset (see Section 7.1 and Table 1.6). The original image space $\mathcal{X}$ consists of 384×384 single channel images. The forward operator downsamples the images using average pooling with a 4×4 kernel, resulting in the image space $\mathcal{Y}$ containing images of dimension 96×96, corresponding to a downsampling ratio of 16:1. The results can be seen in Table 3.1 and Figure 3.1. Similarly to what we saw in Example 2.4.3, there is a general degradation in reconstruction quality for nonnoisy images, but an increase in reconstruction quality for noisy images. We also note that Tikhonov regularization generally provides better reconstructions than TSVD in this example.

CHAPTER 4

Neural Networks

The term *neural network* have come to be rather broad, encompassing a growing plethora of different models and architectures. In this chapter, we provide an overview of terminology and definitions to fully describe feed-forward neural networks, show how they can be considered generalizations of classical statistical models, and discuss properties of universal approximation. We also introduce several commonly used components and architectures, and discuss how they can be applied for data-driven model identification problems.

4.1 Fundamentals of Neural Networks

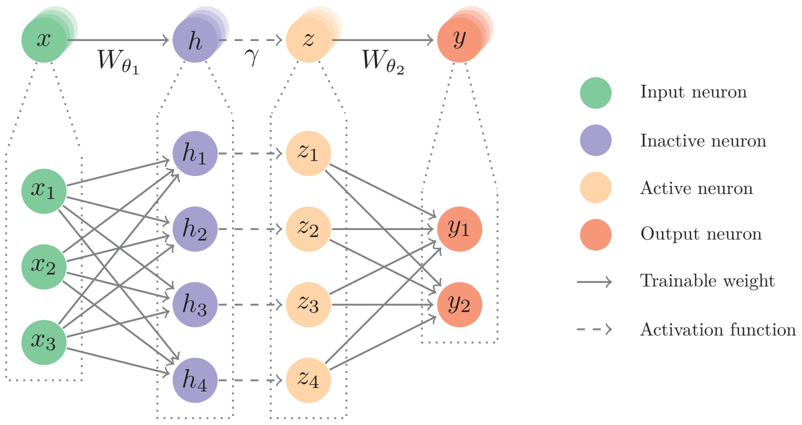

Figure 4.1: Overview of the components of a densely connected feed forward neural network. The network has a single hidden layer of four activated neurons, as well as three inputs neurons, and two non-activated output neurons.

Neural networks are essentially a family of composite nonlinear learning models which can be used for modelling high-dimensional functions with arbitrary levels of complexity. Much of the terminology applied to neural network models reflects their biological inspiration, with the goal of generating high-dimensional function approximations which can be learned using data-driven approaches.

A *network layer* can be considered a computational stage of a neural network, consisting of one or more *neurons* with a set of associated weights, often followed by an *activation function* which are designed to mimic neural activation in biological organisms. A *hidden layer* is any intermediate computational layer of neurons between input and output layers. In a *dense* or *fully connected* network, each neuron has a unique connected path of weights to all other neurons in prior or subsequent layers. Furthermore, the output of every neuron in a strictly *feed forward* network is only dependent on neurons in prior layers, such that there is no feedback from either subsequent neurons or neurons in the same layer. Figure 4.1 details how the standard components of a neural network are connected in a dense feed-forward network.

While the terminological familiarity with neurology seems to imply complex model structures, at their core, neural networks are simple applications of linear predictors with nonlinearities, similar to the generalized linear models introduced in Section 3.4.

Definition 4.1.1 (Activation Function)**.** Let $\mathcal{H}, \mathcal{Y}$ be Hilbert spaces and let $\eta = Wx$ be a linear predictor. An *activation function* $\gamma\colon \mathcal{H} \to \mathcal{Y}$ is a function acting on a linear predictor to yield

$$y = \gamma(\eta). \tag{4.1}$$

Activation functions are often univariate functions which are applied point-wise to each neuron. For an activation function to be useful in a general setting, it should ideally adhere to certain specific functional properties.

Observation 4.1.2 (Properties of activation functions)**.** *An activation function generally exhibits one or more of the following properties;*

(i) *(Non-affine)* $\gamma(z) \neq az + b$*,*

(ii) *(Smooth)* $\gamma \in \mathcal{C}^n$ *for* $n > 0$*,*

(iii) *(Monotonic)* $\gamma(z_1) \leq \gamma(z_2)$ *for* $z_1 \leq z_2$*,*

(iv) *(Low complexity)* γ *has low computational complexity,*

(v) *(Approximately. identity at origin)* $\gamma(h) \approx h$ *for* $|h| < \delta$*,*

It is reasonably clear that Properties 4.1.2.ii and 4.1.2.iii are useful for gradient optimization, while Property 4.1.2.iv yields better computational performance. The benefit of the two remaining properties might be less clear, however. If an activation function approximates identity close to the origin, this promotes optimal gradients when network weights are initialized close to zero [AH17], while non-affine activation ensures *universal approximation* under certain conditions [KL20]. We discuss this property further in the context of Theorem 4.1.6.

The first neural networks – called *perceptrons* [Ros57] – applied neural activation using a binary step function with learnable threshold values. Later, this threshold was replaced by an optional learnable *bias*, and the binary step function made way for continuously differentiable sigmoidal functions like the logistic function or the hyperbolic tangent function. Later, in [Hah+00] the authors outlined the biological inspiration and mathematical benefits of the *rectified linear unit* (RELU), including computational efficiency and sparsity. The rectified linear unit is a projection $\gamma_{\text{RELU}} : \mathbb{R} \to \mathbb{R}_{\geq 0}$ given by

$$\gamma_{\text{RELU}}(x_i) = \begin{cases} x_i, & \text{if } x_i \geq 0; \\ 0, & \text{if } x_i < 0. \end{cases} \tag{4.2}$$

The ReLU activation is widely used in feed-forward neural networks due to its sparsity and convergence properties [KSH12], however several other activation functions are in practical use. Definition 4.1.1 is sufficiently relaxed to allow for a variety of possible activation functions. A few of these are of particular interest to this thesis, particularly the *exponential linear unit* (ELU) [CUH16] given by

$$\gamma_{\text{ELU}}(x_i; \beta) = \begin{cases} x_i, & \text{if } x_i \geq 0; \\ \beta(\exp(x_i) - 1), & \text{if } x_i < 0; \end{cases} \tag{4.3}$$

and the *continuously differentiable exponential linear unit* (CELU) [Bar17], which in turn is given by

$$\gamma_{\text{CELU}}(x_i; \beta) = \begin{cases} x_i, & \text{if } x_i \geq 0; \\ \beta(\exp(x_i/\beta) - 1), & \text{if } x_i < 0. \end{cases} \tag{4.4}$$

In addition to these, discrete classification and categorical decision making tasks often utilize the logistic function introduced in Equation (3.36). The multivariate extension of the logistic function is referred to as the *softmax* function, introduced by Boltzmann [Bol12]. The softmax function is given by

$$\gamma_{\text{SOFTMAX}}(x; \beta)_i = \frac{\exp(\beta x_i)}{\sum_{j \leq d} \exp(\beta x_j)}, \tag{4.5}$$

Similar to the logistic function, the softmax yields a probability distribution which can be used as a parametrization for a categorical distribution on d classes. The name softmax stems from *soft argmax*, referring to the function being a smooth differentiable maximum on a d-dimensional vector, and it approximates the max function as the temperature parameter $\beta \to \infty$. When $\beta \to -\infty$ it instead approximates the min function, and when $\beta \to 0$ the function outputs a discrete uniform distribution over all d classes.

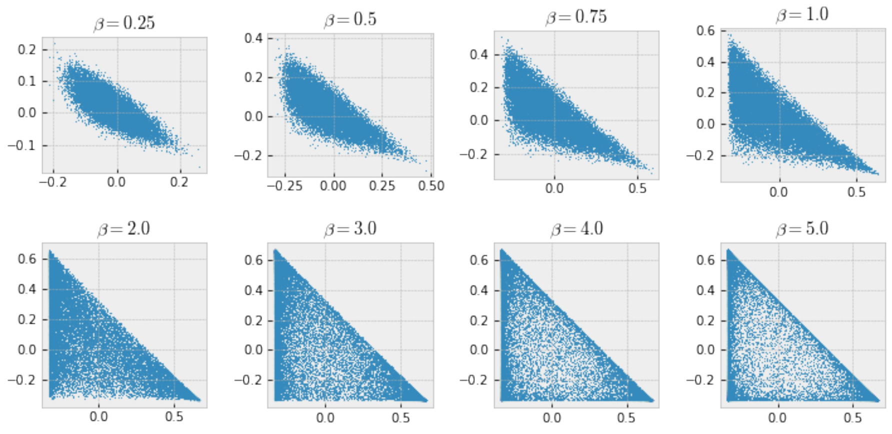

Figure 4.2: Projection of softmax with different temperature parametrizations on three gaussian random variables to two-dimensional simplex using PCA.

Geometrically, the softmax function projects an element of $\mathbb{R}^d$ into a $d-1$ simplex on a plane, visualized in Figure 6.2. In general, increasing the temperature parameter β effectively serves to push points toward the closure of the $d-1$ simplex. Additionally, the function is translation-invariant, i.e. $\gamma(x;\beta) = \gamma(x+b;\beta)$ for all $b \in \mathbb{R}^d$. The apparent overdetermination is connected to this translation invariance, which manifests by $\sum_{j \le d} \gamma(x)_j = 1$, so any combination of $d-1$ parameters determines the last. This is practically applied in statistical classification by assigning one dimension as a *pivot* and estimating the probabilities of the $d-1$ remaining classes.

The nonlinear activation of neurons in the linear predictor allows a neural network to approximate nonlinear solution spaces, and is thus an essential component in *single layer networks* – the fundamental building block for more complex network structures.

Definition 4.1.3 (Single Layer Neural Network)**.** Let $(\mathcal{X} \times \mathcal{Y}, \mathcal{B})$ be an observable space, $\mathcal{H}, \Theta$ be Hilbert spaces, and let $\theta \mapsto W_\theta$ be a parametrization of a compact operator. Let $\eta = W_\theta x$ be a linear predictor, and let $\gamma : \mathcal{H} \to \mathcal{Y}$ be an activation function. Then the mapping $\psi : \mathcal{X} \to \mathcal{Y}$ given by

$$\psi(x;\theta,\gamma) = \gamma(\eta) \tag{4.6}$$

is called *single layer feed forward neural network*. The parameters for the operator W_θ are commonly referred to as the *network weights*.

Note that Definition 4.1.3 is conceptually in line with the discussion in Section 3.4 of linearized transformations. In fact, both generalized linear models and linear basis expansion can be modelled with a single layer neural network by selecting appropriate activation functions. This apparent connection is trivial, but illustrates how modern machine learning techniques have their origin in classical statistical modelling techniques.

The main advantage of modelling with neural networks is the fact that they display nonlinear interpolation properties. These properties do not apply directly to single layer models, but require an extension to the composition of *multilayer neural network* models.

Definition 4.1.4 (Multilayer Neural Network)**.** Let $(\mathcal{Z}_i)_{i=0}^N$ be a sequence of Hilbert spaces, and let $(\psi_i)_{i=1}^N$ be a sequence of single layer neural networks $\psi_i \colon \mathcal{Z}_{i\text{-}1} \to \mathcal{Z}_i$. Let $\psi^{[i]}$ be a composition on the form $\psi_i \circ \cdots \circ \psi_1$, such that the output of the ith layer in the network is given by

$$\begin{aligned} z_i &= \gamma_i(\eta_i) \\ &= \psi_i(z_{i\text{-}1}; \theta_i, \gamma_i) \\ &= (\psi_i \circ \psi_{i\text{-}1} \circ \cdots \circ \psi_1)(x; \theta_{1:i}, \gamma_{1:i}) \\ &= \psi^{[i]}(x; \theta_{1:i}, \gamma_{1:i}). \end{aligned}$$

Now let $(\mathcal{X} \times \mathcal{Y}, \mathcal{B}) = (\mathcal{Z}_0 \times \mathcal{Z}_N, \mathcal{B})$ be an observable space. Then $\psi^{[N]} : \mathcal{X} \to \mathcal{Y}$ is a *multilayer feed-forward neural network*, and we will denote multi-layer networks by $\Psi(x; \vartheta, \Gamma) = \psi^{[N]}(x; \theta_{1:N}, \gamma_{1:N})$.

Remark 4.1.5 (Mathematical Notation of Neural Networks)*.* We consider the notation $\Psi(x; \vartheta, \Gamma)$ as sufficiently concise, but we would like to point out some idiosyncrasies which might not be immediately clear to the reader. The implication is that a neural network consists of a sequence of activation functions

$$\Gamma = (\gamma_i)_{i=1}^N \tag{4.7}$$

as well as a partially ordered set of parameters

$$\vartheta = \{\theta_j \in \Theta_j : j \in 1, \dots, N\}, \tag{4.8}$$

so the full parameter space for the model parametrization can be considered a product space $\Theta_1 \times \cdots \times \Theta_p$. This definition of the model parametrization should however be considered sufficiently relaxed to include more than simply the weight parameters, including parameters for activation functions, dimensionality of layers as well as the general network structure. Note that some of these parameters are estimated and *trainable*, while others are hyperparameters fixed by the modeller, however we will always consider ϑ to include the parametrization of the sequence of linear or augmented affine operators, which we denote by $\mathcal{W}_\vartheta = (W_{\theta_i} : i = 1, \dots, N)$. This is to be consistent with our previous definition of learning models in Definition 3.2.2, where each parametrization of a network exists in an identifiable parametrized hypothesis space $\mathscr{H}$ where each hypothesis h_ϑ represents a neural network model.

We stress that we will occasionally omit the reference to the sequence of activation functions Γ, or even the parameters ϑ when the specification is not required to avoid unnecessary ornamentation in our notation. However the notation is presented, we will naturally assume that the activation functions and parameters are present, even if not explicit in notation.

In practice, almost all neural network models apply multilayer structures due to aforementioned *universal approximation theorems* - a collection of results which show that neural network models with certain general structures can approximate any well behaved function arbitrarily well. The earliest and most ubiquitous of these results is the arbitrary width case for a two-layer network with logistic sigmoidal activation, and can be concisely summarized as follows.

Theorem 4.1.6 (Arbitrary Width Universal Approximation Theorem)**.** *Let $\Phi(x,\theta)+\varepsilon=y$ be a forward problem, let $\gamma_1=\text{logistic}, \gamma_2=\text{id}$, and let θ_1,θ_2 be parametrizations of affine transformations such that $\theta_i \mapsto W_{\theta_i}$. Then there exists some arbitrarily high dimensional $\theta_1,\theta_2\in\vartheta$ such that given*

$$\Psi(x;\vartheta,\Gamma)=W_{\theta_2}\gamma_1(W_{\theta_1}x), \tag{4.9}$$

then for all $\epsilon\in\mathbb{R}$ we have

$$\|\Phi-\Psi\|_\infty<\epsilon. \tag{4.10}$$

The full exposition and proof can be found in [Cyb89]. Following this result, Hornik [Hor91] showed that sigmoidal functions are not a requirement for the universal approximation property to hold, while suggesting that the multi-layer structure of the network was the instrumental factor for the universal approximation property to hold, while [Pin99] demonstrated that nonpolynomial activation is a sufficient constraint on activation. [Yar18] provided another important contribution, showing that permutation invariant operators with arbitrary width and an intermediate polynomial layer can act as universal function approximators. This has implications to convolutional networks, which we introduce in Section 4.3.

Later results focused on showing universal approximation properties in arbitrary depth networks – sometimes referred to as *deep networks* – with RELU activation [Zho+17], and recently [KL20] refined this constraint to nonaffine activation functions on networks of arbitrary depth and bounded width using common activation functions. The authors also rigorously demonstrated that arbitrary depth network in general are expected to perform better than arbitrary width networks, confirming empirical results [Pog+16]. There are however certain limits on the benefits of deeper networks, especially related to estimation, which we return to in Section 4.2.

While the property of universal function approximation is alluring, it is related to polynomial interpolation and can thus be considered special cases of the important *Stone-Weierstass Theorem* [MW99, Theorem 8.9]. However, for multivariate non-linear approximation, neural networks are currently state-of-the-art and have incontrovertibly had a huge impact on the field of machine learning and statistics. The problem exhibited by these models is that the structure of estimated models does not provide any significant insight into how the system they approximate actually works, and are often describes as purely *discriminative black box* models with little inference capabilities. We briefly discussed the limitations of discriminative modelling in Section 3.2, and we will return to address the apparent weaknesses of neural network models in Section 5.1.

4.2 Estimation, Optimization and Learning

As any other parametrized learning model, neural networks require some algorithmic method of estimating the parameters for the sequence of operators $\mathcal{W}_\vartheta$ to allow the model to yield accurate predictions. This procedure is referred to as *training* the network, and is generally performed by iterative optimization methods by empirical risk minimization via an objective function (Equation (3.30)). As in Definition 3.5.3, regularization can be applied to any given model by expanding the objective function to include one or more constraints encoding some a priori information about the solution space. Given an observable product space, the optimized parameters $\hat{\vartheta}$ for a multilayer neural network are then given by

$$\hat{\vartheta} = \arg\min_{\vartheta} C_J\big(y, \Psi(x; \vartheta, \Gamma); \alpha\big) \tag{4.11}$$

$$= \arg\min_{\vartheta} C\big(y, \Psi(x; \vartheta, \Gamma)\big) + \alpha J(\vartheta). \tag{4.12}$$

Parameter estimation – or weight estimation – for neural networks requires a set of initial parameters drawn from some appropriate distribution. In practice, initialization methods are generally designed to ensure that the activated outputs of each layer become too small or too large to be numerically stable, as most well-defined networks quickly find reasonable local minima or maxima in the output space for which a good approximation can be extracted [GB10; He+15]. Once the weights are initialized, the network is subsequently optimized by estimating the weights using observations from a dataset $\mathcal{D}$ Definition 3.1.2; canonically by way of *gradient descent*.

Definition 4.2.1 (Gradient Descent)**.** Let $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$ be a parameterized learning model over a normed observable space such that $\vartheta \mapsto h_\vartheta$ with $h_\vartheta \in \mathscr{H}$. Let $\mathcal{D}$ be a dataset of n observations on $\mathcal{X} \times \mathcal{Y}$ and let $C : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be an differentiable objective function. A *gradient descent scheme* is an iterative process

$$\vartheta^{(i+1)} = \vartheta^{(i)} - \varrho \sum_{x.y \in \mathcal{D}} \nabla C_J\big(y, h_{\vartheta^{(i)}}(x)\big) \tag{4.13}$$

for iterations $i = 1, \ldots, n$.

The hyperparameter $\varrho \in \mathbb{R}$ controls the step size in the direction of the gradient – commonly referred to as the *learning rate* – which can be tuned to improve the performance of the weight estimation process. Moreover, note that the objective function can be extended to include one or more regularization terms without any loss of generality.

The issue with the form of gradient descent in Definition 4.2.1 is that global optimization on the training data is almost always too restrictive for finding sufficiently good approximations, relying more heavily on the weight initialization. As such, it relies heavily on *exploiting* the initialized weights to derive an optima, as opposed to *exploring* the solution space, evaluating more distant candidate solutions which might provide a better global optima towards the defined optimization objective. This tradeoff between exploration and exploitation can be more effectively handled by iteratively updating parameters using *stochastic gradient descent* to promote exploration.

Definition 4.2.2 (Minibatch Stochastic Gradient Descent)**.** Let $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$ be a parameterized learning model over a normed observable space such that $\vartheta \mapsto h_\vartheta$ with $h_\vartheta \in \mathscr{H}$. Let $C : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be a differentiable objective function, and let $\mathcal{D}$ be a dataset with n observations over $\mathcal{X} \times \mathcal{Y}$. Fix $E, k \in \mathbb{N}$ with the condition $k < n$, and let $m = E(n \mod k)$. Now let $\left(B_i \subset \mathcal{D} : \#(B_i) = k\right)_{i=1}^{m}$ be a sequence of randomly permuted partitions of $\mathcal{D}$. Then a *minibatch stochastic gradient descent scheme* is given by the iterative process

$$\vartheta^{(i+1)} = \vartheta^{(i)} - \varrho \sum_{x.y \in B_i} \nabla C_J \left(y, h_{\vartheta^{(i)}}(x)\right) \tag{4.14}$$

for iterations $i = 1, \ldots, m$.

When training neural networks, each permuted subset B_i denotes a *minibatch* of training examples. As we generally want the process to successively iterate in a manner which includes all observations, this process is usually performed by grouping the subsets B_i such that the entire dataset is seen over one *epoch* of $n \mod k$ minibatches. The total number of iterations is then repeated over E epochs such that all observations are encountered several times. The total number of iterations is then $m = E(m \mod n)$, however other methods can be implemented to lower the total number of iterations given some approximate convergence criteria. We note that the apparent early convergence of a network can sometimes be decieving. Recently, [Pow+21] showed that in certain cases, an exceedingly large number of iterations can be necessary for true convergence.

Both definitions of gradient descent are straightforward to apply to single layer neural networks, but implementing this for multilayer networks are not as straightforward. For such models, the parameter update is computed sequentially by calculating successive gradients with respect to each estimated parameter in a reversed order to forward computations. This process is called *backpropagation* [RHW86] and is a direct application of the *chain rule for differentiation.*

Example 4.2.3 (Backpropagation)**.** The backpropagation process can be illustrated using a two-layer feed-forward network structure, yielding

$$\theta_2^{(i+1)} = \theta_2^{(i)} - \varrho \sum_{x,y \in B_i} \frac{\partial C_J}{\partial \gamma_2} \frac{\partial \gamma_2}{\partial \eta_2} \frac{\partial \eta_2}{\partial \theta_2^{(i)}} \tag{4.15}$$

$$\theta_1^{(i+1)} = \theta_1^{(i)} - \varrho \sum_{x,y \in B_i} \frac{\partial C_J}{\partial \gamma_2} \frac{\partial \gamma_2}{\partial \eta_2} \frac{\partial \eta_2}{\partial \gamma_1} \frac{\partial \gamma_1}{\partial \eta_1} \frac{\partial \eta_1}{\partial \theta_1^{(i)}}. \tag{4.16}$$

Equation (4.15) clearly demonstrates the application of the chain rule. In practice, the computation of backpropigation can be simplified by what is known as the *delta-rule* [RHW86], which simply applies the observation that successive applications of the chain rule always have similar leading products, which can be memoized for more effective computations.

In modern software for neural networks, each functional operation has an associated gradient function, and their application produces arrays which store the associated inputs for backward computations [Pas+17]. This generates a computational graph for the backpropagation process and allows for effective subsequent updates to the weights.

Still, there are certain issues with any type of gradient descent algorithm that must be considered. The most important of these is the choice of learning rate, or step size. A low learning rate might provide stronger guarantees for convergence, but the optimization process is more prone to be immobilized in local optima or saddle points. On the other hand, a high learning rate can expediate exploration of the solution space but can render the optimization process divergent altogether. Both these issues can lead to slower convergence rates. To redress these issues, neural networks are generally trained by introducing an *adaptive learning rate* $\varrho^{(i)}$ to Definition 4.2.2. This technique imposes some notion of mechanical weight or friction to the process, which is often described as analogous to propelling some ball onto the solution surface, and apply common conservation laws to yield a momentum for each iteration. Several of these optimization processes have been developed specifically for training neural networks [DHS11; Zei12]. In particular, we will make use of the ADAM optimizer proposed in [KB17], which is currently considered canonical for weight optimization in neural networks.

When estimating neural network models, we are generally optimizing parametrizations of compact linear operators and bias terms with respect to simple activation functions. As alluded to in the properties of activation functions (Observation 4.1.2), certain issues can arise due to the choice of activation for the layers in the network. While the sparsity and computational efficiency of RELU activation is often beneficial, it also maintains some inadequacies. Firstly, it is nonsmooth with discontinuous gradients. Secondly, it is generally unconstrained, which can produce outputs of arbitrary magnitude if the inputs are sufficiently large. The third and arguably most important issue with ReLU activation is a byproduct of the sparsity property. Any nonpositive input will have a gradient of zero, which can be a cause of problems for gradient optimization methods. This issue is referred to as the *dying RELU* problem [Lu+20] . This is the main motivation for the introduction of the ELU and CELU activation functions (Equations (4.3) and (4.4)).

The dying RELU problem circumstantiates the more general problem of low magnitude or zero-valued gradients in network estimation. The domain of the gradient of activation functions are generally constrained to a compact strictly positive Borel set. For commonly used functions adhering to Observation 4.1.2, this domain generally falls in the open interval $]-1,1[$, so the backpropigated gradients are generally contractive. This results in the phenomenon where weights closer to the output have a tendency to absorb most of the errors, and fewer terms are propagated to the earlier weights close to the input. This issue is more prevalent in deep models than in wide models, causing modelled solutions to plateau in saddle points more frequently. [Hoc91] was the first to discuss this issue and coined the term *vanishing gradient* problem, which directly inspired the development of the field of *network learning dynamics* which we discuss in more depth in Section 6.1.

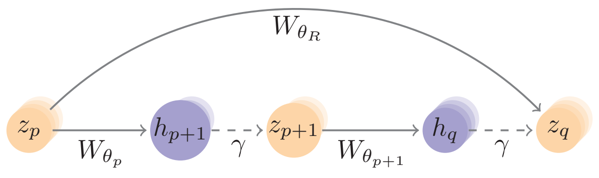

Figure 4.3: The structure of a residual block. The input z_p is propagated deeper into the network by the weight W_{θ_R}.

4.3 Network Components and Layers

In addition to the standard components in a neural network, there are several specialized components which serve particular purposes. In this section, we will outline the general network components and layer types relevant for the network architectures featured in this thesis, and discuss their motivation.

Residual Blocks

For sufficiently deep network structures, each layer of a network should theoretically improve accuracy of an approximation towards an optimal solution. In practice, much of the actual improvement is restricted to only a subset of available layers due to vanishing gradients and dimensionality problems. Thus, much of the informational capacity of a deep network becomes tied up in approximating linear mappings, referred to as the *degradation problem*. In [He+15], the authors demonstrate that a more effective learning process can be induced by allowing portions of the network to deal with residuals, as opposed to having a network learn identity-like mappings in several layers. These components are groups of layers with a *skip connection*, called *residual blocks*.

Definition 4.3.1. Let $\mathcal{Z}_p, \mathcal{Z}_q$ be Hilbert spaces, and let $\psi^{[n]} \colon \mathcal{Z}_p \to \mathcal{Z}_q$ be a composition of $q - p = n$ internal network layers. Furthermore, let θ_R be a parametrization $\theta_R \mapsto W_{\theta_R}$ for a compact operator $W_{\theta_R} \colon \mathcal{Z}_p \to \mathcal{Z}_q$. A *residual block* is then given by

$$z_q = W_{\theta_R} z_p - \psi^{[n]}(z_p; \theta_{p:q}, \gamma_{p:q}). \tag{4.17}$$

The idea behind a residual block is to *hotwire* the output using a direct layer with trainable weights θ_R. This linear connection is allowed to process much of the approximately linear transformations while the multi-layer structure $\psi^{[n]}$ is free to process nonlinearities in the residual space generated by the range of $(I - H_{\theta_R})$. Residual blocks are therefore especially useful for tasks where the input space and output space are sufficiently similar as to warrant some linear mapping, like the identity mapping. Residual blocks can be composed and combined in a modular fashion, similarly to individual layers in feed-forward structures. We note that the original paper specifically refers to residual blocks with non-parametrized identity weights as opposed to a parametrized operator W_R; a special case of our convention in Definition 4.3.1.

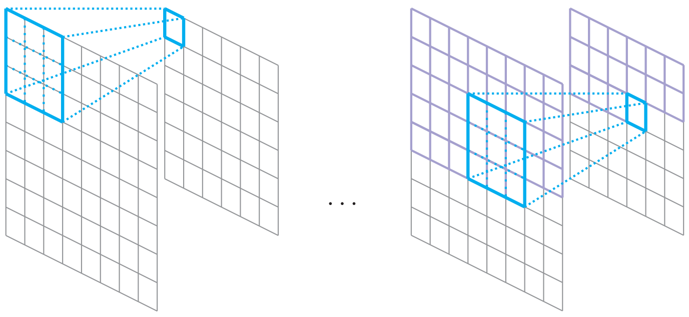

Figure 4.4: An illustration of non-padded two-dimensional convolution with a 3×3 kernel. The kernel slides over the image to produce pixels in the target image.

Convolution Layers

In Section 2.2 we introduced the concept of convolution (Definition 2.2.6), and showed how deblurring problems are naturally formulated as convolution problems in Example 2.2.7. Convolution operators also play an important part in neural networks and have been shown to be especially effective in the field of image processing [Fuk79; LeC+89].

Definition 4.3.2 (Convolutional Layer)**.** Let ψ be a single layer neural network. If the associated operator W_θ is a linear combination of convolution operators, then ψ is a *convolutional* layer.

The success of convolutional layers has led to these layers becoming a de facto standard for image processing with neural networks [GBC16, Sec. 9.1]. The motivation for introducing convolutional layers are specifically tied to spatial dependencies in a signal or image. While any image can be vectorized for applications with a linear operator, this representation discards spatial structures in an image which in turn discards important information. Convolution operators maintain the spatial structure of an image by applying parametrized discrete *kernels* of appropriate dimensions. An illustration of discrete two dimensional convolution over a single channel can be observed in Figure 4.4. Another advantage of convolutional layers is that they display properties of *translational equivariance* [MMD20], meaning a convolutional layer can be effective at processing spatial structures independent of their relative location in an input image.

A convolutional layer is parametrized using *stride*, *padding* and *dilation* . The stride of a convolution is an optional step size for the convolution, while padding expands the convolution domain to outside the range of the image. Lastly, dilation applies spacing between each value in the kernel, imposing sparsity in the kernel. These techniques can alleviate some local spatial bias, but can also introduce certain artifacts and aliasing effects especially for deconvolution [ODO16].

In [Luo+16] the authors highlight an important issue with the application of convolution operators in neural networks by showing that the central values of an image or signal will generally be used more frequently in the computation of the outputs. When successively applying layer-wise discrete convolution, the pixels close to the boundaries of the input signal will ultimately be weighted less than the central pixels in the computed output for each channel. This means there are fewer paths between the values close to the boundary and the output values than the more central values. This means that while a network *theoretically* has the capability of allowing inputs to influence every single output in a signal, the *effective* influence might be minimal to negligible. The authors call these measures of the connectivity between output and input signals for the *receptive field* of the network.

Another important property of convolutional layers is that their periodicities admit inherently sparse operators which can act on high dimensional data. In discrete convolution, the parametrization is commonly of much lower dimension than the signals they operate on. This makes convolution operators highly efficient in terms of number of operations and memory, allowing them to operate on high dimensional images. Furthermore, as the convolution operator is independent of the dimensionality of the inputs, a convolutional model can be constructed to be more or less invariant to the input dimensions using *pooling* layers [Cir+11]. The nature of these operations make them contentious in the eyes of some researchers [Hin17], while [Spr+15] suggests they can be replaced altogether by a series of appropriately strided convolution layers. In Appendix A.8, we show how convolution layers can be constructed in matrix form.

For the purposes of this thesis, we are also particularly interested in the adjoint representations of convolution layers. From the exposition in Appendix A.8, it is clear that the adjoint representation of a convolution simply corresponds to a convolution with a reversed permuted kernel; also known as the *correlation* operator. In signal processing, adjoint operators are often called *matched filters*, and have been shown to be optimal linear filters for maximizing the *signal-to-noise* ratio [Tur60]. In the most common contemporary software packages for neural networks, the correlation operator is commonly applied in place of the standard convolution operator. As each kernel is estimated through training the end result is the same, however the operator loses its commutative properties [GBC16, Sec. 9.1].

Seperable Patch Layers

In Example 2.3.8 we discussed how the form of seperable filters can be applied row-wise and column-wise to an image to reduce the parameter space of a linear operator. Surprisingly, there seems to be few models adopting seperable filters in neural networks, possibly due to the success of convolutional layers.

Recently, an alternative to convolutional layers was proposed in [Tol+21], which applies a variation of linear seperable filters over individual image patches while simultaneously generalizing over multiple patches. These models retain moderately low dimensionality in their parametrization while applying densely connected linear weight operators. The full architecture of the proposed model is referred to as a *mixer model*, however the individual layer structure is not specifically referred to by name. We will however in the context of this thesis instead refer to these layers as *seperable patch layers* which we find more descriptive.

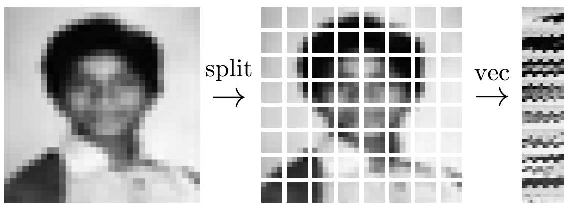

Figure 4.5: Illustration of the process of patch splitting and vectorization of images for seperable patch layers. The input image on the left in the illustrated example is given by $x \in [0,1]^{32\times 32}$ while the output image on the right is given by $\bar{X} \in [0,1]^{64\times 16}$.

Definition 4.3.3 (Seperable Patch Layer)**.** Let $\mathcal{X}$ be a Hilbert space over $\mathbb{K}$ with a product structure, such that $\dim(\mathcal{X}) = c \times h \times w$. For each $x \in \mathcal{X}$, let $\bar{x} = (\bar{x}_p \in \mathbb{K}^k)_{1\leq p\leq n}$ be a sequence of subdivided patches of x. Furthermore, let $\bar{X} = [\mathrm{vec}(\bar{x}_1), ..., \mathrm{vec}(\bar{x}_n)]^\intercal \in \mathbb{K}^{n\times k}$ be a matrix such that each row is a vectorized patch of image x. Now let ψ be a single layer neural network with a compact operator $W_\theta = U^*_{\theta_c} \otimes V_{\theta_r}$ with $U_{\theta_c} \in \mathbb{K}^{m\times n}$ and $V_{\theta_r} \in \mathbb{K}^{k\times q}$, such that

$$\begin{aligned}\psi(x;\theta,\gamma) &= \gamma(W_\theta x) && (4.18)\\ &= \gamma\big((U^*_{\theta_c} \otimes V_{\theta_r})\mathrm{vec}(\bar{X})\big) && (4.19)\\ &= \gamma(U_{\theta_c}\bar{X}V_{\theta_r}). && (4.20)\end{aligned}$$

Then $\psi : \mathbb{K}^{n\times k} \to \mathbb{K}^{m\times q}$ is called a *seperable patch layer.*

We note that the formulation of the split- and vectorize operations of the seperable patch layers is somewhat vague, however Figure 4.5 should help illustrate the process more clearly. Definition 4.3.3 more or less follow the definition of a seperable filter, however the main difference lies in how the image is reworked into patches in a grid. As such, these layers are similar in form to full-stride convolutional layers featuring depthwise seperable convolution [Cho17; How+17], but densely connected across patches and individual pixels in an image.

Coupling Layers and Normalizing Flows

Coupling layers [DKB15; RMW14] are network layer structures which have recently garnered attention in the context of probabilistic modelling and *invertible neural networks.*

Definition 4.3.4 (Affine Coupling Layer)**.** Let $\mathcal{X}, \mathcal{Y}$ be seperable Hilbert spaces with $\dim(\mathcal{X}) = \dim(\mathcal{Y}) = D$. Let $x_{:d} = (x_1, \ldots, x_d)$ and $x_{d:} = (x_{d+1}, \ldots, x_D)$ for $x \in \mathcal{X}$ and let $y_{:d}, y_{d:} \in \mathcal{Y}$ be defined similarly. An *affine coupling layer* is a map $g : \mathcal{X} \to \mathcal{Y}$ such that

$$\begin{aligned}g_{:d}(x_{:d}) &= y_{:d} = x_{:d} && (4.21)\\ g_{d:}(x_{d:}) &= y_{d:} = x_{d:} \odot m(x_{:d}) + b(x_{:d}), && (4.22)\end{aligned}$$

where m, b are multivariate functions with support on $\mathcal{X}_{d:}$.

An affine coupling layer g is bijective if we have access to the transformations m, b and $y_{d:} = x_{d:}$, and can thus be composed to form more complex invertible structures in a neural network model. Coupling layers generally induce a triangular operator structure common in *autoregressive models* [SS17a, Chapter 3], a class of sequence models which have been applied successfully in both spatial and temporal signal processing [OKK16; Oor+16].

The benefit of triangular structures is that calculation of the determinant of the operator induced by g is greatly simplified by $\det(g) = \operatorname{Tr}(g)$, making the change of variables formula (Theorem A.4.8) tractable. In this context, they are the canonical example of invertible transformations used to construct probabilistic neural networks called *normalizing flows*, where certain criteria are applied to ensure that the determinant of the Jacobian of the network layer $| \det(\mathcal{J}_{\Psi}^{-1}(x)) |$ is tractable.

Definition 4.3.5 (Normalizing Flow)**.** Let $(\mathcal{X}, \mathcal{B}_x), (\mathcal{Z}, \mathcal{B}_z)$ be observable Hilbert spaces with $\dim(\mathcal{Z}) = \dim(\mathcal{X})$, and let $\mathrm{Z} : \Omega \to \mathcal{Z}$ be a random element with known probability distribution $\mathbb{P}_{\mathrm{Z}}$. Let $\psi : \mathcal{Z} \to \mathcal{X}$ be a diffeomorphism (Definition A.6.22). Then by the change of variable formula (Equation (A.42)) for $x \in \mathcal{X}$ and $z \in \mathcal{Z}$ we have

$$f_{\mathrm{X}}(x) = f_{\mathrm{Z}}\left(\psi^{-1}(x)\right) | \det \mathcal{J}_{\psi^{-1}}(x) | \tag{4.23}$$

for probability density functions $f_{\mathrm{X}}, f_{\mathrm{Z}}$. Then ψ is called a *normalizing flow.*

A central idea of normalizing flows is that they can be composed sequentially to compose multilayer networks, acting as *universal density estimators* for probability distributions [Pap+21]. Furthermore, normalizing flows are estimated using maximum likelihood methods, firmly cementing them as predominantly statistical models.

We note that affine coupling layers are only one type of transformation used in normalizing flows, and others can be considered. In this thesis, we propose the use of our derived variant of *conditional additive coupling layers*, applied in some of our proposed models.

Definition 4.3.6 (Conditional Additive Coupling Layer)**.** Let $y \in \mathbb{R}^p$ be defined on a simplex such that $\sum_i y_i = 1$. Let $\theta \in \mathbb{R}^{n \times p}$. Moreover, let $b : \mathbb{R}^p \to \mathbb{R}^n$ be the convex combination given by

$$b(y;\theta)_i = \sum_{j=1}^{p} y_j \theta_{ij}. \tag{4.24}$$

Then the map $g : \mathbb{R}^n \to \mathbb{R}^n$ is given by

$$g(x) = x + b(y;\theta) \tag{4.25}$$

and we call g a *conditional additive coupling layer.*

The purpose of a conditional additive coupling layer is to condition the inputs x on the outputs y of a classification subnetwork. As such, the additive coupling layer acts as a conditional replacement for the bias of a standard feed-forward layer ψ.

Stochastic Sampling Layers

As of yet, the network layers we have discussed are wholly deterministic, and while coupling layers and normalizing flows allows the approximation of transformations from sampled data from a known probability distribution to fit data from an observable space, they do not inherently induce any stochasticity in a network architecture. A rudimentary method for inducing randomness in a network is to apply *stochastic sampling layers*, which we later make use of in *variational autoencoders*.

Definition 4.3.7 (Stochastic Sampling Layer)**.** Let Υ be a parameter space for a parameterized family of probability distributions $\mathcal{P}$. Now let $\psi_s : \mathcal{X} \to \Upsilon$ be a single layer network and let $\mathrm{Y} : \Omega \times \Upsilon \to \mathcal{Y}$ be a random variable such that $\mathrm{Y} \sim \mathbb{P}_\upsilon \hookleftarrow \upsilon$. Then $(\mathcal{Y}, \mathbb{P}_\upsilon)$ is an observable space generated by Y, and the mapping $\psi_s : \mathcal{X} \to \mathcal{Y}$ given by

$$\psi_s(x; \theta, \gamma, \omega) = \mathrm{Y}\big(\omega; \psi_s(x; \theta, \gamma)\big) \tag{4.26}$$

is called a *stochastic sampling layer*.

If we select $\mathcal{P}$ to be families in the location-scale family (Definition A.7.15), then the parameters of layer ψ_s can be trained to output parameters which approximate the base distribution $\mathbb{P}_\upsilon$ by maximizing the ELBO (Equation (4.40)). This method is known as the *reparametrization trick* and was first introduced in the setting of [KW14]. Stochastic sampling layers are commonly employed in *variational autoencoder* models (VAE), which we discuss further in Section 4.4. Note that stochastic sampling layers bear a resemblance to affine coupling layers, which are also often used to approximate location-scale distributions.

Diagonal Layers

In this thesis, we will be applying several forms of what we call *diagonal layers*. These are essentially scaling operators $D : \mathcal{Z}_p \to \mathcal{Z}_q$ acting on an input z_p by

$$\begin{aligned} D(z_p; \theta_p) &= \operatorname{diag}(\theta_p) z_p && (4.27)\\ &= \theta_p \odot z_p && (4.28) \end{aligned}$$

and are usually applied between the application of isometric operators to allow the individual scaling of the outputs in z_q, similarly to the effect singular values have in SVE/SVD. Diagonal layers are thus structurally similar to the general operators which they are designed to complement. E.g., a diagonal layer complementing a seperable patch layer thus takes the form

$$\begin{aligned} D(\bar{Z}, \theta) &= \big[\operatorname{diag}(\theta_r)^* \otimes \operatorname{diag}(\theta_c)\big] \bar{Z} && (4.29)\\ &= \operatorname{diag}(\theta_r)\, \bar{Z} \operatorname{diag}(\theta_c). && (4.30) \end{aligned}$$

While other diagonal layer structures can be considered, particularly in the context of convolutional layers, we will generally only apply diagonal layers in the forms of Equations (4.27) and (4.29) in this thesis.

4.4 Network Architectures and Encoder-Decoder Models

The term *architecture* is used to describe the general structure of the network. As of yet, we have mainly focused on fully connected feed-forward architectures. While these architectures are the fundamental building blocks of neural networks, there are several other classes of networks which are structured to model spatial dependence, sequential processes, topological spaces and graph structures (Definition A.6.12) by structuring neural dependence more effectively given domain-specific information. For the purposes of this thesis, we will mostly consider feed-forward models, and only a few of our proposed models feature a more complex topology.

Autoencoders

Our work mainly focuses on *encoder-decoder* models, the canonical example of which are *autoencoder* (AE) models. The origin of autoencoder models can be traced back to the resurgence of neural network models with backpropigation [RHW86], providing a very interesting application of unsupervised learning designed to generate efficient data representations.

Definition 4.4.1 (Autoencoder)**.** Let $(\mathcal{X}, \mathcal{B})$ be an observable space and let $\mathcal{Z}$ be a latent Hilbert space. Let $\Psi^e : \mathcal{X} \to \mathcal{Z}$ and $\Psi^d : \mathcal{Z} \to \mathcal{X}$ be subnetworks, called the *encoder network* and *decoder network* respectively. Then the compositional network given by

$$\begin{aligned} \Psi_{\text{AE}}(x; \vartheta, \Gamma) &= (\Psi^d \circ \Psi^e)(x; \vartheta, \Gamma) && (4.31) \\ &= \Psi^d\big(\Psi^e(x; \vartheta^e, \Gamma^e); \vartheta^d, \Gamma^d\big) && (4.32) \end{aligned}$$

is called an *autoencoder network.*

In the general autoencoder setting, the encoder network Ψ^e is trained in combination with an adjoint network Ψ^d which acts as a decoder such that the full network yields a reconstruction of the original input x, minimizing the reconstruction objective by

$$\hat{\vartheta} = \arg\min_{\vartheta} C\big(x, \Psi_{\text{AE}}(x; \vartheta, \Gamma)\big). \tag{4.33}$$

The goal of this objective is to have the composition of the encoder and decoder networks approximate the identity map for all $x \in \mathcal{X}$. This may at first glance seem a fruitless endeavour, but note that the identity mapping is performed via an intermediary space $\mathcal{Z}$, called a *latent representation space.* The dimensions of the latent space is often chosen to be of a smaller dimensionality than $\mathcal{X}$. When $\dim(\mathcal{Y}) < \dim(\mathcal{X})$ the network is called an *undercomplete autoencoder*, which are commonly used for learning effective low dimensional representations of a dataset $\mathcal{D}$. Figure 4.6 illustrates the architecture of an undercomplete autoencoder network.

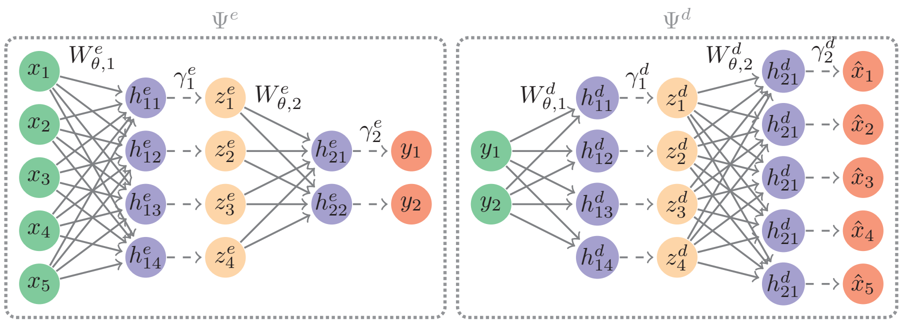

Figure 4.6: An undercomplete autoencoder network.

Conversely, an *overcomplete autoencoder* is an intermediate mapping to an encoded space with higher dimensionality, i.e. $\dim(\mathcal{X}) < \dim(\mathcal{Y})$. With overcomplete autoencoders, there is a susceptibility of all weights approximating identity, so a minor level of noise is usually introduced to discourage identity in the operators of each layer. Noise can also be introduced in order to effectively produce a *denoising autoencoder* [Vin+08] for purposes of general signal restoration.

The link between autoencoders and information theory is well established [YP19] via the concept of informational entropy (Definition A.4.12) – a measure of the expected level of information content contained in a random variable or observable space. In [Vin+10], the authors demonstrate that autoencoders essentially maximize the lower bound on the *mutual information* (Definition A.7.16) between the data and their predicted latent representations. This link to information theory makes autoencoders an excellent tool for inference on deep neural networks, and [YP19] even goes so far as to suggest that the study of autoencoders via the information theoretic route is the most promising tool for opening up the *black box* of deep learning; which we discussed briefly near the end of Section 4.1.

Early investigations into the properties of the special case of linear autoencoder models – i.e., autoencoders where $\gamma = \mathrm{Id}$ for all $\gamma \in \Gamma$ – also demonstrate parallels with principal component analysis [BH89], and has often been informally cited as an equivalence. This equivalence is even suggested in the nonlinear case, and this was shown to be not be the case in [JHG00]. Further investigations [AEE17; Kun+19] provided more evidence of the link between linear autoencoder models and principal component analysis and – by proxy – the SVD. This link to SVE/SVD provides interesting applications for autoencoders in statistical modelling; particularly in the case of inverse problems.

Supervised Autoencoders

Let us now turn our attention to supervised learning with encoder-decoder models. Consider a task where we want to find an approximate bijective map $\Psi : \mathcal{X} \to \mathcal{Y}$ given the observable space $(\mathcal{X} \times \mathcal{Y}, \mathcal{B})$. This setting is central to the work in this thesis, and the question is; can an autoencoder architecture be applied to supervised learning tasks?

Remark 4.4.2. The term supervised autoencoder is a bit of a misnomer. The term *auto* is used to describe the unsupervised learning objective. However, the term 'supervised encoder-decoder' would admittedly would be a more fitting term for these models but is not used in other literature [Gao+15; LPW18; RS08]. Thus we will continue to refer to these models as either supervised autoencoders, autoencoders, or simply by the abbreviation AE – as the learning context is always easily inferred.

Properties of Supervised Autoencoders

While autoencoders see fewer applications in the supervised setting than the unsupervised setting, there are several examples which advocate their practical application [Gao+15; RS08]. A central work on the theory of supervised autoencoders [LPW18] show that the preeminent benefit of supervised AE models is twofold;

- they provide significant improvement in stability and robustness,
- they provide tighter upper bounds on the overall generalization error.

These results suggest the application of supervised autoencoder models as implicit regularization methods for general forward neural network models, which we discuss further in Section 4.5.

The Supervised Objective

While the reconstruction objective provided in Equation (4.33) is natural in an unsupervised setting, it requires modification for supervised learning. A simple addition of a supervised objective to an autoencoder yields

$$\hat{\vartheta} = \arg\min_{\vartheta} C_{\mathcal{X}}\big(x, (\Psi^d \circ \Psi^e)(x; \vartheta)\big) + C_{\mathcal{Y}}\big(y, \Psi^e(x; \vartheta^e)\big). \tag{4.34}$$

However, a more direct objective can be formulated as

$$\hat{\vartheta} = \arg\min_{\vartheta} C_{\mathcal{X}}\big(x, \Psi^d(y; \vartheta^d)\big) + C_{\mathcal{Y}}\big(y, \Psi^e(x; \vartheta^e)\big). \tag{4.35}$$

The difference is that Equation (4.34) compares a reconstruction through the entire network, while Equation (4.35) splits these objectives. This turns out to be an important distinction, as Equation (4.34) can encourage a model to find an inverse for its own mapping, rather than the inverse of the objective, precisely due to the autoencoder models tendency to maximize the lower bound on mutual information [Vin+10]. Any loss of information in the mapping will therefore be minimized, even if this increases the loss in the supervised objective.

The stability and conditioning of the inverse problem can influence the training of a supervised autoencoder. While the condition number quantifies the general ill-conditioning, it does not say anything about which computational direction is the most sensitive to perturbation. To quantify this, we define the *sensitivity imbalance* for an invertible compact linear operator, and show that this has a point of *sensitivity equilibrium.*

Definition 4.4.3 (Sensitivity Imbalance)**.** Let W be an invertible compact linear operator on a Hilbert space with rank n. Let k_1, k_n be the Lipschitz constants of W and $W^{\text{-}1}$ respectively. Then

$$\overset{\rightleftarrows}{\kappa}(W) = \begin{cases} k_1 k_n - 1, & \text{if } k_1 k_n \leq 1; \\ 1 - \frac{1}{k_1 k_n}, & \text{otherwise.} \end{cases} \tag{4.36}$$

is called the *sensitivity imbalance* of a compact linear operator.

Proposition 4.4.4 (Sensitivity Equilibrium)**.** *Let W be an invertible compact linear operator on a Hilbert space with rank n. Then for $\overset{\rightleftarrows}{\kappa}(W) = 0$, the condition number $\kappa(W)$ has equal contribution from W and $W^{\text{-}1}$. Furthermore, $\overset{\rightleftarrows}{\kappa}(W) \in (-1, 1)$ quantifies the relationship between the contribution to the condition number from the forward and inverse operators, with the property $\overset{\rightleftarrows}{\kappa}(W) = -\overset{\rightleftarrows}{\kappa}(W^{\text{-}1})$.*

Proof. The Lipshitz constant of W and $W^{\text{-}1}$ is given by the largest and smallest singular values, so $k_1 = \varsigma_1$ and $k_n = \varsigma_n$. From Equation (3.42) we know that the condition number of a compact linear operator is given by $\kappa(W) = \varsigma_1/\varsigma_n$. Then clearly, the contributions of the terms are equal when $\varsigma_1 = 1/\varsigma_n$. By definition, $\varsigma_1 > \varsigma_n > 0$. Letting $k_1 \to \infty$ and $k_n \to \infty$ shows that $\overset{\rightleftarrows}{\kappa}(W) \in (-1, 1)$. Lastly, we note that $\overset{\rightleftarrows}{\kappa}(W^{\text{-}1})$ will yield a reciprocal relation, which will flip the sign of the expression. ■

The sensitivity imbalance can in principle be extended to hold for any mapping, however we find it most instructive to relate the concept to linear operators in terms of the singular value expansion. Note that the sensitivity imbalance does not directly relate to the actual condition number. Rather, it quantifies some notion of "skewness" in any ill-conditioning between a forward and inverse operator. As an example, consider the operator given by a one-dimensional Gaussian discrete convolution kernel $[1/4, 1/2, 1/4]$. Constructing a 5×5 circulant Toeplitz convolution matrix yields the singular values $\varsigma_1 = 1$, $\varsigma_n = 0.096$, with condition number 10.472. Computing the sensitivity imbalance gives $\overset{\rightleftarrows}{\kappa}(W) = -0.904$ which tells us that the Lipschitz constant of the inverse operator will be the greatest source of instability for this problem. Taking the inverse subsequently yields $\overset{\rightleftarrows}{\kappa}(W^{\text{-}1}) = 0.904$. The condition number is the same, however the sign of the sensitivity imbalance has reversed.

If we now consider this task in the context of a supervised AE model, and let the forward operator be modelled by the encoder, we know that the encoder will be less sensitive to perturbations, and this extends to the training error imposed by the supervised objective $C_{\mathcal{Y}}$. This means the task of approximating the forward operator is consequently "easier" than approximating the inverse operator, as the reconstruction error from $C_{\mathcal{X}}$ can be significantly magnified in the decoder. During training, this can result in higher losses in the reconstruction objective for deblurring than blurring, which means that the gradients will be dominated by the losses from the deblurring task.

Variational Bayesian Inference and Variational Autoencoders

In a classic work [KW14], the authors introduce methods which allow us to turn an autoencoder into a generative model (Definition 3.2.5). The motivation goes as follows; if we are in possession of some known approximation of the distribution over the latent space $\mathcal{Z}$, then an autoencoder can be considered generative. By considering output from the encoder as latent random variables Z, the problem can be restated as a *variational Bayesian* problem, where we have

- a set of observable data $\{x_i : x_i \in \mathcal{X}, i = 1, \dots, n\}$,
- a set of latent variables $\{z_i : z_i \in \mathcal{Z}, i = 1, \dots, n\}$,
- a posterior of interest $f_{\mathrm{Z|X}} = f_{\mathrm{X|Z}} f_{\mathrm{Z}} / f_{\mathrm{X}}$.

The canonical problem when modelling using Bayesian inference methods is that the density $f_{\mathrm{X}} = \int f_{\mathrm{X|Z}} \cdot f_{\mathrm{Z}}$ is often intractable. In a variational Bayesian setting, this is solved by approximating $f_{\mathrm{Z|X}}$ with a tractable density g_{Z}, such that $f_{\mathrm{Z|X}} \approx g_{\mathrm{Z}}$. From Jensen's inequality (Theorem A.7.11) we have that

$$\begin{aligned}
\log f_{\mathrm{X}}(x) &= \log\left(\int f_{\mathrm{XZ}}(x,z)\ dz\right) && (4.37)\\
&= \log\left(\int g_{\mathrm{Z}}(z) \frac{f_{\mathrm{XZ}}(x,z)}{g_{\mathrm{Z}}(z)}\ dz\right) && (4.38)\\
&\geq \mathbb{E}_g\left[\log\left(\frac{f_{\mathrm{XZ}}(x,z)}{g_{\mathrm{Z}}(z)}\right)\right] && (4.39)\\
&= \mathbb{E}_g\left[\log f_{\mathrm{XZ}}(x,z)\right] - \mathbb{E}_g[\log g_{\mathrm{Z}}(z)] && (4.40)\\
&= \textsc{elbo}[f_{\mathrm{X}}], && (4.41)
\end{aligned}$$

so by maximising the *evidence lower bound* of X (Definition A.7.17), we minimize the error of the approximation. To this end, we make use of the related *Kullback-Leibler divergence.*

Definition 4.4.5 (Kullback-Leibler divergence)**.** Let $(\mathcal{X}, \mathcal{B})$ be an observable space, and let $\mathbb{P}_f, \mathbb{P}_g$ be probability distributions over $\mathcal{X}$ with probability density functions $f_{\mathrm{X}}, g_{\mathrm{X}}$ respectively. Then $d_{KL} : L^1(\mathcal{X}) \times L^1(\mathcal{X}) \to \mathbb{R}_{\geq 0}$ given by

$$\begin{aligned}
d_{KL}[g_{\mathrm{X}} \| f_{\mathrm{X}}] &= \mathrm{H}_g[f_{\mathrm{X}}] - \mathrm{H}[g_{\mathrm{X}}] && (4.42)\\
&= \mathbb{E}_g[\log g_{\mathrm{X}}(x)] - \mathbb{E}_g[\log f_{\mathrm{X}}(x)] && (4.43)\\
&= \int_{\mathcal{X}} g_{\mathrm{X}}(x) \log\left(\frac{g_{\mathrm{X}}(x)}{f_{\mathrm{X}}(x)}\right)\ dx. && (4.44)
\end{aligned}$$

is called the *Kullback-Leibler divergence* of g_{X} with respect to f_{X}.

At this point, we require the application of the following useful observation.

Observation 4.4.6 (KL-divergence and ELBO)**.** *Let* X, Z *be random variables with pdfs.* $f_{\mathrm{X}}, f_{\mathrm{Z}}$ *respectively. Let* g_{Z} *be an approximation of* $f_{\mathrm{Z}|\mathrm{X}}$*. Then the optimization objective* $\arg\min_{g_{\mathrm{Z}}} d_{KL}[g_{\mathrm{Z}} \| f_{\mathrm{Z}|\mathrm{X}}]$ *is equivalent to* $\arg\max_{f_{\mathrm{X}}} \textsc{elbo}[f_{\mathrm{X}}]$.

Proof. Rewriting $f_{\mathrm{Z}|\mathrm{X}}$ using the Kullback-Leibler divergence yields

$$\begin{aligned} d_{KL}[g_{\mathrm{Z}} \| f_{\mathrm{Z}|\mathrm{X}}] &= \mathbb{E}_g \left[\log \left(\frac{g_{\mathrm{Z}}(z)}{f_{\mathrm{Z}|\mathrm{X}}(z \mid x)} \right) \right] \\ &= \mathbb{E}_g \left[\log \left(\frac{g_{\mathrm{Z}}(z)}{f_{\mathrm{XZ}}(x, z)} \right) \right] + \log f_{\mathrm{X}}(x) \\ &= \log f_{\mathrm{X}}(x) - (\mathbb{E}_g [\log f_{\mathrm{XZ}}(x, z)] - \mathbb{E}_g[\log g_{\mathrm{Z}}(z)]) \\ &= \log f_{\mathrm{X}}(x) - \textsc{elbo}[f_{\mathrm{X}}], \end{aligned}$$

so minimizing $d_{KL}[g_{\mathrm{Z}} \| f_{\mathrm{Z}|\mathrm{X}}]$ is equivalent to maximizing $\textsc{elbo}[f_{\mathrm{X}}]$. ■

To apply this in an autencoder setting, we can rewrite Equation (4.40) to get the general form of the canonical VAE objective

$$\begin{aligned} \textsc{elbo}[g_{\mathrm{Z}}] &= \mathbb{E}_g[\log f_{\mathrm{XZ}}(x, z)] - \mathbb{E}_g[\log g_{\mathrm{Z}}(z)] && (4.45) \\ &= \mathbb{E}_g[\log f_{\mathrm{X}|\mathrm{Z}}(x \mid z)] - (\mathbb{E}_g[\log g_{\mathrm{Z}}(z)] - \mathbb{E}_g[\log f_{\mathrm{Z}}(z)]) && (4.46) \\ &= \mathbb{E}_g[\log f_{\mathrm{X}|\mathrm{Z}}(x \mid z)] - d_{KL}[g_{\mathrm{Z}} \| f_{\mathrm{Z}}]. && (4.47) \end{aligned}$$

Letting the decoder network be given by $\Psi^d \sim f_{\mathrm{X}|\mathrm{Z}}$ and the encoder network given by $\Psi^e \sim g_{\mathrm{Z}|\mathrm{X}}$, we conceivably have an autoencoder representation of a bidirectional graphical model. Thus Equation (4.47) can be interpreted as an objective function where the first term is simply a form of reconstruction loss and where $d_{KL}[g_{\mathrm{Z}} \| f_{\mathrm{Z}}]$ acts as a regularization term for the divergence between the encoder and decoder.

This interpretation requires that the latent vectors are random variables; and is where we apply the stochastic sampling layer from Definition 4.3.7 to construct the full definition of a *variational autoencoder* [KW14].

Definition 4.4.7 (Variational Autoencoder)**.** Let $\Psi_{\textsc{vae}}$ be an autoencoder model, where the final layer of the encoder is a stochastic sampling layer with parameter space Υ. If the loss function C_J includes a regularization term $J(\upsilon \in \Upsilon) = d_{KL}[g_{\mathrm{Z}|\upsilon} \| f_{\mathrm{Z}}]$ we say that $\Psi_{\textsc{vae}}$ is a *variational autoencoder*.

Instead of letting the autoencoder produce the latent variables directly, we let the output of the stochastic sampling layer be the parameters $\upsilon = (m, s)$ of a multivariate distribution in the location-scale family (Definition A.7.15). We then sample a vector of iid. random variables $\{\zeta_i \sim \mathbb{P}_\upsilon : i = 1, \ldots, k\}$ and let

$$z = s\zeta + m \sim \mathbb{P}_\upsilon, \tag{4.48}$$

As this is differentiable, the gradients from the input $x \in \mathcal{X}$ are propagated throughout the network, minimizing the reconstruction error. This is what is referred to as the *reparametrization trick* in VAE models. Note that the reconstruction objective given in Equation (4.47) is not directly required for a VAE model, however we consider it as the canonical objective for modelling with variational autoencoder models.

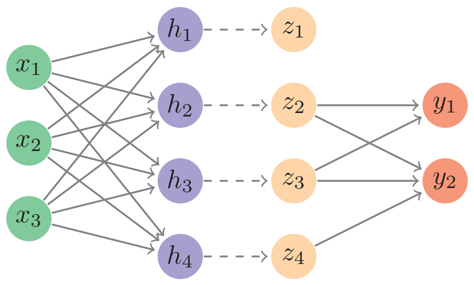

Figure 4.7: Visualization of dropout. Note that z_1 has had all connections to the output layer dropped, whereas z_4 has had only a single connection to y_1 dropped, demonstrating the difference between the two approaches.

4.5 Regularization, Optimization, and Learning Dynamics

In this section, we discuss regularization and optimization methods available for modelling with neural networks. While general L^p regularization methods (Definition 3.5.4) can be applied to network weights, the direct implementation of these methods are generally applied through the optimization algorithms. As an example, ADAM [KB17; LH17] features an optional weight decay term, which is equivalent to Tikhonov regularization. In addition to these methods, neural networks often employ specific regularization methods not seen in other contexts.

Dropout

Dropout is one of the most commonly applied methods of regularization in neural networks. It is a stochastic method applied to hidden neurons during training which give a probability p of transmitting their value [Hin+12; Sri+14]. This can either be done on a single connection between layers or for all connections from a hidden neuron, both illustrated in Figure 4.7. This imposes implicit regularization, making a network less dependent on individual weights or neurons when computing an output. The idea is to utilize similar properties to neural plasticity to distribute the computation more evenly across the weights in a layer. The method has connections to the method of *dilution* in mean-field theory, and has been shown to have an interesting link to Bayesian inference [GG16].

Batch Normalization

Another popular method to promote implicit regularization in neural networks is the *normalization layer*, first introduced in [IS15]. The motivation for this method is derived from standardization of data and the concept of *internal covariate shift*, a loosely defined phenomenon where the distribution of intermediate hidden layers display arbitrary shift and scaling due to randomness of the weight initialization process.

The proposed solution is to perform online calculations of empirical means and standard deviations over each batch B_i in intermediate layers and normalize the outputs accordingly by

$$\mu_{jk}^{(i)} = \frac{1}{\#(B_i)} \sum_{z_k \in \psi^{[j]}(B_i)} z_k \tag{4.49}$$

$$\sigma_{jk}^{(i)} = \frac{1}{\sqrt{\#(B_i)}} \sqrt{\sum_{z_k \in \psi^{[j]}(B_i)} \left(z_k - \mu_{jk}^{(i)}\right)^2}. \tag{4.50}$$

The layer can either be computed for each batch or extended to derive global parametrizations for the empirical distribution of the given layer. The normalization process is commonly followed by a learned one-dimensional affine transformation to compute the final output.

While normalization layers have been shown to be effective, the effect on internal covariate shift is not as clear, and the reason behind the effectiveness of normalization layers is more or less unknown. In [San+18] the authors argue this point and discuss one possible hypothesis for the effectiveness of normalization layers as a form of regularization on the solution space. By applying random Gaussian noise post-normalization, where a nonzero mean and nonunit variance are sampled randomly for each batch, the authors effectively negate any positive effect the normalization process could have on the internal covariate shift. However, the empirical results show that batch normalization methods still produce better results than a non-normalized model. The centering and unit scaling of batch normalization do however, affect the Lipschitz constant of the network – if the batch normalization is non-parametrized – effectively smoothing the solution space. Furthermore, similar effects can be observed for variants using L^p normalization without distributional stability, leading the authors to conclude that this is the main reason for the effectiveness of batch normalization.

Parseval Regularization

In [Cis+17] the authors discuss network stability and robustness in the context of what is known as *adverserial attacks* – a concept we discuss further in Section 5.1. The main idea is that each layer can be enforced to be approximately isometric; such that each layer has Lipschitz constant $k \approx 1$ via a regularization constraint known as *Parseval regularization.*

Definition 4.5.1. Let ϑ be a parametrization over the weights of a neural network Ψ. Let J be a functional over $\vartheta \in \Theta$ given by

$$J(\vartheta) = \sum_{\theta \in \vartheta} \frac{\beta}{2} \|W_\theta^* W_\theta - I\|_2^2. \tag{4.51}$$

Then J is called a *Parseval regularization term.*

This constraint ensures that the weight of each layer is a *Parseval-tight frame* [KC08] – i.e. the network weights are approximately semi-orthogonal. The Parseval regularization term enforces weights to be at least injective and the spectral norm to be approximately one. Networks which are Parseval optimal and feature activation functions with Lipschitz constant $k = 1$ are called *Lipschitz-1 networks.* [ALG18] features a thorough exposition on the subject of Lipschitz-1 networks and special activation functions for enforcing their properties.

Learning Dynamics

The first implementations of deep models showed that simply increasing the model depth had certain disadvantages, as the initial state of the weights combined with the choice of activation function could limit the rate at which the gradient of the learning objective could effectively propagate through the entire network. If the gradients were sufficiently dampened or nullified, this resulted in models that never converged, limiting the effectiveness of deep models [BSF94]. The study of *optimal learning dynamics* in deep learning focuses on methods to improve gradient propagation in deep neural network models. Much of the work on learning dynamics has naturally been focused on the backpropigation algorithm (Equation (4.15)), as the canonical method for optimization in modelling with deep neural networks.

Unsupervised Pretraining

One way to overcome the issue of learning dynamics in deep models was proposed in [Ben+07]. By sequentially training each layer with encoder-decoder structures in order to allow the gradients of the objective to propagate through the entire model, the network can effectively be *pre-trained* to overcome these limitations. This process is dubbed *greedy layer-wise unsupervised pretraining.*

Definition 4.5.2 (Greedy Layerwise Unsupervised Pretraining)**.** Let $\Psi(x; \vartheta, \Gamma)$ be an L-layer neural network, and let $\psi_i(z_{i\text{-}1}; \theta_i, \gamma_i)$ be the ith layer of Ψ. Let $\psi^d(z_i; \theta_i^d, \gamma_i^d)$ be a decoder layer with optimal weights given by

$$\theta_i^d = \arg\min_{\theta} \|(\psi_i^d \circ \psi_i)(z_{i\text{-}1}) - z_{i\text{-}1}\|. \tag{4.52}$$

The process of sequentially estimating θ_i for $i = 1, \ldots, L-1$ is referred to as *greedy layerwise unsupervised pretraining.*

The method utilizes auxiliary decoder structures to generate effective layer-wise representations by enforcing minimal reconstruction errors for each appended layer in the network, separately minimizing information loss for consecutive layers. In [GBC16, p.528], the authors claim that this idea instigated the renaissance of neural networks, by allowing for optimal initialization of network weights – thus allowing for deeper models. Furthermore, [Erh+10] show that unsupervised pretraining acts as an implicit regularizer by adjusting the weights to act as a more appropriate "prior" (sic.) for full signal propagation through the network, improving the learning dynamics. In Section 4.4, we already mentioned that supervised autoencoders have been shown to act as implicit regularization in its own right [LPW18], which corroborates this result.

Dynamical Isometry

Later work on optimal learning dynamics introduced *dynamical isometry* - a property where the singular values of the Jacobian of a network are approximately one. This property ensures that the network avoids vanishing and exploding gradients by ensuring a well-conditioned Jacobian. In [PSG17] the authors show this is achievable by initializing the network with orthogonal weights and using sigmoidal activation functions, and in [Xia+18] the idea is expanded to convolutional networks with similar orthogonality constraints in initialization.

PART II

Methodology

CHAPTER 5

Inverse Problems and Invertible Neural Networks

Thus far, we have outlined relevant theoretical results and definitions for inverse problems, statistical modelling, and neural networks. In this chapter, we discuss the error sources and stability issues we need to be aware of when modelling inverse problems with neural networks. Furthermore, we introduce an approach to the modelling of inverse problems via *invertible neural networks* using normalizing flows and discuss the benefits and weaknesses of this approach.

5.1 Uncertainty and Instability in Neural Networks

We begin by specifying the general problem; recall that from Section 2.1 we considered the general form of an forward problem of model identification by

$$y = \Phi(x; \theta) + \varepsilon, \tag{5.1}$$

and note that this coincides with the notion of predictive modelling from Equation (3.12) where we consider the system Φ as unknown. In Section 3.1 we introduced observable spaces (Proposition 3.1.1) on the form $(\mathcal{X} \times \mathcal{Y}, \mathcal{B})$. In the setting of supervised learning we consider a set of observations $\mathcal{D} = (x, y)_{i \in \mathcal{I}}$ which we want to use to approximate the system $\Phi : \mathcal{X} \to \mathcal{Y}$. However, we are instead interested in directly modelling an inverse map $\mathcal{Y} \mapsto \mathcal{X}$, which we consider to be either inherently ill-conditioned or ill-posed (Definition 2.1.1).

In constructing a learning model (Definition 3.2.2) for inverse problems, it is advisable to consider what types of errors we are likely to encounter. The sources of uncertainty can be classified into two main categories [DD09; HW21].

Definition 5.1.1 (Uncertainty in Learning Models)**.** Let $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$ be a learning model. We consider the *aleatoric* uncertainty to be any uncertainty independent of the choice of $\mathscr{H}$. Consequently, we consider the *epistemic* uncertainty to be any uncertainty which is fundamentally reducible by the choice of learning model.

While our main focus will be on epistemic uncertainty, it is important to consider the fact that any discriminative models do not directly model the aleatoric uncertainty of an inverse problem inherent in the noise component ε. Most importantly, heteroscedastic uncertainty is especially problematic for non-probabilistic modelling techniques [KG17].

Observation 5.1.2 (Epistemic Errors in Inverse Problems)**.** *A learning model* $(\mathcal{X} \times \mathcal{Y}, \mathscr{H})$ *for an inverse problem of model identification* $x = \Phi^{-1}(y; \theta)$ *will have the following main sources of* epistemic *errors [KO01];*

(i) *(Misspecification) the model* Φ *(or* $\mathbb{P}$*) is not an element of* $\mathscr{H}$*,*

(ii) *(Algorithmic) the estimation is not sufficient to find* $\theta \mapsto \Phi$*,*

(iii) *(Ambiguity) the parametrization* θ *is not uniquely identifiable,*

(iv) *(Interpolation) the true model* Φ *is not determinable given* $\mathcal{D}$*.*

Modelling inverse problems with neural networks is an attractive prospect, as we can effectively ignore epistemic errors from misspecification (Property 5.1.2.i) due to the properties of universal approximation (Theorem 4.1.6). As a result of this flexibility, we do however increase the risk of occurrence of other error sources, particularly in terms of ambiguity (Property 5.1.2.iii). Errors of ambiguity (Property 5.1.2.iii) and interpolation (Property 5.1.2.iv) are however more significant, and relates directly to issues of ill-posedness. To effectively generalize a problem, any learning model is required to effectively construct a generalized map between input and output spaces, requiring interpolation or extrapolation for generalization of unseen observations. Extrapolation is naturally more challenging as we expect poor approximations of outlier points sufficiently distant from any observation in the training data. However, [Sze+14] showed that tiny perturbations of inputs can lead to significant errors in the solution space. This phenomenon is usually observed via so-called *adverserial attacks* [GSS15] – malign estimation of particularly destabilizing perturbations by some hypothetical adversary. Moreover, these perturbations generalize over different models trained over the same data.

The issues of adverserial perturbations relate to issues of instability and ill-conditioning of the model. [Ant+20] discusses fundamental differences in classification tasks and image reconstruction tasks, especially in the realm of medical imaging. These concepts relate directly to the fact that deterministic discriminative models imply hard classification boundaries for the underlying discrete decision problem. If the generated solution space is sufficiently nonsmooth, any such hard decision boundaries can effectively be exploited to find sufficiently steep slopes in the solution space that will lead to misclassification. A more fundamental issue is that deep neural networks have similar behaviour when trained on reconstruction tasks for which there exist stable algorithms for image reconstruction - e.g. magnetic resonance imaging and computed tomography. The claim in [Ant+20] is that instabilities in neural networks are fundamentally algorithmic in nature (Property 5.1.2.ii), as stable algorithms for solving inverse problems in compressed sensing exist and are readily available. While stochastic gradient descent methods have been shown to be stable for neural networks [Bot91], this has certain caveats. In particular, certain works have shown limitations of gradient descent methods [SLH20], where the authors highlight the deficiencies of backpropigation for neural networks where persistent weight noise is introduced. In [Com+18], the authors provide an in-depth study of the limitations on network convergence in terms of optimal learning dynamics, which we discussed in Section 4.5.

The rifeness of such instabilities are concluded to be a predominantly epistemic issue related to the model itself, particularily in the realm of deterministic discriminative modelling, which begs the question; *are stability and accuracy at odds*? The work in [ACH21] elaborates on this precise issue, and [Tsi+19] suggests that this tradeoff "probably exists" (sic.) but highlights some unexpected benefits by showing that adverserial robustness can increase the interpretability of neural networks via gradient extraction and visualization. The idea is that models trained with remedial methods against adverserial attacks exhibit some level of invariance to tiny perturbations and requires the model to embed high-level features as a result, more closely emulating human cognition.

While these results can be interpreted as uncertainty due to interpolation of data, there are results that indicate that this is not the cause of the instability. In [OWB18], the authors show that modern neural networks can generalize well even with small amounts of data. In their experiments, they train deep neural network models on a series of comparatively simple datasets [DG17]. They then apply linear programming to decompose the network into smaller subnetworks and proceed to show that that these are uncorrelated ensembles, and demonstrate that the composed networks display similarities to random forests.

Rather surprisingly, [Nak+19] show that increasing the number of observations in a dataset can – in certain cases – increase the error of the final model for sufficiently deep models, suggesting that there is some notion of *efficient model complexity* which determines the performance of deep neural networks. These results highlight the fact that lack of data is not necessarily a direct hindrance for training neural networks. In fact, most modern neural networks have a much larger dimensionality of their parameter space than the number of observations used to estimate their parameters. The instability issues of neural networks thus seem to be most closely related to ambiguity in the parameter space combined with unmanaged aletoric uncertainty from ill-posedness of the original inverse problem.

Furthermore, there is a connection between adverserial attacks and inverse problems. An adverserial perturbation δ can be interpreted as the "worst-case" of the inherent noise component ε. Therefore, adverserial robustness is in a sense a stricter requirement of robustness than managing the aleatoric uncertainty of inverse problems. If there exists a perturbation δ in a ball $\mathbb{B}(y, r)$ around an observation which will severely offset the reconstruction of x, then the model is considered unstable, even if this perturbation δ is far out in the tail of the actual distribution of the noise component ϵ. This raises the question; what distribution can be assigned to the adverserial perturbation δ? If these are selected from the ball $\mathbb{B}(y, r)$ it seems reasonable to assign a uniform distribution over the ball. However, it is more likely that more of the unstable perturbations are found on the perimeter of this ball than in the interior, which yields an antimodal quality to the distribution of adverserial perturbations. Is it then reasonable to assume that this is related to the uncertainty of the distribution of ε?

In this thesis, we take the approach of considering adverserial perturbations as worst case perturbations similarly to the definition of the relative condition number (Definition 3.5.2), which factors in the magnitude of the adverserial perturbation. We discuss this further in Section 7.1 in relation to our experimental methodology.

5.2 Normalizing Flows and Invertible Neural Networks

In the previous section, we discussed the sources of uncertainty and instability in neural networks. In this section, we motivate the need for probabilistic methods and invertible structures to address the inherent aleatoric error of inverse problems. A straightforward approach to modelling an inverse problem using a neural network is to estimate $\Psi : \mathcal{Y} \to \mathcal{X}$ by minimizing a loss function C (Equation (3.30)) with respect to the network parameters $\vartheta \in \Theta$ according to some measure of empirical risk given by

$$\arg\min_{\vartheta} C\left(x, \Psi(y; \vartheta, \Gamma)\right), \quad \forall\ (x, y) \in \mathcal{D} \tag{5.2}$$

with the goal of minimizing $\|x - \Psi(y; \vartheta, \Gamma)\|$. This is the standard discriminative modelling approach (Definition 3.2.3). From the discussion in Chapter 2 we know that if the problem is ill-posed or ill-conditioned, such a method can lead to suboptimal reconstructions. In Sections 3.5 and 4.5 we discussed how this can be remedied by replacing C with a regularized loss function C_J.

Alternatively, we can introduce probabilistic modelling in a neural network context, either by conditional models (Definition 3.2.4) or by modelling a joint distribution with a generative model (Definition 3.2.5). In [Yin18] the authors argue that generative modelling is an effective countermeasure to adverserial attacks, suggesting that at least some of the stability issues of neural network models are caused by their discriminative nature. In Definition 4.4.7 we introduced the variational autoencoder as an important example of generative modelling in neural networks. These models are predominantly unsupervised, and the distributions are approximate estimations at best, although hybrid methods of variational autoencoders and discriminative approaches from generative adverserial networks have been shown to be moderately successful [Lar+16; Tol+19]. *Generative adverserial networks* (GANs) [Goo+14] provide an exotic approach to generative modelling, but generally does not provide an underlying probability distribution for inference, and as such provide less than what we want in terms of uncertainty measures or quantifiable stability. Moreover, as generative modelling is likely to reduce the overall performance, these approaches do not directly address the tradeoff between stability and accuracy discussed in [ACH21; Tsi+19].

Several methods have been proposed to bring uncertainty measures into modelling with deep neural networks. The earliest of these approaches are *probabilistic neural networks* [Spe90], a type of flexible kernel density estimation [HTF09, Section 6.6] using the Bayes optimal decision rule for classification. Another approach was proposed in [Mac95], which formalized Bayesian neural networks. In [Blu+15] the authors propose defining explicit priors and quantifying uncertainty over the weights over the entire network while training via Bayesian backpropigation. Other work has used dropout (Section 4.5) to study neural networks in the context of approximate Bayesian inference [GG16], while [KG17] directly address the problems of heteroscedastic aleatoric uncertainty in neural networks.

Currently, the most prolific probabilistic neural network models are normalizing flows (Definition 4.3.5), which allow for flexible density estimation with tractable distributions. These models have been successfuly applied in a variety of applications [DKB15; DSB16; KD18; RM16]. Only recently, [Tes+20] showed that neural networks with affine coupling layers are *universal diffeomorphism approximators* (Definition A.6.22), issuing elevated credence to approaching *invertible neural network* architectures.

Definition 5.2.1 (Invertible Neural Network)**.** Let $(\mathcal{X} \times \mathcal{Y})$ be an observable space, and let $\Psi : \mathcal{X} \to \mathcal{Y}$ be a neural network. If Ψ is a diffeomorphism, we say that Ψ is an *invertible neural network.*

While coupling layers and normalizing flows can be used to construct invertible neural networks, several other models have been proposed [JSO18; PW19]. There are several benefits of modelling with an invertible neural network. In the context of inverse problems on an observable space $(\mathcal{X} \times \mathcal{Y}, \mathcal{B})$, invertible neural networks can be able to construct bijective maps $\mathcal{X} \to \mathcal{Y}$ where no such map is clearly defined given the forward problem. This naturally extends to representation learning and compression. Furthermore, suppose we have a invertible model $\Psi : \mathcal{X} \to \mathcal{Y}$ with some notion of sufficient accuracy. If we now consider a third domain $\mathcal{Z}$ which is related to $\mathcal{X}$ through observations from $(\mathcal{X} \times \mathcal{Z}, \mathcal{B})$, we can construct a model $\hat{\Psi} : \mathcal{X} \to \mathcal{Z}$. If both models are bijective as well as provide sufficiently high quality reconstructions, we can then consider the possibility of composing models to construct maps on the form $\hat{\Psi}^{-1} \circ \Psi : \mathcal{Z} \to \mathcal{Y}$. In other words, invertible networks can be effective in the domain of inductive transfer learning [PT97].

In [Ard+18] the authors present methods of approaching inverse modelling of inverse problems with underdetermination by constructing such invertible network models by using normalizing flows (RealNVP) [DSB16]. Instead of approaching inverse problems by an approach outlined in Equation (5.2) the idea is to model a bijective map $\mathcal{X} \mapsto \mathcal{Y} \times \mathcal{Z}$ where $\mathcal{Y}$ is augmented by a latent space $\mathcal{Z}$ with a tractable distribution, such that $\dim(\mathcal{X}) = \dim(\mathcal{Y} \times \mathcal{Z})$. In particular, we have a situation where we want to estimate $\mathbb{P}_{\mathrm{X|Y}}$ by $\mathbb{P}_{\mathrm{X|Y,Z}}$ using a generative model of the form

$$x = \Psi(y, z; \theta) \tag{5.3}$$

$$z \sim \mathbb{P}_{\mathrm{Z}}. \tag{5.4}$$

where we usually let $\mathbb{P}_{\mathrm{Z}} = \mathcal{N}(0, I)$, acting as a prior. The result is a full posterior for $\mathrm{X} = x \in \mathcal{X}$ conditioned on $\mathrm{Y} = y \in \mathcal{Y}$ and $\mathrm{Z} = z \in \mathcal{Z}$.

Additionally, the authors show that such models have particularly strong properties for applications to inverse problems of model identification. Firstly, the authors state that the predictive power of invertible neural networks does not seem to be detrimentally affected by imposing structures to enable bijectivity. Secondly, they show that these models compare favorably with approximate Bayesian inference models. Thirdly, they show that these models can be trained bi-directionally similar to the objective of a semisupervised variational autoencoder (Equation (4.34)) using unsupervised learning on the augmented latent variable Z combined with the supervised loss on Y. The authors state that this asymptotically yields a true posterior for X.

While these results are both interesting and promising, there are two issues with this approach:

- Firstly, the issue with applying normalizing flows in invertible neural networks is that they can be computationally heavy, as they require a feed-forward neural network for each coupling in the full model. For high dimensional data such as images and audio signals, the computational costs of these models can be infeasible.
- Secondly, a normalizing flow differs significantly from standard neural network architectures. As such, they require specifically modified network structures and layers, and much of the established theory on feed-forward models can not be applied directly.

The rest of our work in this thesis centers on adapting invertible neural networks and the methods outlined in [Ard+18] to classical feedforward structures, which we motivate and discuss in the next chapter.

CHAPTER 6

Invertible and Pseudo-Invertible Encoders

In this chapter, we outline our main contributions and present novel neural network architectures based on the autoencoder (Definition 4.4.1) which uses implicit and explicit orthogonal and unitary constraints on network weights to encourage invertible properties. Furthermore, we investigate how the network architecture enforces stability similarly to regularization. Lastly, we examine how these models can be used for supervised and generative tasks by applying the variational Bayesian approaches introduced in Definition 4.4.7, and use this to define generative models for ill-posed inverse problems.

6.1 Motivation

In Section 4.4 we introduced autoencoders as models capable of finding useful latent representations of high-dimensional data. Autoencoders and general encoder-decoder models feature prominently in a wide variety of neural network architectures, including – but not limited to – representation learning and embedding [BDV01], sparse coding and dictionary learning [SPH07], and attention layers [BCB15]. In addition, encoder-decoder structures have one main property in common with invertible neural networks – the reconstruction of elements from an observable space. Due to their superficial similarity, one could argue that the invertible neural networks are *supervised autoencoders* with guarantees of bijectivity. The applicability of these models in a variety of settings makes them good candidates for constructing invertible networks.

On the other hand, autoencoders use different parametrizations of weights for the encoder and decoder networks, and are thus only approximations of diffeomorphisms. Theoretically, this implies that the models can be prone to *hallucinatory effects* [Kno+20], where the network encodes using an overfitted eigenspace over the observations [TP91; Zha+16]. These issues can be addressed to promote stronger stability in the network.

Our main idea is to endow the encoder network of an encoder-decoder model with properties to promote a diffemorphic map in a supervised learning setting to effectively make them invertible neural networks. To achieve this, we propose to enforce a joint parametrization for the encoder and decoder, effectively constructing *invertible encoder networks.* To this end, we summarize some important observations from the theoretical exposure of our thesis.

Observation 6.1.1 (Motivation and Properties for Invertible Encoder Networks)**.**

(i) *Linear autoencoders share properties with the* SVE/SVD *and pseudoinverse operators [BH89], which are fundamental tools for with inverse problems (Section 2.3).*

(ii) *Supervised autoencoders (Section 4.4) encourage implicit regularization [LPW18] and tight upper bounds on generalization error.*

(iii) *Optimal learning dynamics can be enforced by orthogonal initialization (Section 4.5) or supervised pretraining using encoder-decoders (Definition 4.5.2).*

(iv) *Parseval regularization (Definition 4.5.1) is a remedial method for adverserial attacks [Cis+17] which improves the general stability of neural networks.*

(v) *Inhomogeneous integral equations (Definition 2.2.5) and resolvent operators (Definition 2.3.9) have strong invertible properties, and can be solved by the application of Liouville-Neumann series (Corollary 2.3.12). Furthermore, these operators share similarities with single layer residual blocks (Definition 4.3.1).*

(vi) *Convolution layers (Definition 4.3.2) are fundamentally constructed using convolution operators (Definition 2.2.6) and the linearity of the Fourier operator in Hilbert spaces (Theorem A.5.2) can be applied to compute the parameters of the kernel.*

The properties of Observation 6.1.1 motivate our method and results, which we derive in this section. As a first step toward invertible encoder models, we begin by introducing the concept of an *ideal autoencoder*.

Definition 6.1.2 (Ideal Autoencoder)**.** Let $\Psi_{\text{IE}}(x; \vartheta, \Gamma)$ be an L-layer autoencoder network with encoder network Ψ^e and decoder network Ψ^d. If the weights ϑ are such that for all $x \in \mathcal{X}$ we have

$$(\Psi^e \circ \Psi^d)(x; \vartheta, \Gamma) = x \tag{6.1}$$

we say Ψ_{IE} is an *ideal autoencoder* network.

The above definition simply states that the only requirement of an ideal autoencoder is to reconstruct the input data perfectly. We also note that in the linear case, an ideal autoencoder can inherit the similiarity to SVE/SVD (Property 6.1.1.i). The next result constructively shows the existence of ideal autoencoders given a particular choice of activation function.

Lemma 6.1.3 (Existence of Nontrivial Ideal Autoencoder)**.** *There exists a family of ideal autoencoders with joint parametrization of the encoder and decoder networks, such that $\vartheta^e = \vartheta^d$, and where $W_\theta \neq I$ for all compact linear operators W_θ occurring in either Ψ^e, Ψ^d.*

Proof. The proof is constructive. Let $\Psi_{\text{IE}}(x; \vartheta, \Gamma)$ be an L-layer autoencoder network with encoder network Ψ^e and decoder network Ψ^d. Let each $\gamma_i \in \Gamma$ for $i = 1, \ldots, L$ be a $\mathcal{C}^1$ piecewise diffeomorphism - i.e. bijective with $\gamma_i, \gamma_i^{-1} \in \mathcal{C}^1$ with piecewise continuous derivatives - and let the parameters for each weight $\theta_i \mapsto W_{\theta_i}$ be such that $W^*_{\theta_i} W_{\theta_i} = I$ so each parametrized compact operator is either unitary or semi-unitary. Furthermore, let the structure of the layers be such that

$$\psi_i^e(z_{i\text{-}1}; \theta_i, \gamma_i) = \gamma_i(W_{\theta_i} z_{i\text{-}1}) \tag{6.2}$$
$$\psi_i^d(z_i; \theta_i, \gamma_i^{-1}) = W^*_{\theta_i} \gamma_i^{-1}(z_i). \tag{6.3}$$

Then for all $i = 1, \ldots, L$ we have

$$\begin{aligned}(\psi_i^e \circ \psi_i^d)(z_{i\text{-}1}) &= W^*_{\theta_i} \gamma_i^{-1}\big(\gamma_i(W_{\theta_i} z_{i\text{-}1})\big) && (6.4)\\ &= W^*_{\theta_i} W_{\theta_i} z_{i\text{-}1} && (6.5)\\ &= z_{i\text{-}1}. && (6.6)\end{aligned}$$

Then Ψ_{IE} is an ideal autoencoder for any joint parametrization ϑ where $W^*_{\theta_i} = W_{\theta_i}$ for each $i = 1, \ldots, L$. ■

Following Lemma 6.1.3 we note that for $\dim(z_i) \neq \dim(z_{i\pm1})$ – i.e. for any underdetermined and overdetermined layers in the network architecture – we have a rank deficiency in either encoder or decoder computations, which we emphasize is not necessarily invertible in both forward and inverse computations. However, for full rank operators these are invertible.

Definition 6.1.4 (Family of Nontrivial Ideal Autoencoders)**.** Let Ψ_{IE} be a nontrivial ideal autoencoder with joint parametrization as given in the proof of Lemma 6.1.3. Then we call Ψ_{IE} an *invertible unitary encoder.*

While seemingly trivial, this result is useful for connecting invertible encoders to the property of optimal learning dynamics (Property 6.1.1.iii). In fact, the idea of supervised pretraining and dynamical isometry is equivalent for nontrivial ideal autoencoders with joint parametrization.

Proposition 6.1.5 (Pretraining and Dynamic Isometry)**.** *Greedy layer-wise unsupervised pretraining and dynamical isometry via orthogonal (or unitary) initialization are equivalent schemes for optimal learning dynamics for invertible unitary encoders.*

Proof. Consider a greedy layer-wise unsupervised pretraining procedure on a network initialized with orthogonal weights and $\mathcal{C}^1$ piecewise diffeomorphic activation functions. Let the decoder Ψ^d be given as in the proof of Lemma 6.1.3. Then the optimization objective is trivially satisfied and no training is necessary. ■

This result connects both ideas of optimal learning dynamics, in that both allow the network to initialize the estimation process with more or less optimal gradient propagation from the input layer to the output layer. Ensuring that this property holds for each appended layer allows for deeper architectures.

The next result connects the invertible encoders to Parseval regularization, showing that these models are trivially robust to adverserial attacks and could allow the models to exhibit stronger properties of interpretability (Property 6.1.1.iv)

Proposition 6.1.6 (Ideal Autoencoders are Parseval Optimal)**.** *Invertible unitary encoders are optimal with respect to Parseval Regularization.*

Proof. As each weight is unitary, we have $W_\theta^* W_\theta = I$ so the network weights are Lipschitz-1, trivially minimizing $J(\vartheta)$ for all $\theta \in \vartheta$. ■

A Parseval network differs from our outline of an ideal autoencoder as the former clearly lacks a supervised objective, but this is easily remedied. Our last result of this section connects the implicit regularization results of supervised autoencoders (Property 6.1.1.ii) to invertible encoders, and follows directly from Definition 5.2.1 and Lemma 6.1.3.

Corollary 6.1.7 (Supervised Invertible Encoder)**.** *Let Ψ_{IE} be a supervised ideal autoencoder. Then Ψ_{IE} satisfies the criteria for an invertible neural network, and we call Ψ_{IE} an* invertible encoder*.*

In this section, we have shown the existence of invertible encoder networks in Lemma 6.1.3, and related these models to some of the desirable properties for dealing with inverse problems which we have discussed throughout our thesis. In particular, we have demonstrated a direct link to optimal learning dynamics, adverserial robustness and general stability. By proxy to general AE models, these models might theoretically exhibit similarities to SVE/SVD (in the linear setting) and interpretable neural networks, however these properties are not directly ensured and need to be verified. All the mentioned properties (Properties 6.1.1.i to 6.1.1.iv) can be enforced via implicit or explicit unitary constraints on the network weights through models defined in ideal encoder-decoder structures and proper choice of activation functions. Furthermore, from Definition 4.4.7 we know that general autoencoder models can be made generative by the addition of stochastic sampling layers (Definition 4.3.7) using KL-divergence as a regularization term on the approximate distribution of the latent variables.

However, some details and components are notably missing. We have yet to discuss exactly how unitary or orthogonal constraints can be enforced, the precise form of the activation functions described in the proof of Lemma 6.1.3, and to relate Properties 6.1.1.v and 6.1.1.vi to our proposed model. The rest of the chapter is dedicated to demonstrating how these concepts can be introduced in our proposed model architecture.

6.2 Bi-Lipschitzian Activation Functions

The autoencoder structure in Lemma 6.1.3 places constraints on the choice of activation functions, namely that they are piecewise $\mathcal{C}^1$ diffeomorphisms. Additionally, we would like to enforce some bounds on the Lipschitz constant k for robustness and stability – in particular with respect to adverserial attacks as discussed in Sections 4.5 and 5.1. In the context of bijectivity, this requires us to expand Definition 3.5.1 to *bi-Lipschitz continuous* functions.

Definition 6.2.1. Let $\mathcal{X}, \mathcal{Y}$ be metric spaces, and let $f : \mathcal{X} \to \mathcal{Y}$ be a mapping such that

$$\frac{1}{k} d_{\mathcal{X}}(x_1, x_2) \leq d_{\mathcal{Y}}\big(f(x_1), f(x_2)\big) \leq k d_{\mathcal{X}}(x_1, x_2) \tag{6.7}$$

for all $x_1, x_2 \in \mathcal{X}$. Then f is *bi-Lipschitz continuous* with Lipschitz constant k.

Bi-Lipschitz continuity will ensure that the activation is bounded and *non-saturating*, meaning their domain is all of $\mathbb{R}$. This property allows us to quantify the maximum slope of the activation in both forward and inverse directions. This is also useful for numerical stability during training.

Unfortunately, there are almost no commonly used activation functions which feature these properties. As an example, consider the sigmoidal logistic function from Equation (3.36). This function is well behaved in the forward direction, but its inverse – the logit function given in Equation (3.35) – is only defined for inputs in $]0, 1[$. Worse yet, it is satured, which almost certainly leads to exploding gradients. The same argument applies to other common sigmoidal functions as well.

Bi-Lipschitzian ReLU

While the *relu* function is clearly not bijective, its counterpart – the *leaky relu* function [MHN13] – is. It does nevertheless require a minor modification to be bi-Lipschitz piecewise $\mathcal{C}^1$ diffeomorphic.

Definition 6.2.2 (Bi-Lipschitzian relu)**.** The *bi-Lipschitzian rectified linear unit function (birelu)* is given by

$$\gamma_{\text{birelu}}(x; k) = \begin{cases} kx, & \text{if } x \geq 0; \\ x/k, & \text{otherwise.} \end{cases} \tag{6.8}$$

The birelu function satisfies our aforementioned criteria. The function is not in $\mathcal{C}^1$ itself, due to the discontinuity of the gradient at the origin, however similar modifications can also be applied to activation functions of a similar nature to the rectified linear unit. In particular, we propose adapting the elu Equation (4.3) and celu functions Equation (4.4) for these purposes.

Bi-Lipschitzian ELU

The exponential linear unit function differs from relu by including negative values which serve to push the mean activation closer to zero. Recall that the function is given by

$$\gamma_{\text{ELU}}(x;\beta) = \begin{cases} x, & \text{if } x \geq 0; \\ \beta(e^x - 1), & \text{otherwise,} \end{cases} \tag{6.9}$$

with a functional inverse given by

$$\gamma^{-1}_{\text{ELU}}(y;\beta) = \begin{cases} y, & \text{if } y \geq 0; \\ \ln(y/\beta + 1), & \text{otherwise.} \end{cases} \tag{6.10}$$

To modify the exponential linear unit function to be bi-Lipschitzian, we want to constrain its Lipschitz constant to k, so we require the derivatives of both forward and inverse functions to have a maximum at k. For our purposes, it is enough to consider the piecewise functions defined on $\mathbb{R}_{<0}$. The derivatives are given by

$$D\gamma_{\text{ELU}<0}(x;\beta) = \beta e^x \tag{6.11}$$
$$D\gamma^{-1}_{\text{ELU}<0}(y;\beta) = (y+\beta)^{-1}, \tag{6.12}$$

which have easily derived inverses

$$(D\gamma_{\text{ELU}<0})^{-1}(y;\beta) = \ln(y/\beta) \tag{6.13}$$
$$(D\gamma^{-1}_{\text{ELU}<0})^{-1}(x;\beta) = x^{-1} - \beta, \tag{6.14}$$

where we let D signify the differential operator for notational clarity. For the function to have Lipschitz constant k we simply scale the outputs on $\mathbb{R}_{\geq 0}$ by k. Evaluating the inverse of the derivatives at $\beta = k$ sets knots at $a = -2\ln(k)$, as well as $b = k^{-1} - k$, where the derivatives are exactly k. This can be used to define the *bi-Lipschitzian exponential linear unit* activation.

Definition 6.2.3 (Bi-Lipschitzian ELU)**.** *Let $k \in \mathbb{R}_{>0}$, and define two points at $a = -2\ln(k)$ and $b = k^{-1} - k$. The bi-Lipschitzian exponential linear unit function (BIELU) is given by*

$$\gamma_{\text{BIELU}}(x;k) = \begin{cases} kx, & \text{if } 0 < x; \\ k(e^x - 1), & \text{if } a < x \leq 0; \\ (x-a)/k + b, & \text{if } x \leq a, \end{cases} \tag{6.15}$$

and its inverse is given by

$$\gamma^{-1}_{\text{BIELU}}(y;k) = \begin{cases} y/k, & \text{if } 0 < y; \\ \ln(y/k + 1), & \text{if } b < y \leq 0; \\ k(y+b) - a, & \text{if } y \leq b. \end{cases} \tag{6.16}$$

Bi-Lipschitzian CELU

The CELU function was proposed as a refinement on ELU designed to use exponential saturation for negative values while still pushing the mean activation towards the origin. The function is given by

$$\gamma_{\text{CELU}}(x;\beta) = \begin{cases} x, & \text{if } x \geq 0; \\ \beta(e^{x/\beta} - 1), & \text{otherwise,} \end{cases} \tag{6.17}$$

whereas its inverse is given by

$$\gamma^{-1}_{\text{CELU}}(y;\beta) = \begin{cases} y, & \text{if } y \geq 0; \\ \beta \ln(y/\beta + 1), & \text{otherwise.} \end{cases} \tag{6.18}$$

As opposed to ELU, the CELU function is more reliant on the parameter β to push the mean activation to zero so requires a dual parametrization k, β. Furthermore, it requires four knots instead of two for a bi-Lipschitz parametrization. This is for the gradient to reach k, which the exponential piecewise component only does on $\mathbb{R}_{\geq 0}$. However, two of these will turn out to be symmetric, requiring three computations. Computing the derivative yields

$$D\gamma_{\text{CELU}<0}(x;\beta) = e^{x/\beta} \tag{6.19}$$
$$D\gamma^{-1}_{\text{CELU}<0}(y;\beta) = \beta(y+\beta)^{-1}, \tag{6.20}$$

and taking their inverses gives us

$$(D\gamma_{\text{CELU}<0})^{-1}(y;\beta) = \beta \ln(y) \tag{6.21}$$
$$(D\gamma^{-1}_{\text{CELU}<0})^{-1}(x;\beta) = \beta(x^{-1} - 1). \tag{6.22}$$

The first knot for the forward operator are thus given by $a_1 = -\beta \ln(k)$ and the second knot is given by $b_1 = \beta \ln(k)$, which is just reflected over the origin. The last two knots correspond to the inverse operator and is given by $a_2 = \beta(k^{-1} - 1)$, and the last knot is subsequently given by $b_2 = \beta(k-1)$. We can thus get away with only computing a_1, a_2, b_2. This is useful if β should be given as a learnable parameter. The application of k as a learnable parameter requires the constraint $k \in \mathbb{R}_{\geq 1}$.

Definition 6.2.4 (Bi-Lipschitzian CELU)**.** Let $\beta \in \mathbb{R}_{>0}, k \in \mathbb{R}_{\geq 1}$, and define points at $a_1 = -k\ln(k), b_1 = -a_1$ and $a_2 = \beta(k^{-1} - 1), b_2 = \beta(k-1)$. The *bi-Lipschitzian continuously differentiable exponential linear unit function (BICELU)* is then given by

$$\gamma_{\text{BICELU}}(x;\beta,k) = \begin{cases} k(x - b_1) + b_2, & \text{if } b_1 < x; \\ \beta(e^{x/\beta} - 1), & \text{if } a_1 < x \leq b_1; \\ k^{-1}(x - a_1) + a_2, & \text{if } x \leq a_1; \end{cases} \tag{6.23}$$

and its inverse is given by

$$\gamma^{-1}_{\text{BICELU}}(y;k) = \begin{cases} k^{-1}(y - b_2) + b_1, & \text{if } b_2 < y; \\ \beta \ln(y/\beta + 1), & \text{if } a_2 < y \leq b_2; \\ k(y - a_2) + a_1, & \text{if } y \leq a_2. \end{cases} \tag{6.24}$$

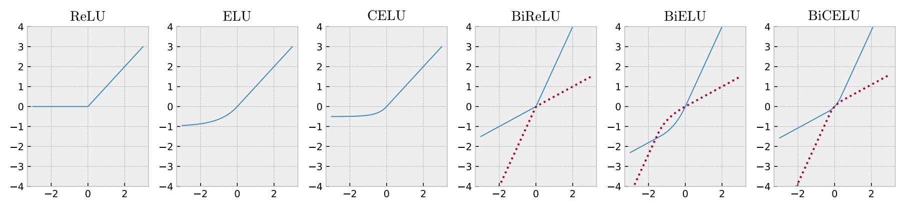

Figure 6.1: Bi-Lipschitzian activation functions and their canonical counterparts.

Act.Func.	Non-Affine	Smooth	Monotonic	Low Complex.	Apr.Id.
BIRELU	✓		✓	✓	
BIELU	✓	✓	✓		
BICELU	✓	✓	✓		✓

Table 6.1: List of bi-Lipschitzian activation functions and their connection to ideal properties from Observation 4.1.2.

The BICELU activation has an advantage over the BIELU function in that it is exactly identity at the origin, highlighted in Property 4.1.2.v as particularly useful in terms of more flexible weight initialization. In fact, the image of the forward and inverse function of BICELU tangentially intersects in a single point at the origin. Apart from this, BIELU and BICELU share most of the other useful properties highlighted in Observation 4.1.2. A comparison of the bi-Lipschitzian activation functions and their properties are listed in Table 6.1.

In addition to these observations, we note that we have $\gamma \to \mathrm{Id}$ as $k \to 1$ for all proposed bi-Lipschitzian activations, which is very much in line with our motivation. Moreover, for BICELU we have that $\gamma_{\text{BICELU}} \to \gamma_{\text{BIRELU}}$ as $\beta \to 0$. We can thus consider BIRELU as a special case of BICELU.

We stress the fact that the parameterization of all functions can be made trainable, as the parameters are easily updated with gradient methods. This is particularly interesting for the β parameter for the BICELU function. The downside with the smooth activation functions BIELU and BICELU is that they are more computationally expensive than the BIRELU function, but the smooth gradients should theoretically lead to better stability in training.

Without setting the parameters as trainable, both hyperparameters k and β thus need to be inferred. For $k >> 1$ we note that the nonlinearities are more prevalent, but increases the overall Lipschitz constant of the network. For $k \approx 1$ this symmetrically implies the network is approximately linear. For this reason, we consider $k = 2$ a reasonable estimate. The parameter β in BICELU controls the sharpness of the curve around the origin. In a rudimentary test, we found that setting $\beta = 1/2$ seemed to slightly improve the initial convergence. We stress that little work was done on estimating the optimal values of these parameters in the model, and further work can be done on this subject.

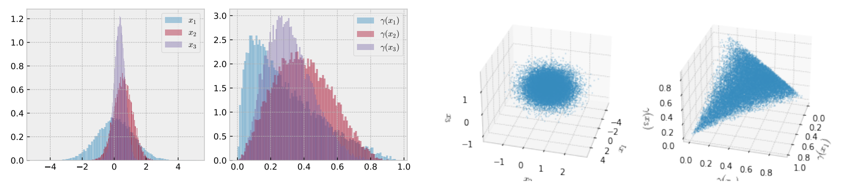

Figure 6.2: Visualization of softmax function. On the left, the histogram of three generated gaussian random variables and their transform via softmax denoted by γ with temperature parameter $\tau = 1$. On the right, the observations are plotted in three dimensions. Notice how γ acts by projecting into a two-dimensional simplex on a plane.

6.3 Invertible Dirichlet-Softmax Transform

In Section 4.1 we introduced the softmax activation in Equation (4.5), and discussed its applications as a multivariate extension of the logistic function (Equation (3.36)). The softmax function is clearly not bijective, making it impossible to apply in invertible neural networks. Seeing as the softmax activation is so commonly used in classification tasks [He+15; KSH17; SZ15; Sze+14], this is something we want to address. We thus need a bijective transformation onto the $d-1$ simplex, which can be used as a probability distribution over a discrete set of classes. In fundamental literature on Bayesian modelling [Gel+14], it is often the case that categorical distributions are modelled with a conjugate Dirichlet prior, which produce more expressive probablity distributions over a $d-1$ simplex. The relation between the softmax and Dirichlet distribution is investigated in [AS80]. We propose to replace the softmax function with a diffemorphic transformation from a location-scale family (Definition A.7.15) to an approximate Dirichlet distribution, with the intent of applying the reparametrization trick in a VAE context (Definition 4.4.7).

Related Work

Several approaches have been proposed for modelling Dirichlet distributions with VAE networks. [Zha+18] uses the Weibull distribution to construct an approximation, while [Joo+19] instead compose a Dirichlet approximation by Gamma random variables. These do not allow for bijective application of the reparametrization trick, however. In [SS17b] the authors instead model a Dirichlet distribution via a Laplace bridge [Mac98]. This method was originally proposed by [Hen+11], and approximates a Dirichlet by LogisticNormal(μ, Σ). This does not directly solve our problem at hand as it directly requires the application of the softmax function – which is what we want to approximate. In [NS17], the authors propose to construct autoencoders based on a *stick breaking process*. The process can be derived using the *marginal beta-sampling scheme* [Fer73] for a Dirichlet distribution.

Theorem 6.3.1 (Marginal Beta-Sampling)**.** *Let $\alpha \in \mathbb{R}^k$. Furthermore, let $\mathrm{Z}_i \sim Beta(\alpha_i, \sum_{i<j} \alpha_j)$ for $i = 1, \ldots, k$. Let Y be a k-dimensional random variable such that for $i < d$ we have*

$$\mathrm{Y}_i = \mathrm{Z}_i \left(\sum_{j<i} 1 - \mathrm{Y}_j \right), \tag{6.25}$$

and lastly we set $\mathrm{Y}_d = 1 - \sum_{j<d} \mathrm{Y}_j$. Then $\mathrm{Y} \sim Dirichlet(\alpha)$, generated by marginal beta-sampling*.*

The sampling process from Theorem 6.3.1 can be rewritten to only include the Z_i terms by

$$\mathrm{Y}_i = \mathrm{Z}_i \prod_{j<i} (1 - \mathrm{Z}_j) \tag{6.26}$$

which is the general form of stick-breaking for a *Dirichlet Process*, a non-parametric stochastic model commonly applied in Bayesian modelling for estimating infinite dimensional distributions. The contribution of [NS17] is to apply a stick-breaking process via Kumaraswamy distributed random variables [Kum80] to generate a Griffiths-Engen-McCloskey (GEM) distribution [PY16, Section 2]. Again, this is not directly bijective.

Symmetric Dirichlet-Softmax Transform

We propose an alternative approach via a stick-breaking process using an approximation of a Beta distribution via Gaussian random variables. By the overdetermination of the $d-1$ simplex, we elect to make the parametrization determinable by treating the normalizing constant in the softmax transform as a random variable S as a sum of log-normally distributed variables. Assuming independent inputs $\mathrm{X}_i \sim \mathcal{N}(0,1)$, we can simplify what is known as the the Fenton-Wilkinson approximation for a sum of log-normal variables [Mar67].

Theorem 6.3.2 (Simplified Fenton-Wilkinson Transform)**.** *Let $\exp(\mathrm{X}_i) \sim \mathrm{LogNormal}(0,1)$ for $i = 1, \ldots, d$. Then the sum is given by $S = \sum_i \exp(\mathrm{X}_i)$ is approximately $\mathrm{LogNormal}(\mu_S, \sigma_S^2)$ with parameters given by*

$$\sigma_S^2 = \ln\left(\frac{e-1}{d} + 1\right) \tag{6.27}$$

$$\mu_S = \ln(d) + \frac{1 - \sigma_S^2}{2}. \tag{6.28}$$

The idea is to let the last input in a hidden layer $x_d \sigma_S + \mu_S = \ln(S)$ act as the normalizing constant and approximate a logistic normal distribution with a Dirichlet distribution. The naive approach is then to take

$$\mathrm{Y}_i = \gamma(X)_i = \exp(\mathrm{X}_i - \ln(S)) \tag{6.29}$$

$$\ln(\mathrm{Y}_i) + \ln(S) = \mathrm{X}_i, \tag{6.30}$$

where γ is the softmax function and $\mathrm{Y} \sim \mathrm{Dirichlet}(\alpha)$, which naturally yield less than optimal approximations.

Instead, we propose to apply a transformation by Wise [Wis60] which transforms beta distributed variables to be approximately Gaussian.

Theorem 6.3.3 (Wise Transform)**.** *Let* $\mathrm{Z}_i \sim \mathrm{Beta}(\alpha_i, \beta_i)$*. Then for* $\alpha_i \geq \beta_i \geq 1$ *we have that* $\sqrt[3]{-\ln(\mathrm{Z}_i)}$ *approximately follows* $\mathcal{N}(\mu_i, \sigma_i^2)$ *with parameters*

$$N_i = \alpha_i + \frac{\beta_i - 1}{2} \tag{6.31}$$

$$\mu_i = \sqrt[3]{\beta_i}\left(1 - \frac{1}{9\beta_i}\right) \sqrt[-3]{N_i - \frac{(\beta_i - 1)(\beta_i + \frac{1}{3})}{12N_i^2}} \tag{6.32}$$

$$\sigma_i \propto (\sqrt[6]{\beta_i}\sqrt[3]{N_i})^{-1}. \tag{6.33}$$

This result is a refinement of [KL59] relating the CDF of a beta distribution to the Chi-squared distribution and applying a normalizing constant – which Wise calls this the 'Wilson-Hilferty transformation'.

We will apply the Wise transform to improve the naive approximation from Equation (6.29) by letting $\sqrt[3]{-\ln(\mathrm{Z}_i)} + \ln(S) = \mathrm{X}_i$ be approximately $\mathcal{N}(\mu_{\mathrm{X}_i}, \sigma_{\mathrm{X}_i})$ for $d-1$ variables. The term $\ln(S)$ comes from the normalization term of the softmax transformation and serves to help correct any skew and generally improve the Gaussian approximation.

As the Wise transform is bijective, this gives us an invertible approximate Beta to Gaussian transformation. Furthermore, as we are free to set the parameters of the transformation, we can apply a stick-breaking process via marginal beta sampling Theorem 6.3.1 to yield a Dirichlet distribution, of which we can construct an inverse of the form

$$\mathrm{Z}_i = \frac{\mathrm{Y}_i}{1 - \sum_{j<i} \mathrm{Y}_j}, \tag{6.34}$$

for $i < d$. The most basic way to achieve this is by considering the Dirichlet variable as an uninformed prior for our transformation, $\mathrm{Y} \sim \mathrm{SymDirichlet}(\alpha, d)$, i.e. $\alpha_i = \alpha_j$ for $i, j \in \mathbb{N}_{\leq d}$ using some specified parameter α.

To this end, there are three considerations we need to address. Wise gives the error of the approximation as proportional to $\sqrt[4]{\beta_i/N_i}$ for $\alpha_i \geq \beta_i$. Furthermore, as we require $\beta_i > \alpha_i$, we note that the Wise transform can easily be extended by considering the symmetry $1 - \mathrm{Z}_i \sim \mathrm{Beta}(\beta_i, \alpha_i)$ to hold for $\alpha_i < \beta_i$. Since $\alpha < (d-i)\alpha$ for $i < d-1$, this is the form required in our approach.

Secondly, in [Wis60] the variance σ_{X_i} is only provided up to proportionality. To tackle this, we empirically estimate the proportionality constants via least squares methods on sampled Dirichlet variables, and determined a constant scalar term $s = .315$ and a minor translation of $t = .001$. This gives us closed form expressions for $\mu_{\mathrm{X}}, \sigma_{\mathrm{X}}$ which most closely approximates $\sqrt[3]{-\ln(1 - \mathrm{Z}_i)}$, allowing us to appropriately scale and shift our approximation.

Finally, as we center our approximations, we will necessarily also center the distribution of the normalization constant, such that $\ln(S) \sim \mathcal{N}(0, \sigma_S^2)$. This allows us to define the full *Symmetric Dirichlet-Softmax transform.*

Proposition 6.3.4 (Symmetric Dirichlet-Softmax Transform)**.** *Let* $\mathrm{Y} \sim \mathrm{SymDirichlet}(\alpha, d)$ *and let*

$$\mathrm{Z}_i = \frac{\mathrm{Y}_i}{1 - \sum_{j<i} Y_j} \tag{6.35}$$

for $i = 1, \dots, d-1$. *Then* $1 - \mathrm{Z}_i \sim \mathrm{Beta}((d-i)\alpha, \alpha)$ *for* $\alpha \geq 1/2$. *Furthermore, define the parameters*

$$\sigma_S = \ln\left(\frac{e-1}{d} + 1\right) \tag{6.36}$$

$$N_i = (d-i)\alpha + \frac{\alpha - 1}{2} \tag{6.37}$$

$$\mu_{\mathrm{X}_i} = \sqrt[3]{\alpha}\left(1 - \frac{1}{9\alpha}\right) \sqrt[-3]{N_i - \frac{(\alpha-1)(\alpha+\frac{1}{3})}{12N_i^2}} \tag{6.38}$$

$$\sigma_{\mathrm{X}_i} = \frac{.315}{\sqrt[6]{\alpha}\sqrt[3]{N_i}} + .001. \tag{6.39}$$

Now let $\ln(S) \sim \mathcal{N}(0, \sigma_S^2)$, *and let*

$$\frac{\sqrt[3]{-\ln(1 - \mathrm{Z}_i)} - \mu_{\mathrm{X}_i}}{\sigma_{\mathrm{X}_i}} + \ln(S) = \mathrm{X}_i. \tag{6.40}$$

then X_i *is approximately* $\mathcal{N}(0,1)$, *and the transformation is bijective for a symmetric Dirichlet distributed variable.*

Proof. From Theorems 6.3.1 and 6.3.3 and Equation (6.27) the approximation is clear, and the forward transform is given by

$$\mathrm{Z}_i = 1 - \exp\left(-\left(\sigma_{\mathrm{X}_i}(\mathrm{X}_i - \ln(S)) + \mu_{\mathrm{X}_i}\right)^3\right) \tag{6.41}$$

$$\mathrm{Y}_i = \mathrm{Z}_i \prod_{j<i}(1 - \mathrm{Z}_j), \tag{6.42}$$

so the transformation is bijective. ■

The symmetric inverse Dirichlet-Softmax transform can be used in place of a softmax function to output a probability distribution in an invertible neural network. The most important applications of Proposition 6.3.4 is that the method can be readily extended to variational autoencoders without any modification of the Gaussian Kullback-Leibner regularization term. While the symmetric parametrization $\alpha_i = \alpha_j$ for all $i, j \in \mathbb{N}_{\leq d}$ can be considered too rigid for more involved modelling, this limitation is mostly artificial due to the practical implications of this thesis, and the method itself could be expanded to allow for setting α_i either as tuneable parameters estimated via gradient descent, or estimated via parallel inference networks with coupling layers.

6.4 Parametrization and Constraints

In this section, we will outline specific approaches to constructing invertible encoder networks, either by implicitly encouraging orthogonality by means of the network architecture, or explicitly enforcing constraints by parametrization using Lie groups and Riemannian gradient descent, or via one-layer residual blocks and resolvent operators derived from methods inspired by the solution of inhomogeneous integral equations.

Approximate Orthogonality by Pseudo-Invertible Encoders

We begin by introducing the concept of *pseudo-invertible encoders*.

Definition 6.4.1 (Pseudo-Invertible Encoders)**.** Let Ψ_{PIE} be an autoencoder model with joint parametrization, such that $\vartheta = \vartheta^e = \vartheta^d$. Let $W^*_{\theta_i} W_{\theta_i} \approx I$ for all $\theta_i \in \vartheta$. Then Ψ_{PIE} is a *pseudo-invertible encoder (PIE).*

The pseudo-invertible encoder is the simplest method of constructing models which have properties in common with invertible encoders. The idea is to train a model as an autoencoder with a supervised objective (Equations (4.34) and (4.35)) to encourage orthogonality in the weights via implicit constraints in the architecture. Given the construction defined in the proof of Lemma 6.1.3, the network weights are optimal precisely when $W^*_{\theta_i} W_{\theta_i} = I$ so a PIE model approximates an IE similar to how an autoencoder approximates an identity mapping on $\mathcal{X}$.

However, this begs the question; *why not use an autoencoder network*? In Observation 6.1.1 we outlined the benefits of invertible encoders, and many of these properties are shared with pseudo-invertible variants. In particular, autoencoder models do not enjoy the benefit of Parsival optimality (Property 6.1.1.iv) without explicit regularization, which the architecture of a PIE implicitly encourages by Proposition 6.1.6. Thus a PIE can effectively be considered equivalent to a *Parseval autoencoder network*. The advantage of PIE models is that they can be applied with almost no modifications to existing network structures. The experiments and results in Chapter 8 investigates the applicability of PIE models compared to AE models.

Orthogonal Constraints via Manifold Learning

To effectively enforce orthogonal constraints on the weights, we looked at relevant work on orthogonal- and unitary constraints on the parameters of the weights in neural networks. These constraints were originally proposed to solve the problem of long-term dependencies in recurrent network structures, motivated by the fact that contractions and dilations in hidden states lead to an eventual loss of information for recurrent iterations, which reduces long-term dependencies in modelled sequences [BFS93]. This essentially reflects the same issues as discussed in Section 4.5 where layerwise pretraining and dynamical isometry were introduced as remedial approaches to optimal gradient propagation for learning dynamics. Most of the earliest works on orthogonal or unitary constraints in neural networks were derived from QR decomposition methods using elementary Householder reflections and Givens rotations (Definition A.6.25) [ASB16; FB19;

Mha+16]. A more nuanced approach surfaced in [HR17], which related the problem of constructing unitary matrices to *Lie groups* and *Lie algebras*.

The rest of this section outlines the fundamental theory for understanding the basics of manifold learning on matrix Lie groups to enforce orthogonal constraints. This involves group theory, topology, manifolds, and differential geometry – all of which are noncentral topics in relation to our thesis. Our exposition should be considered a brief introduction to outline the fundamentals, and we emphasize that the theory is derived from the work in [AMS08; Cas19; Hal15] which provides a more comprehensive exposition. We will defer more detailed results to the appendix, with appropriate references in the text.

Lie Groups, Lie Algebras, and the Matrix Exponential

We begin by denoting the set of all $n \times n$ matrices as $\mathfrak{gl}(n)$, and note that $\mathfrak{gl}(n)$ is a linear inner product space with respect to the trace operator (Theorem A.6.1). This space does not form a group (Definition A.6.3) under matrix multiplication, as the definition of a group requires that every element has an associated inverse operator. By taking the subset of all invertible matrices, we get the maximal set of elements of $\mathfrak{gl}(n)$ required to yield a group structure under matrix multiplication.

Theorem 6.4.2 (General Linear Group)**.** *Let $\mathcal{GL}(n) \subset \mathfrak{gl}(n)$ such that $\mathcal{GL}(n) = \{W \in \mathfrak{gl}(n) : W \text{ is invertible}\}$. Then $\mathcal{GL}(n)$ is a group under matrix multiplication, called the* general linear group.

Proof. We know that $\mathcal{GL}(n)$ is associative (Property A.6.3.i) and contains the identity matrix (Property A.6.3.ii), and by definition it contains only invertible elements (Property A.6.3.iii). To show that $\mathcal{GL}(n)$ is a group, we only need to show that $\mathcal{GL}(n)$ is closed under matrix multiplication. Let $U, V \in \mathcal{GL}(n)$. Then U^{-1}, V^{-1} exists, and we have

$$UV(UV)^{-1} = U(VV^{-1})U^{-1} = UU^{-1} = I. \tag{6.43}$$

This holds for all $U, V \in \mathcal{GL}(n)$, so $\mathcal{GL}(n)$ is a group. ■

The general linear group forms the basis for a number of other matrix subgroups (see Definition A.6.5). For our purposes, we are first and foremost interested in the subgroup of orthogonal matrices, called the *orthogonal group*.

Lemma 6.4.3 (Special Orthogonal Group)**.** *Let $\mathcal{SO}(n)$ be the set of real orthogonal $n \times n$ matrices, i.e. $\mathcal{SO}(n) = \{W \in \mathcal{GL}(n) : W^\intercal = W^{-1}, \det(W) = 1\}$. Then $\mathcal{SO}(n)$ is a subgroup of $\mathcal{GL}(n)$ called the* orthogonal group.

Proof. $\mathcal{SO}(n)$ inherits associativety from $\mathcal{GL}(n)$, and as $I^\intercal I = I$ we have $I \in \mathcal{O}(n)$. Furthermore, from the multiplicative property of the determinant, we have $\det(UV) = \det(U)\det(V) = 1$ for all $U, V \in \mathcal{SO}(n)$. Then by substituting $(UV)^{-1} = (UV)^\intercal$ in Equation (6.43), we have that $\mathcal{SO}(n)$ is closed, and thus a subgroup of $\mathcal{GL}(n)$. ■

The general linear group and the orthogonal group are both classic examples of matrix Lie groups (Definition A.6.9 and Lemma A.6.10), however the orthogonal group has the additional property of being compact in $\mathfrak{gl}(n)$ (Lemma A.6.11). Lie groups are closely related to the *exponential map*. For matrix Lie groups, this takes the form of the *matrix exponential*.

Definition 6.4.4 (Matrix Exponential)**.** Let $W \in \mathfrak{gl}(n)$. We define the *matrix exponential* as

$$\exp(W) = \sum_{k=0}^{\infty} \frac{W^k}{k!}. \tag{6.44}$$

Each Lie group has an associated Lie algebra (Definition A.6.27), with the property that the image of the exponential map of the Lie algebra forms a Lie group. It turns out that the Lie algebra for the general linear group is $\mathfrak{gl}(n)$, and more importantly for our purposes; the Lie algebra for the orthogonal group is the linear space of skew-symmetric matrices (Theorem A.6.30), denoted by $\mathfrak{so}(n)$. Furthermore, the matrix exponential has the following important properties.

Proposition 6.4.5 (Properties of Matrix Exponential)**.** *Let $U, V \in \mathfrak{gl}(n)$. The matrix exponential has the following properties;*

(i) $\exp(U)$ *is convergent,*

(ii) $\exp(tU) \in \mathcal{C}^\infty$,

(iii) $\exp(0) = I$,

(iv) $\exp(U)^* = \exp(U^*)$,

(v) $\exp(U) \in \mathcal{GL}(n)$, *and* $\exp(-U) = \exp(U)^{-1}$,

(vi) $\exp(U + V) = \exp(UV)$ *if* $UV = VU$,

(vii) $\exp(UVU^{-1}) = U\exp(V)U^{-1}$ *for* $U \in \mathcal{GL}(n)$,

(viii) $\det\big(\exp(U)\big) = \exp\big(\mathrm{Tr}(U)\big)$,

(ix) $\frac{\partial}{\partial t}\exp(tU) = U\exp(tU) = \exp(tU)U$.

(x) $\frac{\partial}{\partial t}\exp(tU)\Big|_{t=0} = U$.

The proof for the properties listed in Proposition 6.4.5 can be found in [Hal15, pp.31–35]. It turns out that the elements of matrix Lie groups are *path-connected* (see Definition A.6.24 and Proposition A.6.26), which means we can construct a piecewise smooth path between any elements in the group using the matrix exponential. If the exponential map from a Lie algebra is surjective on its associated Lie group, we can construct a smooth path between any two elements in the group. As it turns out, this is the case for $\mathcal{SO}(n)$ [Roh13].

As our goal is to enforce the orthogonality of linear operators, the question is; why can we not simply let the operators in our network be parametrized by $\exp : \mathfrak{so}(n) \to \mathcal{SO}(n)$? As we will see, this will yield the correct gradient computations for the elements on $\mathcal{SO}(n)$. To fully make use of the orthogonal constraints provided by $\mathcal{SO}(n)$, we need to look at some fundamental theory for *smooth manifolds*.

Manifolds

A smooth manifold $\mathcal{M}$ (Definition A.6.21) is essentially a topological space (Definition A.6.12) which has the property of being locally Euclidean (Definition A.6.18) with a smooth differentiable structure (Definition A.6.20). This simply means that it is sufficiently similar to $\mathbb{R}^n$ to apply calculus in local areas around some point $p \in \mathcal{M}$. As it turns out, all matrix Lie groups yield a manifold structure.

Theorem 6.4.6 (Matrix Lie Groups are Manifolds)**.** *The matrix Lie groups are smooth manifolds.*

Proof. To show Theorem 6.4.6 we make use of Theorem A.6.23 which states that open subsets of a smooth manifold are themselves smooth manifolds. Note that the mapping $\operatorname{vec} : \mathfrak{gl}(n) \to \mathbb{R}^{n^2}$ yields a homeomorphism (Definition A.6.16), so $\mathfrak{gl}(n)$ is trivially a smooth manifold. We need to show that $\mathcal{GL}(n)$ is an open subset of $\mathfrak{gl}(n)$. As the elements of $\mathcal{GL}(n)$ are matrices with non-zero determinant, we note that the determinant is a polynomial, and thus $\det \in \mathcal{C}^\infty$. By considering the preimage

$$\det{}^{-1}\Big(\{\mathbb{R} \setminus \{0\}\}\Big) = \mathcal{GL}(n), \tag{6.45}$$

we have that $\mathcal{GL}(n)$ is an open subset of $\mathfrak{gl}(n)$, and is thus a smooth manifold. Lastly, as any subgroup of $\mathcal{GL}(n)$ can be constructed using the subspace topology (Definition A.6.13), all matrix Lie groups are likewise smooth manifolds. ■

Similarly to how we can define a tangent line at a point of a function in Euclidean space, we can similarly define a *tangent space* $T_p\mathcal{M}$ (Proposition A.6.32) on a point on a manifold, which consists of all possible tangent vectors with respect to the point p. The following result shows that defining the tangent space at a point on a matrix Lie group turns out to be straightforward.

Theorem 6.4.7 (Matrix Lie Algebra is Tangent Space)**.** *Let $\mathfrak{g}(n)$ be a matrix Lie algebra with associated matrix Lie group $\mathcal{G}(n)$. Then $T_I\mathcal{G}(n) = \mathfrak{g}(n)$, i.e. the tangent space at the identity is the Lie algebra.*

The proof for Theorem 6.4.7 is given in [Hal15, Corollary 3.46]. As multiplication is by definition continuous, we can apply left multiplication to each element of the tangent space which translates the tangent space to some other element on the manifold. Since this is invertible for $\mathcal{GL}(n)$, it is a diffeomorphism (Definition A.6.22), which preserves the structure of the tangent space (Definition A.6.20). Constructing the tangent space at a point $U \in \mathcal{SO}(n)$ yields

$$T_U\mathcal{SO}(n) = \{UV : V \in \mathfrak{so}(n)\}. \tag{6.46}$$

This result also shows why we cannot simply parametrize our weights as a skew symmetric matrix θ_i and let $W_{\theta_i} = \exp(\theta_i)$, as this will yield gradients in relation to the identity I as opposed to the point on the manifold we are currently at for our gradient descent algorithm. There is therefore a need for a change in the way we update our weights in the context of the manifold we wish to optimize over.

Riemannian Gradient Descent

To apply gradient descent directly on a manifold, we need to clarify what we mean by a gradient on a manifold. In Euclidean space, we associate the gradient with the direction and magnitude of the steepest incline at the evaluated point. Tangent spaces on a manifold can intuitively be understood as a generalization of a directional derivative on a manifold. To meaningfully characterize the magnitude of each each element in a tangent space, we need to endow the tangent space with some notion of distance or metric. Manifolds that associate an inner product to their tangent space are called *Riemannian manifolds*.

Definition 6.4.8 (Riemannian Manifold)**.** Let $\mathcal{M}$ be a smooth manifold. If $T_p\mathcal{M}$ is an inner product space for all $p \in \mathcal{M}$, then we say that $\mathcal{M}$ is a *Riemannian manifold*. A Riemannian manifold is equipped with a Riemannian metric $g_p : T_p\mathcal{M} \times T_p\mathcal{M} \to \mathbb{R}_{\geq 0}$ given by $g_p = \langle \bullet, \bullet \rangle_p$ induced from the inner product on $T_p\mathcal{M}$.

The induced norm $\|\bullet\|_p$ can be employed to compute lengths of paths between local points on $\mathcal{M}$. For matrix Lie groups, we have already established that the tangent space is given by the Lie algebra. This is an inner product space (Theorem A.6.1) and yields a Riemannian metric on $\mathcal{GL}(n)$ and its submanifolds, so the matrix Lie groups are Riemannian manifolds (Corollary A.6.38).

When we apply gradient descent over Euclidean space, we traverse over a vector field in straight lines defined by the gradient of some loss functional evaluated at our current estimate. The issue is that following a Euclidean straight line on the Euclidean space homeomorphic to a point on a manifold does not necessarily ensure that we stay on the manifold. Instead, a straight line on a manifold is a curve, called a *geodesic*.

Definition 6.4.9 (Curves and Geodesics on Manifolds)**.** Let $\mathcal{M}$ be a smooth manifold. Let $a \leq 0 < b$ and let $c : [a, b] \to \mathcal{M}$ with $c \in C^\infty([a, b])$. Then c is a *curve* on a manifold. Furthermore, fix $p \in \mathcal{M}$ and $v \in T_p\mathcal{M}$. Let $c_{p,v}$ be such that $c_{p,v}(0) = p$ and $c'_{p,v}(0) = v$. Then $c_{p,v}$ is called a *geodesic* on $\mathcal{M}$.

The element $v \in T_p\mathcal{M}$ can be interpreted as the initial velocity along a geodesic curve. A geodesic has the property of being the curve with the shortest length between two points on $\mathcal{M}$, and thus naturally generalizes the notion of a straight line to manifolds. For any smooth real-valued function $f : \mathcal{M} \to \mathbb{R}$, the function composition $f \circ c_{p,v}$ has a well-defined classical derivative

$$\frac{\partial f\big(c_{p,v}(t)\big)}{\partial t} = f'(c_{p,v}(t))c'_{p,v}(t), \tag{6.47}$$

which allows us to define gradients for f on a Riemannian manifold.

Definition 6.4.10 (Gradients on Riemannian Manifolds)**.** Let $\mathcal{M}$ be a smooth manifold with a Riemannian metric, and let $c_{p,v}$ be a geodesic. Let $f : \mathcal{M} \to \mathbb{R}$ be a smooth scalar field on $\mathcal{M}$, and let $\nabla_{\mathcal{M}} f(p) \in T_p\mathcal{M}$ be such that

$$\langle \nabla_{\mathcal{M}} f(p), v \rangle_p = \left.\frac{\partial f\big(c_{p,v}(t)\big)}{\partial t}\right|_{t=0} = D_p(f)(v). \tag{6.48}$$

where D_p is a derivation (Definition A.6.31). Then $\nabla_{\mathcal{M}} f(p)$ is called the *gradient* of f at the point p.

In this context, the term $D_p(f)$ can be interpreted as the directional derivative of f at point p. To apply gradient descent to Riemannian manifolds, we are particularly interested in the following properties of the gradient.

Proposition 6.4.11 (Properties of Gradient on Riemannian Manifolds)**.** *Let $\mathcal{M}$ be a smooth manifold with a Riemannian metric, let $p \in \mathcal{M}$, and let $\nabla_{\mathcal{M}} f(p)$ be the gradient of a smooth function f at the point p. Then we have the following;*

$$\frac{\nabla_{\mathcal{M}} f(p)}{\|\nabla_{\mathcal{M}} f(p)\|_p} = \underset{v \in T_p\mathcal{M}, \|v\|_p = 1}{\arg\max} \; D_p(f)(v), \tag{6.49}$$

$$\|\nabla_{\mathcal{M}} f(p)\|_p = D_p(f)\left(\frac{\nabla_{\mathcal{M}} f(p)}{\|\nabla_{\mathcal{M}} f(p)\|}\right). \tag{6.50}$$

In [AMS08, p.46] suggests that the implication of Proposition 6.4.11 is that among all unit vectors in $T_p\mathcal{M}$, the gradient $\nabla_{\mathcal{M}} f(p)$ is the direction of steepest incline for f at p, and that the norm of the gradient $\|\nabla_{\mathcal{M}} f(p)\|_p$ yields the steepest slope of f at p. This are the components needed for gradient descent on a Riemannian manifold. Consider the minimization problem

$$\min_{p \in \mathcal{M}} f(p). \tag{6.51}$$

With Riemannian gradient descent, we compute the gradient of a differentiable real-valued function f at a point p_i on a manifold $\mathcal{M}$, and subsequently update our position on $\mathcal{M}$ by moving along a geodesic c_v using some small step size ϱ, yielding the next point p_{i+1}. This process is then iterated over until some convergence criteria are met.

The issue now is that the gradient on the manifold is only defined implicitly in Definition 6.4.10. In [Cas19, Theorem 4.3] the author instead shows that if we have a mapping $\phi_p : T_p\mathcal{M} \mapsto \mathcal{M}$, then the minimization objective given by

$$\min_{v \in T_p\mathcal{M}} f(\phi_p(v)) \tag{6.52}$$

is equivalent to the optimization objective given in Equation (6.51), with a change of metric induced by the mapping ϕ_p. Notably, the optimization objective for Riemannian gradient descent is equivalent when selecting $\phi_p = c_{p,v}(t)$, the geodesic given by the canonical Riemannian exponential map. By the chain rule, we can then derive the update rules for Riemannian gradient descent given by

$$v_{i+1} = -\varrho\nabla(f \circ \phi_{p_i})(v_i) \tag{6.53}$$

$$p_{i+1} = \phi_{p_i}\big(-\varrho\nabla(f \circ \phi_{p_i})(v_i)\big) \tag{6.54}$$

How this directly relates to matrix manifolds might not be sufficiently clear. Recall that the matrix exponential yields smooth curves on the manifolds of Lie groups. It turns out that the matrix exponential provides the geodesics on these manifolds [AMS08, p.102]. For $U \in \mathcal{SO}(n), V \in \mathfrak{so}(n)$ and $\hat{V} = UV \in T_U\mathcal{SO}(n)$ we can define the geodesic

$$c_{U,\hat{V}}(t) = U\exp(tU^*\hat{V}) = U\exp(tB^*BV) = U\exp(V). \tag{6.55}$$

From Theorem A.6.1 we have that the inner product of elements of $T_U\mathcal{SO}(n)$ is given by

$$\langle V, W\rangle = \mathrm{Tr}\big((U^*V)^*U^*W\big) \tag{6.56}$$
$$= \mathrm{Tr}(V^*W) \tag{6.57}$$

which corresponds to the standard inner product, and this is bi-invariant for reductive Lie algebras (Definition A.6.35). The last piece of the puzzle comes from the practical concerns of numerical computation. In general, evaluating the geodesics of a manifold can be computationally heavy. In this case, we often replace the canonical Riemannian exponential map from the tangent bundle $T\mathcal{M}$ to $\mathcal{M}$ (Definition A.6.33) by a *retraction map* [AMS08, p.55].

Definition 6.4.12 (Retraction)**.** Let $\mathcal{M}$ be a manifold with tangent bundle $T\mathcal{M}$. Let $R_p : T_p\mathcal{M} \to \mathcal{M}$ be a smooth mapping such that for $p \in \mathcal{M}$ we have

(i) $R_p(0) = p$,

(ii) $D_p(R_p(0)) = \mathrm{Id}_{T_p\mathcal{M}}$.

Now, let $R = \{R_p : p \in \mathcal{M}\}$. Then R is known as a *retraction.*

A retraction can thus be considered a first-order approximation to the canonical retraction of the Riemannian exponential map. For the Lie matrix groups, this is equivalent to the matrix exponential. We can thus apply a suitable retraction map to Equation (6.52) to perform Riemannian gradient descent on matrix manifolds. From here, we can outline the algorithm for Riemannian gradient descent on $\mathcal{SO}(n)$, given in Algorithm 1. For the sake of simplicity, we only consider a single layer network without bias for illustrating the weight update process, however the most significant difference from Euclidean gradient descent is how the weight update is performed, and this can easily be generalized to a multilayer network.

Algorithm 1: Riemannian Gradient Descent on $\mathcal{SO}(n)$

```
Data: Activation function $\gamma$, base weight $U \in \mathcal{SO}(n)$, parametrization
      $V_\theta \in \mathfrak{so}(n)$, operator $W \in \mathcal{SO}(n)$, cost function $C$, data
      $(x_i, y_i)_{i=1}^N$, learning rate $\varrho$.
Initialize parametrization $V_\theta = 0$;
Initialize base weight $U$ in $\mathcal{SO}(n)$;
Choose $E$ = max epochs;
for $j = 1$ to $E$ do
    for $i = 1$ to $N$ do
        $W = R_U(V_\theta)$;
        $\hat{y}_i = \gamma(Wx_i)$;
        $d_U = \nabla(C \circ R_U)(V_\theta)|_{y_i,\hat{y}_i}$;
        $d_V = \nabla C(y_i, \hat{y}_i)$;
        $U = -\varrho d_U$;
        $V_\theta = -\varrho d_V$;
    end
end
```

In terms of semiorthogonal operators, we are also interested in optimization on the closely related *Stiefel manifold*.

Definition 6.4.13 (Stiefel Manifold)**.** Let $\mathrm{St}(n,k) = \{W \in \mathbb{R}^{n\times k} : W^*W = I_k\}$. Then $\mathrm{St}(n,k)$ is called the *Stiefel manifold*.

Applying Riemannian gradient descent to the Stiefel manifold requires addressing the underdetermination. In [AMS08, Example 3.5.2],the authors show that the tangent space $T_U\mathrm{St}(n,k)$ is given by

$$T_U\mathrm{St}(n,k) = \{UV + U_\perp V_\perp : V \in \mathfrak{so}(n), U_\perp \in \mathbb{R}^{n\times n-k}, V_\perp \in \mathbb{R}^{n-k\times n}\}, \tag{6.58}$$

suggesting the completion of $U \in \mathrm{St}(n,k)$ by a matrix $U_\perp \in \mathbb{R}^{n\times n-k}$ using a Gram-Schmidt process. Thus, the block matrix $[U\ U_\perp] \in \mathcal{O}(n)$. Furthermore, from [AMS08, Example 3.6.2], we have that any element of $T_U\mathcal{M}$ can be decomposed into a symmetric and skew symmetric element so that for $\xi \in T_U\mathrm{St}(n,k)$ we have

$$P_U\xi = (I - UU^*)\xi + U\,\mathrm{skew}(U^*\xi) \tag{6.59}$$

$$P_U^\perp\xi = U\,\mathrm{sym}(U^*\xi), \tag{6.60}$$

where $\mathrm{skew}(U) = \frac{1}{2}(U - U^*)$ and $\mathrm{sym}(U) = \frac{1}{2}(U + U^*)$. Then the gradients of a smooth function f are given by the projection $\nabla P_U f(U)$, allowing us to apply similar methods as for the case of $\mathcal{SO}(n)$. In practical terms, the only difference from Algorithm 1 is the completion of the base matrix U via a Gram-Schmidt process. In [Cas19, p.21], the author suggest applying a thin QR factorization for this process, which unfortunately is computationally costly. However, this is not a problem with any immediate remedial solution, and the final algorithm is outlined in Algorithm 2.

Up until recently, the application of the canonical retraction given by the matrix exponential was computationally costly, prompting the application of alternative retractions. The most common retraction for $\mathcal{SO}(n)$ is the *Cayley transform*.

Proposition 6.4.14 (Cayley Transform)**.** *Let $V_\theta \in \mathfrak{so}(n)$. Let R be a retraction such that $R_U : T_U\mathcal{SO}(n) \to \mathcal{SO}(n)$ is given by*

$$R_U(V) = U(I+V)(I-V)^{-1}. \tag{6.61}$$

Then R is a retraction map given by the Cayley transform.

Proof. Let $W = (I+V)(I-V)^{-1}$. For R_U to be a retraction, it suffices to show that $W \in \mathcal{O}(n)$. We have that

$$WW^* = (I+V)(I-V)^{-1}\big((I+V)(I-V)^{-1}\big)^* \tag{6.62}$$

$$= (I+V)(I-V)^{-1}(I+V)^{-1}(I-V) \tag{6.63}$$

$$= I \tag{6.64}$$

since $(I-V)^{-1}$ commutes with $(I+V)$. Then R is a retraction map, as we wanted. ■

Algorithm 2: Riemannian Gradient Descent on $\mathrm{St}(n,k)$

```
Data: Activation function $\gamma$, base weight $U \in \mathcal{SO}(n)$, parametrization
      $V_\theta \in \mathbb{R}^{n \times k}$, operator $W \in \mathrm{St}(n,k)$, cost function $C$, data
      $(x_i, y_i)_{i=1}^N$, learning rate $\varrho$.
Initialize parametrization $V_\theta = 0$;
Initialize base weight $U$ in $\mathcal{SO}(n)$;
Choose $E =$ max epochs;
Choose $L =$ max QR iterations;
for $j = 1$ to $E$ do
    for $i = 1$ to $N$ do
        Initialize $V_0 = 0 \in \mathbb{R}^{n \times n-k}$ ;
        $V = \mathrm{skew}\big([V_\theta \ V_0]\big)$;
        $W = R_U\big(V\big)$;
        $\hat{y}_i = \gamma(W x_i)$;
        $d_U = \nabla(C \circ R_U)(V_\theta)|_{y_i, \hat{y}_i}$;
        $d_V = \nabla C(y_i, \hat{y}_i)$;
        $d_\perp = V_0$;
        for $i = 1$ to $L$ do
            $Q, R = \mathrm{qr}\big(d_\perp - d_U(d_U d_\perp)\big)$;
            $d_\perp = Q$;
        end
        $d_U = [d_U \ d_\perp]$ ;
        $U = -\varrho d_U$;
        $V_\theta = -\varrho d_V$;
    end
end
```

Recently, [BBC19] introduced a computationally cheap algorithm for the matrix exponential, allowing the use of canonical retraction maps for optimization with Riemannian gradient descent. In Section 9.1 we perform experiments with both the matrix exponential and the Cayley transform in invertible encoder networks.

In the context of optimization on the Stiefel manifold, it is important to note that the method does not yield invertible matrices in both directions for the method outlined in Lemma 6.1.3. For underdetermination between network layers – i.e. where $k > n$ – we simply apply the convention of using a transposed operator from $\mathrm{St}(n,k)$. No matter how this is implemented, any underdetermination or overdetermination in the network will cause certain network layers to not be strictly bijective.

Resolvent Operators and Liouville-Neumann Series

Another method for constructing invertible encoders comes from the spectral theory of inhomogeneous integral operators from Section 2.2. Recall that an inhomogeneous integral operator equation can be expressed on the form

$$y = (I - \lambda^{-1} W)x. \tag{6.65}$$

In Corollary 2.3.12 we introduced the Liouville-Neumann series as a method for constructing invertible operators for the inhomogeneous integral operator equations. Furthermore, in Definition 4.3.1 we discussed residual blocks as a method to ease the process of training neural networks by letting nonlinear layers act on the residual space of a linear operator, often in the form of an identity matrix. As previously mentioned, residual blocks have been applied to construct invertible networks [JSO18]. Our idea is to approach invertible networks by combining elements from these two methods, directly addressing Property 6.1.1.v as a motivation for invertible encoder networks.

Proposition 6.4.15 (Resolvent Layer)**.** *Let ψ be a single layer neural network, and let γ be a piecewise diffeomorphism. Let $\lambda \in \rho(U_\theta)$ and $W_\theta = I - \lambda^{-1}U_\theta \in \mathcal{GL}(n)$, such that*

$$\begin{aligned} y &= \psi(x;\theta,\gamma) && (6.66)\\ &= \gamma\big((I - \lambda^{-1}U_\theta)x\big). && (6.67)\end{aligned}$$

Then ψ is a piecewise diffeomorphism, called a resolvent layer*.*

Proof. As γ is a piecewise diffeomorphism, and the linear operator is diffeomorphic, it suffices to show that ψ is invertible. This is clearly the case as $W_\theta \in \mathcal{GL}(n)$ by definition. More constructively, for $\lambda \in \rho(U_\theta)$ and $\|U_\theta\| < |\lambda|$ we have by Corollary 2.3.12 that the inverse of ψ is given by

$$\begin{aligned} \psi^{-1}(y;\theta,\gamma) &= W_\theta^{-1}\gamma^{-1}(y) && (6.68)\\ &= \sum_{k\in\mathbb{Z}_{\geq 0}} \lambda^{-k}U_\theta^k\gamma^{-1}(y). && (6.69)\end{aligned}$$

Then ψ is a piecewise diffeomophism, as we wanted. ∎

While resolvent layers somewhat resemble single layer residual blocks, they lack the property of allowing a linear combination of an input to be further propagated throughout layers, thus the the similarity is rather superficial. However, this idea could be pursued further by applying the theory of nonlinear integral operators, such as the Urysohn operator [Kra64], a possible avenue of research we refer to in Section 10.3.

The power of resolvent layers is that they can be applied in the context of convolutional layers from Definition 4.3.2, allowing us to construct invertible encoder networks with convolutional layers. To show how this can be implemented, we provide the following results.

Proposition 6.4.16 (Resolvent Convolution Kernels)**.** *Let W_θ be a discrete resolvent convolution operator with circular boundary conditions, such that $W_\theta = I - \lambda^{-1}U_\theta$. Let θ be the kernel of U_θ. Then for $\|U_\theta\| < |\lambda|$ the kernel θ^{-1} of the operator W_θ^{-1} is given by*

$$\theta^{-1} = \mathcal{F}^{-1}\left[\sum_{k=0}^{\infty}\lambda^{-k}(\mathcal{F}\theta)^{\odot k}\right], \tag{6.70}$$

with appropriate circular shifts defined by the Fourier operator.

Proof. By the convolution theorem in Theorem A.7.7 we have that for a convolution with circular boundary conditions, the operator can be expressed by

$$\begin{aligned} y &= W_\theta x && (6.71)\\ &= \theta * x && (6.72)\\ &= \mathcal{F}^{\text{-}1}[\mathcal{F}\theta \odot \mathcal{F}x]. && (6.73) \end{aligned}$$

Using the Liouville-Neumann series, we have

$$\begin{aligned} x &= \sum_{k=0}^{\infty} \lambda^{-k} W_\theta^k y && (6.74)\\ &= \sum_{k=0}^{\infty} \lambda^{-k} \mathcal{F}^{\text{-}1}[(\mathcal{F}\theta)^{\odot k} \odot \mathcal{F}y] && (6.75)\\ &= \mathcal{F}^{\text{-}1}\left[\mathcal{F}y \odot \sum_{k=0}^{\infty} \lambda^{-k} (\mathcal{F}\theta)^{\odot k}\right], && (6.76) \end{aligned}$$

where we let the notation $\bullet^{\odot k}$ denote the pointwise exponentiation operator with the convention of letting $\bullet^{\odot 0} = \mathrm{Id}$ and use the linearity of the Fourier operator and Hademard product in Equation (6.76). Then the kernel $\theta^{\text{-}1}$ is of the form we wanted. ■

As the implementation requires an infinite series, the computation of the actual kernel is infeasible. For practical implementations, we can instead construct an approximation of the inverse kernel by taking

$$\theta_n^{\text{-}1} = \mathcal{F}^{\text{-}1}\left[\sum_{k=0}^{n} \lambda^{-k} (\mathcal{F}\theta)^{\odot k}\right] \qquad (6.77)$$

for some choice of n such that $|\theta_{n-1}^{\text{-}1} - \theta_n^{\text{-}1}| < \epsilon$. This kernel can then be applied to standard convolution operators for efficient computations. For multichannel convolution layers, this method requires the same number of input channels as the output channels – i.e. when the block matrix W_θ has full rank. When computing a multichannel convolution, we also need to account for the linear combinations of the channels. For this, we first permute $\mathcal{F}\theta$ by $c_i \times c_o \times h \times w \mapsto h \times w \times c_i \times c_o$ where c_i, c_o are the input and output channels respectively. By considering the resulting tensor as a collection of $c_i \times c_o$ matrices, we can apply the matrix power operator in place of the pointwise exponential for each k, and permute the result back to the original dimensions before adding the terms, resulting in correct computations.

Interestingly, this method of constructing invertible kernels can possibly be extended to Riemannian gradient descent via approximation of the canonical retraction of the matrix exponential. We highlight this as a priority for further work in Section 10.3.

PART III

Results

CHAPTER 7

Methodology and Baselines

In this chapter, we outline the general methodology behind our experiments. We begin by introducing the data and experimental tasks, before moving on to discuss the metrics used for estimation and evaluation. Finally, we perform baseline experiments to compare the applicability of our proposed activation functions from Sections 6.2 and 6.3.

7.1 Experimental Methodology

When planning our experiments, our main goal was to compare our proposed neural network architecture with existing autoencoder models to determine whether these models are applicable for solving inverse problems, while emphasizing methodological simplicity, low complexity, and reproducibility. As such, we retain moderately low dimensionality in our models and our experiments should be viewed in the context of 'proof-of-concept' experimentation rather than large-scale optimization and model benchmarks.

Datasets

The experiments were performed on images sourced from well-known datasets, two of which have been briefly introduced through our previous practical examples (Examples 2.4.3, 3.5.5, 2.2.7 and 2.3.8). An overview of the data used in the experiments can be found in Table 1.6, but we provide a more thorough exposition in this subsection.

CIFAR100

The CIFAR dataset [KH09] consists of $3 \times 32 \times 32$ natural images, and was used in Examples 2.4.3, 2.2.7 and 2.3.8. The dataset comes in variants of either 10 or 100 class labels. We only consider the 100 class variant in our thesis, and we do not use the class labels in our experiments. Instead, we augment the data by applying an average box blur with a 3×3 kernel, which we use in a supervised deblurring task. As the color channels are superflous to this task, we preprocess the images by applying a monochromatic filter to produce greyscale images with dimension 32×32. The training set consists of 50000 sample images, and the test set consists of 10000 images.

MS-COCO

In Example 3.5.5 we made use of image samples from the COCO dataset [Lin+14]. The dataset consists of three-channel natural images of varying sizes paired with image annotations. We do not use the provided annotations in this thesis, and we preprocess the data by first selecting only images with dimension 400×400 or more, resulting in 96336 samples in the training set and 4002 samples in the test set. These images are then cropped to 384×384 and a monochromatic filter is applied to produce single-channel images. We then augment the data by applying an average pooling downsampling filter to produce images of 96×96 pixels, which is subsequently used in a supervised image upscaling task.

EMNIST

The EMNIST dataset [Coh+17] is an extended version of the common MNIST dataset, and is the only dataset we have not yet introduced through our previous examples. The dataset consists of single-channel 28×28 images of handwritten digits and letters seperated into several different subsets, including the original MNIST data. In this thesis, we look at the *balanced* subset of 47 classes where the number of training and test examples for each class is equal. The training set consists of 112800 sampled images, and the validation set consists of 18800 images. No preprocessing was applied to the images.

Experimental Tasks

We look at four distinct learning tasks in our experiments, for each of which we designate a specific dataset. Two of these tasks have already been briefly introduced in our previous examples. For each task, we train an invertible- or pseudoinvertible encoder and compare it with a comparable autoencoder model to evaluate the performance and robustness. Note that the relationship between the dimension of the parameter space of an autoencoder to an invertible encoder with the same exact architecture and layer dimensions is given by $\dim(\Theta_{\text{AE}}) = 2\dim(\Theta_{\text{PIE}})$, however any difference in architecture and dimensions will offset this relationship.

Deblurring

The deblurring task was introduced in Examples 2.4.3, 2.2.7 and 2.3.8, and can be stated as follows. We are provided a dataset $\mathcal{D} = (x_i, y_i)_{i \in \mathcal{I}}$ consisting of natural images with $x_i, y_i \in [0,1]^{32 \times 32}$. The images are the result of a data generation process given by the linear forward problem $\Phi(x) = y$, where Φ is a convolution operator with an average box blur kernel. The task is to construct an approximately bijective map $\Psi : \mathcal{X} \to \mathcal{Y}$ using an invertible- or pseudoinvertible neural network. From the previous examples and theoretical exposition, we know that this problem is severely ill-conditioned. In our experiments, the deblurring task is considered exclusively in relation to the CIFAR dataset.

Image Upscaling

We briefly looked at an example with upscaling in Example 3.5.5. For this task, the dataset $\mathcal{D}$ consists of natural images with $x_i \in [0,1]^{384 \times 384}$ and

$y_i \in [0,1]^{96\times 96}$. The problem is a severely ill-posed underdetermined linear problem without any clearly defined inverse and the task is to construct a bijective map which approximates the upscaling operator with an invertible- or pseudoinvertible neural network. We exclusively consider the image upscaling task in the context of the COCO dataset in our experiments.

Unsupervised Autoencoding

The unsupervised autoencoding task is the general task of an autoencoder model and is fully described in Section 4.4. When dimensionality reduction is introduced in any operator in the model, the problem is ill-posed due to underdetermination. In the variational autoencoder setting, the problem is a density estimation problem, and the well-posedness is contingent on how well the choice of the approximate tractable distribution corresponds to the unknown marginal distribution. It is possible that the true multivariate probability distribution over a set of images has lower dimension than the images themselves, however a full rank approximation guarantees that no information is lost between the inputs and the latent probability space. The encoding task is applied to the EMNIST dataset, with images $x_i \in [0,1]^{28\times 28}$.

Conditional Image Reconstruction from Labels

The general classification task of images is well known and can be considered a nonlinear causal inverse problem in its own right. In a model identification setting, we can instead pose the classification task as a forward problem $y_i = \Phi(x_i;\theta)$ where x_i is an input image and y_i is a sample from a probability distribution over one-hot categorical labels. The associated inverse problem $\Phi^{-1}(y_i) = x_i$ is formulated as generating the input image given y_i. For $\dim(\mathcal{Y}) < \dim(\mathcal{X})$ the problem is underdetermined and subsequently ill-posed. To remedy the ill-posedness of this problem, we augment the dimensionality by introducing a latent space $\mathcal{Z}$ such that $\dim(\mathcal{X}) = \dim(\mathcal{Y}\times\mathcal{Z})$. This task was also trained and evaluated using the EMNIST dataset.

Conditional Image Upscaling

Similar to the conditional image reconstruction from labels, this task investigates a conditional upscaling model where we augment the dimensionality of the downsampled images $y_i \in \mathcal{Y}$ with a latent space $\mathcal{Z}$ such that $\dim(\mathcal{X}) = \dim(\mathcal{Y}\times\mathcal{Z})$. Note that we do not use any label information in this task, so the $y_i \in \mathcal{Y}$ variables are downsampled images, and the map $\Psi_x : \mathcal{X} \to \mathcal{Y}$ is deterministic. We trained and evaluated the models for this task using the COCO dataset.

Evaluation, Training and Metrics

Our proposed pseudo-invertible encoder (PIE)- and invertible encoder (IE) models were compared with general autoencoder architectures as baseline models for comparison in both supervised and unsupervised settings. Generative models were similarly compared with variational autoencoder models with Gaussian latent variables. In all experiments, the training process was generally performed concurrently for both models over the same batches. As the dimensionality

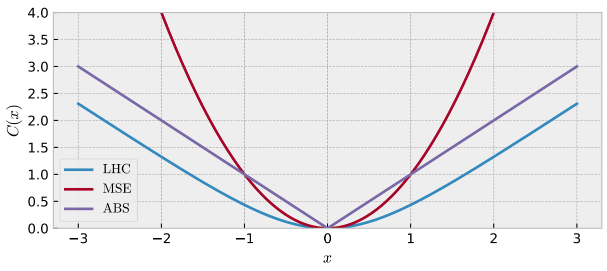

Figure 7.1: Illustration of empirical risk functions, comparing the logarithmic hyperbolic cosine (LHC) function, the squared error (MSE) and absolute error (ABS) functions.

of the parameter space and general architectures generally differed between models, we initialized the model weights separately, but all networks use the uniform weight initialization proposed in [He+15]. Regularization is not applied in any of the experiments, except for when explicitly mentioned. All models were trained using the ADAM optimization algorithm discussed in Section 4.2. We used a variety of relevant metrics to train and evaluate the models. An overview of the applied metrics is provided in Table 1.5, but we discuss some of the less well-known metrics in more detail.

Loss Functions

The logarithmic hyperbolic cosine (LHC) [Xu+21] is a parametrized empirical risk function which acts as a generalization of the absolute error function (ABS). The function is given by

$$C_{\text{LHC}}(x, \hat{x}; \beta) = \sum_{i=1}^{n} \frac{\ln\cosh\left(\beta(x_i - \hat{x}_i)\right)}{n\beta} \tag{7.1}$$

$$= \sum_{i=1}^{n} \frac{\ln\left[1 + \exp\left(-2\beta(x_i - \hat{x}_i)\right)\right] + \beta(x_i - \hat{x}_i) - \ln(2)}{n\beta}. \tag{7.2}$$

The parameter β acts as a softening of the function around the origin can be shown to be related to the parametrization of the Softplus function given by

$$\gamma_{\text{Softplus}}(x; \beta) = \frac{\ln(1 + \exp(\beta x))}{\beta} \tag{7.3}$$

$$= \frac{\ln[1 + \exp(-\beta x)] + \beta x}{\beta}, \tag{7.4}$$

where the similarities are particularly clear from Equation (7.2) and Equation (7.4). Additionally, we have that

$$\lim_{\beta \to \infty} C_{\text{LHC}}(x, \hat{x}; \beta) = |x - \hat{x}|,$$

so C_{LHC} is a smooth function which asymptotically approximates C_{ABS} as β approaches infinity. This promotes linear convergence further from the origin while maintaining continuity around zero. An plot showing the behaviour of LHC around the origin can be observed in Figure 7.1.

Image Reconstruction Metrics

The Peak Signal-to-Noise ratio (PSNR) has long been considered a canonical metric for quantifying image reconstruction errors. It is closely related to MSE, and given by

$$C_{\text{PSNR}}(x, \hat{x}) = 10 \log_{10} \frac{\max(\mathcal{X})^2}{C_{\text{MSE}}(x, \hat{x})}.$$

where $\max(\mathcal{X})$ is the maximum possible value of a pixel in the image space. However, [HG08] demonstrated that the PSNR can yield inaccurate results for image reconstruction purposes, and instead proposes the structural similarity metric (SSIM) which has shown to better account for perceived differences in image reconstruction compared to human cognition [Wan+04]. Structural similarity is computed over window functions for two images by applying

$$C_{\text{SSIM}}(x, \hat{x}; c_1, c_2) = \frac{(2\mu_x \mu_{\hat{x}} + c_1)(2\sigma_{x\hat{x}} + c_2)}{(\mu_x + \mu_{\hat{x}} + c_1)(\sigma_x + \sigma_{\hat{x}} + c_2)} \tag{7.5}$$

where c_1, c_2 are parameters derived from the local dynamic range for each window. We typically apply SSIM with a window size of 11×11 pixels in our applications.

Robustness and Stability

In Section 5.1 we discussed adverserial attacks and remarked that the adverserial perturbation vector δ can be interpreted as a worst-case estimate of the noise component ε of an inverse problem. To quantify the robustness of our models, we look for an adverserial perturbation $\hat{\delta}$ for each model by estimating

$$\hat{\delta} = \operatorname*{arg\,max}_{\delta : \|\delta\|_2 \leq r} \|\Psi(x + \delta; \vartheta) - \Psi(x; \vartheta)\|_2^2 \tag{7.6}$$

using *projective gradient descent* [Ant+20; Tsi+19] for a ball of radius $r = 1$ over the training set. We then evaluate relative condition number (Definition 3.5.2) of the model using the estimated perturbation vector, and call this the *adverserial condition number* (CNA) for the model. For unsupervised learning, this is computed as a perturbation on the inputs $x_i \in \mathcal{X}$ over the test set of n samples by taking

$$\text{CNA}(\hat{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{\|(\Psi^d \circ \Psi^e)(x_i + \hat{\delta}_x) - (\Psi^d \circ \Psi^e)(x_i)\|_2}{\|(\Psi^d \circ \Psi^e)(x_i)\|_2} \cdot \frac{\|x_i\|_2}{\|\hat{\delta}_x\|_2}. \tag{7.7}$$

For supervised learning, we are more interested in perturbations of the outputs $y_i \in \mathcal{Y}$, so we additionally compute

$$\text{CNA}(\hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \frac{\|\Psi^d(y_i + \hat{\delta}_y) - \Psi^d(y_i)\|_2}{\|\Psi^d(y_i)\|_2} \cdot \frac{\|y_i\|_2}{\|\hat{\delta}_y\|_2}. \tag{7.8}$$

In addition to the condition numbers estimated via adverserial perturbation, we estimated the relative condition number computed via the Jacobian (CNJ) using a fixed subset of the relevant validation set. We empirically estimate the

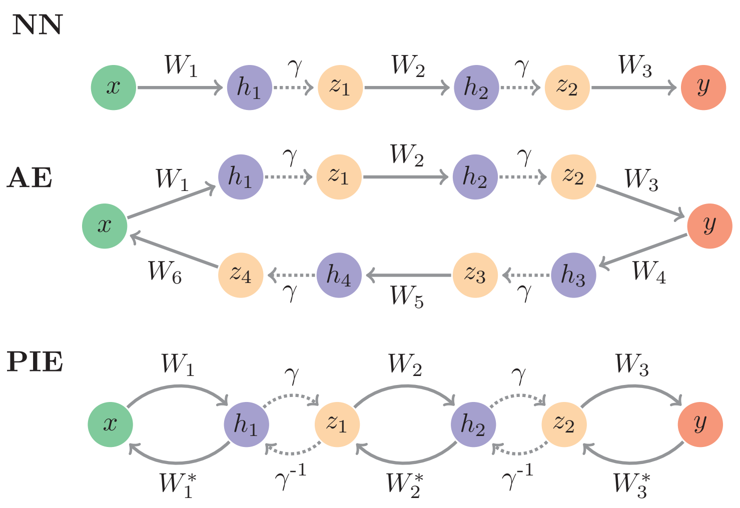

Figure 7.2: Computational graphs of similar three-layer feedforward networks, showing the difference between standard neural networks (above), autoencoders (middle) and invertible encoders (bottom).

Jacobian of the network evaluated for the s samples in the subset, and compute the mean relative error by

$$\kappa_{\text{CNJ}}(\Psi) = \sum_{i=1}^{s} \frac{\|\mathcal{J}_\Psi(x_i)\|}{s\|\Psi(x_i)\|/\|x_i\|}. \tag{7.9}$$

We stress that the software implementation of calculating the Jacobian of a neural network is quite memory intensive and time consuming. Additionally, the implementation in PyTorch had certain issues for more complex gradient computation, which limited our application of this method.

Hyperparameters

The goal of our experiments was to compare general autoencoder models to invertible and pseudo-invertible encoder models. We applied similar hyperparameters for the compared models in each experiment to promote correspondence in the experimental results. Our work could be extended by performing a hyperparameter search over the models or optimization algorithms.

7.2 Assessing Bi-Lipschitzian Activation

To ensure the practical applicability of our proposed activation functions, we conducted an experiment where we compared them to their original counterparts. In the case of our proposed bi-Lipschitzian activation functions (Section 6.2) we first constructed six models with the activation functions seen in Figure 6.1. For the experiment, we trained a three-layer classifier with two hidden layers of 857 neurons on EMNIST data. For optimization we used binary cross-entropy loss on logit-transformed outputs. In this experiment, each network was initialized with

Exp. 7.A – NN Classifier					
Act.	Dim.In	Dim.Hid.1	Dim.Hid.2	Dim.Out	Loss
ReLU	784	857	857	47	LogitBCE
ELU	784	857	857	47	LogitBCE
CELU	784	857	857	47	LogitBCE
BiReLU	784	857	857	47	LogitBCE
BiELU	784	857	857	47	LogitBCE
BiCELU	784	857	857	47	LogitBCE

Exp. 7.B – AE Unsupervised Encoding					
Act.	Dim.In	Dim.Hid.1	Dim.Hid.2	Dim.Out	Loss
BiReLU	784	857	857	47	MSE
BiELU	784	857	857	47	MSE
BiCELU	784	857	857	47	MSE

Table 7.1: Overview of models used for assessment of bi-Lipschitzian activation functions.

Act.	RE	ACC	AC5	CNJ	CNA
ReLU	**0.275**	**0.863**	**0.988**	4.412	7.893
ELU	0.300	0.853	0.987	**3.747**	**6.460**
CELU	0.287	0.859	0.988	3.848	6.821
BiReLU	0.294	0.852	0.987	5.333	13.018
BiELU	0.318	0.845	0.984	4.096	11.222
BiCELU	0.299	0.852	0.987	5.393	15.501

Table 7.2: Results of classification experiment 7.A with bi-Lipschitzian activation.

the same exact weights to minimize any statistical discrepancies. An overview of the model dimensions is provided in Figure 7.2 and Table 7.1 while results of the experiments can be seen in Table 7.2.

While the differences are minor, we noted that the bi-Lipschitz functions performed marginally worse than their counterparts in terms of the classification objective. We also noticed that the RELU functions performed marginally better than the smoother functions. We postulate that this is due to the inherent sparsity properties which [Hah+00] states as an asset in classification tasks.

In terms of stability, we observed that the CNA for the bi-Lipschitzian functions were approximately twice the magnitude of their original counterparts, which aligns with the fact that the Lipschitz constant is doubled compared to the original functions. While the estimated CNJ somewhat corroborates this result, the differences are less pronounced. While the estimation of condition numbers by adverserial perturbation is effective, we still consider CNJ to be a better estimate of the true condition number when it is available. In tests with linear models, we observed that the estimated CNJ was practically identical to the true condition number of the networks.

We also performed testing in an image reconstruction setting. Using the

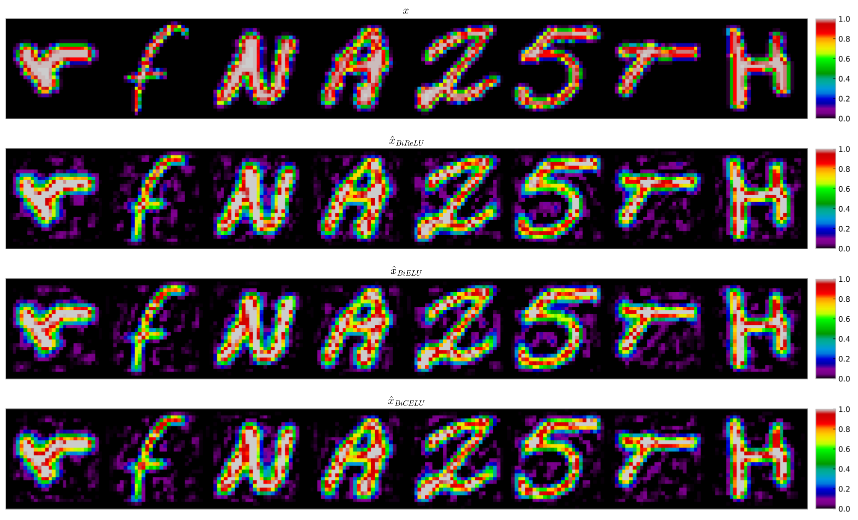

Figure 7.3: Reconstructed images from experiment 7.B with bi-Lipschitzian activation.

Act.	RE	PSNR	SSIM	CNJ	CNA
BiReLU	0.230	21.553	0.800	2.600	7.897
BiELU	0.232	21.517	0.782	**2.576**	7.921
BiCELU	**0.220**	**21.874**	**0.810**	2.804	**6.589**

Table 7.3: Results from image reconstruction experiment 7.B with bi-Lipschitzian activation.

same same three-layer model architecture, we performed unsupervised training with an autoencoder to compare the three bi-Lipschitzian activation functions. A summary of the experimental results is provided Table 7.3. Here, the BICELU function seemed to marginally outperform the BIRELU function, while the BIELU function achieved the lowest score in terms of the image reconstruction metrics. The BIRELU function performed most favorably in terms of the condition numbers estimated via the Jacobian, while the BICELU displayed better stability with respect to adverserial perturbation.

Taken together, these findings suggest that bi-Lipschitzian activation functions perform adequately in classification and image reconstruction tasks. We note that the BICELU and CELU functions used a fixed $\beta = 0.5$, such that the curve around the origin was slightly sharper than the standard parametrization $\beta = 1$.

7.3 Assessing the Dirichlet-Softmax Transform

In Section 6.3 we proposed a bijective simplex transformation to facilitate generative image reconstruction using class labels. To determine how applicable our transform was in practice, we compared the transform with the softmax

Exp. 7.C – NN Classification						
Final.Act.	**Act.**	Dim.In	Dim.Hid.1	Dim.Hid.2	Dim.Out	Loss
Softmax	**ReLU**	784	857	857	47	BCE
DST	**ReLU**	784	857	857	47	BCE

Table 7.4: Overview of models used for the assessment of the Dirichlet-Softmax transformation.

Act.	RE	ACC	AC5	CNJ	CNA
Softmax	**0.243**	**0.858**	**0.988**	**9.843**	**24.195**
DST	0.319	0.831	0.982	10.098	26.697

Table 7.5: Results of experiment with Dirichlet-Softmax transform.

function on the classification task with EMNIST. We expected a decrease in performance and accuracy compared to the standard softmax function. The experiment was performed using the same three-layer network architecture presented in Figure 7.2. We used RELU activation for the hidden layers, while the output layer was then computed with either the softmax activation or the Dirichlet-Softmax transform. An overview of the model dimensions and numerical results from the experiment are listed in Tables 7.4 and 7.5.

While the softmax model clearly performs better, the results for the Dirichlet-Softmax transform exceed our initial expectations, and the transformation performs comparatively well with the standard softmax in terms of accuracy. Considering that the transform is performed with symmetric parametrization, the transform could likely be improved by letting the parameters α_i be estimated or trainable, which is an interesting area for further work. We did note some instability during training, which can be seen in the plot of the training error provided in Figure 7.4.

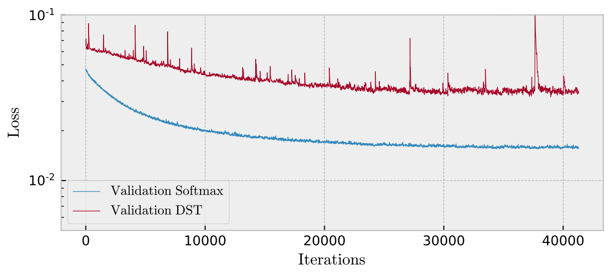

Figure 7.4: Training results for softmax and Dirichlet-Softmax transform. Note the spikes for the Dirichlet-Softmax transform, indicating minor instability.

CHAPTER 8

Experiments with Pseudo-Invertible Encoders

In this chapter, we present the experiments and numerical results we performed for determining the viability of the Pseudo-Invertible Encoder (PIE) model we proposed in Chapter 6. To uncover the potential benefits and limitations of the proposed architecture, most of the experiments compare the PIE architecture to a general autoencoder architecture. To this end, we formulated a set of relevant enquiries.

(i) Do the restrictions of joint parametrization affect reconstruction quality when compared to standard encoder-decoder models?

(ii) How does joint parametrization and underdetermination affect robustness?

(iii) Does the addition of a diagonal weight matrix in the hidden layers improve results?

(iv) Is there a link to SVD, and can PIEs reproduce principal components?

(v) To what extent do the estimated weights in a PIE have orthogonal properties?

(vi) Can PIE models be applied to construct generative network models?

(vii) Can convolutional layers be applied in PIE architectures?

To answer these questions, we constructed several experiments which can be grouped into two main categories; densely connected networks and convolutional networks. In addition, we differentiated between unsupervised and supervised learning objectives, as well as between the three image reconstruction tasks of blurring, image reconstruction from labels, and image upscaling on our three respective datasets; CIFAR, EMNIST, and COCO.

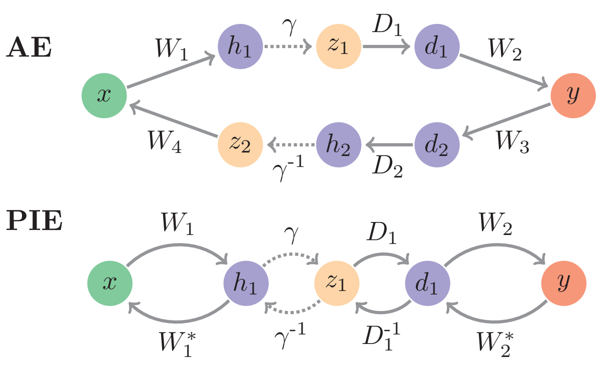

Figure 8.1: Computational graphs of two-layer network architecture comparing autoencoders (top) and invertible encoders (bottom). The edges denoted D_i represent optional diagonal layers featured in some experiments.

Exp.	In.Dim.	Hid.Dim.	Out.Dim.	Act.	Diag.	Gen.	Loss	Data
8.A	784	784	784	ID			MSE	EMNIST
8.B	784	784	784	ID	✓		MSE	EMNIST
8.C	784	784	784	BiCELU			MSE	EMNIST
8.D	784	784	784	BiCELU	✓		MSE	EMNIST
8.E	784	835	128	BiCELU			MSE	EMNIST
8.F	784	835	128	BiCELU	✓		MSE	EMNIST
8.G	784	784	784	BiCELU		✓	MSE + KL	EMNIST
8.H	784	835	128	BiCELU		✓	MSE + KL	EMNIST

Table 8.1: Overview of experiments on unsupervised learning in densely connected networks.

8.1 Unsupervised Learning with Dense PIEs

We designed the first set of experiments to provide a baseline for comparison of pseudo-invertible encoders with standard autoencoders in terms of unsupervised learning using the EMNIST dataset. The models were trained concurrently on the same mini-batches with adaptive moment estimation of weights (ADAM) [KB17] using MSE loss. All models were constructed using two single layer networks with bias and activation only in the hidden layer. A figure and general overview of the experiments is provided in Figure 8.1 and Table 8.1. The dimensions for each layer are equal in PIE and AE models, thus the relationship between the dimensionality of the parameter space each model is $\dim(\Theta_{\text{AE}}) = 2\dim(\Theta_{\text{PIE}})$. We considered methods for balancing this discrepancy, but decided that keeping similar architectures was more conducive to comparative results in this experiment. We were especially interested in the effect of diagonal layers (Equation (4.27)) on the models, which was introduced in experiments 8.B, 8.D, 8.F. Theoretically, these should help mitigate any issues caused by isometry in the weights as discussed in Section 4.3, however possibly at the cost of general robustness.

Exp.	**Model**	RE	PSNR	SSIM	CNJ	CNA
8.A	**PIE**	0.019	42.088	0.990	1.038	4.179
	AE	0.025	40.139	0.975	1.252	4.090
8.B	**PIE**	**0.018**	**42.223**	**0.990**	**1.032**	4.142
	AE	0.024	40.310	0.975	1.292	**4.037**
8.C	**PIE**	0.045	35.168	0.951	1.160	4.054
	AE	0.038	36.831	0.954	1.703	5.221
8.D	**PIE**	0.043	35.486	0.957	1.154	4.092
	AE	0.038	36.790	0.953	1.728	4.129
8.E	**PIE**	0.159	24.544	0.849	1.408	4.995
	AE	0.149	24.968	0.865	2.135	7.056
8.F	**PIE**	0.156	24.586	0.854	1.453	5.113
	AE	0.149	24.963	0.862	2.422	7.908
8.G	**PIE**	0.342	17.867	0.685	1.552	5.509
	AE	0.345	17.780	0.685	1.961	6.823
8.H	**PIE**	0.355	17.550	0.674	1.674	5.971
	AE	0.337	17.980	0.701	1.866	6.569

Table 8.2: Results of experiments on unsupervised learning in densely connected networks. The overall best results are highlighted in bold for readability, while the comparatively best results between models are shaded in gray.

The latent dimension in the undercomplete models was decreased from 784 to 128. All models were trained over 12 epochs with a learning rate of 10^{-4}, which we visually confirmed to be enough to ensure some level of convergence for all models in the experiment. Based on the results from Section 7.2, we applied BICELU in all non-linear models, including using inverse BICELU activation in the decoder of the AE models to keep the results as comparative as possible. The numerical results are summarized in Table 8.2.

Robustness

A few interesting observations can be made from the results. We noted that both estimates of relative condition numbers CNJ, CNA remained relatively consistent between models. The addition of diagonal layers seemed to have little impact on the overall robustness of the models, both in terms of CNA and CNJ, as can be observed in experiments 8.B, 8.D, and 8.E. As there is no constraint imposed on the parameters in these layers other than the initialization of values close to 1, we expected some instability for values closer to zero, however this turned out not to manifest in the models, indicating that the addition of diagonal layers are not necessarily a source of instability on unsupervised learning tasks. Overall, we noticed that the PIE models seem to be more robust than the respective AE models, which is as expected from the properties of the invertible encoder models outlined in Observation 6.1.1.

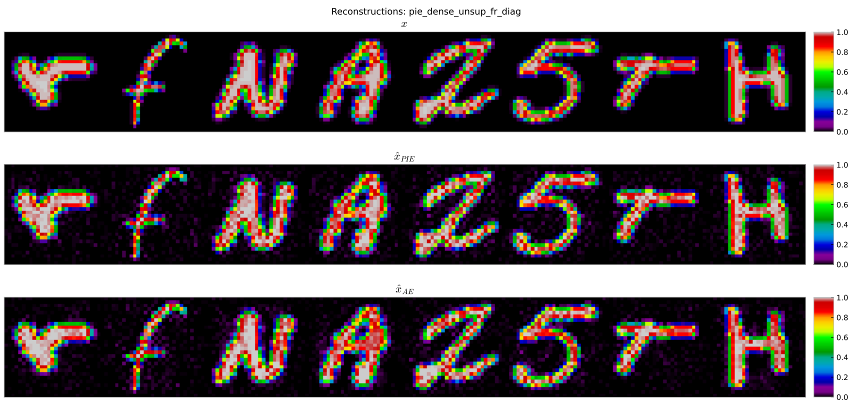

Figure 8.2: Reference image (above) and reconstructed images from experiment 8.D, with full rank PIE (middle) and AE (below) with diagonal layers.

Improved Reconstructions with Diagonal Layers

We noted a particularly interesting disparity in the image reconstruction metrics. While the computed RE and PSNR naturally agree across all models, the results of SSIM did turn out to disagree with these scores in one case. In general, we observed that while the addition of diagonal layers (8.B, 8.D, and 8.F) did marginally improve the structural similarity of the pseudo-invertible encoders, we noted the opposite effect on the autoencoder models. The improvement is particularly prevalent in the nonlinear full rank models (8.C, 8.D), where the addition of a diagonal layer resulted in an improvement of SSIM, corresponding to the previously mentioned disparity. This was enough for the PIE to overtake the AE model as the best performing full-rank nonlinear model in our experiments with unsupervised learning on EMNIST. The reconstructions from experiment 8.D can be seen in Figure 8.2.

As mentioned in Section 7.1, the observed disparity in model 8.D aligns with the expected behaviour, as the SSIM is generally considered to be a more reliable metric for evaluating image reconstruction errors. As such, the results imply that the PIE model with diagonal weights should be preferred, as it performs better with respect to the two most important metrics – SSIM and CNJ. The fact that the addition of diagonal layers has a negative effect on AE models also aligns with our intuition (Section 4.3). As the weights of the PIE models are ideally isometric, the introduced diagonal layer can serve to independently scale the outputs, which will have little effect on an AE model as it has the necessary freedom to estimate mutually orthogonal weights in the encoder and decoder without isometry. Hence, the diagonal weights serve as parameter redundancy, which could be the cause of the weaker results.

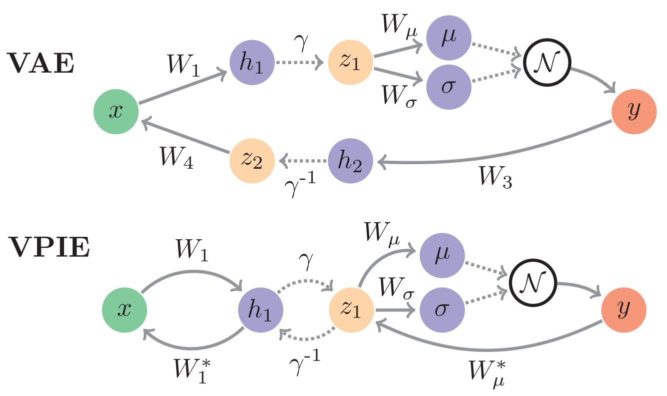

Figure 8.3: Computational graphs of generative two-layer network architectures.

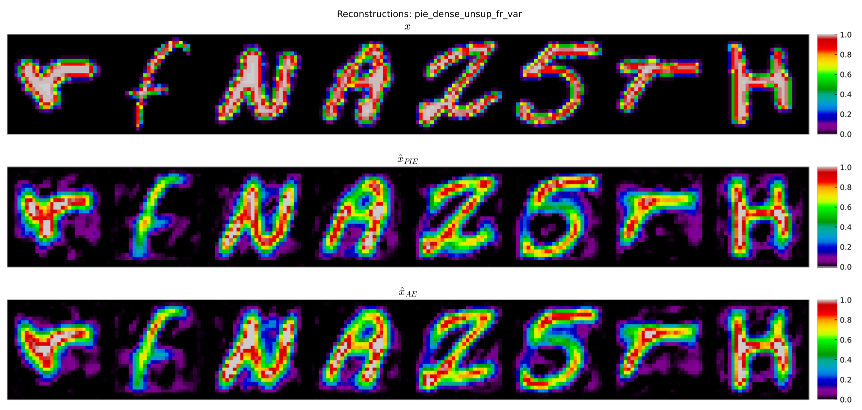

Figure 8.4: Reference image (above) and reconstructed images from experiment 8.H with undercomplete variational PIE (middle) and AE (below).

Generative Modelling

For the experiment with generative models, we modified the final layer to be stochastic (Definition 4.3.7) using the reparametrization trick, and added KL-divergence (Definition 4.4.5) regularization to the objective functions to encourage a normal mean field approximation in the latent space, corresponding with a VAE model (Definition 4.4.7). An example of reconstructions is featured in Figure 8.4. The results show that PIE models can be trained using variational inference. In the full rank model, the models achieved approximately equal results (8.G), but in the undercomplete example (8.H), the AE model achieved better results and actually improved compared to the full rank model. While this seems surprising, we know there is redundancy in the images, so the full rank models are necessarily overparametrized. More surprising is the fact that the results indicate that the optimal ratio of input- and output dimensions is closer to 784:128 than 784:784, which is lesser than expected even for the comparatively simple EMNIST data.

Network	$\text{err}_{\text{REID}}(W_1^* W_1)$	$\text{err}_{\text{REID}}(W_2^* W_2)$	z_1	z_2
PIE	**0.569**	0.610	**−98.987**	−98.313
AE (encoder)	0.793	0.712	88.374	−79.646
AE (decoder)	0.645	**0.604**	−96.972	**−98.435**

Table 8.3: Results from randomly initialized Gramian weight matrices on models from experiment 8.A. Lower scores indicate stronger orthogonality properties. The z-scores are computed from 10,000 randomly initialized matrices and compared with the estimated weights after training.

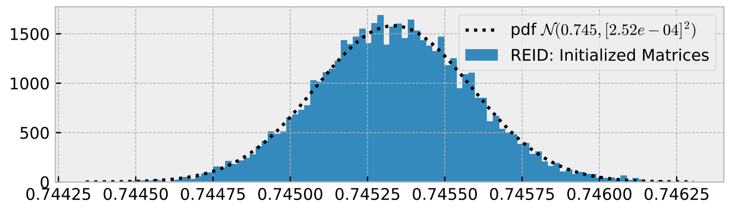

Figure 8.5: Distribution of relative error w.r.t. identity (REID) scores for Gramians of individual layers in model A. The errors are distributed approximately normal, with $\epsilon \sim \mathcal{N}\left(.745, (2.52 \times 10^{-4})^2\right)$.

Orthogonality of Estimated Weights

PIE networks are more or less dependent on the orthogonality of the estimated weights. We initially considered the relative error of the Gramians of the estimated weights w.r.t. identity (REID, see Table 1.5) as a useful measure of orthogonality. However, as any matrix in $\mathbb{R}^{n \times n}$ of iid. random variables will be asymptotically orthogonal as $n \to \infty$, a randomly initialized weight matrix will exhibit some degree of inherent orthogonality. There was thus a need for a quantitative measure of the orthogonality of the estimated network weights, to eliminate the possibility that any such property was merely inherited from the random weight initialization.

We conducted an experiment where 10,000 random matrices were initialized using the uniform distribution described in [He+15] and the relative error of their Gramians with respect to the identity was computed using the Frobenius norm (REID). The results can be seen in Table 8.3 and Figure 8.5. The distribution of the relative error was approximately normal, and we compared the results to the estimated weights from the linear models in experiment 8.A. We found that the REID of the estimated weights of the PIE and the decoder of the AE were far into the left tail of the distribution, indicating that the weights were significantly more orthogonal than randomly initialized matrices. Interestingly, the weights of the encoder in the AE model exhibited significantly decreased orthogonal properties compared to the randomly initialized weights. The fact that the weights of the decoder of a linear AE is more closely orthogonal than randomly initialized weights indicates that the decoder networks of autoencoders approximate orthogonal bases, and could be a possible reason for why pseudo-invertible encoders can be effectively implemented with only implicit constraints.

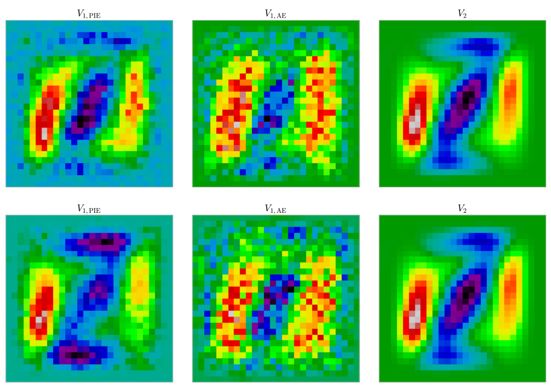

Figure 8.6: Reproduced principal vectors of PIE and AE models without diagonal (above) and with diagonal (below). The first principal vector for the PIE (left) and the AE (middle) is compared to ground truth (right) computed via SVD on EMNIST.

Reproducing Principal Vectors

In [BH89] the authors demonstrated a similarity between linear autoencoder models and principal component analysis. While this result has been reported slightly indiscriminately in both linear and nonlinear contexts, [Kun+19] demonstrated that linear autoencoders can approximate the eigenvectors of the covariance matrix of centered data with specific training regimes and Tikhonov regularization.

As noted in Observation 6.1.1 PIE models should theoretically share similarities to singular value decomposition. We wanted to see if these results were applicable to pseudo-invertible encoder models, so we constructed linear two-layer PIE and AE models, which were trained on EMNIST data in two separate processes. Firstly, the models were trained using masked gradients over 32 epochs, where one row and one column of each layer were sequentially updated using gradient descent to enforce an ordering of the principal vectors. After this initial training, the network received additional training over 8 epochs without masks for final convergence.

Our results indicate that unsupervised PIE and AE models are both able to replicate the most significant principal vectors of the data, however, the results are slightly more significant in the PIE model, which achieved an SSIM of 0.653 to the real principal vector in contrast to 0.608 for the AE model. We were also interested in how the introduction of a diagonal matrix before the last layer would affect the result, and whether the values would converge to the singular values of the data. While we did not observe this behaviour, we noted that the introduction of a diagonal matrix had a notable effect on the resulting principal vectors, which can be observed in Figure 8.6.

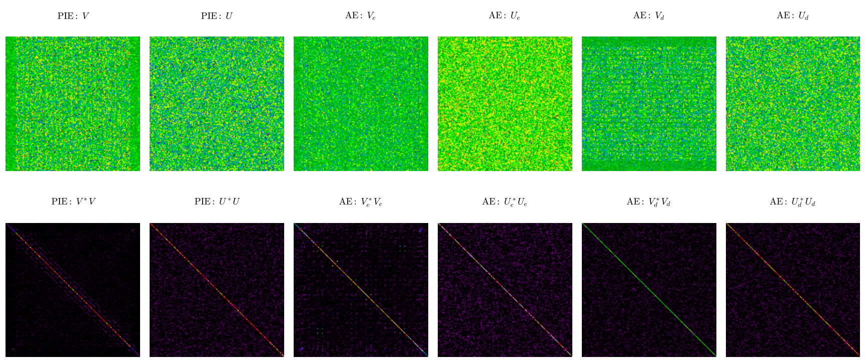

Figure 8.7: Visualizations of weights (above) and respective Gramians (below) of estimated weights in the linear full-rank model (8.A).

Summary: Unsupervised Dense PIEs

Taken together, the results suggest that the differences of the reconstructed images are minor for unsupervised training with PIE and AE models. While this may at first seem unremarkable, recall that the dimensionality of the parameter space of an AE model in this experiment is approximately double the respective dimensionality of a PIE model, suggesting that a PIE model could be effectively twice as memory efficient as the respective AE model for unsupervised learning tasks in dense networks, however, we note that this result is from rather simple tests and does not necessarily scale in terms of model complexity.

As noted in Section 4.4, a concern with full-rank autoencoders is that their architecture encourages the weights to approximate the identity, resulting in trivial models. We did however, not observe any such behaviour in the network weights of either PIE nor AE models. Instead, the PIE model as well as the decoder of the AE model converged to approximately orthogonal weights. Visualizations of the estimated weights in model 8.A as well as their respective Gramians can be seen in Figure 8.7.

In terms of robustness, we observed that PIE models achieved better CNJ scores in all comparisons, while achieving better CNA scores in all but the linear models. This indicates an overall better conditioning of the models and improved robustness to adverserial attacks in unsupervised learning tasks, as was expected and theoretically motivated by Property 6.1.1.iv.

The addition of diagonal layers seems to improve the performance of PIE models, as we theoretically hypothesised in Section 4.3. Additionally, we were able to reproduce the principal components of the data, confirming the link between invertible encoders and singular value expansion in Property 6.1.1.i.

Exp.	In.Dim.	Hid.Dim.	Out.Dim.	Act.	Gen.	Patch Size	Loss	Data
8.I	32^2	32^2	32^2	BiCELU		1×1	LHC	CIFAR
8.J	32^2	32^2	32^2	BiCELU		4×4	LHC	CIFAR
8.K	32^2	$2 \cdot 32^2$	32^2	BiCELU		1×1	LHC	CIFAR
8.L	32^2	128^2	32^2	BiCELU		4×4	LHC	CIFAR
8.M	28^2	Multi	47	BiCELU	✓	Multi	LHC+KL	EMNIST
8.N	28^2	Multi	28^2	BiCELU	✓	Multi	LHC+KL	EMNIST
8.O	384^2	Multi	96^2	BiCELU		Multi	LHC	COCO
8.P	384^2	Multi	384^2	BiCELU	✓	Multi	LHC+KL	COCO

Table 8.4: Overview of experiments on supervised learning with dense and seperable PIE networks. Experiment 8.M to 8.P feature deeper layer structures, which are detailed in Table 8.5.

8.2 Supervised Learning with Dense and Seperable PIEs

The second set of experiments was designed to investigate the feasibility of PIEs in a supervised setting by comparing it to supervised autoencoder models. Recall the discussion in Section 4.4, where we introduced learning objectives Equations (4.34) and (4.35), and the implications of Proposition 4.4.4, which implies that the source of ill-posedness can influence the apparent difficulty of a learning task for these models. As autoencoders maximize the lower bound on mutual information [Vin+10] this can lead to suboptimal results when training with the standard autoencoding objective. Therefore the experiments apply the objective proposed in Equation (4.35) for all but the generative models.

As previously, all models were trained concurrently with the ADAM optimizer using a learning rate of 5.0×10^{-5}, and the models of each experiment were trained concurrently over the same minibatches for 12 to 25 epochs, depending on the convergence of each model. Note that we had to omit the Jacobian condition numbers CNJ for the supervised models, as we had problems computing these metrics due to issues in the implementation of Jacobian computations in PyTorch.

Model Architectures

The models in experiments 8.I through 8.L were constructed with the same two-layer architecture as in Section 8.1, visualized in Figure 8.1. The models in 8.K and 8.L were constructed with overdetermination in the hidden layer, while the models in experiments 8.M through 8.P use deeper multilayer structures. We again emphasize that the dimensionality of the parameter space of the models in experiments 8.I through 8.L is $\dim(\Theta_{\text{AE}}) = \dim(\Theta_{\text{PIE}})$ for all models without diagonal layers. However, in experiments 8.K through 8.L we compensate for this by modifying the architecture of the AE models. An overview of all models is provided in Tables 8.4 and 8.5. We emphasize that the models in experiments 8.N and 8.P apply separate subnetworks for computing a target output y, as well as a latent output z which is used to probabilistically augment the dimensions of the output using a variational mean field approximation with a standard normal prior, similar to the idea proposed in [Ard+18]. Figures 8.9 and 8.10 illustrates the network architecture for these models. In Table 8.5, the dimensions of this split are denoted by lettering to indicate the association of each subnetwork.

Exp. 8.M – PIE				
Layer	No.In	No.Out	Dim.In	Dim.Out
1	784	784	1	1
2	784	196	1	4
3	196	196	4	4
4	196	49	4	16
5	49	49	16	16
6	49	784	16	1
7	784	1024	1	1
8	1024	47	1	1

Exp. 8.M – AE				
Layer	No.In	No.Out	Dim.In	Dim.Out
1	784	196	1	4
2	196	196	4	4
3	196	49	4	16
4	49	784	16	1
5	784	784	1	1
6	784	47	1	1

Exp. 8.N – PIE				
Layer	No.In	No.Out	Dim.In	Dim.Out
1	784	784	1	1
2	784	196	1	4
3	196	196	4	4
4	196	196	4	4
5	196	49	4	16
6	49	49	16	16
7	49	784	16	1
8y	47	47	1	1
9y	47	47	1	1
8z	737	737	1	1
9z	737	737	1	1

Exp. 8.N – AE				
Layer	No.In	No.Out	Dim.In	Dim.Out
1	784	196	1	4
2	196	196	4	4
3	196	196	4	4
4	196	49	4	16
5	49	784	16	1
6y	47	47	1	1
7y	47	47	1	1
6z	737	737	1	1
7z	737	737	1	1

Exp. 8.O – PIE				
Layer	No.In	No.Out	Dim.In	Dim.Out
1	256	256	576	484
2	256	256	484	361
3	256	256	361	196
4	256	256	196	36

Exp. 8.O – AE				
Layer	No.In	No.Out	Dim.In	Dim.Out
1	256	256	576	225
2	256	256	225	144
3	256	256	144	64
4	256	256	64	36

Exp. 8.P – PIE				
Layer	No.In	No.Out	Dim.In	Dim.Out
1	256	1024	576	144
2	1024	256	144	576
3y	256	256	39	39
4y	256	256	39	39
3z	256	1280	540	108
4z	1280	256	108	540

Table 8.5: Overview of layer structures in experiments 8.M and 8.N. The two left columns indicate the number of patches, and the two right columns indicate the patch dimensions. Layers denominated with letters indicate a split in the network to produce different outputs.

All networks applied the BICELU activation function in all hidden layers. In addition, the models in experiments 8.M and 8.N apply the Dirichlet softmax activation on the target outputs to estimate a discrete probability distribution over the classes. While certain models applied only dense layers (8.I, 8.K), most models were constructed with separable patch layers, introduced in Definition 4.3.3. For the image upscaling task, the image resolution of 384×384 pixels makes applying dense networks infeasible due to the size of the parameter space, while a separable-patch network can be made much more efficient. Lastly, experiments 8.I through 8.N both with and without diagonal layers in the PIE models to verify our earlier observations in Section 8.1.

Exp.	**Model**	RE (10^{-2})	PSNR	SSIM($\hat{x}$)	SSIM($\hat{y}$)	CNA($\hat{x}$)	CNA($\hat{y}$)
8.I	**PIE**	5.671	32.514	0.917	0.985	**3.612**	12.168
	PIED	5.615	32.682	0.919	0.986	3.673	12.808
	AE	5.235	33.630	0.924	0.993	4.000	16.159
8.J	**PIE**	9.490	27.872	0.922	0.954	4.568	12.521
	PIED	8.838	28.474	0.928	0.957	6.161	20.228
	AE	8.514	28.725	0.927	0.961	5.308	25.806
8.K	**PIE**	5.604	32.398	0.921	0.983	3.935	12.765
	PIED	6.775	30.441	0.922	0.984	3.896	13.983
	AE	**4.904**	34.054	0.930	0.995	3.743	16.665
8.L	**PIE**	7.151	30.863	0.935	0.978	4.399	**11.920**
	PIED	5.905	34.400	**0.944**	0.990	4.354	20.298
	AE	5.127	**36.629**	0.943	**0.998**	4.537	23.811

Table 8.6: Results of experiments on supervised learning in dense networks. The models denoted PIED include a diagonal layer. RE and PSNR are averaged between blurred images ($\hat{y}$) and deblurred images ($\hat{x}$).

Deblurring

The purpose of this experiment is to see how PIE models performed in terms of deblurring tasks, where we want the encoder to approximate the blur operator and the decoder to approximate the deblurring operator. We refer to the blurred images as y and the original images as x. In the setting of invertible networks, we want to find an approximate bijective map between these two image domains. For experiments 8.J, 8.L, we used a separable patch network which significantly reduced the parameter space. In all experiments, we compared three networks; two PIE models with and without diagonal layers, and one AE model without diagonal layers. In the separable patch networks, the diagonals were applied to both left- and right operators. To avoid confusion, we will refer to the models with added diagonals as PIED.

An overview of the numerical results are shown in Table 8.6. Our experiments show that the PIED architecture on average performs comparative to the AE models for deblurring mappings. In the examples with overcomplete separable-patch networks, we achieve the best deblurring results with a PIED model, by a very slight margin. The reconstructed images from experiment 8.L can be seen in Figure 8.8. Note that the AE models seemed to outperform the PIE networks in terms of approximating blurred images across all experiments. The overcomplete separable-patch architecture achieved the best approximation to the blur operator, where the AE model achieved a SSIM score of 0.998. The improvement in the deblurring task for the PIE models and subsequent decrease in the blurring objective can be theoretically justified by sensitivity imbalance, discussed in Proposition 4.4.4 and Definition 4.4.3.

In terms of robustness, the CNA results indicate that the PIE models are better conditioned than the AE models, indicating that they are inherently less sensitive to perturbations, which aligns with the theoretical motivation of Parseval regularization. However, we note that PIE models without added diagonal layers generally displayed better conditioning for inverse computations. Thus, the improvement in accuracy seems to come at the cost of decreased

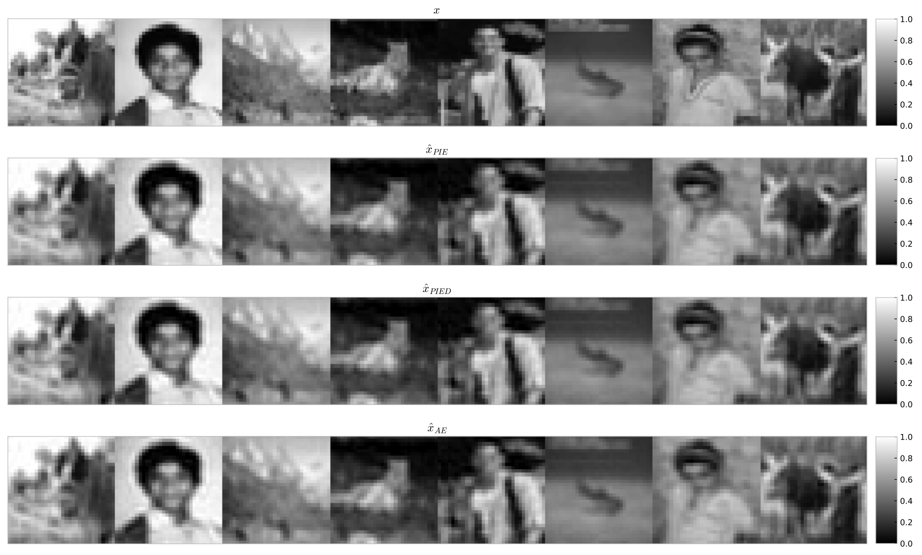

Figure 8.8: Reference image (above) and deblurred images (below) from experiment 8.L, where the PIED model achieved the highest SSIM score by a small margin.

robustness. We also note that the seperable patch networks generally show worse conditioning than fully dense networks.

Overall, we consider the performance of the deblurring models to be modest. We noted that the separable patch models tended to produce square grid-like artifacts in the reconstructions. These are expected byproducts of the partitioning of the image (see Figure 4.5) which adds boundaries between each patch. Different methods for smoothing patches by overlapping or introducing intermediate convolution can be considered for these layer types.

Classification and Conditional Image Generation

Next, we wanted to examine how generative modelling could be combined with invertible neural networks to generate images from image labels. The task is to map the input images x to a probability distribution over the labels $y \mid x \sim \text{Dirichlet}(\alpha)$ with- or without latent variables $z \mid x, y \sim \mathcal{N}(0, I)$. Unlike the previous experiments, we let the dimensions of the parameter space PIE and the AE be approximately equal in these experiments, as we wanted to see how this would impact the model performance. In effect, these models are thus applications of the ideas proposed in [Ard+18] discussed in Section 5.2.

In experiment 8.M, we map between x and y directly using a stochastic layer and the symmetric Dirichlet-Softmax transform outlined in Proposition 6.3.4. The models in experiment 8.N are latent variable models, for which the architecture is derived from a conditional variational autoencoder (Definition 4.4.7) with certain modifications. In addition to the classification sub-network, we apply another sub-network using conditional additive coupling layers (Definition 4.3.4) conditioned on the estimated labels y. Graphs depicting the model architec-

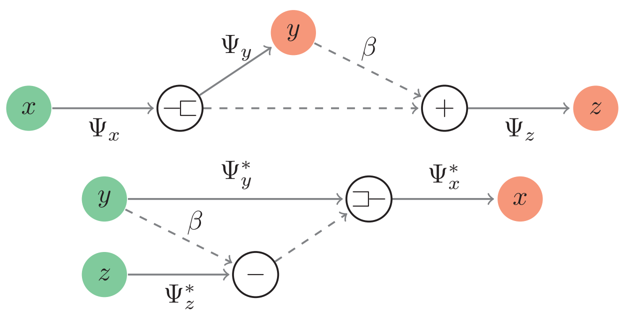

Figure 8.9: Simplified architecture of conditional encoder (above) and decoder (below) in experiment 8.N. The network consists of three sub-networks Ψ_x, Ψ_y, Ψ_z and conditional additive coupling layers β. The split operator is denoted $-\!\sqsubset : \mathbb{K}^n \to \mathbb{K}^{n-k} \times \mathbb{K}^k$ where $1 < k < n$, while the concatenation operator $\sqsupset\!- : \mathbb{K}^{n-k} \times \mathbb{K}^k \to \mathbb{K}^n$ acts as its inverse.

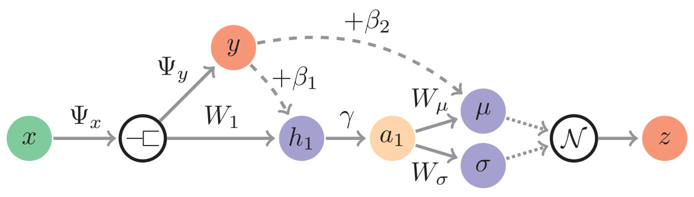

Figure 8.10: Detailed architecture of forward pass through subnetwork Ψ_z taken from model for conditional image generation from labels in experiment 8.N. The conditional additive coupling layers β_1, β_2 conditions on y and shifts the distribution accordingly. Note that activated neurons are denoted a_i to differentiate between the latent output z.

ture for the encoders is provided in Figures 8.9 and 8.10. We note that the sub-networks Ψ_z, Ψ_y both contain stochastic sampling layers (Definition 4.3.7) as their final layers, acting as variational autoencoders. The results in Table 8.7 indicate that the PIE model perform better than the AE counterparts in all metrics except for RE and SSIM. However we note that these metrics are computed without sampling, so the result of the generative process is better evaluated visually.

Exp.	**Model**	RE	ACC	AC5	SSIM	KL	CNA($\hat{x}$)	CNA($\hat{y}$)
8.M	**PIE**	0.719	0.847	0.965	0.706	0.858	11.249	14.228
	AE	0.644	0.846	0.963	0.767	0.888	14.560	14.236
8.N	**PIE**	0.583	**0.851**	**0.980**	0.780	**0.265**	**2.276**	**4.962**
	AE	**0.455**	0.805	0.969	**0.877**	0.337	6.931	4.962

Table 8.7: Results of experiments on conditional image generation with EMNIST.

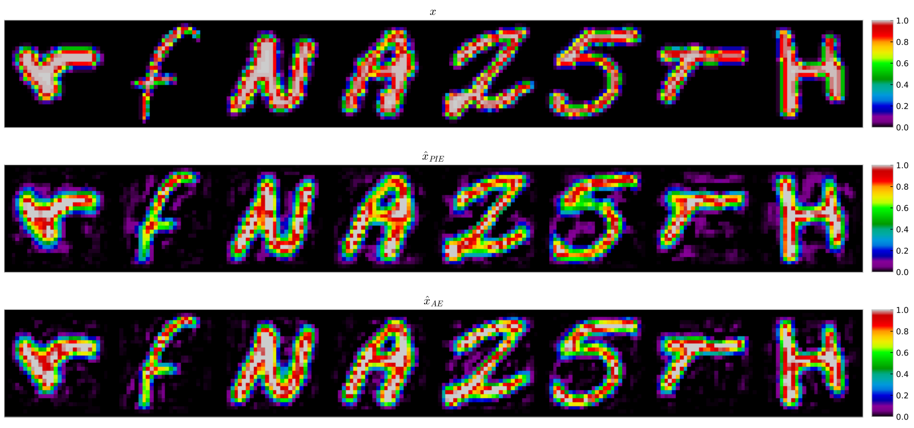

Figure 8.11: Image reconstructions from experiment N without sampling.

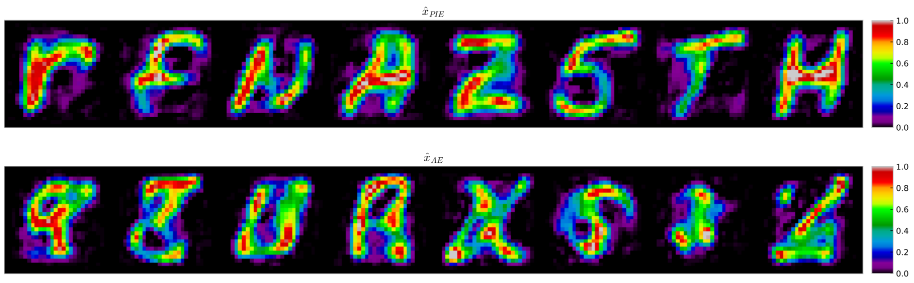

Figure 8.12: Image synthesis from experiment N sampled from $z \sim \mathcal{N}(0, I)$ conditioned on class labels y computed from the original.

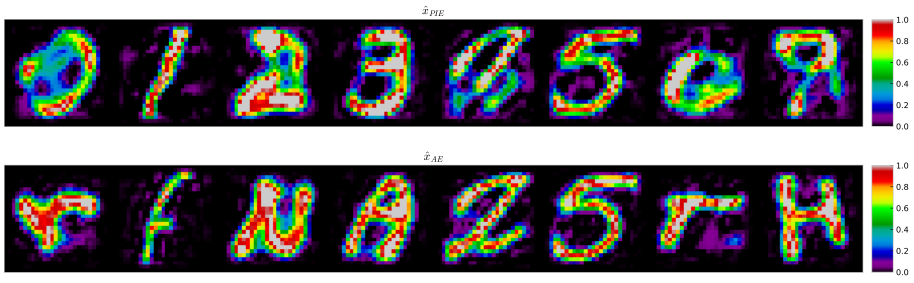

Figure 8.13: Image synthesis from experiment N using labels $y = (0, \ldots, 7)$ conditioned on the latent vector z computed from the original.

Exp.	**Model**	RE (10^{-2})	PSNR	SSIM($\hat{x}$)	SSIM($\hat{y}$)	CNA($\hat{x}$)	CNA($\hat{y}$)
8.O	**PIE**	7.527	**33.811**	0.676	**0.998**	**0.051**	0.810
	AE	9.097	28.432	0.666	0.983	0.052	0.810
8.P	**PIE**	**7.417**	31.170	**0.698**	0.985	0.052	**0.247**

Table 8.8: Results from image upscaling experiments with dense encoders.

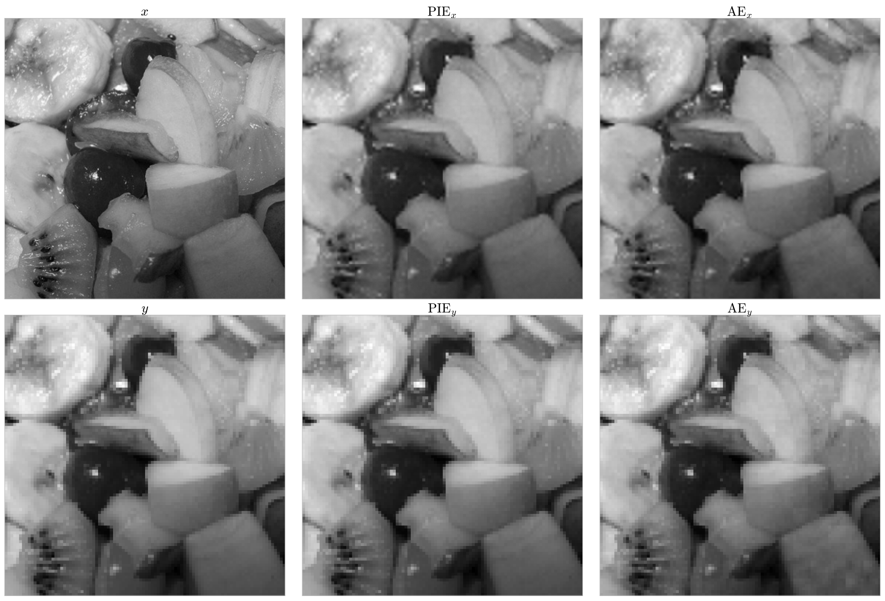

Figure 8.14: Image reconstructions from image upscaling experiment 8.O.

Figures 8.11 to 8.12 demonstrate that the PIE model was able to approximate the target distribution, while the AE model did not. The synthesized images show that the latent variables z parameterize the shape of the handwritten characters, while the labels y control which character to synthesize. The increase in accuracy for the PIE models in both experiments indicate that the network actively uses the classified labels to reduce the overall objective. Additionally, we note that the PIE models achieved both lower KL-divergence and improved stability compared to the respective AE models.

Image Upscaling

Next, we look at how the models perform in terms of image upscaling, where we are interested in mapping an a downsampled image y to the original image x. In our experiment, the original images are 384×384 pixels, and the downsampled images are 96×96 pixels, corresponding to a downsampling ratio of 16:1. Fully dense networks become infeasible for images of this size, so all models in the experiment are constructed using separable patch networks (Definition 4.3.3). Similar to experiments 8.M and 8.N, we design our models such that the dimensions of the parameter space is approximately equal for PIE and AE models.

Experiment 8.O uses a straightforward supervised encoder-decoder structure. In experiment 8.P, we wanted to see if augmenting the latent space with a generative model would improve results. As the model is constructed with a similar architecture as experiments 8.M and 8.N (see Figures 8.9 and 8.10), we only trained a PIE model in this experiment. Table 8.8 shows that the PIE model achieved significantly better scores in experiment 8.O than the AE model, particularly in terms of SSIM for the downsampled images. We also observed a slight improvement in SSIM score in the generative model in experiment 8.P. From our observations on the synthesised images, the model seems to apply the latent variable to improve the texture of the image, which slightly remedies the square artifacts from the separable patch architecture. Surprisingly, all models seemed to be relatively well conditioned.

Summary: Supervised Dense PIEs

From our experiments, we can conclude that PIE models are viable for supervised learning tasks and that they have practical applications for upscaling and conditional image synthesis from image labels, where they outperform the respective AE models. This indicates that the ideas from [Ard+18] can be applied to invertible encoder networks and corroborates the claim that invertible neural networks can be trained in a semi-generative setting without significantly impacting the performance of the model. In the deblurring task the results were more unclear, and neither the PIE nor AE models yielded any particularly impressive results.

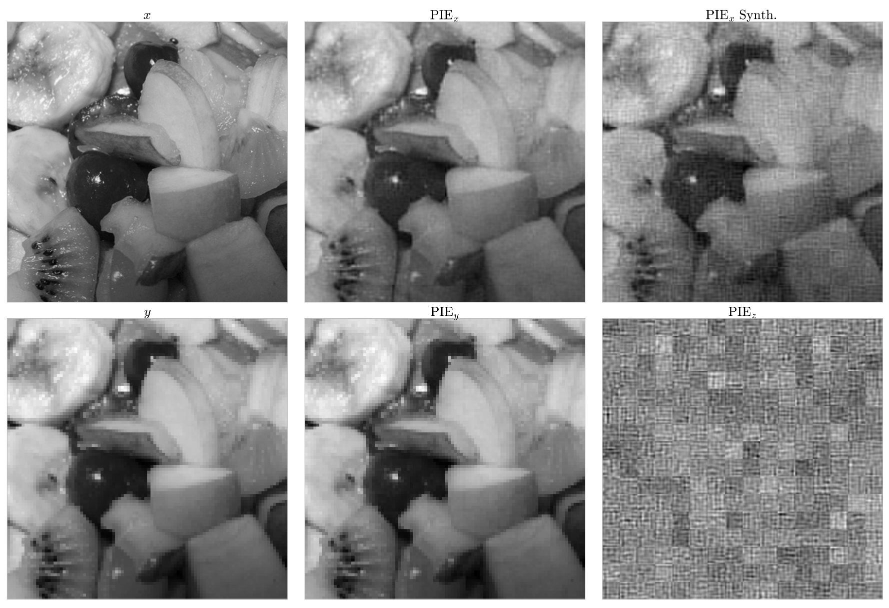

Figure 8.15: Image reconstructions and synthesis from experiment P with superresolution. On the top right we see a synthesized image, giving the reconstructed image a different texture. On the bottom right we see the randomly synthesized texture isolated from the image.

Exp.	**Model**	Features	Act.	Diag.	Loss	Data
8.Q	**PIE**	$(1, 1, 1, 1, 1)$	BiCELU		LHC	CIFAR
	AE	$(1, 1, 1, 1, 1)$	BiCELU		LHC	CIFAR
8.R	**PIE**	$(1, 3, 6, 3, 1)$	BiCELU		LHC	CIFAR
	PIELD	$(1, 3, 6, 3, 1)$	BiCELU	Lin.	LHC	CIFAR
	PIESD	$(1, 3, 6, 3, 1)$	BiCELU	Sep.	LHC	CIFAR
	AE	$(1, 3, 6, 3, 1)$	BiCELU		LHC	CIFAR
8.S	**PIE**	$(1, 6, 12, 24, 12, 6, 1)$	BiCELU		LHC	COCO
	AE	$(1, 4, 9, 16, 9, 4, 1)$	BiCELU		LHC	COCO
8.T	**PIE**	$(1, 6, 12, 24, 12, 6, 1)$	BiCELU	Sep.	LHC	COCO
	AE	$(1, 4, 9, 16, 9, 4, 1)$	BiCELU	Sep.	LHC	COCO

Table 8.9: Experiments with convolutional pseudo-invertable encoders.

Exp. 8.R – PIE					
Layer	In Ch.	Out Ch.	Kernel	Pad.	Stride
1	1	3	3×3	1×1	1×1
2	3	6	5×5	2×2	1×1
3	6	3	5×5	2×2	1×1
4	3	1	3×3	1×1	1×1

Exp. 8.S, 8.T – PIE					
Layer	In Ch.	Out Ch.	Kernel	Pad.	Stride
1	1	6	3×3	1×1	1×1
2	6	12	3×3	1×1	2×2
3	12	24	5×5	2×2	2×2
4	24	12	7×7	3×3	1×1
5	12	6	5×5	2×2	1×1
6	6	1	3×3	1×1	1×1

Table 8.10: Overview of layer structure in experiments 8.R, 8.S, 8.T. Note that experiment 8.Q follows the same layer structure as 8.R without channel expansion, i.e. a single input and output channel for all convolution operators.

8.3 Supervised Learning with Convolutional PIEs

To conclude our experiments with pseudo-invertible encoders, we looked at the application of convolutional layers in PIE models. We constructed two experiments similar in form to the experiments with deblurring and upscaling in the Section 8.2. The training process was conducted with identical hyperparameters and optimization as in the experiments on dense networks. The models in experiments 8.Q, 8.R were constructed with a kernel size of 3×3 with 1×1 padding in all layers as well as identical input and output features, so the dimensionality of the parameter space in the AE model is approximately double that of the PIE model. Experiments 8.S, 8.T used different numbers of features such that the models had roughly similar dimensionality in their respective parameter spaces. An overview of the experiments is given in Table 8.9, while the parameters of the layer structure in experiments 8.S, 8.T is given in Table 8.10.

We looked at different options for a convolutional counterpart to diagonal layers. A linear diagonal layer applied over the flattened image would tie up a large number of parameters, while separable diagonal layers would be less effective, but have a lower dimensional parameter space. Both layer types were tested in experiment 8.R, but due to the high dimensionality of the images of the COCO dataset, seperable diagonal layers are applicable and were applied in experiment 8.S.

Exp.	**Model**	RE (10^{-2})	PSNR	SSIM($\hat{x}$)	SSIM($\hat{y}$)	CNA($\hat{x}$)	CNA($\hat{y}$)
8.L	**PIE**	7.151	30.863	0.935	0.978	4.399	11.920
	PIED	5.905	34.400	0.944	0.990	4.354	20.298
	AE	5.127	36.629	0.943	0.998	4.537	23.811
8.Q	**PIE**	8.684	28.579	0.883	0.979	**3.583**	**0.996**
	AE	3.964	37.952	0.954	0.999	3.627	24.585
8.R	**PIE**	3.387	39.716	0.965	0.999	6.481	15.545
	PIELD	3.334	39.421	**0.969**	0.999	13.928	31.597
	PIESD	3.339	**39.981**	0.965	0.998	14.241	36.827
	AE	**3.334**	39.143	0.966	**1.000**	5.280	37.334

Table 8.11: Results of experiments on deblurring with convolutional PIE networks (8.Q) compared to the best experiment with dense networks (8.L). RE and PSNR are averaged between blurred images ($\hat{y}$) and deblurred images ($\hat{x}$).

For transposed circular convolution with stride one, the standard convolution operation can be applied using a lexicographical reversed kernel as explained in Section 4.3 and Appendix A.8. However, this does not work with strides greater than one. In this case, a transposed convolution operator is used instead, however in the PyTorch software, transposed convolution does not support circular boundary conditions. This meant we had to apply constant boundary conditions with zero padding in experiments 8.S, 8.T, which does not guarantee orthogonality in the estimated weights.

Deblurring with Convolutional PIEs

The results of the deblurring experiments can be observed in Table 8.11, compared to the best results from the experiments on dense networks. The results provided some interesting insights. Most notably, we observed that the full rank PIE in experiment 8.Q fails to approximate the deblurring operator, while the AE achieved relatively good results. However, we observed that the introduction of multiple features in experiment 8.R yielded much better results, indicating that a linear combination of orthogonal convolution operators can approximate the deblurring operator, which can possibly be explained by the width requirements for universal approximation in [Yar18]. We noted that the introduction of linear combinations resulted in a minor increase in CNA for the models with diagonal layers, indicating that the under- and overdetermination from channel expansion in combination with diagonal layers in PIE models can lead to instability. Generally, the PIE models without added diagonals were the most robust, which is as expected, but the effect of added diagonals on the robustness of the convolutional models was greater than we anticipated. Still, the PIE models were all more robust than the comparative AE models.

Notably, the PIE model with linear diagonal performed better than the AE model at approximating the inverse deblurring operator, while all models achieved a nearly perfect SSIM on the blurred image. We generally see an increase in performance compared to the seperable patch models in 8.L. Visual comparison of Figure 8.16 to Figure 8.8 corroborates the results, indicating that the models with convolutional layers perform better in deblurring tasks than the dense variants in experiment 8.L. This is as expected, as convolution

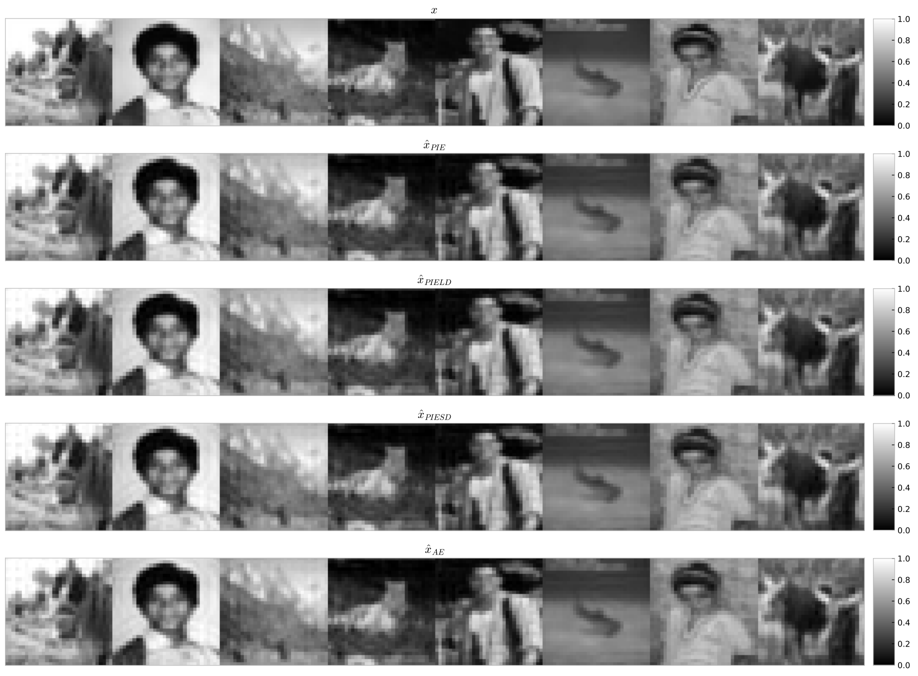

Figure 8.16: Reference image (above) and deblurred images (below) from experiment 8.R.

operators generally outperform dense layers on spatial data.

While the number of parameters of the PIE model is approximately doubled in the AE model, we note that the PIELD model features a linear diagonal layer which increases the number of parameters to 4026, compared to 1908 in the AE model. This discrepancy is lessened in deeper models, as the parameters of the convolution operators will increase the total number of parameters while the parameters of the diagonal layer will stay constant.

Image Upscaling with Convolutional PIEs

In experiment 8.O, we observed that PIE models with seperable patch layers performed relatively well when compared to similar AE models. In Table 8.6 we see that convolutional layers outperformed the seperable patch networks. Furthermore, the results from the PIE models were more or less comparable to the AE model, however we note that the AE models performed marginally better in terms of almost all our evaluated metrics except for CNA. Interestingly, we see an increase in performance in the AE model when a diagonal layer is added in 8.T. Generally speaking, the differences are minor enough that we can conclude that PIE models can effectively be applied in high-dimensional convolutional imaging tasks.

Exp.	**Model**	RE (10^{-2})	PSNR	SSIM($\hat{x}$)	SSIM($\hat{y}$)	CNA($\hat{x}$)	CNA($\hat{y}$)
8.O	**PIE**	7.527	33.811	0.676	0.998	**0.049**	0.810
	AE	9.097	28.432	0.666	0.983	0.049	0.810
8.S	**PIE**	7.084	34.519	0.701	0.999	0.053	**0.788**
	AE	6.822	37.519	0.704	0.999	0.053	0.801
8.T	**PIE**	6.949	36.098	0.701	0.999	0.053	0.794
	AE	**6.665**	**38.887**	**0.709**	**1.000**	0.053	0.813

Table 8.12: Results from upscaling experiments with convolutional networks (8.S) compared to dense models with seperable patch layers (8.O).

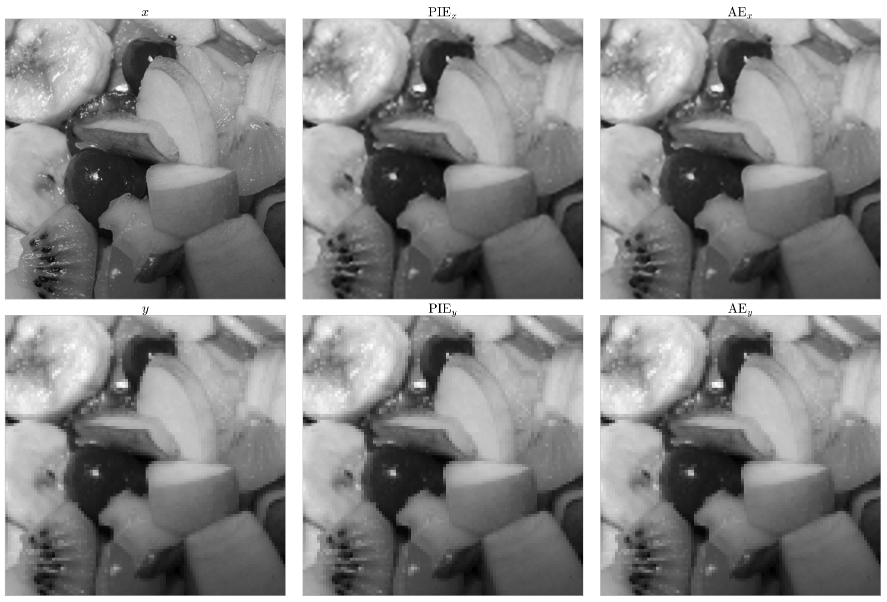

Figure 8.17: Image reconstructions from upscaling experiment 8.S.

Summary: Convolutional PIEs

The results from our experiments indicate that PIE models are compatible with convolutional layers with certain considerations. Firstly, the implicit constraints on the parameter space for full-rank single channel models could be too restrictive for PIE models. Secondly, any beneficial increases in stability from PIE architecture are slightly offset by the addition of diagonal layers. We stress that while the best results on deblurring were achieved by PIE models, more in-depth testing on convolutional pseudo-invertible networks in the context of different tasks is recommended before fully endorsing their application in a practical setting.

8.4 Conclusions: PIEs

At the start of this chapter, we outlined a set of enquiries regarding pseudo-invertible encoders we wanted to find answers to. We thus summarize our conclusions from the experiments we performed;

(i) **Does joint parametrization affect performance?** The joint parametrization in PIE models does not seem to affect their performance on imaging tasks in dense or separable-patch networks. However, our results in 8.L indicate that there are limitations to consider for convolutional full rank models.

(ii) **Does joint parametrization affect robustness?** Our results indicate that PIE models tend to exhibit better conditioning than AE models. Underdetermination does generally not seem to significantly affect the models adversely in terms of robustness.

(iii) **Do diagonal layers affect results?** The results indicate that adding diagonal layers to the PIE models generally improved their performance, at the cost of a decrease in robustness.

(iv) **Can PIEs reproduce principal components?** Linear PIE and AE models are able to reproduce the most significant principal vectors. Furthermore the ordering of the principal vectors can be encouraged given an appropriate training regime.

(v) **Do the weights have orthogonal properties?** The results in Table 8.3 and Figure 8.5 demonstrate that the estimated weights for trained PIE models – as well as the decoder in AE models – exhibit significant orthogonal properties compared to random initialization.

(vi) **Can PIEs models be applied with generative models?** PIE models can effectively be used in generative modelling, and they can even outperform AE models in certain tasks. In particular, we observed that they were more applicable to conditioning with additive coupling layers in our experiments.

(vii) **Can we construct convolutional PIEs?** In Section 8.3 we showed that convolutional layers can be applied with pseudo-invertible encoders with certain considerations, and the theoretical benefit of improved stability is largely confirmed in experiments with convolutional models.

In Observation 6.1.1 we outlined the motivation for invertible encoder networks, and our experiments on pseudo-invertible encoders show that they perform comparatively to autoencoders – even outperforming AE models on some occasions. Additionally, they display better robustness to adverserial attacks, which make them more suited for inverse problems. We conclude that the results of the experiments goes some way to support the theoretical benefits of invertible encoder networks.

CHAPTER 9

Experiments with Invertible Encoders

So far, we have presented results demonstrating how pseudo-invertible encoder networks can be applied to solve inverse problems by approximating bijective maps using implicit constraints given by the network architecture. We now shift our focus to experiments on models where these constraints are made explicit, and how this impacts learning and performance.

In Section 6.4 we discussed how orthogonality can be ensured via constraining optimization on the manifolds $\mathcal{SO}(n)$ and $\mathrm{St}(n, k)$, allowing us to apply Riemannian gradient descent to constrain the orthogonality properties of the weights. Furthermore in Propositions 6.4.15 and 6.4.16, we introduced methods for constructing invertible operators in the form of resolvent operators via Liouville-Neumann series. This method allows us to compute approximations of inverse convolution kernels for full rank operators. In this section, we investigate the applicability of these methods. We designed our experiments looking to answer the following questions.

(i) How does the explicit orthogonal constraints in IE models and resolvent operators in IRE models affect performance and robustness compared to PIE and AE models?

(ii) How do these constraints affect underdetermination in IE models?

(iii) How well does invertible encoders function for convolutional networks, and can a hybrid of pseudoinvertible and invertible encoders be applied for channel expansion?

The chapter thus consists of two sections. In the first section, we investigate dense and seperable patch invertible encoder networks via Riemannian gradient descent and invertible resolvent operators. In the second section, we look at convolutional invertible resolvent encoders and hybrid models for convolutional networks. Lastly, we summarize our findings by providing answers to the aforementioned enquires.

Exp.	In.Dim.	Hid.Dim.	Out.Dim.	Act.	Diag.	Patch Size	λ	Iter.	Loss	Data
9.A	32^2	32^2	32^2	BiCELU	(✓)	1×1			LHC	CIFAR
9.B	32^2	32^2	32^2	BiCELU	(✓)	1×1	2	16	LHC	CIFAR
9.C	32^2	32^2	32^2	BiCELU	(✓)	4×4			LHC	CIFAR
9.D	32^2	32^2	32^2	BiCELU	(✓)	4×4	2	16	LHC	CIFAR
9.E	32^2	$2 \cdot 32^2$	32^2	BiCELU	✓	1×1			LHC	CIFAR
9.F	32^2	128^2	32^2	BiCELU	✓	4×4			LHC	CIFAR

Table 9.1: Overview of experiments with dense IE and IRE models. Experiments including both models with diagonal layers and without are indicated with a parenthesis. Note that experiments with IRE models are denoted with the number of Liouville-Neumann iterations and λ.

9.1 Dense and Seperable IEs and IREs

For these experiments, we let all experimental parameters and model architectures be the same as in comparable experiments with pseudo-invertible models, except for the omission of models without diagonal layers in the overdetermined experiments 9.E and 9.F. Similar to an autoencoder, we postulated that the addition of diagonal layers would have little theoretical effect on the resolvent operators. We included models with diagonal layers to see how this related to practice. Note that overdetermined architectures do not feature experiments with invertible resolvent networks, as these models can only be applied as full rank operators.

It is worth mentioning that the IE models have an increased parameter space compared to the PIE models, however, only the skew symmetric operators have associated gradients and are used to directly update the base weight. The result is that while IE models have a lower number of parameters during estimation, however, the effective number of parameters can either be interpreted as being less than or greater than the parameter space of a comparable PIE model by including the base. No matter how it is interpreted, the parameter space will necessarily have lower dimensionality than a comparative AE model. Note that this does not apply to IRE models, which have the same number of parameters as a PIE model.

Deblurring with Dense IEs and IREs

We begin by looking at our deblurring experiment in the context of invertible encoder networks. One of the goals of the experiment was to compare the retraction mappings of the canonical matrix exponential for Lie groups (IEL) and the Cayley transform (IEC). In experiments 9.A through 9.D we performed experiments with and without added diagonal layers, and we denote models with added diagonal layers by IELD, IECD, and IRED. The training and model architecture is identical to experiments 8.I through 8.L, and the only difference is computation and update of the weights. An overview of the experiments is provided in Table 9.1.

The results in Table 9.2 shows that the full rank invertible encoder models (9.A, 9.B) display a slight drop in the performance of the blurring task (SSIM($\hat{y}$)) in the full rank dense models compared to the PIE models, with certain models achieving a minor increase in deblurring (SSIM($\hat{x}$)). In the seperable patch networks (9.C), this relationship is reversed, with the IE models displaying a

Exp.	**Model**	RE (10^{-2})	PSNR	SSIM($\hat{x}$)	SSIM($\hat{y}$)	CNA($\hat{x}$)	CNA($\hat{y}$)
8.I	**PIE**	5.671	32.514	0.917	0.985	3.612	12.168
	PIED	5.615	32.682	0.919	0.986	3.673	12.808
	AE	5.235	33.630	0.924	0.993	4.000	16.159
9.A	**IEL**	6.670	29.647	0.923	0.947	3.511	6.920
	IEC	7.172	29.047	0.908	0.939	3.511	6.918
	IELD	6.222	30.418	0.923	0.958	3.511	15.505
	IECD	6.569	29.936	0.911	0.954	3.511	19.277
9.B	**IRE**	6.684	29.645	0.935	0.956	3.820	34.913
	IRED	6.551	30.023	0.938	0.960	3.503	35.097
8.J	**PIE**	9.490	27.872	0.922	0.954	4.568	12.521
	PIED	8.838	28.474	0.928	0.957	6.161	20.228
	AE	8.514	28.725	0.927	0.961	5.308	25.806
9.C	**IEL**	5.974	31.454	0.915	0.976	**3.492**	6.768
	IEC	5.825	31.605	0.920	0.976	3.611	6.802
	IELD	5.517	32.308	0.920	0.981	3.512	14.709
	IECD	5.233	32.635	0.928	0.982	3.672	18.540
9.D	**IRE**	9.468	27.739	0.926	0.923	9.033	122.439
	IRED	9.657	27.508	0.922	0.929	8.052	125.209
8.K	**PIE**	5.604	32.398	0.921	0.983	3.935	12.765
	PIED	6.775	30.441	0.922	0.984	3.896	13.983
	AE	**4.904**	34.054	0.930	0.995	3.743	16.665
9.E	**IEL**	5.974	31.145	0.915	0.976	3.492	6.768
	IEC	5.825	31.605	0.920	0.976	3.611	6.802
	IELD	5.517	32.308	0.920	0.981	3.512	14.709
	IECD	5.233	32.635	0.928	0.982	3.672	18.540
8.L	**PIE**	7.151	30.863	0.935	0.978	4.399	11.920
	PIED	5.905	34.400	0.944	0.990	4.354	20.298
	AE	5.127	**36.629**	0.943	**0.998**	4.537	23.811
9.F	**IEL**	7.966	29.981	0.921	0.974	3.618	**6.310**
	IEC	8.094	29.881	0.919	0.973	3.634	6.380
	IELD	5.702	34.152	0.945	0.991	5.155	32.771
	IECD	5.532	34.653	**0.947**	0.993	5.440	30.265

Table 9.2: Results of deblurring experiment with dense IE/IRE models compared to the respective experiments with PIE and AE models from Section 8.1.

general increase in SSIM($\hat{y}$), and a general decrease in SSIM($\hat{x}$). The experiment with the IRE model on dense networks performed very well on the deblurring task (9.B), however for seperable patch networks (9.D) it performed poorly in terms of both objectives. For the overdetermined experiments (9.E, 9.F), the IE models optimize the weights over the Stiefel manifold $\mathrm{St}(n, k)$ instead of $\mathcal{SO}(n)$. We noticed that these models were especially computationally demanding due to the QR factorization involved in each weight update, which is why we elected to only include models with diagonal layers in these experiments. Contrary to the full rank experiments, both dense- and seperable patch networks saw a general decrease in SSIM($\hat{y}$) and an increase in SSIM($\hat{x}$) compared to the PIE models. Overall, the best results on deblurring were obtained by the IRED model in the experiments with full rank dense networks, and the IECD model in the experiments with overdetermined seperable patch networks – the results of which can be seen in Figure 9.1.

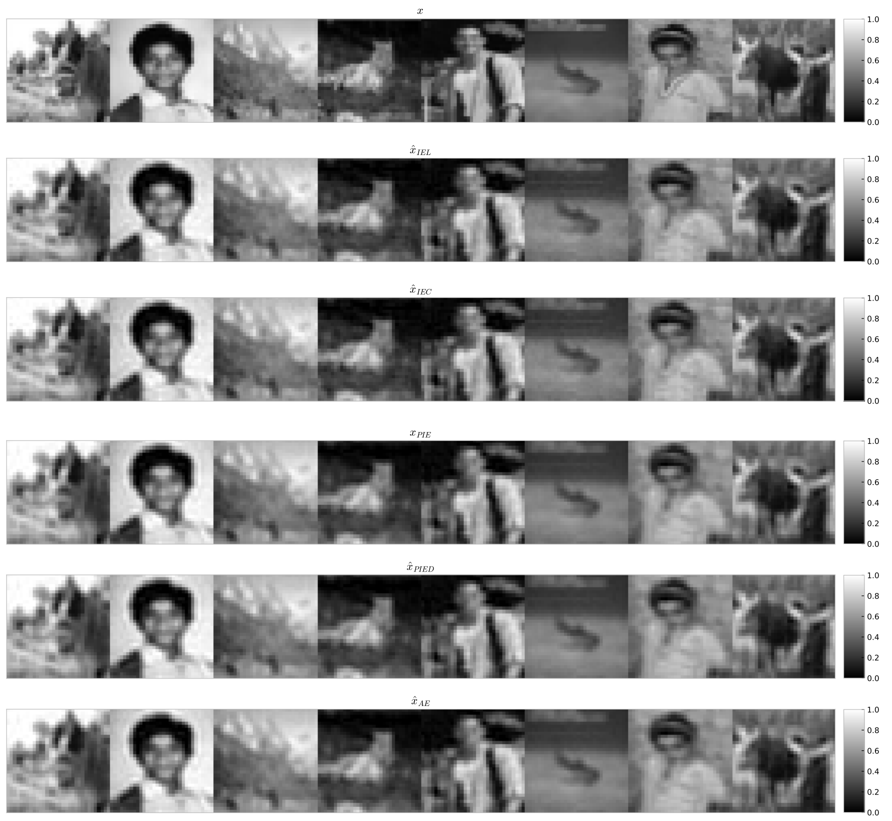

Figure 9.1: Comparison of image reconstructions from experiment 9.F with an invertible encoder (above) and 8.L with a pseudoinvertible encoder (below).

In terms of robustness, we note that the IE models with orthogonal constraints and no diagonal layers were the most robust, which is as expected as they are inherently Parseval optimal (Definition 4.5.1). We observed that the addition of diagonal layers to these models significantly decreased the overall robustness. Moreover, we see that the IRE models exhibit very significant ill-conditioning compared to the other models. This is somewhat expected, as we apply an approximation by Liouville-Neumann series, which is truncated at a set number of iterations. This could possibly be improved by a method which instead stopped iterations when the difference between successive approximations was lower than some set threshold. However, this does not necessarily directly address any ill-conditioning of the resolvent operators, and would increase the computational complexity. Overall, we were generally surprised to see the significant impact diagonal layers had on robustness in IE models given the results on PIE models. The poor robustness of the IRE models was somewhat disappointing, but not entirely unanticipated. Theoretically, the stability of the resolvent operators are highly dependent on the choice of λ, which in this experiment was set to a moderately high value to promote stability in training. It is possible that the stability could be improved by setting λ closer to 1.

Exp.	Model	RE (10^{-2})	PSNR	SSIM($\hat{x}$)	SSIM($\hat{y}$)	CNA($\hat{x}$)	CNA($\hat{y}$)
8.O	**PIE**	**7.527**	**33.811**	**0.676**	**0.998**	**0.051**	0.810
	AE	9.097	28.432	0.666	0.983	0.052	0.810
9.G	**IEL**	8.207	30.366	0.660	0.991	0.072	**0.793**
	IEC	8.611	29.367	0.650	0.986	0.080	0.803

Table 9.3: Results of superresolution experiment with dense IE models compared to the relevant results on PIE models from Section 8.1.

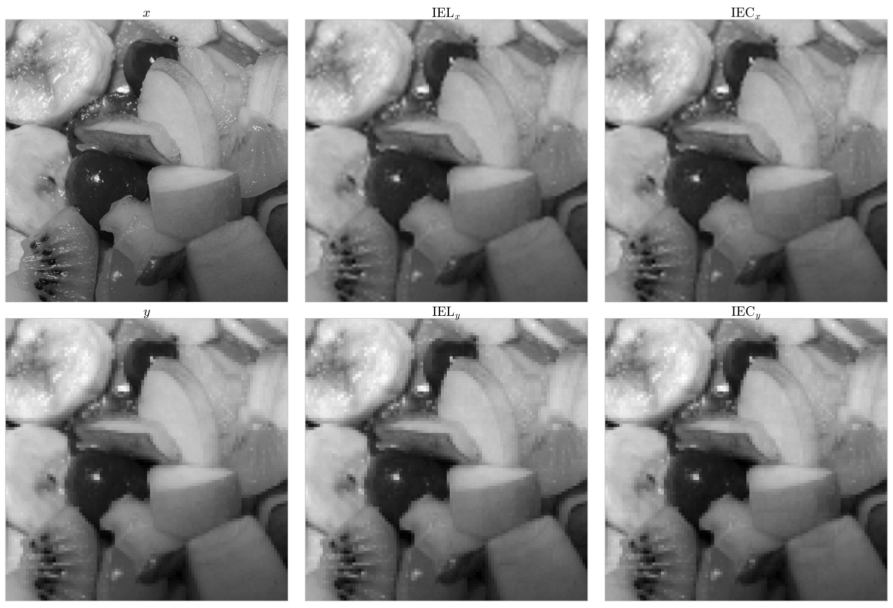

Figure 9.2: Reconstructions from IE superresolution experiment 9.F.

Image Upscaling with Dense IEs

Next, we investigated how the application of orthogonal constraints via Riemannian manifold learning affected the models in experiment 8.O. As in the previous experiment, all other experiment parameters are equal in these experiments. The seperable patch layers in this experiment are constructed with underdetermnination in the forward layers, thus the weights in the IE models are semiorthogonal, i.e., we optimize the operators over the Stiefel manifold $\text{St}(n, k)$ in each layer. For this reason, models using resolvent operators are not applicable as they require full rank operators. The results can be seen in Table 9.3 and Figure 9.2.

From the results, we noted that the IE models performed better in the downsampling task than the AE, however performed slightly worse than the PIE model overall. This runs counter to previous observations from deblurring, where we generally see a decrease in performance for approximating the forward operator with orthogonality constraints. We have previously postulated that this phenomenon relates to the sensitivity imbalance of the blurring and deblurring operators. However, the ill-posedness of the deblurring operator is related to stability (Property 2.1.1.iii), while underdetermination in the upsampling operator is related to uniqueness (Property 2.1.1.ii). These results indicate that Definition 4.4.3 does not necessarily imply that sensitivity imbalance have the same impact in underdetermined problems. Interestingly, the CNA results indicate that semiorthogonal constraints do not necessarily imply better adverserial robustness, as both the PIE and AE models achieved better results. Taken together, the results imply that IE models with semiorthogonal constraints are not always preferable to PIE models where orthogonality is implicitly encouraged by the network architecture.

Summary: Dense IEs

The experiments on IE models with orthogonal constraints from Riemannian manifold learning yielded some interesting points and observations which were quite surprising. In the experiments with dense networks, we see that invertible encoder networks generally provide a slight improvement in reconstruction quality for deblurring as well as better robustness. However, in the image upscaling task, we observed a decrease in the performance in upscaling at the same time as the IE models generally performed better than the AE model with downscaling. Interestingly, the PIE model from experiment 8.O provided the best performance and robustness in superresolution tasks with dense networks – except for the augmented generative latent model from experiment 8.P. While we would expect explicit constraints to improve robustness, this experiment shows that this might not always be the case for underdetermined problems, and implicit constraints can prove a more effective remedy for stability issues in underdetermined problems.

Lastly, it is important to emphasize that optimization on the Stiefel manifold is especially computationally expensive due to the QR-factorization steps in the Gram-Schmidt process required to complete the basis when updating weights. This will necessarily be an important consideration in the choice of model, and without stronger guarantees of improved stability, the computational cost might outweigh any potential benefit of semiorthogonality.

9.2 Convolutional IREs

Next, we look at our proposed method for constructing invertible convolutional networks using resolvent operators. A disadvantage of these models is that they require full rank operators, so any under- or overdetermination in terms of channel expansion between network layers can not be applied directly in these models. Channel expansion can however be introduced using pseudo-invertible convolution layers. The natural task for these experiments is the deblurring task, as it accommodates for full rank operators. We conducted one experiment on fully invertible convolutional networks (9.H) and one experiment on a hybrid model, where we apply a PIE model for channel expansion and contraction in the first and last layer, and apply invertible resolvent convolutional layers between the hidden layers (9.I). All models were constructed with the same methodology as in the previous experiment with convolution with pseudo-invertible networks (8.Q, 8.R), however, to balance the dimension of the parameter space of the hybrid model in experiment 9.I, we slightly modified the number of input

Exp.	Model	Features	Act.	Diag.	λ	Iter.	Loss	Data
9.H	**IRE**	$(1, 1, 1, 1, 1)$	BiCELU		4/3	16	LHC	CIFAR
	IRELD	$(1, 1, 1, 1, 1)$	BiCELU	Lin.	4/3	16	LHC	CIFAR
	IRESD	$(1, 1, 1, 1, 1)$	BiCELU	Sep.	4/3	16	LHC	CIFAR
9.I	**PIE-IRE**	$(1, 3, 3, 3, 1)$	BiCELU		20/19	24	LHC	CIFAR
	PIE-IRED	$(1, 3, 3, 3, 1)$	BiCELU		20/19	24	LHC	CIFAR

Table 9.4: Experiments with convolutional pseudo-invertable encoders.

Exp.	Model	RE (10^{-2})	PSNR	SSIM($\hat{x}$)	SSIM($\hat{y}$)	CNA($\hat{x}$)	CNA($\hat{y}$)
8.Q	**PIE**	8.684	28.579	0.883	0.979	**3.583**	**0.996**
	AE	3.964	37.952	0.954	0.999	3.627	24.585
9.H	**IRE**	4.163	35.542	0.959	0.996	18.028	31.311
	IRELD	4.248	35.082	0.958	0.995	9.791	28.250
	IRESD	4.295	35.241	0.957	0.995	14.507	43.935
8.R	**PIE**	3.387	39.716	0.965	0.999	6.481	15.545
	PIELD	3.334	39.421	0.969	0.999	13.928	31.597
	PIESD	3.339	39.981	0.965	0.998	14.241	36.827
	AE	3.334	39.143	0.966	**1.000**	5.280	37.334
9.I	**PIE-IRE**	3.051	41.057	0.968	0.999	8.150	43.996
	PIE-IRELD	**2.876**	**41.331**	**0.972**	0.999	10.110	30.024
	PIE-IRESD	3.308	38.340	0.970	0.999	7.738	46.228

Table 9.5: Results of experiments on deblurring with a convolutional IRE model compared to convolutional PIE models.

and output channels compared to experiment 8.Q, resulting in comparable dimensionality of the respective parameter space of the models. An overview of the experiments is provided in Table 9.4.

Deblurring with Convolutional IREs

In experiment (8.Q) from Section 8.3 we observed that the pseudo-invertible encoder network with full rank was not able to approximate the deblurring operator with the architecture in our experiment. We were interested to see if invertible resolvent networks would fare any better. In the last experiments, we noticed that the IRE models performed poorly in terms of robustness, and hypothesized that this could be related to our choice of λ. We therefore decreased λ for each operator in the experiment to see if this would have a positive impact on the robustness of the models. All experimental parameters were the same as in experiments 8.Q and 8.R, except for aformentioned differences in channel expansion in model 9.I, which can be seen in Table 9.4.

From the results in Table 9.5, we see that the networks with resolvent convolution operators performed very well, and were generally comparable to the AE models. In particular, the hybrid models performed markedly better than the AE model at approximating the deblurring operator, resulting in an improved SSIM($\hat{x}$) score. Overall, the hybrid convolutional PIE/IRE models performed the best of all our deblurring models, and the resulting deblurred images can be seen in Figure 9.3.

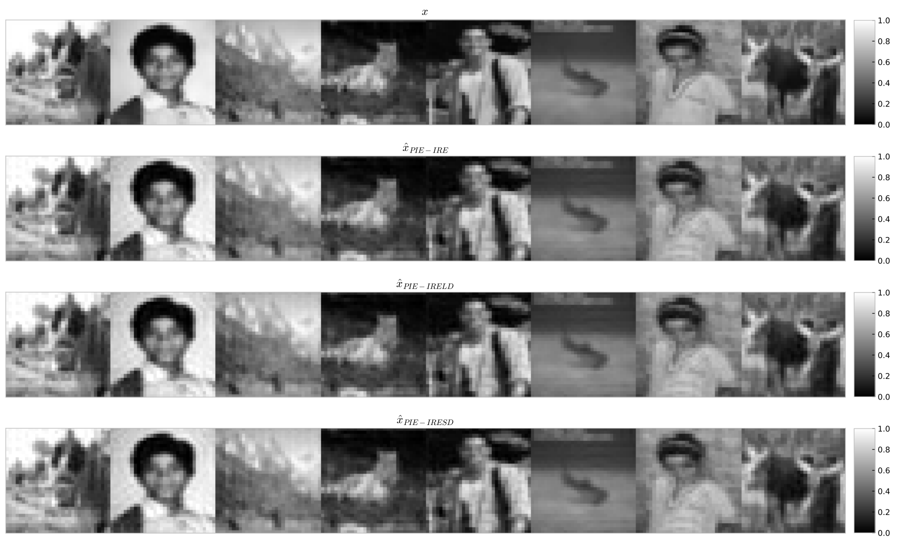

Figure 9.3: Reconstructions from hybrid PIE/IRE models from experiment 9.I.

In this experiment, we also observed an improvement in the IRE models with diagonal layers, which also seemed to improve the general robustness of the models in certain models. In general, the robustness of the convolutional IRE models were comparable to the PIE models with diagonals as well as the AE model. This indicates that the choice of λ could indeed be a factor in the stability of resolvent operators. We note that the IRE models with linear diagonals achieved slightly better CNA($\hat{y}$) scores than the AE models. In particular, the hybrid model with linear diagonal in experiment 9.I achieved comparatively decent stability results, second only to the PIE model. However, we want to stress that this result is slightly isolated from our other results, which indicate that IRE models are generally less robust than other models. Remedial methods for improved stability via regularization should therefore be prioritized in any subsequent work on these models.

Conclusion: Convolutional IREs

Our results indicate that invertible convolutional networks with resolvent operators perform comparatively well compared to supervised convolutional autoencoders and convolutional pseudo-invertible encoder networks. The addition of diagonal layers had a positive effect on both performance and stability. Furthermore, full rank invertible models without channel expansion can be constructed using resolvent convolutional layers, as opposed to what we observed with PIE networks in experiment 8.Q. We conclude that our method of constructing invertible convolutional networks with resolvent operators has the potential for being successfully applied in practical settings and is a completely new take on invertible neural networks. We emphasize that remedial methods for stability should be considered when deploying these models.

9.3 Conclusions: Invertible Encoders

The experiments in this chapter were designed to investigate the applicability of invertible encoder networks, and we wanted to determine the applicability of these models. From our experiments, we draw the following conclusions.

(i) **How do the constraints affect performance and robustness?** In terms of performance, we noted that invertible encoders performed similarly to the compared models, while in the full rank experiments, we noted an increase in performance of either the deblurring task (9.A) or the blurring task (9.C). In terms of robustness, we observed a general increase in robustness for the IE models with orthogonality constraints, which were by far the most robust models in our experiments. For the IRE models we noticed an improvement in performance, with a general decrease in adverserial robustness.

(ii) **How do the constraints affect underdetermination?** In the deblurring task, we observed only minor differences in the results of over- or underdetermined models. Nevertheless, we did note a general decrease in performance on the image upscaling task (9.G) compared to the PIE model, indicating that semiorthogonality is not always a useful property in over- or underdetermined problems.

(iii) **How applicable are resolvent operators for convolutional networks?** The convolutional IRE models provided good performance. Our overall best results for deblurring came from a hybrid PIE/IRE model with an added linear diagonal layer, indicating that these models can be applied in a practical setting. While we reported comparatively decent adverserial robustness, our testing was limited, and we can not yet conclude that these models provide improved guarantees in terms of robustness. We highlight this as an possible area of further work.

In conclusion, we have shown that invertible encoder models have practical applications in the modelling of inverse problems, particularly as an alternative to standard encoder-decoder models. The main benefits of IE models with orthogonal constraints are improved robustness to adverserial attacks, guarantees of bijectivity, and in some cases, improved performance in approximating the inverse mapping, as we saw in the deblurring tasks. For convolutional networks, IRE models and hybrid PIE/IRE models provide an interesting method of constructing invertible models. Lastly, we mention that invertible encoder networks can provide significant reduction of the overall dimensionality of the parameter space of a model, however, this comes at the cost of more computationally expensive operations during training.

CHAPTER 10

Summary and Further Work

In this chapter, we summarize our work in this thesis and show how the different parts achieve the outlined goals in the project description. Lastly, we discuss potential ideas and avenues for further work in terms of both theory and experimental methodology.

10.1 Summary and Project Goals

Recall from Section 1.2 that the project description of this master thesis outlined our four main goals, all of which have been achieved in our work. In Chapter 2, we outlined the fundamentals of the mathematical theory of inverse problems, which the first goal of our thesis. In Chapter 3 we presented the fundamental theory of statistical modelling and learning with a probabilistic framework and demonstrated how this is applied for modelling inverse problems, while in Chapter 4 we connected this to the theory of neural networks. This addressed the second goal given in the project description of our master thesis. In Chapter 5 we discussed the main sources of instability and uncertainty in neural networks, and introduced invertible neural networks as interesting models for modelling inverse problems, which – along with the general exposition of autencoder architectures from Section 4.4 – looked at some of the current methods for modelling inverse problems in neural networks, in line with the third goal in our project description. Lastly, in Chapter 6 we proposed to extend invertible neural networks to feed-forward models using invertible- and pseudoinvertible encoders, and in the subsequent chapters in part III we demonstrated that these models have certain clear benefits when compared to traditional autoencoder models. In particular, through our experiments we observed that our proposed models

- provided stronger guarantees of invertibility,
- generally yields increased robustness to adverserial attacks without explicit regularization terms,
- exhibit lower parametric dimensionality via joint parametrization of encoder and decoder networks,
- can be successfully applied in existing feedforward architectures, including convolutional models.

We have thus proposed a new model for invertible neural networks, and our experiments indicate that these models show promise for modelling inverse problems, achieving the fourth and last goal of our master thesis.

10.2 Further Work: Experimental Methodology

While we have endeavoured to cover all topics to the fullest extent in this thesis, there is still much to be said on the subject, and our work will inevitably leave some stones unturned. In this section, we address some of the ideas we had which we did not have time to investigate further, as well as any potential blindsides in our experimental methodology.

In terms of the number of experiments, we believe we have managed to outline the applicability of the models for inverse problems on a small scale, and demonstrated that many of the theorized properties of invertible encoder models seem to hold. Still, there is still much room for improvement, and we therefore outline some potential experiments and methodologies which are conspicuously absent from our work.

Variety

We would have liked to extend our results to more interesting datasets and model scaling. A lot of time was spent on baseline modelling with very similar experiments – as is expected with proof-of-concept work. To fully test our proposed models, however, this is a required and necessary step in any subsequent work on these models.

Comparison with Other Architectures

While we discussed normalizing flows to some extent, we did not perform any tests to compare the two approaches. We also omitted any generative modelling based on generative adverserial architectures, which could have been very interesting, particularly in light of the results in Section 9.1.

Hyperparameter Search

We did little in terms of hyperparameter estimation; both in the context of bi-Lipschitzian activation and λ for resolvent layers. This is likely to be a tedious, but necessary step for the further application of invertible encoder models. Likewise, learnable parameters should be experimented with – particularly for BICELU and the Dirichlet Softmax.

Regularization

The effect of added regularization should clearly be pursued in subsequent work on these models. We have intentionally avoided regularization to provide baselines for our models without the interference of remedial methods for stability, precisely to evaluate the inherent robustness of our proposed models. However, one could easily argue that regularization would have a positive effect, particularly in the case of PIE models.

Estimating Lipschitz Constant

The proposed method for estimating Lipschitz constants in [Faz+19] would have proved a valuable metric in our experiments. Unfortunately, the authors implementation is restricted to MATLAB as well as proprietary DSP solvers. This should be rewritten in open source, especially considering the almost exclusive use of Python in the research community.

Bootstrap Estimates

Due to the sheer number of small-scale experiments, we did not prioritize the computation of bootstrap estimates for our models, as this would have been computationally costly. Limiting the number of models and providing bootstrap estimates should be prioritized in further work on these models.

10.3 Further Work: Theoretical Ideas

Lastly, we discuss some theoretical avenues which could be potentially interesting to explore. While we believe we managed to present our most important ideas on the topic of invertible neural networks and inverse problems, we still had some avenues we did not have the time to pursue further.

Orthogonal Convolutions

The application of the methods for constructing invertible convolution operators using Liouville-Neumann series from Proposition 6.4.16 to compute inverse kernels can likely be applied to construct retraction maps for Riemannian manifold learning with gradient descent. As previously noted, this could have applications outside of neural networks and is a method we would like to see come to practical fruition in the future.

Frame Theory

Our work could very likely be better contextualized in the theory of frames [CKP13], which provides a generalization of the notion of orthogonal bases and explicit ties to series expansion methods. In particular, the construction of equiangular frames can possibly be extended to manifold learning on the Grassmann manifold via Riemannian gradient descent. Such an approach could be used to extend the IE and PIE architectures to more robust frameworks for dealing with inverse problems.

Extending Resolvent Operators to Residual Blocks

Residual blocks have been successfully applied to invertible networks, and the resolvent formalism can allow for residual blocks to be constructed with iterative methods, extending our proposed resolvent layer. In particular, much work has been done on iterative methods for nonlinear integral equations, which can possibly be applied to construct nonlinear residual blocks. More work on this connection could be a potentially powerful tool for constructing invertible neural networks.

Learning Permutations

One interesting application of unitary matrices over the complex numbers is the link to doubly stochastic matrices, which could be used to construct learnable permutation matrices in neural networks. This could allow for more direct SVD layers with explicit constraints on diagonal layers – or singular values – for better control of ill-conditioning of individual linear operators in a network. Furthermore, it has potential applications to graph neural networks and pruning, which are both highly interesting use cases.

Other Practical Ideas

We also had several ideas relating to the practical implementations of our models.

- We did not have time to give any attention to the application of sequence modelling using invertible encoders. This is a promising avenue for further research on these models. In particular, we consider attention mechanisms as an interesting application of invertible encoders.
- General representation learning tasks are interesting applications for invertible encoders. The idea of contrastive learning – especially with momentum [He+19] – is of interest, and we would have liked to see invertible encoder architectures compared to traditional autoencoder structures in this setting.
- While we achieved good results in reconstructing principal vectors in pseudo-invertible encoders, the training regime is tedious. A better approach would be to construct a specific dropout regime which prioritizes neurons sequentially to encourage the importance of output weights more naturally.

Appendices

APPENDIX A

Fundamental Theory

This section provides a set of fundamental definitions and results required for the thesis. As such, most of this section is likely to be known to the reader, but are included for reference. Note however, that this is by no means an exhaustive exposition, and we leave the details to the more rigorous literature. We especially recommend the source material for this section, which is based on [Bil95; Hal15; Lin17; MW99; Rud87; RY08].

A.1 Spaces and Analysis

Definition A.1.1 (Index set)**.** Let $\mathcal{X}$ be a set, and let $\mathcal{I} \subseteq \mathbb{Z}$. Let $x : \mathcal{I} \to \mathcal{X}$ be a surjective mapping $x(i) = x_i \in \mathcal{X}$ for all $i \in \mathcal{I}$. Then $\mathcal{I}$ is an *index set*, and we write $\mathcal{X} = (x_i)_{i \in \mathcal{I}}$.

Definition A.1.2 (Linear Space)**.** Let $\mathcal{X}$ be a set, and let $\mathbb{K}$ be a field, either $\mathbb{R}$ or $\mathbb{C}$. Let $\bullet + \bullet : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$ and $\bullet \cdot \bullet : \mathbb{K} \times \mathcal{X} \to \mathcal{X}$ be maps following standard conventions and notation of addition and multiplication such that for all $a, b \in \mathbb{K}$ and $x, y, z \in \mathcal{X}$ we have

(i) $x + y = y + x$,

(ii) $(x + y) + z = x + (y + z)$,

(iii) there exists $0 \in \mathcal{X}$ such that $x + 0 = x$,

(iv) there exists $-x \in \mathcal{X}$ such that $x + (-x) = 0$,

(v) $a(x + y) = ax + ay$,

(vi) $(a + b)x = ax + bx$,

(vii) $a(bx) = (ab)x$,

(viii) there exists $1 \in \mathbb{K}$ such that $1x = x$.

Then $\mathcal{X}$ is a *linear space*, also called a *vector space*.

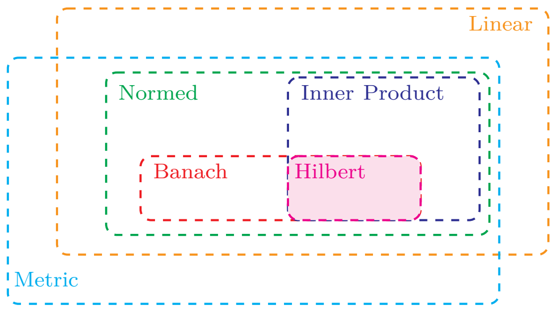

Figure A.1: A diagram of the hierarchical relations between spaces.

Definition A.1.3 (Image of a Mapping)**.** Let $f : \mathcal{X} \to \mathcal{Y}$. Let $f(\mathcal{X}) \subset \mathcal{Y}$ and $f^{-1}(\mathcal{Y}) \subset \mathcal{X}$ be sets such that

$$f(\mathcal{X}) = \{y \in \mathcal{Y} : \text{there exists } x \in \mathcal{X} \text{ such that } f(x) = y\} \tag{A.1}$$
$$f^{-1}(\mathcal{Y}) = \{x \in \mathcal{X} : \text{there exists } y \in \mathcal{Y} \text{ such that } f(x) = y\}. \tag{A.2}$$

Then $f(\mathcal{X})$ is the *image of* f over $\mathcal{X}$, and $f^{-1}(\mathcal{Y})$ is the *pre-image of* f over $\mathcal{Y}$.

Definition A.1.4 (Indicator Function)**.** Let $\mathcal{X}$ be a linear space, let $A \subset \mathcal{X}$ and let $\mathbb{I}_A : \mathcal{X} \to \{0, 1\}$ be a map given by

$$\mathbb{I}_A(x) = \begin{cases} 1, & \text{if } x \in A; \\ 0, & \text{otherwise.} \end{cases} \tag{A.3}$$

Then $\mathbb{I}_A$ is called the *indicator function* of A.

Definition A.1.5 (Simple Function)**.** Let $\mathcal{X}$ be a linear space and let $(A_j)_{j \in \mathcal{I}}$ be a sequence of sets with $A_j \subset \mathcal{X}$. Let f be a function with a finite range $\{a_1, a_2, \ldots, a_k\}$ such that $f^{-1}(a_j) = A_j$. Then f can be expressed as a linear combination of indicator functions by

$$f = \sum_{i=1}^{k} a_j \mathbb{I}_{A_j}, \tag{A.4}$$

and we call f a *simple function*.

Definition A.1.6 (Metric Space)**.** Let $\mathcal{X}$ be a set, and let $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ be a map defined on $\mathcal{X}$ such that for all $x, y, z \in \mathcal{X}$ we have

(i) $d(x, y) = 0$ iff. $x = y$,

(ii) $d(x, y) = d(y, x)$,

(iii) $d(x, y) \leq d(x, z) + d(y, z)$.

Then d is a *metric* on $\mathcal{X}$, and $\mathcal{X}$ is called a *metric space*.

Definition A.1.7 (Cauchy Sequence)**.** Let $\mathcal{X}$ be a metric space and let $(x_i)_{i\in\mathbb{N}}$ be a sequence where $x_i \in \mathcal{X}$ for all $i \in \mathbb{N}$. If for every $\epsilon \in \mathbb{R}_{\geq 0}$ there exists $N \in \mathbb{N}$ such that for all $m, n > N$ we have

$$d(x_m, x_n) < \epsilon, \tag{A.5}$$

then $(x_i)_{i\in\mathbb{N}}$ is a *Cauchy sequence.*

Definition A.1.8 (Normed Space)**.** Let $\mathcal{X}$ be a linear space. Let $\|\bullet\| : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be a map such that for all $a \in \mathbb{K}$ and $x, y \in \mathcal{X}$ we have

(i) $\|x\| = 0$ iff. $x = 0$,

(ii) $\|ax\| = |a|\|x\|$,

(iii) $\|x + y\| \leq \|x\| + \|y\|$.

then $\|\bullet\|$ is a *norm* on $\mathcal{X}$ and we call $\mathcal{X}$ a normed space.

Lemma A.1.9 (Metric Induced by the Norm)**.** *Let $\mathcal{X}$ be a normed space. Let $d(x, y) = \|x - y\|$. Then d is a metric, and we say d is the* metric induced by the norm *of $\mathcal{X}$.*

Proof. Property A.1.6.i is satisfied by Property A.1.8.i. Property A.1.6.ii is satisfied by

$$\begin{aligned} d(x, y) &= \|x - y\| & \text{(A.6)}\\ &= |-1|\|x - y\| & \text{(A.7)}\\ &= \|y - x\| & \text{(A.8)}\\ &= d(y, x), & \text{(A.9)} \end{aligned}$$

showing symmetry. Then the triangle inequality of Property A.1.6.iii is clearly satisfied, as we have

$$\begin{aligned} d(x, y) &= \|x - y\| & \text{(A.10)}\\ &= \|(x - z) + (z - y)\| & \text{(A.11)}\\ &\leq \|x - z\| + \|z - y\| & \text{(A.12)}\\ &= d(x, z) + d(z, y) & \text{(A.13)}\\ &= d(x, z) + d(y, z). & \text{(A.14)} \end{aligned}$$

■

Definition A.1.10 (Complete Space)**.** Let $\mathcal{X}$ be a metric space. If every Cauchy sequence converges to an element in $\mathcal{X}$, then $\mathcal{X}$ is a *complete space.*

Definition A.1.11 (Banach Space)**.** Let $\mathcal{X}$ be a normed space. If $\mathcal{X}$ is complete w.r.t. the metric induced by the norm, then $\mathcal{X}$ is a *Banach space.*

Definition A.1.12 (Compact Set)**.** Let $\mathcal{X}$ be a metric space, and let $A \subset \mathcal{X}$ be a subset. If every sequence in A has a convergent subsequence with a limit point in A, then A is a *compact set.*

Definition A.1.13 (Inner Product Space)**.** Let $\mathcal{X}$ be a linear space, and let $\langle \bullet, \bullet \rangle : \mathcal{X} \times \mathcal{X} \to \mathbb{K}$ be a map such that for all $a, b \in \mathbb{K}$ and $x, y, z \in \mathcal{X}$ we have

(i) $\langle ax + by, z \rangle = a\langle x, z \rangle + b\langle y, z \rangle$,

(ii) $\langle x, y \rangle = \overline{\langle y, x \rangle}$,

(iii) $\langle x, x \rangle \geq 0$,

(iv) $\langle x, x \rangle = 0$ iff. $x = 0$.

Then $\langle \bullet, \bullet \rangle$ is an inner product on $\mathcal{X}$ and we call $\mathcal{X}$ an *inner product space.*

Lemma A.1.14 (Norm Induced by Inner Product)**.** *Let $\mathcal{X}$ be an inner product space, and let $\|x\| = \sqrt{\langle x, x \rangle}$. Then $\|\bullet\|$ is a norm induced by the inner product of $\mathcal{X}$.*

A proof for Lemma A.1.14 is given in [MW99, pp.460–461].

Definition A.1.15 (Hilbert Space)**.** Let $\mathcal{X}$ be an inner product space. If $\mathcal{X}$ is complete w.r.t. the metric induced by the norm, then $\mathcal{X}$ is a *Hilbert space.*

Definition A.1.16 (Orthogonality)**.** Let $\mathcal{X}$ be an inner product space. If for any two elements $x, y \in \mathcal{X}$ we have $\langle x, y \rangle = 0$, we say that x, y are *orthogonal*, and we write $x \perp y$.

Definition A.1.17 (Orthogonal Set)**.** Let $\mathcal{X}$ be an inner product space and let $A \subset \mathcal{X}$. If for all $a, b \in A$ where $a \neq b$ we have $a \perp b$ then A is an *orthogonal set.*

Definition A.1.18 (Orthogonal Compliment)**.** Let $\mathcal{X}$ be an inner product space and let $A \subset \mathcal{X}$ be a proper subspace. Let $B = A^c$. If for all $a \in A, b \in B$ we have $a \perp b$ then B is the *orthogonal complement of* A, and we write $B = A^{\perp}$.

A.2 Operators

Definition A.2.1 (Linear Operator)**.** Let $\mathcal{X}, \mathcal{Y}$ be linear spaces over the same field $\mathbb{K}$. Let $W : \mathcal{X} \to \mathcal{Y}$ be a map such that for all $x \in \mathcal{X}, a \in \mathbb{K}$ we have

(i) $W(x + y) = W(x) + W(y)$,

(ii) $W(ax) = aW(x)$.

Then W is a *linear operator*, and we write $W(x) = Wx$.

Definition A.2.2 (Bounded Operator)**.** Let $\mathcal{X}, \mathcal{Y}$ be normed spaces over the same field $\mathbb{K}$, and let $W : \mathcal{X} \to \mathcal{Y}$ be a linear operator such that for all $x \in \mathcal{X}$ there exists some $k \in \mathbb{K}$ with $k > 0$ such that

$$\|Wx\| \leq k\|x\|. \tag{A.15}$$

Then W is a *bounded operator*, and we write $W \in B(\mathcal{X}, \mathcal{Y})$

Definition A.2.3 (Operator Norm)**.** Let $\mathcal{X}, \mathcal{Y}$ be normed spaces, and let $W : \mathcal{X} \to \mathcal{Y}$ be a bounded operator. Let $\|\bullet\|_{\text{op}} : B(\mathcal{X}, \mathcal{Y}) \to \mathbb{R}_{\geq 0}$ such that

$$\|W\|_{\text{op}} = \inf\{k \geq 0 : \|Wx\| \leq k\|x\| \ \forall \ x \in \mathcal{X}\}. \tag{A.16}$$

Then $\|\bullet\|_{\text{op}}$ is called the *operator norm* on $B(\mathcal{X}, \mathcal{Y})$.

Lemma A.2.4 (Continuous Linear Operators)**.** *Let $\mathcal{X}, \mathcal{Y}$ be normed spaces over the same field $\mathbb{K}$, and let $W : \mathcal{X} \to \mathcal{Y}$ be a linear operator. Then the following statements are equivalent.*

(i) *W is continuous,*

(ii) *W is continuous at 0,*

(iii) *W is uniformly continuous,*

(iv) *W is bounded.*

A proof for Lemma A.2.4 is provided in [RY08, pp.88–89]

Theorem A.2.5 (Adjoint Operator)**.** *Let $\mathcal{X}, \mathcal{Y}$ be Hilbert spaces over $\mathbb{C}$ and let $W : \mathcal{X} \to \mathcal{Y}$ be a bounded operator. Then for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ there exists a unique operator $W^* : \mathcal{Y} \to \mathcal{X}$ such that*

$$\langle Wx, y \rangle = \langle x, W^* y \rangle, \tag{A.17}$$

and we call W^* the adjoint operator *of W.*

[RY08, pp.168–169] provides a concise proof for Theorem A.2.5, and shows the properties for adjoint operators found in Lemma A.2.6.

Lemma A.2.6 (Properties of adjoint operators)**.** *Let $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ be Hilbert spaces over $\mathbb{C}$ and let $V, W \in B(\mathcal{X}, \mathcal{Y})$ and $U \in B(\mathcal{Y}, \mathcal{Z})$. Let $a, b \in \mathbb{C}$. Then*

(i) $(W^*)^* = W$,

(ii) $(UV)^* = V^*U^*$,

(iii) $(aW + bV)^* = \overline{a}W^* + \overline{b}V^*$,

(iv) $\|W^*\|_{\text{op}} = \|W\|_{\text{op}}$,

(v) $\|W^*W\|_{\text{op}} = \|W\|_{\text{op}}^2$.

Definition A.2.7 (Self-adjoint Operators)**.** Let $\mathcal{X}$ be a Hilbert space over $\mathbb{C}$ and let $W : \mathcal{X} \to \mathcal{X}$ be a bounded operator such that

$$W = W^*. \tag{A.18}$$

Then W is a *self-adjoint operator.*

Definition A.2.8 (Positive Operator)**.** Let $\mathcal{X}$ be a Hilbert space, and let $W : \mathcal{X} \to \mathcal{X}$ be an operator such that for all $x \in \mathcal{X}$ we have

$$\langle x, Wx \rangle \geq 0. \tag{A.19}$$

Then W is a *positive operator.*

Definition A.2.9 (Compact Linear Operator)**.** Let $\mathcal{X}, \mathcal{Y}$ be Hilbert spaces, and let $W \colon \mathcal{X} \to \mathcal{Y}$. If for any bounded sequence $(x_i)_{i\in\mathbb{N}}$ in $\mathcal{X}$ we have that $(Wx_i)_{i\in\mathbb{N}}$ contains a converging subsequence, we say that W is a *compact linear operator.*

Definition A.2.10 (Orthogonal and Orthonormal Sequences)**.** Let $\mathcal{X}$ be a inner product space and let $V = (v_n \in \mathcal{X})_{n\in\mathcal{I}}$ be a sequence such that v_m, v_n are orthogonal for all $m, n \in \mathcal{I}$. Then V is an *orthogonal sequence* in $\mathcal{X}$. Additionally, if $\|v_n\| = 1$ for all $n \in \mathcal{I}$, then V is an *orthonormal sequence.*

Definition A.2.11 (Orthogonal and Orthonormal Bases)**.** Let $\mathcal{X}$ be a Hilbert space and let $\mathscr{B}$ be an orthogonal sequence. If for $v_n \in \mathscr{B}$, $x = \sum_{n\in\mathcal{I}} \langle x, v_n \rangle v_n$ for all $x \in \mathcal{X}$ we call $\mathscr{B}$ is an *orthogonal basis* for $\mathcal{X}$. Likewise, if $\mathscr{B}$ is an orthonormal sequence, then $\mathscr{B}$ is an *orthonormal basis* for $\mathcal{X}$.

Definition A.2.12 (Unitary Operator)**.** Let $\mathcal{X}$ be a Hilbert space over $\mathbb{C}$ and let $W : \mathcal{X} \to \mathcal{X}$ be a bounded operator such that

$$W^*W = WW^* = I. \tag{A.20}$$

Then W is a *unitary operator.*

A.3 Measure Theory

Definition A.3.1 (Power Set)**.** Let $\mathcal{X}$ be a set, and let $\mathscr{P}(\mathcal{X}) = \{A : A \subseteq \mathcal{X}\}$. Then $\mathscr{P}(\mathcal{X})$ is the *power set of* $\mathcal{X}$.

Definition A.3.2 (σ-algebra)**.** Let $\mathcal{X}$ be a set. Let $\mathcal{A} \subseteq \mathscr{P}(\mathcal{X})$ such that

(i) $A \in \mathcal{A}$ iff. $A^c \in \mathcal{A}$,

(ii) $(A_i)_{i\in\mathcal{I}} \subset \mathcal{A}$ if and only if $\bigcup_{i\in\mathcal{I}} A_i \subset \mathcal{A}$.

Then $\mathcal{A}$ is a *σ-algebra.*

Definition A.3.3 (Measurable Space)**.** Let $\mathcal{X}$ be a set, and let $\mathcal{A}$ be a σ-algebra on $\mathcal{X}$. Then the pair $(\mathcal{X}, \mathcal{A})$ is called a *measurable space.*

Definition A.3.4 (Pairwise Disjoint Sequence)**.** Let $\mathcal{A}$ be a σ-algebra, and let $(A_i)_{i\in\mathcal{I}} \subset \mathcal{A}$ be a sequence such that $A_i \cap A_j = \emptyset$ for all $i \neq j \in \mathcal{I}$. Then the sequence $(A_i)_{i\in\mathcal{I}}$ is *pairwise disjoint.*

Definition A.3.5 (Measure)**.** Let $(\mathcal{X}, \mathcal{A})$ be a measurable space. Consider $\overline{\mathbb{R}}_{\geq 0} = \{r \in \mathbb{R} : r \geq 0\} \cup \{\infty\}$ and let $\nu : \mathcal{A} \to \overline{\mathbb{R}}_{\geq 0}$ be a map such that

(i) $\nu(\emptyset) = 0$,

(ii) $\nu\left(\bigcup_{i\in\mathcal{I}} A_i\right) = \sum_{i\in\mathcal{I}} \nu(A_i)$ for pairwise disjoint $\{A_i\}_{i\in\mathcal{I}}$.

Then ν is a *measure on* $\mathcal{A}$.

Definition A.3.6 (Measure Space)**.** Let $(\mathcal{X}, \mathcal{A})$ be a measurable space and let ν be a measure. Then the triple $(\mathcal{X}, \mathcal{A}, \nu)$ is a *measure space.*

Definition A.3.7 (Almost Everywhere)**.** Let $(\mathcal{X}, \mathcal{A}, \nu)$ be a measure space and let $A \subset \mathcal{A}$ be a subset for which some property holds. If $\nu\big(A^c\big) = 0$ then we say that property holds *almost everywhere* with respect to the measure ν, and we abbreviate this by a.e..

Definition A.3.8 ($\mathcal{A}$-measurable Functions)**.** Let $(\mathcal{X}, \mathcal{A})$ be a measurable space. Let $f : \mathcal{X} \to \mathbb{K}$ be a map such that for all open subsets $O \subset \mathbb{K}$ we have

$$f^{-1}(O) = \big\{x \in \mathcal{X} : f(x) \in O\big\} \in \mathcal{A}. \tag{A.21}$$

Then f is an *$\mathcal{A}$-measurable function.*

Definition A.3.9 (Space of $\mathcal{A}$-measurable Functions)**.** Let $(\mathcal{X}, \mathcal{A})$ be a measurable space, and let

$$\mathcal{M}(\mathcal{X}) = \{f \colon \mathcal{X} \to \mathbb{K} : f \text{ is a measurable function}\}. \tag{A.22}$$

Then we call $\mathcal{M}(\mathcal{X})$ *the space of $\mathcal{A}$-measurable functions* on $\mathcal{X}$.

Definition A.3.10 (Borel σ-algebra and Sets)**.** Let $\mathcal{B}$ denote the smallest σ-algebra containing the open sets of some space $\mathcal{X}$. Then $\mathcal{B}$ is called a *Borel σ-algebra*, and $B \in \mathcal{B}$ is called a *Borel set.*

Definition A.3.11 (Positive-Valued Lebesgue Integral)**.** Let $(\mathcal{X}, \mathcal{A}, \nu)$ be a measure space, and let $f \in \mathcal{M}(\mathcal{X})$. Let $f'(t) = \nu(\{x \in \mathcal{X} : f(x) > t\})$. Then the integral

$$\int_+ f \; d\nu = \int_0^\infty f'(t) \; dt. \tag{A.23}$$

is called the *positive valued Lebesgue integral* of f w.r.t. the measure ν.

Definition A.3.12 (Lebesgue Integral)**.** Let $(\mathcal{X}, \mathcal{A}, \nu)$ be a measure space, and let $f \in \mathcal{M}(\mathcal{X})$. Let $f_{>0}(x) = \max\{f(x), 0\}$ and let $f_{<0}(x) = \max\{-f(x), 0\}$. Then the integral

$$\int f \; d\nu = \int_+ f_{>0} \; d\nu - \int_+ f_{<0} \; d\nu \tag{A.24}$$

is called the *Lebesgue integral* of f w.r.t. the measure ν.

Remark A.3.13*.* The *Lebesgue measure* is central in measure theory, and its exact definition is rather involved. For a more detailed discussion on its construction, the relation to Lebesgue integration, and more details of the properties of both the Lebesgue integral and Lebesgue measure, see [Lin17, Chapter 7, 8] and [MW99, Chapters 3, 4, 5].

Definition A.3.14 (L^p-norm)**.** Let $(\mathcal{X}, \mathcal{A}, \nu)$ be a measure space. Let $p \geq 1$, and let $\|\bullet\|_p : \mathcal{M}(\mathcal{X}) \to \overline{\mathbb{R}}_{\geq 0}$ be a mapping such that for all $f \in \mathcal{M}(\mathcal{X})$ we have

$$\|f\|_p = \begin{cases} \left(\displaystyle\int |f|^p \; d\nu \right)^{1/p}, & \text{if } p < \infty; \\ \inf \left\{ k \in \mathbb{K} : f(x) \leq k \;\; \text{a.e.} \right\}, & \text{otherwise.} \end{cases} \tag{A.25}$$

Then $\|\bullet\|_p$ is called the *L^p-norm.*

Definition A.3.15 (L^p-spaces)**.** Let $(\mathcal{X}, \mathcal{A}, \nu)$ be a measure space, and let $p \geq 1$. Let $\mathcal{L}^p(\mathcal{X})$ be given by

$$\mathcal{L}^p(\mathcal{X}) = \Big\{ f \colon \mathcal{X} \to \mathbb{K} \mid f \in \mathcal{M}(\mathcal{X}), \|f\|_p < \infty \Big\}. \tag{A.26}$$

Now let

$$f^* = \{g \in \mathcal{L}^p(\mathcal{X}) : \|f - g\|_p = 0\}, \tag{A.27}$$

such that

$$L^p(\mathcal{X}) = \{f^* : f \in \mathcal{L}^p(\mathcal{X})\}. \tag{A.28}$$

Then $L^p(\mathcal{X})$ is called the *L^p-space* on $\mathcal{X}$.

Lemma A.3.16 (L^p-spaces are Banach)**.** *Let $(\mathcal{X}, \mathcal{A}, \nu)$ be a measure space. Then $L^p(\mathcal{X})$ is a* Banach space*.*

[MW99, pp.477–479] shows that L^p-spaces are linear and complete.

Definition A.3.17 (Dirac Measure)**.** Let $(\mathcal{X}, \mathcal{A})$ be a measurable space. For a fixed $x \in \mathcal{X}$, let $\delta_x : \mathcal{X} \to \overline{\mathbb{R}}_{\geq 0}$ be a measure such that

$$\delta_x(A) = \begin{cases} 1, & \text{if } x \in A; \\ 0, & \text{otherwise.} \end{cases} \tag{A.29}$$

Then δ_x is called the *Dirac measure* on $(\mathcal{X}, \mathcal{A})$.

Remark A.3.18. The Dirac measure is similar in form to the indicator function $\mathbb{I}_A$ from Definition A.1.4. The only difference is in how we define the domain. Furthermore, it is worth noting that the Dirac measure is related to the *Dirac-δ function*,

$$\delta(x) = \begin{cases} \infty, & \text{if } x = 0; \\ 0, & \text{otherwise,} \end{cases} \tag{A.30}$$

which in turn is related to the *Kronecker-δ* defined over integers $i, j \in \mathbb{Z}$, given by

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j; \\ 0, & \text{otherwise.} \end{cases} \tag{A.31}$$

Lastly, we note that the Dirac measure is a probability measure (see Definition A.4.2) for any fixed x, as $\delta_x(\mathcal{X}) = 1$.

Definition A.3.19 (Counting Measure)**.** Let $\big(\mathcal{X}, \mathcal{A}\big)$ be a measurable space. Let $\# : \mathcal{A} \to \overline{\mathbb{R}}_{\geq 0}$ be a measure such that for any $A \in \mathcal{A}$ we have

$$\#(A) = \sum_{x \in \mathcal{X}} \delta_x(A). \tag{A.32}$$

Then # is called the *counting measure*.

Remark A.3.20. The Dirac measure and the counting measure can be applied to connect Lebesgue integration with summation. To see how, let $\mathcal{X}$ be a discrete sequence $(x_i)_{i \in \mathcal{I}}$, and let $(\mathcal{X}, \mathscr{P}(\Omega))$ be a measurable space. Then the Lebesgue integral can be expressed as a sum using the counting measure by

$$\int f \; d\# = \sum_{i \in \mathcal{I}} f(x_i). \tag{A.33}$$

Similarly, for the Dirac measure δ_x we let $\mathcal{X}$ be a continuous space and let f be a simple function with range $(a_i)_{i\in\mathcal{I}}$ with a corresponding sequence $(A_i : A_i = f^{-1}(a_i))_{i\in\mathcal{I}}$. Then we can instead use the Dirac measure δ_x to express the Lebesgue integral as

$$\begin{aligned} \int f \; d\delta_x &= \sum_{i\in\mathcal{I}} a_i \delta_x(A_i) && \text{(A.34)} \\ &= \sum_{i\in\mathcal{I}} a_i \mathbb{I}_{A_i}(x) && \text{(A.35)} \\ &= f(x). && \text{(A.36)} \end{aligned}$$

Importantly, this can be used to show that we can express any function as a limit of simple functions. The details are outlined in [MW99, Chapter 5].

Definition A.3.21 (Sequence Space)**.** Let $\mathbb{K}$ be either of the fields $\mathbb{R}$ or $\mathbb{C}$, and let $\mathbb{K}^{\mathbb{N}} = \{x : x = \{x_i : x_i \in \mathbb{K}\}_{i\in\mathbb{N}}\}$ be the set of all infinite sequences of scalars of $\mathbb{K}$. Then any linear subspace $\mathcal{X} \subseteq \mathbb{K}^{\mathbb{N}}$ is a *sequence space.*

Remark A.3.22 (ℓ^p-norm and space). Consider the measure space $(\mathbb{N}, \mathscr{P}(\mathbb{N}), \#)$ and let $x : \mathbb{N} \to \mathbb{K}$ with $x \in \mathcal{M}(\mathbb{N})$. Then $\mathcal{I} \subseteq \mathbb{N}$ is an index set, and we have

$$\|x\|_p = \left(\sum_{i\in\mathcal{I}} |x_i|^p \right)^{1/p}. \tag{A.37}$$

This means that $\mathcal{X} = \{x \in \mathcal{M}(\mathbb{N})\}$ is a sequence space. Furthermore, we refer to $\|\bullet\|_p$ as the ℓ^p-norm on $\mathcal{X}$, and denote the induced space $L^p(\mathbb{N})$ w.r.t. the counting measure $\#$ on $\mathscr{P}(\mathbb{N})$ simply as $\ell^p(\mathcal{X})$.

Theorem A.3.23 (L^2-space is Hilbert)**.** *The space $L^2(\mathbb{R})$ is a Hilbert Space with an inner product given by*

$$\langle f, g\rangle = \int_{\mathbb{R}} f\,\overline{g}\; d\nu, \tag{A.38}$$

called the canonical inner product*.*

A proof for Theorem A.3.23 is supplied in [Chr10, p.118].

A.4 Probability Theory

Definition A.4.1 (Pushforward Measure)**.** Let $(\mathcal{X}, \mathcal{A}), (\mathcal{Y}, \mathcal{B})$ be measurable spaces, and let ν be a measure on $(\mathcal{X}, \mathcal{E})$. Let $f : \mathcal{X} \to \mathcal{Y}$ be a mapping such that for all $B \in \mathcal{B}$ we have $f^{\text{-}1}(B) \in \mathcal{A}$. Then the measure

$$f_*\nu(B) = \nu(f^{\text{-}1}(B)) \tag{A.39}$$

is called the *pushforward measure* of ν under f.

Definition A.4.2 (Probability Space)**.** Let $(\Omega, \mathcal{E}, P)$ be a measure space, and let $P(\Omega) = 1$. Then P is a *probability measure*. Furthermore, we call Ω the *sample space*, $\mathcal{E}$ the *event space*. Then $(\Omega, \mathcal{E}, P)$ is a *probability space*.

Definition A.4.3 (Random Elements and Variables)**.** Let $(\Omega, \mathcal{E}, P)$ be a probability space, and let $(\Lambda, \mathcal{F})$ be a measurable space. Let $\mathrm{X} : \Omega \to \Lambda$ be a map such that $\mathrm{X} \in \mathcal{M}(\Omega)$. Then X is called a *random element*. If $(\Lambda, \mathcal{F}) = (\mathbb{R}^d, \mathcal{B})$, then we say X is a *random variable*.

Definition A.4.4 (Probability Distribution)**.** Let $(\Omega, \mathcal{E}, P)$ be a probability space and let $\mathrm{X} : \Omega \to \mathbb{R}^d$ be a random variable. Denote the pushforward measure $\mathrm{X}_*P = \mathbb{P}$ and let $\mathcal{B}$ be the Borel σ-algebra on $\mathbb{R}^d$, such that $\mathbb{P} : \mathcal{B} \to \overline{\mathbb{R}}_{\geq 0}$. Then $\mathbb{P}$ is called a *probability distribution* or *probability law* on X, and we denote this relation as $\mathrm{X} \sim \mathbb{P}$.

Definition A.4.5 (Probability Density Function)**.** Let $(\Omega, \mathcal{E}, P)$ be a probability space, and let $\mathrm{X} : \Omega \to \mathbb{R}^d$ be a random variable. Let $f_\mathrm{X} \in \mathcal{M}(\mathbb{R}^d)$ with $f_\mathrm{X} : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ be a map such that for any Borel set $B \in \mathcal{B}$ we have

$$\mathbb{P}(B) = \int_B f_\mathrm{X}(x)\ dx \tag{A.40}$$

then f_X is a *probability density function* (pdf.) on X.

Definition A.4.6 (Cumulative Distribution Function)**.** Let $(\Omega, \mathcal{E}, P)$ be a probability space, and let $\mathrm{X} : \Omega \to \mathbb{R}^d$ be a random variable. Let $F_\mathrm{X} : \mathbb{R}^d \to [0, 1]$ be a mapping such that for $x \in \mathbb{R}^d$ we have

$$F_\mathrm{X}(x) = P(\{\omega \in \Omega : \mathrm{X}(\omega) \leq x\}). \tag{A.41}$$

Then F is a *cumulative distribution function* (cdf.) on X.

Definition A.4.7 (Discrete Random Variable)**.** Let $(\Omega, \mathcal{E}, P)$ be a probability space, and let $\mathrm{X} : \Omega \to \mathbb{R}^d$ be a random variable. If there exists a countable set $\mathcal{X}$ such that $P(\mathrm{X} \in \mathcal{X}) = 1$, then X is a *discrete random variable*.

Theorem A.4.8 (Change of Variable Formula)**.** *Let $(\mathcal{X}, \mathcal{A}), (\mathcal{Y}, \mathcal{B})$ be measurable spaces, and let f, g be real valued functions on $\mathcal{X}, \mathcal{Y}$ respectively. Let ν be a probability measure on $\mathcal{A}$. Then*

$$\int_\mathcal{X} (g \circ f)(x)\ d\nu(x) = \int_\mathcal{Y} g(y)\ df_*\nu(y). \tag{A.42}$$

A proof for Theorem A.4.8 can be found in [Bil95, p.229].

Definition A.4.9 (Expected Value)**.** Let $(\Omega, \mathcal{E}, P)$ be a probability space, and let $\mathrm{X} : \Omega \to \mathbb{R}$ be a random variable. Let $\mathbb{E} : L^1(\Omega, \mathcal{E}, P) \to \mathbb{R}$ be given by

$$\mathbb{E}[\mathrm{X}] = \int_\Omega \mathrm{X}_i(\omega)\ dP(\omega). \tag{A.43}$$

Definition A.4.10 (Law of the Unconscious Statistician)**.** Let $(\Omega, \mathcal{E}, P)$ be a probability space, let $\mathrm{X} : \Omega \to \mathbb{R}$ be a random variable with pdf. f_X, and let $g : \mathbb{R} \to \mathbb{R}$ with $g \in \mathcal{M}(\mathcal{B})$. Then

$$\begin{aligned}\mathbb{E}[g(\mathrm{X})] &= \int_\Omega (g \circ \mathrm{X})(\omega)\ dP(\omega) && \text{(A.44)}\\ &= \int_\mathbb{R} g(x)\ d\mathrm{X}_*P(x) && \text{(A.45)}\\ &= \int_\mathbb{R} g(x) f_\mathrm{X}(x)\ dx && \text{(A.46)}\end{aligned}$$

is the *expected value* of $g(\mathrm{X})$.

Remark A.4.11. For multivariate random variables $\mathrm{X} : \Omega \to \mathbb{R}^d$ we let $\mathrm{X}_i : \Omega \to \mathbb{R}$ for $i = 1, \dots, d$ and define the expected value as

$$\mathbb{E}[\mathrm{X}] = (\mathbb{E}[\mathrm{X}_1], \mathbb{E}[\mathrm{X}_2], \dots, \mathbb{E}[\mathrm{X}_d])^\intercal. \tag{A.47}$$

Definition A.4.12 (Entropy and Cross-Entropy)**.** Let $(\Omega, \mathcal{E}, P)$ be a probability space, let Q be an alternate probability measure on Ω, and let $\mathrm{X} : \Omega \to \mathbb{R}^d$ be a random variable with probability density function f_X. Let g_X be a probability density function for the distribution $\mathbb{P}_Q$ over X with respect to Q. Then

$$\begin{aligned}\mathrm{H}_f[g_\mathrm{X}] &= \mathbb{E}_f[-\log g_\mathrm{X}(\mathrm{X})] && \text{(A.48)}\\ &= -\int_{\mathbb{R}^d} f_\mathrm{X}(x) \log g_\mathrm{X}(x)\ dx && \text{(A.49)}\end{aligned}$$

is called the *cross-entropy* of g_X with respect to f_X. If $f_\mathrm{X} = g_\mathrm{X}$, we instead call this the *entropy* of f_X and simply write $\mathrm{H}[f_\mathrm{X}]$.

Remark A.4.13. Definition A.4.12 is in fact a simplification. We differentiate between the *Shannon entropy* of a discrete variable given by

$$\mathrm{H}[P] = -\sum_{x \in \mathcal{X}} P(x) \log P(x) \tag{A.50}$$

and continuous *differential entropy* given by

$$\mathrm{H}[f_\mathrm{X}] = -\int_\mathcal{X} f_\mathrm{X}(x) \ln f_\mathrm{X}(x)\ dx. \tag{A.51}$$

Claude Shannon [Sha48] derived entropy for the discrete case, but the continuous analog of differential entropy is not always well defined. Differential entropy is still used, as it is often sufficiently close to the more correct limiting density of discrete points - closely related to the *Kullback-Leibler divergence* (Definition 4.4.5).

A.5 Harmonic Analysis

Definition A.5.1 (Fourier Transform)**.** Let $x \in L^1(\mathbb{R}^d)$. The *Fourier transform* of x is given by

$$\tilde{x}(\xi) = \int_{\mathbb{R}} e^{-2\pi i s\cdot\xi} x(s)\ ds. \tag{A.52}$$

Theorem A.5.2 (Plancherel's Theorem)**.** *There exists a unique* Fourier operator $\mathcal{F} : L^2(\mathbb{R}^d) \to L^2(\mathbb{R}^d)$ *such that*

(i) *for each* $x \in L^2(\mathbb{R}^d) \cap L^1(\mathbb{R}^d)$, $\mathcal{F}x = \tilde{x}$ *almost everywhere,*

(ii) *for each* $x \in L^2(\mathbb{R}^d)$, $\lim_{\epsilon\to\infty} \|\mathcal{F}x - \widetilde{\mathbb{I}_{\mathbb{B}(0,\epsilon)}x}\|_2 = 0$,

(iii) $\|\mathcal{F}x\|_2 = \|x\|_2$ *for all* $x \in L^2(\mathbb{R}^d)$,

(iv) $\langle \mathcal{F}x, \mathcal{F}y\rangle = \langle x, y\rangle$ *for* $x, y \in L^2(\mathbb{R}^d)$,

(v) $(\mathcal{F}^2 x)(s) = x(-s)$ *almost everywhere for all* $x \in L^2(\mathbb{R}^d)$.

[MW99, pp.583–586] gives a proof for Theorem A.5.2, as well as more discussion on Fourier transform for $L^2(\mathbb{R})$. The theorem can be extended to hold for d dimensions using Fubini (Theorem A.7.5).

Lemma A.5.3 (Inverse Fourier Transform)**.** *Let $\tilde{x}$ be the Fourier transform of a function $x \in L^2(\mathbb{R})$. Then the* inverse Fourier transform *is given by*

$$x(s) = \int_{\mathbb{R}} e^{2\pi i \xi\cdot s} \tilde{x}(\xi)\ d\xi. \tag{A.53}$$

Proof. We have

$$\int_{\mathbb{R}} e^{2\pi i \xi\cdot s} \tilde{x}(\xi)\ d\xi = \int_{\mathbb{R}} \int_{\mathbb{R}} e^{-2\pi i \xi(t-s)} x(t)\ dt\, d\xi \tag{A.54}$$

and by Fubini, we can exchange the order of integration, yielding

$$\int_{\mathbb{R}} x(t) \int_{\mathbb{R}} e^{-2\pi i \xi(t-s)}\ d\xi\, dt = \int_{\mathbb{R}} x(t)\delta(t-s)\ dt = x(s), \tag{A.55}$$

where δ is the Dirac delta function. ■

Corollary A.5.4 (Fourier Operator is Unitary)**.** *The Fourier operator $\mathcal{F}$ is a unitary operator.*

This follows directly from Theorems A.2.5 and A.3.23, Definition A.2.12, and Lemma A.5.3.

A.6 Matrix Groups and Manifolds

Theorem A.6.1 (Inner Product Space on General Linear Algebra)**.** *Let* $\mathfrak{gl}(n)$ *be the set of all* $n \times n$ *matrices over* $\mathbb{C}$*. Moreover, let* $U, V \in \mathfrak{gl}(n)$ *and let* $\langle \bullet, \bullet \rangle : \mathfrak{gl}(n) \times \mathfrak{gl}(n) \to \mathbb{C}$ *be an inner product given by*

$$\langle U, V \rangle = \mathrm{Tr}(UV^*). \tag{A.56}$$

Then $\mathfrak{gl}(n)$ *is an inner product space w.r.t.* $\langle \bullet, \bullet \rangle$*, and we call* $\mathfrak{gl}(n)$ *the* general linear algebra*.*

Proof. We need to show the properties of Definition A.1.13. Firstly, we have

$$\begin{aligned}
\langle aU + bV, W \rangle &= \mathrm{Tr}\big((aA + bV)W\big) && \text{(A.57)}\\
&= \sum_{i=j=1}^{n} (aA_{ij} + bV_{ij})W_{ij} && \text{(A.58)}\\
&= \sum_{i=j=1}^{n} aA_{ij}W_{ij} + bV_{ij}W_{ij} && \text{(A.59)}\\
&= \langle aU, W \rangle + \langle bV, W \rangle, && \text{(A.60)}
\end{aligned}$$

satisfying property Property A.1.13.i. Furthermore, as $\mathrm{Tr}(UV^*) = \mathrm{Tr}(\overline{UV^*}) = \mathrm{Tr}(VU^*)$, Property A.2.1.ii is satisfied. Lastly, from the properties of the trace operator, we have that $\mathrm{Tr}(UU^*) \geq 0$ with equality iff. $U = 0$. Thus, $\mathfrak{gl}(n)$ is an inner product space. ■

Corollary A.6.2 (Induced Norm of $\mathfrak{gl}(n)$)**.** *The induced norm on* $\mathfrak{gl}(n)$ *is given by* $\sqrt{\mathrm{Tr}(UU^*)}$*.*

This follows from Lemma A.1.14.

Definition A.6.3 (Group)**.** Let $\mathcal{G}$ be a set, and let $\bullet \cdot \bullet : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ be an associated binary operator such that for all $a, b, c \in \mathcal{G}$ we have

(i) (Associative) $(a \cdot b) \cdot c = a \cdot (b \cdot c)$,

(ii) (Identity) there exists $e \in \mathcal{G}$ such that $e \cdot a = a$,

(iii) (Inverse) there exists $a^{-1} \in \mathcal{G}$ such that $a \cdot a^{-1} = a^{-1} \cdot a = e$.

Then $\mathcal{G}$ is a *group*. If the binary operator is commutative, i.e. $a \cdot b = b \cdot a$, then we call $\mathcal{G}$ an *Abelian group*.

Definition A.6.4 (Homomorphisms and Isomorphisms)**.** Let $\mathcal{G}, \mathcal{H}$ be groups with respective operations $\bullet \cdot_g \bullet$ and $\bullet \cdot_h \bullet$. Furthermore, let $f : \mathcal{G} \to \mathcal{H}$ be a mapping. If for all $a, b \in \mathcal{G}$ we have

$$f(a \cdot_g b) = f(a) \cdot_h f(b) \tag{A.61}$$

then we say that f is a *homomorphism*. If f is bijective, then f is called an *isomorphism*.

Definition A.6.5 (Subgroup)**.** Let $\mathcal{G}$ be a group, and let $\mathcal{H} \subset \mathcal{G}$. If $\mathcal{H}$ is closed under the group operation of $\mathcal{G}$, then $\mathcal{H}$ is a *subgroup of* $\mathcal{G}$.

Lemma A.6.6 ($\mathcal{SL}(n)$ is subgroup of $\mathcal{GL}(n)$)**.** *Let $\mathcal{SL}(n) \subset \mathcal{GL}(n)$ such that $\mathcal{SL}(n) = \{W \in \mathcal{GL}(n) : \det(W) = 1\}$. Then $\mathcal{SL}(n)$ is a subgroup of $\mathcal{GL}(n)$, called the* special linear group*.*

Proof. $\mathcal{SL}(n)$ is naturally a proper subset of $\mathcal{GL}(n)$, and as $\det(I) = 1$, we have $I \in \mathcal{SL}(n)$, satisfying Property A.6.3.ii. Clearly Property A.6.3.i follows from $\mathcal{GL}(n)$. From the properties of the determinant, we have that $\det(W^{-1}) = \det(W)^{-1}$, and $\det(UV) = \det(U)\det(V)$, so $\mathcal{SL}(n)$ is closed, and Property A.6.3.iii is satisfied. Thus $\mathcal{SL}(n)$ is a subgroup of $\mathcal{GL}(n)$. ■

Lemma A.6.7 ($\mathcal{O}(n)$ is a subgroup of $\mathcal{GL}(n)$)**.** *Let $\mathcal{O}(n)$ be the set of real orthogonal $n \times n$ matrices, i.e. $\mathcal{O}(n) = \{W \in \mathcal{GL}(n) : W^\intercal = W^{-1}\}$. Then $\mathcal{O}(n)$ is a subgroup of $\mathcal{GL}(n)$ called the* orthogonal group*.*

Proof. $\mathcal{O}(n)$ is naturally a proper subset of $\mathcal{GL}(n)$, and as $I^\intercal = I^{-1} = I$, we necessarily have $I \in \mathcal{O}(n)$. Furthermore, Property A.6.3.i and Property A.6.3.iii naturally follow by the group operation and definition. Thus, for $\mathcal{O}(n)$ to be a group, we need to show that it is closed under matrix multiplication. For all $U, V \in \mathcal{O}(n)$ we have $UV^\intercal(UV^\intercal)^\intercal = UV^\intercal(UV^\intercal)^{-1} = UV^\intercal VU^\intercal = I$, so $\mathcal{O}(n)$ is closed, and thus a subgroup of $\mathcal{GL}(n)$. ■

Remark A.6.8. The orthogonal and special orthogonal group can be extended to complex valued matrices by exchanging the transposed matrices $W^\intercal$ with the conjugate transpose W^*. The corresponding groups to $\mathcal{O}(n), \mathcal{SO}(n)$ are the unitary group $\mathcal{U}(n)$ and special unitary group $\mathcal{SU}(n)$, respectively. For more details, we refer to [Hal15, pp.6–8].

Definition A.6.9 (Matrix Lie Group). Let $\mathcal{G} \subset \mathcal{GL}(n)$, and let $(W_i \in \mathcal{G})_{i\in\mathcal{I}}$ be a convergent sequence of matrices, such that $W_n \to W$ as $n \to \infty$. If either $W \in \mathcal{G}$ or $W \notin \mathcal{GL}(n)$, then $\mathcal{G}$ is a *matrix Lie group*. More concretely, the matrix Lie groups are precisely the closed subsets of $\mathcal{GL}(n)$.

Lemma A.6.10 (Matrix Lie Groups)**.** *The groups $\mathcal{GL}(n)$ and $\mathcal{SO}(n)$ are matrix Lie groups.*

Proof. As $\mathcal{GL}(n)$ is a subset of itself, it is necessarily a matrix Lie group. For $\mathcal{SL}(n)$ we note that a sequence of matrices $(U_i)_{i\in\mathcal{I}}$ with $\det(U_i) = 1$ for all $i \in \mathcal{I}$ implies that for the limit $U_i \to U$ we necessarily have $\det(U) = 1$, so $U \in \mathcal{SL}(n)$. Next, recall that for $W \in \mathcal{O}(n)$ we have $W^* = W^\intercal$. Thus, for $\mathcal{O}(n), \mathcal{U}(n)$ we have that $W \in \mathcal{U}(n)$ if and only if $W^*W = I$. Now let $(V_i)_{i\in\mathcal{I}}$ with $V_n \to V$ as $n \to \infty$. Next, define $(V_i^*)_{i\in\mathcal{I}}$ similarly. Then $V_n^* \to V^*$, and as $V^*V = I$ we have that $V \in \mathcal{U}(n)$, so $\mathcal{U}(n)$ and $\mathcal{O}(n)$ are matrix Lie groups. The arguments for $\mathcal{SO}(n)$ and $\mathcal{SU}(n)$ naturally follow from the proofs for $\mathcal{SL}(n), \mathcal{O}(n)$ and $\mathcal{U}(n)$, so all listed groups are matrix Lie groups. ■

Lemma A.6.11 (Compact Matrix Lie Groups)**.** *The groups $\mathcal{O}(n)$, $\mathcal{SO}(n)$, $\mathcal{U}(n)$ and $\mathcal{SU}(n)$ are compact with respect to $\mathfrak{gl}(n)$.*

Proof. Clearly, the listed matrix Lie groups are closed in $\mathfrak{gl}(n)$. As $|W_{ij}| \leq 1$ for all $i, j \leq n$, they are necessarily also bounded. Then by the Heine-Borel theorem (Theorem A.7.14) they are compact in $\mathfrak{gl}(n)$. ■

Definition A.6.12 (Topological Space)**.** Let $\mathcal{X}$ be a set, and let $\mathscr{T}$ be a collection of subsets of $\mathcal{X}$ such that

(i) $\emptyset \in \mathscr{T}$ and $\mathcal{X} \in \mathscr{T}$,

(ii) $\mathscr{T}$ is closed under finite or infinite unions,

(iii) $\mathscr{T}$ is closed under finite intersections.

Then $\mathscr{T}$ is called a *topology* on $\mathcal{X}$, the elements of $\mathscr{T}$ are called *open sets*, and $(\mathcal{X}, \mathscr{T})$ is a *topological space.*

Definition A.6.13 (Subspace Topology)**.** Let $(\mathcal{X}, \mathscr{T}_x)$ be a topological space, and let $\mathcal{Y} \subset \mathcal{X}$ be nonempty. Furthermore, let

$$\mathscr{T}_y = \{\mathcal{Y} \cup S : S \in \mathscr{T}_x\} \subset \mathscr{P}(\mathcal{X}). \tag{A.62}$$

Then $\mathscr{T}_y$ is called the *subspace topology* on $\mathcal{Y}$ induced by $\mathscr{T}_x$, and we say that $(\mathcal{Y}, \mathscr{T}_y)$ is a *topological subspace* of $(\mathcal{X}, \mathscr{T}_x)$.

Definition A.6.14 (Analytic Basis and Second-Countable Spaces)**.** Let $(\mathcal{X}, \mathscr{T})$ be a topological space, and let $\mathscr{B} \subseteq \mathscr{T}$. If there exists some $\mathcal{T} \subseteq \mathscr{B}$ such that for all $S \in \mathscr{T}$ we have

$$S = \bigcup_{T \in \mathcal{T}} T, \tag{A.63}$$

then $\mathscr{B}$ is an *analytic basis for* $\mathscr{T}$. If $\mathscr{B}$ is a countable set, then we say that $(\mathcal{X}, \mathscr{T})$ is *second-countable.*

Definition A.6.15 (Hausdorff Space)**.** Let $(\mathcal{X}, \mathscr{T})$ be a topological space. If for all $x, y \in \mathcal{X}$ with $x \neq y$ there exists $S, T \in \mathscr{T}$ such that $x \in S, y \in T$ and $S \cap T = \emptyset$, then $(\mathcal{X}, \mathscr{T})$ is a *Hausdorff space.*

Definition A.6.16 (Homeomorphism)**.** Let $(\mathcal{X}, \mathscr{T}_x), (\mathcal{Y}, \mathscr{T}_y)$ be topological spaces, and let $f : \mathcal{X} \to \mathcal{Y}$ be a bijection. If f and f^{-1} are continuous under $\mathscr{T}_x, \mathscr{T}_y$, we say that f is a *homeomorphism.*

Definition A.6.17 (Neighbourhood)**.** Let $(\mathcal{X}, \mathscr{T})$ be a topological space, and let $x \in S \subset N \subset \mathcal{X}$ for some $S \in \mathscr{T}$. Then N is a *neighbourhood of* x. If $N \in \mathscr{T}$ we say that N is an *open neighbourhood.*

Definition A.6.18 (Locally Euclidean Space)**.** Let $(\mathcal{X}, \mathscr{T})$ be a topological space. If each point in $\mathcal{X}$ has an open neighbourhood which is homeomorphic to an open subset of $\mathbb{K}^n$, we say that $(\mathcal{X}, \mathscr{T})$ is *locally Euclidean.*

Definition A.6.19 (Coordinate Chart)**.** Let $(\mathcal{X}, \mathscr{T})$, and let $S \subseteq \mathscr{T}, R \subseteq \mathbb{R}^n$ such that R is open. Let $f : S \to R$ be a homeomorphism. Then S is a *coordinate neighbourhood* and (S, f) is a *coordinate chart* of $(\mathcal{X}, \mathscr{T})$.

Definition A.6.20 (Smooth Differentiable Structure)**.** Let $(\mathcal{X}, \mathscr{T})$ be a topological space and let $\mathscr{A} = \{(S_\alpha, f_\alpha) : (S_\alpha, f_\alpha) \text{ is a coordinate chart}\}$ such that

(i) (Full Covering) $\mathcal{X} = \bigcup_\alpha S_\alpha$,

(ii) (Compatibility) for all a, b we have $f_b \circ f_a^{-1} \in \mathcal{C}^\infty$,

(iii) (Maximality) $(S, f) \in \mathscr{A}$ if (S, f) is compatible with all $(S_\alpha, f_\alpha) \in \mathscr{A}$.

Then $\mathscr{A}$ is called a $\mathcal{C}^\infty$*-atlas*, and we say that $(\mathcal{X}, \mathscr{T})$ has a *smoooth differentiable structure*.

Definition A.6.21 (Manifold)**.** Let $\mathcal{M} = (\mathcal{X}, \mathscr{T})$ be a second-countable locally Euclidean Hausdorff space. Then we call $\mathcal{M}$ a *manifold*. If in addition, $\mathcal{M}$ is equipped with a smooth differentiable structure, we call $\mathcal{M}$ a *smooth manifold*.

Definition A.6.22 (Smooth Maps and Diffeomorphisms)**.** Let $\mathcal{M}, \mathcal{N}$ be smooth manifolds, and let $f : \mathcal{M} \to \mathcal{N}$. If for any $p \in \mathcal{M}$ there exists a chart (S, g) on $\mathcal{M}$ and (T, h) on $\mathcal{N}$ such that for $p \in S, f(p) \in T$ we have

$$h \circ f \circ g^{-1} \in \mathcal{C}^\infty, \tag{A.64}$$

then f is a *smooth map*. If in addition, f is bijective with f^{-1} smooth, then we say that f is a $\mathcal{C}^\infty$*-diffeomorphism*.

Theorem A.6.23 (Open Subset of Smooth Manifold is Smooth Manifold)**.** *Let $\mathcal{M}$ be a smooth manifold, and let $M \in \mathscr{T}$. Then M is a smooth manifold.*

Proof. First, we show that M is Hausdorff. Let $s, t \in M$ such that $s \in S, t \in T$ with $S \cap T = \emptyset$ for $S, T \in \mathscr{T}$. Then $s \in S \cap M, t \in T \cap M$ and $(T \cap M) \cap (S \cap M) = (S \cap T) \cap M = \emptyset$, so M is Hausdorff. Secondly, if $\mathcal{M}$ is second-countable, then M is trivially also second-countable by the properties of the subspace topology (Definition A.6.13).

We now need to show that an open subset of a smooth manifold inherits the differentiable structure. Let $\{(S_\alpha, f_\alpha)\}$ be an atlas for some manifold $\mathcal{M}$, and let $M \subset \mathcal{M}$ be open. Let $f_\alpha\big|_{S_\alpha \cap M} : S_\alpha \cap M \to \mathbb{R}^n$ denote the restriction of the homeomorphism f_α to $S_\alpha \cap M$. Then $\{(S_\alpha \cap M, f_\alpha\big|_{S_\alpha \cap M})\}$ is necessarily an atlas for M, so an open subset of a smooth manifold is a smooth manifold. ■

Definition A.6.24 (Connected and Simply Connected Groups)**.** Let $\mathcal{G}$ be a group. If for all $U, V \in \mathcal{G}$ and $t \in [a, b]$ for $a, b \in \mathbb{R}$ there exists a continuous map $U(t) : [a, b] \to \mathcal{G}$ with $U(a) = U$ and $U(b) = V$, then we say that $\mathcal{G}$ is *connected*, and we call $\{U(t) : t \in [a, b]\}$ the *path* between two elements U, V.

Definition A.6.25 (Givens Rotation)**.** Let $G_n(\theta; p, q) \in \mathcal{SO}(n)$ be a parametrized matrix which for $1 \le q < p \le n$ and $1 \le i, j \le n$ is given by

$$G_n(\theta; p, q)_{ij} = \begin{cases} 1, & \text{for } i = j \notin \{p, q\}; \\ \cos(\theta), & \text{for } i = j \in \{p, q\}; \\ -\sin(\theta), & \text{for } i = p, j = q; \\ \sin(\theta), & \text{for } i = q, j = p; \\ 0, & \text{otherwise.} \end{cases} \tag{A.65}$$

Then $G_n(\theta; p, q)$ is called a *Givens rotation*.

Proposition A.6.26 (Path-connected Lie Groups)**.** *The groups $\mathcal{GL}(n)$, $\mathcal{SL}(n)$, $\mathcal{SO}(n)$, $\mathcal{U}(n)$, $\mathcal{SU}(n)$ are path-connected.*

Proof. A proof for all groups except $\mathcal{SO}(n)$ can be found in [Hal15, pp.17–18], so we will only give a proof for $\mathcal{SO}(n)$. Note that connected paths are transitive, i.e. if U, V are path-connected and V, W are path-connected, then U, W are necessarily path-connected. Let $U, V \in \mathcal{SO}(n)$ be elements which only differ in the elements of rows and columns p, q. Then there exists a path given by $\{G_n(\theta, p, q)U : \theta \in [0, b]\}$ where $G_n(\theta, p, q)$ is a Givens rotation such that $G_n(0, p, q)U = U$ and $G_n(b, p, q)U = V$. Applying this process recursively we can connect the paths to another matrix W with differing elements p', q', thus the rest of the proof follows inductively. ■

Definition A.6.27 (Lie Algebra)**.** Let $\mathfrak{g}$ be a linear space over $\mathbb{K}$, and let $[\![\bullet, \bullet]\!] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ be a binary operator such that

(i) $[\![g, g]\!] = 0$ for all $g \in \mathfrak{g}$,

(ii) $[\![g, [\![h, k]\!]]\!] + [\![h, [\![k, g]\!]]\!] + [\![k, [\![g, h]\!]]\!] = 0$ for all $g, h, k \in \mathfrak{g}$.

Then $\mathfrak{g}$ with $[\![\bullet, \bullet]\!]$ is a *Lie algebra.*

Definition A.6.28 (Lie Sub-Algebra)**.** Let $\mathfrak{g}$ be a Lie algebra, and let $\mathfrak{h} \subset \mathfrak{g}$. If for all $g, h \in \mathfrak{h}$ we have

$$[\![h, h]\!] \in \mathfrak{h}, \tag{A.66}$$

then $\mathfrak{h}$ is a *Lie sub-algebra* of $\mathfrak{g}$.

Corollary A.6.29 (Matrix Commutator)**.** *Let $U, V \in \mathfrak{gl}(n)$. The Lie bracket on $\mathfrak{gl}(n)$ is given by $[\![U, V]\!] = UV - VU$, so $\mathfrak{gl}(n)$ is a Lie algebra.*

Theorem A.6.30 (Matrix Lie Groups and Lie Algebras)**.** *Let $\mathfrak{g}(n)$ be a Lie algebra of matrices. Then $\mathcal{G}(n) = \{\exp(tW) : W \in \mathfrak{g}(n), t \in \mathbb{R}\}$ is a matrix Lie group. Furthermore, we have the mappings*

(i) $\exp : \mathfrak{gl}(n) \times \mathbb{R} \to \mathcal{GL}(n)$,

(ii) $\exp : \mathfrak{sl}(n) \times \mathbb{R} \to \mathcal{SL}(n)$, *where* $\mathfrak{sl}(n) = \{W \in \mathfrak{gl}(n) : \mathrm{Tr}(W) = 0\}$,

(iii) $\exp : \mathfrak{u}(n) \times \mathbb{R} \to \mathcal{U}(n)$, *where* $\mathfrak{u}(n) = \{W \in \mathfrak{gl}(n) : W = -W^*\}$,

(iv) $\exp : \mathfrak{su}(n) \times \mathbb{R} \to \mathcal{SU}(n)$, *where* $\mathfrak{u}(n) = \{W \in \mathfrak{u}(n) \cap \mathfrak{sl}(n)\}$,

(v) $\exp : \mathfrak{so}(n) \times \mathbb{R} \to \mathcal{O}(n)$, *where* $\mathfrak{u}(n) = \{W \in \mathfrak{su}(n) : W_{ij} \in \mathbb{R}\}$,

(vi) $\exp : \mathfrak{so}(n) \times \mathbb{R} \to \mathcal{SO}(n)$.

Theorem A.6.30 is demonstrated part by part in [Hal15, Chapters 1-3].

Definition A.6.31 (Derivations and Tangent Vectors)**.** Let $\mathcal{M}$ be a smooth manifold and let $p \in \mathcal{M}$. Let $D_p : \mathcal{C}^\infty(\mathcal{M}) \to \mathbb{R}$ be a linear transformation such that

$$D_p(fg) = D_p(f)g(p) + f(p)D_p(g). \tag{A.67}$$

Then D_p is called a *derivation* or a *tangent vector* at p.

Proposition A.6.32 (Tangent Space)**.** *Let $\mathcal{M}$ be a smooth manifold, let $p \in \mathcal{M}$ and let $T_p\mathcal{M} = \{D_p(f) : f \in \mathcal{C}^\infty(\mathcal{M})\}$. Then $T_p\mathcal{M}$ is a real linear space called the* tangent space *of $\mathcal{M}$ at p.*

Proof. Let $D_p, D'_p \in T_p\mathcal{M}$. By definition, D_p, D'_p are linear transformations, satisfying Definition A.1.2. However, we need to show that scalar multiplication of a linear combination satisifies Definition A.6.31. Let $a \in \mathbb{R}$. Then we have

$$\begin{aligned}(aD_p + D'_p)(fg) &= aD_p(fg) + D'_p(fg) && \text{(A.68)}\\ &= aD_p(f)g(p) + af(p)D_p(g)\\ &\quad + D'_p(f)g(p) + f(p)D'_p(g) && \text{(A.69)}\\ &= (aD_p + D'_p)(f)g(p) + f(p)(aD_p + D'_p)(g), && \text{(A.70)}\end{aligned}$$

so $T_p\mathcal{M}$ is a real linear space. ■

Definition A.6.33 (Tangent Bundle)**.** Let $\mathcal{M}$ be a smooth manifold, and let $T_p\mathcal{M}$ be the tangent space of $p \in \mathcal{M}$. Let $T\mathcal{M} = \bigsqcup_{p\in\mathcal{M}} T_p\mathcal{M}$, i.e. the disjoint union of all tangent spaces on $\mathcal{M}$. Then $T\mathcal{M}$ is called the *tangent bundle* of $\mathcal{M}$.

Definition A.6.34 (Complexification of Linear Space)**.** Let $\mathcal{X}$ be a linear space, and let $\mathcal{X}_{\mathbb{C}} = \{x_1 + ix_2 : x_1, x_2 \in \mathcal{X}\}$. Then $\mathcal{X}_{\mathbb{C}}$ is called the *complexification of* $\mathcal{X}$.

Definition A.6.35 (Reductive Lie Algebra)**.** Let $\mathfrak{g}$ be a Lie algebra over $\mathbb{C}$. If there exists a compact Lie group $\mathcal{H}$ with Lie algebra $\mathfrak{h}$ such that $\mathfrak{g} \simeq \mathfrak{h}_{\mathbb{C}}$, then $\mathfrak{g}$ is a *reductive Lie algebra.*

Proposition A.6.36 (Adjoint Bi-Invariant Inner Product)**.** *Let $\mathfrak{g} \simeq \mathfrak{g}_{\mathbb{C}}$ be a reductive Lie algebra. Then for all $W \in \mathfrak{h}, U, V \in \mathfrak{g}$ there exists an inner product on $\mathfrak{g}$ that is real valued on $\mathfrak{h}$ such that*

$$\langle WUW^{-1}, V\rangle = -\langle U, WVW^{-1}\rangle, \tag{A.71}$$

called the adjoint bi-invariant inner product*.*

A proof can be found in [Hal15, Proposition 7.4].

Lemma A.6.37 ($\mathfrak{gl}(n)$ is Reductive)**.** *The Lie algebra $\mathfrak{gl}(n)$ is reductive.*

A proof for Lemma A.6.37 is provided in [Hal15, Example 7.3].

Corollary A.6.38 (Lie Manifolds are Riemannian)**.** *The matrix Lie manifolds are Riemannian manifolds and admit a Riemannian metric.*

Remark A.6.39. The proof of Corollary A.6.38 follows from Theorems 6.4.7 and A.6.1, Proposition A.6.36, and Lemma A.6.37.

A.7 Selected Fundamental Results and Theorems

Theorem A.7.1 (Cauchy-Schwartz)**.** *Let $\mathcal{X}$ be an inner product space. Then for all $x, y \in \mathcal{X}$ we have*

(i) $\langle x+y, x+y\rangle = \langle x,x\rangle + \mathscr{R}(\langle x,y\rangle) + \langle y,y\rangle$,

(ii) $|\langle x,y\rangle|^2 \leq \langle x,x\rangle\langle y,y\rangle$,

where $\mathscr{R}(c)$ denotes the real part of a complex number $c \in \mathbb{C}$.

A proof for Theorem A.7.1 is derived in [MW99, p.535].

Theorem A.7.2 (Lebesgue's Dominated Convergence Theorem)**.** *Let $(\mathcal{X}, \mathcal{A}, \nu)$ be a measure space, let $\{f_n \in L^1(\mathcal{X})\}_{n\in\mathbb{N}}$ be a sequence of complex valued functions that converges ν-*a.e.*. If there exists non-negative $g \geq |f_n|$ which holds ν-*a.e. *for all $n \in \mathbb{N}$, then for all $A \in \mathcal{A}$ we have*

$$\int_A \lim_{n\to\infty} f_n \, d\nu = \lim_{n\to\infty} \int_A f_n \, d\nu \tag{A.72}$$

A proof of Theorem A.7.2 is provided in [Rud87, pp.26–27].

Lemma A.7.3 (Riemann-Lebesgue)**.** *Let $x \in L^1$ and let $\tilde{x}$ be the Fourier transform of x. Then for all $x \in L^1$ we have*

$$\lim_{|\xi|\to\infty} \tilde{x}(\xi) = 0 \tag{A.73}$$

Proof. First, assume that $x = \mathbb{I}_{[a,b]}$. Then

$$\tilde{x}(\xi) = \frac{e^{-i\xi a} - e^{-i\xi b}}{-i\xi}.$$

which clearly goes to 0 as ξ goes to infinity. Next, we let

$$x = \sum_{n=1}^{N} c_n \mathbb{I}_{[a_n, b_n]}.$$

It follows that $\lim_{|\xi|\to\infty} \hat{x}(\xi) = 0$, and since $x \in L^1$ and simple functions are dense in L^1, we must also have $\|x - f\|_1 < \varepsilon/2$ for some simple function f and $\varepsilon > 0$. Then there must exist some $M \in \mathbb{N}$ such that $|\tilde{f}(\xi)| < \varepsilon/2$. Finally, for the same M, the triangle inequality yields

$$\begin{aligned}
|\tilde{x}(\xi)| &= |\tilde{x}(\xi) - \tilde{f}(\xi) + \tilde{f}(\xi)| \\
&\leq |\tilde{x}(\xi) - \tilde{f}(\xi)| + \frac{\varepsilon}{2} \\
&\leq \left| \int_{\mathbb{R}} (x(s) - f(s)) e^{-i\xi s} \, ds \right| + \frac{\varepsilon}{2} \\
&\leq \int_{\mathbb{R}} |x(s) - f(s)| \, ds + \frac{\varepsilon}{2} \\
&= \|x - f\|_{L^1} + \frac{\varepsilon}{2} \leq \varepsilon
\end{aligned}$$

which concludes our proof. ■

Definition A.7.4 (σ-finite Measure)**.** Let $(\mathcal{X}, \mathcal{A}, \nu)$ be a measure space. If there exists a sequence $\{A_i\}_{i\in\mathcal{I}}$ of $\mathcal{A}$-measurable sets such that

(i) $\cup_{i\in\mathcal{I}} A_i = \mathcal{X}$,

(ii) $\nu(A_i) < \infty$ for all $i \in \mathcal{I}$,

then $(\mathcal{X}, \mathcal{A}, \nu)$ is a *σ-finite measure space.*

Theorem A.7.5 (Fubini's Theorem)**.** *Let $(\mathcal{X}, \mathcal{A}, \nu), (\mathcal{Y}, \mathcal{B}, \rho)$ be σ-finite measure spaces. Let $\mathcal{X} \times \mathcal{Y}$ be a product space, and let $f \in \mathcal{M}(\mathcal{X} \times \mathcal{Y})$. Then*

$$\int_{\mathcal{X}\times\mathcal{Y}} f(x,y)\ (\nu \times \rho)(x,y) = \int_{\mathcal{Y}} \left[\int_{\mathcal{X}} f(x,y)\ \nu(x)\right] \rho(y) \tag{A.74}$$

$$= \int_{\mathcal{X}} \left[\int_{\mathcal{Y}} f(x,y)\ \rho(y)\right] \nu(x). \tag{A.75}$$

A proof for Theorem A.7.5 can be found in [MW99, pp.247–248].

Theorem A.7.6. *The Hilbert-Schmidt integral operator is compact.*

Proof. Note that since $w \in L^2(\mathcal{S} \times \mathcal{T})$ we have $\int_{\mathcal{T}} \int_{\mathcal{S}} |w(s,t)|^2\ ds\, dt < \infty$. By Fubini (Theorem A.7.5) and Cauchy-Schwartz (Theorem A.7.1) we can construct

$$\int_{\mathcal{T}} \left|\int_{\mathcal{S}} w(s,t)x(s)\ ds\right|^2 dt \leq \int_{\mathcal{T}} \left(\left|\int_{\mathcal{S}} w(s,t)\ ds\right|^2 \cdot \left|\int_{\mathcal{S}} x(s)\ ds\right|^2\right) dt$$
$$= \|w\|_2^2 \cdot \|x\|_2^2,$$

where we let $\|w\|_2$ be the norm over $L^2(\mathcal{S} \times \mathcal{T})$. This implies the inequality

$$\|W\|_{\text{op}} \leq \|w\|_2. \tag{A.76}$$

Now let $(w_n)_{n\in\mathbb{N}} \subset L^2(S \times T)$ be a sequence of functions such that $w_n \to w$ almost everywhere. Then, for simple functions $f_i \in L^2(S), g_i \in L^2(T)$ we let

$$w_n(s,t) = \sum_{i<n} f_i(s) g_i(t),$$

and note that simple functions are dense in L^2 [KF61, pp.97–99]. Then $W_n x(t) \to W x(t)$ weakly, so from Equation (A.76) and Lebesgue's dominated convergence theorem (Theorem A.7.2) it will also converge in norm. Thus for any bounded sequence $(x_m)_{m\in\mathbb{N}} \subset \mathcal{X}$ we can construct a sequence $(W x_m)_{m\in\mathbb{N}} \subset \mathcal{Y}$ with a convergent subsequence $(W_n x_m)_{n,m\in\mathbb{N}} \to (W x_m)_{m\in\mathbb{N}}$. Then it follows that W is a compact linear operator. ∎

Theorem A.7.7 (The Convolution Theorem)**.** *Let $w, x \in L^1(\mathbb{R}^d)$ and let $\tilde{w}, \tilde{x}$ be their respective Fourier transforms (Definition A.5.1). Then*

$$\widetilde{w * x} = \tilde{w} \cdot \tilde{x}, \tag{A.77}$$

so convolution over time corresponds to a (point-wise) multiplication over frequency.

Proof. Firstly, note that

$$(\tilde{w} \cdot \tilde{x})(\xi) = \int_{\mathcal{Z}} w(z)e^{-2\pi i z\cdot\xi}\, dz \cdot \int_{\mathcal{S}} x(s)e^{-2\pi i s\cdot\xi}\, ds.$$

Rewriting $\widetilde{w * x}$ using Fubini (Theorem A.7.5) yields

$$\begin{aligned}(\widetilde{w * x})(\xi) &= \int_{\mathcal{S}}\int_{\mathcal{T}} w(t-s)x(s)e^{-2\pi i t\cdot\xi}\, dt\, ds \\ &= \int_{\mathcal{S}} x(s)\left[\int_{\mathcal{T}} w(t-s)e^{-2\pi i t\cdot\xi}\, dt\right] ds \\ &= \int_{\mathcal{S}} x(s)\left[\int_{\mathcal{Z}} w(z)e^{-2\pi i (z+s)\cdot\xi}\, dz\right] ds \\ &= \int_{\mathcal{Z}} w(z)e^{-2\pi i z\cdot\xi}\, dz \cdot \int_{\mathcal{S}} x(s)e^{-2\pi i s\cdot\xi}\, ds,\end{aligned}$$

as we wanted. ■

Definition A.7.8 (Kronecker Tensor Product)**.** Let $U \in \mathbb{K}^{m\times n}$ and $V \in \mathbb{K}^{p\times q}$ be matrices. Then the mapping $\bullet \otimes \bullet : \mathbb{K}^{m\times n} \times \mathbb{K}^{p\times q} \to \mathbb{K}^{mp\times nq}$ given by

$$(U \otimes V) = \begin{bmatrix} u_1 1V & \dots & u_1 nV \\ \vdots & \ddots & \vdots \\ u_m 1V & \dots & u_m nV \end{bmatrix} \tag{A.78}$$

is called the *Kronecker Tensor product* of U, V.

Definition A.7.9 (Cauchy Power Series Product)**.** Let $p_\infty(\mathbb{K})$ be the set of power series expansions, and let $f, g \in p_\infty(\mathbb{K})$, such that

$$f(x) = \sum_{k\in\mathbb{Z}_{\geq 0}} a_k x^k \tag{A.79}$$

$$g(x) = \sum_{k\in\mathbb{N}_{\geq 0}} b_k x^k, \tag{A.80}$$

for $a_k, b_k \in \mathbb{K}$ for $k \in \mathbb{Z}_{\geq 0}$. Let $\bullet \cdot \bullet : p_\infty(\mathbb{K}) \times p_\infty(\mathbb{K}) \to p_\infty(\mathbb{K})$ be a mapping such that for any $f, g \in p_\infty(\mathbb{K})$ we have

$$f \cdot g = \left(\sum_{k\in\mathbb{Z}_{\geq 0}} a_k x^k\right) \cdot \left(\sum_{k\in\mathbb{N}} b_k x^k\right) \tag{A.81}$$

$$= \sum_{k\in\mathbb{Z}_{\geq 0}} \sum_{j=0}^{k} a_j b_{k-j} x^k \tag{A.82}$$

$$= \sum_{k\in\mathbb{Z}_{\geq 0}} c_k x^k. \tag{A.83}$$

Then $\bullet \cdot \bullet$ is called the *Cauchy product* for power series.

Theorem A.7.10 (Merten's Convergence Theorem)**.** *Let $f, g \in p_\infty(\mathcal{X})$ with coefficients $a_k, b_k \in \mathbb{K}$ for $k \in \mathbb{Z}_{\geq 0}$ respectively. Let $\sum_k a_k = A, \sum_k b_k = B$, and let either series converge absolutely, i.e.*

$$\sum_{k\in\mathbb{Z}_{\geq 0}} |a_k| < \infty, \quad or \quad \sum_{k\in\mathbb{Z}_{\geq 0}} |b_k| < \infty. \tag{A.84}$$

Then the Cauchy product $f \cdot g = h = \sum_{k\in\mathbb{Z}_{\geq 0}} c_k x^k$.

Proof. Denote the partial sum $A_n = \sum_{k\leq n} a_k$, with similar notation for B_n. Let $C_n = \sum_{k\leq n} a_k b_{n-k} = \sum_{k\leq n} c_k$. We need to show that the series $C_n \to AB$. Assume A_n as $n \to \infty$ converges absolutely. Then

$$C_n = \sum_{k\leq n} a_k B_{n-k} \tag{A.85}$$

$$AB = (A - A_n)B + \sum_{k\leq n} a_k B \tag{A.86}$$

$$AB - C_n = (A - A_n)B - \sum_{k\leq n} a_k(B_{n-k} - B) \tag{A.87}$$

Since A_n converges absolutely and B_n converges to B there exists some N such that for all $n \geq N$ we have

$$\sum_{k\leq N} |a_k|\ |B - B_k| \leq \left(\max_{N\leq k\leq n} |B - B_k|\right) \sum_{k\leq N} |a_k| \qquad \to 0 \tag{A.88}$$

$$\sum_{N<k\leq n} |a_k|\ |B - B_k| \leq N \sum_{N<k\leq n} |a_k| \qquad \to 0 \tag{A.89}$$

and as $A_n \to A$, the term $|A - A_n|B \to 0$. Then C_n converges to AB. ■

Theorem A.7.11 (Jensen's Inequality)**.** *Let $(\Omega, \mathcal{A}, P)$ be a probability space, let $\mathrm{X} : \Omega \to \mathbb{R}$ be a random variable, and let $g : \mathbb{R} \to \mathbb{R}$ be a convex function, such that for $0 < c < 1$ we have*

$$g(cx + (1 - c)y) \leq cg(x) + (1 - c)g(y). \tag{A.90}$$

Then

$$g(\mathbb{E}[\mathrm{X}]) \leq \mathbb{E}[g(\mathrm{X})]. \tag{A.91}$$

A proof for Theorem A.7.11 is derived in [Bil95, p.477].

Corollary A.7.12 (Orthonormal Series)**.** *Let $\mathcal{X}$ be an inner product space with orthonormal basis $\mathscr{B} = (v_n)_{n\in\mathcal{I}}$, and let $(c_n \in \mathbb{K})_{n\in\mathcal{I}}$ be some sequence. Then*

$$\|\sum_{n\in\mathcal{I}} c_n v_n\|^2 = \sum_{n\in\mathcal{I}} |c_n|^2. \tag{A.92}$$

Corollary A.7.13 (Parseval's Identity)**.** *Let $\mathscr{B}$ be an orthonormal basis for a Hilbert space $\mathcal{X}$. Then for all $x \in \mathcal{X}, v_n \in \mathscr{B}$ we have*

$$\|x\|^2 = \sum_{n\in\mathcal{I}} |\langle x, v_n\rangle|^2. \tag{A.93}$$

Both Corollaries A.7.12 and A.7.13 follow directly from Definitions A.2.10 and A.1.13.

Theorem A.7.14 (Heine-Borel)**.** *Let $\mathcal{X}$ be a metric space. Then $\mathcal{X}$ is compact (Definition A.1.12) if and only if it is complete (Definition A.1.10) and for every $\epsilon \in \mathbb{R}_{\geq 0}$ there exists finitely many points $x_1, \ldots, x_n$ such that*

$$\inf_{1 \leq i \leq m} d(x_i, x) \leq \epsilon \tag{A.94}$$

for all $x \in \mathcal{X}$. More generally, we say that a space is compact *if and only if it is* closed *and* bounded*.*

A proof for Theorem A.7.14 is provided in [MW99, pp.402–404] in the context of Euclidean n-dimensional space.

Definition A.7.15 (Location-Scale Family)**.** Let $\mathrm{X} : \Omega_x \to \mathbb{R}^d$ be a random variable. Let $\mathrm{Z} : \Omega_z \to \mathbb{R}^d$ be another random variable such that given $s, m \in \mathbb{R}^d$ we have that

(i) if $F_{\mathrm{X}}(x)$ is a cdf. for X, then $F_{\mathrm{Z}}(z) = F_{\mathrm{X}}(s^{\text{-}1}(z - m))$,

(ii) if $f_{\mathrm{X}}(x)$ is a pdf. for X, then $f_{\mathrm{Z}}(z) = f_{\mathrm{X}}(s^{\text{-}1}(z - m))$,

(iii) if $f_{\mathrm{X}}(x)$ is a pmf. for X, then $f_{\mathrm{Z}}(z) = s^{\text{-}1} f_{\mathrm{X}}(s^{\text{-}1}(z - m))$.

Then X, Z belong to a family of probability distributions $\mathcal{P}$, called a *location-scale family.*

Definition A.7.16 (Mutual Information)**.** Let X, Y be random variables with pdfs. $f_{\mathrm{X}}, f_{\mathrm{Y}}$ respectably. The *mutual information* of X, Y given $f_{\mathrm{X}}, f_{\mathrm{Y}}$ is then given by

$$\begin{aligned} \mathrm{I}[f_{\mathrm{X}}, f_{\mathrm{Y}}] &= \mathrm{H}[f_{\mathrm{X}}] - \mathrm{H}[f_{\mathrm{X}|\mathrm{Y}}] \\ &= \mathrm{H}[f_{\mathrm{Y}}] - \mathrm{H}[f_{\mathrm{Y}|\mathrm{X}}] \end{aligned}$$

Definition A.7.17 (Evidence Lower Bound)**.** Let X, Z be random variables with pdfs. $f_{\mathrm{X}}, f_{\mathrm{Z}}$ respectably. Let g_{Z} be an approximate probability density over Z. The *evidence lower bound* of X given f_{X} is then given by

$$\textsc{elbo}[f_{\mathrm{X}}] = \mathrm{H}[g_{\mathrm{Z}}] - \mathrm{H}_g[f_{\mathrm{XZ}}].$$

Definition A.7.18 (Augmented Affine Transformation)**.** Let $y = Ax + b$ be a linear affine transformation with $x\mathbb{R}^n$ and $y, b \in \mathbb{R}^m$. Let $\tilde{A}\tilde{x}$ denote an augmentation such that

$$\begin{aligned} &\tilde{y} = \tilde{A}\tilde{x} \\ &\begin{bmatrix} y_1 & \ldots & y_m \\ 1 & \ldots & 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & A & & b \\ 0 & \ldots & 0 & 1 \end{array}\right] \begin{bmatrix} x_1 & \ldots & x_m \\ 1 & \ldots & 1 \end{bmatrix}. \end{aligned}$$

Then we say the transformation $\tilde{A}\tilde{x}$ is an *augmented affine transformation.*

Definition A.7.19 (Convex Sets and Functions)**.** Let $\mathcal{X}$ be a linear space and let $A \subset \mathcal{X}$ such that for all $x, y \in \mathcal{X}$ and for all $t \in [0, 1]$ we have $tx + (1 - t)y \in A$. Then A is called a *convex set.* Subsequently, we call any $f : A \to \mathcal{X}$ with the property that $f(tx + (1 - t)y) = tf(x) + (1 - t)f(y)$ a *convex function.*

A.8 Discrete Convolutions as Matrices

In Section 2.2 we showed that convolutions are Hilbert-Schmidt operators, but it might not be immediately clear how convolutions can be constructed as rudimentary matrix operations. We assume a circular full rank convolution operator. Given an input signal $x \in \mathbb{K}^n$, output signal $y \in \mathbb{K}^n$, and a kernel parametrized by $\theta \in \mathbb{K}^q$, the general form of discrete circular one-dimensional convolution is given by

$$y_i = \sum_{j=1}^{q} x_j \theta_{c_j} : c_j = (i-j)_{\operatorname{mod} q} + 1 \tag{A.95}$$

which can be expressed by a *circulant Toeplitz matrix* on the form

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \theta_1 & 0 & \dots & \theta_2 \\ \theta_2 & \theta_1 & \dots & \theta_3 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \theta_2 & \theta_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{A.96}$$

Now let $\mathbf{0}_{r,c}$ be a $r \times c$ zero matrix, allowing for zero dimensional matrices given either $r = 0$ or $c = 0$ for notational simplicity. If for $i = 1, \dots, n$ and $A \in \mathbb{K}^{r \times s}$ we now define the circular roll operator

$$\tau_i(A; n) = \begin{cases} \begin{bmatrix} \mathbf{0}_{i\text{-}1,s} \\ A \\ \mathbf{0}_{n\text{-}r\text{-}i,s} \end{bmatrix} & \text{if } n - r - i \geq 0, \\ \begin{bmatrix} A_{i\text{-}n+2:r} \\ \mathbf{0}_{n\text{-}r,s} \\ A_{1:i\text{-}n+1} \end{bmatrix} & \text{otherwise,} \end{cases} \tag{A.97}$$

then we can instead express the convolution matrix concisely as

$$W_\theta = [\tau_1(\theta; n), \tau_2(\theta; n), \dots \tau_n(\theta; n)] \in \mathbb{K}^{n \times n}. \tag{A.98}$$

This result can readily be extended to higher dimensional convolution operators. For two dimensions, we consider a kernel $\theta \in \mathbb{K}^{p\times q}$ and a lexicographical flattening of a two-dimensional input signal $x \in \mathbb{K}^{m\times n}$ and output signal $y \in \mathbb{K}^{m\times n}$ such that for $x_1^\intercal, \dots, x_r^\intercal \in \mathbb{K}^n$ we have

$$\operatorname{vec}(x) = \begin{bmatrix} x_1^\intercal \\ \vdots \\ x_n^\intercal \end{bmatrix} \in \mathbb{K}^{mn}, \tag{A.99}$$

and a similar flattening can be performed to generate $\operatorname{vec}(y)$. Then for $i = 1, \dots, p$ we can define one dimensional convolution operators W_{θ_i} for each row θ_i by Equation (A.98), which in turn can be concatenated into a block matrix $V_\theta^\intercal = [W_{\theta_1}, W_{\theta_2}, \dots, W_{\theta_p}]^\intercal$ where each block corresponds to one row of the kernel θ. Then a two-dimensional convolution operator is given by

$$W_\theta = [\tau_1(V_\theta; m), \tau_2(V_\theta; m), \dots, \tau_m(V_\theta; m)], \tag{A.100}$$

which can be expressed in block matrix form

$$W_\theta = \begin{bmatrix} W_{\theta_1} & 0 & \dots & W_{\theta_2} \\ W_{\theta_2} & W_{\theta_1} & \dots & W_{\theta_3} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & W_{\theta_2} & W_{\theta_1} \end{bmatrix}. \tag{A.101}$$

In Definition 4.3.2 we mentioned that convolutional layers can be represented as linear combinations of convolution operators. For multichannel inputs, separate convolutional filters are applied to each channel and the output is combined via linear combinations. Channels are thus often referred to as *features* in convolutional networks. For a multi-channel convolutional layer, we apply kernel parametrizations on the form $\theta \in \mathbb{K}^{c\times d\times p\times q}$ yielding operators of the form

$$W_\theta = \begin{bmatrix} W_{\theta_{1,1}} & W_{\theta_{1,2}} & \dots & W_{\theta_{1,d}} \\ W_{\theta_{2,1}} & W_{\theta_{2,2}} & \dots & W_{\theta_{2,d}} \\ \vdots & \ddots & \dots & \vdots \\ W_{\theta_{c,1}} & W_{\theta_{c,2}} & \dots & W_{\theta_{c,d}} \end{bmatrix}, \tag{A.102}$$

which can then be applied to appropriately lexicographically vectorized two-dimensional multi-channel signals $\operatorname{vec}(x)$.

Bibliography

[Abe81] Abel, N. H. *Œuvres complètes de Niels Henrik Abel. Tome II.* Contenant les mémoirs posthumes d'Abel. [Containing the posthumous memoirs of Abel], Edited and with notes by L. Sylow and S. Lie. Imprimerie de Grøndahl & Son, Christiania; distributed by the Norwegian Mathematical Society, Oslo, 1981, pp. vi+341 (cit. on p. 11).

[ACH21] Antun, V., Colbrook, M. J. and Hansen, A. C. 'Can stable and accurate neural networks be computed? - On the barriers of deep learning and Smale's 18th problem'. In: *CoRR* vol. abs/2101.08286 (2021). arXiv: 2101.08286 (cit. on pp. 73, 74).

[AEE17] Almotiri, J., Elleithy, K. and Elleithy, A. 'Comparison of autoencoder and Principal Component Analysis followed by neural network for e-learning using handwritten recognition'. In: *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT).* 2017, pp. 1–5 (cit. on p. 61).

[Agr15] Agresti, A. *Foundations of linear and generalized linear models.* Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, 2015, pp. xiv+444 (cit. on p. 39).

[AH17] Aghdam, H. H. and Heravi, E. J. *Guide to convolutional neural networks.* Vol. 10. 978-973. Springer, 2017, p. 51 (cit. on p. 46).

[ALG18] Anil, C., Lucas, J. and Grosse, R. B. 'Sorting out Lipschitz function approximation'. In: *CoRR* vol. abs/1811.05381 (2018). arXiv: 1811.05381 (cit. on p. 67).

[AMS08] Absil, P.-A., Mahony, R. and Sepulchre, R. *Optimization algorithms on matrix manifolds.* With a foreword by Paul Van Dooren. Princeton University Press, Princeton, NJ, 2008, pp. xvi+224 (cit. on pp. 4, 90, 94–96).

[Ant+20] Antun, V. et al. 'On instabilities of deep learning in image reconstruction and the potential costs of AI'. In: *Proceedings of the National Academy of Sciences* vol. 117, no. 48 (2020), pp. 30088–30095. eprint: https://www.pnas.org/content/117/48/30088.full.pdf (cit. on pp. 72, 107).

[Ard+18] Ardizzone, L. et al. ‘Analyzing inverse problems with invertible neural networks’. In: *arXiv preprint arXiv:1808.04730* (2018) (cit. on pp. 1, 75, 76, 121, 124, 128).

[AS80] Aitchison, J. and Shen, S. M. ‘Logistic-Normal Distributions: Some Properties and Uses’. In: *Biometrika* vol. 67, no. 2 (1980), pp. 261–272 (cit. on p. 85).

[ASB16] Arjovsky, M., Shah, A. and Bengio, Y. ‘Unitary Evolution Recurrent Neural Networks’. In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016.* Ed. by Balcan, M.-F. and Weinberger, K. Q. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 1120–1128 (cit. on p. 89).

[BAK18] Bzdok, D., Altman, N. and Krzywinski, M. ‘Statistics versus machine learning’. In: *Nature Methods* vol. 15, no. 4 (Apr. 2018), pp. 233–234 (cit. on p. 32).

[Bar17] Barron, J. T. ‘Continuously Differentiable Exponential Linear Units’. In: *CoRR* vol. abs/1704.07483 (2017). arXiv: 1704.07483 (cit. on p. 47).

[BBC19] Bader, P., Blanes, S. and Casas, F. ‘Computing the Matrix Exponential with an Optimized Taylor Polynomial Approximation’. In: *Mathematics* vol. 7, no. 12 (2019) (cit. on p. 97).

[BCB15] Bahdanau, D., Cho, K. and Bengio, Y. ‘Neural Machine Translation by Jointly Learning to Align and Translate’. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* Ed. by Bengio, Y. and LeCun, Y. 2015 (cit. on p. 77).

[BDV01] Bengio, Y., Ducharme, R. and Vincent, P. ‘A Neural Probabilistic Language Model’. In: *Advances in Neural Information Processing Systems.* Ed. by Leen, T., Dietterich, T. and Tresp, V. Vol. 13. MIT Press, 2001 (cit. on p. 77).

[Ben+07] Bengio, Y. et al. ‘Greedy layer-wise training of deep networks’. In: vol. 19. Jan. 2007 (cit. on p. 68).

[Ben08] Ben-Gal, I. ‘Bayesian Networks’. In: *Encyclopedia of Statistics in Quality and Reliability.* American Cancer Society, 2008. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470061572.eqr089 (cit. on p. 34).

[Beu65] Beutler, F. J. ‘The operator theory of the pseudo-inverse I. Bounded operators’. In: *Journal of Mathematical Analysis and Applications* vol. 10, no. 3 (1965), pp. 451–470 (cit. on p. 18).

[BFS93] Bengio, Y., Frasconi, P. and Simard, P. ‘The problem of learning long-term dependencies in recurrent networks’. In: *IEEE International Conference on Neural Networks.* 1993, 1183–1188 vol.3 (cit. on p. 89).

[BH89] Baldi, P. and Hornik, K. ‘Neural networks and principal component analysis: Learning from examples without local minima’. In: *Neural Networks* vol. 2, no. 1 (1989), pp. 53–58 (cit. on pp. 61, 78, 119).

[Bil95] Billingsley, P. *Probability and measure.* Third. Wiley Series in Probability and Mathematical Statistics. A Wiley-Interscience Publication. John Wiley & Sons, Inc., New York, 1995, pp. xiv+593 (cit. on pp. 4, 27, 30, 151, 161, 173).

[BK19] Brunton, S. L. and Kutz, J. N. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control.* 1st. USA: Cambridge University Press, 2019 (cit. on p. 38).

[Blo10] Blockeel, H. 'Hypothesis Space'. In: *Encyclopedia of Machine Learning.* Ed. by Sammut, C. and Webb, G. I. Boston, MA: Springer US, 2010, pp. 511–513 (cit. on p. 33).

[Blu+15] Blundell, C. et al. *Weight Uncertainty in Neural Networks.* 2015. arXiv: 1505.05424 [stat.ML] (cit. on p. 74).

[Bol12] Boltzmann, L. *Wissenschaftliche Abhandlungen.* Ed. by Hasenöhrl, F. Vol. 1. Cambridge Library Collection - Physical Sciences. Cambridge University Press, 2012 (cit. on p. 47).

[Bot91] Bottou, L. 'Stochastic Gradient Learning in Neural Networks'. In: 1991 (cit. on p. 72).

[Bre+84] Breiman, L. et al. *Classification and Regression Trees.* Taylor & Francis, 1984 (cit. on p. 32).

[BSF94] Bengio, Y., Simard, P. and Frasconi, P. 'Learning long-term dependencies with gradient descent is difficult'. In: *IEEE Transactions on Neural Networks* vol. 5, no. 2 (1994), pp. 157–166 (cit. on p. 68).

[Cas19] Casado, M. L. 'Trivializations for Gradient-Based Optimization on Manifolds'. In: *CoRR* vol. abs/1909.09501 (2019). arXiv: 1909.09501 (cit. on pp. 4, 90, 94, 96).

[Cho17] Chollet, F. 'Xception: Deep Learning with Depthwise Separable Convolutions'. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2017, pp. 1800–1807 (cit. on p. 57).

[Chr10] Christensen, O. *Functions, spaces, and expansions.* Applied and Numerical Harmonic Analysis. Mathematical tools in physics and engineering. Birkhäuser Boston, Ltd., Boston, MA, 2010, pp. xx+263 (cit. on p. 160).

[Cir+11] Ciresan, D. et al. 'Flexible, High Performance Convolutional Neural Networks for Image Classification.' In: July 2011, pp. 1237–1242 (cit. on p. 56).

[Cis+17] Cisse, M. et al. *Parseval Networks: Improving Robustness to Adversarial Examples.* 2017. arXiv: 1704.08847 [stat.ML] (cit. on pp. 67, 78).

[CKP13] Casazza, P. G., Kutyniok, G. and Philipp, F. *Introduction to Finite Frame Theory.* Ed. by Casazza, P. G. and Kutyniok, G. Boston: Birkhäuser Boston, 2013 (cit. on p. 147).

[Coh+17] Cohen, G. et al. 'EMNIST: an extension of MNIST to handwritten letters'. In: *CoRR* vol. abs/1702.05373 (2017). arXiv: 1702.05373 (cit. on pp. 6, 104).

[Com+18] Combes, R. T. des et al. 'On the Learning Dynamics of Deep Neural Networks'. In: *CoRR* vol. abs/1809.06848 (2018). arXiv: 1809.06848 (cit. on p. 72).

[Con90] Conway, J. B. *A course in functional analysis*. Second. Vol. 96. Graduate Texts in Mathematics. Springer-Verlag, New York, 1990, pp. xvi+399 (cit. on p. 11).

[Cox04] Cox, D. D. 'The Theory of Statistics and Its Applications'. 2004 (cit. on p. 27).

[CUH16] Clevert, D.-A., Unterthiner, T. and Hochreiter, S. 'Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)'. In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Bengio, Y. and LeCun, Y. 2016 (cit. on p. 47).

[Cyb89] Cybenko, G. 'Approximation by superpositions of a sigmoidal function'. In: *Mathematics of Control, Signals, and Systems (MCSS)* vol. 2, no. 4 (Dec. 1989), pp. 303–314 (cit. on p. 50).

[DD09] Der Kiureghian, A. and Ditlevsen, O. 'Aleatory or epistemic? Does it matter?' In: *Structural safety* vol. 31, no. 2 (2009), pp. 105–112 (cit. on p. 71).

[DFO20] Deisenroth, M. P., Faisal, A. A. and Ong, C. S. *Mathematics for Machine Learning*. Cambridge University Press, 2020 (cit. on p. 34).

[DG17] Dua, D. and Graff, C. *UCI Machine Learning Repository*. 2017 (cit. on p. 73).

[DHS11] Duchi, J., Hazan, E. and Singer, Y. 'Adaptive Subgradient Methods for Online Learning and Stochastic Optimization'. In: *J. Mach. Learn. Res.* vol. 12, no. null (July 2011), pp. 2121–2159 (cit. on p. 53).

[DKB15] Dinh, L., Krueger, D. and Bengio, Y. 'NICE: Non-linear Independent Components Estimation'. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*. Ed. by Bengio, Y. and LeCun, Y. 2015 (cit. on pp. 57, 74).

[DSB16] Dinh, L., Sohl-Dickstein, J. and Bengio, S. 'Density estimation using Real NVP'. In: *CoRR* vol. abs/1605.08803 (2016). arXiv: 1605.08803 (cit. on pp. 74, 75).

[Erh+10] Erhan, D. et al. 'Why Does Unsupervised Pre-Training Help Deep Learning?' In: *J. Mach. Learn. Res.* vol. 11 (Mar. 2010), pp. 625–660 (cit. on p. 68).

[Faz+19] Fazlyab, M. et al. 'Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks'. In: *CoRR* vol. abs/1906.04893 (2019). arXiv: 1906.04893 (cit. on p. 147).

[FB19] Frerix, T. and Bruna, J. 'Approximating Orthogonal Matrices with Effective Givens Factorization'. In: *Proceedings of the 36th International Conference on Machine Learning.* Ed. by Chaudhuri, K. and Salakhutdinov, R. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 1993–2001 (cit. on p. 89).

[Fer73] Ferguson, T. S. 'A Bayesian Analysis of Some Nonparametric Problems'. In: *The Annals of Statistics* vol. 1, no. 2 (1973), pp. 209–230 (cit. on p. 85).

[FH89] Fix, E. and Hodges, J. L. 'Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties'. In: *International Statistical Review / Revue Internationale de Statistique* vol. 57, no. 3 (1989), pp. 238–247 (cit. on p. 32).

[Fre03] Fredholm, I. 'Sur une classe d'équations fonctionnelles'. In: *Acta Math.* vol. 27, no. 1 (1903), pp. 365–390 (cit. on pp. 11, 21).

[Fri98] Friedman, J. H. 'Data Mining and Statistics: What's the connection?' In: *Computing science and statistics* vol. 29, no. 1 (1998), pp. 3–9 (cit. on p. 32).

[Fuk79] Fukushima, K. 'Self-Organization of a Neural Network which Gives Position-Invariant Response'. In: *Proceedings of the Sixth International Joint Conference on Artificial Intelligence, IJCAI 79, Tokyo, Japan, August 20-23, 1979, 2 Volumes.* Ed. by Buchanan, B. G. William Kaufmann, 1979, pp. 291–293 (cit. on pp. 1, 55).

[Gao+15] Gao, S. et al. 'Single sample face recognition via learning deep supervised autoencoders'. In: *IEEE transactions on information forensics and security* vol. 10, no. 10 (2015), pp. 2108–2118 (cit. on p. 62).

[GB10] Glorot, X. and Bengio, Y. 'Understanding the difficulty of training deep feedforward neural networks'. In: *AISTATS.* 2010 (cit. on p. 51).

[GBC16] Goodfellow, I., Bengio, Y. and Courville, A. *Deep learning.* Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2016, pp. xxii+775 (cit. on pp. 18, 39, 55, 56, 68).

[GDB19] Gagnon, P., Desgagné, A. and Bédard, M. 'A New Bayesian Approach to Robustness Against Outliers in Linear Regression'. In: *Bayesian Analysis* (May 2019) (cit. on p. 34).

[Gel+14] Gelman, A. et al. *Bayesian data analysis.* Third. Texts in Statistical Science Series. CRC Press, Boca Raton, FL, 2014, pp. xiv+661 (cit. on pp. 32, 34, 42, 85).

[GG16] Gal, Y. and Ghahramani, Z. 'Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning'. In: *Proceedings of The 33rd International Conference on Machine Learning.* Ed. by Balcan, M. F. and Weinberger, K. Q. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, June 2016, pp. 1050–1059 (cit. on pp. 66, 74).

[GH13] Givens, G. H. and Hoeting, J. A. *Computational statistics.* Second. Wiley Series in Computational Statistics. John Wiley & Sons, Inc., Hoboken, NJ, 2013, pp. xviii+469 (cit. on pp. 29, 34).

[Goo+14] Goodfellow, I. J. et al. *Generative Adversarial Networks.* 2014. arXiv: 1406.2661 [stat.ML] (cit. on p. 74).

[Gro93] Groetsch, C. *Inverse Problems in the Mathematical Sciences.* Jan. 1993 (cit. on p. 9).

[GSS15] Goodfellow, I. J., Shlens, J. and Szegedy, C. *Explaining and Harnessing Adversarial Examples.* 2015. arXiv: 1412.6572 [stat.ML] (cit. on p. 72).

[Had48] Hadamard, J. 'Sur le cas anormal du problème de Cauchy pour l'équation des ondes'. In: *Studies and Essays Presented to R.Courant on his 60th Birthday.* Interscience Publishers, Inc., New York, 1948, pp. 161–165 (cit. on p. 10).

[Hah+00] Hahnloser, R. H. R. et al. 'Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit'. In: *Nature* vol. 405, no. 6789 (June 2000), pp. 947–951 (cit. on pp. 47, 109).

[Hal15] Hall, B. *Lie groups, Lie algebras, and representations.* Second. Vol. 222. Graduate Texts in Mathematics. An elementary introduction. Springer, Cham, 2015, pp. xiv+449 (cit. on pp. 4, 90–92, 151, 165, 167–169).

[Han10] Hansen, P. C. *Discrete inverse problems.* Vol. 7. Fundamentals of Algorithms. Insight and algorithms. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2010, pp. xii+213 (cit. on p. 9).

[He+15] He, K. et al. 'Deep Residual Learning for Image Recognition'. In: *CoRR* vol. abs/1512.03385 (2015). arXiv: 1512.03385 (cit. on pp. 51, 54, 85, 106, 118).

[He+19] He, K. et al. 'Momentum Contrast for Unsupervised Visual Representation Learning'. In: *CoRR* vol. abs/1911.05722 (2019). arXiv: 1911.05722 (cit. on p. 148).

[Hen+11] Hennig, P. et al. 'Kernel Topic Models'. In: *CoRR* vol. abs/1110.4713 (2011). arXiv: 1110.4713 (cit. on p. 85).

[HG08] Huynh-Thu, Q. and Ghanbari, M. 'Scope of validity of PSNR in image/video quality assessment'. In: *Electronics letters* vol. 44, no. 13 (2008), pp. 800–801 (cit. on p. 107).

[Hin+12] Hinton, G. E. et al. 'Improving neural networks by preventing co-adaptation of feature detectors'. In: *CoRR* vol. abs/1207.0580 (2012). arXiv: 1207.0580 (cit. on p. 66).

[Hin17] Hinton, G. *What is wrong with convolutional neural nets?* Aug. 2017 (cit. on p. 56).

[HNO06] Hansen, P. C., Nagy, J. G. and O'Leary, D. P. *Deblurring images.* Vol. 3. Fundamentals of Algorithms. Matrices, spectra, and filtering. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006, pp. xiv+130 (cit. on pp. 9, 20).

[Hoc91] Hochreiter, S. 'Untersuchungen zu dynamischen neuronalen Netzen'. In: Institut für Informatik - TU München, 1991 (cit. on p. 53).

[Hor91] Hornik, K. ‘Approximation capabilities of multilayer feedforward networks’. In: *Neural Networks* vol. 4, no. 2 (1991), pp. 251–257 (cit. on p. 50).

[How+17] Howard, A. G. et al. ‘MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications’. In: *CoRR* vol. abs/1704.04861 (2017). arXiv: 1704.04861 (cit. on p. 57).

[HR17] Hyland, S. and Rätsch, G. *Learning Unitary Operators with Help From u(n)*. 2017 (cit. on p. 90).

[HTF09] Hastie, T., Tibshirani, R. and Friedman, J. *The elements of statistical learning*. Second. Springer Series in Statistics. Data mining, inference, and prediction. Springer, New York, 2009, pp. xxii+745 (cit. on pp. 32, 39, 40, 74).

[HW21] Hüllermeier, E. and Waegeman, W. ‘Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods’. In: *Machine Learning* vol. 110, no. 3 (Mar. 2021), pp. 457–506 (cit. on pp. 33, 71).

[IS15] Ioffe, S. and Szegedy, C. ‘Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift’. In: *CoRR* vol. abs/1502.03167 (2015). arXiv: 1502.03167 (cit. on p. 66).

[Jeb03] Jebara, T. *Machine Learning: Discriminative and Generative (Kluwer International Series in Engineering and Computer Science)*. USA: Kluwer Academic Publishers, 2003 (cit. on pp. 33, 34, 37).

[JHG00] Japkowicz, N., Hanson, S. and Gluck, M. ‘Nonlinear Autoassociation Is Not Equivalent to PCA’. In: *Neural Computation* vol. 12 (Mar. 2000), pp. 531–545 (cit. on p. 61).

[Joo+19] Joo, W. et al. ‘Dirichlet Variational Autoencoder’. In: *CoRR* vol. abs/1901.02739 (2019). arXiv: 1901.02739 (cit. on p. 85).

[JSO18] Jacobsen, J.-H., Smeulders, A. W. and Oyallon, E. ‘i-RevNet: Deep Invertible Networks’. In: *International Conference on Learning Representations*. 2018 (cit. on pp. 75, 98).

[Kal02] Kallenberg, O. *Foundations of modern probability*. Second. Probability and its Applications (New York). Springer-Verlag, New York, 2002, pp. xx+638 (cit. on p. 27).

[KB17] Kingma, D. P. and Ba, J. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: `1412.6980 [cs.LG]` (cit. on pp. 53, 66, 114).

[KC08] Kovacevic, J. and Chebira, A. ‘An Introduction to Frames’. In: *Foundations and Trends in Signal Processing* vol. 2 (Feb. 2008), pp. 1–94 (cit. on p. 67).

[KD18] Kingma, D. P. and Dhariwal, P. *Glow: Generative Flow with Invertible 1x1 Convolutions*. 2018. arXiv: `1807.03039 [stat.ML]` (cit. on p. 74).

[KF61] Kolmogorov, A. N. and Fomin, S. V. *Elements of the theory of functions and functional analysis. Vol. 2: Measure. The Lebesgue integral. Hilbert space.* Translated from the first (1960) Russian ed. by Hyman Kamel and Horace Komm. Graylock Press, Albany, N.Y., 1961, pp. ix+128 (cit. on p. 171).

[KG17] Kendall, A. and Gal, Y. 'What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?' In: *Proceedings of the 31st International Conference on Neural Information Processing Systems.* NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 5580–5590 (cit. on pp. 71, 74).

[KH09] Krizhevsky, A. and Hinton, G. 'Learning multiple layers of features from tiny images'. In: *Master's thesis, Department of Computer Science, University of Toronto* (2009) (cit. on pp. 6, 13, 103).

[KL20] Kidger, P. and Lyons, T. 'Universal Approximation with Deep Narrow Networks'. In: *Proceedings of Thirty Third Conference on Learning Theory.* Ed. by Abernethy, J. and Agarwal, S. Vol. 125. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 2306–2327 (cit. on pp. 46, 50).

[KL59] Kimball, A. W. and Leach, E. 'Approximate Linearization of the Incomplete Beta-Function'. In: *Biometrika* vol. 46, no. 1/2 (1959), pp. 214–218 (cit. on p. 87).

[Kno+20] Knoll, F. et al. 'Advancing machine learning for MR image reconstruction with an open competition: Overview of the 2019 fastMRI challenge'. In: *Magnetic Resonance in Medicine* vol. 84, no. 6 (June 2020), pp. 3054–3070 (cit. on p. 77).

[KO01] Kennedy, M. C. and O'Hagan, A. 'Bayesian calibration of computer models'. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* vol. 63, no. 3 (2001), pp. 425–464. eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/1467-9868.00294 (cit. on p. 72).

[Kol02] Kolbjornsen, O. *Nonlinear topics in the Bayesian approach to inverse problems with applications to seismic inversion.* Thesis (Dr.Ing.)–Norges teknisk-naturvitenskapelige universitet (Norway). ProQuest LLC, Ann Arbor, MI, 2002, p. 166 (cit. on p. 9).

[Kra64] Krasnosel'skii, M. A. 'Topological Methods in the Theory of Nonlinear Integral Equations. (International Series of Monographs on Pure and Applied Mathematics, Vol. 45) X + 395 S. Oxford/London/New York/Paris 1964. Pergamon Press. Preis geb. 70 s. net .' Trans. by Müller, P. H. In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* vol. 44, no. 10-11 (1964), pp. 521–521. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/zamm.19640441041 (cit. on p. 98).

[KSH12] Krizhevsky, A., Sutskever, I. and Hinton, G. 'ImageNet Classification with Deep Convolutional Neural Networks'. In: *Neural Information Processing Systems* vol. 25 (Jan. 2012) (cit. on p. 47).

[KSH17] Krizhevsky, A., Sutskever, I. and Hinton, G. E. 'ImageNet Classification with Deep Convolutional Neural Networks'. In: *Commun. ACM* vol. 60, no. 6 (May 2017), pp. 84–90 (cit. on p. 85).

[Kum80] Kumaraswamy, P. 'A generalized probability density function for double-bounded random processes'. In: *Journal of Hydrology* vol. 46, no. 1 (1980), pp. 79–88 (cit. on p. 86).

[Kun+19] Kunin, D. et al. 'Loss Landscapes of Regularized Linear Autoencoders'. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Chaudhuri, K. and Salakhutdinov, R. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 3560–3569 (cit. on pp. 61, 119).

[KW14] Kingma, D. P. and Welling, M. *Auto-Encoding Variational Bayes*. 2014. arXiv: `1312.6114 [stat.ML]` (cit. on pp. 59, 64, 65).

[Lar+16] Larsen, A. B. L. et al. *Autoencoding beyond pixels using a learned similarity metric*. 2016. arXiv: `1512.09300 [cs.LG]` (cit. on p. 74).

[LeC+89] LeCun, Y. et al. 'Backpropagation Applied to Handwritten Zip Code Recognition'. In: *Neural Comput.* vol. 1, no. 4 (1989), pp. 541–551 (cit. on pp. 1, 55).

[LH17] Loshchilov, I. and Hutter, F. 'Fixing Weight Decay Regularization in Adam'. In: *CoRR* vol. abs/1711.05101 (2017). arXiv: 1711.05101 (cit. on p. 66).

[Lin+14] Lin, T.-Y. et al. 'Microsoft COCO: Common Objects in Context'. In: *CoRR* vol. abs/1405.0312 (2014). arXiv: 1405.0312 (cit. on pp. 6, 104).

[Lin17] Lindstrøm, T. L. *Spaces—an introduction to real analysis*. Vol. 29. Pure and Applied Undergraduate Texts. American Mathematical Society, Providence, RI, 2017, pp. xii+369 (cit. on pp. 4, 151, 158).

[LPW18] Le, L., Patterson, A. and White, M. 'Supervised autoencoders: Improving generalization performance with unsupervised regularizers'. In: *Advances in neural information processing systems* vol. 31 (2018), pp. 107–117 (cit. on pp. 62, 68, 78).

[Lu+20] Lu, L. et al. 'Dying ReLU and Initialization: Theory and Numerical Examples'. In: *Communications in Computational Physics* vol. 28, no. 5 (June 2020), pp. 1671–1706 (cit. on p. 53).

[Luo+16] Luo, W. et al. 'Understanding the Effective Receptive Field in Deep Convolutional Neural Networks'. In: *Advances in Neural Information Processing Systems 29*. Ed. by Lee, D. D. et al. Curran Associates, Inc., 2016, pp. 4898–4906 (cit. on p. 55).

[Mac95] MacKay, D. J. 'Bayesian neural networks and density networks'. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* vol. 354, no. 1 (1995). Proceedings of the Third Workshop on Neutron Scattering Data Analysis, pp. 73–80 (cit. on p. 74).

[Mac98] MacKay, D. J. ‘Choice of Basis for Laplace Approximation’. In: *Machine Learning* vol. 33, no. 1 (Oct. 1998), pp. 77–86 (cit. on p. 85).

[Mar67] Marlow, N. A. ‘A normal limit theorem for power sums of independent random variables’. In: *The Bell System Technical Journal* vol. 46, no. 9 (1967), pp. 2081–2089 (cit. on p. 86).

[McC02] McCullagh, P. ‘What is a statistical model?’ In: *Ann. Statist.* vol. 30, no. 5 (2002). With comments and a rejoinder by the author, pp. 1225–1310 (cit. on p. 28).

[Mha+16] Mhammedi, Z. et al. ‘Efficient Orthogonal Parametrisation of Recurrent Neural Networks Using Householder Reflections’. In: *CoRR* vol. abs/1612.00188 (2016). arXiv: 1612.00188 (cit. on p. 90).

[MHN13] Maas, A. L., Hannun, A. Y. and Ng, A. Y. ‘Rectifier nonlinearities improve neural network acoustic models’. In: *Proc. icml*. Vol. 30. 1. Citeseer. 2013, p. 3 (cit. on p. 81).

[MMD20] Mouton, C., Myburgh, J. C. and Davel, M. H. ‘Stride and Translation Invariance in CNNs’. In: *Artificial Intelligence Research*. Ed. by Gerber, A. Cham: Springer International Publishing, 2020, pp. 267–281 (cit. on p. 55).

[MP43] McCulloch, W. S. and Pitts, W. ‘A logical calculus of the ideas immanent in nervous activity’. In: *The bulletin of mathematical biophysics* vol. 5, no. 4 (Dec. 1943), pp. 115–133 (cit. on p. 1).

[MW99] McDonald, J. N. and Weiss, N. A. *A course in real analysis*. Biographies by Carol A. Weiss. Academic Press, Inc., San Diego, CA, 1999, pp. xx+745 (cit. on pp. 4, 12, 50, 151, 154, 158–160, 163, 170, 171, 174).

[Nak+19] Nakkiran, P. et al. ‘Deep Double Descent: Where Bigger Models and More Data Hurt’. In: *CoRR* vol. abs/1912.02292 (2019). arXiv: 1912.02292 (cit. on p. 73).

[NJ02] Ng, A. and Jordan, M. ‘On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes’. In: *Advances in Neural Information Processing Systems*. Ed. by Dietterich, T., Becker, S. and Ghahramani, Z. Vol. 14. MIT Press, 2002 (cit. on p. 34).

[NS17] Nalisnick, E. and Smyth, P. *Stick-Breaking Variational Autoencoders*. 2017. arXiv: 1605.06197 [stat.ML] (cit. on pp. 85, 86).

[NVF20] NVIDIA, Vingelmann, P. and Fitzek, F. H. *CUDA, release: 10.2.89*. 2020 (cit. on p. 3).

[NW72] Nelder, J. A. and Wedderburn, R. W. M. ‘Generalized Linear Models’. In: *Journal of the Royal Statistical Society. Series A (General)* vol. 135, no. 3 (1972), pp. 370–384 (cit. on p. 38).

[ODO16] Odena, A., Dumoulin, V. and Olah, C. ‘Deconvolution and Checkerboard Artifacts’. In: *Distill* (2016) (cit. on p. 55).

[OKK16] Oord, A. van den, Kalchbrenner, N. and Kavukcuoglu, K. 'Pixel Recurrent Neural Networks'. In: *CoRR* vol. abs/1601.06759 (2016). arXiv: 1601.06759 (cit. on p. 58).

[Oor+16] Oord, A. van den et al. 'WaveNet: A Generative Model for Raw Audio'. In: *CoRR* vol. abs/1609.03499 (2016). arXiv: 1609.03499 (cit. on p. 58).

[OWB18] Olson, M., Wyner, A. and Berk, R. 'Modern Neural Networks Generalize on Small Data Sets'. In: *Advances in Neural Information Processing Systems*. Ed. by Bengio, S. et al. Vol. 31. Curran Associates, Inc., 2018 (cit. on p. 73).

[Pap+21] Papamakarios, G. et al. *Normalizing Flows for Probabilistic Modeling and Inference.* 2021. arXiv: 1912.02762 [stat.ML] (cit. on p. 58).

[Pas+17] Paszke, A. et al. 'Automatic Differentiation in PyTorch'. In: *NIPS 2017 Workshop on Autodiff*. Long Beach, California, USA, 2017 (cit. on p. 53).

[Pas+19] Paszke, A. et al. 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. In: *Advances in Neural Information Processing Systems 32*. Ed. by Wallach, H. et al. Curran Associates, Inc., 2019, pp. 8024–8035 (cit. on p. 2).

[Pen55] Penrose, R. 'A generalized inverse for matrices'. In: *Mathematical Proceedings of the Cambridge Philosophical Society* vol. 51, no. 3 (1955), pp. 406–413 (cit. on p. 18).

[Pin99] Pinkus, A. 'Approximation theory of the MLP model in neural networks'. In: *Acta Numerica* vol. 8 (1999), pp. 143–195 (cit. on p. 50).

[Pog+16] Poggio, T. A. et al. 'Why and When Can Deep - but Not Shallow - Networks Avoid the Curse of Dimensionality: a Review'. In: *CoRR* vol. abs/1611.00740 (2016). arXiv: 1611.00740 (cit. on p. 50).

[Pow+21] Power, A. et al. 'Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets'. In: *ICLR MATH-AI Workshop*. 2021 (cit. on p. 52).

[PSG17] Pennington, J., Schoenholz, S. S. and Ganguli, S. 'Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice'. In: *CoRR* vol. abs/1711.04735 (2017). arXiv: 1711.04735 (cit. on p. 68).

[PT97] Pratt, L. and Thrun, S. 'Guest Editors' Introduction: Special Edition on Inductive Transfer'. In: *Machine Learning* vol. 28, no. 1 (July 1997), pp. 5–5 (cit. on p. 75).

[PW19] Putzky, P. and Welling, M. 'Invert to Learn to Invert'. In: *CoRR* vol. abs/1911.10914 (2019). arXiv: 1911.10914 (cit. on p. 75).

[PY16] Pitman, J. and Yakubovich, Y. *Successive maxima of samples from a GEM distribution.* 2016. arXiv: 1609.01601 [math.PR] (cit. on p. 86).

[RHW86] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. ‘Learning representations by back-propagating errors’. In: *Nature* vol. 323, no. 6088 (Oct. 1986), pp. 533–536 (cit. on pp. 52, 60).

[Ric66] Rice, J. R. ‘A Theory of Condition’. In: *SIAM Journal on Numerical Analysis* vol. 3, no. 2 (1966), pp. 287–310. eprint: https://doi.org/10.1137/0703023 (cit. on p. 40).

[RM16] Rezende, D. J. and Mohamed, S. *Variational Inference with Normalizing Flows*. 2016. arXiv: 1505.05770 [stat.ML] (cit. on p. 74).

[RMW14] Rezende, D. J., Mohamed, S. and Wierstra, D. ‘Stochastic Backpropagation and Approximate Inference in Deep Generative Models’. In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Xing, E. P. and Jebara, T. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, June 2014, pp. 1278–1286 (cit. on p. 57).

[Roh13] Rohan, R.-A. ‘Some remarks on the exponential map on the groups SO (n) and SE (n)’. In: *Proceedings of the Fourteenth International Conference on Geometry, Integrability and Quantization*. Institute of Biophysics and Biomedical Engineering, Bulgarian Academy of Sciences. 2013, pp. 160–175 (cit. on p. 91).

[Ros57] Rosenblatt, F. *The perceptron - A perceiving and recognizing automaton*. Tech. rep. 85-460-1. Ithaca, New York: Cornell Aeronautical Laboratory, Jan. 1957 (cit. on p. 47).

[RS08] Ranzato, M. and Szummer, M. ‘Semi-supervised learning of compact document representations with deep networks.’ In: *ICML*. Ed. by Cohen, W. W., McCallum, A. and Roweis, S. T. Vol. 307. ACM International Conference Proceeding Series. ACM, 2008, pp. 792–799 (cit. on p. 62).

[Rud87] Rudin, W. *Real and complex analysis*. Third. McGraw-Hill Book Co., New York, 1987, pp. xiv+416 (cit. on pp. 4, 24, 151, 170).

[RY08] Rynne, B. P. and Youngson, M. A. *Linear functional analysis*. Second. Springer Undergraduate Mathematics Series. Springer-Verlag London, Ltd., London, 2008, pp. x+324 (cit. on pp. 4, 151, 155).

[San+18] Santurkar, S. et al. ‘How Does Batch Normalization Help Optimization?’ In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS’18. Montréal, Canada: Curran Associates Inc., 2018, pp. 2488–2498 (cit. on p. 67).

[Sch95] Schervish, M. J. *Theory of statistics*. eng. Springer series in statistics. New York: Springer, 1995 (cit. on pp. 4, 27, 30, 31).

[Sha48] Shannon, C. E. ‘A mathematical theory of communication’. In: *The Bell System Technical Journal* vol. 27, no. 3 (1948), pp. 379–423 (cit. on p. 162).

[SLH20] Sum, J., Leung, C.-S. and Ho, K. ‘A Limitation of Gradient Descent Learning’. In: *IEEE Transactions on Neural Networks and Learning Systems* vol. 31, no. 6 (2020), pp. 2227–2232 (cit. on p. 72).

[Spe90] Specht, D. F. ‘Probabilistic neural networks’. In: *Neural Networks* vol. 3, no. 1 (1990), pp. 109–118 (cit. on p. 74).

[SPH07] Schölkopf, B., Platt, J. and Hofmann, T. ‘Efficient sparse coding algorithms’. In: *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference.* 2007, pp. 801–808 (cit. on p. 77).

[Spr+15] Springenberg, J. T. et al. ‘Striving for Simplicity: The All Convolutional Net’. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings.* Ed. by Bengio, Y. and LeCun, Y. 2015 (cit. on p. 56).

[Sri+14] Srivastava, N. et al. ‘Dropout: A Simple Way to Prevent Neural Networks from Overfitting’. In: *Journal of Machine Learning Research* vol. 15, no. 56 (2014), pp. 1929–1958 (cit. on p. 66).

[SS17a] Shumway, R. H. and Stoffer, D. S. *Time series analysis and its applications.* Fourth. Springer Texts in Statistics. With R examples. Springer, Cham, 2017, pp. xiii+562 (cit. on p. 58).

[SS17b] Srivastava, A. and Sutton, C. *Autoencoding Variational Inference For Topic Models.* 2017. arXiv: 1703.01488 [stat.ML] (cit. on p. 85).

[SZ15] Simonyan, K. and Zisserman, A. ‘Very Deep Convolutional Networks for Large-Scale Image Recognition’. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* Ed. by Bengio, Y. and LeCun, Y. 2015 (cit. on p. 85).

[Sze+14] Szegedy, C. et al. ‘Intriguing properties of neural networks’. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings.* Ed. by Bengio, Y. and LeCun, Y. 2014 (cit. on pp. 72, 85).

[Tes+20] Teshima, T. et al. ‘Coupling-based Invertible Neural Networks Are Universal Diffeomorphism Approximators’. In: *CoRR* vol. abs/2006.11469 (2020). arXiv: 2006.11469 (cit. on p. 75).

[Tib96] Tibshirani, R. ‘Regression Shrinkage and Selection via the Lasso’. In: *Journal of the Royal Statistical Society (Series B)* vol. 58 (1996), pp. 267–288 (cit. on p. 41).

[Tik43] Tikhonov, A. N. ‘On the stability of inverse problems’. In: *C. R. (Doklady) Acad. Sci. URSS (N.S.)* vol. 39 (1943), pp. 176–179 (cit. on pp. 1, 41).

[Tol+19] Tolstikhin, I. et al. *Wasserstein Auto-Encoders.* 2019. arXiv: 1711.01558 [stat.ML] (cit. on p. 74).

[Tol+21] Tolstikhin, I. O. et al. ‘MLP-Mixer: An all-MLP Architecture for Vision’. In: *CoRR* vol. abs/2105.01601 (2021). arXiv: 2105.01601 (cit. on pp. 3, 56).

[TP91] Turk, M. and Pentland, A. ‘Face recognition using eigenfaces’. In: *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* 1991, pp. 586–591 (cit. on p. 77).

[Tsi+19] Tsipras, D. et al. *Robustness May Be at Odds with Accuracy.* 2019. arXiv: 1805.12152 [stat.ML] (cit. on pp. 73, 74, 107).

[Tur48] Turing, A. M. ‘ROUNDING-OFF ERRORS IN MATRIX PROCESSES’. In: *The Quarterly Journal of Mechanics and Applied Mathematics* vol. 1, no. 1 (Jan. 1948), pp. 287–308. eprint: https://academic.oup.com/qjmam/article-pdf/1/1/287/5323145/1-1-287.pdf (cit. on p. 41).

[Tur60] Turin, G. ‘An introduction to matched filters’. In: *IRE Transactions on Information Theory* vol. 6, no. 3 (1960), pp. 311–329 (cit. on p. 56).

[Van16] Van Vleck, E. B. ‘Current tendencies of mathematical research’. In: *Bull. Amer. Math. Soc.* vol. 23, no. 1 (1916), pp. 1–13 (cit. on p. 1).

[Vap92] Vapnik, V. ‘Principles of Risk Minimization for Learning Theory’. In: *Advances in Neural Information Processing Systems.* Ed. by Moody, J., Hanson, S. and Lippmann, R. P. Vol. 4. Morgan-Kaufmann, 1992 (cit. on p. 37).

[Vin+08] Vincent, P. et al. ‘Extracting and composing robust features with denoising autoencoders’. In: *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008.* Ed. by Cohen, W. W., McCallum, A. and Roweis, S. T. Vol. 307. ACM International Conference Proceeding Series. ACM, 2008, pp. 1096–1103 (cit. on p. 61).

[Vin+10] Vincent, P. et al. ‘Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion’. In: *Journal of Machine Learning Research* vol. 11, no. 110 (2010), pp. 3371–3408 (cit. on pp. 61, 62, 121).

[Wan+04] Wang, Z. et al. ‘Image quality assessment: from error visibility to structural similarity’. In: *IEEE Transactions on Image Processing* vol. 13, no. 4 (2004), pp. 600–612 (cit. on p. 107).

[Wis60] Wise, M. E. ‘On Normalizing the Incomplete Beta-Function for Fitting to Dose-Response Curves’. In: *Biometrika* vol. 47, no. 1/2 (1960), pp. 173–175 (cit. on p. 87).

[Xia+18] Xiao, L. et al. *Dynamical Isometry and a Mean Field Theory of CNNs: How to Train 10,000-Layer Vanilla Convolutional Neural Networks.* 2018. arXiv: 1806.05393 [stat.ML] (cit. on p. 68).

[Xu+21] Xu, X. et al. ‘Toward Effective Intrusion Detection Using Log-Cosh Conditional Variational Autoencoder’. In: *IEEE Internet of Things Journal* vol. 8, no. 8 (2021), pp. 6187–6196 (cit. on p. 106).

[Yar18] Yarotsky, D. ‘Universal approximations of invariant maps by neural networks’. In: *CoRR* vol. abs/1804.10306 (2018). arXiv: 1804.10306 (cit. on pp. 50, 130).

[Yin18] Yingzhen Li John Bradshaw, Y. S. ‘Are Generative Classifiers More Robust to Adversarial Attacks?’ In: *CoRR* vol. abs/1802.06552 (2018). arXiv: 1802.06552 (cit. on pp. 34, 74).

[YP19] Yu, S. and Príncipe, J. C. ‘Understanding autoencoders with information theoretic concepts’. In: *Neural Networks* vol. 117 (2019), pp. 104–123 (cit. on p. 61).

[Zei12] Zeiler, M. D. ‘ADADELTA: An Adaptive Learning Rate Method’. In: *CoRR* vol. abs/1212.5701 (2012). arXiv: 1212.5701 (cit. on p. 53).

[Zha+16] Zhang, C. et al. ‘Understanding deep learning requires rethinking generalization’. In: *CoRR* vol. abs/1611.03530 (2016). arXiv: 1611.03530 (cit. on p. 77).

[Zha+18] Zhang, H. et al. *WHAI: Weibull Hybrid Autoencoding Inference for Deep Topic Modeling*. 2018. arXiv: 1803.01328 [stat.ML] (cit. on p. 85).

[Zho+17] Zhou, L. et al. ‘The Expressive Power of Neural Networks: A View from the Width’. In: *CoRR* vol. abs/1709.02540 (2017). arXiv: 1709.02540 (cit. on p. 50).