

*Digital Comprehensive Summaries of Uppsala Dissertations  
from the Faculty of Science and Technology 2392*

# On Deep Learning for Low- Dimensional Representations

DANIEL GEDON



ACTA UNIVERSITATIS  
UPSALIENSIS  
2024

ISSN 1651-6214  
ISBN 978-91-513-2102-8  
urn:nbn:se:uu:diva-526130



UPPSALA  
UNIVERSITET

Dissertation presented at Uppsala University to be publicly examined in room 80121, Ångströmlaboratoriet, Lägerhyddsvägen 1, Uppsala, Friday, 14 June 2024 at 09:15 for the degree of Doctor of Philosophy. The examination will be conducted in English. Faculty examiner: Prof. Adam Johansen (Department of Statistics, University of Warwick, Coventry, U.K.).

## Abstract

Gedon, D. 2024. On Deep Learning for Low-Dimensional Representations. *Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology* 2392. 110 pp. Uppsala: Acta Universitatis Upsaliensis. ISBN 978-91-513-2102-8.

In science and engineering, we are often concerned with creating mathematical models from data. These models are abstractions of observed real-world processes where the goal is often to understand these processes or to use the models to predict future instances of the observed process. Natural processes often exhibit low-dimensional structures which we can embed into the model. In mechanistic models, we directly include this structure into the model through mathematical equations often inspired by physical constraints. In contrast, within machine learning and particularly in deep learning we often deal with high-dimensional data such as images and learn a model without imposing a low-dimensional structure. Instead, we learn some kind of representations that are useful for the task at hand. While representation learning arguably enables the power of deep neural networks, it is less clear how to understand real-world processes from these models or whether we can benefit from including a low-dimensional structure in the model.

Learning from data with intrinsic low-dimensional structure and how to replicate this structure in machine learning models is studied within this dissertation. While we put specific emphasis on deep neural networks, we also consider kernel machines in the context of Gaussian processes, as well as linear models, for example by studying the generalisation of models with an explicit low-dimensional structure. First, we argue that many real-world observations have an intrinsic low-dimensional structure. We can find evidence of this structure for example through low-rank approximations of many real-world data sets. Then, we face two open-ended research questions. First, we study the behaviour of machine learning models when they are trained on data with low-dimensional structures. Here we investigate fundamental aspects of learning low-dimensional representations and how well models with explicit low-dimensional structures perform. Second, we focus on applications in the modelling of dynamical systems and the medical domain. We investigate how we can benefit from low-dimensional representations for these applications and explore the potential of low-dimensional model structures for predictive tasks. Finally, we give a brief outlook on how we go beyond learning low-dimensional structures and identify the underlying mechanisms that generate the data to better model and understand these processes.

This dissertation provides an overview of learning low-dimensional structures in machine learning models. It covers a wide range of topics from representation learning over the study of generalisation in overparameterized models to applications with time series and medical applications. However, each contribution opens up a range of questions to study in the future. Therefore this dissertation serves as a starting point to further explore learning of low-dimensional structure and representations.

**Keywords:** Machine Learning, Deep Learning, Gaussian Process, Low-Dimensional Representations, Representation Learning, Dynamical Systems, Electrocardiogram

*Daniel Gedon, Department of Information Technology, Division of Systems and Control, Box 337, Uppsala University, SE-75105 Uppsala, Sweden. Department of Information Technology, Artificial Intelligence, Box 337, Uppsala University, SE-75105 Uppsala, Sweden.*

© Daniel Gedon 2024

ISSN 1651-6214

ISBN 978-91-513-2102-8

URN urn:nbn:se:uu:diva-526130 (<http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-526130>)

*To hope and passion.*



# Sammanfattning

Artificiell intelligens handlar i sin essens om att ta fram matematiska modeller från data. Dessa data samlas in från verkliga observationer till exempel naturliga bilder av din katt, smaken av vin baserat på olika egenskaper eller ditt hjärtas elektriska aktivitet över tid mätt i ett elektrokardiogram. Modellerna som vi erhåller används sedan för att antingen bättre förstå fenomenet som beskriver den verkliga observationen eller för att göra förutsägelser för andra liknande observationer.

För det mesta upplever vi högdimensionella observationer. Som ett exempel tänk på bilder av katter. Dessa bilder består av pixlar. Således har en bild på  $50 \times 50$  pixlar en total dimensionalitet på 2,500. Om vi vill designa en modell som kan beskriva om en bild innehåller en katt eller inte, skulle ett sätt vara att titta på varje pixel individuellt. I så fall måste vi analysera alla 2,500 pixlar. Däremot, om vi människor skulle klassificera katter i bilder, skulle vi leta efter särdrag som päls, ben, morrhår och så vidare. Vanligtvis finns det en liten mängd sådana särdrag som beskriver bilden. Den högdimensionella bilden har alltså i verkligheten en lågdimensionell struktur. Detta gäller inte bara för bilder utan för de flesta verkliga observationer. Ett annat exempel är elektrokardiogram. Medan ett elektrokardiogram består av flera kanaler och många tidsmätningar, är den intressanta delen för kardiologer, för att kunna att ställa en diagnos, inbördes förhållandet mellan olika segment. Detta kan i sin tur modelleras från själva hjärtats pulserande beteende. Således har högdimensionella observationer av ett elektrokardiogram faktiskt någon underliggande lågdimensionell struktur.

Historiskt sett, när vi som människor skapade matematiska modeller av observationer, beaktade vi ofta denna underliggande struktur i den observerade datan. Till exempel, när Newton härleddes sin universella gravitationslag, insåg han att en enda konstant — gravitationskonstanten — beskriver förhållandet mellan två kroppar till varandra. Inom området artificiell intelligens använder vi ofta en modell som kallas ett djupt neuralt nätverk. Dessa modeller består av flera lager som tillsammans bildar modellen. Djupa neurala nätverk är kraftfulla och gör att vi kan förutsäga klassen för ny osedd data exakt. Strukturen hos djupa neurala nätverk är dock ofta generell och inkluderar inte direkt den lågdimensionella strukturen i data som tidigare, så kallade mekanistiska modeller gör. Därför kan vi inte förstå data genom att skapa en modell av den och vi kan inte fullt ut följa beslutsprocessen inom modellen.

Om vi kunde lära oss den lågdimensionella strukturen i en maskininlärningsmodell eller explicit påtvinga den, så har vi en vad potentiella fördelar. Till exempel, om strukturen som ligger till grund för datan inte är känd i förväg, kan vi förstå det verkliga fenomenet bättre genom modellen. Dessutom kommer vår modell potentiellt att prestera bättre, särskilt för osedda närliggande datapunkter. Om vi explicit påtvingar en känd struktur, kommer modellen möjligen att kräva mindre data för att fungera korrekt.

Denna avhandling handlar om kombinationen av högdimensionell data som har en inneboende lågdimensionell struktur med maskininlärningsmodeller. Målet är att identifiera om generella maskininlärningsmodeller lär sig denna inneboende lågdimensionella struktur från data av sig själva eller hur vi kan påtvinga strukturen. I vissa fall kommer proceduren för hur man hittar den bästa modellen att driva modellen mot att lära sig strukturen som är kodad i data. I andra fall måste vi uttryckligen påtvinga denna struktur. Det enklaste sättet att göra detta på är att låta vissa lager i det neurala nätverket ha kapacitet att bara lära in ett begränsat antal särdrag. Ofta kallas detta ett flaskhalslager eftersom det begränsar informationsflödet och tvingar nätverket att lära sig en komprimerad representation av datat.

Utöver dessa mer grundläggande aspekter av vad djupa neurala nätverk lär sig, är vi också intresserade av tillämpningar. Här är vi intresserade av hur lågdimensionella strukturer inom maskininlärningsmodellen kan gynna den specifika tillämpningen. Specifikt tittar vi på två typer av tillämpningar. Den första gäller modellering av dynamiska system. Sådana system beskriver i huvudsak hur saker och ting förändras över tiden, oavsett om det är planeternas rörelser eller hjärtats slag. Därför är det av intresse hur vi kan bygga in lågdimensionella strukturer i modeller av sådana system. Den andra typen av tillämpning kretsar kring medicinsk diagnosstöd, specifikt för elektrokardiogram som vi nämnde ovan. Vi beskriver olika sätt på hur vi kan påtvinga lågdimensionella strukturer i modellen och hur vi kan analysera modellens beslutsprocess.

Även om denna avhandling ger en detaljerad introduktion till ämnet att lära sig lågdimensionella strukturer i maskininlärningsmodeller, finns det många återstående frågor. Genom att utforska dessa anvisningar mer i detalj kommer maskininlärningsmodeller att bättre förstå verkliga data och öka deras prestanda. Så småningom tillåter det oss att identifiera strukturen på själva datan och därigenom bidra till vetenskapliga upptäckter i framtiden.

# Acknowledgements

People say that “it takes a village to raise a child”. I don’t know much about children but I got to learn that it also takes a village of support to form a PhD student. Even though for me this village is spread over multiple countries, every single person played a crucial role in helping me get where I am today. The PhD journey over the last five years has been challenging but also incredibly rewarding and nothing short of wonderful. Therefore, this part is dedicated with gratitude to putting those into the spotlight and to giving back a small portion to those who contributed to this dissertation.

First, I would like to thank my supervisor Thomas Schön. Obviously, this dissertation would not have been possible without you. Thank you for your guidance, support and advice in every part that a PhD journey comes with; for giving me the time and freedom to find my own academic path; for always staying calm and relaxed, no matter how stressed I was; simply for taking care. A huge thank you also to my co-supervisors. Niklas Wahlström, your valuable feedback have always improved our projects. You decisively advanced my research with your critical comments and commitment. To Antônio H. Ribeiro, thank you for all the work we have done together and your detailed help with my problems. I aspire to your level of expertise and how you never lose the fun part of all the seriousness in academia. Thank you also to Lawrence Murray who initially secured the funding for my position, even though we never got to work together and I ended up with a completely different topic. I learnt an incredible amount from all of you and will always be grateful for it.

This dissertation rests on a pile of research which I would not have been able to do without my collaborators and co-authors: Daniel Martins Teixeira, Manoel Horta Ribeiro, Antonio L. Pinho Ribeiro, Wagner Meira, Stefan Gustafsson, Erik Lampa, Martin J. Holzmann, Johan Sundström, Philipp von Bachmann, Fredrik K. Gustafsson, Carl Jidling, Claudia Di Lorenzo Oliveira, Clareci Silva Cardoso, Ariela Mota Ferreira, Luana Giatti, Sandhi Maria Barreto, Ester C. Sabino, Theogene Habineza, Joachim A. Behar, Lennart Ljung, Gianluigi Pillonetto, Aleksandr Aravkin, Amirhesam Abedsoltan, Mikhail Belkin, Thomas Schön, Niklas Wahlström and Antônio H. Ribeiro. It has been a pleasure and an honour to work with all of you and learn from your expertise. You were a fundamental component of my development as a researcher.

This journey started long before the start of my PhD. I would like to mention Thomas Ott, Jens Levenhagen and Stefan Winkler at Airbus who provided the

right guidance to find my path already during my Bachelor's. Thank you to my supervisors and mentors during my Master's at TU Delft Pieter Piscaer, Kim Batselier, and Manon Kok who inspired my research and especially to Michel Verhaegen for his fascinating passion.

The Division of Systems and Control at Uppsala University has been an incredible place to do a PhD and become a researcher. While it was not as easy to start a PhD in a new city right before a (hopefully) once-in-a-lifetime pandemic, the support from the group and the wonderful colleagues have been invaluable. Many of you have become close friends and often brightened my days. Thank you to Fredrik for being an inspiration in your dedication, ideas and kindness; to Ludvig for always taking the time for long in-depth discussions; to Sofia for our deep conversations, your calm view of things and your open attitude; to Paul for always convincing me to go out and have fun; to Anna F for trying so hard to convince us of spikeball; to Bernhard for sharing your fun stories; to Linnéa for adding a powerful female voice and never stopping to question machine learning; to Tung for being one of the nicest persons I met; to Alessandro, Lovisa, Fabio and Daniel for the fun fika and lunch breaks, I wish you would have joined earlier and we could have spent more time together. As well as to Ruoqi, Philipp, Ziwei and many more for the insightful discussions and to all the former PhD students including David, the actual Anna (W), Calle J, Calle A, Osama, Fredrik, Håkan and many more to share all your wisdom and small tips. Special thanks also to Fredrik and Calle J for reading and improving this dissertation. Thanks to Zheng and Sebastian for answering all my random questions. Also, not to forget a large thank you to Eva, Anna-Lena, Gunilla, Marina and all the administrative staff without whom nothing would frankly work.

I am also very grateful to Misha Belkin at UC San Diego for hosting me and being the most critical lovely researcher. The discussions with Amir, Daniel, Neil, Adit, Parthe and all the others in the lab were inspiring. Especially thank you to Amir, Chester, Daniel, Jonathan and Partha for integrating me so quickly and going out on all these wonderful hikes and adventures.

To the whole Bikini Bottom crew Adrian, Dominik, Lukas, Michi, Niklas, Sebi, Simon, Tim and Tobi for actively refusing to grow up. Our stupid ideas definitely help me to stay (in)sane. You never fail to provide a sense of home even though I am barely around. The friendship with each one of you simply amazes me. Thank you also to the Platen crew Simon, Patrick, and Andreas. I hope to still be in touch in decades. Also a huge thanks to Dominik for it always being as in old times with you, to Alex for all the beer and to Lysann for us never losing our friendship. To all the people from FN, especially Johannes, Thomas, Marius, and Flo for our absolutely well-planned trips and also to Chris and Fabi. To Carsten, Dirk and Edgar for convincing me to start running and eventually even get into cycling. It was the necessary relief from staring into that

screen for way too long. Thanks to everyone in Delft who made life so much more fun. Especially Corrado, Elia and Daniela for being the best roommates and enduring my Germanness. But also to Elena, Jorge, Ana, Wouter, Maria and Magda. Thank you Daniel and Alex for the fun rides, Marcin for the good time and André for all the random fun. To Martijn for being the best Yoga teacher and an inspiration. I would also like to thank all the wonderful people that I got to spend time with in Uppsala. Especially to Darius for all our long conversations about any random topic, your openness and for always helping out. Colin, thank you for the deep talk and for adopting me to the maths gang. To Amandine, Yu, Carmina and Maeva for all the nights out. Thanks to Abeba for relentlessly proving that it is worth to stand up for the things that matter. A huge part of this dissertation also goes to Sri. You are an amazing friend and I am truly sad that our paths might split (for) now. Also to you and Shruthi for not trying to kill me with Indian food. To Žana for never losing touch.

I try to be aware and grateful for all the little things. Therefore, I am incredibly thankful for all the lucky circumstances over my lifetime that led me to this point and allowed me to be who I am. To all the strangers who crossed my path and made my day, inspired deep thoughts and impacted my future.

An Mama und Papa. Danke dass ihr mir immer die Freiheit gegeben habt, das zu tun und machen, was ich mir in den Kopf setze und ihr mich immer dabei unterstützt. Ihr habt mir gezeigt, nie den Mut zu verlieren, bestimmt meinen Weg zu gehen und immer freundlich und fröhlich zu bleiben. Danke, dass ihr jederzeit mit Rat und Tat zur Seite steht. Danke an Nina, die beste Schwester, die man sich nur wünschen kann. Egal, was passiert oder wie weit wir entfernt sind, ich weiß dass ich immer auf dich zählen kann.

Finally, to the most important person. Pauline. I am proud of the relationship we have built and all the experiences we have collected together. You always provide the right perspective, you manage to calm me down and cheer me up, you teach me uncountable small things, and showed me the power of a Drinnie. You simply never cease to amaze me. I can not imagine how I could have managed this dissertation without your support, help, words of encouragement and your unbounded love. Thank you!

*Uppsala, April 2024  
Daniel Gedon*

**Funding** My research was financially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.



# List of Papers

This thesis includes the following papers:

- I **Uncertainty Estimation with Recursive Feature Machines.** *Daniel Gedon, Amirhesam Abedsoltan, Thomas B. Schön, Mikhail Belkin.* Submitted, 2024.
- II **Invertible Kernel PCA with Random Fourier Features.** *Daniel Gedon, Antônio H. Ribeiro, Niklas Wahlström, Thomas B. Schön.* IEEE Signal Processing Letters, vol. 30, pp. 563-567, 2023.
- III **No Double Descent in Principal Component Regression: A High-Dimensional Analysis.** *Daniel Gedon, Antônio H. Ribeiro, Thomas B. Schön.* Submitted, 2024.
- IV **Deep State Space Models for Nonlinear System Identification.** *Daniel Gedon, Niklas Wahlström, Thomas B. Schön, Lennart Ljung.* 19th IFAC Symposium on System Identification (SYSID), 2021.
- V **First Steps Towards Self-Supervised Pretraining of the 12-Lead ECG.** *Daniel Gedon, Antônio H. Ribeiro, Niklas Wahlström, Thomas B. Schön.* Computing in Cardiology (CinC), 2021.
- VI **Development and Validation of Deep Learning ECG-Based Prediction of Myocardial Infarction in Emergency Department Patients.** *Stefan Gustafsson, Daniel Gedon, Erik Lampa, Antônio H. Ribeiro, Martin J. Holzmann, Thomas B. Schön, Johan Sundström.* Scientific Reports, vol. 12, 19615, 2022.



# Contents

1	Introduction .....	1
1.1	Broad motivation .....	2
1.2	Summary of the main results .....	4
1.3	Contributions .....	7
2	Modelling and learning .....	15
2.1	Motivation for learning .....	15
2.2	Deep neural networks .....	18
2.3	Kernel machines .....	20
2.4	Probabilistic modelling .....	23
2.5	Combining deep neural networks and kernel machines .....	27
3	Learning low-dimensional representations .....	31
3.1	Supervised learning .....	33
3.2	Unsupervised learning .....	34
3.3	Self-supervised learning .....	41
4	Overparameterization .....	47
4.1	Background .....	48
4.2	Generalisation risk .....	51
4.3	Analysis with low-dimensional manifold data .....	53
4.4	Discussion .....	57
5	Dynamical systems .....	59
5.1	Classical modelling .....	60
5.2	Modelling dynamics with neural networks .....	62
5.3	Deep state-space models .....	65

6	Medical applications .....	69
6.1	Modelling of electrocardiograms .....	70
6.2	Interpretability and explainability .....	72
7	Concluding remarks .....	75
	References .....	79

# CHAPTER 1

## Introduction

This dissertation concerns the learning of low-dimensional representations with the use of deep learning-based approaches. Let me first add some personal perspective and background on this topic. Before starting my PhD studies in the fall of 2019, the focus of my studies was on automatic control and specifically system identification for large-scale systems. We were concerned with linear systems, stability proofs, identifiability and small models with physically explainable parameters. Meanwhile, machine learning and in particular deep neural networks became widespread. Multiple breakthroughs lead to this ubiquitous adoption of machine learning including advances in computational power, availability of large-scale datasets, and improvements in modelling techniques [GBC16]. Specifically, in computer vision, the continuing improvements in image classification as seen by the ImageNet challenge [Den+09] after AlexNet [KSH12] and ResNet [He+16]; results from the reinforcement learning community about AlphaGo beating the world champion Lee Sedol in the game of Go [Sil+17]; state-of-the-art generated images through GANs [Goo+20]; and the revolution of NLP through transformers [Vas+17] demonstrated the power of deep learning. These breakthroughs put the spotlight on machine learning-based models and highlighted that future research cannot be imagined without those models. Nevertheless, in certain research communities, there were hesitations against deep models because of their black-box nature with millions of parameters, the absence of performance or stability guarantees, and the existance of adversarial examples which demonstrates their brittle behaviour [Sze+14]. These arguments hint that understanding the internal details of the decision-making process for predictions within deep neural networks may be challenging.

I started my PhD studies in deep learning with the hope of understanding some aspects of deep models better and sharing this knowledge. Amazed by the modelling power of deep neural networks, we quickly deviated from my initial hope and started with applied research in the fields of dynamical system identification and medical prediction models. Yet, the thought of trying to understand neural networks better never left my mind. Thus, we started to study representation learning [BCV13] including self-supervised methods [Che+20a] and dis-

entanglement [Loc+19]. The latter relies on the assumption that a small set of factors generates natural data. The goal is then to learn useful low-dimensional representations for these data. It became clear that the structure of the data is a crucial aspect of understanding the power of neural networks. Therefore, in the second half of my PhD, we returned to understanding neural networks by focusing on the data structure, particularly with low-dimensional representations. In this dissertation, we will cover the theoretical and applied aspects of my research over the past years.

## 1.1 Broad motivation

When humans describe their understanding of the world, we attribute certain structures to observations of real-world processes often to formulate explanations and possibly predictions. In this broad motivation, we will first provide examples that historically early modelling often contained low-dimensional components. Their ability to describe reality precisely implies that natural observations exhibit intrinsic low-dimensional structure. Modelling in such an interpretable way guided development in the early so-called Artificial Intelligence (AI) community with logic-based and expert models. However, the success of AI with deep learning was based on the abstract idea of learning some kind of representation of the data, effectively removing low-dimensional interpretable components in the model. Within this dissertation, we will argue and show that modern deep learning can actually benefit from incorporating low-dimensional elements in the model.

**Modelling of observations.** In the ancient Europe of the 5th century BCE, the Greek philosopher Empedocles proposed the classical elements consisting of earth, water, air and fire as a philosophical concept to explain matter in simpler terms by decomposing them into four elements [Rus04]<sup>1</sup>. This idea was later refined by Aristotle [Joa84] and was expanded by Plato with the decomposition into his definition of platonic solids [Pla98]. With the start of the Scientific Revolution in the 16th century, Copernicus offered an alternative to the ancient philosophical view of real-world concepts towards mechanical, mathematical descriptions [Cop43]. For example, experiments such as Galilei's leaning Tower of Pisa experiment<sup>2</sup> identified the relation between speed and time of falling objects [Gal38] which Newton formalised in his law of universal

---

<sup>1</sup>Notably, outside of Europe equivalent concepts were derived. Examples include Mahābhūta in Indian Hinduism, Godai in Japan, Wuxing in Chinese Taoism and in the ancient Bön and Indo-Tibetan Buddhism.

<sup>2</sup>Again, there exist multiple parallel experiments and discoveries around the law of gravitation outside of the Scientific Revolution in Europe.

gravitation [New87]

$$F = G \frac{m_1 m_2}{r^2}. \quad (1.1)$$

Here, the force  $F$  is proportional to the masses  $m$  of two objects and the distance of their centres of masses  $r$ . These variables are connected by the gravitational constant  $G$ . The equation gives mathematical structure to the natural phenomenon of gravity. It provides a model for gravity based on a single constant factor  $G$ , to describe the gravitational relation of two bodies. We can argue that this simplified model does not describe reality exactly since Einstein's general theory of relativity [Ein16] provides a model that offers more precise predictions. However, in many settings, Newton's model sufficiently describes a real-world phenomenon with a small number of parameters to identify (the gravitational constant  $G$ ). Therefore, it hints at the intrinsic low-dimensional structure of the phenomenon.

Newton's theory of gravity is just one example which indicates intrinsic low-dimensional structure. Numerous other phenomena in the real world also suggest the presence of such a structure. This includes physical models such as electromagnetic systems which can be described by Maxwell's equations [Max65] or chaotic dynamical systems with the presence of attractors which lead the system's trajectories from a high-dimensional state-space onto a low-dimensional manifold [OGY90]. In mathematics, fractal geometry describes complex geometric structures by simple mathematical equations [Man82]. Biological systems tend towards compression of information, e.g. through the visual system which identifies the relevant lower-dimensional information for further processing in the brain [OF96]. As a final example, social networks exhibit sparse connections suggesting that a small set of factors such as shared interests and geographic proximity determines social interaction [GN02].

**Modelling with AI.** The idea of modelling complex systems with interpretable, simple and structured models encompasses not just natural science and engineering. Similarly, starting in the early 20th century analytic philosophy was based on the use of formal structure, especially logic to discern underlying principles and arguments [Moo03]. Whitehead and Russell applied this idea to formalise mathematics in terms of logic [WR27]. Correspondingly, in computer science, early models of the AI community relied on logic-based symbolic computing. The goal was to create systems that could reason and solve problems using logical rules and representations, aiming to mimic human cognitive abilities and ultimately provide intelligent systems<sup>3</sup>. Later this was encompassed into so-called expert systems [GR98]. However, the discrete logical calculus of these models was difficult to deal with for real-world data [RN10]. Consider describing for example a cat only with logical operators.

---

<sup>3</sup>While the term ‘intelligent’ is until today not clearly defined which renders the term AI ambiguous and possibly misleading.

This endeavour quickly reaches its limits due to the variations in real-world data. Defining that cats have fur is not true for the Sphynx breed or considering four legs and whiskers can confuse it with dogs that have similar features. Therefore, a more general modelling strategy has to be developed.

**Representation learning.** In contrast to hard, analytic approaches of describing for example a cat, Wittgenstein explores the philosophical concept of family resemblance around the same time as logic-based systems were developed [Wit53]. Family resemblance is a less strictly logical perspective to say that a cat is identified based on resembling other members of its general category of cats. Similarly, early connectionist approaches [Ros58] up to current deep learning research [RHW+85; LBH15] moved toward the abstract idea of representations. Here, a data point such as a cat is encoded as a point in a representation space and similar data points such as the general category of cats are encoded close in that space.<sup>4</sup> Models based on abstract representations are often large, containing millions of parameters defying the principle of parsimony [Sob81] stating that the simplest model structure should be used. Furthermore, they challenge common statistical learning assumptions about models with more parameters than training data points, which were considered to overfit the nuisance effects in the data. Incorporating low-dimensional elements in the model to resemble the structure of the data can help modern deep neural networks. This applies for example to the modelling of temporal data from dynamical systems and to medical modelling for decision support systems.

**This dissertation.** We have argued that natural data has an underlying low-dimensional structure. Whereas in many domains, this structure is modelled explicitly, in the field of machine learning this has shifted towards learning useful representations. The connection between both fields—data with inherent low-dimensional structure and representation learning—is the topic of this dissertation. We will first explore how modern machine learning approaches encode high-dimensional observations into lower-dimensional representations. Then, we will demonstrate the usefulness of such low-dimensional representations for certain applications.

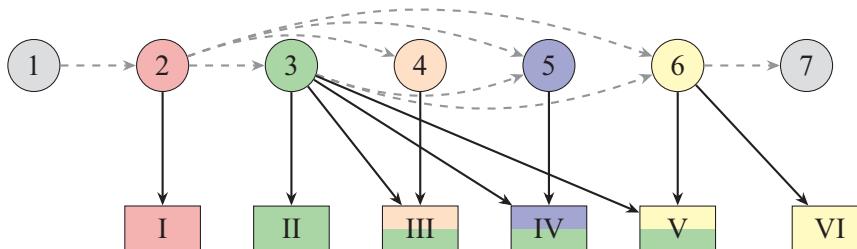
## 1.2 Summary of the main results

This dissertation is divided into two main parts. Part A includes an overview of the topic of learning from low-dimensional representations and embeds it into the wider context of current machine learning research. Part B contains six scientific papers in this field, which are sorted according to the chapters in Part A. Figure 1.1 provides a guideline of how the chapters in Part A connect

---

<sup>4</sup>This argument is in part influenced by comments from Sophia Sanborn during NeurIPS 2023.

and which chapters are beneficial to read for which paper in Part B. The goal of Part A is to provide the relevant background to help study the six included papers which constitute Part B. These papers form the relevant scientific contribution to this dissertation that I conducted during my PhD studies.



**Figure 1.1:** Relation of chapters (top) to each other with dashed arrows and relation of chapters to papers (bottom) with solid arrows.

We start with a brief introduction to supervised learning in Chapter 2. Specifically, we introduce modelling using deep neural networks and kernel machines. We show how we can utilise interconnections between both modelling approaches. In Chapter 3, we provide an overview of different representation learning methods. We discuss possible supervised learning methods and why they are limited in practice. We move to the realm of unsupervised representation learning including Principal Component Analysis (PCA) and its nonlinear extension with kernel PCA before explaining how we can generate supervised signals from the data itself using self-supervised learning. In Chapter 4 we discuss theoretical properties when utilising data with low-dimensional structure. We assume that the training data has a low-dimensional structure. The question is then, how well different kinds of machine learning models generalise when learning from this data. We are particularly interested in the so-called generalisation risk. This line of study concerns the phenomenon of double descent, where the generalisation risk first follows the classical U-shaped bias-variance trade-off when we include more and more parameters into the model. But once we include more parameters than training data points, the classical statistical theory does not hold anymore and we observe that the risk decreases again—the double descent.

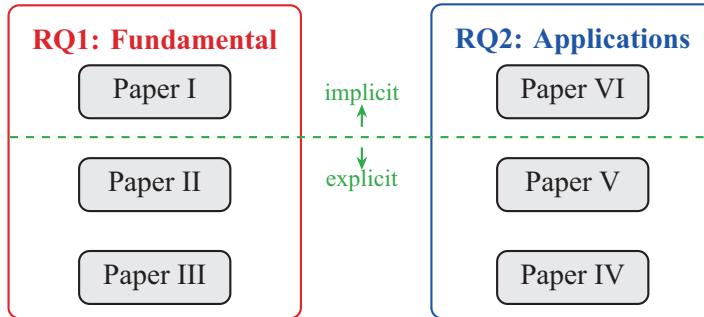
In Chapter 5 we discuss how we can apply deep neural networks to dynamical systems. We introduce the problem of system identification where we want to find a model that describes the dynamics of a system over time and how we can use deep neural networks for this task. In Chapter 6, we go into medical applications, specifically the modelling of electrocardiogram (ECG) data. Here, we start by explaining the specific problem with electrocardiogram data and how deep neural networks for diagnosis prediction can be utilised as decision-support tools for medical practitioners.

Overall, this set of topics is defined by the following two research questions (RQs):

**RQ 1. What fundamental observations can we make when using machine learning approaches for low-dimensional data structures?**

**RQ 2. How can we utilise low-dimensional representations for deep learning in applications?**

Figure 1.2 provides an overview of how the research questions that we address in the background in Part A are connected to the papers in Part B. Furthermore, the green dashed line shows another divide of the papers: into those where we (1) implicitly learn low-dimensional representations e.g. through inductive biases of the optimisation; and those where we (2) explicitly enforce this structure through design choices.



**Figure 1.2:** Conceptual divide of the six papers in Part B and how they are related to the research questions addressed in Part A.

**RQ 1: Fundamental.** To answer this research question, we investigate how methods implicitly learn low-dimensional representations, e.g. through the implicit biases inherent in learning approaches. Further, we analyse how we can explicitly encode a low-dimensional structure in the model and how well such models work in practice.

In *Paper I* [Ged+24] we explore the Recursive Feature Machine (RFM) model. This model provides one possible connection between kernel machines including Gaussian Processes and deep neural networks. Here, the optimisation algorithm implicitly provides low-dimensional representations through the re-weighting of input features.

In *Paper II* [Ged+23] we focus on kernel PCA as a nonlinear extension of PCA to find nonlinear transformations for low-dimensional structures of the data. To verify that the found representation is useful, we often rely on generative models, which can reconstruct the original input from the low-dimensional representation. Contemporary methods rely on solving a supervised learning

problem for the reconstruction of the input to a kernel PCA representation. We provide a novel method termed invertible kernel PCA that allows a direct input reconstruction without optimisation.

*Paper III* [GRS24] continues the line of work on PCA and extends it with linear regression to obtain Principal Component Regression (PCR). We analyse the generalisation capabilities of this model to unseen test data. Specifically, if we have observations from data on low-dimensional (linear) manifolds, we can quantify the generalisation risk precisely.

**RQ 2: Application.** In this research question, we focus on using deep neural network-based models that encode low-dimensional data structures within dynamical system identification and for medical applications, specifically with data from the ECG.

*Paper IV* [Ged+21b] is concerned with a specific model for dynamical systems, called deep state-space models. Here, we encode the input signal into a low-dimensional hidden state. This state is supposed to represent past information and is used to decode the output time series.

In *Paper V* [Ged+21a] we design a self-supervised learning method for ECG data and explain how it learns useful low-dimensional representations for downstream tasks. While this is a methodological development, it is tailored to ECG data and therefore part of the answer to RQ 2.

In *Paper VI* [Gus+22] we draw connections to one application for ECG modelling as a decision support tool. We train a standard ResNet-based deep neural network for heart attack prediction. In this setting, we implicitly learn low-dimensional representations due to the architectural setup of the network into lower dimensions. We explore this with interpretability tools such as Grad-CAM to explain the predictions of the model.

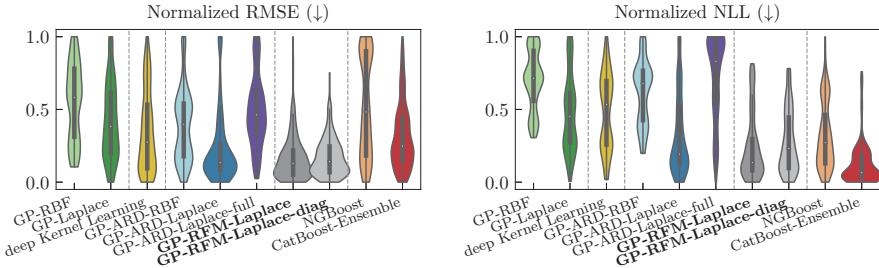
## 1.3 Contributions

### Papers included in the dissertation

The six papers included in Part B of this dissertation are briefly summarised next, together with a short explanation of my contributions.

## Paper I

**Uncertainty Estimation with Recursive Feature Machines.** *Daniel Gedon, Amirhesam Abedsoltan, Thomas B. Schön, Mikhail Belkin.* Submitted, 2024.



**Summary** We study a novel feature learning kernel machine, called Recursive Feature Machine (RFM) in the context of uncertainty quantification. To achieve this, we embed this new kernel in Gaussian Processes (GPs). We compare our resulting GP-RFM-Laplace<sup>5</sup> with a range of state-of-the-art GP and Boosting approaches on tabular benchmark datasets. Overall, we find that the resulting construction either outperforms or matches the performance of existing state-of-the-art methods.

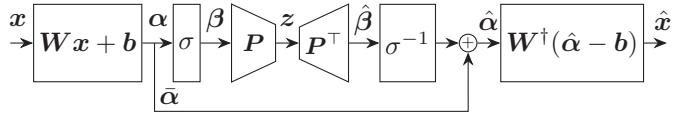
**Statement of Contribution** Amirhesam and I explored new directions for the RFM and jointly came up with the idea of embedding RFMs in GPs for uncertainty quantification. I implemented and conducted all experiments in discussion and with feedback from Amirhesam. The writing of the paper was split between Amirhesam and me with feedback from Mikhail and Thomas. This work started during a research visit to Mikhail Belkin’s lab at UCSD in the spring of 2023 and was finalised between all authors after the visit.

---

<sup>5</sup>While often the Gaussian or RBF kernel (2.10) is used for GPs, we utilise the less often applied Laplace kernel (2.11). For an introduction to kernels, see Section 2.3.

**Paper II**

**Invertible Kernel PCA with Random Fourier Features.** *Daniel Gedon, Antônio H. Ribeiro, Niklas Wahlström, Thomas B. Schön.* IEEE Signal Processing Letters, vol. 30, pp. 563-567, 2023.

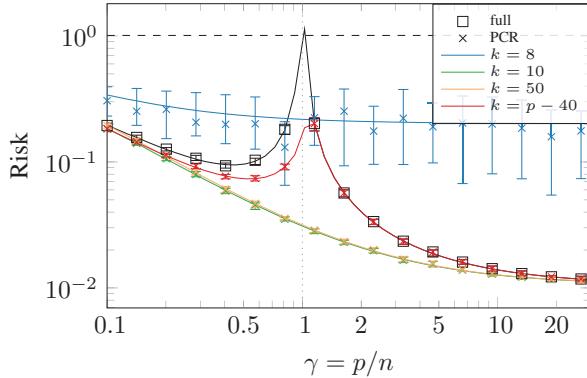


**Summary** We propose a method for the reconstruction of low-dimensional representations obtained through kernel Principal Component Analysis (PCA). To achieve this, we approximate the kernel with random Fourier features and exploit the fact that the nonlinear transformation of the kernel is invertible in a certain subdomain. Therefore, the reconstruction follows naturally from the method without solving an optimisation problem. We showcase the effectiveness of our approach in different data modalities including images and ECGs.

**Statement of Contribution** Antônio came up with the idea and we refined it together. I conducted all the experiments and came up with the final solution. Antônio and I wrote the paper with feedback from Thomas and Niklas throughout the entire process.

**Paper III**

**No Double Descent in Principal Component Regression: A High-Dimensional Analysis.** *Daniel Gedon, Antônio H. Ribeiro, Thomas B. Schön.* Submitted, 2024.

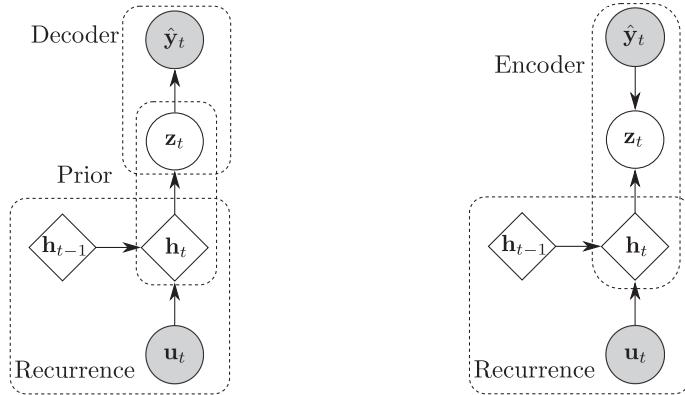


**Summary** We study the high-dimensional behaviour of principal component regression (PCR), which combines principal component analysis and linear regression, on data from a low-dimensional manifold. Specifically, we are interested in the risk when the number of samples and features becomes very large. Our analysis is based on random matrix theory. We find precise asymptotics for the risk for in-distribution data as well as for covariate shift, which we can verify in simulation. The results show that there is no interpolation peak for PCR when the number of principal components is appropriately chosen—a crucial knowledge for practitioners.

**Statement of Contribution** I came up with the basic idea, which was refined in discussions with Antônio. The theory was derived by myself with guidance, feedback and ideas from Antônio. I wrote the majority paper while Antônio helped with Section 2. Throughout the whole process, Thomas supervised the project with feedback.

**Paper IV**

**Deep State Space Models for Nonlinear System Identification.** *Daniel Gedon, Niklas Wahlström, Thomas B. Schön, Lennart Ljung.* 19th IFAC Symposium on System Identification (SYSID), IFAC-PapersOnLine, vol. 54(7), pp. 481-486, 2021.

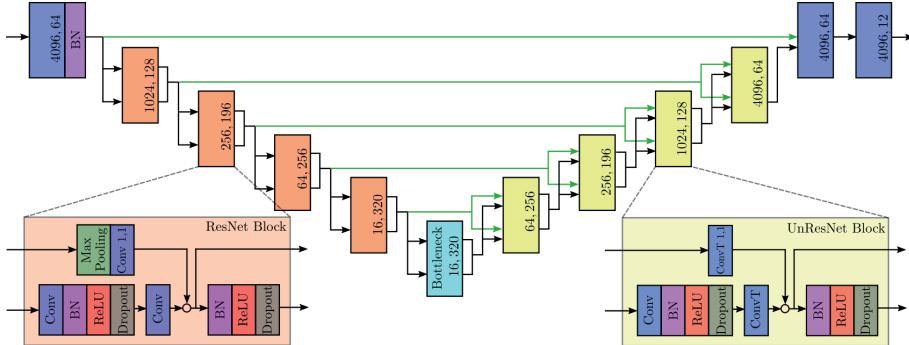


**Summary** We study the problem of learning nonlinear state space models based on observed input-output data from dynamical systems. We consider deep state-space models which utilise deep neural networks to learn nonlinear mappings between input and output. Specifically, these models apply recurrent neural networks with mappings to hidden states as well as variational autoencoders with low-dimensional latent variables for probabilistic predictions. We analyse six different variants of deep state space models from the deep learning literature which achieve competitive results on system identification benchmarks. Through detailed explanations of the model architecture and the learning procedure, we bridge the gap between the deep learning and system identification community and introduce a new model type for system identification.

**Statement of Contribution** I came up with the underlying idea of the paper in discussions with Niklas and Thomas with inspiration from previous papers in our group. I conducted all experiments and did the majority of the writing with detailed feedback from Niklas, Thomas and Lennart.

## Paper V

**First Steps Towards Self-Supervised Pretraining of the 12-Lead ECG.** *Daniel Gedon, Antônio H. Ribeiro, Niklas Wahlström, Thomas B. Schön.* Computing in Cardiology (CinC), 2021.

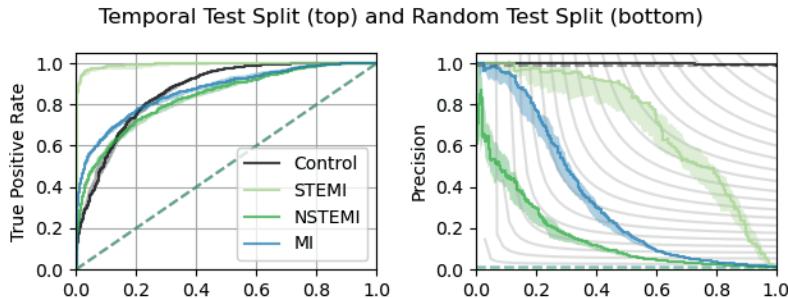


**Summary** We study one approach of representation learning for 12-lead electrocardiograms. To achieve this, we apply self-supervised learning through the reconstruction of manipulated input data. We utilise an encoder-decoder-based U-Net model structure with ResNet blocks. After self-supervised pre-training, we test the benefits of the low-dimensional pre-trained encoder representation through fine-tuning on downstream tasks with standard ECG benchmark data.

**Statement of Contribution** I came up with the idea of the paper during work on previous related papers. I conducted all experiments with the help of Antônio who prepared the data. The writing was split between Antônio and myself with feedback from Niklas and Thomas through the entire process.

## Paper VI

**Development and Validation of Deep Learning ECG-Based Prediction of Myocardial Infarction in Emergency Department Patients.** Stefan Gustafsson, Daniel Gedon, Erik Lampa, Antônio H. Ribeiro, Martin J. Holzmann, Thomas B. Schön, Johan Sundström. Scientific Reports, vol. 12, 2022.



**Summary** There are two types of myocardial infarctions: one which can easily be identified by doctors from the ECG and one where there are no known diagnosis rules from the ECG. We propose a ResNet-based model which can identify both types of myocardial infarctions from the ECG alone. This model is trained on real-world data of all-comers to hospital emergency departments and can therefore be utilised as a crucial decision-support tool for medical practitioners.

**Statement of Contribution** The idea for the paper was developed by Thomas and Johan and refined by the complete team. The data was acquired and prepared by Martin, Stefan and Antônio. Data analysis was done by Stefan and Erik. I implemented the models, conducted the experiments and created most of the figures with help from Stefan. The first draft was mainly written by me, Stefan and Johan, and all authors contributed with feedback and in the revisions. Throughout the project, Thomas and Johan supervised the progress.

## Other work

In addition to the six included papers, the research described in the following papers (to which I have contributed) is also relevant to this dissertation:

**Automatic 12-lead ECG Classification Using a Convolutional Network Ensemble.** *Antônio H. Ribeiro, Daniel Gedon, Daniel Martins Teixeira, Manoel H. Ribeiro, Antonio L. Pinho Ribeiro, Thomas B. Schön, Wagner Meira Jr.* Computing in Cardiology (CinC), 2020.

**Screening for Chagas Disease from the Electrocardiogram Using a Deep Neural Network.** *Carl Jidling, Daniel Gedon, Thomas B. Schön, Claudia Di Lorenzo Oliveira, Clareci Silva Cardoso, Ariela Mota Ferreira, Luana Giatti, Sandhi Maria Barreto, Ester C. Sabino, Antonio L. P. Ribeiro, Antônio H. Ribeiro.* PLOS Neglected Tropical Diseases, vol. 17(7), e0011118, 2023.

**End-to-End Risk Prediction of Atrial Fibrillation from the 12-Lead ECG by Deep Neural Networks.** *Theogene Habineza, Antônio H. Ribeiro, Daniel Gedon, Joachim A. Behar, Antonio Luiz P. Ribeiro, Thomas B. Schön.* Journal of Electrocardiology, vol. 81, pp. 193–200, 2023.

**ECG-Based Electrolyte Prediction: Evaluating Regression and Probabilistic Methods.** *Philipp Von Bachmann, Daniel Gedon, Fredrik K. Gustafsson, Antônio H. Ribeiro, Erik Lampa, Stefan Gustafsson, Johan Sundström, Thomas B. Schön.* Submitted, 2024.

**Deep Networks for System Identification: A Survey.** *Gianluigi Pillonetto, Aleksandr Aravkin, Daniel Gedon, Lennart Ljung, Antônio H. Ribeiro, Thomas B. Schön.* Provisionally accepted at Automatica, 2024.

Furthermore, the following papers (to which I have contributed) are peer-reviewed and accepted but not published workshop papers:

**ResNet-based ECG Diagnosis of Myocardial Infarction in the Emergency Department.** *Daniel Gedon, Stefan Gustafsson, Erik Lampa, Antônio H. Ribeiro, Martin J. Holzmann, Thomas B. Schön, Johan Sundström.* Machine learning from ground truth: New medical imaging datasets for unsolved medical problems Workshop at NeurIPS, 2021.

**On Feature Learning of Recursive Feature Machines and Automatic Relevance Determination.** *Daniel Gedon, Amirhesam Abedsoltan, Thomas B. Schön, Mikhail Belkin.* UniReps: the First Workshop on Unifying Representations in Neural Models Workshop at NeurIPS, 2023.

# CHAPTER 2

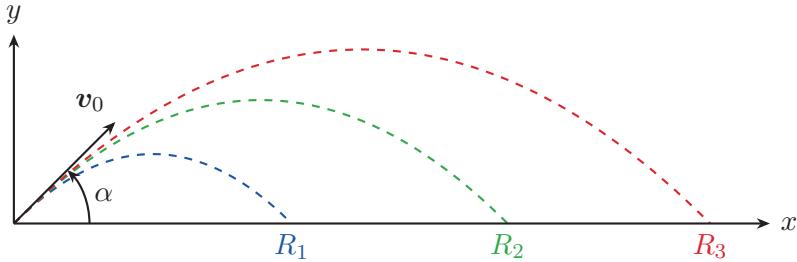
## Modelling and learning

The two core concepts in this dissertation are (a) the use of machine learning and (b) low-dimensional representation. This chapter provides the necessary background information for (a). Specifically, we introduce the models and methods that are used in Part B, where the scientific papers are included.

Let us briefly connect the papers in Part B to the models that we introduce in each section of this chapter. We start with a basic introduction of why learning-based modelling might be necessary for some situations in Section 2.1. In Section 2.2 we introduce modelling with deep neural networks in general without going into specific design choices. These models are relevant for Paper IV, V and VI [Ged+21b; Ged+21a; Gus+22]. As a different modelling idea, we move on to kernel machines in Section 2.3 which are necessary to understand Paper I and II [Ged+24; Ged+23]. In Section 2.4 we extend both deep neural networks and kernel machines to probabilistic settings that can provide uncertainty estimates which we utilise in Paper I and IV [Ged+24; Ged+21b]. Finally, we give an outlook on how deep learning approaches and kernel machines can be combined. This is an idea connected to Paper I [Ged+24].

### 2.1 Motivation for learning

As we argued in Chapter 1, describing systems through equations and models has been at the centre of understanding the world around us. At the core has been mechanistic modelling which is based on our understanding of the components of the systems using principles from physics, chemistry, mathematics or other disciplines. Mechanistic models often follow Occam’s razor [Sob15] or the principle of parsimony [Sob81] stating that the smallest set of assumptions should be used and, therefore, the model with the most simple structure where unnecessary parts are removed, is the most suitable. These models further follow Popper’s falsification principle [Pop59] in that they can be falsified through empirical testing. While mechanistic models are powerful, we demonstrate their limitations and a solution by data-driven learning-based modelling for more complex phenomena. Therefore, two problems for prediction tasks



**Figure 2.1:** Trajectory motion of an object with initial velocity  $v_0$ , launch angle  $\alpha$  and the range of its landing  $R$ .

are introduced and we derive how we decide to model them. Afterwards, we will be going into the types of machine learning models that are used within this dissertation.

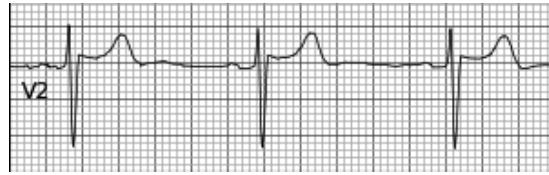
**Trajectory motion.** Consider the problem of calculating the trajectory of an object<sup>1</sup>. We assume that the ground is flat and that we know the initial velocity  $v_0$  and the launch angle  $\alpha$ . Our goal is to find out how far a certain object flies, that is, the horizontal distance  $R$  in Figure 2.1. This means, that we want to obtain a model  $f : (v_0, \alpha) \mapsto R$ . With these assumptions, we can use physical principles of kinematics, as well as mathematical knowledge of trigonometry to analyse horizontal and vertical components. This allows us to write down the equations for the trajectory and find the model  $f$  as the following where  $g$  is the gravitational constant

$$R = f(v_0, \alpha) = \frac{v_0^2 \sin 2\alpha}{g}. \quad (2.1)$$

This model allows us to predict the range of objects with great precision. It follows Occam's razor or the principle of parsimony due to its simple structure. Furthermore, the model can be falsified through experimental data when we collect a set of  $n$  examples  $\{(v_{0,i}, \alpha_i, R_i)\}_{i=1}^n$ . However, we have to be aware that this model is based on simplifying assumptions, which we have not mentioned explicitly. The strongest is that  $f$  neglects air resistance. This includes the assumption that we have an idealised point-like object without aerodynamic effects. Therefore, this idealised model will fail to provide accurate predictions in many real-world scenarios that do not follow these simplifying assumptions. When we want to remove these assumptions and include, for example, aerodynamic information in mechanistic models, modelling becomes increasingly complex. One possible solution might be to utilise data-driven learning-based modelling where we can include more complex behaviour into the model.

---

<sup>1</sup>Often also considered as projectile motion.



**Figure 2.2:** One trace of an electrocardiogram.

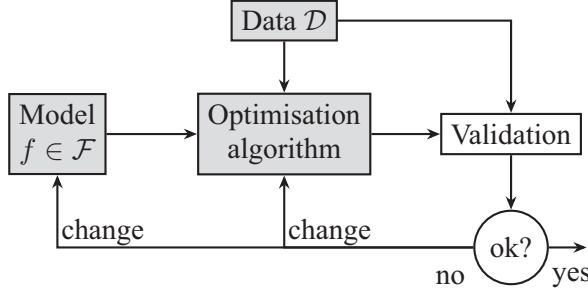
**Electrocardiogram classification.** One of the applications within this dissertation concerns the modelling of electrocardiogram (ECG) data, which we will further investigate in Chapter 6. An ECG is a recording of the temporal activity of the heart over time, see Figure 2.2. It allows physicians to analyse the functioning of the cardiovascular system to diagnose potential problems. For diagnosis, physicians rely on segmenting the ECG into distinct, repeating segments and relating them to each other. Then, they follow guidelines and rules as well as their knowledge of the cardiovascular system and the patient's history to decide on a diagnosis.

If we want to design a model that can perform automated analysis of ECGs, we can in principle follow the same idea. However, this requires detailed cardiovascular knowledge which we might not have for certain diagnoses such as for example some specific type of heart attack, the so-called non-ST-elevation myocardial infarction. Furthermore, a rule-based approach is limited in its generality since it might have difficulty covering inter- and intra-patient variance.

The alternative is to consider the problem as a supervised machine learning problem, where we want to predict some target  $y_* \in \mathcal{Y}$  from some inputs  $\mathbf{x}_* \in \mathcal{X}$  with a model  $f : \mathcal{X} \mapsto \mathcal{Y}$ . To find the right model  $f$ , we require a data set for training consisting of  $n$  i.i.d. samples of inputs and targets  $\mathcal{D} = (\mathbf{X}, \mathbf{y}) = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$  which follow some joint distribution  $(\mathbf{x}_i, y_i) \sim p(\mathbf{x}, y)$ . For our ECG examples, the input  $\mathbf{x}_i$  would be an ECG trace from a patient and the target  $y_i$  is the diagnosis (or target) that the physician assigned to this ECG in their examination. We could add more variables to  $\mathbf{x}_i$  such as age, sex or patient history to consider this information.

How do we find the best model  $f$ ? In supervised machine learning, we turn towards optimisation to find the best model [Lin+22]. First, we define a loss function between the output of our model and the true output  $\ell(f(\mathbf{x}_i), y_i) : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$  which is some measure of discrepancy. We would like to minimise the expected loss but generally, we do not have access to the joint distribution  $p(\mathbf{x}, y)$ . Therefore, we take advantage of our collected dataset  $(\mathbf{X}, \mathbf{y})$  to numerically minimise the empirical risk  $R$  [Mur22]

$$\arg \min_{f \in \mathcal{F}} R(f) = \arg \min_{f \in \mathcal{F}} \sum_{i=1}^n \ell(f(\mathbf{x}_i), y_i). \quad (2.2)$$



**Figure 2.3:** Supervised machine learning modelling procedure as an iterative process. Adapted from [Pil+23].

We consider the best model  $f$  to lie in some defined function class  $\mathcal{F}$ . Next, we will restrict  $\mathcal{F}$  for specific types of models to find a solution to the optimisation problem. We want to emphasise the importance of the main ingredients for modelling supervised machine learning problems: (1) the model  $f \in \mathcal{F}$ , (2) the training data  $\mathcal{D}$  and (3) the optimisation algorithm, which is determined by empirical risk minimisation. Figure 2.3 shows the modelling of supervised machine learning as an iterative process to find the best model by adapting the model and optimisation algorithm to fit the data.

## 2.2 Deep neural networks

To find a solution for the optimisation problem in (2.2), we can restrict  $\mathcal{F}$  to the class of parametric functions. Thus we obtain the function  $f_{\theta} : \mathcal{X} \mapsto \mathcal{Y}$  parameterised by  $\theta \in \mathbb{R}^p$ . Therefore, empirical risk minimisation tries to find the optimal parameters

$$\theta_* = \arg \min_{\theta} R(f_{\theta}) = \arg \min_{\theta} \sum_{i=1}^n \ell(f_{\theta}(x_i), y_i). \quad (2.3)$$

Deep learning approaches define a specific parameterisation of  $f_{\theta}$  through deep neural networks. This class of models has a hierarchical structure and can be described by  $L$  individual functions, describing  $L$  layers

$$f_{\theta} = f_{\theta_L}^L \circ \cdots \circ f_{\theta_2}^2 \circ f_{\theta_1}^1. \quad (2.4)$$

The parameters are  $\theta = \{\theta_L, \dots, \theta_1\}$  and the output layer is given by  $f_{\theta_L}^L : \mathcal{Z}^L \mapsto \mathcal{Y}$  from some intermediate representation space  $\mathcal{Z}^L$  to the target space  $\mathcal{Y}$ . Similarly, intermediate layers are given by  $f_{\theta_l}^l : \mathcal{Z}^l \mapsto \mathcal{Z}^{l+1}$  where  $\mathcal{Z}^1 = \mathcal{X}$ . Note that this formulation allows for skip connections such as in U-nets [RFB15], ResNets [He+16] or transformers [Vas+17].

We can focus on multiple aspects of working with the specific model structure that deep neural networks provide. Let us discern research development according to the three main ingredients of supervised learning from Figure 2.3 by important milestones:

1. *Model structure.* The exact structure of a deep neural network  $f_\theta$  can vary and lots of research has been dedicated to this endeavour. For temporal data, there are RNNs [Elm90], and their gated version for longer memory including LSTMs [HS97] and GRUs [Cho+14] leading to the attention mechanism for more flexible routing of information [BCB15]. For data on a regular grid such as images, there is AlexNet [KSH12], VGGs [SZ15], ResNet [He+16; ZK16] up to vision transformers [Dos+21]. For generative models, research developments include GANs [Goo+20], VAEs [KW14; RMW14], normalising flows [RM15], diffusion models [Soh+15] but also auto-regressive models such as PixelCNN [Oor+16b] and transformers [Vas+17] including models such as GPT [Rad+18].
2. *Training data.* In deep learning the data is often considered pre-collected and therefore static. An exception is the field of active learning where the model actively selects the most informative data points [Set09]. Notably, progress in deep learning has been fuelled by the concept of training data augmentations, especially for image classification [SK19]. This idea has been exploited for self-supervised learning, especially contrastive methods [Che+20a; He+20]. Self-supervised learning is a specific form of pre-training on unlabelled datasets. Since we do not require labels in this context, we can use larger quantities of training data [Erh+10]. Once the model is pre-trained, we can finetune it on the desired task which is often denoted the downstream task. Similarly, semi-supervised learning utilises an additional set of unlabelled data [CSZ06]. However, not just additional, unlabelled data are useful. Instead, also labelled datasets which are related to the original task are utilised to pre-train the model before transferring those learnt representations as initialisation for the model on the original datasets, known as transfer learning [Yos+14].
3. *Optimisation algorithm.* Deep learning optimisation is mainly based on first-order methods, specifically the idea of stochastic gradient descent [RM51] adapted to deep learning [RHW86]. The most notable development is the incorporation of optimisation algorithms with adaptive learning rates such as AdaGrad [DHS11], RMSProp [TH12], Adam [KB15] to enhance convergence. While higher-order optimisation methods, e.g., using Hessian information are proposed [Mar10], they are less applied due to their computational and memory complexity. Next to direct advances in optimisation, regularisation techniques embedded into the model architecture were proposed including Dropout [Sri+14] and batch normalisation [IS15].

Within this dissertation, we are not focusing on specific advancements in the model, training data or optimisation algorithm. Rather, we are interested in the representations from an intermediate layer  $l$  defined as  $\mathbf{z}_i^l = f^l(\mathbf{z}_i^{l-1})$ . We are looking at the interaction of these ingredients with representations  $\mathbf{z}_i$ . Specifically, in Paper IV and VI [Ged+21b; Gus+22], we consider a special model structure  $f_\theta$ , and in Paper V [Ged+21a] we use a self-supervised learning technique which modifies the data  $\mathcal{D}$ . Furthermore, Paper II and III [Ged+23; GRS24] consider simplified deep neural networks, specifically with one hidden layer only.

## 2.3 Kernel machines

Deep neural networks are parametric predictive models where  $f_\theta$  maps the input space  $\mathcal{X}$  through explicit functions (its layers) to hidden representations, i.e. a feature space  $\mathcal{Z}^l$ . Finally, this feature space is mapped to the output space  $\mathcal{Y}$ . The goal is that data points, which are close in  $\mathcal{X}$  are mapped to close points in  $\mathcal{Y}$ . Meaning that similar examples  $\mathbf{x}_i$  are assigned similar targets  $y_i$ . Can we perform this mapping of similar data points differently, for example, without explicitly defining the hidden layers?

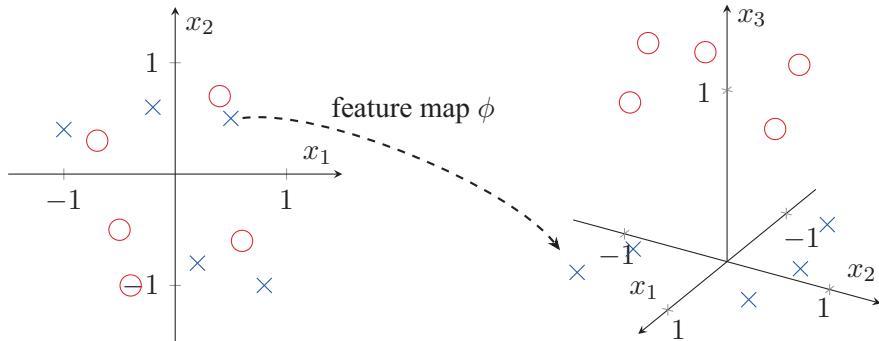
Let us define a function  $k$  that characterises the similarity of two data points  $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ . Now, we have to find a suitable similarity measure that characterises this function. Considering data points  $\mathbf{x}$  and  $\mathbf{x}'$  as vectors in the space  $\mathcal{X} = \mathbb{R}^d$ , a natural similarity measure is the inner product of the two vectors

$$k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle = \sum_{j=1}^d x_j x'_j. \quad (2.5)$$

This similarity measure has a geometric interpretation. It allows measuring the alignment or more precisely the cosine of the angle between the vectors  $\mathbf{x}$  and  $\mathbf{x}'$ , indicating their directional similarity in the  $d$ -dimensional input space  $\mathcal{X}$ . However, the inner product is a linear similarity measure which limits its applicability in many cases. We want to be able to deal with nonlinear models, which require a nonlinear similarity measure. Therefore, we can define a feature map  $\phi$  that can transform the input space into some dot product space  $\phi : \mathcal{X} \mapsto \mathcal{H}$ . Thus, we obtain

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle. \quad (2.6)$$

This formulation of  $k$  allows us to design a large set of possible similarity measures for the input space. The function  $k : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$  is also called a kernel function [SC08]. To define a predictive model, we require the kernel



**Figure 2.4:** Example of applying feature maps  $\phi$  to a 2D dataset with two classes.

to be positive semi-definite symmetric [Aro50]. Then, we obtain the kernel machine as a linear combination of kernel functions applied to the input data [SS02]

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i k(\mathbf{x}, \mathbf{x}_i). \quad (2.7)$$

Now the question becomes which feature map  $\phi$  we should apply. Figure 2.4 gives one example with a binary problem. The feature map  $\phi$  maps all points of the ‘red circle’ class in the  $x_3$ -dimension to larger values than points of the ‘blue cross’ class. This allows a linear classifier to separate both classes perfectly with a plane at  $x_3 = \text{const}$ . The problem is that the space  $\mathcal{H}$  that we should map into is not always clearly defined and we may want to use very high-dimensional feature spaces to handle complex similarity patterns in the data. In turn, this will lead to high computational complexity in computing the feature vectors  $\phi(\mathbf{x})$ .

The so-called ‘kernel trick’ alleviates this problem by implicitly operating in high-dimensional feature space without explicitly computing the feature map  $\phi$ . Instead of defining the feature map, we can directly define the kernel function  $k$ . For this reason, let us define some kernel functions used within this dissertation.

**Constant kernel.** One of the most simple kernels given by a constant  $c \geq 0$

$$k(\mathbf{x}, \mathbf{x}') := c. \quad (2.8)$$

Thus, the feature map is also constant. More specifically, we have  $\phi = \sqrt{c}$ .

**Linear kernel.** We already mentioned this kernel in the motivation of this section. Here, the feature map  $\phi$  is the identity function. Therefore, the kernel is given by the inner product and describes a linear transformation

$$k(\mathbf{x}, \mathbf{x}') := \langle \mathbf{x}, \mathbf{x}' \rangle. \quad (2.9)$$

**Gaussian kernel.** This kernel is also often referred to as radial basis function (RBF) kernel and is one of the most used kernel functions. It is defined by the following squared exponential function

$$k(\mathbf{x}, \mathbf{x}') := \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2}\right), \quad (2.10)$$

with the bandwidth parameter  $\ell > 0$  as hyperparameter. Reducing  $\ell \rightarrow 0$  decreases the similarity of the inputs  $\mathbf{x}$  and  $\mathbf{x}'$  as measured in the kernel features space  $\mathcal{H}$ . Increasing  $\ell \rightarrow \infty$  also increases the similarity of  $\mathbf{x}$  and  $\mathbf{x}'$ , leading to the constant kernel in the limit. For this kernel, the feature map  $\phi$  transforms the input into an infinite dimensional space<sup>2</sup>. Explicitly computing the mapping therefore becomes computationally impossible. For this reason, we introduced the kernel trick to directly compute the kernel without explicit computation of the feature maps. This direct computation is given by the RBF kernel function in (2.10).

**Laplace kernel.** A less popular kernel than the RBF kernel, but highly efficient in many practical applications due to its sparsity-inducing property is the Laplace kernel defined with the bandwidth  $\ell$  as

$$k(\mathbf{x}, \mathbf{x}') := \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|}{\ell}\right). \quad (2.11)$$

Once we choose a kernel and its hyperparameters for the problem at hand, we want to find the optimal prediction function (2.7). The representer theorem [KW70] states that there is a unique solution of the form in (2.7) to the infinite-dimensional optimisation problem given by

$$\arg \min_{f \in \mathcal{H}} \sum_{i=1}^n (f(\mathbf{x}_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}}^2. \quad (2.12)$$

$\mathcal{H}$  is the reproducing kernel Hilbert space corresponding to  $k$ . Finally, we have to find the prediction function weights  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)$  in (2.7) as the solution to the linear system

$$\mathbf{y} = (k(\mathbf{X}, \mathbf{X}) + \lambda \mathbf{I}_n) \boldsymbol{\alpha}. \quad (2.13)$$

As we discussed, deep neural networks have the explicit feature representation as the output of each layer. In contrast, for kernel machines we defined the feature map  $\phi$  to lift the original input space  $\mathcal{X}$  to  $\mathcal{H}$ . However, with the kernel trick, we avoid computing this feature space explicitly. The implicit feature space dimension is infinite for common kernels such as the RBF or Laplace

---

<sup>2</sup>To see this, note that we can approximate the squared exponential kernel with an infinitely long Tailor series to obtain  $\phi$ .

kernel. This allows kernel machines to operate in high-dimensional spaces without explicitly computing the feature mappings. Yet, this complicates the analysis when we want to identify what features the function  $f$  has learnt through the implicit mapping with  $\phi : \mathcal{X} \mapsto \mathcal{H}$ . In Section 2.5 we will define one specific kernel function that alleviates the problem.

## 2.4 Probabilistic modelling

So far, we have looked at predictive models  $f : \mathcal{X} \mapsto \mathcal{Y}$  with deep neural networks as parametric models and kernel machines as non-parametric models. Equivalently we can write the predictive model in terms of the conditional target distribution which is given by its conditional expectation  $f(\mathbf{x}) = \mathbb{E}[y|\mathbf{x}]$ . However, in many applications, the restriction to prediction itself is insufficient and we are asked to provide uncertainty quantification for the respective prediction.

For classification problems, i.e., where  $\mathcal{Y} = \{0, 1, \dots, C\}$  is the set of  $C$  class labels there is a natural probabilistic formulation. A predictive classifier  $f$  usually predicts class probabilities. In the case of deep neural networks, the class probabilities are provided by applying a softmax activation layer  $\sigma : \mathbb{R}^C \mapsto (0, 1)^C$ . We can interpret the output of the softmax activation as a probabilistic model. In classification, we are often interested in model calibration [Guo+17; Vai+19]. This means that the predicted probabilities reflect the true class likelihood. For example, if the model predicts a certain class with 42 % probability, it should be correct for 42 % of the time. See the dissertation of Widmann [Wid23] for an in-depth overview of calibration.

In contrast for regression problems, i.e., where  $\mathcal{Y} = \mathbb{R}^C$  is a continuous target space, when we design a point predictor, we explicitly model  $f(\mathbf{x}) = \mathbb{E}[y|\mathbf{x}]$  which does not allow for direct probabilistic extension or interpretation of the predictor  $f$ . Instead, we have to expand point predictions with a predictive distribution  $p(f(\mathbf{x})|\mathbf{x})$ . In the following, we will focus on regression and explain how we can obtain such a predictive distribution for deep neural networks and kernel machines.

### Deep probabilistic regression

For deep neural networks, we defined the optimisation problem in (2.3) by minimising the empirical risk, which is the loss of all  $n$  data points. The choice of loss function  $\ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$  implicitly defines the model predictive distribution  $p(y|\mathbf{x}; \boldsymbol{\theta})$  that we learn. This allows a probabilistic perspective. We can view minimising the loss function  $\ell$  as minimising the negative

log-likelihood

$$R(f_{\theta}) = \sum_{i=1}^n -\log p(y_i | \mathbf{x}_i; \theta), \quad (2.14)$$

for a specific model  $p(y|\mathbf{x}; \theta)$  of the conditional target distribution  $p(y|\mathbf{x})$ <sup>3</sup>. Thus choosing the L2 loss, we implicitly assume a Gaussian distribution for our model  $p(y|\mathbf{x}; \theta) = \mathcal{N}(y; f_{\theta}(\mathbf{x}), \sigma^2 \mathbf{I}_k)$  with a fixed variance  $\sigma^2$ . This implies that our model learns the mean of a Gaussian distribution. Similarly, when choosing the L1 loss, we implicitly assume a Laplace distribution for our model with a fixed scale.

**Probabilistic regression.** The simplest approach to utilise the probabilistic perspective and enable neural networks to estimate uncertainty is to change the output space of the model to  $f_{\theta} : \mathcal{X} \mapsto \mathcal{O}$  [Gal16; KG17; LPB17]. This allows the model to predict not just the mean of e.g. a Gaussian distribution but the parameters  $\varphi$  of a parameterised family of a probability distribution  $p(y; \varphi)$ . Hence, our model becomes  $p(y|\mathbf{x}; \theta) = p(y; \varphi_{\theta}(\mathbf{x}))$ . If we assume, for example, that the conditional target distribution  $p(y|\mathbf{x})$  follows a unimodal Gaussian distribution, we can adapt our model to output the mean  $\mu$  and variance  $\sigma^2$  of a Gaussian distribution. Specifically, our model becomes

$$p(y|\mathbf{x}, \theta) = \mathcal{N}(y; \mu_{\theta}(\mathbf{x}), \sigma_{\theta}^2(\mathbf{x})), \quad (2.15)$$

where the model outputs  $[\mu_{\theta}(\mathbf{x}) \ \sigma_{\theta}^2(\mathbf{x})]^{\top} = f_{\theta}(\mathbf{x})$ . To train this model, minimising the negative log-likelihood amounts to minimising the following empirical risk

$$R(f_{\theta}) = \frac{1}{2} \sum_{i=1}^n \log \sigma_{\theta}^2(\mathbf{x}_i) + \frac{(y_i - \mu_{\theta}(\mathbf{x}_i))^2}{\sigma_{\theta}^2(\mathbf{x}_i)}. \quad (2.16)$$

If we assume that  $p(y|\mathbf{x})$  follows a different distribution than a unimodal Gaussian, we can adapt our model. For example, we can change the model to output a multimodal distribution to allow for more flexible probability distributions.

**Bayesian neural networks.** Whereas probabilistic regression relies on a standard neural network with modified output space, Bayesian neural networks treat the network parameters as random variables according to some probability distribution [Mac92; Nea96]. This perspective allows us to incorporate Bayesian principles into neural networks.

We treat the distributions of the parameters  $p(\theta)$  as prior distributions. During training, we want to estimate the posterior distribution given training data

---

<sup>3</sup>While we have a double use of the notation  $p$  for both the model and the data distribution, we believe that it is clear from context to what we refer.

$p(\boldsymbol{\theta}|\mathcal{D})$ . This allows us to obtain a predictive distribution by

$$p(y|\mathbf{x}) = \int_{\boldsymbol{\theta}} p(y|\mathbf{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}. \quad (2.17)$$

We can rephrase the integral as the expectation of the fixed parameter neural network under the posterior distribution  $\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[p(y|\mathbf{x}, \boldsymbol{\theta})]$ .

However, the posterior  $p(\boldsymbol{\theta}|\mathcal{D})$  is not tractable in general. Thus, we have to find suitable approximate inference methods. Suggested solutions rely on variational inference [HV93; Gra11; Blu+15], Markov Chain Monte Carlo [WT11] or MC dropout [GG16]. While the former ones are computationally expensive, MC dropout as the cheaper method has proven less reliable [GDS20; Ova+19].

**Deep ensembles.** Probabilistic regression relies on one neural network with fixed parameters  $\boldsymbol{\theta}$  to predict a parametric predictive distribution  $p(y|\mathbf{x}, \boldsymbol{\theta})$ . However, even if we consider  $p(y|\mathbf{x}, \boldsymbol{\theta})$  as a multimodal distribution, most parameters  $\boldsymbol{\theta}$  are shared within the neural network. Similarly, we are often limited to Gaussian distributions in a Bayesian neural network to solve for the posterior distribution tractably. The question is whether we can obtain a model that samples from the true posterior distribution.

One solution can be to use deep ensembles [LPB17]. Here, we train multiple deep neural networks from scratch as standard predictors  $f_{\boldsymbol{\theta}}$ . Therefore, we obtain multiple sets of parameters  $\boldsymbol{\theta}$  that yield multiple predictions. Combining these predictions, we can form a predictive distribution  $p(y|\mathbf{x}, \boldsymbol{\theta})$ . This is generally done by averaging the prediction for the mean  $\mu$  and computing the variance  $\sigma^2$ . In a Bayesian view, we can interpret deep ensemble as sampling from the true posterior distribution  $p(\boldsymbol{\theta}|\mathcal{D})$ . Since the parameters are not tied as in probabilistic regression, there is higher variance within the predictions that possibly cover more modes of the true distribution.

**Other approaches.** Next to the approaches we described above there exist many more ideas for uncertainty quantification in deep learning. One approach is to discretise the regression problem into multiple bins and to treat it as a classification problem [CMR20; DM19]. Other uncertainty quantification methods include stochastic weight averaging with its extension to estimate the covariance of the weights for a Gaussian predictive distribution [Izm+18; Mad+19]. Furthermore, Laplace approximations with Kronecker factorisation can be utilised as an effective posthoc method for trained neural networks [RBB18; Dax+21]. In a different line of work, energy-based models provide a flexible approach for modelling the predictive distribution [LeC+06; LeC+07; Gus+20].

## Gaussian processes

We have established how to obtain uncertainty quantification for deep neural networks. Can we also equip kernel machines with this capability? The Bayesian framework of Gaussian processes (GP) allows us to do that efficiently [RW06]. Due to its Bayesian nature, it allows the incorporation of prior knowledge by choosing the mean and covariance functions.

As the name suggests, a GP is defined by a Gaussian distribution. Therefore, it is fully characterised by its mean and covariance function. The GP thus extends kernel machines as point predictors with uncertainty quantification given by a covariance function. Note that the covariance function measures the similarity of inputs with the same properties as the kernel function. We can utilise kernel functions with their ability to capture nonlinear, complex patterns in the data.

With a mean function  $m$  and a kernel  $k$ , the GP is defined as a distribution over functions  $f \sim \mathcal{GP}(m, k)$ . Thus we have

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \quad (2.18)$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \quad (2.19)$$

The mean function  $m$  and the kernel  $k$  are considered as priors within the Bayesian framework and allow us to incorporate knowledge about the system, which we model. For the mean function  $m$ , we can include deterministic trends in the data, e.g. in the case of linear trends by specifying  $m(\mathbf{x}) = \mathbf{a}^\top \mathbf{x} + b$ . Often we consider such trends to be removed in pre-processing steps and therefore set the mean function to be constant  $m(\mathbf{x}) = c$ . For the kernel function  $k$ , prior knowledge can be e.g. smoothness assumption about the system. This often leads to the RBF kernel from (2.10) or the Laplace kernel (2.11). Assuming the function to be approximately linear, a linear kernel such as (2.9) can be chosen. If we have periodicity in the data, we can encode that in the model by choosing  $k$  to be a periodic kernel<sup>4</sup>.

To predict for a test input  $\mathbf{x}_*$ , we have to find the posterior predictive distribution, which is defined as the following Gaussian distribution

$$p(f(\mathbf{x}_*) | \mathbf{x}_*, \mathcal{D}) = \mathcal{N}(\mu(\mathbf{x}_*), \Sigma(\mathbf{x}_*)). \quad (2.20)$$

The posterior predictive mean function  $\mu$  is given by (2.7). For a specific test input  $\mathbf{x}_*$ , we have to compare the test data  $\mathbf{x}_*$  with the training data  $\mathbf{X}$ . This is, where the similarity measure defined through the kernel function  $k$  comes into play. We obtain<sup>5</sup>

$$\mu(\mathbf{x}_*) = k_*^\top (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}. \quad (2.21)$$

<sup>4</sup>Also called the exponential-sine-squared kernel given by  $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{2 \sin^2(\pi d(\mathbf{x}, \mathbf{x}')/\ell)}{\ell^2}\right)$  with length scale  $\ell > 0$ , periodicity parameter  $p > 0$  and Euclidean distance  $d(\cdot, \cdot)$ .

<sup>5</sup>For notational simplicity we assume the mean function  $m$  to be zero here and in the following. For non-zero mean functions we would obtain  $\mu(\mathbf{x}_*) = m(\mathbf{x}_*) + k_*^\top (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} (\mathbf{y} - m(\mathbf{X}))$

Here, we abbreviate the notation by denoting the kernel or Gram matrix  $\mathbf{K} \in \mathbb{R}^{n \times n}$  with  $K_{i,j} = k(\mathbf{X}_i, \mathbf{X}_j)$  and  $k_* = k(\mathbf{X}, \mathbf{x}_*)$ . The measurement noise is assumed to be Gaussian with zero mean and variance  $\sigma^2$ . For the covariance function on a specific test input  $\mathbf{x}_*$  we get the following posterior

$$\Sigma(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - k_*^\top (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} k_*. \quad (2.22)$$

The Bayesian framework generally allows us to refine the posterior predictive function when new observations  $\mathbf{x}, y$  are available.

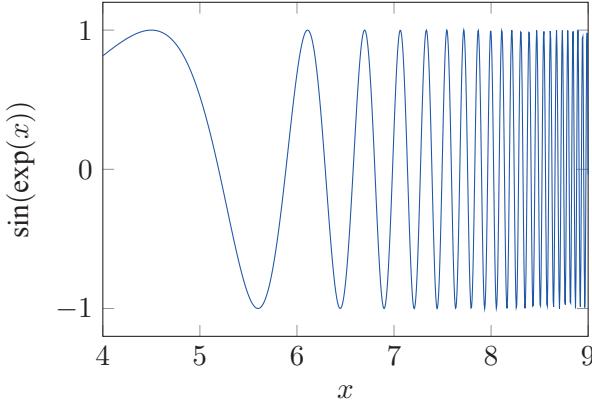
The parameters  $\boldsymbol{\theta}$  of the model include the measurement noise  $\sigma$  used in the predictive mean  $\mu(\mathbf{x}_*)$  and covariance function  $\Sigma(\mathbf{x}_*)$ , the free parameters of the chosen mean function  $m$  and kernel function  $k$ . For example, choosing a constant mean function  $m(\mathbf{x}) = c$  and a Laplace kernel (2.11) results in the parameters  $\boldsymbol{\theta} = \{c, \ell, \sigma\}$ . We can find the optimal  $\boldsymbol{\theta}$  through maximum likelihood estimation. Specifically, this implies minimising the negative log-likelihood  $-\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ , similarly as in (2.14), which yields

$$\begin{aligned} R(f) &= \sum_{i=1}^n -\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) \\ &= \frac{1}{2} \mathbf{y}^\top (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y} + \frac{1}{2} \log |\mathbf{K} + \sigma^2 \mathbf{I}|. \end{aligned} \quad (2.23)$$

Frameworks such as GPytorch [Gar+18] allow for automatic differentiation with a modular implementation of GPs in PyTorch [Pas+19]. Therefore, the optimisation problem can be efficiently solved. The computational complexity is dominated by the inverse of  $\mathbf{K} \in \mathbb{R}^{n \times n}$  which yields  $\mathcal{O}(n^3)$ . However, research has enabled kernel machines including GPs to scale to millions of data points  $n$ . This includes methods such as inducing points [QR05], the EigenPro series [MB17; MB19; ABP23] and FALCON [RCR17; Mea+20].

## 2.5 Combining deep neural networks and kernel machines

We have treated deep neural networks and kernel machines separately so far. While kernel machines are flexible, the choice of kernel has to be chosen with specific assumptions about the system such as smoothness, linearity or periodicity. This limits the generalisability of kernel-based approaches. In contrast, deep neural networks offer a more adaptive and data-driven, feature-learning approach which can capture a wide range of patterns in data. Therefore, the question arises, whether we can utilise the feature learning power of deep neural networks within kernel machines.



**Figure 2.5:** Example of composite function where standard kernels are problematic because of high-frequency information. The example is adapted from [Pil+23].

MacKay [Mac98] argues that we can either transform the input and use standard kernels or develop advanced kernels. This is reflected in the following two approaches which we will investigate in more detail. The first approach relies on using deep neural networks as feature extractors. This allows the kernel machine to operate on an often low-dimensional feature space. The second approach designs a novel feature learning, data-adaptive kernel machine called the Recursive Feature Machine (RFM). This kernel comes with a specific learning algorithm which learns features that are tightly connected to features in fully connected neural networks.

**Deep feature extraction for kernels.** The main idea here is to utilise a feature extraction function as input to the kernel. Often, we deal with high-dimensional input spaces  $\mathcal{X}$ . Therefore, a feature extractor, which helps the kernel to operate on a lower dimensional, transformed space  $\tilde{\mathcal{X}}$  can improve computational complexity. Since the kernel operates on a low-dimensional manifold, this is termed a manifold Gaussian Process [Cal+16]. Specifically, we have

$$k(\mathbf{x}, \mathbf{x}') = \tilde{k}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \tilde{k}(G(\mathbf{x}), G(\mathbf{x}')), \quad (2.24)$$

where  $G : \mathcal{X} \mapsto \tilde{\mathcal{X}}$  is the feature extractor often with  $\dim(\tilde{\mathcal{X}}) < \dim(\mathcal{X})$ . Consider the example in Figure 2.5 where the data is generated according to  $\sin(\exp(x))$  which contains high-frequency information due to the exponential component. When we want to model this function, a useful transformation is given by  $G(x) = \exp(x)$ . Thus, we obtain  $f(x) = \tilde{f} \circ G$  which will yield  $f(x) = \sin(x)$ . Hence, the transformation  $G$  removed high-frequency information and we can use a smooth GP, e.g., with an RBF kernel to model  $\tilde{f}(x)$ . However, often it is not as clear what the transformation  $G$  should look like, especially for higher-dimensional inputs. Therefore, deep kernel learning ex-

## 2.5. Combining deep neural networks and kernel machines

tends manifold GPs to utilise deep neural networks as feature extractors  $G$  [Wil+16].

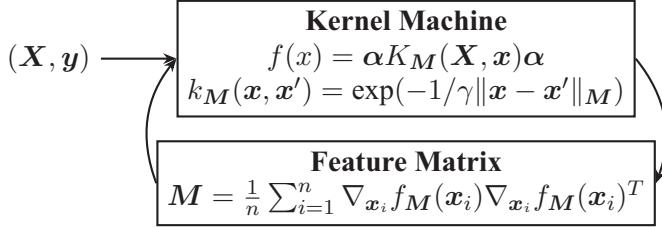
At first, the idea of using  $G$  to extract low-dimensional features from the input may seem counterintuitive. Especially, since we argued above that one benefit of kernel machines is to lift the input space to higher dimensions where the data may become linearly separable. However, the choice of kernel embeds certain assumptions on the data  $\tilde{x}$ . For example, the RBF kernel holds certain smoothness assumptions. If there is a mismatch between the data and this assumption, the kernel will not be able to completely exploit its representational power. In contrast, a feature extractor  $G$  may be able to identify nonlinear patterns in the data and transform it such that e.g. smoothness assumptions of the transformed input  $\tilde{x}$  hold. Thus the power of the kernel is fully exploited.

While this is one example of combining deep neural networks with kernels, there exist more approaches. For example, Huang et al. [Hua+15] train deep neural networks and perform Bayesian regression on the last layer, as an approximation of a GP. Similarly, Liu et al. [Liu+20] include distance awareness into neural network training and combine it with GPS to obtain a spectral-normalised GP.

**Recursive feature machine.** A second approach according to MacKay [Mac98] is to develop advanced kernel machines which can utilise the feature-extracting power of deep neural networks. For this approach, we can exploit a mechanism for feature learning in fully connected neural networks shown by Radhakrishnan et al. [Rad+24]. Consider a deep neural network  $f$  and its weight matrix  $W_l$  at layer  $l$ . Then the Gram matrix  $W_l^\top W_l$  can be considered as a feature matrix. It describes the weighting of inputs  $\tilde{x}$  to layer  $l$ . It is then identified that this Gram matrix is proportional and correlates highly with the so-called ‘average gradient outer product’ (AGOP). The AGOP is computed w.r.t. the input  $\tilde{x}$  to the layer  $l$ . We can formalise this as

$$W_l^\top W_l \propto \frac{1}{n} \sum_{i=1}^n \nabla_{\tilde{x}_i} f^l(\tilde{x}_i) \nabla_{\tilde{x}_i} f^l(\tilde{x}_i)^\top, \quad (2.25)$$

where  $\tilde{x}$  is the transformed input to layer  $l$  and  $\nabla_{\tilde{x}} f^l$  is the gradient w.r.t. that input. The AGOP can be seen as the covariance matrix of the gradients of a layer  $l$  w.r.t. its inputs. Therefore, the fully connected neural network puts higher weight on inputs with larger gradients covariance and thus considers these inputs more relevant. Additionally, this formulation has the benefit of including cross-correlation information. What (2.25) describes is a relation between the Gram matrix of a neural network layer and a specific algorithm, the AGOP. This algorithm can compute the Gram matrix from the function  $f^l$ , without access to the weights themselves. This intuition is phrased as the ‘Neural Feature Ansatz’.



**Figure 2.6:** Visualisation of the RFM iteration to learn  $\alpha$  and  $\mathbf{M}$ .

As a next step, we can exploit the Neural Feature Ansatz to design a new model that can learn features. We do so by utilising the AGOP as an algorithm which can efficiently compute a feature matrix. Specifically, we will exploit the AGOP algorithm in a kernel-based method called the Recursive Feature Machine, sketched in Figure 2.6. Let us define a Mahalanobis-distance-based Laplace kernel

$$k_M(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_M^2}{\ell}\right), \quad (2.26)$$

with  $\mathbf{M}$  as positive semi-definite, symmetric matrix such that  $\|\mathbf{x} - \mathbf{x}'\|_M^2 = (\mathbf{x} - \mathbf{x}')^\top \mathbf{M} (\mathbf{x} - \mathbf{x}')$  denotes the Mahalanobis distance. This matrix can be considered a feature matrix similar to the Gram matrix  $\mathbf{W}^\top \mathbf{W}$  from neural networks. In the RFM, we try to find this feature matrix  $\mathbf{M}$  and the kernel weights  $\alpha$  iteratively. Specifically, we solve for the kernel weights  $\alpha$  with a fixed  $\mathbf{M}$ . Then, we update the feature matrix  $\mathbf{M}$  with fixed kernel weights  $\alpha$ . This yields the RFM learning procedure as illustrated in Algorithm 1.

---

**Algorithm 1** Recursive feature machine, adapted from [Rad+24].

---

```

Input:  $\mathbf{X}, \mathbf{y}, k_M, T$        $\triangleright$  Data  $(\mathbf{X}, \mathbf{y}) = \mathcal{D}$ , kernel function  $k_M$ , iterations  $T$ 
1:  $\mathbf{M} \leftarrow \mathbf{I}_d$ 
2: for  $t \in T$  do
3:    $\mathbf{K}_M \leftarrow K_M(\mathbf{X}, \mathbf{X})$            $\triangleright \mathbf{K}_M = K_M(\mathbf{X}_i, \mathbf{X}_j)$ 
4:    $\alpha \leftarrow \mathbf{y} \mathbf{K}_M^{-1}$             $\triangleright$  Thus:  $f(\mathbf{x}) = \alpha K_M(\mathbf{X}, \mathbf{x})$ 
5:    $M \leftarrow \frac{1}{n} \sum_{i=1}^n (\nabla_{\mathbf{x}_i} f(\mathbf{x}_i)) (\nabla_{\mathbf{x}_i} f(\mathbf{x}_i))^\top$   $\triangleright$  Average Gradient Outer Product
return  $\alpha, \mathbf{M}$ 

```

---

In Paper I [Ged+24], we utilise the RFM. Since the RFM as a kernel machine is only suitable for point predictions, we embed it in the framework of GP to enable uncertainty quantification with RFMs. Furthermore, for linear RFMs, we can provably learn low-dimensional features in  $\mathbf{M}$  [RBD24].

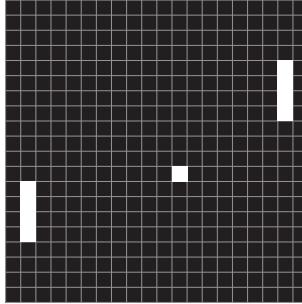
# CHAPTER 3

## Learning low-dimensional representations

So far, we first established that many real-world observations exhibit some low-dimensional structure or can at least be well approximated in lower dimensions. The machine learning approaches which we described in Chapter 2 are general and do not include specific structures in the model. In this chapter, we will expand the set of machine learning techniques to also include those that can explicitly learn low-dimensional structure.

**Curse of dimensionality.** In many real-world applications, we deal with data which is high-dimensional. Specifically, we observe some  $\mathbf{x} \in \mathcal{X}$  where in practice we have  $\mathcal{X} = \mathbb{R}^d$  with  $d$  very large. Dealing with high-dimensional spaces leads to phenomena which are generally described as the ‘curse of dimensionality’ [Bel61; Bis06]. To understand why high-dimensional spaces are a problem, consider a planar space with  $d = 2$ . Data samples  $\mathbf{x}$  are points in that space. If we restrict this 2-dimensional space to the square spanning  $[-1, 1]^2$ , then the samples can be distributed in the complete area given by  $s^2$ , with the side length  $s = 2$ . When we lift the space  $\mathcal{X}$  by one dimension to  $d = 3$ , and stay within the cube  $[-1, 1]^3$ , the space where samples are distributed expands to the volume of the cube given by  $2^3$ . When we increase the dimensions  $d$  of  $\mathcal{X}$  even further, we deal with the volume of hypercubes given by  $2^d$ . As we can see, with increasing dimension  $d$ , we obtain exponential growths in the volume. This exponentially increasing volume is the space where data samples are distributed.

When we want to learn a model  $f$ , we train with samples from the underlying data distribution  $(\mathbf{x}_i, y_i) \sim p(\mathbf{x}, y)$ . With a fixed number of samples  $n$ , the coverage of the space in which the samples lie becomes more and more sparse with higher dimensionality  $d$ . This sparsity of samples leads to difficulties for the model  $f$  to learn the underlying data distribution  $p(\mathbf{x}, y)$  and limits its applicability to new data. In other words, the model  $f$  will face difficulties when generalising beyond the training data.

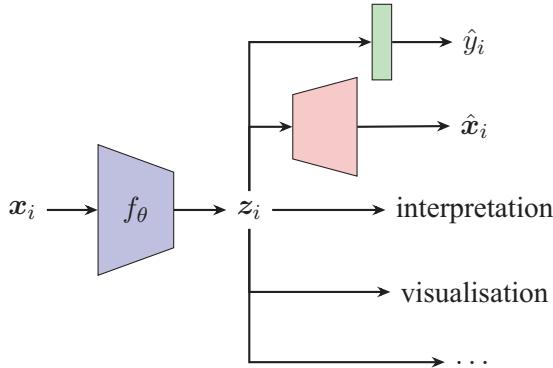


**Figure 3.1:** Digital version of the Atari game pong with two players visualised by paddles and one ball moving from left to right.

**A solution.** Despite the problems of learning good models due to the curse of dimensionality, a remedy is provided by recognising that most real-world data inhibit a low-dimensional structure. This means the data  $\mathbf{X}$  is low-rank in the linear case or more generally that the data lies on a lower-dimensional manifold.

For example, consider the computer game of pong visualised in Figure 3.1. In this game, two players control one paddle on each side of the screen by moving it up or down. The goal is for one player to use their paddle to bounce the ball behind the opponent's paddle to score a point. For a computer to control the paddle, an important piece of information is the position of the ball. Consider that a computer receives the image of the game that we see on the screen such as in Figure 3.1. In the example figure, the image has a size of  $20 \times 20$  pixels. This means our observations  $\mathbf{x}$  lie in a  $d = 20 \times 20 = 400$  dimensional space. In contrast, the information that we are interested in, the position of the ball, can be exactly described by only two dimensions: the vertical and horizontal position in pixel space, e.g. in our example, this would be  $\mathbf{z} = [11, 8]^\top$ . Therefore, the data lies on a 2-dimensional manifold  $\mathcal{Z} = \mathbb{R}^2$  on which our model could operate.

**Goals of learning low-dimensional representations.** We obtain a range of benefits when we explicitly incorporate structure into the model or learning algorithms that yield low-dimensional representations. For example, it can help overcome the curse of dimensionality. More generally, we hope to learn useful low-dimensional representations of the data [BCV13]. The question then arises of what a ‘useful’ representation is. Once we have some representations of the data, we want to perform some tasks. A representation that simplifies that task or improves the performance of the task can therefore be considered as useful. We list some tasks in Figure 3.2. Such a task can for example be classification for which we want to have separable representations to help predict  $\hat{y}_i$ . Alternatively, the task can be de-noising for which we want to discard noise components in the representation before reconstructing the input  $\hat{\mathbf{x}}_i$ . Yet



**Figure 3.2:** Possible tasks of representation learning.

another alternative, as in the case of the pong example in Figure 3.1, is that the low-dimensional representation can be interpretable or that it allows for visualisation of the data.

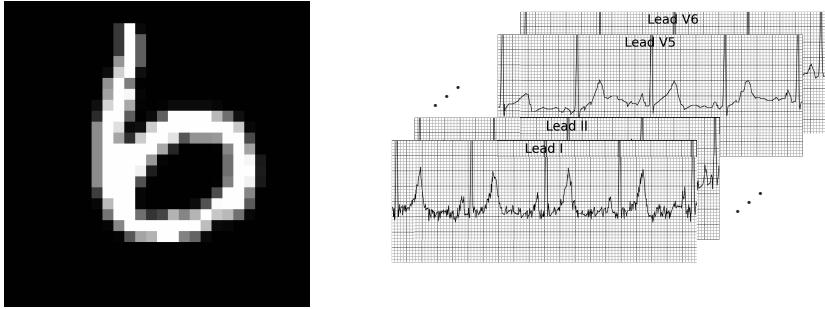
### 3.1 Supervised learning

In supervised machine learning, we have access to both inputs  $x_i$  and outputs  $y_i$ . The optimal case for supervised learning of low-dimensional representations would be if we had access to ground-truth low-dimensional representation  $z_i \in \mathcal{Z}$ . This allows us to directly train models to learn this representation as  $f : \mathcal{X} \mapsto \mathcal{Z}$ . In the example of the game Pong from Figure 3.1, this would mean that  $x$  is the image of the screen and  $z$  is a vector containing the vertical and horizontal position of the ball. This enables the use of  $z$  as a supervision signal when training the model  $f$ .

Now consider the example in Figure 3.3. The left panel shows a grayscale image of a handwritten digit of size  $28 \times 28$  from the MNIST<sup>1</sup> dataset [LC10] and the right panel shows a 12-lead 10 seconds ECG exam from the PTB-XL<sup>2</sup> database [Wag+20] sampled to have a size of  $12 \times 4096$ . When we want to follow the Pong example, we have to identify the underlying low-dimensional representations  $z$  from these data examples to use as a supervision signal. However, it is unclear what  $z$  should be for such real-world examples. Even trying to identify the dimension of  $z$  is a difficult problem where some attempts were made on simple datasets, including MNIST [Li+18; Pop+21]. But having  $\dim(z)$  is insufficient as a supervision signal. Therefore, for most real-world

<sup>1</sup>Modified National Institute of Standards and Technology database.

<sup>2</sup>Physikalisch-Technische Bundesanstalt XL.



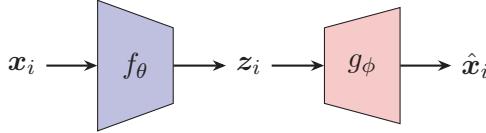
**Figure 3.3:** High-dimensional real-world data. Left: handwritten digit from MNIST. Right: ECG trace from PTB-XL.

datasets, supervised learning of low-dimensional representation is not possible.

One view of how neural networks learn lower-dimensional representations of data is the following. We established that deep neural networks are hierarchical models consisting of multiple layers, recall (2.4). When we consider image classification as for the MNIST example in Figure 3.3, we may consider convolutional neural networks. Each layer maps its input to some intermediate representation  $f^l : \mathcal{Z}^l \mapsto \mathcal{Z}^{l+1}$ . In general, we have  $\dim(\mathcal{Z}^l) \geq \dim(\mathcal{Z}^{l+1})$ , meaning that the input  $x$  is compressed as it is transformed within the network. Arguably each of these representations  $z^l$  is ‘useful’ according to our informal definition from above to solve the task of image classification. Additionally, for convolutional networks, the intermediate representations also turn out to be interpretable. The convolutional kernel, which is part of the layer  $f^l$ , in early layers converge to edge detectors [KSH12] similar to Gabor filters [Gab46]. The convolution kernels in later layers converge to higher semantic detectors for the representative task. Similarly, the features that are learnt by convolutional neural networks are visualised to analyse what representations are learnt by neural networks [SVZ14a; Yos+15; NYC16; OMS17]. We will return to the idea of analysing lower-dimensional representations of supervised neural networks in the context of medical modelling in Chapter 6.

## 3.2 Unsupervised learning

Since we do not have a supervision signal of the low-dimensional representation available, we are limited to unsupervised training using only inputs  $x_i$  without outputs  $y_i$ . In this section, we will focus on an autoencoder structure consisting of an encoder and a decoder, see Figure 3.4, similar to the denoising example in Figure 3.2. In this model, we embed the input  $x_i$  through an encoder model  $f_\theta$  into a low-dimensional representation  $z_i$ . Then a decoder



**Figure 3.4:** Encoder-decoder architecture for unsupervised representation learning.

model  $g_\phi$  decodes the representation and tries to reconstruct the original input  $\hat{x}_i$ . This encoder-decoder architecture has the advantage that we can work with unlabelled data  $x_i$ . In what follows, we will discuss unsupervised representation learning in increasing complexity beginning with linear models, advancing to nonlinear models and finally nonlinear probabilistic models.

## Linear case

In this case, both the encoder  $f_\theta$  and the decoder  $g_\phi$  are linear functions. We can view the linear case as a one-layer, linear autoencoder. We can construct such an autoencoder by Principal Component Analysis (PCA) [Pea01; Jol02]. Consider the data matrix  $\mathbf{X} \in \mathbb{R}^{n \times d}$ . The full singular value decomposition of the matrix is given by

$$\mathbf{X} = \sum_{i=1}^d s_i \mathbf{u}_i \mathbf{v}_i^\top = \mathbf{U} \mathbf{S} \mathbf{V}^\top, \quad (3.1)$$

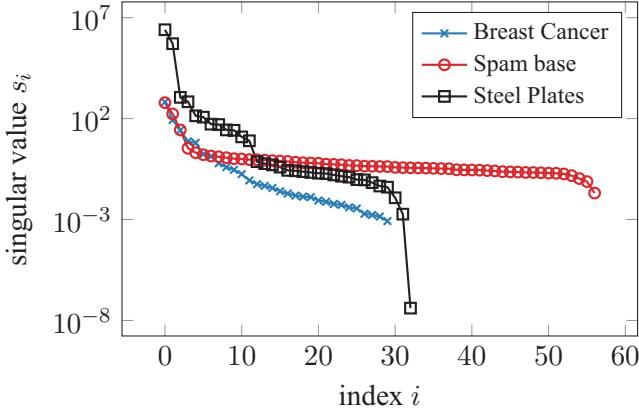
where  $\mathbf{S} = \text{diag}(s_1, \dots, s_d)$  is a diagonal matrix of value-sorted singular values,  $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_p]$  and  $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_p]$  are matrices of the left  $\mathbf{u}_i$  and right  $\mathbf{v}_i$  singular vectors, respectively. The first left singular vector  $\mathbf{u}_1$  denotes the direction of greatest variance in the dataset.

Let  $\hat{\mathbf{V}} = [\hat{\mathbf{v}}_1, \dots, \hat{\mathbf{v}}_k]$  be the estimated matrix of right sample singular vectors truncated to the first  $k \leq d$  principal components. The encoding in PCA is performed by the transformation  $f : \mathcal{X} \mapsto \mathcal{Z}$  which we can write as  $\mathbf{z}_i = \hat{\mathbf{V}}^\top \mathbf{x}_i$ . Since we have a linear encoding, the decoding can be written down in closed form by the inverse transformation  $g : \mathcal{Z} \mapsto \mathcal{X}$  written as  $\hat{\mathbf{x}}_i = \hat{\mathbf{V}} \mathbf{z}_i$ . This implies that the encoder will be  $f = \hat{\mathbf{V}}^\top$  and the decoder will be  $g = \hat{\mathbf{V}}$ . If we choose to select all principle components, i.e.  $k = d$ , then we will reconstruct the original input without loss of information since  $g \circ f = \hat{\mathbf{V}} \hat{\mathbf{V}}^\top = \mathbf{I}_d$  due to the orthonormal property of the singular vectors.

The question arises as to why the linear transformation provided by the linear decomposition in (3.1) works in practical applications which generally exhibit nonlinear patterns in the data. In Figure 3.5 we consider three real-world tabular datasets. Specifically, we have from the UCI<sup>3</sup> Machine Learning Reposi-

---

<sup>3</sup>UC Irvine.



**Figure 3.5:** Distribution of singular values in three real-world datasets.

tory [DG17] the breast cancer, the spam base datasets and the Steel-plates-fault dataset provided from Semeion<sup>4</sup>. Despite the presence of nonlinear patterns in the data, we can see the distribution of the singular values of these datasets suggests that the linear transformation provided by PCA in (3.1) captures a significant part of the information. This is evident due to the quick decay of singular values, indicating an approximate linear low-dimensional relationship.

Here, we defined PCA according to (3.1) as a fixed decomposition and used its components for our model  $f$  and  $g$ . In contrast, before, we viewed finding an optimal model as an optimisation problem, see (2.2). Similarly, we can view PCA as an optimisation problem. One approach is to optimise a linear subspace of the data [RB15]

$$\begin{aligned} \min_{\mathbf{V}, \mathbf{Z}} & \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{V}\mathbf{z}_i\|_2^2 \\ \text{s.t. } & \mathbf{V}^\top \mathbf{V} = \mathbf{I}. \end{aligned} \tag{3.2}$$

In Paper III [GRS24], we extend PCA by replacing the decoder with linear regression. Specifically, we utilise this model to analyse how well the size of  $\mathbf{z}_i$  regularises the risk on new test data, i.e. the generalisation risk.

## Nonlinear case

**Kernel PCA.** A straightforward extension allowing for more general models is to let the encoder model  $f$  and the decoder  $g$  be nonlinear. One approach

---

<sup>4</sup>This dataset is provided by Semeion, Research of Sciences of Communication, Via Sersale 117, 00128, Rome, Italy.

to achieve this is through kernel PCA [SSM97] which generalises traditional PCA. Specifically, kernel PCA applies PCA after a nonlinear transformation  $\phi : \mathcal{X} \mapsto \mathcal{H}$ . With this transformation, we can define a kernel  $k(\mathbf{x}, \mathbf{x}')$  as in (2.7). PCA is then performed in the feature space  $\mathcal{H}$ . Above in (3.1), we use the singular value decomposition to find the left singular vectors  $\mathbf{V}$ . Analogously, we can use the eigendecomposition on the sample covariance matrix  $\hat{\Sigma}$

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^n \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^\top = \sum_{j=1}^k \hat{\lambda}_j \hat{\mathbf{v}}_j \hat{\mathbf{v}}_j^\top, \quad (3.3)$$

with  $\hat{\lambda}_j$  as the empirical eigenvalues and  $\hat{\mathbf{v}}_j$  as the empirical left singular vectors. Then, we can compute a low-dimensional representation  $\mathbf{z}_i = \hat{\mathbf{V}}^\top \phi(\mathbf{x}_i)$ . Therefore, the encoder will be  $f(\cdot) = \hat{\mathbf{V}}^\top \phi(\cdot)$ . However, the model for the decoder is not a simple inverse as it was in the linear case. Multiple solutions are proposed [Bur96; Mik+98; KT03] with the most widely adopted probably being the one by Bakır et al. [BWS04]. They propose to find the nonlinear mapping  $g : \mathcal{Z} \mapsto \mathcal{X}$  by solving a supervised learning problem using the dataset  $(\mathbf{x}_i, \mathbf{z}_i)_{i=1}^n$  since we can pre-compute the low-dimensional representation  $\mathbf{z}_i$ .

In Paper II [Ged+23] we propose an alternative solution to find the decoder mapping for kernel PCA. In contrast to the supervised optimisation problem from Bakır et al. [BWS04], the reconstruction in our approach is a direct by-product and therefore without the need for optimisation. Our solution relies on the concept of random Fourier features [RR08]. The feature map  $\phi : \mathcal{X} \mapsto \mathcal{H}$  works in an infinite dimensional space which we can write as  $\phi(\mathbf{x}) = (\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \dots)$ . To work in finite dimensions, we can truncate the feature map to the first  $r$  components  $\tilde{\phi}(\mathbf{x}) = (\varphi_1(\mathbf{x}), \dots, \varphi_r(\mathbf{x}))$ . This means that the kernel  $k$  is approximated as  $k(\mathbf{x}, \mathbf{x}') \approx \langle \tilde{\phi}(\mathbf{x}) \tilde{\phi}(\mathbf{x}') \rangle$ . When we consider an RBF kernel, see (2.10), the implicit feature space is infinite. Random Fourier features enable finite approximations for translation invariant kernels, which includes the RBF kernel. Specifically, we can use the feature map

$$\tilde{\phi}(\mathbf{x}_i) = \sqrt{2} \sin(\mathbf{W} \mathbf{x}_i + \mathbf{b}), \quad (3.4)$$

with  $\mathbf{b} \sim \mathcal{U}(-\pi, \pi)$ , the elements of  $\mathbf{W}$  sampled from  $w_{i,j} \sim \sqrt{2\ell} \mathcal{N}(0, \mathbf{I})$  and  $\ell$  as the lengthscale of the original RBF kernel. For details on how we utilise random Fourier features to find the decoder function as a by-product of the encoder, see Paper II [Ged+23].

**Manifold learning.** We will briefly discuss manifold learning as an alternative approach to kernel PCA for encoding input data into low-dimensional representations. Here the goal is not reconstruction and de-noising but instead visualisation, interpretation and eventually use for downstream tasks such as classification. Standard methods include isometric mapping [TSL00] as an extension of kernel PCA which preserves intrinsic geometric distances between

the data points; local linear embeddings [RS00] assume that in a local neighbourhood, the data points lie on a linear manifold. It therefore maintains the geometric distance of data points in this local neighbourhood. Spectral embedding with Laplacian eigenmaps [BN03] utilises the eigenstructure of the graph Laplacian matrix to map the data onto a low-dimensional manifold, preserving both local and global geometric relationships. Finally, we have t-SNE<sup>5</sup> [VH08] which is highly utilised for visualisation since it allows disentangling clusters on multiple different low-dimensional manifolds.

## Nonlinear probabilistic case

**Probabilistic PCA.** A probabilistic model allows the low-dimensional latent space  $\mathcal{Z}$  to be represented with a probability distribution instead of distinct points in the non-probabilistic case. When considering probabilistic autoencoder structures, the simplest version is the probabilistic PCA [TB99]. While it is a linear model, it will serve as an entry point for more complex nonlinear model structures. In probabilistic PCA, we assume that the latent space  $\mathcal{Z}$  is normally distributed (the prior) and that we can decode the input  $\mathbf{x}$  with a linear transformation

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, \mathbf{I}_k), \quad (3.5)$$

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{W}\mathbf{z} + \mathbf{b}, \sigma^2 \mathbf{I}_d). \quad (3.6)$$

Similarly, we can find the encoding distribution as

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mathbf{M}^{-1}\mathbf{W}^\top(\mathbf{x} - \mathbf{b}), \sigma^2 \mathbf{M}^{-1}), \quad (3.7)$$

with  $\mathbf{M} = \mathbf{W}^\top\mathbf{W} + \sigma^2 \mathbf{I}$ . With the former, we have the distribution of  $\mathbf{x}$  as

$$p(\mathbf{x}; \mathbf{b}, \mathbf{W}, \sigma^2) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} = \mathcal{N}(\mathbf{x}; \mathbf{b}, \mathbf{C}), \quad (3.8)$$

with  $\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \sigma^2 \mathbf{I}$ . Since the model is linear Gaussian, we can either find a closed-form solution for the parameters, or we can minimise the negative log-likelihood w.r.t. the parameters  $\boldsymbol{\theta} = (\mathbf{b}, \mathbf{W}, \sigma^2)$

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\mathbf{x}) &= -\frac{nd}{2} \log(2\pi) - \frac{n}{2} \log |\mathbf{C}| \\ &\quad - \frac{1}{2} \sum_{i=1}^n (\mathbf{x}_i - \mathbf{b})^\top \mathbf{C}^{-1} (\mathbf{x}_i - \mathbf{b}). \end{aligned} \quad (3.9)$$

This optimisation yields the optimal parameters  $\boldsymbol{\theta}^*$ . We can thus utilise the optimised model  $p_{\boldsymbol{\theta}^*}$  as a generative model. Specifically, we can sample from

---

<sup>5</sup>T-distributed Stochastic Neighbour Embedding.

the prior distribution  $p(z)$  and use the decoder  $p(x|z)$  to generate new data samples  $x$  from the same distribution as  $p(x)$ .

**Variational Autoencoder.** Even though the probabilistic PCA is more flexible than PCA due to its probabilistic nature, it assumes a linear relationship between inputs  $x_i$  and the latent low-dimensional space  $z_i$  which is restrictive. This linearity assumption is removed by variational autoencoders (VAEs) [KW14; RMW14]. Generally, we can view the VAE as an extension of probabilistic PCA where the encoder and decoder are deep neural networks. Therefore, we have the following model

$$p(z) = \mathcal{N}(z; 0, I), \quad (3.10)$$

$$p_{\theta}(x|z) = \mathcal{N}(x; \mu_{\theta}(z), \sigma_{\theta}^2 I), \quad (3.11)$$

where the decoder  $p_{\theta}(x|z)$  is described by a mean function  $\mu_{\theta}$  and the variance  $\sigma_{\theta}^2 > 0$ . While the variance is a fixed learned value, the mean function is described by a deep neural network. The prior of the low-dimensional latent state  $p(z)$  is chosen as a standard Gaussian distribution. In contrast to probabilistic PCA where we have a closed-form solution for the posterior distribution  $p(z|x)$ , in the VAE the true posterior is intractable. Therefore, we approximate it by a parametric Gaussian distribution

$$q_{\phi}(z|x) = \mathcal{N}(z; \mu_{\phi}(x), \sigma_{\phi}^2(x)I). \quad (3.12)$$

Thus the encoder  $q_{\phi}(z|x)$  is described by a mean function  $\mu_{\phi}$  and  $\sigma_{\phi}^2$  given by deep neural networks. While this allows for more general mappings to and from the low-dimensional space  $\mathcal{Z}$ , it has the disadvantage that we no longer have closed-form solutions for the optimal parameters. Also, since the parameters  $\theta$  and  $\phi$  of the decoder and encoder are not coupled, we have to learn them jointly.

Similarly to the probabilistic PCA case, we rely on maximising the log-likelihood<sup>6</sup>. This yields

$$\log p_{\theta}(x) = \log \int p_{\theta}(x, z) dz = \log \int p_{\theta}(x|z)p(z) dz. \quad (3.13)$$

However, since we use a deep neural network to parameterize the decoder function the integral is intractable. Therefore, we rely on the framework of variational inference [BKM17; WJ+08] to solve the optimisation problem. Rewriting the log-likelihood with Jensen's inequality [Jen06] gives us the ‘Evidence Lower Bound’ (ELBO) to maximise w.r.t. the parameters  $\theta, \phi$

$$\log p_{\theta}(x) \geq \mathbb{E}_{q_{\phi}(z|x)}[\log p_{\theta}(x|z)] - D_{KL}(q_{\phi}(z|x)||p_{\theta}(z)), \quad (3.14)$$

---

<sup>6</sup>Note that so far we minimised the negative log-likelihood which is equivalent.

where  $D_{KL}$  is the Kullback-Leibler divergence. In the VAE we can first encode a sample  $\mathbf{x}_i$  with the encoder  $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$  to obtain a latent representation  $\mathbf{z}_i$ . Then we try to reconstruct the original input  $\mathbf{x}_i$  by decoding the representation  $\mathbf{z}_i$  through the decoder  $p_\theta(\mathbf{x}_i|\mathbf{z}_i)$  to obtain  $\hat{\mathbf{x}}_i$ . Thus, the first term in the loss function aims to maximise the likelihood that a reconstructed sample  $\hat{\mathbf{x}}_i$  is close to the true input sample  $\mathbf{x}_i$ . Therefore, this is often described as the reconstruction term. The second term ensures that the encoding distribution  $q_\phi(\mathbf{z}|\mathbf{x})$  is close to the specified prior distribution  $p(\mathbf{z})$ . It is therefore considered a regularisation term. However, we cannot directly compute the gradients w.r.t. the parameters in the loss (3.14). Specifically, in the reconstruction term, we have to differentiate through the sampling operation  $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ . A solution is provided by the reparametrisation trick [KW14]. For general tutorials on the VAE, see Doersch [Doe16] and Kingma and Welling [KW19]. In Chapter 5, we will utilise the VAE for dynamic systems by merging it with temporal models.

**Disentanglement.** The loss function of the VAE in (3.14) encodes a specific structure. Notably, the choice of prior  $p(\mathbf{z})$  defines what low-dimensional representations  $\mathbf{z}$  we want to learn. However, the trade-off between reconstruction loss and regulariser can limit learning specific structures in the encoder  $q_\phi(\mathbf{z}|\mathbf{x})$ . Here the idea of disentanglement comes into play<sup>7</sup>. The general assumption is that data is generated in the following [Loc+19]: There exists a small set of factors which are independent of each other. These factors generate the high-dimensional observations  $\mathbf{x}$ . Then, disentanglement aims to learn a specific structure for the low-dimensional latent representations which reflect the true set of factors that generated the observations.

The simplest approach to enhance independent latent variables is suggested by Higgins et al. [Hig+17] with the  $\beta$ -VAE. Specifically, here the effect of the regularisation term is enlarged with a hyperparameter  $\beta$

$$\max_{\theta, \phi} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})). \quad (3.15)$$

For  $\beta > 1$ , we put greater emphasis on the latent structure. More generally, we can write disentangled VAEs through a regularising term that enforces certain properties on the encoder [Loc20]

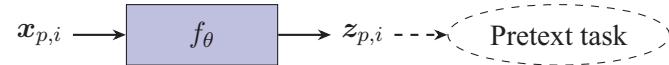
$$\begin{aligned} \max_{\theta, \phi} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) \\ + \beta R_u(q_\phi(\mathbf{z}|\mathbf{x})), \end{aligned} \quad (3.16)$$

where  $R_u$  is a regularising function. Notable works on disentanglement include Burgess et al. [Bur+17] on understanding the  $\beta$ -VAE, Kim and Mnih [KM18] introducing the FactorVAE and others [EW18; KSB18; Che+18; RM18]. Sim-

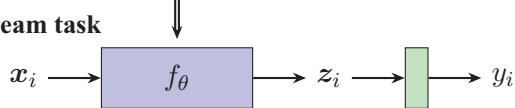
---

<sup>7</sup>This is closely connected to the line of work on independent component analysis (ICA) [HO00].

### 1. Pretraining



### 2. Downstream task



**Figure 3.6:** Two stage training procedure of using self-supervised learning.

ilar work has been published by introducing the InfoGAN [Che+16] and on the emergence of disentangled representations in neural networks [AS18].

However, Locatello et al. [Loc+19] have shown that learning structured, disentangled representations with an unsupervised approach, i.e. without an explicit supervision signal from the low-dimensional latent space, is impossible in general. To overcome this limitation, we need to include assumptions about the data or the model. Solutions to this problem are proposed which include partial information about the latent space which is obtained through different approaches [Loc+20a; Loc+20b; Dit+21; Kli+21].

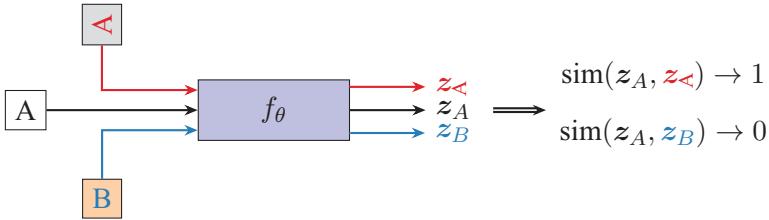
## 3.3 Self-supervised learning

We have seen that learning a low-dimensional representation in a supervised approach is not possible in general. Unsupervised learning of good representations is often limited to VAE-based models and assumptions about the data or the model have to be included to learn more structure representations. A more direct approach to obtaining structured low-dimensional representations is through self-supervised learning, which is a subset of unsupervised learning. Thus we only have access to input data  $x_i$  but not to outputs  $y_i$ . The benefit is that we can potentially use vast amounts of unlabelled data, e.g., from the internet without the explicit need for (human) annotations. The main idea is to generate pseudo-labels<sup>8</sup> from the inputs  $x_i$ . Therefore, the supervised learning problem from (2.2) can be applied where the pseudo-labels are some transformation  $T$  of the inputs

$$\arg \min_{f \in \mathcal{F}} \sum_{i=1}^n \ell(f(x_i), T(x_i)). \quad (3.17)$$

Using self-supervised learning consists of two steps, see Figure 3.6. First, the model backbone is trained with unlabelled pre-training data  $x_{p,i}$  on some form

<sup>8</sup>We adopt the term pseudo-labels from literature, while in general in this dissertation we denote  $y$  as the output instead of the label to allow for regression outcomes.



**Figure 3.7:** Conceptual idea of contrastive self-supervised learning. We have an anchor image  $A$  and an image  $B$  as well as a transformed version of  $A$ , e.g. by rotation or colour changes. The goal is to minimise the similarity of the representations  $z_A$  and  $z_B$  while maximising the similarity between  $z_A$  and of its transformed version  $z_{\tilde{A}}$ .

of pretext task. The goal is to learn ‘useful’ representations for a downstream task. Therefore, both the pre-training data and the pretext task have to be adapted to the respective goal for the representations. Second, the pre-trained model  $f_\theta$  is transferred to the downstream task and a lightweight task-specific head is added. In this stage, we train on labelled data  $\{(x_i, y_i)\}_{i=1}^n$ . Either only the task-specific head is updated with fixed backbone weights  $f_\theta$ , or the complete model is fine-tuned. While a fine-tuned model performs better in most applications, it is computationally more demanding.

Early approaches for self-supervised learning, i.e. the specific pretext tasks, in computer vision, include generative approaches such as removing parts of the image and inpainting [Pat+16], or colourisation from grayscale images [ZIE16]. Non-generative approaches include predicting rotations [GSK18], relative position [DGE15] or jigsaw puzzles [NF16; MM20] of image patches. In what follows, we will look at the two most dominant types of self-supervised learning [Liu+21], i.e. contrastive and generative learning.

## Contrastive learning

One of the main approaches of self-supervised learning is contrastive learning which is mainly applied to computer vision, see [Jai+20] for an overview. Differing from generative approaches, contrastive learning can be categorized as a discriminative approach. The general idea of contrastive approaches is outlined in Figure 3.7. Consider that we have two images,  $A$  and  $B$ . The goal is that the representations from these images  $z_A$  and  $z_B$  should be distinct from each other. However, this alone would have a trivial solution by moving all representations  $z_i$  as far away from each other as possible. To avoid this collapse, we create positive images from the anchor image  $A$ . These positive images are transformed versions of image  $A$ . The transformations include rotations, flipping, noise injection, colour changes, etc. Thus we obtain positive pairs, whose representations should be as close to each other as possible by maximis-

ing their similarity. Therefore, by defining a set of transformations  $T$  which we apply to the anchor image  $A$ , we implicitly define the transformations to which the representation and therefore also the model becomes invariant.

We have an anchor sample  $\mathbf{x}$  and its representation  $\mathbf{z} = f_{\theta}(\mathbf{x})$ . The representations of positive pairs are denoted by  $\mathbf{z}^+$  and the negative pairs as  $\mathbf{z}^-$ . We compare the representations using a noise contrastive estimation (NCE) loss [GH10]. The learning task becomes harder with more negative samples since there are more samples to compare with. Thus the model will learn to distinguish between the anchor and negative samples more effectively. Typically a modification of the standard NCE loss, known as InfoNCE is applied

$$\min_{\theta} - \log \frac{\exp(\text{sim}(\mathbf{z}, \mathbf{z}^+)/\tau)}{\exp(\text{sim}(\mathbf{z}, \mathbf{z}^+)/\tau) + \sum_{i=1}^k \exp(\text{sim}(\mathbf{z}, \mathbf{z}_i^-)/\tau)}, \quad (3.18)$$

with  $k$  negative samples,  $\tau$  as temperature parameters and  $\text{sim}()$  as a similarity measure, e.g. the cosine similarity. Prominent approaches that adapted this framework with different modifications include deep cluster [Car+18], DIM [Hje+19; BHB19], SimCLR [Che+20a; Che+20b], MoCo [He+20; Che+20c], SwAV [Car+20], BYOL [Gri+20] and Dino [Car+21]. Notably, SimCLR is often seen as a key development stage that introduced the use of a large set of data augmentations as transformations and large batch sizes such that there are many negative examples in the batch, which led to increased performance.

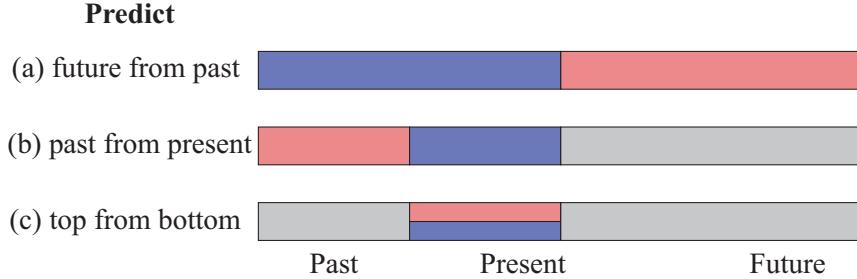
While these examples are all applied to computer vision, there are many developments outside of images. These include audio signal with contrastive predictive learning by splitting the time sequence in multiple segments [OLV18], graph contrastive learning [You+20]. Relevant for Chapter 6, also in medical applications such as ECGs there are pre-training methods based on contrastive learning [KZC21; Gop+21].

## Generative learning

Generative self-supervised learning has previously been applied to images with inpainting or colourisation [Pat+16; ZIE16]. However, in many cases the solution space, e.g. for inpainting is large and non-unique, rendering this self-supervision pretext task complicated. For other data modalities, we can benefit from the inherent structure of the data in generative approaches. For example, temporal data provides sequential dependencies<sup>9</sup> that can effectively be leveraged by generative approaches. Temporal data has a direction from past to present and future. The general idea in generative self-supervised learning on temporal data is outlined in Figure 3.8. The goal is to mask some part of the data so that the model has to predict it. This can include predicting the future

---

<sup>9</sup>These dependencies can include autocorrelation or causal influences.



**Figure 3.8:** Conceptual idea of generative self-supervised learning. Adapted from Yann LeCun’s talk at EPFL (École Polytechnique Fédérale de Lausanne) titled ‘Self-supervised learning: could machines learn like humans?’ on 05.10.2018.

from the past, the past from the present or in multi-dimensional data to predict some dimensions from others. We can also combine all three approaches to obtain more specific self-supervised learning tasks.

This approach has become dominant in the area of natural language processing (NLP). Specifically, generative self-supervised learning was applied as the so-called ‘masked language model’. Here, we consider the input sequence of tokens<sup>10</sup> and mask a certain percentage of the inputs. The influential model BERT for example randomly masks 15 % of the tokens [Dev+19]. The goal is then to predict the masked tokens. This allows the model to learn the structure of the text by capturing the contextual information of the text. The specific training objective is to minimise the negative log-likelihood of the masked tokens

$$\min_{\theta} - \log p(\mathbf{x}_{\text{masked}} | \mathbf{x}_{\text{unmasked}}). \quad (3.19)$$

If, however, the goal is to generate new text, then the auto-regressive approach of GPT<sup>11</sup> is the most suitable task [Rad+18; Rad+19; Bro+20]. Here, the model predicts the next token in the sequence conditioned on all previous tokens, i.e.

$$\min_{\theta} - \log p(\mathbf{x}_{t+1} | \mathbf{x}_{0:t}), \quad (3.20)$$

for  $\mathbf{x}_t$  as one token. Note that a token is usually embedded into a vector representation to allow the processing of symbolic inputs in a continuous mathematical space.

Outside of NLP, the idea of masked or auto-regressive self-supervised learning has been applied to computer vision by reconstructing images pixel-wise [VKK16; Van+16] or patch-wise [Dos+21]. Similarly, it has been applied

<sup>10</sup>A token is a unit of text, e.g. part of a word or punctuation, which allows for discrete processing of text.

<sup>11</sup>Generative Pre-trained Transformer.

### 3.3. Self-supervised learning

to audio signals [Oor+16a]. In Paper V [Ged+21a], we use generative self-supervised learning of the ‘masked language modelling’ type for ECG data. Since the time series is not discrete, as it is for language with tokens, we also have to consider a segment’s length to mask. A segment length of a single time sample could trivially be solved by interpolation. Segments which are too long may become too difficult to reconstruct. Similarly, Zhang et al. [Zha+23] expanded on this work for ECG-based self-supervised learning.



# Overparameterization

The methods from Chapter 3 are based on deep neural networks combined with explicit or implicit mechanisms that extract low-dimensional representations from the data. On a high level, the success of deep learning is based on three main factors:

1. the availability of large datasets which provide sufficient coverage of the underlying distribution  $p(x, y)$  for realistic scenarios.
2. the advancement of computational power that facilitates training and inference with deep neural networks, especially fuelled by innovation in parallel computing such as GPUs or TPUs.
3. methodological breakthroughs in the design of deep neural networks.

For the last point, one dominant factor has been scale. This concerns utilising increasingly more parameters to boost the modelling capacity of deep neural networks and therefore improve their representational capabilities. ResNets have scaled models to hundreds of layers [He+16] but scaling to a higher number of layers or parameters has been more generally an efficient strategy [TL19]. This is captured by the analysis of neural scaling laws which try to capture how a specific model architecture performs when the parameters are scaled up [Kap+20; Bah+21].

Using more complex models by scaling contradicts the classical bias-variance trade-off [Has+09]. This states that the generalisation risk can be decomposed into three components

$$R(\theta) = \text{Bias}(\theta)^2 + \text{Var}(\theta) + \sigma^2, \quad (4.1)$$

consisting of the square of the bias, the variance and the irreducible error  $\sigma^2$  caused by noise. As we increase the model complexity, the bias term is reduced but the variance increases. Eventually, the model starts to fit the noise of the training data, resulting in very high variance and poor generalisation performance on unseen test data. This is captured by the classical U-shape, see the left half of Figure 4.1. This also holds for deep neural networks. Zhang et al. [Zha+17] showed that deep neural networks have sufficient capacity to fit random labels on large datasets. Despite this very high capacity the perfor-

mance of deep neural networks is not affected as we might expect from the classical bias-variance trade-off.

Therefore, studying the interplay between model complexity and performance is a crucial aspect when dealing with deep neural networks. It allows us to study the generalisation behaviour and provide precise characterisation of the risk. In turn, this analysis of the factors that control the risk yields insights into the components that influence performance such as implicit regularisation mechanisms. Such insights eventually aid in designing efficient and robust models with good performance.

## 4.1 Background

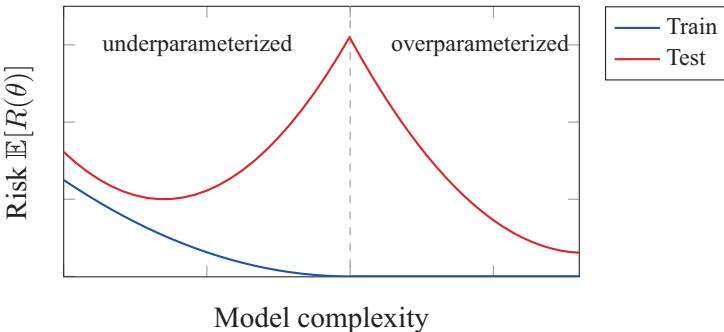
In this chapter, we will consider model parameters  $p$  as a measure of the model complexity<sup>1</sup>. The classical U-shaped bias-variance trade-off holds in the under-parameterized regime. That is where we have fewer parameters  $p$  than training samples  $n$ , or  $\gamma = \frac{p}{n} < 1$ . For linear regression,  $p$  translates to the number of features in  $\mathbf{x}_i$ . Let us consider linear regression on data generated from a random Fourier features model [RR08]. This data generator can be viewed as a 2-layer neural network where the weights in the first layer are fixed [Bel+19]. Using more Fourier features thus results in a network with increasing width. Therefore, the number of features  $p$  in  $\mathbf{x}_i$  increases which we use for linear regression. When continuing to use more random Fourier features, at some point we will have more features  $p$  than training samples  $n$ , or  $\gamma = \frac{p}{n} > 1$ . This is what we denote as the overparameterized regime. While parameter count is often not a suitable measure for model complexity, we can replicate the idea of scaling the width in deep neural networks. Thus, when modelling with deep neural networks, we effectively work with models in the overparameterized regime [Nak+21].

## Generalisation risk

The bias-variance trade-off [Has+09] established that with increasing model complexity the variance of the test data error rises and the bias decreases. Thus, more complex models can lead to overfitting on the noise instead of learning a model that generalises. At the complexity, where the model can memorise the complete training data, i.e. where the training error or the risk on training data is zero, we say that the model can interpolate the data. Generally, we can reach this point at  $\gamma = 1$ , meaning that we have as many parameters  $p$  as we

---

<sup>1</sup>Although other measures could be used including VC dimensional [VC71], Rademacher complexity [BM02], or effective model complexity measures such as AIC or BIC [SS04].



**Figure 4.1:** Double descent curve containing the classical U-shaped bias-variance trade-off in the underparameterized regime and the overparameterized regime of modern deep neural networks.

have training data points  $n$ . This point is therefore also called the interpolation threshold. If we increase the number of parameters beyond the interpolation threshold, we reach the overparameterized regime. From the classical theory, we would assume that the variance on the test data increases further leading to worse generalisation. However, in contrast, Belkin et al. [Bel+19] based on prior kernel analysis [BMM18] showed that the test risk decreases for  $\gamma > 1$ , leading to a second descent. Therefore, the term ‘double descent’ is established. Figure 4.1 shows the double descent.

The generalisation risk of large models has been analysed before the study of double descent [NTS15; DR17; ASS20]. Furthermore, the observation of a double descent curve has been made previously [KH91; GBD92; Opp95] but its implications for understanding large models in modern deep learning were overlooked. Since the seminal work by Belkin et al. [Bel+19] the phenomenon of double descent has been studied in many different contexts. This includes the analysis of weak features [BHX20], random features [MM22], but also deep models including modern architectures such as convolutional networks and transformers [dAs+20; Nak+21]. However, for tractable analysis, the majority of studies rely on linear models [Bar+20; Mut+20; Has+22].

In what follows and in Paper III [GRS24], we focus on linear models. Therefore the question arises of how the study of linear models is justified when we are interested in the generalisation capabilities of nonlinear deep neural networks. In many settings, linear models are simpler to analyse by allowing the use of well-developed tools such as random matrix theory [Tao23] or uniform convergence bounds [Ver18; Wai19] which yield closed-form solutions. Additionally, linear models enable interpretable insights which can serve as a baseline for more complex model structures. However, here we will further justify the use of linear models by following the arguments of Bartlett et al. [BMR21] and Misiakiewicz and Montanari [MM23].

## Optimisation

We mentioned the influence of overparameterization on generalisation connected through the double descent curve in Figure 4.1. Here, we will focus on another aspect of overparameterization: its effect on the optimisation procedure. We will establish the neural tangent model which provides a connection between linear models for which we analyse their generalisation risk and non-linear models that we are interested in [JGH18]. Assume that we are interested in regression problems, then the risk of deep neural networks can be defined using the mean squared error loss function as

$$R(\theta) = \frac{1}{2n} \sum_{i=1}^n (f_\theta(\mathbf{x}_i) - y_i)^2. \quad (4.2)$$

Using empirical risk minimisation as in (2.4) with gradient descent or stochastic gradient descent, we will update the parameters  $\theta$  at iteration  $t$  in the negative direction of the gradient  $\frac{df_t}{dt} = -\eta \nabla_\theta R(\theta_t)$  with  $\eta$  as the learning rate. Due to the nature of deep neural networks, this optimisation problem is highly non-convex. However, gradient descent or stochastic gradient descent converges for real problems to near global optimality. For the optimisation, we have to initialise the parameters of the neural network to some values  $\theta_0$ . Under some technical assumptions, we can prove that the parameters do not change significantly during training. Therefore, we can apply a first-order Taylor expansion to linearise the model around its initialisation point  $\theta_0$

$$f_\theta(\mathbf{x}) \approx f_{\theta_0}(\mathbf{x}) + \nabla_\theta f_{\theta_0}(\mathbf{x})^\top (\theta - \theta_0). \quad (4.3)$$

This linearisation allows us to approximate the trajectory of the parameters during training [Lee+19]. Applying the update step, we obtain

$$\frac{df_{\theta_t}(\mathbf{x}')}{dt} = \frac{\eta}{N} \sum_{i=1}^n (f_{\theta_t}(\mathbf{x}_i) - y_i) \nabla_{\theta_t} f_{\theta_t}(\mathbf{x}_i)^\top \nabla_{\theta_t} f_{\theta_t}(\mathbf{x}'). \quad (4.4)$$

Let us define the mapping  $\phi : \mathbf{x} \mapsto \nabla_{\theta_t} f_{\theta_t}(\mathbf{x})$ . Thus the trajectory of the parameters during training is determined by the inner product  $\phi_{\theta_t}(\mathbf{x}_i)^\top \phi_{\theta_t}(\mathbf{x}_i)$ . For neural networks  $f_\theta$ , we can define the inner product of this mapping of gradients as a kernel, see (2.5)

$$k_{\theta_t}(\mathbf{x}, \mathbf{x}') = \phi_{\theta_t}(\mathbf{x}_i)^\top \phi_{\theta_t}(\mathbf{x}_i) = \nabla_{\theta_t} f_{\theta_t}(\mathbf{x}_i)^\top \nabla_{\theta_t} f_{\theta_t}(\mathbf{x}'). \quad (4.5)$$

When we consider neural networks where the layers become infinitely wide the resulting kernel is the so-called neural tangent kernel [JGH18]. The kernel describes the dynamics of the neural network parameters during training. We can see that close to the initialisation parameters  $\theta_0$ , deep neural networks behave like linear models. Therefore, studying the generalisation risk for linear models can provide insights that translate to deep nonlinear neural networks.

## 4.2 Generalisation risk

For models where we fit parameters to training data, studying their performance on unseen test data is a crucial aspect. This is captured by the so-called generalisation risk. Let us give a brief review of statistical learning theory, see e.g. Vapnik [Vap99]. Early work to quantify the generalisation risk focused on the idea of VC dimension, named after its authors Vapnik and Chervonenkis [VC71], which offers theoretical bounds on the capacity of the hypothesis space by measuring their ability to shatter finite sets of points<sup>2</sup>. However, the VC dimension may become difficult to compute for more complicated models resulting in loose performance bounds. Rademacher complexity by Bartlett and Mendelson [BM02] provides a similar measure while being computed more efficiently and offers tighter bounds for a range of learning algorithms. This measure and others can be categorised as PAC<sup>3</sup>-bounds which provide a framework for the analysis of sample complexity and its relation to generalisation risk [Val84]. These bounds are often derived using probability or concentration inequalities and bounds such as Chernoff bounds [Che52] or Hoeffding's inequality [Hoe63].

The latter analysis based on non-asymptotic high probability inequalities provides valuable insights and is used for the characterisation of high-dimensional models. These bounds are non-asymptotic since both, the number of parameters  $p$  and the number of samples  $n$  are considered fixed. Notably Bartlett et al. [Bar+20] provides conditions for benign overfitting using concentration inequalities. In contrast, we will focus on high-dimensional asymptotic results. Here, we consider the case where the number of parameters  $p$  and the number of training samples goes to infinity, i.e.  $p, n \rightarrow \infty$  while their ratio  $\frac{p}{n} \rightarrow \gamma$  remains constant. This is the complexity measure as used e.g. in Figure 4.1. To increase the complexity, we can either increase the number of parameters  $p$  or decrease the number of training samples  $n$ . Specifically, we utilise the tool of random matrix theory which provides asymptotic results for spectral properties of matrices with random entries [BS10; Tao23]. This tool has been used in the study of deep nonlinear neural networks [PW17] but also to derive double-descent curves for certain models [BHX20; Has+22].

**Example: Linear regression.** Let us briefly exemplify the double descent on a concrete example from Hastie et al. [Has+22], i.e. linear regression. Here, we have  $x_i$  as the input features and  $y_i$  as the output that we want to predict in our model. Assume that the data are generated by

$$y_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \epsilon_i, \quad i = 1, \dots, n, \quad (4.6)$$

---

<sup>2</sup>Effectively a measure of the complexity of a hypothesis space in terms of how many distinct sets of points it can classify perfectly.

<sup>3</sup>Probably Approximately Correct.

with  $(\mathbf{x}_i, \epsilon_i) \sim \mathcal{P}_{\mathbf{x}} \times \mathcal{P}_\epsilon$  where the distribution  $\mathcal{P}_{\mathbf{x}}$  in  $\mathbb{R}^p$  is such that  $\mathbb{E}[\mathbf{x}] = 0$  and  $\mathbb{V}[\mathbf{x}] = \mathbf{C}$ . Equivalently for  $\mathcal{P}_\epsilon$  we have  $\mathbb{E}[\epsilon] = 0$ ,  $\mathbb{V}[\epsilon] = \sigma^2$ . Furthermore, we have  $\mathbf{x}_i \perp\!\!\!\perp \epsilon_i$  and the  $\|\beta\|_2^2 = r^2$ . We want to estimate  $\beta$  using

$$\begin{aligned}\hat{\beta} &= \arg \min_{\beta} \|\beta\|_2^2, \\ \text{s.t. } \mathbf{y} &= \mathbf{X}\beta.\end{aligned}\tag{4.7}$$

The solution of which is  $\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^\dagger \mathbf{X}^\top \mathbf{y}$  using the pseudo-inverse. This solution is the ridgeless least squares estimator. Note that for this optimisation problem using the minimum norm solution has two domains. In the underparameterized regime  $\gamma < 1$ , we have one unique solution but in the overparameterized regime  $\gamma > 1$ , we have infinitely many solutions. With the choice of the optimisation problem (4.7), we explicitly constrain the space of solutions for  $\gamma > 1$ , to one unique solution, i.e. the minimum norm solution.

However, we are interested in the generalisation risk  $R(\beta)$  and decomposing into a bias, variance and irreducible error term as in (4.1). When we consider the mean squared error loss, we can write

$$R(\beta) = \mathbb{E}_{\mathbf{x}}[(y_i - \mathbf{x}_i^\top \hat{\beta})^2] = \|\beta - \hat{\beta}\|_{\mathbf{C}}^2 + \sigma^2.\tag{4.8}$$

Denote the uncentered sample covariance matrix as  $\hat{\mathbf{C}} = \mathbf{X}^\top \mathbf{X}/n$ , then we obtain

$$\beta - \hat{\beta} = \Pi \beta - \frac{1}{n} \hat{\mathbf{C}}^\dagger \mathbf{X}^\top \epsilon,\tag{4.9}$$

with  $\Pi = \mathbf{I} - \hat{\mathbf{C}}^\dagger \hat{\mathbf{C}}$  as the projection onto the null space of  $\mathbf{X}$ . Finally, we utilise this to write the complete risk in terms of the squared bias and variance as the following [Has+22, Lemma 1]

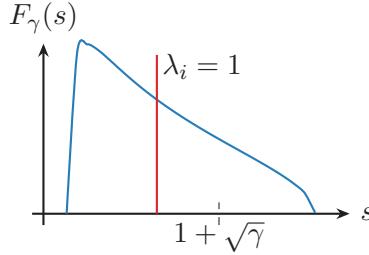
$$R(\beta) = \text{Bias}(\beta)^2 + \text{Var}(\beta) + \sigma^2\tag{4.10a}$$

$$= \beta^\top \Pi \mathbf{C} \Pi \beta + \frac{\sigma^2}{n} \text{Tr}(\hat{\mathbf{C}}^\dagger \hat{\mathbf{C}}) + \sigma^2,\tag{4.10b}$$

where  $\text{Tr}$  denotes the trace.

**High-dimensional analysis.** Hastie et al. [Has+22] provides high-dimensional analyses of the ridgeless least squares solution for the following three different data generators:

1. Isotropic features. We have  $\mathbf{C} = \mathbf{I}$ . We will show the asymptotic solutions of the risk for this case below.
2. Latent space features. We have  $\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \mathbf{I}$  with  $\mathbf{W} \in \mathbb{R}^{p \times d}$  where  $d \ll p$  to have low-dimensional features. A similar data generator is considered in the next section and Paper III [GRS24].



**Figure 4.2:** Visualisation of the Marčenko-Pastur distribution for  $\gamma = 0.3$ . In blue is the sample eigenvalue distribution given by the Marčenko-Pastur distribution and in red is the value of the true eigenvalues. The lower and upper bounds of the Marčenko-Pastur distribution are given by  $(1 \pm \sqrt{\gamma})^2$  and we show the phase transition.

3. Nonlinear model. Here we consider a one-layer neural network with random parameters and nonlinear activation function.

For asymptotic results, we consider as an example the first case with isotropic features  $C = I$ . With results from random matrix theory, we know that the eigenvalues of the sample covariance matrix  $\hat{C}$  converge to the Marčenko-Pastur distribution [MP67], see Figure 4.2. Thus in the asymptotic limit of  $p, n \rightarrow \infty$  such that  $\frac{p}{n} \rightarrow \gamma$ , we have almost surely [Has+22, Theorem 1]

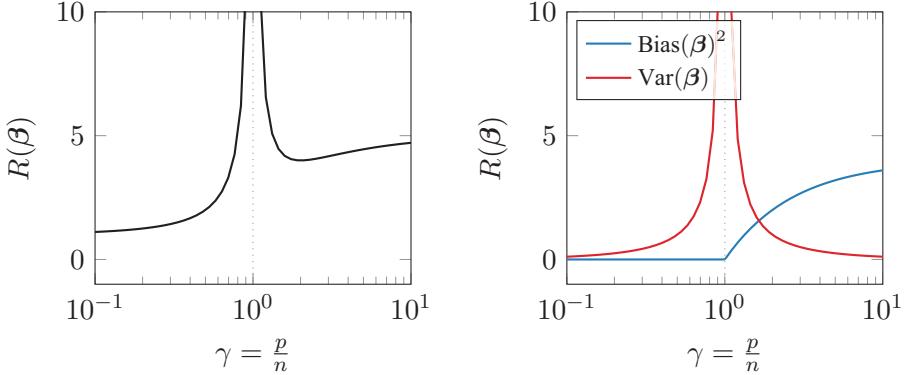
$$R(\beta) \rightarrow \begin{cases} \sigma^2 \frac{\gamma}{1-\gamma} + \sigma^2 & \text{for } \gamma < 1, \\ r^2 \left(1 - \frac{1}{\gamma}\right) + \sigma^2 \frac{1}{\gamma-1} + \sigma^2 & \text{for } \gamma > 1. \end{cases} \quad (4.11)$$

For  $\gamma < 1$  we have no contribution of the bias while the variance increases with  $\gamma$ . Conversely, for  $\gamma > 1$  the bias increases and the variance decreases with  $\gamma$ . In Figure 4.3 we show the double descent curve of the ridgeless least squares estimator on isotropic features (left) and the decomposition in bias and variance terms according to (4.11).

## 4.3 Analysis with low-dimensional manifold data

Let us connect the analysis of the generalisation risk to the learning of low-dimensional representations. The goal of extracting these low-dimensional representations was to be ‘useful’. Here, we characterise the usefulness of the representations by their generalisation risk. Earlier<sup>4</sup>, we noted that for many real-world observations, the data lie on a low-dimensional manifold. Therefore, extracting representations that match this manifold and remove the noise contribution is crucial to obtaining a low generalisation risk.

<sup>4</sup>In Chapter 1 and Chapter 3



**Figure 4.3:** Left: Double descent of least squares estimator on isotropic features. Right: decomposition of the left plot in squared bias and variance terms.

To be able to analyse the generalisation risk of a specific model on observed data, we follow the example from the last section and present linear data-generating processes. This allows for close-form solutions of the generalisation risk. Specifically, we generate data which lie on a low-dimensional linear manifold. We have already mentioned one such model as presented in Hastie et al. [Has+22] with the latent space model

$$y_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \epsilon_i, \quad \mathbf{x}_i \sim \mathcal{P}_{\mathbf{x}}, \quad (4.12a)$$

$$\mathbb{E}[\mathbf{x}] = 0, \quad \mathbb{V}[\mathbf{x}] = \mathbf{C} = \mathbf{W}\mathbf{W}^\top + \mathbf{I}, \quad (4.12b)$$

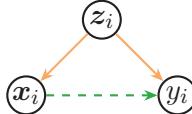
where  $\mathbf{W} \in \mathbb{R}^{p \times d}$  with  $d \ll p$ . Thus the linear manifold has dimension  $d$ , while the observations have dimension  $p$ . If  $\mathbf{x}$  is Gaussian distributed, we can write

$$\mathbf{x}_i = \mathbf{C}^{1/2} \mathbf{z}_i, \quad \mathbf{z}_i \sim \mathcal{N}(0, \mathbf{I}_d). \quad (4.13)$$

In the following, we consider a more general data generator, that is not restricted to isotropic low-dimensional variables  $\mathbf{z}_i$ . Instead, we use the spiked covariance model [Joh01]. In this model the covariance matrix  $\mathbb{E}[\mathbf{x}] = \mathbf{C}$  is given by a base covariance matrix  $\mathbf{C}_0$  and a  $d$ -dimensional data subspace

$$\mathbf{C} = \mathbf{C}_0 + \sum_{j=1}^d \lambda_j \mathbf{v}_j \mathbf{v}_j^\top. \quad (4.14)$$

The base covariance is often considered  $\mathbf{C}_0 = \mathbf{I}$  and the low-rank perturbation consists of  $d$  spiked eigenvalues  $\lambda_1, \dots, \lambda_d$  and corresponding eigenvectors  $\mathbf{v}_1, \dots, \mathbf{v}_d$ . This allows for precise control of the singular value distribution of  $\mathbf{X}$  and is more general than the isotropic assumption by the latent space



**Figure 4.4:** Data generator for latent factor regression model, where observations  $x_i$ ,  $y_i$  are samples from a low-dimensional manifold. The green dashed line shows the alternative description directly from  $x_i$  to  $y_i$ .

model from Hastie et al. [Has+22]. We combine the spiked covariance model for regression tasks to obtain the latent factor regression model [Bin+21]

$$\mathbf{x}_i = \mathbf{W} \mathbf{z}_i + \mathbf{e}_i, \quad (4.15a)$$

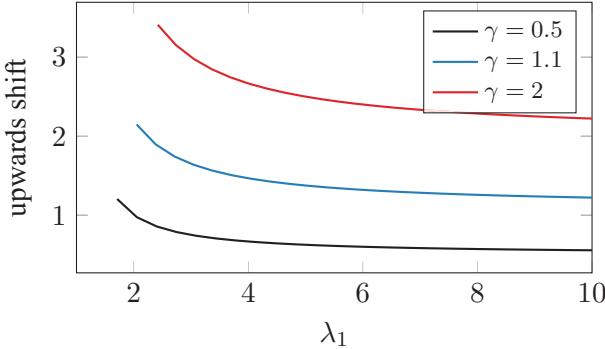
$$y_i = \boldsymbol{\theta}^\top \mathbf{z}_i + \varepsilon_i, \quad (4.15b)$$

with  $\mathbf{z}_i \sim \mathcal{N}(0, \mathbf{C}_z)$  where  $\mathbf{C}_z = \sum_{j=1}^d \mathbf{v}_j \lambda_j \mathbf{v}_j^\top$  is the low-rank perturbation of the spiked covariance model. Equivalently we can write the model in the form of (4.12) that Hastie et al. [Has+22] uses as  $y_i = \mathbf{x}_i^\top \boldsymbol{\omega} + \nu_i$  with  $\mathbf{x}_i \sim \mathcal{N}(0, \mathbf{C})$ ,  $\nu_i \sim \mathcal{N}(0, \sigma_\nu^2)$ ,  $\boldsymbol{\omega} = \mathbf{W} \mathbf{C}_z (\mathbf{I}_d + \mathbf{C}_z)^{-1} \boldsymbol{\theta}$ , and  $\sigma_\nu^2 = \sigma_e^2 + \boldsymbol{\theta}^\top (\mathbf{I}_d + \mathbf{C}_z)^{-1} \mathbf{C}_z \boldsymbol{\theta}$ . For details, see Appendix A of Paper III [GRS24]. In Figure 4.4, we visualise this data generator where we highlight that the observations  $x_i$  and  $y_i$  depend on the low-dimensional manifold given by the variable  $z_i$ .

Besides the improved expressiveness of the spiked covariance model over the latent space from Hastie et al. [Has+22], we can utilise known properties of this model from random matrix theory. Notably Johnstone and Paul [JP18] provides an in-depth overview of results for the spiked covariance model. Let us point to the two most relevant results for this dissertation.

**Eigenvalue shift.** Consider the case of a single spike eigenvalue, i.e.  $d = 1$ . In this case the eigenvalues are  $\{\lambda_1, 1, \dots, 1\}$  with  $\lambda_1 > 1$ . Depending on the value of  $\lambda_1$ , the distribution of sample eigenvalues will have a different form. We established, that for  $\lambda_1 = 1$  we have an isotropic covariance whose sample eigenvalues will follow a Marčenko-Pastur distribution [MP67]. This distribution is defined in the interval  $[(1 - \sqrt{\gamma})^2; (1 + \sqrt{\gamma})^2]$  with an additional point mass at 0 for  $\gamma > 1$ . However, there exists a phase transition for the sample eigenvalue distribution at  $1 + \sqrt{\gamma}$  depending on  $\lambda_1$ :

- Case  $\lambda_1 \in [1, 1 + \sqrt{\gamma}]$ : The sample eigenvalue distribution of the bulk stays Marčenko-Pastur distributed. The sample distribution of the eigenvalue  $\lambda_1$  follows a limiting Tracy-Widom distribution  $n^{2/3} \frac{\lambda_1 - \mu(\gamma)}{\sigma(\gamma)} \xrightarrow{\mathcal{D}} TW_1$ , with  $\mu(\gamma) = (1 + \sqrt{\gamma})^2$  and  $\sigma(\gamma) = (1 + \sqrt{\gamma})^{4/3} \gamma^{-1/6}$  [Joh01; BAP05]. The mean  $\mu(\gamma)$  lies on the upper interval bound of the Marčenko-Pastur distribution and therefore cannot be distinguished from the noise eigenvalues according to the eigenvalue distribution.



**Figure 4.5:** Upwards shift of eigenvalues. All curves start on the point  $(1 + \sqrt{\gamma}; (1 + \sqrt{\gamma})^2)$ . The x-axis value is the phase transition threshold  $1 + \sqrt{\gamma}$  and the y-axis value is the upper boundary of the Marčenko-Pastur distribution.

- Case  $\lambda_1 > 1 + \sqrt{\lambda}$ : Similarly to the previous case, the sample eigenvalue distribution of the bulk stays Marčenko-Pastur distributed. However, the sample distribution of the eigenvalue  $\lambda_1$  follow a Gaussian  $n^{1/2} \frac{\hat{\lambda}_1 - \mu(\lambda, \gamma)}{\sigma(\lambda, \gamma)} \xrightarrow{\mathcal{D}} \mathcal{N}(0, 1)$ , with  $\mu(\lambda, \gamma) = \gamma \frac{\lambda}{\lambda-1} + \lambda$  and  $\sigma^2(\lambda, \gamma) = 2\lambda^2(1 - \frac{\gamma}{(\lambda-1)^2})$  [BS06; Pau07; YJ18]. The mean function has an upward shift compared with the upper bound of the Marčenko-Pastur distribution. This upward shift decreases with larger  $\lambda_1$ . We visualised this in Figure 4.5.

**Eigenvector shift.** Next to the eigenvalue shift, the corresponding eigenvectors can also be inconsistent. For the population eigenvectors  $\mathbf{v}$  and sample eigenvectors  $\hat{\mathbf{v}}$  corresponding to the spiked eigenvalue  $\lambda_1$ , we have as  $\frac{p}{n} \rightarrow \gamma$  [Pau07; Joh+09]

$$(\mathbf{v}^\top \hat{\mathbf{v}})^2 \rightarrow \begin{cases} \frac{1-\gamma/(\lambda-1)^2}{1+\gamma/(\lambda+1)} & \text{for } \lambda_1 > 1 + \sqrt{\gamma}, \\ 0 & \text{for } \lambda_1 \in [1, 1 + \sqrt{\gamma}]. \end{cases} \quad (4.16)$$

This result implies that if the spike eigenvalue  $\lambda_1$  lies below the phase transition threshold  $1 + \sqrt{\gamma}$ , the sample eigenvector is not close to the population eigenvector. Instead, the sample and population eigenvector are orthogonal which means that they can lie anywhere in the space  $\mathbb{R}^p$  as  $p \rightarrow \infty$ . However, if the spike eigenvalue  $\lambda_1$  is above the phase transition threshold, we can precisely characterise the relation between sample and population eigenvectors.

In Paper III [GRS24], we make use of the eigenvalue and eigenvector shift for spiked covariance models with  $d \geq 1$ . We use these relations to characterise the generalisation risk  $R$  of a model consisting of PCA and linear regression, in short, principal component regression (PCR). The model takes the sample eigenvectors  $\hat{\mathbf{V}} = [\hat{\mathbf{v}}_1, \dots, \hat{\mathbf{v}}_k]$  of the sample covariance matrix  $\hat{\mathbf{C}}$  where  $k \leq$

$p$  is the number of principal components. The observations are then mapped to a low-dimensional state through the PCA-based projection  $\hat{z}_i = \hat{V}^\top \mathbf{x}_i$ . Then we perform unregularized linear regression in that space to obtain  $\hat{\theta} : \hat{\mathcal{Z}} \mapsto \mathcal{Y}$

$$\hat{\theta} = (\hat{\mathbf{Z}}^\top \hat{\mathbf{Z}})^\dagger \hat{\mathbf{Z}}^\top \mathbf{y}. \quad (4.17)$$

This model mimics the data-generating process of the latent factor regression models and allows us to learn low-dimensional representations of the data. We can quantify the change in risk for misspecified models, i.e. when the number of principal components in the PCR model does not match the true dimension of the low-dimensional latent space  $\mathcal{Z}$ .

## 4.4 Discussion

**Regularisation.** The results in Paper III [GRS24] indicate that the PCR model effectively regularises the regression problem. This is achieved by reducing the dimensionality of the feature space to those dimensions which explain most of the variance in the data. Therefore, data directions that are related to noise are dismissed and we retain the information, which is most pertinent to the regression problem. Next to regularisation with PCA, where there are multiple analyses based on different assumptions [BPR07; DFH17; XH19; WX20], there are many regularisation strategies which can be analysed and compared. For linear regression models, the effect of Ridge or L2 regularisation has been analysed by Hastie et al. [Has+22]. The connections between adversarial training of different types to Lasso and Ridge regularisation are drawn in Ribeiro et al. [Rib+24]. For binary linear classification, the effect of Lasso or L1 regularisation is studied in Deng et al. [DKT22]. More generally, we can consider the class of spectral regularisation methods, see Bauer et al. [BPR07]. This class includes Landweber iterations which is a special case of gradient descent and allows to draw parallels to the optimisation procedure [Lan51].

**Benign overfitting.** Bartlett et al. [Bar+20] coined the term benign overfitting which described the phenomenon where in certain scenarios the generalisation risk becomes low despite interpolating or overfitting on the training data. This can be imagined as such that the function which the model  $f$  learns is fit globally while the noise is fit only locally. In contrast to benign overfitting, there exists catastrophic overfitting, which describes the classical scenario where the model overfits the noise and reaches a high generalisation risk. However, in many practical settings Mallinar et al. [Mal+22] observe that deep neural networks do not fall in either of these regimes. Therefore the authors call for a new taxonomy of overfitting by including the intermediate regime as ‘tempered’. Deep neural networks which are stopped early [Pre02] fall into the category of benign overfitting. However, if the neural network is trained until

interpolation, it reaches the tempered regime. Therefore, for practical applications, we have to be cautious about the potential for benign overfitting.

**Implicit assumption.** When formulating the regression problem in (4.7), we observed that in the underparameterized regime  $\gamma < 1$ , there is one solution while in the overparameterized regime  $\gamma > 1$  there are infinitely many solutions. Among those solutions, we chose the one with the minimum norm. This assumption is made implicitly by the problem formulation. The solutions which are found are then combined with the complexity measure, e.g. the degree of overparameterization  $\gamma$ , into a single plot as in Figures 4.1 and 4.3. Curth et al. [CJS24] challenge this depiction by providing a perspective that multiple complexity measures are plotted on one single axis. The authors suggest that the different criteria for the selection of the solution in the overparameterized regime should not be combined into one axis but should rather be two dimensions of the analysis. This perspective provides a more nuanced understanding of the observed phenomena and suggests new paths for model selection criteria.

# CHAPTER 5

## Dynamical systems

This chapter is the first of two chapters which are less focused on general methods of deep learning and more on how we can apply them in certain contexts. We will first introduce the specific field and then highlight how we can learn low-dimensional representations and why these representations are useful. This chapter specifically concerns the field of dynamical systems [KKH95], which models the evolution of systems over time. This general definition hints that the study of dynamical systems encompasses a wide range of phenomena including mathematics, physics, chemistry, and biology, but also econometrics and social sciences [Str18]. The important concept in dynamical systems is their temporal nature. Thus understanding and modelling of dynamical systems is a central aspect of when we try to model observations in the real world.

One of the possible categories in which we can distinguish systems are autonomous and non-autonomous systems. The first category, autonomous systems, describes systems which do not explicitly depend on an input variable. However, we can measure parts of the system to obtain measurements  $y_t$  at time  $t$ . This describes systems such as population dynamics or nonlinear oscillators. In contrast, within this chapter, we focus on the second category, non-autonomous systems, i.e. systems which depend on some input sequence  $u_t$ . Such systems include pharmaceutical systems, financial markets, and engineering systems such as robotics.

A dynamical system can be described by the temporal evolution of some internal states  $h^1$ . We can describe this with  $\frac{d}{dt}h_t = f(h_t, u_t)$ . In general, we do not have access to the internal state  $h_t$  but only to measurements of the input  $u_t$  and outputs  $y_t$ . Here, we are interested in discrete time series, i.e. where  $u_t$ ,  $y_t$  and  $h_t$  are discretized with a fixed sampling rate. Let us consider the univariate or single-input single-output case with  $u_t \in \mathbb{R}$  and  $y_t \in \mathbb{R}$ . We consider time measurements from  $t = 1, \dots, T$ . Figure 5.1 gives a general block diagram of a non-autonomous dynamical system.

---

<sup>1</sup>More often in dynamical systems the state is denoted by  $x_t$ . However, in the context of machine learning  $x_t$  describes the input to a model. Therefore, we denote the state as  $h_t$  which coincides with the internal states of recurrent neural networks.

**Figure 5.1:** Block diagram of a dynamical system with input  $u_t$  and outputs  $y_t$ .

Once we have collected the observations  $\{(u_{i,1}, y_{i,1}), \dots, (u_{i,T}, y_{i,T})\}_{i=1}^n$  of  $n$  input-output samples<sup>2</sup>, we want to obtain a model of the system in a data-driven approach. Similar to the goal of modelling, which we discussed in Chapter 1, we want to uncover the structure of the modelled system to better understand it or we want to use the model for predictions. In this chapter, we focus on the latter. Thus, learning a good model means that the model can accurately predict the output  $y_t$  given the past outputs  $y_{1:t-1}$  as well as the past and current inputs  $u_{1:t}$ . To unify the input, we create a new input vector  $\mathbf{x}_t = [u_t, \dots, u_{t-m_u}, y_{t-1}, \dots, y_{t-m_y}]^\top$  which includes the past inputs and outputs that we want to use for prediction with a memory of  $m_u$  and  $m_y$  time steps for input and output, respectively. Often not the complete sequence  $1 : t$  is utilised but a fixed memory of  $m$  time steps. Formally, we want to obtain a model for  $p(y_t | \mathbf{x}_t)$ <sup>3</sup>. Note that  $\mathbf{x}_t$  includes past inputs and outputs up to a memory of  $m$  time steps. In the following, we will discuss two approaches of obtaining a model of dynamical systems. In both sections, we focus on parametric models and leave the discussion of non-parametric models to other sources [PD10; Pil+14; Pil+23].

## 5.1 Classical modelling

Modelling of dynamical systems has a long history. Here, we consider the field of system identification [Zad56; Lju98] which concerns the mathematical modelling of dynamical systems from input-output measurements. Continuous-time systems, which we can describe with differential equations, give in the linear time-invariant case rise to transfer function<sup>4</sup> descriptions through the Laplace transformation [Doe13]. The equivalent transfer function for discrete-time systems, which are described by difference equations, is given by the z-transformation [Opp99]. The use of transfer functions is especially useful in control engineering. However, to be able to write down the differential equation of the system, we need insights into the system's properties. For example, if we have a full understanding of the system, we can derive the set of differ-

<sup>2</sup>In general, not all samples  $i$  have to have the same length over time  $T$ .

<sup>3</sup>This is often considered a one-step-ahead prediction. In contrast, there exists multi-step ahead prediction where we want to predict multiple steps  $\tau$ , formalised as  $p(y_{t:t+\tau} | \mathbf{x}_t)$ .

<sup>4</sup>Note that this holds only for linear time-invariant systems which are often described by first-order differential equations.

ential equations that describe the system from first principles. This approach is generally described as white-box modelling.

**Auto-regressive models.** However, if we do not have specific knowledge about the system, we can consider input-output relations directly. Thus, we can formulate auto-regressive models with exogenous inputs (ARX)<sup>5</sup> or auto-regressive models moving average with exogenous inputs (ARMAX). We can describe an ARX model with

$$y_t = \sum_{i=1}^{n_a} a_i y_{t-i} + \sum_{j=0}^{n_b} b_j^\top u_{t-j} + e_t. \quad (5.1)$$

Here  $n_a$  is the order of the auto-regressive part with coefficients  $a_i$  and  $n_b$  is the order of the exogenous part with coefficients  $b_j$ . Note that higher order implies considering more past values for the current prediction leading to more complex relations. While (5.1) describes a linear model, we can generalise this to nonlinear ARX (NARX) models as

$$y_t = f(\mathbf{x}_t) + e_t = f(u_t, \dots, u_{t-m_u}, y_{t-1}, \dots, y_{t-m_y}) + e_t. \quad (5.2)$$

This type of model is often considered as a black-box model which is purely described by its input-output relationship. It does not directly consider the dynamics of the underlying system.

**State-space models.** We can include more structure into the model by introducing an internal state  $\mathbf{h}_t$  to formulate state-space models. These models are given by<sup>6</sup>

$$\mathbf{h}_{t+1} = f(\mathbf{h}_t, \mathbf{x}_t, e_t), \quad (5.3a)$$

$$y_t = g(\mathbf{h}_t, \mathbf{x}_t, v_t), \quad (5.3b)$$

with state transition function  $f$ , output reading function  $g$  as well as process noise  $e_t$  and measurement noise  $v_t$  [SWN11]. In this formulation, it becomes more clear that we try to use the internal state  $\mathbf{h}_t$  as a compact representation of the system's dynamics. In many instances, the hidden state encodes information about past inputs and the system's evolution. Thus the hidden state  $\mathbf{h}_t$  can be viewed as a low-dimensional representation of the dynamical system at time  $t$ . There are instances where we can derive a state-space model from first principles, such as a double pendulum. In other cases, we do not have full knowledge of the system and can derive a more general mapping of  $g$  and  $h$  for example using deep neural networks. Thus, depending on the knowledge we have about the system, a state-space model can be closer to a white-box

<sup>5</sup>Here, exogenous inputs refer to non-autonomous systems with an input to the system  $u_t$ .

<sup>6</sup>This is the nonlinear formulation. Equivalently to the linear formulation of ARX models in (5.1), we can formulate linear state-space models with  $f$  and  $g$  described by linear functions.

or a black-box model which aims to capture the system's behaviour without explicit knowledge of its internal mechanisms [Lju98].

The modelling of dynamical systems with transfer functions, (nonlinear) state-space models or (N)AR(MA)X models describes only a small set of possible models to choose from [Lju98]. However, it highlights that we have to consider the knowledge we have about the system, which leads us to choosing different types of models to embed prior information. Additionally, we can combine partial in-depth knowledge about the system in a white-box nature with an unknown modelling part in a black-box nature to obtain a middle ground. This results in a hybrid modelling approach such as grey-box models. For example, we can incorporate structural knowledge about the system with empirical data-driven components.

Once we decide on a model structure, we have to train the model parameters given the data we have. We will utilise the same idea as we introduced early in this dissertation about risk minimisation of parameterised models (2.3). While the temporal component is often considered implicit in the data set through the instance iterator  $i$ , we explicitly formulate it as

$$\arg \min_{\theta} \sum_{i=1}^n \sum_{t=1}^{T-1} \ell(f_{\theta}(\mathbf{x}_{i,t}), y_{i,t}), \quad (5.4)$$

for the model  $f_{\theta}$  with parameters  $\theta$ . This indicates that we not only iterate over all sample instances but also over all time instances. The parameters  $\theta$  include for example for the ARX model (5.1) all  $a_i$  and  $b_j$  and for state-space models (5.3) the parameters of the functions  $f$  and  $g$ . Within this introduction, we will not go into details on specific methods to solve this optimisation problem, model selection criteria or theoretical discussions about model complexity and generalisation but we refer to established resources [SS88; Lju98; VV07].

## 5.2 Modelling dynamics with neural networks

While classical system identification is a well-established field with a long history, it has its limitations. In the last section, we discussed some model structures including (nonlinear) state-space models and (N)AR(MA)X models. These structures are studied in-depth for the linear case [Lju98] but extensions to nonlinear system identification exist [SL19; SWN11]. However, limitations arise when dealing with complex and nonlinear systems where linear models are unsuited, and traditional nonlinear models require prior knowledge about the model structure or struggle to capture the system dynamics effectively.

This limitation can be overcome by the use of deep neural networks as models. See Ljung et al. [Lju+20] and Pillonetto et al. [Pil+23] for an overview of the

topic. Thus, we change the model structure  $f_\theta$ . The optimisation problem (5.4) remains the same. Note, however, that for deep neural networks, we focus on first-order gradient-based optimisation methods such as stochastic gradient descent instead of higher-order methods.

For the model, we are interested in modelling  $f_\theta : \mathbf{x}_t \mapsto y_t$ . Later, we will also describe probabilistic models which can model  $p(y_t | \mathbf{x}_t)$ . Let us briefly discuss some methods of how  $f_\theta$  can be modelled with deep neural networks [Pil+23]:

- **Fully-connected neural network.** This is the most basic version where we model the relation between input  $\mathbf{x}_t$  and output  $y_t$  using a fully connected neural network. For one hidden layer with weights  $\mathbf{W}$  including biases and nonlinear activation function  $\sigma$

$$y_t = \mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{x}_t). \quad (5.5)$$

These types of networks represent an ARX structure as in (5.1). The fixed-size input  $\mathbf{x}_t$  and the lack of inherent memory mechanisms limit the ability of this type of model to effectively capture the dynamic patterns.

- **Convolutional neural networks.** While this architecture is most dominantly applied to computer vision for its shift invariance, recent work has highlighted that convolutional neural networks can effectively be used for time series [And+19; BKK18]. The convolutional layer outputs a filtered version of the input by applying a convolution  $\mathbf{w} * \mathbf{x}_{t-k:t} = \sum_{j=1}^k \mathbf{w}_j^\top \mathbf{x}_{t-j}$  with  $\mathbf{w}_j$  as a vector of weights. Thus a convolutional neural network with one hidden layer results in

$$y_t = \mathbf{W}_2 * \sigma(\mathbf{W}_1 * \mathbf{x}_t), \quad (5.6)$$

with  $\mathbf{W}_1$  as the set of weights  $\mathbf{w}_j$  and similar for  $\mathbf{W}_2$ . This formulation shows that a convolutional network has the same ARX structure as a fully connected neural network. Through stacking multiple convolutional layers, we obtain an increasing receptive field which can model dependencies over longer ranges. However, convolutional networks have no inherent dynamic structure such as memory mechanisms which limit their modelling capacity.

- **Recurrent neural networks.** These models, abbreviated as RNNs, have an explicit memory mechanism through the use of an internal hidden state. Therefore, we can view them as deterministic nonlinear state-space models with

$$\mathbf{h}_{t+1} = \sigma(\mathbf{W}_{hh} \mathbf{h}_t + \mathbf{W}_{hx} \mathbf{x}_t), \quad (5.7a)$$

$$y_t = \mathbf{W}_{yh} \mathbf{h}_t + \mathbf{W}_{yx} \mathbf{x}_t. \quad (5.7b)$$

We can extend these models with gating mechanisms for long-range modelling [HS97; Cho+14]. Recently, stacking of these state-space models

with a special initialisation of  $\mathbf{W}_{hh}$  and linear activation function  $\sigma$ <sup>7</sup> has led to the development of structured state-space sequence models (S4) and its successors [Gu+20; GGR22; SWL23; GD23].

- **Attention mechanism.** Transformer architectures are used for time series modelling including sequence-to-sequence modelling [Vas+17]. The attention mechanism allows for flexible routing to re-weight the input dynamically when making predictions [BCB15]. Specifically, it is given by

$$\mathbf{h}_{t+1} = \sigma \left( \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}} \right) \mathbf{V}, \quad (5.8)$$

with  $\sigma$  as softmax activation function and where  $\mathbf{Q}$ ,  $\mathbf{K}$  and  $\mathbf{V}$  are the query, key, and value matrices which are transformations of  $\mathbf{x}_t$  and  $d$  is the dimension of  $\mathbf{x}_t$ . Utilising attention mechanisms allows for more flexible modelling.

While we have discussed a range of possible deep neural network structures for the modelling of dynamical systems, there are more possible structures. This includes graph neural networks [Vel+18; Wu+20; Vel23], energy-based models [Hen+21], and latent variable models. The latter include models such as recurrent VAEs, which we will cover in the next section.

Let us briefly discuss some theoretical aspects of using deep neural networks for dynamical modelling. In Chapter 4 we discussed the relation of model complexity to the generalisation risk. Similar observations and discussions can be held about dynamic models such as classic nonlinear ARX models [Rib+21]. Therefore, the discussion on generalisation risk extends to dynamical models. While for classical modelling with state-space or ARX models, there is a large interest in the stability of the obtained dynamic model, the equivalent for deep neural networks is less well studied [Bon+20; BFS21a; BFS21b]. More work is necessary in this direction. Similarly, deep neural networks are generally considered black-box models. Thus the interpretability of learnt parameters is limited. So far, we considered this aspect within this dissertation as connected to learning low-dimensional representations which can potentially be connected to the data-generating process. However, currently, there is a trade-off between the flexibility and performance of a model on the one side and model interpretability on the other side. A final point of discussion should be data requirements. While deep neural networks offer flexible modelling, they often necessitate a substantial amount of collected data. This highlights a key consideration in the choice of model between classical system identification models and deep neural networks.

---

<sup>7</sup>Even though the activation  $\sigma$  in within one state-space layer is linear in S4, there are nonlinear activation functions at the output  $y_t$  of each state-space layer which renders S4 a nonlinear model.

## 5.3 Deep state-space models

So far in this chapter, we discussed how to model dynamical systems with classical methods from system identification and with deep neural networks. However, we have omitted how to model low-dimensional representations except for (5.3) where we discussed the use of the hidden state  $\mathbf{h}_t$  in state-space models as a compact representation of the dynamics. In this section, will present on example of extending state-space models with deep neural networks with the so-called deep state-space model. This model combines state-space models with probabilistic deep neural networks for probabilistic predictions. In Paper IV [Ged+21b] the deep state-space model is studied in-depth for nonlinear system identification.

Deep state-space models<sup>8</sup> combine the ability of state-space models to describe dynamics with the expressive modelling power of deep neural networks. The model is therefore more flexible than a standard state-space model but allows improving modelling of temporal data because of its recurrent nature with a set of compact hidden states. Furthermore, its probabilistic nature allows it to capture prediction uncertainty and to generate unseen outputs. Deep state-space models have been extensively studied in different contexts and many variations have been proposed [BO14; Chu+15; FAK15; Fra+16; Ali+17; Ali+17; Ran+18; Li+19; MB21]. Girin et al. [Gir+21] provides an overview.

### Model architecture

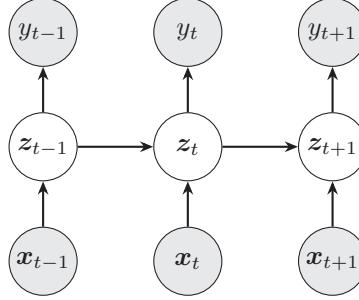
We will first elaborate on the most generic form of combining a probabilistic neural network with a state-space model to obtain a deep state-space model. In this general case, we will see that learning the model parameters becomes difficult. As a solution, we present the VAE-RNN [Fra18] as one specific example.

**General model.** The key components of the deep state-space model include a VAE for learning low-dimensional representations with a time-varying prior for its latent state  $\mathbf{z}$  which therefore depends on previous time steps and allows for modelling of the dynamics. For training, we have to distinguish between the inference and generative network. Figure 5.2 shows a graphical model of a basic state-space model. With d-separation [GVP90] we can write the joint distribution of the model as

$$p_{\theta}(y_{1:T}, \mathbf{z}_{1:T} | \mathbf{x}_{1:T}, \mathbf{z}_0) = \prod_{t=1}^T p_{\theta}(y_t | \mathbf{z}_t) p_{\theta}(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x}_t), \quad (5.9)$$

---

<sup>8</sup>Deep state-space models are known under a variety of names including dynamic / sequential / temporal VAEs, recurrent latent variables models, stochastic RNNs and more.



**Figure 5.2:** Graphical model of a state-space model. Round blocks indicate probabilistic variables and shaded blocks are observed variables.

with the initial state  $\mathbf{z}_0$ . When we formulate this model as a VAE the latent state  $\mathbf{z}_t$  becomes the prior [RMW14; KW14]. We can directly see that the prior is time-dependent. Defining the prior as a Gaussian distribution, we obtain

$$p_{\theta}(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x}_t) = \mathcal{N}(\mathbf{z}_t | \mu_t^{\text{prior}}, \sigma_t^{\text{prior}} \mathbf{I}), \quad (5.10a)$$

$$[\mu_t^{\text{prior}}, \sigma_t^{\text{prior}}] = f_{\theta}^{\text{prior}}(\mathbf{z}_{t-1}, \mathbf{x}_t). \quad (5.10b)$$

Similarly, the decoder of the VAE is defined as a Gaussian distribution with

$$p_{\theta}(y_t | \mathbf{z}_t) = \mathcal{N}(y_t | \mu_t^{\text{dec}}, \sigma_t^{\text{dec}} \mathbf{I}), \quad (5.11a)$$

$$[\mu_t^{\text{dec}}, \sigma_t^{\text{dec}}] = f_{\theta}^{\text{dec}}(\mathbf{z}_t). \quad (5.11b)$$

For training<sup>9</sup>, we need to define the encoder distribution as well. Here, we rely on variational inference with the variational distribution

$$q_{\phi}(\mathbf{z}_{1:T} | y_{1:T}, \mathbf{x}_{1:T}, \mathbf{z}_0) = \prod_{t=1}^T q_{\phi}(\mathbf{z}_t | \mathbf{z}_{t-1}, y_{t:T}, \mathbf{x}_{t:T}). \quad (5.12)$$

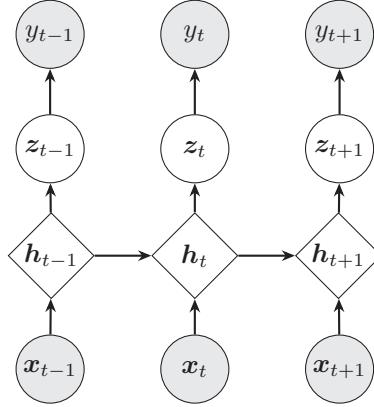
From this factorisation, we see that the conditioning on future time steps  $t : T$  requires a smoothing step. The smoothing complicates the training procedure because for nonlinear, non-Gaussian systems it requires methods based on particle filters [SGN05; Gus10] or sequential monte carlo methods [DDG01].

**VAE-RNN.** A solution to avoid such a smoothing step is through the introduction of a hidden state  $\mathbf{h}_t$  which allows for the state  $\mathbf{z}_t$  to be conditionally independent given  $\mathbf{h}_t$ . This model describes a basic version of a deep state-space model, namely the VAE-RNN [Fra18], see Figure 5.3. Thus, we can define the joint distribution including the hidden state  $\mathbf{h}_t$

$$p_{\theta}(y_{1:T}, \mathbf{z}_{1:T}, \mathbf{h}_{1:T} | \mathbf{x}_{1:T}, \mathbf{h}_0) = \prod_{t=1}^T p_{\theta}(y_t | \mathbf{z}_t) p_{\theta}(\mathbf{z}_t | \mathbf{h}_t) p_{\theta}(\mathbf{h}_t | \mathbf{h}_{t-1}, \mathbf{x}_t). \quad (5.13)$$

---

<sup>9</sup>Or equivalently, parameter inference.



**Figure 5.3:** Graphical model of the VAE-RNN model. Rectangular blocks indicate non-probabilistic variables.

In this factorisation, we note the recurrence through the introduced hidden state  $\mathbf{h}_t$ . Since this variable is deterministic, we have  $p_\theta(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{x}_t) = \delta(\mathbf{h}_t - \tilde{\mathbf{h}}_t)$  centred at  $\tilde{\mathbf{h}}_t$ . In practice, this recurrence is implemented with a recurrent neural network  $\mathbf{h}_t = f_\theta^{RNN}(\mathbf{h}_{t-1}, \mathbf{x}_t)$ . According to the graphical model in Figure 5.3, the prior distribution becomes conditionally independent  $p_\theta(\mathbf{z}_{1:T}|\mathbf{h}_{1:T}) = \prod_{t=1}^T p_\theta(\mathbf{z}_t|\mathbf{h}_t)$  which we can describe again with a Gaussian distribution to obtain

$$p_\theta(\mathbf{z}_t|\mathbf{h}_t) = \mathcal{N}(\mathbf{z}_t|\mu_t^{\text{prior}}, \sigma_t^{\text{prior}} \mathbf{I}), \quad (5.14a)$$

$$[\mu_t^{\text{prior}}, \sigma_t^{\text{prior}}] = f_\theta^{\text{prior}}(\mathbf{h}_t). \quad (5.14b)$$

Since the decoder distribution does not depend on  $\mathbf{h}_t$ , it is given by the same distribution as in the state-space model, see (5.11). In the state-space model, we were limited in the encoder (variational) distribution since we required a smoothing step for parameter inference. When choosing the structure of the VAE-RNN with the introduction of the hidden state  $\mathbf{h}_t$ , we obtain the following factorisation

$$q_\phi(\mathbf{z}_{1:T}, \mathbf{h}_{1:T}|\mathbf{y}_{1:T}, \mathbf{x}_{1:T}, \mathbf{h}_0) = \prod_{t=1}^T q_\phi(\mathbf{z}_t|y_t, \mathbf{h}_t) p_\theta(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{x}_t). \quad (5.15)$$

Thus, we obtain a tractable factorisation for training.

## Training

The training of deep state-space models relies on a temporal extension of the training objective for VAEs. This is defined through the ELBO, see (3.14). Let

us first rewrite the ELBO optimisation for a standard VAE to obtain

$$\log p_{\theta}(\mathbf{x}) \geq \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[ \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right] := \mathcal{L}(\theta, \phi). \quad (5.16)$$

In this formulation, we can easily extend the ELBO for temporal data which yields the following for the VAE-RNN

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_{\phi}} \left[ \log \frac{p_{\theta}(y_{1:T}, \mathbf{z}_{1:T}, \mathbf{h}_{1:T} | \mathbf{x}_{1:T}, \mathbf{h}_0)}{q_{\phi}(\mathbf{z}_{1:T}, \mathbf{h}_{1:T} | y_{1:T}, \mathbf{x}_{1:T}, \mathbf{h}_0)} \right]. \quad (5.17)$$

Here the expectation is with respect to the complete variational distribution in the denominator. Utilising the factorisations in Equations (5.13) and (5.15), we obtain

$$\mathcal{L}(\theta, \phi) = \sum_{t=1}^T \mathbb{E}_{q_{\phi}} \left[ \log \frac{p_{\theta}(y_t | \mathbf{z}_t) p_{\theta}(\mathbf{z}_t, \mathbf{h}_t)}{q_{\phi}(\mathbf{z}_t | y_t, \mathbf{h}_t)} \right]. \quad (5.18)$$

We can write this in the same form as the ELBO in (3.14)

$$\begin{aligned} \mathcal{L}(\theta, \phi) &= \sum_{t=1}^T \mathbb{E}_{q_{\phi}(\mathbf{z}_t | y_t, \mathbf{h}_t)} [\log p_{\theta}(y_t | \mathbf{z}_t)] \\ &\quad - D_{KL}(q_{\phi}(\mathbf{z}_t | y_t, \mathbf{h}_t) || p_{\theta}(\mathbf{z}_t, \mathbf{h}_t)). \end{aligned} \quad (5.19)$$

Thus minimising the negative log-likelihood<sup>10</sup> with respect to the parameter, including the variational parameters  $\phi$  defines our optimisation problem.

## Discussion

Deep state-space models enhance the temporal modelling ability of state-space models with the flexible representations of deep neural networks. Thus the model allows for the capturing of complex dynamics of real-world sequential data. Furthermore, its probabilistic prediction enable quantification of uncertainty which is a crucial aspect in many applications. However, these models do not come without their limitations. Defining all distributions with Gaussians might not be correct and lead to suboptimal performance. Furthermore, the increased complexity compared to state-space models affects the model's interpretability and sample complexity for training.

---

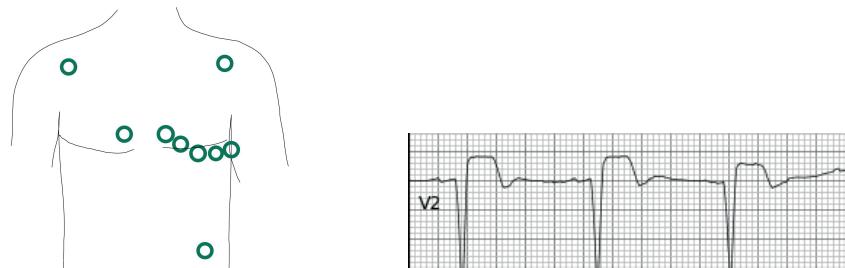
<sup>10</sup>Since  $-\log p_{\theta} \leq \mathcal{L}(\theta, \phi)$  is now an upper bound on the negative log-likelihood, minimising of this reduces the upper bound.

# CHAPTER 6

## Medical applications

This chapter investigates how we can utilise tools from deep learning in the medical field. We will focus specifically on data from the electrocardiogram (ECG) and deep learning-based prediction models for the decision support of physicians when handling ECG data in different scenarios. Researchers have to be particularly careful when handling medical data and promoting automated analysis of medical data through machine learning and prediction models [OE16; CSM18]. Biases in medical datasets through the collection process can lead to unfair treatment and reinforcement of systematical ethical problems [Abr+23; Omi+23; Che+23]. While addressing these issues is particularly important in the medical domain due to the impact on the treatment of human life, the problem of biased datasets and algorithmic biases extends beyond healthcare [Pau+21; Bir21]. Neglecting to address biases through ethical considerations can not only perpetuate disparities in healthcare and propagate existing socio-economic inequities but also erode trust in the deployment of machine learning-based solutions across a wider domain of applications. This necessitates robust validation processes, ethical considerations, and close collaboration with medical professionals to ensure the reliability and safety of such applications. We will not discuss these considerations in more detail but rather refer to existing literature from experts on the topic [Flo+18; JIV19; Dig19]. Thus, we consider machine learning-based medical prediction models not as a fully automated system but rather as a decision support tool for the physician who makes the final decision on the treatment of a patient. To reach this decision, it is imperative to be able to interpret the prediction from such a decision support tool. We will look into interpretability and explainability for deep learning systems and discuss how they relate to low-dimensional representations which neural networks learn.

Let us focus on data from the ECG [Wal87] as the specific medical application in this chapter [ML88; AL12]. The ECG is a diagnostic tool for cardiovascular diseases. It records the electric activity of the heart from multiple electrodes or leads placed on the skin of the patient, see Figure 6.1 (left). The result is a graph of the recorded voltage over time, see Figure 6.1 (right). Cardiologists analyse the resulting ECG data to identify abnormalities in the ECG pattern as



**Figure 6.1:** Left: Placement of leads on the human body for recording the 12-lead ECG. Right: an ECG trace, specifically measurements from lead V2.

deviations from the typical waveform and timing. The goal is the diagnosis of potential cardiac conditions. Abnormalities include arrhythmia<sup>1</sup> and ischemia<sup>2</sup> among others. ECGs are recorded in various scenarios including routine health screenings, evaluation of chest pain as a result of cardiac events, monitoring e.g. during surgery, and assessment of drug effects.

The ECG is a widely adopted diagnostic tool, which allows for fast, low-cost, non-invasive diagnosis. It is established in primary and specialised healthcare settings and allows for remote analysis [Alk+19]. The importance of ECG in clinical practice is underlined by the fact that cardiovascular diseases are the leading cause of death [Rot+18]. Thus, reliable and quick analysis is essential for high-quality care and to improve patient outcome. This is especially helpful in situations where quick decisions are necessary such as in the chaotic environment of an emergency department [Wri+19] where diagnostic error is common [Med+16].

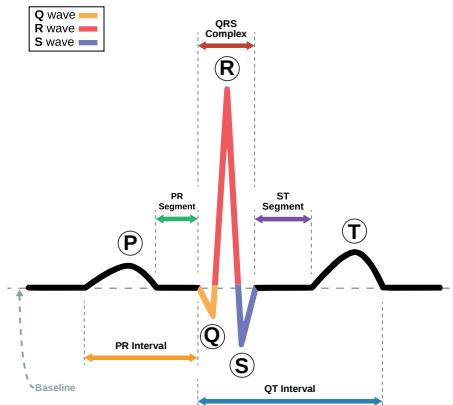
## 6.1 Modelling of electrocardiograms

ECGs can be recorded using single-lead, 3-lead, 5-lead or 12-lead<sup>3</sup> configurations, with the 12-lead setting being the clinically most prevalent one as it captures the most information about the cardiac system. Within this dissertation, we consider 12-lead ECG as the major type of ECG in clinical settings. Modern ECGs are mainly recorded with 400 – 1000 Hz. A standard 12-lead 10-second ECG thus results in a size of data samples  $x$  of  $8 \times 4000$  assuming 400 Hz sampling rate. Often various pre-processing steps are applied to

<sup>1</sup>Arrhythmias are irregularities in the rhythm of the heartbeat.

<sup>2</sup>Ischemia is a loss or restriction of blood flow.

<sup>3</sup>Note that in the 12-lead ECG, only 8 leads are independent. The remaining four leads are linear combinations of the recorded leads. The eight independent leads are  $I$ ,  $II$ ,  $V1$  to  $V6$  and the four computed ones are  $III = II - I$ ,  $aVR = -(I + II)/2$ ,  $aVL = I - II/2$ ,  $aVF = II - I/2$ .



**Figure 6.2:** One sinus rhythm from an ECG split into different segments. Image from Wikimedia Commons [Wik22].

the raw signal [Mor77]. This includes the removal of low-frequency trends and biases, so-called baseline wanders [LH05; BWB08]. Additionally, we can remove high-frequency noise elements such as muscle tremors or distortions from other sources such as power line noise. The ECG has a frequency range of 0.05 to 150 Hz [Kli+07] which has to be considered when filtering out noise components.

Modelling of the ECG for computer-assisted diagnosis prediction has a long history, see Macfarlane and Kennedy [MK21] for an overview. An important step for computerised analysis is the detection of key features in the ECG signal. From Figure 6.1 (right), we can observe the periodic behaviour of the ECG guided by the heartbeat. Cardiologists have segmented the ECG into distinct components, see Figure 6.2. Of particular interest are the P-wave, the QRS complex and the T-wave. Different algorithms based on signal processing techniques are proposed for this task [MAA05; MZ07]. Once key points and features are identified, they are utilised in classification algorithms to predict abnormalities in the ECG [MM01; Cli+17].

More recently, collections of large datasets of ECG with expert annotations [Gol+00; Wag+20] allowed for end-to-end training of deep neural networks for diagnosis prediction. Most notably Hannun et al. [Han+19] and Ribeiro et al. [Rib+20b] proposed deep neural network architectures for processing of raw ECG signals to obtain medical prediction models with high performance—even cardiologist-level performance is reached. Both of these methods are based on convolutional neural networks, more precisely a form of ResNet [He+16]. The convolutional model architecture allows for relevant ECG features to appear at different times within the ECG signal due to its transla-

tion invariance. Furthermore, temporal dependencies can be effectively captured through the locality in time provided by the convolutional filters. While convolutional-based neural networks are currently the dominant architecture type with many works being proposed, there are alternative architectures suggested for various specific medical tasks [Ach+17; XL20; Lim+21]. This includes recurrent neural networks [LPK19], transformer architectures [Che+21; Men+22], and hybrid approaches [Zha+20].

Let us briefly highlight the range of use cases for modelling ECGs with deep neural networks by papers that the author has contributed to which are not within this dissertation. Deep neural networks are able to diagnose a wide range of heart abnormalities from the ECG. In Ribeiro et al. [Rib+20a] we present a method based on Ribeiro et al. [Rib+20b] which can diagnose 27 different classes through an ensemble of models. Jidling et al. [Jid+23] provides a model that allows us to identify Chagas disease, a neglected tropical disease. While Chagas has a direct effect on the heart only in its late stage, we show promising results for early detection of Chagas disease that allows more detailed screening of patients. Similarly, in Von Bachmann et al. [Von+24] we study an indirect influence on the heart through blood electrolyte concentration. This depicts a regression task and we elaborate on uncertainty quantification for various types of electrolytes. Habineza et al. [Hab+23] studies risk prediction for the development of atrial fibrillation. For this task, we utilise the outcome of a deep neural network into a model for survival analysis to provide risk prediction for patients. Furthermore, Paper VI [Gus+22] shows how to classify specific types of myocardial infarctions<sup>4</sup>. While there are known patterns in the ECG for cardiologists to diagnose ST-elevation myocardial infarctions, there are no known patterns for the so-called non-ST-elevation myocardial infarction. The paper shows that deep neural networks are able to classify both types of myocardial infarctions with high performance.

However, automated analysis of ECGs with machine learning methods has its limitations [SW17]. Challenges remain in achieving robustness and generalisation across diverse patient populations and clinical settings without perpetuating biases encoded in the training datasets. In practical setups such as emergency care, we not only require high-performing predictive models but also insights into their decision-making process so that practising physicians can evaluate the relevance of the prediction for practical care.

## 6.2 Interpretability and explainability

Many techniques to enhance the interpretability and explainability of deep neural networks have been proposed. Such methods are particularly relevant for

---

<sup>4</sup>Myocardial infarctions are also known as heart attacks.

critical applications such as decision in medical treatment since a lack of understanding can reduce trust in the model. See Petch et al. [PDN22] for an overview of methods with a focus on cardiology. Let us first discuss the meaning of the terms interpretability and explainability.

Machine learning models are interpretable if they can be understood by humans. This means that we can trace back how the prediction was made. Therefore, interpretable machine learning models are for example linear regression or decision trees [Mol20]. In contrast, black box models such as neural networks with millions of parameters cannot be interpreted the same way. Explainable machine learning are tools that provide insight into the model to make the predictions understandable to humans [AB18; LPK20]. However, there is no consensus on a clear definition of both fields and the terms are often used interchangeably. Since this dissertation concerns deep neural networks, we will focus on explainability methods.

Popular methods include the following: LIME<sup>5</sup> which perturbs the input data e.g. by removing certain parts to provide local explanations [RSG16]. SHAP<sup>6</sup> compute Shapley values for each feature. Shapley values originate in game theory and measure a player’s contribution to the game. Thus SHAP provides the contribution of each feature to the prediction [LL17]. Integrated gradients consider a baseline input and compute for each feature the integral of the path to the actual input to provide a score for the importance of the feature [STY17]. DeepLIFT<sup>7</sup> considers the activations of each neuron and compares it to the activation of a baseline input to assign importance scores to input features [SGK17]. Saliency maps highlight regions of the input that are most influential for a model’s decision-making by computing gradients of the model’s output w.r.t. the input to provide feature importance [SVZ14b]. Grad-CAM<sup>8</sup> can be seen as an extension of Saliency maps for specific layers in a convolutional neural network [Sel+17]. There also exists toolboxes for simplified analysis of explainable AI algorithms such as Quantus [Hed+23].

While most methods consider the relation of model inputs to their predictions and therefore no intermediate model representations, Grad-CAM is computed for a specific layer of a convolutional network. It therefore focuses on intermediate representations. Specifically, Grad-CAM obtains feature importance in a two-step procedure:

1. Forward pass: compute the representations  $z$  obtained after a chosen convolutional layer.
2. Backward pass: compute the gradient w.r.t. these representations.

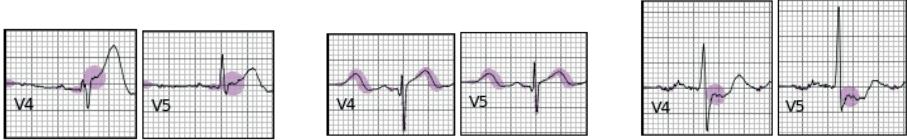
---

<sup>5</sup>Local Interpretable Model-agnostic Explanations.

<sup>6</sup>SHapley Additive exPlanations.

<sup>7</sup>Deep Learning Important FeaTures.

<sup>8</sup>Gradient-weighted Class Activation Mapping.



**Figure 6.3:** Importance maps of Grad-CAM computed on three ECG examples. Adapted from Paper VI Gustafsson et al. [Gus+22]. Left and middle are correctly predicted ECG with STEMI and right is a correctly predicted ECG with NSTEMI diagnosis.

Then, the gradients  $g$  are averaged across the representations of the chosen layer to obtain weights

$$\alpha_k = \frac{1}{K} \sum_{k=1}^K g_k, \quad (6.1)$$

where  $K$  is the number of channels in  $z$  after the convolutional layer. Finally, the class activation map  $L$  is given by computing a weighted linear combination of the representation. Positive values are then plotted

$$L = \text{ReLU} \left( \sum_k^{n_z} \alpha_k g_k \right). \quad (6.2)$$

As an example, we show Grad-CAM activation maps  $L$  for three examples of ECGs in Figure 6.3. The purple discs show the parts which are most important for the prediction.

How does this method connect to low-dimensional representations? In Section 3.1 we discussed that the representations which are learnt by convolutional neural networks can be visualised. Similarly, we can view the explainable activation maps  $L$  of Grad-CAM as visual, human-interpretable representations of the most important features extracted by the network at specific layers. Thus we obtain an insight into the lower dimensional representation within the neural network. This insight allows medical practitioners to partly understand the inner workings of the neural network and what inputs drive the decision-making of its prediction.

# Concluding remarks

This chapter encompasses remarks on the complete dissertation including Chapters 1 to 6 of Part A and the six attached research papers in Part B. We have argued and outlined how deep neural networks can be used to learn low-dimensional representations from data and why those representations are useful. We observe that real-world data are high-dimensional but are generated from a small set of factors. Thus, learning low-dimensional representations in deep neural networks has the benefit of eventually exploiting the underlying true structure of the data.

## Conclusion

Let us review the research questions with which Chapter 1 was started. The first one is more fundamental, stated as the following.

### RQ 1. What fundamental observations can we make when using machine learning approaches for low-dimensional data structures?

We introduced modelling with deep neural networks as well as with kernels in Chapter 2. While kernels offer a flexible approach of modelling, we saw that their ability to learn features in a data-driven approach is limited. Therefore, we discuss how we can translate the success of deep neural networks with their ability to learn features for kernel machines. The framework of Recursive Feature Machines provides this missing link. In Paper I [Ged+24] we show how the RFM can be embedded into the general framework of Gaussian processes. We see that for synthetic toy data and real-world applications, the features that the RFM learns are low-dimensional. Thus there is an intricate relation between the learning algorithms provided by the Average Gradient Outer Product in the RFM and the underlying structure of the data as revealed by low-dimensional features.

Next, in Chapter 3, we discuss different approaches for learning low-dimensional representations in deep neural networks. For the unsupervised case, the auto-encoding framework of embedding observations in a low-

dimensional space and reconstructing them from this space is especially intriguing. Therefore, in Paper II [Ged+23], we propose an algorithm for the reconstruction of inputs that are encoded with kernel PCA. The algorithm provides an approach to first encode high-dimensional observations through non-linear mappings by kernel PCA into a low-dimensional space. Then we can reconstruct the original input without the need to solve a supervised optimisation problem. The quality of the reconstructions indicates that the low-dimensional representations describe the data sufficiently well.

Finally, in Chapter 4, we dive into theoretical aspects of how well a machine learning model generalises to unseen data as measured by the generalisation risk. Specifically, we discuss the setup when observations are generated linearly from a low-dimensional set of factors. We analyse this data generator to precisely characterise the risk of the principal component regression model for the phenomenon known as double descent. In Paper III [GRS24] we provide the in-depth analysis. We observe that if we choose the number of principal components close to the true number of factors that generate the data, i.e. if the model replicates the data-generating process, the risk is well behaved and we avoid the interpolation peak.

The second research question focuses more on applications and their connection to learning low-dimensional representations. We stated it as the following.

**RQ 2. How can we utilise low-dimensional representations for deep learning in applications?**

The first application or rather field where we can apply deep neural networks is the modelling of dynamical systems, described in Chapter 5. We describe the relation to encoding dynamics in low-dimensional representations through classical models such as state-space models and extend it for deep state-space models in Paper IV [Ged+21b]. We describe how these models are temporal extensions of the VAE and therefore borrow from the strength of representing data in a low-dimensional space. The experimental observations indicate the power of this model construction for modelling time sequences accurately while also being able to capture uncertainty.

Second, we discuss the use of deep neural networks in medical applications with a focus on electrocardiograms in Chapter 6. This connects to Paper V [Ged+21a] which extracts low-dimensional representations using self-supervised learning and demonstrates their effectiveness for downstream classification tasks. Furthermore, we discuss the connection of interpretability and explainability tools to low-dimensional representations. Grad-CAM provides such a view into the inner workings of deep neural networks and how the model makes its predictions. This tool was utilised in Paper VI [Gus+22] to evaluate the predictions of a medical prediction model for the automated analysis of ECGs.

## Future Work

Each chapter of this dissertation and every paper raises its individual questions which are discussed in the respective papers. Here we try to zoom out on the bigger picture to provide an outlook on the research that inspired this dissertation.

Initially, we motivated the learning of low-dimensional representations with the observation that many real-world data are governed by data-generating mechanisms which have an intrinsic small set of factors. The ultimate goal would be to learn models that can uncover this structure. We will discuss two avenues which can address this challenge.

**Mechanistic machine learning.** We discussed that modelling of real-world observations is often based on our understanding of the system to include knowledge and model by first principles. In many cases, we can describe the processes through ordinary or partial differential equations that embed our knowledge of the system. However, there are instances where the process is too complex. Then we can rely on data-driven modelling tools. The field of physics-informed neural networks [RPK19], incorporates principles from physics into neural networks and therefore enhances their performance and interpretability. This essentially allows us to uncover the data-generating mechanism and better understand the system that we model.

**Causal representation learning.** This field combines the idea of causal discovery with the power of representation learning in deep neural networks. Causal discovery tries to identify the causal variables which generated the observations and ultimately to identify the data-generating mechanism. It relies on the idea that some variables are causally related to others and therefore have an effect which we will observe. Schölkopf et al. [Sch+21] provides an overview of the topic and raises points to address.



# References

- [AB18] A. Adadi and M. Berrada. “Peeking inside the black-box: a survey on explainable artificial intelligence (XAI).” In: *IEEE access* 6 (2018), pp. 52138–52160 (cit. on p. 73).
- [ABP23] A. Abedsoltan, M. Belkin, and P. Pandit. “Toward Large Kernel Models.” In: *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202. Proceedings of Machine Learning Research. 2023, pp. 61–78 (cit. on p. 27).
- [Abr+23] M. D. Abràmoff, M. E. Tarver, N. Loyo-Berrios, S. Trujillo, D. Char, Z. Obermeyer, M. B. Eydelman, and W. H. Maisel. “Considerations for addressing bias in artificial intelligence for health equity.” In: *npj Digital Medicine* 6.1 (2023) (cit. on p. 69).
- [Ach+17] U. R. Acharya, H. Fujita, S. L. Oh, Y. Hagiwara, J. H. Tan, and M. Adam. “Application of deep convolutional neural network for automated detection of myocardial infarction using ECG signals.” In: *Information Sciences* 415 (2017), pp. 190–198 (cit. on p. 72).
- [AL12] M. AlGhatrif and J. Lindsay. “A brief review: history to understand fundamentals of electrocardiography.” In: *Journal of community hospital internal medicine perspectives* 2.1 (2012), p. 14383 (cit. on p. 69).
- [Ali+17] A. G. Alias Parth Goyal, A. Sordoni, M.-A. Côté, N. R. Ke, and Y. Bengio. “Z-Forcing: Training Stochastic Recurrent Networks.” In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 6713–6723 (cit. on p. 65).
- [Alk+19] M. B. Alkmim, C. B. G. Silva, R. M. Figueira, D. V. V. Santos, L. B. Ribeiro, M. C. da Paixão, M. S. Marcolino, J. C. Paiva, and A. L. Ribeiro. “Brazilian National Service of telediagnosis in Electrocardiography.” In: *Stud. Health Technol. Inform.* 264 (2019), pp. 1635–1636 (cit. on p. 70).
- [And+19] C. Andersson, A. H. Ribeiro, K. Tiels, N. Wahlström, and T. B. Schön. “Deep convolutional networks in system identification.” In: *2019 IEEE 58th conference on decision and control (CDC)*. IEEE. 2019, pp. 3670–3676 (cit. on p. 63).
- [Aro50] N. Aronszajn. “Theory of reproducing kernels.” In: *Transactions of the American mathematical society* 68.3 (1950), pp. 337–404 (cit. on p. 21).

- [AS18] A. Achille and S. Soatto. “Emergence of invariance and disentanglement in deep representations.” In: *The Journal of Machine Learning Research* 19.1 (2018), pp. 1947–1980 (cit. on p. 41).
- [ASS20] M. S. Advani, A. M. Saxe, and H. Sompolinsky. “High-dimensional dynamics of generalization error in neural networks.” In: *Neural Networks* 132 (2020), pp. 428–446 (cit. on p. 49).
- [Bah+21] Y. Bahri, E. Dyer, J. Kaplan, J. Lee, and U. Sharma. “Explaining neural scaling laws.” In: *arXiv preprint arXiv:2102.06701* (2021) (cit. on p. 47).
- [BAP05] J. Baik, G. B. Arous, and S. Péché. “Phase transition of the largest eigenvalue for nonnull complex sample covariance matrices.” In: *The Annals of Probability* 33.5 (2005), pp. 1643–1697 (cit. on p. 55).
- [Bar+20] P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler. “Benign overfitting in linear regression.” In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30063–30070 (cit. on pp. 49, 51, 57).
- [BCB15] D. Bahdanau, K. Cho, and Y. Bengio. “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*. 2015 (cit. on pp. 19, 64).
- [BCV13] Y. Bengio, A. Courville, and P. Vincent. “Representation learning: A review and new perspectives.” In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828 (cit. on pp. 1, 32).
- [Bel+19] M. Belkin, D. Hsu, S. Ma, and S. Mandal. “Reconciling modern machine-learning practice and the classical bias–variance trade-off.” In: *Proceedings of the National Academy of Sciences* 116.32 (2019), pp. 15849–15854 (cit. on pp. 48, 49).
- [Bel61] R. Bellman. *Adaptive control processes*. Princeton, NJ: Princeton University Press, 1961 (cit. on p. 31).
- [BFS21a] F. Bonassi, M. Farina, and R. Scattolini. “On the stability properties of gated recurrent units neural networks.” In: *Systems & Control Letters* 157 (2021), p. 105049 (cit. on p. 64).
- [BFS21b] F. Bonassi, M. Farina, and R. Scattolini. “Stability of discrete-time feed-forward neural networks in NARX configuration.” In: *IFAC-PapersOnLine* 54.7 (2021), pp. 547–552 (cit. on p. 64).

- [BHB19] P. Bachman, R. D. Hjelm, and W. Buchwalter. “Learning Representations by Maximizing Mutual Information Across Views.” In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019 (cit. on p. 43).
- [BHX20] M. Belkin, D. Hsu, and J. Xu. “Two models of double descent for weak features.” In: *SIAM Journal on Mathematics of Data Science* 2.4 (2020), pp. 1167–1180 (cit. on pp. 49, 51).
- [Bin+21] X. Bing, F. Bunea, S. Strimas-Mackey, and M. Wegkamp. “Prediction under latent factor regression: Adaptive PCR, interpolating predictors and beyond.” In: *The Journal of Machine Learning Research* 22.1 (2021), pp. 7994–8043 (cit. on p. 55).
- [Bir21] A. Birhane. “The Impossibility of Automating Ambiguity.” In: *Artificial Life* 27.1 (2021), pp. 44–61 (cit. on p. 69).
- [Bis06] C. M. Bishop. “Pattern Recognition and Machine Learning (Information Science and Statistics).” In: (2006) (cit. on p. 31).
- [BKK18] S. Bai, J. Z. Kolter, and V. Koltun. “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.” In: *arXiv preprint arXiv:1803.01271* (2018) (cit. on p. 63).
- [BKM17] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. “Variational inference: A review for statisticians.” In: *Journal of the American statistical Association* 112.518 (2017), pp. 859–877 (cit. on p. 39).
- [Blu+15] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. “Weight uncertainty in neural network.” In: *International conference on machine learning*. PMLR. 2015, pp. 1613–1622 (cit. on p. 25).
- [BM02] P. L. Bartlett and S. Mendelson. “Rademacher and Gaussian complexities: Risk bounds and structural results.” In: *Journal of Machine Learning Research* 3.Nov (2002), pp. 463–482 (cit. on pp. 48, 51).
- [BMM18] M. Belkin, S. Ma, and S. Mandal. “To understand deep learning we need to understand kernel learning.” In: *International Conference on Machine Learning*. PMLR. 2018, pp. 541–549 (cit. on p. 49).
- [BMR21] P. L. Bartlett, A. Montanari, and A. Rakhlin. “Deep learning: a statistical viewpoint.” In: *Acta Numerica* 30 (2021), pp. 87–201 (cit. on p. 49).
- [BN03] M. Belkin and P. Niyogi. “Laplacian eigenmaps for dimensionality reduction and data representation.” In: *Neural computation* 15.6 (2003), pp. 1373–1396 (cit. on p. 38).

- [BO14] J. Bayer and C. Osendorfer. “Learning Stochastic Recurrent Networks.” In: *CoRR* abs/1411.7610 (2014) (cit. on p. 65).
- [Bon+20] F. Bonassi, E. Terzi, M. Farina, and R. Scattolini. “LSTM neural networks: Input to state stability and probabilistic safety verification.” In: *Learning for Dynamics and Control*. PMLR. 2020, pp. 85–94 (cit. on p. 64).
- [BPR07] F. Bauer, S. Pereverzev, and L. Rosasco. “On regularization algorithms in learning theory.” In: *Journal of complexity* 23.1 (2007), pp. 52–72 (cit. on p. 57).
- [Bro+20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. “Language models are few-shot learners.” In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901 (cit. on p. 44).
- [BS06] J. Baik and J. W. Silverstein. “Eigenvalues of large sample covariance matrices of spiked population models.” In: *Journal of multivariate analysis* 97.6 (2006), pp. 1382–1408 (cit. on p. 56).
- [BS10] Z. Bai and J. W. Silverstein. *Spectral analysis of large dimensional random matrices*. Vol. 20. Springer, 2010 (cit. on p. 51).
- [Bur+17] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. “Understanding disentangling in beta-vae.” In: *Workshop on Learning Disentangled Representations at the 31st Conference on Neural Information Processing Systems*. 2017 (cit. on p. 40).
- [Bur96] C. J. C. Burges. “Simplified Support Vector Decision Rules.” In: *International Conference on Machine Learning*. 1996, pp. 71–77 (cit. on p. 37).
- [WBW08] M. Blanco-Velasco, B. Weng, and K. E. Barner. “ECG signal denoising and baseline wander correction based on the empirical mode decomposition.” In: *Computers in biology and medicine* 38.1 (2008), pp. 1–13 (cit. on p. 71).
- [BWS04] G. H. Bakir, J. Weston, and B. Schölkopf. “Learning to find pre-images.” In: *Advances in neural information processing systems* 16 (2004), pp. 449–456 (cit. on p. 37).
- [Cal+16] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. “Manifold Gaussian processes for regression.” In: *2016 International joint conference on neural networks (IJCNN)*. IEEE. 2016, pp. 3338–3345 (cit. on p. 28).

- [Car+18] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. “Deep clustering for unsupervised learning of visual features.” In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 132–149 (cit. on p. 43).
- [Car+20] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. “Unsupervised learning of visual features by contrasting cluster assignments.” In: *Advances in neural information processing systems* 33 (2020), pp. 9912–9924 (cit. on p. 43).
- [Car+21] M. Caron, H. Tovvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. “Emerging properties in self-supervised vision transformers.” In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 9650–9660 (cit. on p. 43).
- [Che+16] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. “Infogan: Interpretable representation learning by information maximizing generative adversarial nets.” In: *Advances in neural information processing systems* 29 (2016) (cit. on p. 41).
- [Che+18] R. T. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud. “Isolating sources of disentanglement in variational autoencoders.” In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 40).
- [Che+20a] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. “A simple framework for contrastive learning of visual representations.” In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607 (cit. on pp. 1, 19, 43).
- [Che+20b] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton. “Big self-supervised models are strong semi-supervised learners.” In: *Advances in neural information processing systems* 33 (2020), pp. 22243–22255 (cit. on p. 43).
- [Che+20c] X. Chen, H. Fan, R. Girshick, and K. He. “Improved baselines with momentum contrastive learning.” In: *arXiv preprint arXiv:2003.04297* (2020) (cit. on p. 43).
- [Che+21] C. Che, P. Zhang, M. Zhu, Y. Qu, and B. Jin. “Constrained transformer network for ECG signal processing and arrhythmia classification.” In: *BMC Medical Informatics and Decision Making* 21.1 (2021), p. 184 (cit. on p. 72).
- [Che+23] R. J. Chen, J. J. Wang, D. F. K. Williamson, T. Y. Chen, J. Lipkova, M. Y. Lu, S. Sahai, and F. Mahmood. “Algorithmic fairness in artificial intelligence for medicine and healthcare.” In: *Nature Biomedical Engineering* 7.6 (2023), pp. 719–742 (cit. on p. 69).

- [Che52] H. Chernoff. “A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations.” In: *The Annals of Mathematical Statistics* 23.4 (1952), pp. 493–507 (cit. on p. 51).
- [Cho+14] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.” In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1724–1734 (cit. on pp. 19, 63).
- [Chu+15] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. “A Recurrent Latent Variable Model for Sequential Data.” In: *Advances in Neural Information Processing Systems* 28. 2015, pp. 2980–2988 (cit. on p. 65).
- [CJS24] A. Curth, A. Jeffares, and M. van der Schaar. “A u-turn on double descent: Rethinking parameter counting in statistical learning.” In: *Advances in Neural Information Processing Systems* 36 (2024) (cit. on p. 58).
- [Cli+17] G. D. Clifford, C. Liu, B. Moody, H. L. Li-wei, I. Silva, Q. Li, A. Johnson, and R. G. Mark. “AF classification from a short single lead ECG recording: The PhysioNet/computing in cardiology challenge 2017.” In: *2017 Computing in Cardiology (CinC)*. IEEE. 2017, pp. 1–4 (cit. on p. 71).
- [CMR20] W. Cao, V. Mirjalili, and S. Raschka. “Rank consistent ordinal regression for neural networks with application to age estimation.” In: *Pattern Recognition Letters* 140 (2020), pp. 325–331 (cit. on p. 25).
- [Cop43] N. Copernicus. *De revolutionibus orbium coelestium*. Available in various editions and translations. Nuremberg: Johann Petreius, 1543 (cit. on p. 2).
- [CSM18] D. S. Char, N. H. Shah, and D. Magnus. “Implementing machine learning in health care—addressing ethical challenges.” In: *The New England journal of medicine* 378.11 (2018), p. 981 (cit. on p. 69).
- [CSZ06] *Semi-Supervised Learning*. The MIT Press, 2006 (cit. on p. 19).
- [dAs+20] S. d’Ascoli, M. Refinetti, G. Biroli, and F. Krzakala. “Double trouble in double descent: Bias and variance (s) in the lazy regime.” In: *International Conference on Machine Learning*. PMLR. 2020, pp. 2280–2290 (cit. on p. 49).

- [Dax+21] E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig. “Laplace redux-effortless bayesian deep learning.” In: *Advances in Neural Information Processing Systems 34* (2021), pp. 20089–20103 (cit. on p. 25).
- [DDG01] A. Doucet, N. De Freitas, and N. Gordon. “An introduction to sequential Monte Carlo methods.” In: *Sequential Monte Carlo methods in practice* (2001), pp. 3–14 (cit. on p. 66).
- [Den+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “Imagenet: A large-scale hierarchical image database.” In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255 (cit. on p. 1).
- [Dev+19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186 (cit. on p. 44).
- [DFH17] L. H. Dicker, D. P. Foster, and D. Hsu. “Kernel ridge vs. principal component regression: Minimax bounds and the qualification of regularization operators.” In: *Electronic Journal of Statistics* 11.1 (2017), pp. 1022–1047 (cit. on p. 57).
- [DG17] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2017 (cit. on p. 36).
- [DGE15] C. Doersch, A. Gupta, and A. A. Efros. “Unsupervised visual representation learning by context prediction.” In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1422–1430 (cit. on p. 42).
- [DHS11] J. Duchi, E. Hazan, and Y. Singer. “Adaptive subgradient methods for online learning and stochastic optimization.” In: *Journal of machine learning research* 12.7 (2011) (cit. on p. 19).
- [Dig19] V. Dignum. *Responsible artificial intelligence: how to develop and use AI in a responsible way*. Vol. 1. Springer, 2019 (cit. on p. 69).
- [Dit+21] A. Dittadi, F. Träuble, F. Locatello, M. Wuthrich, V. Agrawal, O. Winther, S. Bauer, and B. Schölkopf. “On the Transfer of Disentangled Representations in Realistic Settings.” In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021 (cit. on p. 41).

- [DKT22] Z. Deng, A. Kammoun, and C. Thrampoulidis. “A model of double descent for high-dimensional binary linear classification.” In: *Information and Inference: A Journal of the IMA* 11.2 (2022), pp. 435–495 (cit. on p. 57).
- [DM19] R. Diaz and A. Marathe. “Soft labels for ordinal regression.” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4738–4747 (cit. on p. 25).
- [Doe13] G. Doetsch. *Theorie und Anwendung der Laplace-transformation*. Vol. 67. Springer-Verlag, 2013 (cit. on p. 60).
- [Doe16] C. Doersch. “Tutorial on variational autoencoders.” In: *arXiv preprint arXiv:1606.05908* (2016) (cit. on p. 40).
- [Dos+21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.” In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021 (cit. on pp. 19, 44).
- [DR17] G. K. Dziugaite and D. M. Roy. “Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data.” In: *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017 (cit. on p. 49).
- [Ein16] A. Einstein. “Die Grundlage der allgemeinen Relativitätstheorie.” In: *Annalen der Physik* 354.7 (1916), pp. 769–822 (cit. on p. 3).
- [Elm90] J. L. Elman. “Finding structure in time.” In: *Cognitive science* 14.2 (1990), pp. 179–211 (cit. on p. 19).
- [Erh+10] D. Erhan, A. Courville, Y. Bengio, and P. Vincent. “Why does unsupervised pre-training help deep learning?” In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 201–208 (cit. on p. 19).
- [EW18] C. Eastwood and C. K. Williams. “A framework for the quantitative evaluation of disentangled representations.” In: *International conference on learning representations*. 2018 (cit. on p. 40).
- [FAK15] O. Fabius, J. R. van Amersfoort, and D. P. Kingma. “Variational Recurrent Auto-Encoders.” In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*. 2015 (cit. on p. 65).

- [Flo+18] L. Floridi, J. Cowls, M. Beltrametti, R. Chatila, P. Chazerand, V. Dignum, C. Luetge, R. Madelin, U. Pagallo, F. Rossi, et al. “AI4People—an ethical framework for a good AI society: opportunities, risks, principles, and recommendations.” In: *Minds and machines* 28 (2018), pp. 689–707 (cit. on p. 69).
- [Fra+16] M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther. “Sequential Neural Models with Stochastic Layers.” In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Barcelona, Spain, 2016, pp. 2207–2215 (cit. on p. 65).
- [Fra18] M. Fraccaro. “Deep Latent Variable Models for Sequential Data.” PhD thesis. Technical University of Denmark., 2018 (cit. on pp. 65, 66).
- [Gab46] D. Gabor. “Theory of communication. Part 1: The analysis of information.” In: *Journal of the Institution of Electrical Engineers-part III: radio and communication engineering* 93.26 (1946), pp. 429–441 (cit. on p. 34).
- [Gal16] Y. Gal. “Uncertainty in Deep Learning.” PhD thesis. University of Cambridge, 2016 (cit. on p. 24).
- [Gal38] G. Galilei. *Discorsi e dimostrazioni matematiche intorno a due nuove scienze*. Available in various editions and translations. Leiden: Lodewijk Elzevir, 1638 (cit. on p. 2).
- [Gar+18] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. “Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration.” In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 27).
- [GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016 (cit. on p. 1).
- [GBD92] S. Geman, E. Bienenstock, and R. Doursat. “Neural Networks and the Bias/Variance Dilemma.” In: *Neural Computation* 4.1 (1992), pp. 1–58 (cit. on p. 49).
- [GD23] A. Gu and T. Dao. “Mamba: Linear-time sequence modeling with selective state spaces.” In: *arXiv preprint arXiv:2312.00752* (2023) (cit. on p. 64).
- [GDS20] F. K. Gustafsson, M. Danelljan, and T. B. Schön. “Evaluating scalable bayesian deep learning methods for robust computer vision.” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 318–319 (cit. on p. 25).

- [Ged+21a] D. Gedon, A. H. Ribeiro, N. Wahlström, and T. B. Schön. “First Steps Towards Self-Supervised Pretraining of the 12-Lead ECG.” In: *Computing in Cardiology (CinC)*. Online, 2021 (cit. on pp. 7, 15, 20, 45, 76).
- [Ged+21b] D. Gedon, N. Wahlström, T. B. Schön, and L. Ljung. “Deep State Space Models for Nonlinear System Identification.” In: *19th IFAC Symposium on System Identification (SYSID)*. Padova, Italy, 2021 (cit. on pp. 7, 15, 20, 65, 76).
- [Ged+23] D. Gedon, A. H. Ribeiro, N. Wahlström, and T. B. Schön. “Invertible Kernel PCA with Random Fourier Features.” In: *IEEE Signal Processing Letters*. 2023 (cit. on pp. 6, 15, 20, 37, 76).
- [Ged+24] D. Gedon, A. Abedsoltan, T. B. Schön, and M. Belkin. “Uncertainty Estimation with Recursive Feature Machines.” Submitted. 2024 (cit. on pp. 6, 15, 30, 75).
- [GG16] Y. Gal and Z. Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning.” In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059 (cit. on p. 25).
- [GGR22] A. Gu, K. Goel, and C. Ré. “Efficiently Modeling Long Sequences with Structured State Spaces.” In: *The International Conference on Learning Representations (ICLR)*. 2022 (cit. on p. 64).
- [GH10] M. Gutmann and A. Hyvärinen. “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models.” In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 297–304 (cit. on p. 43).
- [Gir+21] L. Girin, S. Leglaive, X. Bie, J. Diard, T. Hueber, and X. Alameda-Pineda. “Dynamical Variational Autoencoders: A Comprehensive Review.” In: *Foundations and Trends® in Machine Learning* 15.1–2 (2021), pp. 1–175 (cit. on p. 65).
- [GN02] M. Girvan and M. E. Newman. “Community structure in social and biological networks.” In: *Proceedings of the national academy of sciences* 99.12 (2002), pp. 7821–7826 (cit. on p. 3).
- [Gol+00] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. “PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals.” In: *circulation* 101.23 (2000), e215–e220 (cit. on p. 71).

- [Goo+20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. “Generative adversarial networks.” In: *Communications of the ACM* 63.11 (2020), pp. 139–144 (cit. on pp. 1, 19).
- [Gop+21] B. Gopal, R. Han, G. Raghupathi, A. Ng, G. Tison, and P. Rajpurkar. “3KG: Contrastive learning of 12-lead electrocardiograms using physiologically-inspired augmentations.” In: *Machine Learning for Health*. PMLR. 2021, pp. 156–167 (cit. on p. 43).
- [GR98] J. C. Giarratano and G. Riley. *Expert systems*. PWS Publishing Co., 1998 (cit. on p. 3).
- [Gra11] A. Graves. “Practical variational inference for neural networks.” In: *Advances in neural information processing systems* 24 (2011) (cit. on p. 25).
- [Gri+20] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al. “Bootstrap your own latent-a new approach to self-supervised learning.” In: *Advances in neural information processing systems* 33 (2020), pp. 21271–21284 (cit. on p. 43).
- [GRS24] D. Gedon, A. H. Ribeiro, and T. B. Schön. “No Double Descent in Principal Component Regression: A High-Dimensional Analysis.” Submitted. 2024 (cit. on pp. 7, 20, 36, 49, 52, 55–57, 76).
- [GSK18] S. Gidaris, P. Singh, and N. Komodakis. “Unsupervised Representation Learning by Predicting Image Rotations.” In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018 (cit. on p. 42).
- [Gu+20] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré. “Hippo: Recurrent memory with optimal polynomial projections.” In: *Advances in neural information processing systems* 33 (2020), pp. 1474–1487 (cit. on p. 64).
- [Guo+17] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. “On calibration of modern neural networks.” In: *International conference on machine learning*. PMLR. 2017, pp. 1321–1330 (cit. on p. 23).
- [Gus+20] F. K. Gustafsson, M. Danelljan, G. Bhat, and T. B. Schön. “Energy-based models for deep probabilistic regression.” In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*. Springer. 2020, pp. 325–343 (cit. on p. 25).

- [Gus+22] S. Gustafsson, D. Gedon, E. Lampa, A. H. Ribeiro, M. J. Holzmann, T. B. Schön, and J. Sundström. “Development and validation of deep learning ECG-based prediction of myocardial infarction in emergency department patients.” In: *Scientific Reports*. Vol. 12. 2022 (cit. on pp. 7, 15, 20, 72, 74, 76).
- [Gus10] F. Gustafsson. “Particle filter theory and practice with positioning applications.” In: *IEEE Aerospace and Electronic Systems Magazine* 25.7 (2010), pp. 53–82 (cit. on p. 66).
- [GVP90] D. Geiger, T. Verma, and J. Pearl. “d-separation: From theorems to algorithms.” In: *Machine intelligence and pattern recognition*. Vol. 10. Elsevier, 1990, pp. 139–148 (cit. on p. 65).
- [Hab+23] T. Habineza, A. H. Ribeiro, D. Gedon, J. A. Behar, A. L. P. Ribeiro, and T. B. Schön. “End-to-end risk prediction of atrial fibrillation from the 12-Lead ECG by deep neural networks.” In: *Journal of Electrocardiology* 81 (2023), pp. 193–200 (cit. on p. 72).
- [Han+19] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng. “Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network.” In: *Nature medicine* 25.1 (2019), pp. 65–69 (cit. on p. 71).
- [Has+09] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009 (cit. on pp. 47, 48).
- [Has+22] T. Hastie, A. Montanari, S. Rosset, and R. J. Tibshirani. “Surprises in high-dimensional ridgeless least squares interpolation.” In: *The Annals of Statistics* 50.2 (2022), pp. 949–986 (cit. on pp. 49, 51–55, 57).
- [He+16] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. on pp. 1, 18, 19, 47, 71).
- [He+20] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. “Momentum contrast for unsupervised visual representation learning.” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 9729–9738 (cit. on pp. 19, 43).
- [Hed+23] A. Hedström, L. Weber, D. Krakowczyk, D. Bareeva, F. Motzkus, W. Samek, S. Lapuschkin, and M. M.-C. Höhne. “Quantus: An explainable ai toolkit for responsible evaluation of neural network explanations and beyond.” In: *Journal of Machine Learning Research* 24.34 (2023), pp. 1–11 (cit. on p. 73).

- [Hen+21] J. N. Hendriks, F. K. Gustafsson, A. H. Ribeiro, A. G. Wills, and T. B. Schön. “Deep energy-based NARX models.” In: *IFAC-PapersOnLine* 54.7 (2021), pp. 505–510 (cit. on p. 64).
- [Hig+17] I. Higgins, L. Matthey, A. Pal, C. P. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner. “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework.” In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017 (cit. on p. 40).
- [Hje+19] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. “Learning deep representations by mutual information estimation and maximization.” In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019 (cit. on p. 43).
- [HO00] A. Hyvärinen and E. Oja. “Independent component analysis: algorithms and applications.” In: *Neural networks* 13.4-5 (2000), pp. 411–430 (cit. on p. 40).
- [Hoe63] W. Hoeffding. “Probability Inequalities for Sums of Bounded Random Variables.” In: *Journal of the American Statistical Association* 58.301 (1963), pp. 13–30 (cit. on p. 51).
- [HS97] S. Hochreiter and J. Schmidhuber. “Long short-term memory.” In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on pp. 19, 63).
- [Hua+15] W. Huang, D. Zhao, F. Sun, H. Liu, and E. Chang. “Scalable Gaussian process regression using deep neural networks.” In: *Twenty-fourth international joint conference on artificial intelligence*. 2015 (cit. on p. 29).
- [HV93] G. E. Hinton and D. Van Camp. “Keeping the neural networks simple by minimizing the description length of the weights.” In: *Proceedings of the sixth annual conference on Computational learning theory*. 1993, pp. 5–13 (cit. on p. 25).
- [IS15] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” In: *International conference on machine learning*. pmlr. 2015, pp. 448–456 (cit. on p. 19).
- [Izm+18] P. Izmailov, D. Podoprikhin, T. Garipov, D. P. Vetrov, and A. G. Wilson. “Averaging Weights Leads to Wider Optima and Better Generalization.” In: *UAI*. AUAI Press, 2018, pp. 876–885 (cit. on p. 25).

- [Jai+20] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Make-don. “A survey on contrastive self-supervised learning.” In: *Technologies* 9.1 (2020), p. 2 (cit. on p. 42).
- [Jen06] J. L. W. V. Jensen. “Sur les fonctions convexes et les inégalités entre les valeurs moyennes.” In: *Acta mathematica* 30.1 (1906), pp. 175–193 (cit. on p. 39).
- [JGH18] A. Jacot, F. Gabriel, and C. Hongler. “Neural tangent kernel: Convergence and generalization in neural networks.” In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 50).
- [Jid+23] C. Jidling, D. Gedon, T. B. Schön, C. D. L. Oliveira, C. S. Cardoso, A. M. Ferreira, L. Giatti, S. M. Barreto, E. C. Sabino, A. L. Ribeiro, et al. “Screening for Chagas disease from the electrocardiogram using a deep neural network.” In: *PLoS Neglected Tropical Diseases* 17.7 (2023), e0011118 (cit. on p. 72).
- [JIV19] A. Jobin, M. Ienca, and E. Vayena. “The global landscape of AI ethics guidelines.” In: *Nature machine intelligence* 1.9 (2019), pp. 389–399 (cit. on p. 69).
- [Joa84] H. H. Joachim. “On generation and corruption.” In: (1984) (cit. on p. 2).
- [Joh+09] I. M. Johnstone, A. Y. Lu, B. Nadler, D. M. Witten, T. Hastie, R. Tibshirani, and J. O. Ramsay. “On Consistency and Sparsity for Principal Components Analysis in High Dimensions.” In: *Journal of the American Statistical Association* 104.486 (2009), pp. 682–703 (cit. on p. 56).
- [Joh01] I. M. Johnstone. “On the distribution of the largest eigenvalue in principal components analysis.” In: *Annals of statistics* 29.2 (2001), pp. 295–327 (cit. on pp. 54, 55).
- [Jol02] I. T. Jolliffe. *Principal component analysis for special types of data*. Springer, 2002 (cit. on p. 35).
- [JP18] I. M. Johnstone and D. Paul. “PCA in high dimensions: An orientation.” In: *Proceedings of the IEEE* 106.8 (2018), pp. 1277–1292 (cit. on p. 55).
- [Kap+20] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. “Scaling laws for neural language models.” In: *arXiv preprint arXiv:2001.08361* (2020) (cit. on p. 47).
- [KB15] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization.” In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015 (cit. on p. 19).

- [KG17] A. Kendall and Y. Gal. “What uncertainties do we need in bayesian deep learning for computer vision?” In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 24).
- [KH91] A. Krogh and J. Hertz. “A simple weight decay can improve generalization.” In: *Advances in neural information processing systems* 4 (1991) (cit. on p. 49).
- [KKH95] A. Katok, A. Katok, and B. Hasselblatt. *Introduction to the modern theory of dynamical systems*. 54. Cambridge university press, 1995 (cit. on p. 59).
- [Kli+07] P. Kligfield, L. S. Gettes, et al. “Recommendations for the Standardization and Interpretation of the Electrocardiogram: Part I: The Electrocardiogram and Its Technology: A Scientific Statement From the American Heart Association Electrocardiography and Arrhythmias Committee, Council on Clinical Cardiology; the American College of Cardiology Foundation; and the Heart Rhythm Society Endorsed by the International Society for Computerized Electrocardiology.” In: *Circulation* 115.10 (2007), pp. 1306–1324 (cit. on p. 71).
- [Kli+21] D. A. Klindt, L. Schott, Y. Sharma, I. Ustyuzhaninov, W. Brendel, M. Bethge, and D. M. Paiton. “Towards Nonlinear Disentanglement in Natural Data with Temporal Sparse Coding.” In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021 (cit. on p. 41).
- [KM18] H. Kim and A. Mnih. “Disentangling by factorising.” In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2649–2658 (cit. on p. 40).
- [KSB18] A. Kumar, P. Sattigeri, and A. Balakrishnan. “Variational Inference of Disentangled Latent Concepts from Unlabeled Observations.” In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018 (cit. on p. 40).
- [KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks.” In: *Advances in neural information processing systems* 25 (2012) (cit. on pp. 1, 19, 34).
- [KT03] J. T. Kwok and I. W. Tsang. “The pre-image problem in kernel methods.” In: *Proceedings of the 20th International Conference on Machine Learning*. 2003, pp. 408–415 (cit. on p. 37).

- [KW14] D. P. Kingma and M. Welling. “Auto-Encoding Variational Bayes.” In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014 (cit. on pp. 19, 39, 40, 66).
- [KW19] D. P. Kingma and M. Welling. “An Introduction to Variational Autoencoders.” In: *Found. Trends Mach. Learn.* 12.4 (2019), pp. 307–392 (cit. on p. 40).
- [KW70] G. S. Kimeldorf and G. Wahba. “A correspondence between Bayesian estimation on stochastic processes and smoothing by splines.” In: *The Annals of Mathematical Statistics* 41.2 (1970), pp. 495–502 (cit. on p. 22).
- [KZC21] D. Kiyasseh, T. Zhu, and D. A. Clifton. “Clocs: Contrastive learning of cardiac signals across space, time, and patients.” In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5606–5615 (cit. on p. 43).
- [Lan51] L. Landweber. “An iteration formula for Fredholm integral equations of the first kind.” In: *American journal of mathematics* 73.3 (1951), pp. 615–624 (cit. on p. 57).
- [LBH15] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning.” In: *nature* 521.7553 (2015), pp. 436–444 (cit. on p. 4).
- [LC10] Y. LeCun and C. Cortes. “MNIST handwritten digit database.” In: (2010) (cit. on p. 33).
- [LeC+06] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. “A tutorial on energy-based learning.” In: *Predicting structured data* 1.0 (2006) (cit. on p. 25).
- [LeC+07] Y. LeCun, S. Chopra, M. Ranzato, and F.-J. Huang. “Energy-based models in document recognition and computer vision.” In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 1. IEEE. 2007, pp. 337–341 (cit. on p. 25).
- [Lee+19] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. “Wide neural networks of any depth evolve as linear models under gradient descent.” In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 50).
- [ŁH05] J. M. Łęski and N. Henzel. “ECG baseline wander and powerline interference reduction using nonlinear filter bank.” In: *Signal processing* 85.4 (2005), pp. 781–793 (cit. on p. 71).

- [Li+18] C. Li, H. Farkhoor, R. Liu, and J. Yosinski. “Measuring the Intrinsic Dimension of Objective Landscapes.” In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, Canada, April 30 - May 3, 2019*. OpenReview.net, 2018 (cit. on p. 33).
- [Li+19] L. Li, J. Yan, X. Yang, and Y. Jin. “Learning interpretable deep state space model for probabilistic time series forecasting.” In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence. IJCAI’19*. Macao, China: AAAI Press, 2019, pp. 2901–2908 (cit. on p. 65).
- [Lim+21] E. M. Lima, A. H. Ribeiro, G. M. Paixão, M. H. Ribeiro, M. M. Pinto-Filho, P. R. Gomes, D. M. Oliveira, E. C. Sabino, B. B. Duncan, L. Giatti, et al. “Deep neural network-estimated electrocardiographic age as a mortality predictor.” In: *Nature communications* 12.1 (2021), p. 5117 (cit. on p. 72).
- [Lin+22] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön. *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press, 2022 (cit. on p. 17).
- [Liu+20] J. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax Weiss, and B. Lakshminarayanan. “Simple and principled uncertainty estimation with deterministic deep learning via distance awareness.” In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7498–7512 (cit. on p. 29).
- [Liu+21] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang. “Self-supervised learning: Generative or contrastive.” In: *IEEE transactions on knowledge and data engineering* 35.1 (2021), pp. 857–876 (cit. on p. 42).
- [Lju+20] L. Ljung, C. Andersson, K. Tiels, and T. B. Schön. “Deep learning and system identification.” In: *IFAC-PapersOnLine* 53.2 (2020), pp. 1175–1181 (cit. on p. 62).
- [Lju98] L. Ljung. “System identification.” In: *Signal analysis and prediction*. Springer, 1998, pp. 163–173 (cit. on pp. 60, 62).
- [LL17] S. M. Lundberg and S.-I. Lee. “A unified approach to interpreting model predictions.” In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 73).
- [Loc+19] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem. “Challenging common assumptions in the unsupervised learning of disentangled representations.” In: *international conference on machine learning*. PMLR, 2019, pp. 4114–4124 (cit. on pp. 2, 40, 41).

- [Loc+20a] F. Locatello, B. Poole, G. Rätsch, B. Schölkopf, O. Bachem, and M. Tschannen. “Weakly-supervised disentanglement without compromises.” In: *Proceedings of the 37th International Conference on Machine Learning*. ICML’20. JMLR.org, 2020 (cit. on p. 41).
- [Loc+20b] F. Locatello, M. Tschannen, S. Bauer, G. Rätsch, B. Schölkopf, and O. Bachem. “Disentangling Factors of Variations Using Few Labels.” In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020 (cit. on p. 41).
- [Loc20] F. Locatello. “Enforcing and Discovering Structure in Machine Learning.” PhD thesis. 2020 (cit. on p. 40).
- [LPB17] B. Lakshminarayanan, A. Pritzel, and C. Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles.” In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 24, 25).
- [LPK19] H. M. Lynn, S. B. Pan, and P. Kim. “A deep bidirectional GRU network model for biometric electrocardiogram classification based on recurrent neural networks.” In: *IEEE Access* 7 (2019), pp. 145395–145405 (cit. on p. 72).
- [LPK20] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis. “Explainable ai: A review of machine learning interpretability methods.” In: *Entropy* 23.1 (2020), p. 18 (cit. on p. 73).
- [MAA05] S. Mahmoodabadi, A. Ahmadian, and M. Abolhasani. “ECG feature extraction using Daubechies wavelets.” In: *Proceedings of the fifth IASTED International conference on Visualization, Imaging and Image Processing*. 2005, pp. 343–348 (cit. on p. 71).
- [Mac92] D. J. MacKay. “Bayesian interpolation.” In: *Neural computation* 4.3 (1992), pp. 415–447 (cit. on p. 24).
- [Mac98] D. J. MacKay. “Introduction to Gaussian processes.” In: *NATO ASI series F computer and systems sciences* 168 (1998), pp. 133–166 (cit. on pp. 28, 29).
- [Mad+19] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson. “A simple baseline for bayesian uncertainty in deep learning.” In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 25).
- [Mal+22] N. Mallinar, J. Simon, A. Abedsoltan, P. Pandit, M. Belkin, and P. Nakkiran. “Benign, tempered, or catastrophic: Toward a refined taxonomy of overfitting.” In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 1182–1195 (cit. on p. 57).

- [Man82] B. B. Mandelbrot. *The fractal geometry of nature*. New York, NY: W.H. Freeman, 1982 (cit. on p. 3).
- [Mar10] J. Martens. “Deep learning via Hessian-free optimization.” In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML’10. Haifa, Israel: Omnipress, 2010, pp. 735–742 (cit. on p. 19).
- [Max65] J. C. Maxwell. “A dynamical theory of the electromagnetic field.” In: *Philosophical Transactions of the Royal Society of London* 155 (1865), pp. 459–513 (cit. on p. 3).
- [MB17] S. Ma and M. Belkin. “Diving into the shallows: a computational perspective on large-scale shallow learning.” In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 27).
- [MB19] S. Ma and M. Belkin. “Kernel machines that adapt to GPUs for effective large batch training.” In: *Proceedings of Machine Learning and Systems* 1 (2019), pp. 360–373 (cit. on p. 27).
- [MB21] D. Masti and A. Bemporad. “Learning nonlinear state–space models using autoencoders.” In: *Automatica* 129 (2021), p. 109666 (cit. on p. 65).
- [Mea+20] G. Meanti, L. Carratino, L. Rosasco, and A. Rudi. “Kernel methods through the roof: handling billions of points efficiently.” In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 14410–14422 (cit. on p. 27).
- [Med+16] L. Medford-Davis, E. Park, G. Shlamovitz, J. Suliburk, A. N. Meyer, and H. Singh. “Diagnostic errors related to acute abdominal pain in the emergency department.” In: *Emergency Medicine Journal* 33.4 (2016), pp. 253–259 (cit. on p. 70).
- [Men+22] L. Meng, W. Tan, J. Ma, R. Wang, X. Yin, and Y. Zhang. “Enhancing dynamic ECG heartbeat classification with lightweight transformer model.” In: *Artificial Intelligence in medicine* 124 (2022), p. 102236 (cit. on p. 72).
- [Mik+98] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. “Kernel PCA and de-noising in feature spaces.” In: *Advances in neural information processing systems* 11 (1998) (cit. on p. 37).
- [MK21] P. W. Macfarlane and J. Kennedy. “Automated ecg interpretation—a brief history from high expectations to deepest networks.” In: *Hearts* 2.4 (2021), pp. 433–448 (cit. on p. 71).

- [ML88] P. Macfarlane and T. Lawrie. *Comprehensive Electrocardiology: Theory and Practice in Health and Disease*. Comprehensive Electrocardiology: Theory and Practice in Health and Disease v. 3. Pergamon Press, 1988 (cit. on p. 69).
- [MM01] G. Moody and R. Mark. “The impact of the MIT-BIH Arrhythmia Database.” In: *IEEE Engineering in Medicine and Biology Magazine* 20.3 (2001), pp. 45–50 (cit. on p. 71).
- [MM20] I. Misra and L. v. d. Maaten. “Self-supervised learning of pre-text-invariant representations.” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 6707–6717 (cit. on p. 42).
- [MM22] S. Mei and A. Montanari. “The generalization error of random features regression: Precise asymptotics and the double descent curve.” In: *Communications on Pure and Applied Mathematics* 75.4 (2022), pp. 667–766 (cit. on p. 49).
- [MM23] T. Misiakiewicz and A. Montanari. “Six lectures on linearized neural networks.” In: *arXiv preprint arXiv:2308.13431* (2023) (cit. on p. 49).
- [Mol20] C. Molnar. *Interpretable machine learning*. Lulu. com, 2020 (cit. on p. 73).
- [Moo03] G. E. Moore. *Principia Ethica*. Mineola, N.Y.: Dover Publications, 1903 (cit. on p. 3).
- [Mor77] D. W. Mortara. “Digital filters for ECG signals.” In: *Computers in Cardiology* (1977), pp. 511–514 (cit. on p. 71).
- [MP67] V. A. Marčenko and L. A. Pastur. “The distribution of eigenvalues in certain sets of random matrices.” In: *Mathematics of the USSR-Sbornik* 1.4 (1967), pp. 457–483 (cit. on pp. 53, 55).
- [Mur22] K. P. Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022 (cit. on p. 17).
- [Mut+20] V. Muthukumar, K. Vodrahalli, V. Subramanian, and A. Sahai. “Harmless interpolation of noisy data in regression.” In: *IEEE Journal on Selected Areas in Information Theory* 1.1 (2020), pp. 67–83 (cit. on p. 49).
- [MZ07] A. I. Manriquez and Q. Zhang. “An algorithm for QRS onset and offset detection in single lead electrocardiogram records.” In: *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. 2007, pp. 541–544 (cit. on p. 71).

- [Nak+21] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. “Deep double descent: Where bigger models and more data hurt.” In: *Journal of Statistical Mechanics: Theory and Experiment* 2021.12 (2021), p. 124003 (cit. on pp. 48, 49).
- [Nea96] R. M. Neal. *Bayesian Learning for Neural Networks*. 1996 (cit. on p. 24).
- [New87] I. Newton. *Philosophiae Naturalis Principia Mathematica. Auctore Js. Newton ... Jussu Societatis Regiae ac Typis Josephi Streater ...*, 1687 (cit. on p. 3).
- [NF16] M. Noroozi and P. Favaro. “Unsupervised learning of visual representations by solving jigsaw puzzles.” In: *European conference on computer vision*. Springer. 2016, pp. 69–84 (cit. on p. 42).
- [NTS15] B. Neyshabur, R. Tomioka, and N. Srebro. “In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning.” In: *ICLR (Workshop)*. 2015 (cit. on p. 49).
- [NYC16] A. M. Nguyen, J. Yosinski, and J. Clune. “Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks.” In: *CoRR* abs/1602.03616 (2016) (cit. on p. 34).
- [OE16] Z. Obermeyer and E. J. Emanuel. “Predicting the Future — Big Data, Machine Learning, and Clinical Medicine.” In: *New England Journal of Medicine* 375.13 (2016), pp. 1216–1219 (cit. on p. 69).
- [OF96] B. A. Olshausen and D. J. Field. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images.” In: *Nature* 381.6583 (1996), pp. 607–609 (cit. on p. 3).
- [OGY90] E. Ott, C. Grebogi, and J. A. Yorke. “Controlling chaos.” In: *Phys. Rev. Lett.* 64 (11 1990), pp. 1196–1199 (cit. on p. 3).
- [OLV18] A. van den Oord, Y. Li, and O. Vinyals. “Representation Learning with Contrastive Predictive Coding.” In: *CoRR* abs/1807.03748 (2018) (cit. on p. 43).
- [Omi+23] J. A. Omiye, J. C. Lester, S. Spichak, V. Rotemberg, and R. Daneshjou. “Large language models propagate race-based medicine.” In: *npj Digital Medicine* 6.1 (2023) (cit. on p. 69).
- [OMS17] C. Olah, A. Mordvintsev, and L. Schubert. “Feature Visualization.” In: *Distill* (2017). <https://distill.pub/2017/feature-visualization> (cit. on p. 34).
- [Oor+16a] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. “WaveNet: A Generative Model for Raw Audio.” In: *Arxiv*. 2016 (cit. on p. 45).

- [Oor+16b] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. “Conditional image generation with PixelCNN decoders.” In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS’16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 4797–4805 (cit. on p. 19).
- [Opp95] M. Opper. “Statistical mechanics of learning: Generalization.” In: *The handbook of brain theory and neural networks* (1995), pp. 922–925 (cit. on p. 49).
- [Opp99] A. V. Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999 (cit. on p. 60).
- [Ova+19] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek. “Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift.” In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 25).
- [Pas+19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. “Pytorch: An imperative style, high-performance deep learning library.” In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 27).
- [Pat+16] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. “Context encoders: Feature learning by inpainting.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544 (cit. on pp. 42, 43).
- [Pau+21] A. Paullada, I. D. Raji, E. M. Bender, E. Denton, and A. Hanna. “Data and its (dis) contents: A survey of dataset development and use in machine learning research.” In: *Patterns* 2.11 (2021) (cit. on p. 69).
- [Pau07] D. Paul. “Asymptotics of sample eigenstructure for a large dimensional spiked covariance model.” In: *Statistica Sinica* (2007), pp. 1617–1642 (cit. on p. 56).
- [PD10] G. Pillonetto and G. De Nicolao. “A new kernel-based approach for linear system identification.” In: *Automatica* 46.1 (2010), pp. 81–93 (cit. on p. 60).
- [PDN22] J. Petch, S. Di, and W. Nelson. “Opening the black box: the promise and limitations of explainable machine learning in cardiology.” In: *Canadian Journal of Cardiology* 38.2 (2022), pp. 204–213 (cit. on p. 73).

- [Pea01] K. Pearson. “LIII. On lines and planes of closest fit to systems of points in space.” In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572 (cit. on p. 35).
- [Pil+14] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung. “Kernel methods in system identification, machine learning and function estimation: A survey.” In: *Automatica* 50.3 (2014), pp. 657–682 (cit. on p. 60).
- [Pil+23] G. Pillonetto, A. Aravkin, D. Gedon, L. Ljung, A. H. Ribeiro, and T. B. Schön. “Deep networks for system identification: a Survey.” In: *arXiv preprint arXiv:2301.12832* (2023) (cit. on pp. 18, 28, 60, 62, 63).
- [Pla98] Plato. *Timaeus*. Trans. by B. Jowett. Project Gutenberg, 1998 (cit. on p. 2).
- [Pop+21] P. Pope, C. Zhu, A. Abdelkader, M. Goldblum, and T. Goldstein. “The Intrinsic Dimension of Images and Its Impact on Learning.” In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021 (cit. on p. 33).
- [Pop59] K. R. Popper. *The Logic of Scientific Discovery*. London: Routledge, 1959 (cit. on p. 15).
- [Pre02] L. Prechelt. “Early stopping-but when?” In: *Neural Networks: Tricks of the trade*. Springer, 2002, pp. 55–69 (cit. on p. 57).
- [PW17] J. Pennington and P. Worah. “Nonlinear random matrix theory for deep learning.” In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 51).
- [QR05] J. Quinonero-Candela and C. E. Rasmussen. “A unifying view of sparse approximate Gaussian process regression.” In: *The Journal of Machine Learning Research* 6 (2005), pp. 1939–1959 (cit. on p. 27).
- [Rad+18] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. “Improving language understanding by generative pre-training.” In: (2018) (cit. on pp. 19, 44).
- [Rad+19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. “Language models are unsupervised multitask learners.” In: *OpenAI blog* 1.8 (2019), p. 9 (cit. on p. 44).
- [Rad+24] A. Radhakrishnan, D. Beaglehole, P. Pandit, and M. Belkin. “Mechanism for feature learning in neural networks and backpropagation-free machine learning models.” In: *Science* 383.6690 (2024), pp. 1461–1467 (cit. on pp. 29, 30).

- [Ran+18] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. “Deep state space models for time series forecasting.” In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 65).
- [RB15] R. Reris and J. P. Brooks. “Principal component analysis and optimization: A tutorial.” In: (2015) (cit. on p. 36).
- [RBB18] H. Ritter, A. Botev, and D. Barber. “A scalable laplace approximation for neural networks.” In: *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*. Vol. 6. International Conference on Representation Learning. 2018 (cit. on p. 25).
- [RBD24] A. Radhakrishnan, M. Belkin, and D. Drusvyatskiy. “Linear Recursive Feature Machines provably recover low-rank matrices.” In: *arXiv preprint arXiv:2401.04553* (2024) (cit. on p. 30).
- [RCR17] A. Rudi, L. Carratino, and L. Rosasco. “Falkon: An optimal large scale kernel method.” In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 27).
- [RFB15] O. Ronneberger, P. Fischer, and T. Brox. “U-net: Convolutional networks for biomedical image segmentation.” In: *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III 18*. Springer. 2015, pp. 234–241 (cit. on p. 18).
- [RHW+85] D. E. Rumelhart, G. E. Hinton, R. J. Williams, et al. *Learning internal representations by error propagation*. 1985 (cit. on p. 4).
- [RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors.” In: *Nature* 323.6088 (1986), pp. 533–536 (cit. on p. 19).
- [Rib+20a] A. H. Ribeiro, D. Gedon, D. M. Teixeira, M. H. Ribeiro, A. L. P. Ribeiro, T. B. Schön, and W. Meira. “Automatic 12-lead ECG classification using a convolutional network ensemble.” In: *2020 Computing in Cardiology*. IEEE. 2020, pp. 1–4 (cit. on p. 72).
- [Rib+20b] A. H. Ribeiro, M. H. Ribeiro, G. M. Paixão, D. M. Oliveira, P. R. Gomes, J. A. Canazart, M. P. Ferreira, C. R. Andersson, P. W. Macfarlane, W. Meira Jr, et al. “Automatic diagnosis of the 12-lead ECG using a deep neural network.” In: *Nature communications* 11.1 (2020), p. 1760 (cit. on pp. 71, 72).
- [Rib+21] A. H. Ribeiro, J. N. Hendriks, A. G. Wills, and T. B. Schön. “Beyond occam’s razor in system identification: Double-descent when modeling dynamics.” In: *IFAC-PapersOnLine* 54.7 (2021), pp. 97–102 (cit. on p. 64).

- [Rib+24] A. Ribeiro, D. Zachariah, F. Bach, and T. Schön. “Regularization properties of adversarially-trained linear regression.” In: *Advances in Neural Information Processing Systems 36* (2024) (cit. on p. 57).
- [RM15] D. Rezende and S. Mohamed. “Variational Inference with Normalizing Flows.” In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 1530–1538 (cit. on p. 19).
- [RM18] K. Ridgeway and M. C. Mozer. “Learning deep disentangled embeddings with the f-statistic loss.” In: *Advances in neural information processing systems 31* (2018) (cit. on p. 40).
- [RM51] H. Robbins and S. Monro. “A Stochastic Approximation Method.” In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407 (cit. on p. 19).
- [RMW14] D. J. Rezende, S. Mohamed, and D. Wierstra. “Stochastic Backpropagation and Approximate Inference in Deep Generative Models.” In: *Proceedings of the 31st International Conference on Machine Learning*. Vol. 32. Proceedings of Machine Learning Research 2. Beijing, China: PMLR, 2014, pp. 1278–1286 (cit. on pp. 19, 39, 66).
- [RN10] S. J. Russell and P. Norvig. *Artificial intelligence a modern approach*. London, 2010 (cit. on p. 3).
- [Ros58] F. Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386 (cit. on p. 4).
- [Rot+18] G. A. Roth, D. Abate, K. H. Abate, S. M. Abay, C. Abbafati, N. Abbasi, H. Abbastabar, F. Abd-Allah, J. Abdela, A. Abdelalim, et al. “Global, regional, and national age-sex-specific mortality for 282 causes of death in 195 countries and territories, 1980–2017: a systematic analysis for the Global Burden of Disease Study 2017.” In: *The lancet* 392.10159 (2018), pp. 1736–1788 (cit. on p. 70).
- [RPK19] M. Raissi, P. Perdikaris, and G. E. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.” In: *Journal of Computational physics* 378 (2019), pp. 686–707 (cit. on p. 77).
- [RR08] A. Rahimi and B. Recht. “Random Features for Large-Scale Kernel Machines.” In: *Advances in Neural Information Processing Systems 20*. 2008, pp. 1177–1184 (cit. on pp. 37, 48).

- [RS00] S. T. Roweis and L. K. Saul. “Nonlinear dimensionality reduction by locally linear embedding.” In: *science* 290.5500 (2000), pp. 2323–2326 (cit. on p. 38).
- [RSG16] M. T. Ribeiro, S. Singh, and C. Guestrin. “” Why should I trust you?” Explaining the predictions of any classifier.” In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144 (cit. on p. 73).
- [Rus04] B. Russell. *History of western philosophy*. Routledge, 2004 (cit. on p. 2).
- [RW06] C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning*. Vol. 1. Springer, 2006 (cit. on p. 26).
- [SC08] I. Steinwart and A. Christmann. *Support vector machines*. Springer Science & Business Media, 2008 (cit. on p. 20).
- [Sch+21] B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio. “Toward causal representation learning.” In: *Proceedings of the IEEE* 109.5 (2021), pp. 612–634 (cit. on p. 77).
- [Sel+17] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. “Grad-cam: Visual explanations from deep networks via gradient-based localization.” In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626 (cit. on p. 73).
- [Set09] B. Settles. “Active learning literature survey.” In: (2009) (cit. on p. 19).
- [SGK17] A. Shrikumar, P. Greenside, and A. Kundaje. “Learning important features through propagating activation differences.” In: *International conference on machine learning*. PMLR. 2017, pp. 3145–3153 (cit. on p. 73).
- [SGN05] T. Schön, F. Gustafsson, and P.-J. Nordlund. “Marginalized particle filters for mixed linear/nonlinear state-space models.” In: *IEEE Transactions on signal processing* 53.7 (2005), pp. 2279–2289 (cit. on p. 66).
- [Sil+17] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. “Mastering the game of go without human knowledge.” In: *nature* 550.7676 (2017), pp. 354–359 (cit. on p. 1).
- [SK19] C. Shorten and T. M. Khoshgoftaar. “A survey on image data augmentation for deep learning.” In: *Journal of big data* 6.1 (2019), pp. 1–48 (cit. on p. 19).

- [SL19] J. Schoukens and L. Ljung. “Nonlinear system identification: A user-oriented road map.” In: *IEEE Control Systems Magazine* 39.6 (2019), pp. 28–99 (cit. on p. 62).
- [Sob15] E. Sober. *Ockham’s razors*. Cambridge University Press, 2015 (cit. on p. 15).
- [Sob81] E. Sober. “The principle of parsimony.” In: *The British Journal for the Philosophy of Science* 32.2 (1981), pp. 145–156 (cit. on pp. 4, 15).
- [Soh+15] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. “Deep unsupervised learning using nonequilibrium thermodynamics.” In: *International conference on machine learning*. PMLR. 2015, pp. 2256–2265 (cit. on p. 19).
- [Sri+14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting.” In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958 (cit. on p. 19).
- [SS02] B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002 (cit. on p. 21).
- [SS04] P. Stoica and Y. Selen. “Model-order selection: a review of information criterion rules.” In: *IEEE Signal Processing Magazine* 21.4 (2004), pp. 36–47 (cit. on p. 48).
- [SS88] T. Söderstrom and P. Stoica. *System Identification*. Prentice Hall International Series in Systems and Control Engineering. London, England: Prentice-Hall, 1988 (cit. on p. 62).
- [SSM97] B. Schölkopf, A. Smola, and K.-R. Müller. “Kernel principal component analysis.” In: *International conference on artificial neural networks*. Springer. 1997, pp. 583–588 (cit. on p. 37).
- [Str18] S. H. Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018 (cit. on p. 59).
- [STY17] M. Sundararajan, A. Taly, and Q. Yan. “Axiomatic attribution for deep networks.” In: *International conference on machine learning*. PMLR. 2017, pp. 3319–3328 (cit. on p. 73).
- [SVZ14a] K. Simonyan, A. Vedaldi, and A. Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.” In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*. 2014 (cit. on p. 34).

- [SVZ14b] K. Simonyan, A. Vedaldi, and A. Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.” In: *Workshop at International Conference on Learning Representations*. 2014 (cit. on p. 73).
- [SW17] J. Schläpfer and H. J. Wellens. “Computer-interpreted electrocardiograms: benefits and limitations.” In: *Journal of the American College of Cardiology* 70.9 (2017), pp. 1183–1192 (cit. on p. 72).
- [SWL23] J. T. Smith, A. Warrington, and S. Linderman. “Simplified State Space Layers for Sequence Modeling.” In: *The Eleventh International Conference on Learning Representations*. 2023 (cit. on p. 64).
- [SWN11] T. B. Schön, A. Wills, and B. Ninness. “System identification of nonlinear state-space models.” In: *Automatica* 47.1 (2011), pp. 39–49 (cit. on pp. 61, 62).
- [SZ15] K. Simonyan and A. Zisserman. “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*. 2015 (cit. on p. 19).
- [Sze+14] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. “Intriguing properties of neural networks.” In: *International Conference on Learning Representations*. 2014 (cit. on p. 1).
- [Tao23] T. Tao. *Topics in random matrix theory*. Vol. 132. American Mathematical Society, 2023 (cit. on pp. 49, 51).
- [TB99] M. E. Tipping and C. M. Bishop. “Probabilistic principal component analysis.” In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 61.3 (1999), pp. 611–622 (cit. on p. 38).
- [TH12] T. Tieleman and G. Hinton. “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.” In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31 (cit. on p. 19).
- [TL19] M. Tan and Q. Le. “Efficientnet: Rethinking model scaling for convolutional neural networks.” In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114 (cit. on p. 47).
- [TSL00] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. “A global geometric framework for nonlinear dimensionality reduction.” In: *science* 290.5500 (2000), pp. 2319–2323 (cit. on p. 37).

- [Vai+19] J. Vaicenavicius, D. Widmann, C. Andersson, F. Lindsten, J. Roll, and T. Schön. “Evaluating model calibration in classification.” In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Vol. 89. Proceedings of Machine Learning Research. PMLR, 2019, pp. 3459–3467 (cit. on p. 23).
- [Val84] L. G. Valiant. “A theory of the learnable.” In: *Communications of the ACM* 27.11 (1984), pp. 1134–1142 (cit. on p. 51).
- [Van+16] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. “Conditional image generation with pixelenn decoders.” In: *Advances in neural information processing systems* 29 (2016) (cit. on p. 44).
- [Vap99] V. N. Vapnik. “An overview of statistical learning theory.” In: *IEEE transactions on neural networks* 10.5 (1999), pp. 988–999 (cit. on p. 51).
- [Vas+17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. “Attention is all you need.” In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 1, 18, 19, 64).
- [VC71] V. N. Vapnik and A. Y. Chervonenkis. “On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities.” In: *Theory of Probability & Its Applications* 16.2 (1971), pp. 264–280 (cit. on pp. 48, 51).
- [Vel+18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. “Graph Attention Networks.” In: *International Conference on Learning Representations*. 2018 (cit. on p. 64).
- [Vel23] P. Veličković. “Everything is connected: Graph neural networks.” In: *Current Opinion in Structural Biology* 79 (2023), p. 102538 (cit. on p. 64).
- [Ver18] R. Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics 47. Cambridge: Cambridge University Press, 2018 (cit. on p. 49).
- [VH08] L. Van der Maaten and G. Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008) (cit. on p. 38).
- [VKK16] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu. “Pixel recurrent neural networks.” In: *International conference on machine learning*. PMLR. 2016, pp. 1747–1756 (cit. on p. 44).

- [Von+24] P. Von Bachmann, D. Gedon, F. K. Gustafsson, A. H. Ribeiro, E. Lampa, S. Gustafsson, J. Sundström, and T. B. Schön. “ECG-Based Electrolyte Prediction: Evaluating Regression and Probabilistic Methods.” Submitted. 2024 (cit. on p. 72).
- [VV07] M. Verhaegen and V. Verdult. *Filtering and System Identification: A Least Squares Approach*. Cambridge university press, 2007 (cit. on p. 62).
- [Wag+20] P. Wagner, N. Strodtboff, R.-D. Bousseljot, W. Samek, and T. Schaeffter. *PTB-XL, a large publicly available electrocardiography dataset*. 2020 (cit. on pp. 33, 71).
- [Wai19] M. J. Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*. Vol. 48. Cambridge university press, 2019 (cit. on p. 49).
- [Wal87] A. D. Waller. “A Demonstration on Man of Electromotive Changes accompanying the Heart’s Beat.” In: *The Journal of Physiology* 8.5 (1887), pp. 229–234 (cit. on p. 69).
- [Wid23] D. Widmann. “Reliable Uncertainty Quantification in Statistical Learning.” PhD thesis. Acta Universitatis Upsaliensis, 2023 (cit. on p. 23).
- [Wik22] Wikimedia Commons. *Schematic diagram of normal sinus rhythm for a human heart as seen on ECG (with English labels)*. Created by Agateller (Anthony Atkielski), converted to svg by atom. Accessed: 20-03-2024. 2022. URL: <https://en.wikipedia.org/wiki/Electrocardiography#/media/File:SinusRhythmLabels.svg> (cit. on p. 71).
- [Wil+16] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. “Deep kernel learning.” In: *Artificial intelligence and statistics*. PMLR. 2016, pp. 370–378 (cit. on p. 29).
- [Wit53] L. Wittgenstein. *Philosophical Investigations*. Oxford: Basil Blackwell, 1953 (cit. on p. 4).
- [WJ+08] M. J. Wainwright, M. I. Jordan, et al. “Graphical models, exponential families, and variational inference.” In: *Foundations and Trends® in Machine Learning* 1.1–2 (2008), pp. 1–305 (cit. on p. 39).
- [WR27] A. N. Whitehead and B. A. W. Russell. *Principia mathematica; 2nd ed.* Cambridge: Cambridge Univ. Press, 1927 (cit. on p. 3).
- [Wri+19] B. Wright, N. Faulkner, P. Bragge, and M. Gruber. “What interventions could reduce diagnostic error in emergency departments? A review of evidence, practice and consumer perspectives.” In: *Diagnosis* 6.4 (2019), pp. 325–334 (cit. on p. 70).

- [WT11] M. Welling and Y. W. Teh. “Bayesian learning via stochastic gradient Langevin dynamics.” In: *Proceedings of the 28th international conference on machine learning*. 2011, pp. 681–688 (cit. on p. 25).
- [Wu+20] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. “A comprehensive survey on graph neural networks.” In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24 (cit. on p. 64).
- [WX20] D. Wu and J. Xu. “On the Optimal Weighted  $\ell_2$  Regularization in Overparameterized Linear Regression.” In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 10112–10123 (cit. on p. 57).
- [XH19] J. Xu and D. J. Hsu. “On the number of variables to use in principal component regression.” In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 57).
- [XL20] X. Xu and H. Liu. “ECG heartbeat classification using convolutional neural networks.” In: *IEEE access* 8 (2020), pp. 8614–8619 (cit. on p. 72).
- [YJ18] J. Yang and I. M. Johnstone. “Edgeworth correction for the largest eigenvalue in a spiked pca model.” In: *Statistica Sinica* 28.4 (2018), p. 2541 (cit. on p. 56).
- [Yos+14] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. “How transferable are features in deep neural networks?” In: *Advances in neural information processing systems* 27 (2014) (cit. on p. 19).
- [Yos+15] J. Yosinski, J. Clune, A. M. Nguyen, T. J. Fuchs, and H. Lipson. “Understanding Neural Networks Through Deep Visualization.” In: *CoRR* abs/1506.06579 (2015) (cit. on p. 34).
- [You+20] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen. “Graph contrastive learning with augmentations.” In: *Advances in neural information processing systems* 33 (2020), pp. 5812–5823 (cit. on p. 43).
- [Zad56] L. Zadeh. “On the identification problem.” In: *IRE Transactions on Circuit Theory* 3.4 (1956), pp. 277–281 (cit. on p. 60).
- [Zha+17] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. “Understanding deep learning requires rethinking generalization.” In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017 (cit. on p. 47).

- [Zha+20] J. Zhang, A. Liu, M. Gao, X. Chen, X. Zhang, and X. Chen. “ECG-based multi-class arrhythmia detection using spatio-temporal attention-based convolutional recurrent neural network.” In: *Artificial Intelligence in Medicine* 106 (2020), p. 101856 (cit. on p. 72).
- [Zha+23] H. Zhang, W. Liu, J. Shi, S. Chang, H. Wang, J. He, and Q. Huang. “MaeFE: Masked Autoencoders Family of Electrocardiogram for Self-Supervised Pretraining and Transfer Learning.” In: *IEEE Transactions on Instrumentation and Measurement* 72 (2023), pp. 1–15 (cit. on p. 45).
- [ZIE16] R. Zhang, P. Isola, and A. A. Efros. “Colorful image colorization.” In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part III* 14. Springer. 2016, pp. 649–666 (cit. on pp. 42, 43).
- [ZK16] S. Zagoruyko and N. Komodakis. “Wide Residual Networks.” In: *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19–22, 2016*. BMVA Press, 2016 (cit. on p. 19).



# Acta Universitatis Upsaliensis

*Digital Comprehensive Summaries of Uppsala Dissertations from  
the Faculty of Science and Technology 2392*

Editor: The Dean of the Faculty of Science and Technology

A doctoral dissertation from the Faculty of Science and Technology, Uppsala University, is usually a summary of a number of papers. A few copies of the complete dissertation are kept at major Swedish research libraries, while the summary alone is distributed internationally through the series Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology. (Prior to January, 2005, the series was published under the title "Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology".)



# Paper I

## Title

Uncertainty Estimation with Recursive Feature Machines

## Authors

Daniel Gedon, Amirhesam Abedsoltan, Thomas B. Schön,  
Mikhail Belkin

## Edited version of

D. Gedon, A. Abedsoltan, T. B. Schön, and M. Belkin. “Uncertainty Estimation with Recursive Feature Machines.” Submitted. 2024

An early version of this paper appeared as:

D. Gedon, et al. ”On Feature Learning of Recursive Feature Machines and Automatic Relevance Determination.” In: *UniReps: the First Workshop on Unifying Representations in Neural Models Workshop* at NeurIPS. 2023



# Uncertainty Estimation with Recursive Feature Machines

## Abstract

In conventional regression analysis, predictions are typically represented as point estimates derived from covariates. The Gaussian Process (GP) offer a kernel-based framework that predicts and additionally quantifies associated uncertainties. However, kernel-based methods often underperform ensemble-based decision tree approaches in regression tasks involving tabular and categorical data. Recently, Recursive Feature Machines (RFMs) were proposed as a novel feature-learning kernel which strengthens the capabilities of kernel machines. In this study, we harness the power RFMs in a probabilistic GP-based approach to enhance uncertainty estimation through feature extraction within kernel methods. We employ this learned kernel for in-depth uncertainty analysis. On tabular datasets, our RFM-based method surpasses other leading uncertainty estimation techniques, including NGBoost and CatBoost-ensemble. Additionally, when assessing out-of-distribution performance, we found that boosting-based methods are surpassed by our RFM-based approach.

## 1 Introduction

Regression analysis traditionally predicts future outcomes by providing definitive values based on empirical data. However, as the applications of predictive modelling expand into critical areas like healthcare [Nic+22; Tra+21; Ava+18] and weather forecasting [GK14], there is an increasing need to understand the confidence or uncertainty surrounding these predictions, beyond just point estimates. As stated in Kompa et al. [KS21] “*medical ML should have the ability to say “I don’t know” and potentially abstain from providing a diagnosis or prediction when there is a large amount of uncertainty for a given patient*”. A rising number of publications underscore the importance of uncertainty quantification, evident in fields like radiology [Chu+23], digital pathology [Lin+23], cancer digital histopathology [Dol+22], and radiation oncology [Bar+22], to name a few.

The Recursive Feature Machine (RFM) [Rad+24] represents an innovative data-adaptive kernel-based method, which provides a unique lens for data inter-

pretation. Our research explores the capabilities of RFMs, focusing on their aptitude for uncertainty estimation in both in-distribution and out-of-distribution contexts. We pit our probabilistic RFMs against other prominent techniques, especially state-of-the-art probabilistic decision tree-based methods like NGBoost [Dua+20] and CatBoost-ensembles [Pro+18], underscoring their competitive edge.

The Gaussian process (GP) is often the method of choice for gauging uncertainty in predictions [RW06], offering a sophisticated perspective that transcends mere point estimates. However, with the ongoing evolution in machine learning, decision tree-based techniques such as NGBoost and CatBoost-ensembles are gaining traction. These methods not only challenge the GP in terms of prediction accuracy but have also showcased superior results in specific uncertainty metrics like Negative Log Likelihood (NLL), coverage error (CE) and prediction interval length (IL), especially for tabular or categorical data.

In our study, we demonstrate that by integrating GPs with the data-adaptive kernel derived from the RFM, we can bridge this performance gap, achieving results that are on par with or even surpass gradient-based boosting approaches. In summary, (i) we introduced the RFM to the GP community and (ii) established that the performance of RFM is comparable to, or even superior to, existing state-of-the-art methods. More specifically, we have the following contributions:

- Our findings reveal that GP-RFM stands as a strong alternative to leading boosting-based techniques, particularly by enhancing uncertainty estimation in tabular datasets via features generated from the RFM. This capability to match or in certain instances exceed the performance of existing top-tier methods establishes the RFM as a new benchmark for applications that demand accurate uncertainty assessments.
- We bring RFMs to the GP community and illustrate that features derived from the RFM notably improve uncertainty performance on tabular datasets. Comparing the RFM with traditional GP techniques, we further show that the RFM can extract more general feature representations due to its ability to capture correlation between features. This can in turn significantly improve the resulting uncertainty estimates.
- To highlight the robustness of the RFM we compare it on out-of-distribution data for label and covariate shift where the RFM surpasses other uncertainty quantification methods.

## 2 Prior work

Numerous uncertainty quantification methods have been proposed in the literature for utilization with tabular data. Here, we focus on discussing flexible methods with state-of-the-art predictive performance.

**Gaussian processes.** As a non-parametric, flexible Bayesian regression model, the Gaussian process is a well-studied and natural choice for uncertainty quantification [RW06]. The GP is characterized by a mean function and a kernel function as covariance. The crucial challenge is to choose the right kernel as it encodes high-level assumptions about the data. Commonly, the Radial Basis Function (RBF) or Laplace kernel is chosen, which has a limited number of parameters to optimize. For more flexibility, kernels with Automatic Relevance Determination (ARD) introduce covariate weighting through learnable parameters [Mac92; Nea96]. Vivarelli and Williams [VW98] generalizes the diagonal ARD weighting to general positive-definite weighting matrices or low-rank factorisations. Garnett et al. [GOH14] and Letham et al. [Let+20] use a factorized weighting matrix and approximate the posterior with Laplace approximation for active learning and Bayesian optimization. For the latter, sparse axis-aligned Subspace GPs leverage structural sparsity in the kernel [EJ21]. Instead of utilizing advanced kernels, we can equivalently transform the input and use standard kernels [Mac98]. Neural networks have been studied as feature extractors [Cal+16; Wil+16], or where the last layer approximates a GP [Hua+15; Liu+20]. Our approach combines both strategies, leveraging the recently proposed Recursive Feature Machine [Rad+24], which introduces a novel feature-extracting kernel with the probabilistic expressivity of GPs.

**Probabilistic boosting.** Boosting-based approaches [FS95; Fri01] allow for flexible models, which have found widespread application on tabular datasets [SA22; GOV22; McE+23]. Such methods include among others AdaBoost, XGBoost, LightGBM or CatBoost [CG16; Ke+17; Pro+18]. For classification problems, most methods have a natural probabilistic interpretation through estimated class probabilities. However, for regression problems, there is no such straightforward concept. Therefore, probabilistic extensions of boosting such as NGBoost, CatBoost-Ensembles [Dua+20; MPU21] or extensions to Random Forests [Sch+19; SH20] have been proposed. Notably, when comparing the performance of probabilistic boosting approaches against our GP-RFM, our approach performs on par or even outperforms them across a range of evaluation metrics and tabular regression datasets.

**Neural networks.** The ability to learn features from data is a key advantage of the predictive power of neural networks (NN). For uncertainty quantification, Bayesian NNs [Mac92; Nea96] are a natural choice. However, the need for approximate inference methods such as variational inference [Gra11; Blu+15]

or Markov Chain Monte Carlo [WT11] makes them computationally expensive. Conversely, the use of Monte Carlo dropout [GG16] provides less reliable uncertainty estimates [Ova+19; GDS20] than ensembles of NNs [LPB17]. Although deep ensembles set the gold standard for NNs, they necessitate training multiple NNs resulting in high computational and memory burden. We leverage the idea of feature learning in NN through the use of RFMs since the learnt features in the latter are intricately linked to features learnt in feedforward NNs [Rad+24].

### 3 Background

Most machine learning algorithms focus on estimating the predictive model  $f(\mathbf{x}) = \mathbb{E}[y|\mathbf{x}]$  from a training dataset  $\mathcal{D} = (\mathbf{X}, \mathbf{y}) = \{\mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}_{i=1}^n$ . However, in many applications, this is not sufficient. We are therefore interested in augmenting point estimates with reliable uncertainty quantification to obtain the predictive distribution  $p(f(\mathbf{x}_*)|\mathbf{x}_*, \mathcal{D})$  for a new test data point  $\mathbf{x}_*$ . In our approach, we leverage GPs in conjunction with feature learning kernels through RFMs.

#### 3.1 Kernel machines

Kernel machines [SS02] are non-parametric predictive models. Given training data  $\mathcal{D}$  a kernel machine is a model of the form

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i k(\mathbf{x}, \mathbf{x}_i). \quad (1)$$

Here,  $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$  is a positive semi-definite symmetric kernel function [Aro50]. According to the representer theorem [KW70], the unique solution to the infinite-dimensional optimization problem

$$\arg \min_{f \in \mathcal{H}} \sum_{i=1}^n (f(\mathbf{x}_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}}^2 \quad (2)$$

has the form given in (1). Here  $\mathcal{H}$  is the (unique) reproducing kernel Hilbert space corresponding to  $k$ . It can be seen that  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)$  in (1) is the unique solution to the linear system,

$$(k(\mathbf{X}, \mathbf{X}) + \lambda I_n) \boldsymbol{\alpha} = \mathbf{y}. \quad (3)$$

## 3.2 Gaussian processes

To extend kernel machines into a probabilistic setting, we can define a distribution over the predictive function which yields a GP  $f \sim \mathcal{GP}(m, k)$  specified by its mean function  $m$  and its covariance function  $k$ . Because of its properties, we utilize the kernel function  $k$  as the covariance function in the GP. The posterior predictive distribution of the GP is then given by

$$p(f(\mathbf{x}_*) | \mathbf{x}_*, \mathcal{D}) = \mathcal{N}(f(\mathbf{x}_*), \mathbb{V}[f(\mathbf{x}_*)]), \quad (4)$$

with the mean as in (1) and the covariance  $\mathbb{V}[f(\mathbf{x}_*)] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_*$ . We denote the kernel matrix as  $\mathbf{K}$  with  $K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$ ,  $\mathbf{k}_* = k(\mathbf{X}, \mathbf{x}_*)$  and the measurement noise variance as  $\sigma^2$ .

For the mean function, we choose the constant function  $m = 0$ . The choice of kernel encodes high-level assumptions about the resulting function. We consider an exponential kernel of the form

$$k(\mathbf{x}, \mathbf{z}) = \exp(g(\mathbf{x}, \mathbf{z})). \quad (5)$$

When we define  $g(\mathbf{x}, \mathbf{z}) = -\frac{1}{2\ell^2} \|\mathbf{x} - \mathbf{z}\|^2$ , we arrive at the widely adopted Radial Basis Function (RBF) kernel. Conversely,  $g(\mathbf{x}, \mathbf{z}) = -\frac{1}{\ell} \|\mathbf{x} - \mathbf{z}\|$  leads to the Laplace kernel.

The parameters  $\boldsymbol{\theta}$  of the kernel include the noise variance  $\sigma$  and the length scale  $\ell$ . These parameters are often found by solving an optimization problem dictated by Maximum Likelihood Estimation (MLE). Specifically, we can estimate these parameters in a Bayesian framework by minimizing the Negative Log Likelihood (NLL), defined as  $-\log p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta})$ .

## 3.3 Recursive Feature Machines

A fundamental limitation of kernel machines is their reliance on kernel functions that are not adaptive to data. As a result, for certain tasks, kernel machines can significantly underperform compared to neural networks. The recently introduced RFM constitute a type of kernel machine capable of learning features, making them data-adaptive.

To develop kernel machines that can learn features, RFM integrates a positive semi-definite, symmetric matrix,  $\mathbf{M}$ , as a learnable parameter into the kernel function. Specifically, this is suited for kernel functions that depend on the distance between points, such as  $k(\mathbf{x}, \mathbf{z}) = \phi(\|\mathbf{x} - \mathbf{z}\|^2)$  where  $\phi : \mathbb{R} \rightarrow \mathbb{R}$  and  $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$ . We incorporate the learnable matrix  $\mathbf{M}$  by using the Mahalanobis distance

$$\|\mathbf{x} - \mathbf{z}\|_{\mathbf{M}} := \sqrt{(\mathbf{x} - \mathbf{z})^T \mathbf{M} (\mathbf{x} - \mathbf{z})}. \quad (6)$$

Therefore, the matrix  $\mathbf{M}$  re-weights the individual covariates and can incorporate correlation between covariates, for which we call  $\mathbf{M}$  the *feature matrix*. While any kernel function can be used for  $\phi$ , we utilize the Laplace kernel based on the Mahalanobis distance

$$k_{\mathbf{M}}(\mathbf{x}, \mathbf{z}) := \exp\left(-\frac{1}{\gamma} \|\mathbf{x} - \mathbf{z}\|_{\mathbf{M}}\right). \quad (7)$$

The prediction function corresponding to this kernel is given by

$$f_{\mathbf{M}}(\mathbf{x}) = k_{\mathbf{M}}(\mathbf{x}, \mathbf{X})\boldsymbol{\alpha}, \quad (8)$$

with  $\boldsymbol{\alpha} = k_{\mathbf{M}}(\mathbf{X}, \mathbf{X})^{-1}\mathbf{y}$ . To learn the feature matrix  $\mathbf{M}$  we make use of the proposed idea of the Average Gradient Outer Product (AGOP) from Radhakrishnan et al. [Rad+24]: We start by initializing  $\mathbf{M}^{(0)} = \mathbf{I}_d$ . At each iteration step  $t$  we first solve for the kernel weights  $\boldsymbol{\alpha}$  from (8) with fixed  $\mathbf{M}$ . Second, we update  $\mathbf{M}$  using the AGOP defined as

$$\mathbf{M}^{(t+1)} = \frac{1}{n} \sum_{i=1}^n \nabla_{\mathbf{x}} f_{\mathbf{M}^{(t)}}(\mathbf{x}_i) \nabla_{\mathbf{x}} f_{\mathbf{M}^{(t)}}(\mathbf{x}_i)^T. \quad (9)$$

Essentially the AGOP and the resulting matrix  $\mathbf{M}$  is the covariance matrix of the function gradients. Intuitively, RFM prioritises the covariates that have the most impact on the prediction function. Thus, RFMs learn the presentation most relevant to the underlying task.

A special case of RFMs is when we restrict the feature matrix  $\mathbf{M}$  to be diagonal. This is equivalent to learning a separate length scale for each covariate. In contrast to the *RFM* without this restriction, we refer to the diagonally restricted model as *RFM-diag*.

**Remark.** Another way of covariate weighting specifically in GPs is through the extension with Automatic Relevance Determination (ARD) [Nea96]. The RBF kernel is extended by using  $g(\mathbf{x}, \mathbf{z}) = -\frac{1}{\ell^2} \|\mathbf{x} - \mathbf{z}\|_{\mathbf{M}}^2$  with  $\mathbf{M}^{-1} = \text{diag}([\ell_1^2, \dots, \ell_d^2])$ . While barely utilised in practice, a similar construction can be generated for the Laplace kernel with ARD. This effectively increases the parameter vector  $\boldsymbol{\theta}$  learnt through MLE optimization in the GP framework.

## 4 Method: GP-RFM

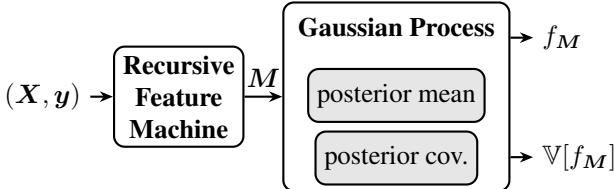
While GPs are powerful non-parametric models which offer uncertainty quantification, they are limited by their reliance on kernel functions that are not adaptive to data. RFMs are a type of kernel machine capable of learning features, making them data-adaptive. We propose to integrate RFMs into GPs by

replacing the kernel function  $k(\mathbf{x}, \mathbf{z})$  with the RFM-based kernel  $k_M(\mathbf{x}, \mathbf{z})$ . Since we are using the Laplace kernel within the RFM, we refer to the resulting construction as *GP-RFM-Laplace*.

Specifically, we consider a combination of a scale kernel with the RFM-based kernel to obtain  $\sigma_f^2 k_M(\mathbf{x}, \mathbf{z})$ . Since we are interested in the predictive distribution, we can set the mean function  $m$  of the GP to zero. The resulting predictive distribution is then given by (4) where  $f(\mathbf{x})$  is given by (8). The parameters of the GP are the feature matrix  $M$  and the kernel parameters  $\theta$  consisting of noise variance  $\sigma$ , length scale  $\ell$  as well as scale  $\sigma_f$ .

A visual representation of our algorithm is provided in Figure 1. It is important to highlight that the RFM algorithm, which identifies the matrix  $M$  through the AGOP iteration, see (9), shares the same time and memory complexity as the GP regression algorithm, see [Rad+24]. Consequently, incorporating this additional step does not complicate the overall complexity. For a comparison of the actual running times between our algorithm and other methods, please see Appendix C. In that section, we illustrate the time efficiency advantages of our method compared to boosting-based approaches like NGBoost and Cat-Boost ensembles.

**Training.** We disentangle the training of the GP-RFM-Laplace into two steps. First, we learn the feature matrix  $M$  using the recursive iteration between solving for the kernel weights  $\alpha$  for (8) and updating  $M$  using the AGOP defined in (9). To learn the kernel weights, we solve the linear system in (3) with a Ridge regularization term for stability to obtain  $\alpha = (\mathbf{K} + \lambda_\alpha \mathbf{I}_n)^{-1} \mathbf{y}$ . For the AGOP we need to compute the gradient of the prediction function w.r.t. the covariates  $\mathbf{x}_i$ . For the Laplace kernel, there exist closed-form solutions which we make use of [Rad+24]. Second, we learn the GP-specific kernel parameters  $\theta$  by MLE optimization, specifically by minimizing the NLL with fixed  $M$ .



**Figure 1:** The GP-RFM algorithm operates in two stages: initially, we learn the feature matrix  $M$ , followed by the standard GP regression algorithm. This process utilizes a kernel that employs the  $M$  matrix to compute the Mahalanobis distance between data points.

**Eliminating spurious correlation.** In many datasets, there exist spurious correlations between covariates. To avoid overfitting to spurious covariate cor-

relation, we add a Ridge regularization term to the AGOP in (9) to obtain

$$\mathbf{M}^{(t+1)} = \frac{1}{n} \sum_{i=1}^n \nabla_{\mathbf{x}} f_{\mathbf{M}^{(t)}}(\mathbf{x}_i) \nabla_{\mathbf{x}} f_{\mathbf{M}^{(t)}}(\mathbf{x}_i)^T + \lambda_M \mathbf{I}_d. \quad (10)$$

The Ridge regularization acts in this case as a noise filter by shrinking off-diagonal elements of  $\mathbf{M}$  towards zero. Therefore, the learnt feature correlation in the off-diagonal elements of  $\mathbf{M}$  is only kept if it is supported by the data, making the model more robust to random variations in the data.

**Uncertainty quantification.** During inference, we can quantify the uncertainty of the GP-RFM-Laplace by computing the predictive variance  $\mathbb{V}[f(\mathbf{x}_*)]$  from (4). In traditional GPs, we have to choose the kernel function carefully to encode assumptions about the resulting function. In contrast, the GP-RFM-Laplace is able to learn features from the data and therefore it is more flexible in its assumptions about the resulting function. While Radhakrishnan et al. [Rad+24] showed the predictive power of RFMs, we show that the learnt features provide additional insight into the variability or ambiguity of the data which is crucial for uncertainty quantification.

**Implementation details.** We implement the GP-RFM-Laplace in PyTorch [Pas+19]. For the computation of the feature matrix in the first step of the training procedure, we rely on the official implementation by Radhakrishnan et al. [Rad+24]. For the GP implementation, we use GPyTorch [Gar+18] which provides a modular implementation of GPs in PyTorch. To optimize the GP parameters, we use the Adam optimizer [KB15] with a cosine annealing learning rate scheduler [LH16]. The hyperparameters we have to select are the learning rate, the Ridge regularization parameters  $\lambda_\alpha$  and  $\lambda_M$  for the solver and the AGOP, respectively. Our code will be made publicly available upon acceptance.

## 5 Experiments

**Datasets.** We evaluate our GP-RFM-Laplace on a variety of regression tasks. Specifically, we use two tabular regression benchmarks with datasets from UCI [AN07] and OpenML [Van+14], respectively. For the UCI benchmark we use 7 datasets inspired by Duan et al. [Dua+20] and for the OpenML benchmark, we utilize the collection of 16 numerical regression datasets by Grinsztajn et al. [GOV22].

**Hyperparameter tuning** We follow the protocol proposed in Hernández-Lobato and Adams [HA15] for data splitting and hyperparameter tuning. For the UCI benchmark, we follow Duan et al. [Dua+20] to hold out 10% of the data as a test set. For the OpenML benchmark, we follow Grinsztajn et al.

[GOV22] to hold out 30% of the data as a test set. The remaining data is split into a 70% training set and a 30% validation set to tune the hyperparameters. We use grid-search over all combinations of hyperparameters and select the best hyperparameters based on the validation set NLL. Details on the hyperparameter search space can be found in the appendix. Finally, we train the model on the full training set and evaluate it on the test set. The process is repeated for 20 random seeds and we report the mean and standard deviation of the results.

**Baselines.** We compare our GP-RFM-Laplace and its diagonal version GP-RFM-Laplace-diag to a variety of probabilistic baseline methods. The details are described in Appendix A. For GPs, we consider the standard *RBF* and *Laplace* kernel. As a neural networks-based GP, we regard *deep kernel learning* [Wil+16]. Additionally, we compare our method to kernels with ARD, specifically the *ARD-RBF* [Nea96] which is used in many settings and to the *ARD-Laplace* kernel. The latter is a rarely used kernel in GPs but is a natural extension of the Laplace kernel to incorporate covariate weighting, learnt through MLE optimization. Finally, we use *ARD-Laplace-full* as a direct counterpart to the RFM-Laplace with full weighting matrix but learnt through MLE here instead of AGOP [VW98].

Furthermore, we consider probabilistic extensions of boosting approaches, which are known to be powerful for predictive tasks. Firstly, we use *NG-Boost* [Dua+20] which learns the parameters of a Gaussian distribution through boosting enhanced with a natural gradient update. Secondly, we use *CatBoost-Ensemble* [MPU21] for which we use an ensemble of 10 gradient boosting-based models. From the ensemble, the predictive distribution is obtained by computing statistics of the individual predictions. Following Duan et al. [Dua+20], we standardize covariates and labels to have zero mean and unit variance for all GP-based methods but not for the boosting-based methods.

**Evaluation metrics.** We are interested in the predictive performance of the models as well as their uncertainty quantification. Therefore, we evaluate the models on their *root mean squared error (RMSE)* as well as their *NLL* on the test set. We also require the model uncertainty to be calibrated, i.e. the predictive distribution should reflect the likelihood of prediction errors. To evaluate calibration, we compute the *95% coverage error (CE)* which refers to the proportion of data points for which the 95% prediction interval does not contain the true value. For the model to be well-calibrated, the coverage should be 95% and the corresponding CE should be zero. Finally, we evaluate the *interval length (IL)* of the 95% confidence interval. This measure is important for models with similar CE since a smaller IL indicates a more precise uncertainty quantification.

## 5.1 Main results

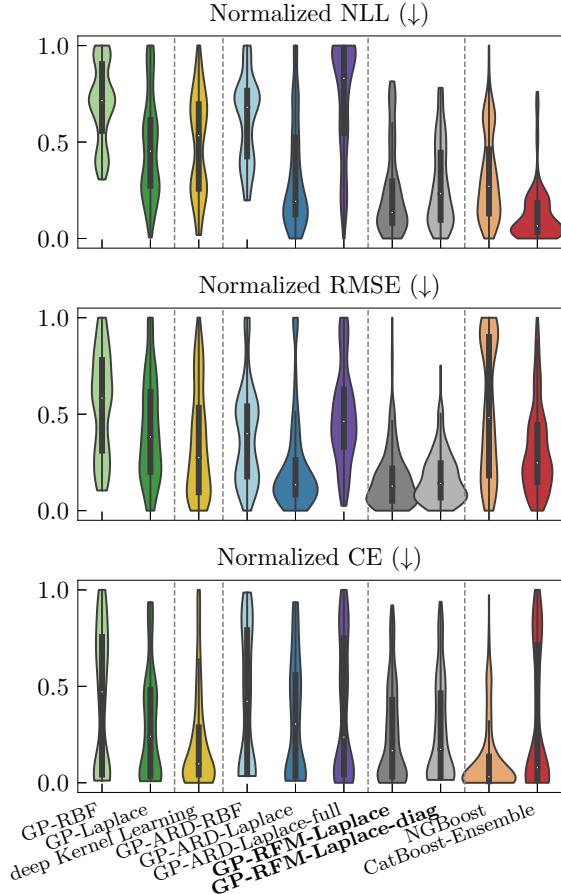
Here we present the main results of our experiments. We compare our GP-RFM-Laplace to all baseline methods on the UCI and OpenML benchmark datasets. Due to varying scales, we normalize metrics for comparison across datasets. We achieve this by calculating the minimum and maximum values for each dataset across all methods and seeds, followed by normalizing results to the range  $[0, 1]$ . The results for each dataset of the OpenML benchmark in terms of NLL are in Table 1. Summary figures for NLL, RMSE and CE are shown in Figure 2 using violin plots to indicate the distribution of the results including a boxplot for the median and quartiles. The results for the UCI benchmark are shown in Appendix B, Figure 7. Note that the results for IL are omitted from the summary figure as comparing IL across datasets is not meaningful. Detailed performance results for each method on all datasets individually can be found in the Appendix C.

We observe that both GP-RFM-Laplace variants are only outperformed by the CatBoost-Ensemble in terms of NLL. However, the GP-RFM-Laplace is the best method in terms of RMSE, closely followed by the GP-ARD-Laplace. Regarding calibration in terms of CE, we observe that the boosting methods are dominant, followed by the GP-RFM-Laplace. Overall, both the GP-RFM-Laplace and the GP-ARD-Laplace perform similarly well across all metrics, demonstrating a competitive approach to boosting-based approaches for probabilistic regression.

Notably, the ARD-Laplace-full, serving as a complement to the RFM-Laplace, exhibits a significantly poorer performance while both methods utilize full feature matrices  $M$ . Directly optimising  $M$  through MLE in the ARD-Laplace-full is challenging due to the increased complexity associated with often high-dimensional feature spaces. Hence, while the parameterization of both methods is equal, the RFM-based learning method of alternately solving convex problems seems to be simpler to optimise.

**Table 1:** Probabilistic performance measured by NLL (↓) on the OpenML benchmark. The best method for each dataset by the mean value is bolded; the second best is underlined. Detailed results for all metrics are in Appendix C.1.

Dataset	Gaussian Process						Ours				Boosting		
	ARD			RBF			RFM-diag		RFM		NGBoost		CatBoost
	RBF	Laplace	deep KL	RBF	Lap.	Lap.-full							
cpu-act	2.80 ±0.07	2.55 ±0.02	2.67 ±0.03	2.67 ±0.04	2.30 ±0.02	3.71 ±0.10	<b>2.21</b> ±0.01	<b>2.17</b> ±0.01	2.33 ±0.14	<b>2.17</b> ±0.03			
pol	3.73 ±0.01	3.43 ±0.01	3.40 ±0.01	3.07 ±0.01	2.84 ±0.01	4.41 ±0.02	<u>2.73</u> ±0.01	3.10 ±0.01	3.55 ±0.01	<b>2.09</b> ±0.03			
elevators	-4.46 ±0.03	-4.67 ±0.01	-4.85 ±0.01	-4.53 ±0.02	-4.75 ±0.01	-4.31 ±0.16	<u>4.86</u> ±0.01	<u>4.79</u> ±0.01	-4.48 ±0.02	-4.73 ±0.02			
islet	3.43 ±0.01	3.43 ±0.01	2.62 ±0.09	3.43 ±0.01	3.43 ±0.01	3.43 ±0.01	<b>2.34</b> ±0.04	2.57 ±0.02	2.71 ±0.02	<u>2.52</u> ±0.02			
wine	1.04 ±0.02	<b>0.95</b> ±0.02	<u>1.01</u> ±0.02	1.04 ±0.03	<b>0.95</b> ±0.02	<b>0.95</b> ±0.02	<b>0.95</b> ±0.02	<b>0.95</b> ±0.02	1.04 ±0.03	1.03 ±0.03			
Ailerons	-7.18 ±0.03	-7.31 ±0.01	-7.31 ±0.02	-7.19 ±0.01	-7.33 ±0.01	-6.72 ±0.00	-7.37 ±0.01	-7.36 ±0.01	<b>-7.42</b> ±0.01	-7.41 ±0.01			
houses	0.16 ±0.01	0.07 ±0.01	0.09 ±0.02	0.16 ±0.01	<u>-0.10</u> ±0.01	0.17 ±0.00	-0.07 ±0.01	-0.04 ±0.01	0.07 ±0.01	-0.12 ±0.02			
houses-16H	0.84 ±0.03	0.72 ±0.02	0.95 ±0.05	0.84 ±0.02	0.72 ±0.02	0.90 ±0.03	0.69 ±0.02	0.71 ±0.03	<u>0.57</u> ±0.05	<b>0.51</b> ±0.06			
Bra-houses	-0.99 ±0.67	-1.49 ±0.07	-0.64 ±0.04	-2.19 ±0.03	-1.82 ±0.13	0.27 ±0.04	-2.11 ±0.06	-2.07 ±0.07	-2.18 ±0.15	<b>-2.66</b> ±0.23			
bike	6.17 ±0.01	6.15 ±0.01	6.08 ±0.05	6.05 ±0.01	6.03 ±0.01	6.07 ±0.01	6.04 ±0.01	6.03 ±0.01	6.04 ±0.01	<b>5.58</b> ±0.01			
house-sales	0.04 ±0.02	-0.19 ±0.01	-0.22 ±0.02	0.00 ±0.01	-0.30 ±0.01	-0.06 ±0.16	<b>-0.32</b> ±0.01	<b>-0.32</b> ±0.01	-0.27 ±0.01	-0.31 ±0.01			
sulfur	-2.42 ±0.02	-2.83 ±0.03	-2.44 ±0.24	-2.42 ±0.02	<u>-2.83</u> ±0.04	-2.63 ±0.15	-2.80 ±0.05	-2.74 ±0.06	-2.59 ±0.41	<b>-2.88</b> ±0.08			
Miami2016	0.01 ±0.00	-0.36 ±0.01	-0.35 ±0.02	0.00 ±0.00	-0.46 ±0.01	-0.17 ±0.18	-0.47 ±0.01	-0.50 ±0.01	-0.38 ±0.01	<b>-0.53</b> ±0.01			
superconduct	4.21 ±0.00	4.01 ±0.01	4.10 ±0.03	4.20 ±0.00	3.96 ±0.01	4.34 ±0.18	4.05 ±0.11	4.03 ±0.01	<u>3.65</u> ±0.02	<b>3.46</b> ±0.11			
california	-0.31 ±0.01	-0.40 ±0.01	-0.34 ±0.09	-0.32 ±0.01	<b>-0.64</b> ±0.01	-0.40 ±0.03	-0.60 ±0.01	<u>-0.61</u> ±0.01	-0.45 ±0.01	-0.59 ±0.02			
fifa	1.27 ±0.01	1.24 ±0.01	1.19 ±0.01	1.23 ±0.01	1.22 ±0.01	1.19 ±0.01	1.21 ±0.02	1.19 ±0.01	<b>1.09</b> ±0.01	1.10 ±0.02			



**Figure 2:** Violin plot results on the OpenML benchmark including boxplots with median and quartiles for each method.

## 5.2 Toy data set

Given the qualitatively similar performance of the GP-RFM-Laplace and its diagonal version, we investigate the differences between the two methods in more detail. Mathematically, in the RFM-Laplace-diag we restrict the feature matrix  $\mathbf{M}$  to be diagonal. Therefore, the RFM-Laplace-diag is a special case of the RFM-Laplace where the latter can additionally capture covariate correlations which are relevant for the predictive task.

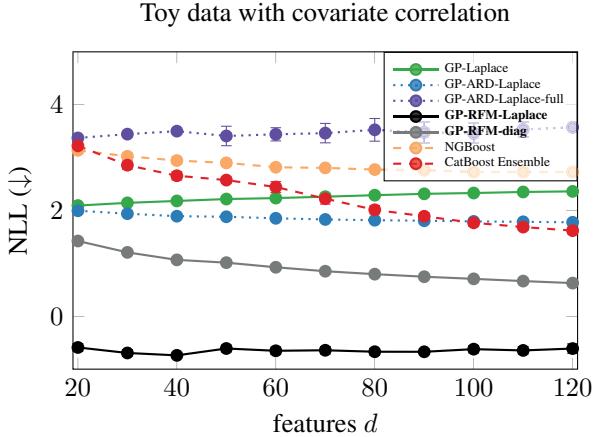
To highlight the advantage of the RFM-Laplace, we create a toy dataset. The covariates  $\mathbf{x}$  are independent and the labels  $y$  are nonlinearly transformed using

the first 10 covariates

$$\mathbf{x} \sim \mathcal{U}(0_d, 1_d); \quad y = \left( \sum_{i=1}^{10} \mathbf{x}_{[i]} \right)^2. \quad (11)$$

This dataset is designed to be challenging for the methods which do not consider covariate correlation. We compare the performance for a range of feature sizes in Figure 3; results for RMSE on all methods can be found in the Figure 6.

We observe that the GP-RFM-Laplace outperforms all methods for all covariate dimensions. This demonstrates that a non-diagonal metric in the RFM-Laplace in contrast to diagonal metrics used in kernels with ARD can benefit the performance considerably and has been underexplored in the community. Furthermore, the results in Table 1 and Figure 2 show that no GP-based method outperforms the RFM-Laplace. However, for many datasets, the diagonal kernel with ARD performs similarly well. Therefore, we conjecture that in many real-world datasets, there is either little covariate correlation or the covariate correlation is not relevant for the predictive task. For datasets where the GP-RFM-Laplace considerably outperforms the GP-ARD-Laplace, such as the ‘isolet’ (Isolated Letter Speech Recognition) dataset from OpenML, we observe that there is indeed considerable covariate correlation.



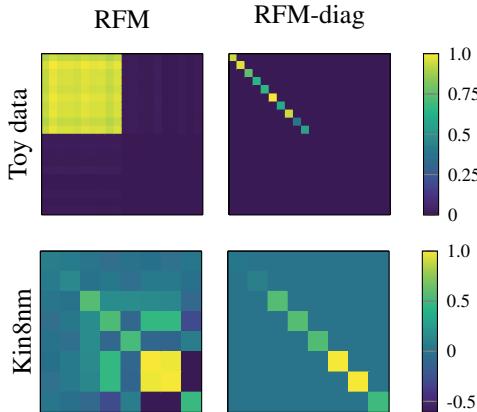
**Figure 3:** Toy dataset with covariate correlation for prediction. We scale the number of train samples with  $n = 20d$ .

### 5.3 Visualizing feature matrices

To get a better understanding of the learnt feature matrix  $\mathbf{M}$ , we visualize the normalized feature matrices for the RFM-Laplace and its diagonal version

RFM-Laplace-diag in Figure 4. On the top row, we compare both methods for the toy dataset, where we generated the labels with correlating covariates according to (11). For this dataset, we can compute the true feature matrix through the Jacobian of the labels with respect to the covariates to obtain the true feature matrix  $\mathbf{M}$ . The true feature matrix is a block matrix with a  $10 \times 10$  block of  $\frac{1}{n} \sum_{i=1}^n (\sum_{j=1}^{10} \mathbf{x}_{i[j]})^2$  and the remaining entries are zero, where  $\mathbf{x}_{i[j]}$  denotes the  $j$ th dimension of the  $i$ th sample. It is necessary to learn this non-zero block to capture the relevant covariate correlation. Experimentally, as we expected, in Figure 4 the RFM-Laplace learns relevant covariate correlation as indicated by nonzero off-diagonal values of the feature matrix while the diagonal methods are unable to capture this relation.

On the bottom row, we compare both methods on the Kin8nm dataset from the UCI benchmark. In this real-world dataset, the RFM-Laplace captures the non-zero covariate correlation and focuses on a low-dimensional set of covariates. This ability of RFMs to learn low-dimensional features has been proven for linear RFMs in Radhakrishnan et al. [RBD24]. Additionally, we can qualitatively see that the RFM-Laplace learns the same diagonal covariate re-weighing as the RFM-Laplace-diag. Therefore, the RFM-Laplace is a direct generalisation of the RFM-Laplace-diag and can learn more complex features, which allows for both of these datasets to be predicted more accurately.



**Figure 4:** Normalized feature matrices for toy data (top) and Kin8nm dataset from UCI benchmark (bottom).

## 5.4 Out-of-distribution data

Having established that the GP-RFM-Laplace is a competitive method for probabilistic regression, we now investigate its performance on out-of-distribution

(OOD) data. Distribution shift depicts a common scenario in real-world applications where for example the test data distribution changes over time. One hope of utilising a probabilistic model is to obtain more reliable predictions by indicating when the model is uncertain about its predictions. Understanding how well the GP-RFM-Laplace performs in such scenarios is essential for assessing its robustness and applicability in real-world settings.

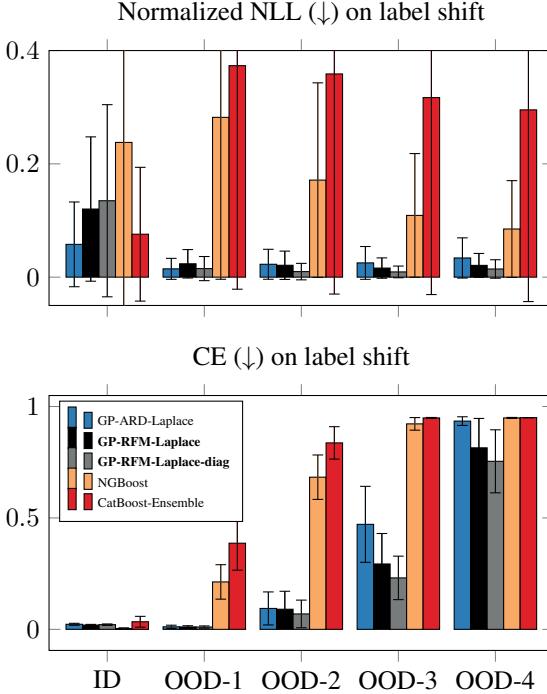
In our setting, we concentrate on real-world data shifts. Here, we focus on label shifts, i.e. the marginal distribution of the labels  $p(y)$  change, while in the Appendix B.7 we also consider covariate shifts, i.e. the marginal distribution of the covariates  $p(\mathbf{x})$  change. Specifically, we take four house datasets from the OpenML benchmark for which the labels describe the house value and include a covariate for latitude and longitude. We define the ID data such that  $p(y > a) = 0.7$  where  $a$  is the 70% quantile of the labels and the OOD data such that  $p(y < a)$ . We then split the OOD data into four consecutive non-overlapping datasets, where each OOD dataset contains 7.5% of the data. This results in one ID dataset and four OOD datasets (denoted with OOD-1 to OOD-4) with increasingly severe label shifts.

Figure 5 shows the results on ID and OOD data for different methods. We notice in the top figure that the NLL of the boosting-based method rises with increasing severity of label shift while the GP-based methods improve. Overall, the GP-based methods including the GP-RFM are the most robust. This reliability is confirmed by the lower CE of the GP-based methods which shows that the model is better calibrated under label shift, see Figure 5 (bottom). We have to note that for large distribution shifts, none of the methods are calibrated anymore. Generally, our results indicate that Boosting-based methods are less robust to label shifts as defined in our scenario.

## 6 Discussion and future work

In this study, we adopted the RFM—a novel data-adaptive, feature learning kernel—for uncertainty quantification through integration into GPs. We rigorously tested our method across various datasets and metrics to ensure consistency. Our results demonstrate that our RFM-based GP can either outperform or match the performance of existing state-of-the-art methods, including boosting-based approaches such as NGBoost [Dua+20] and CatBoost-ensembles [MPU21].

In the GP literature, there is a focus on ARD-based approaches or low-rank feature matrices  $\mathbf{M}$  [GOH14; Let+20]. We show and provide examples illustrating that the presented GP-RFM with full feature matrix  $\mathbf{M}$  outperforms these approaches since it is able to reliably model relevant covariate correlation. We



**Figure 5:** Out-of-distribution experiment: NLL (top) and CE (bottom) on four house datasets with label shift. We show mean and standard deviation.

therefore bridge fields and demonstrate an approach that the GP community has been missing.

However, our empirical findings suggest that RFMs might occasionally be surpassed by their diagonal version, RFM-diag or kernels with ARD. We observed that sample complexity plays a pivotal role in this behaviour. Given sufficient training samples, leveraging the capabilities of RFM is always preferable. However, in cases where the sample size is limited, the diagonal RFM can be preferable. While delving deeper into determining the optimal method for various settings is beyond the scope of this paper, it presents a crucial direction for future research.

Another line of future research is to integrate more intriguing kernels within the RFM framework. RFM is a broad feature learning framework based on kernels, suitable for any radial kernel. This study primarily concentrates on the two most prevalent kernels: RBF and Laplacian. The results clearly show that the Laplacian outperforms the Gaussian kernel. There is potential to select a task-specific kernel to further enhance these performances. For example, Neural Tangent Kernels (NTK) [JGH18] or Convolutional Neural Tangent Kernels (CNTK) [Li+19].

Another crucial aspect is scalability. Decision tree-boosting methods are naturally adept at handling large datasets. On the other hand, kernel methods historically have faced challenges in scaling. However, with the advent of recent state-of-the-art techniques, scaling kernels has become feasible. Notable examples are the EigenPro series [MB17; MB19; ABP23] and FALKON [RCR17; Mea+20]. These advancements can enable our method to scale effectively to large datasets.

**Acknowledgments** This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The computations were enabled by the Berzelius resource provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre, Sweden.

## References

- [ABP23] A. Abedsoltan, M. Belkin, and P. Pandit. “Toward Large Kernel Models.” In: *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202. Proceedings of Machine Learning Research. 2023, pp. 61–78 (cit. on p. I-17).
- [AN07] A. Asuncion and D. Newman. *UCI machine learning repository*. 2007 (cit. on p. I-8).
- [Aro50] N. Aronszajn. “Theory of reproducing kernels.” In: *Transactions of the American mathematical society* 68.3 (1950), pp. 337–404 (cit. on p. I-4).
- [Ava+18] A. Avati, K. Jung, S. Harman, L. Downing, A. Ng, and N. H. Shah. “Improving palliative care with deep learning.” In: *BMC medical informatics and decision making* 18.4 (2018), pp. 55–64 (cit. on p. I-1).
- [Bar+22] A. Barragán-Montero, A. Bibal, M. H. Dastarac, C. Draguet, G. Valdes, D. Nguyen, S. Willems, L. Vandewincke, M. Holmström, F. Löfman, et al. “Towards a safe and efficient clinical implementation of machine learning in radiation oncology by exploring model interpretability, explainability and data-model dependency.” In: *Physics in Medicine & Biology* 67.11 (2022), 11TR01 (cit. on p. I-1).
- [Blu+15] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. “Weight uncertainty in neural network.” In: *International conference on machine learning*. PMLR. 2015, pp. 1613–1622 (cit. on p. I-3).

- [Cal+16] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. “Manifold Gaussian processes for regression.” In: *2016 International joint conference on neural networks (IJCNN)*. IEEE. 2016, pp. 3338–3345 (cit. on p. I-3).
- [CG16] T. Chen and C. Guestrin. “Xgboost: A scalable tree boosting system.” In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794 (cit. on p. I-3).
- [Chu+23] M. Chua, D. Kim, J. Choi, N. G. Lee, V. Deshpande, J. Schwab, M. H. Lev, R. G. Gonzalez, M. S. Gee, and S. Do. “Tackling prediction uncertainty in machine learning for healthcare.” In: *Nature Biomedical Engineering* 7.6 (2023), pp. 711–718 (cit. on p. I-1).
- [Dol+22] J. M. Dolezal, A. Srisuwananukorn, D. Karpeyev, S. Ramesh, S. Kochanny, B. Cody, A. S. Mansfield, S. Rakshit, R. Bansal, M. C. Bois, et al. “Uncertainty-informed deep learning models enable high-confidence predictions for digital histopathology.” In: *Nature communications* 13.1 (2022), p. 6572 (cit. on p. I-1).
- [Dua+20] T. Duan, A. Anand, D. Y. Ding, K. K. Thai, S. Basu, A. Ng, and A. Schuler. “Ngboost: Natural gradient boosting for probabilistic prediction.” In: *International conference on machine learning*. PMLR. 2020, pp. 2690–2700 (cit. on pp. I-2, I-3, I-8, I-9, I-15).
- [EJ21] D. Eriksson and M. Jankowiak. “High-dimensional Bayesian optimization with sparse axis-aligned subspaces.” In: *Uncertainty in Artificial Intelligence*. PMLR. 2021, pp. 493–503 (cit. on p. I-3).
- [Fri01] J. H. Friedman. “Greedy function approximation: a gradient boosting machine.” In: *Annals of statistics* (2001), pp. 1189–1232 (cit. on p. I-3).
- [FS95] Y. Freund and R. E. Schapire. “A desicion-theoretic generalization of on-line learning and an application to boosting.” In: *European conference on computational learning theory*. Springer. 1995, pp. 23–37 (cit. on p. I-3).
- [Gar+18] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. “Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration.” In: *Advances in neural information processing systems* 31 (2018) (cit. on pp. I-8, I-23).
- [GDS20] F. K. Gustafsson, M. Danelljan, and T. B. Schön. “Evaluating scalable bayesian deep learning methods for robust computer vision.” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 318–319 (cit. on p. I-4).

- [GG16] Y. Gal and Z. Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning.” In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059 (cit. on p. I-4).
- [GK14] T. Gneiting and M. Katzfuss. “Probabilistic Forecasting.” In: *Annual Review of Statistics and Its Application* 1.1 (2014), pp. 125–151 (cit. on p. I-1).
- [GOH14] R. Garnett, M. A. Osborne, and P. Hennig. “Active Learning of Linear Embeddings for Gaussian Processes.” In: *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*. Arlington, Virginia, USA: AUAI Press, 2014, pp. 230–239 (cit. on pp. I-3, I-15).
- [GOV22] L. Grinsztajn, E. Oyallon, and G. Varoquaux. “Why do tree-based models still outperform deep learning on typical tabular data?” In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 507–520 (cit. on pp. I-3, I-8).
- [Gra11] A. Graves. “Practical variational inference for neural networks.” In: *Advances in neural information processing systems* 24 (2011) (cit. on p. I-3).
- [HA15] J. M. Hernández-Lobato and R. Adams. “Probabilistic back-propagation for scalable learning of bayesian neural networks.” In: *International conference on machine learning*. PMLR. 2015, pp. 1861–1869 (cit. on p. I-8).
- [Hua+15] W. Huang, D. Zhao, F. Sun, H. Liu, and E. Chang. “Scalable Gaussian process regression using deep neural networks.” In: *Twenty-fourth international joint conference on artificial intelligence*. 2015 (cit. on p. I-3).
- [JGH18] A. Jacot, F. Gabriel, and C. Hongler. “Neural tangent kernel: Convergence and generalization in neural networks.” In: *Advances in neural information processing systems* 31 (2018) (cit. on p. I-16).
- [KB15] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization.” In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015 (cit. on pp. I-8, I-23).
- [Ke+17] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. “Lightgbm: A highly efficient gradient boosting decision tree.” In: *Advances in neural information processing systems* 30 (2017) (cit. on p. I-3).
- [KSB21] B. Kompa, J. Snoek, and A. L. Beam. “Second opinion needed: communicating uncertainty in medical machine learning.” In: *NPJ Digital Medicine* 4.1 (2021), p. 4 (cit. on p. I-1).

- [KW70] G. S. Kimeldorf and G. Wahba. “A correspondence between Bayesian estimation on stochastic processes and smoothing by splines.” In: *The Annals of Mathematical Statistics* 41.2 (1970), pp. 495–502 (cit. on p. I-4).
- [Let+20] B. Letham, R. Calandra, A. Rai, and E. Bakshy. “Re-examining linear embeddings for high-dimensional Bayesian optimization.” In: *Advances in neural information processing systems* 33 (2020), pp. 1546–1558 (cit. on pp. I-3, I-15).
- [LH16] I. Loshchilov and F. Hutter. “Sgdr: Stochastic gradient descent with warm restarts.” In: *arXiv preprint arXiv:1608.03983* (2016) (cit. on pp. I-8, I-23).
- [Li+19] Z. Li, R. Wang, D. Yu, S. S. Du, W. Hu, R. Salakhutdinov, and S. Arora. “Enhanced convolutional neural tangent kernels.” In: *arXiv preprint arXiv:1911.00809* (2019) (cit. on p. I-16).
- [Lin+23] J. Linmans, S. Elfwing, J. van der Laak, and G. Litjens. “Predictive uncertainty estimation for out-of-distribution detection in digital pathology.” In: *Medical Image Analysis* 83 (2023), p. 102655 (cit. on p. I-1).
- [Liu+20] J. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax Weiss, and B. Lakshminarayanan. “Simple and principled uncertainty estimation with deterministic deep learning via distance awareness.” In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7498–7512 (cit. on p. I-3).
- [LPB17] B. Lakshminarayanan, A. Pritzel, and C. Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles.” In: *Advances in neural information processing systems* 30 (2017) (cit. on p. I-4).
- [Mac92] D. J. MacKay. “Bayesian interpolation.” In: *Neural computation* 4.3 (1992), pp. 415–447 (cit. on p. I-3).
- [Mac98] D. J. MacKay. “Introduction to Gaussian processes.” In: *NATO ASI series F computer and systems sciences* 168 (1998), pp. 133–166 (cit. on p. I-3).
- [MB17] S. Ma and M. Belkin. “Diving into the shallows: a computational perspective on large-scale shallow learning.” In: *Advances in neural information processing systems* 30 (2017) (cit. on p. I-17).
- [MB19] S. Ma and M. Belkin. “Kernel machines that adapt to GPUs for effective large batch training.” In: *Proceedings of Machine Learning and Systems* 1 (2019), pp. 360–373 (cit. on p. I-17).

- [McE+23] D. McElfresh, S. Khandagale, J. Valverde, G. Ramakrishnan, M. Goldblum, C. White, et al. “When Do Neural Nets Outperform Boosted Trees on Tabular Data?” In: *arXiv preprint arXiv:2305.02997* (2023) (cit. on p. I-3).
- [Mea+20] G. Meanti, L. Carratino, L. Rosasco, and A. Rudi. “Kernel methods through the roof: handling billions of points efficiently.” In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 14410–14422 (cit. on p. I-17).
- [MPU21] A. Malinin, L. Prokhorenkova, and A. Ustimenko. “Uncertainty in Gradient Boosting via Ensembles.” In: *International Conference on Learning Representations*. 2021 (cit. on pp. I-3, I-9, I-15, I-23).
- [Nea96] R. M. Neal. *Bayesian Learning for Neural Networks*. 1996 (cit. on pp. I-3, I-6, I-9).
- [Nic+22] G. Nicora, M. Rios, A. Abu-Hanna, and R. Bellazzi. “Evaluating pointwise reliability of machine learning prediction.” In: *Journal of Biomedical Informatics* 127 (2022), p. 103996 (cit. on p. I-1).
- [Ova+19] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek. “Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift.” In: *Advances in neural information processing systems* 32 (2019) (cit. on p. I-4).
- [Pas+19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. “Pytorch: An imperative style, high-performance deep learning library.” In: *Advances in neural information processing systems* 32 (2019) (cit. on p. I-8).
- [Pro+18] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. “CatBoost: unbiased boosting with categorical features.” In: *Advances in neural information processing systems* 31 (2018) (cit. on pp. I-2, I-3).
- [Rad+24] A. Radhakrishnan, D. Beaglehole, P. Pandit, and M. Belkin. “Mechanism for feature learning in neural networks and backpropagation-free machine learning models.” In: *Science* 383.6690 (2024), pp. 1461–1467 (cit. on pp. I-1, I-3, I-4, I-6, I-7, I-8).
- [RBD24] A. Radhakrishnan, M. Belkin, and D. Drusvyatskiy. “Linear Recursive Feature Machines provably recover low-rank matrices.” In: *arXiv preprint arXiv:2401.04553* (2024) (cit. on p. I-14).

- [RCR17] A. Rudi, L. Carratino, and L. Rosasco. “Falkon: An optimal large scale kernel method.” In: *Advances in neural information processing systems* 30 (2017) (cit. on p. I-17).
- [RW06] C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning*. Vol. 1. Springer, 2006 (cit. on pp. I-2, I-3).
- [SA22] R. Shwartz-Ziv and A. Armon. “Tabular data: Deep learning is not all you need.” In: *Information Fusion* 81 (2022), pp. 84–90 (cit. on p. I-3).
- [Sch+19] L. Schlosser, T. Hothorn, R. Stauffer, and A. Zeileis. “Distributional regression forests for probabilistic precipitation forecasting in complex terrain.” In: *The Annals of Applied Statistics* 13.3 (2019), pp. 1564–1589 (cit. on p. I-3).
- [SH20] M. H. Shaker and E. Hüllermeier. “Aleatoric and epistemic uncertainty with random forests.” In: *Advances in Intelligent Data Analysis XVIII: 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27–29, 2020, Proceedings* 18. Springer, 2020, pp. 444–456 (cit. on p. I-3).
- [SS02] B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002 (cit. on p. I-4).
- [Tra+21] K. A. Tran, O. Kondrashova, A. Bradley, E. D. Williams, J. V. Pearson, and N. Waddell. “Deep learning in cancer diagnosis, prognosis and treatment selection.” In: *Genome Medicine* 13.1 (2021), pp. 1–17 (cit. on p. I-1).
- [Van+14] J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo. “OpenML: networked science in machine learning.” In: *ACM SIGKDD Explorations Newsletter* 15.2 (2014), pp. 49–60 (cit. on p. I-8).
- [VW98] F. Vivarelli and C. Williams. “Discovering hidden features with Gaussian processes regression.” In: *Advances in Neural Information Processing Systems* 11 (1998) (cit. on pp. I-3, I-9).
- [Wil+16] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. “Deep kernel learning.” In: *Artificial intelligence and statistics*. PMLR. 2016, pp. 370–378 (cit. on pp. I-3, I-9, I-23).
- [WN15] A. Wilson and H. Nickisch. “Kernel interpolation for scalable structured Gaussian processes (KISS-GP).” In: *International conference on machine learning*. PMLR. 2015, pp. 1775–1784 (cit. on p. I-23).
- [WT11] M. Welling and Y. W. Teh. “Bayesian learning via stochastic gradient Langevin dynamics.” In: *Proceedings of the 28th international conference on machine learning*. 2011, pp. 681–688 (cit. on p. I-4).

# Appendix

## A Implementation details

### A.1 Model and training details

**Gaussian process.** For the mean, we use a constant function. For the covariance, we use the concatenation of a scale kernel with the respective RBF/Laplace or Mahalanobis distance kernel. All GP-based methods are optimized with Adam optimizer over 250 epochs [KB15] with a cosine annealing learning rate schedule [LH16] to a minimum learning rate of  $\text{lr}_{min} = 10^{-7}$  and with all data points in the training set as a mini-batch using GPyTorch [Gar+18].

**Deep kernel learning.** For the mean, we use a constant function. For the covariance, we follow the GPyTorch tutorial implementation of deep kernel learning<sup>1</sup>. Our model consists of a ReLU fully-connected deep neural network with dimensions  $\{d, 1000, 500, 50, 2\}$  as in Wilson et al. [Wil+16]. Notably by following the GPyTorch implementation, we do not pre-train the deep neural network. Further, for a fair comparison with other GP-based methods, we did not use the ideas of KISS-GP [WN15]. Hence, we use the same optimization scheme as for all other GPs, but we reduced the number of epochs to 100 due to stability issues for longer training schemes.

**GP-ARD-Laplace-full.** For the mean, we use a constant function. For the covariance, we use the Mahalanobis distance kernel from (7). To stabilize training, we decompose the Mahalanobis distance as  $\mathbf{M} = \mathbf{U} + \mathbf{U}^\top + \mathbf{D}$ . Here,  $\mathbf{U}$  is a learnable upper triangular matrix of small values to enforce symmetry and the learnable  $\mathbf{D}$  is a diagonal matrix to focus initialization on the diagonal, similar to the RFM. We use the same optimization scheme as for all other GPs, but we reduced the number of epochs to 100 due to stability issues for longer training schemes.

**NGBoost.** We use the NGBoost regressor from the official implementations of the respective authors<sup>2</sup>. For this model, we use the default set of hyperparameters.

**CatBoost-ensemble.** We use the CatBoost regressor from the official implementation of the respective authors<sup>3</sup>. We choose to select 10 ensemble members each consisting of 1000 trees as done similarly in Malinin et al. [MPU21].

<sup>1</sup>[https://docs.gpytorch.ai/en/stable/examples/06\\_PyTorch\\_NN\\_Integration\\_DKL/KISSGP\\_Deep\\_Kernel\\_Regression\\_CUDA.html](https://docs.gpytorch.ai/en/stable/examples/06_PyTorch_NN_Integration_DKL/KISSGP_Deep_Kernel_Regression_CUDA.html), accessed 05.02.2024.

<sup>2</sup><https://stanfordmlgroup.github.io/ngboost/>, accessed 05.02.2024.

<sup>3</sup><https://catboost.ai/>, accessed 05.02.2024.

Furthermore, we consider the depth of the trees to be 6 and keep the remaining default hyperparameters

## A.2 Hyperparameter search

For our main results, we perform a hyperparameter search for the specific hyperparameters of each method. We run a grid search over the hyperparameters and select the best ones based on the validation set’s NLL value.

**Gaussian processes.** The only hyperparameter is the learning rate which we optimize over the values  $\text{lr} = \{0.05, 0.01, 0.005, 0.001\}$ . The learning rate is decreased to a minimal value of  $10^{-7}$ .

**Recursive feature machines.** For the RFM, we optimize the hyperparameters of the GP-based methods as described above. Additionally, we optimize the Ridge regularization for the optimization of  $\alpha$  over the values  $\lambda_\alpha = \{0.5, 0.1, 0.5, 0.01, 0.001, 0.0001\}$ . Furthermore, we optimize the Ridge regularization for the optimization of  $M$  over the values  $\lambda_M = \{0.1, 0.01, 0.001, 5 \cdot 10^{-5}, 10^{-6}, 10^{-7}, 0\}$ .

**Boosting-based.** For NGBoost, we optimize the number of estimators from  $\{100, 200, 300, 400, 500\}$ . For CatBoost-ensembles, we optimize the learning rate over the values  $\text{lr} = \{0.05, 0.01, 0.005, 0.001\}$

## A.3 Post-processing

Some methods did not converge for some seeds and datasets. The metrics computed from these runs would heavily distort the actual results but are easy to detect in practice. To evaluate only successful runs, we remove outliers for each dataset and metric individually. We achieve this by removing entries with z-values  $\geq 3.5$ . Almost exclusively the results from deep kernel learning and the GP-ARD-Laplace-full are affected by this post-processing.

# B Additional experimental results

## B.1 Computational infrastructure

All experiments are run on a single NVIDIA A100 GPU with 40GB memory. The GPU is part of an internal cluster supported by local resources. To run the experiments for all methods on all datasets in a sequence of our used OpenML benchmark, we require approximately 1 hour of computation time for one seed.

For all methods on all datasets in a sequence of the UCI benchmark, we require approximately 10 minutes of computation time for one seed.

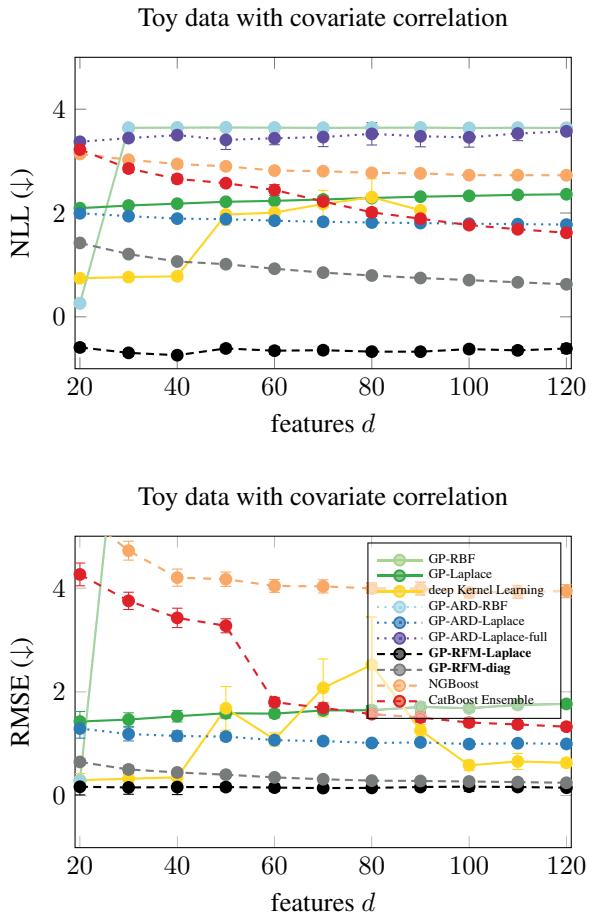
## B.2 Main results

Complementary to the main results, in Figure 7 we list the normalized results for the OpenML benchmark and the UCI benchmark. Qualitatively the observations from the OpenML benchmark also hold for the UCI benchmark.

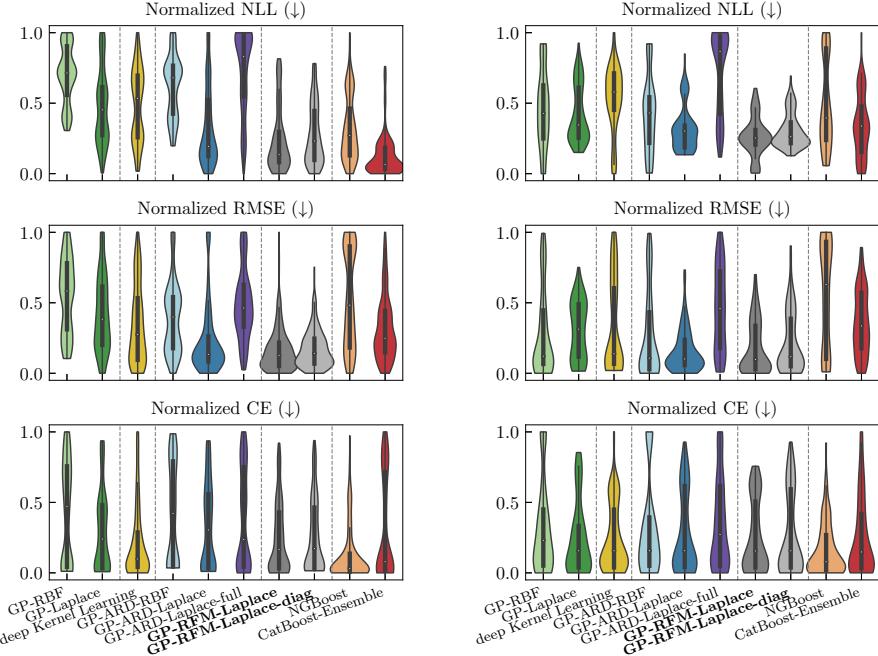
Additionally to the metrics NLL, RMSE and CE, in Figures 7c and 7d we show the combined time for training the model and prediction on the test set. We have to note that the GP-based methods utilize the GPytorch library which enables GPU utilization. For NGBoost and CatBoost-ensembles, the official implementations do not allow for GPU utilization. Therefore, the time comparison is on the one side biased because we utilize different hardware, on the other side it utilizes the best openly available implementations for all respective methods. Note, that here, we excluded the method ‘deep Kernel learning’. This method reduces the GP dimensionality to 2 dimensions because of the neural networks extractor and is therefore the fastest method. Including it in the violin plot would distort the visualization. Detailed timing results for each method on every dataset can be found in Appendix C.

## B.3 Toy data set

We show results from the toy dataset which was designed to be difficult for methods which do not capture feature correlation. This includes diagonal feature weighting such as the ARD-based methods. We included correlation to be modelled into the data by choosing  $\mathbf{x} = \mathcal{U}(0_d, 1_d)$  and  $y = (\sum_{i=1}^1 0\mathbf{x}_{[i]})^2$ . In Figure 6, we plot additionally to the NLL also the RMSE which show a qualitatively similar behaviour such that the GP-RFM-Laplace outperforms other methods. Notably the diagonal method, GP-RFM-diag becomes close for high feature dimensions. We argue that this is because we fix the number of dimensions which are relevant for the prediction but grow the actual dimension. Hence, fewer relative dimensions become relevant. However, the diagonal method will never outperform the GP-RFM-Laplace on this toy dataset.

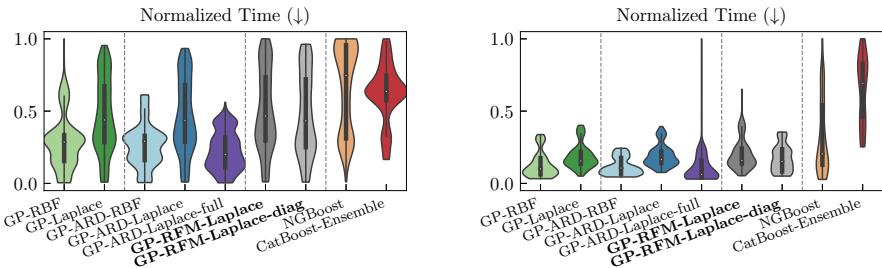


**Figure 6:** Toy data set with varying feature dimensions. NLL (left) is a repetition of the main text figure; RMSE (right) shows a similar pattern. Note that for NLL the deep Kernel Learning blows up at  $d = 100$ , hence these values are omitted. Similarly for RMSE, the GP-ARD-Laplace-full  $> 5$  and therefore not depicted.



**(a)** OpenML benchmark datasets results.

**(b)** UCI benchmark results.



**(c)** OpenML benchmark results: Timing.

**(d)** UCI benchmark results: Timing.

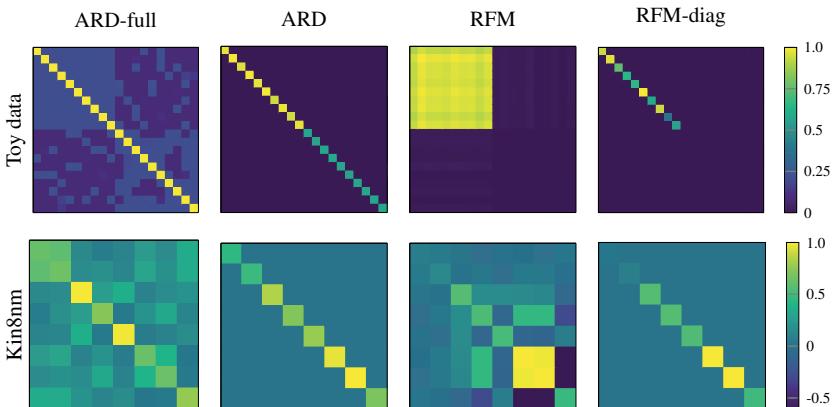
**Figure 7:** Violin plot results on OpenML benchmark datasets and UCI benchmark. Note that Figure 7a is a repetition of Figure 2 from the main text. Note also that in Figures 7c and 7d we excluded the method ‘deep Kernel Learning’ since it is the fastest and distorts the visualization in the violin plots.

## B.4 Visualizing feature matrices

In Figure 8, we compare the learnt feature matrices  $M$  of the RFM-based methods with the ones from Kernels with ARD on the toy dataset. The only method which can capture the necessary feature correlation is the GP-RFM-Laplace. Notably, the GP-ARD-Laplace-full is not able to learn the correlation despite its structure ability through parameterization with a full feature matrix  $M$ . This might be justified by the more complicated optimization problem resulting in poor performance as Figure 6 indicates for this method.

For the diagonal methods, the GP-RFM-Laplace-diag capture the exact dimensionality of the problem by weighting all irrelevant dimensions in the toy problem with zeros. In contrast, the GP-ARD-Laplace also capture the necessary dimensions but does not suppress irrelevant dimensions to zero. We experimented with increasing the compute budget for this method from 250 epochs up to 2,000 epochs which reduces the weighting of the irrelevant dimensions but does get close to zero weighting.

Similar conclusions can be drawn about the bottom row for real data. Again, we observe that the GP-ARD-Laplace-full struggles with the task. The remaining three methods learn similar features on the diagonal but only the GP-RFM-Laplace can model the necessary correlation to achieve the best performance.



**Figure 8:** Normalized feature matrices for toy data (top) and Kin8nm dataset from UCI benchmark (bottom). Note that the two right-most columns are a repetition of Figure 4.

## B.5 Comparison of learnt features

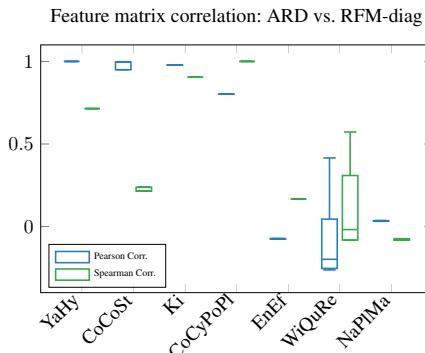
Given the comparable performance of the GP-RFM-Laplace and the GP-ARD-Laplace and their similar mathematical structure based on the Mahalanobis

distance kernel, a question arises whether the learnt features in  $M$  are similar. To investigate this, we compare the Pearson and Spearman correlation between the feature matrices  $M$  learnt in RFM-based kernels and kernels with ARD:

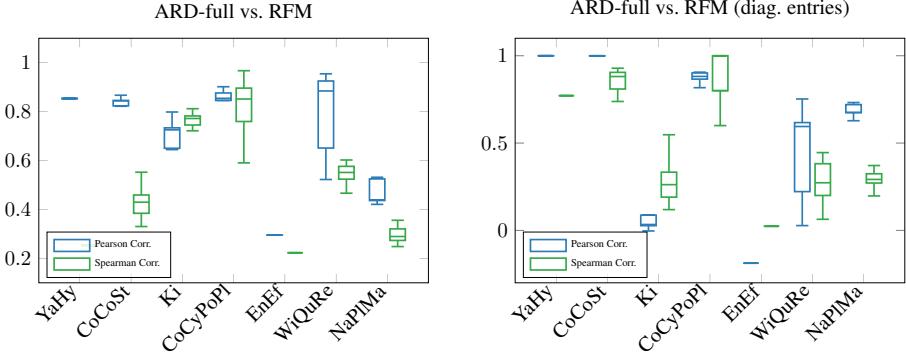
- **Diagonal methods:** We compare diagonal of  $M$  from the GP-RFM-Laplace-diag with GP-ARD-Laplace.
- **Non-diagonal methods:** Here we perform two comparisons. (1) we compare the full feature matrix  $M$  of GP-RFM-Laplace with the one of GP-ARD-Laplace-full. (2) to capture how the features are re-weighted, we also compare the diagonal of the full feature matrix  $M$  of both methods.

Comparing diagonal methods separately from methods which learn the full  $M$  to disentangle the effects of the parameterization and the learning paradigm. We compare methods with the same parameterization. Hence, the main difference lies in the feature learning procedure: the RFM-based kernels learn the features through AGOP iterations while the ARD-based kernels learn the features through MLE optimization.

Figure 9 shows the Pearson and Spearman correlation between the diagonal methods and Figure 10 on the non-diagonal methods on the UCI benchmark dataset. There is the same trend for the diagonal and the non-diagonal methods: For some datasets, there is a high correlation between the RFM-based kernel and the kernels with ARD, but there are also datasets where the feature correlation is low. This indicates that learning the features with AGOP iterations in the RFM or with MLE optimization in the ARD-based kernel may result in the same features in some cases but is not guaranteed to do so. The similarity between the learning paradigms opens up investigations of the widely applied MLE framework from a different perspective. Further investigation is required to understand the differences between feature learning in the two paradigms.



**Figure 9:** Correlation of feature matrices  $M$  between diagonal methods GP-RFM-Laplace-diag and GP-ARD-Laplace.



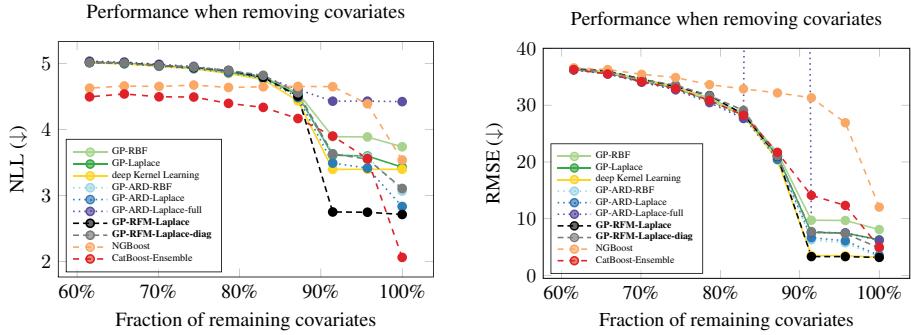
**Figure 10:** Correlation of feature matrices  $M$  between non-diagonal methods GP-RFM-Laplace and GP-ARD-Laplace-full. Left: full  $M$  of both methods. Right: diagonal  $M$  of both methods.

## B.6 Feature importance

Here, we analyse if the features learnt by the RFM are meaningful in the sense of weighting the covariates with the highest predictive power. We consider the feature matrix  $M$  of a trained GP-RFM-Laplace and successively remove the covariates with the highest weight of  $\text{diag}(M)$ . Then, we re-train and evaluate all models on the reduced dataset. Specifically, we use the ‘pol’ dataset from the OpenML benchmarks since it has a high number of covariates (26). In Figure 11 we observe that the NLL and RMSE increase for all methods when removing the most important covariates but this plateaus. Additionally, we observe that the two most important covariates according to the RFM feature matrix can be removed without a significant increase in NLL and RMSE. This indicates that for this dataset the two highest weighted covariates are equally important for the prediction and contain most of the predictive power. The sharp increase in NLL and RMSE after removing more covariates indicates that the RFM feature matrix can identify the most important covariates for the prediction.

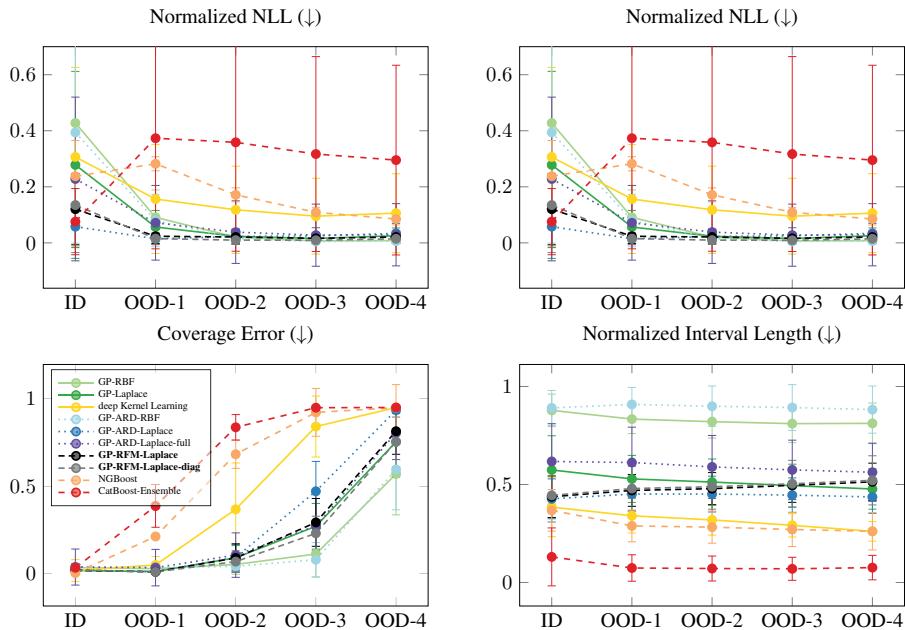
## B.7 Distribution shift

In the main text, we show the results of label shift in terms of normalized NLL and CE for some selected methods. Here, we additionally present results for normalized RMSE and normalized interval length in Figure 12. Furthermore, we experimented with covariate shifts. Specifically, we consider the covariate for the latitude of the house location. Similarly to the label shift we define the ID data such that  $p(x_{lat} < a) = 0.7$  where  $a$  is the 70% quantile of the labels and the OOD data such that  $p(x_{lat} > a)$ . We split the OOD data into four

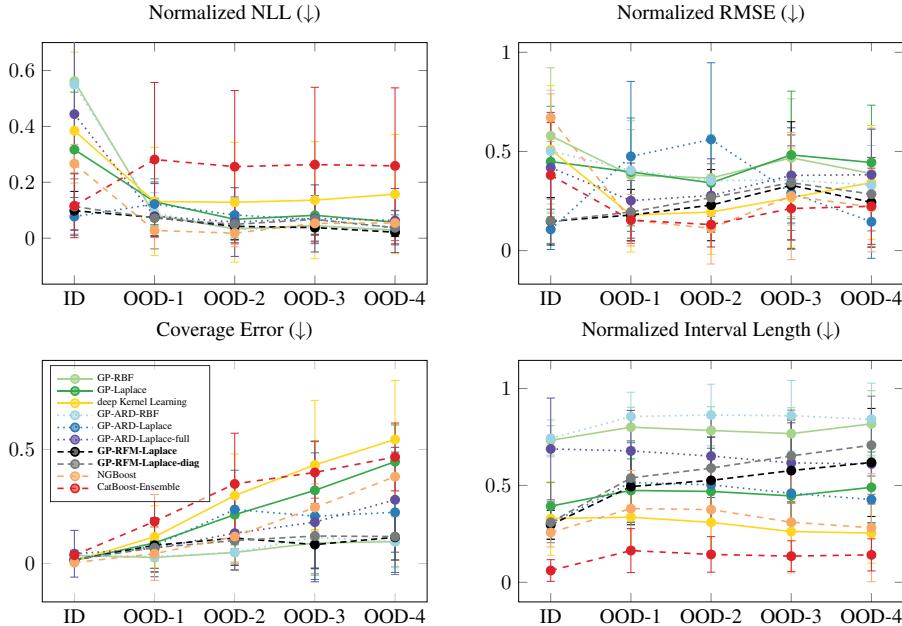


**Figure 11:** NLL (left) and RMSE (right) when removing the most important covariates according to the diagonal of the RFM feature matrix.

consecutive non-overlapping datasets, where each contains 7.5% of the data. The results for the covariate shift over the four house datasets are in Figure 13. The results are qualitatively similar to the results on label shift in Figure 12.



**Figure 12:** Label shift on four house datasets from the OpenML benchmark.



**Figure 13:** Covariate shift (we use the covariate for the latitude of the house position) on four house datasets from the OpenML benchmark.

## C Detailed results on tabular benchmarks

In the main text and Figure 7 we show summary results over all datasets of the two benchmarks we consider. The following tables present the results individually for each dataset in both benchmarks.

### C.1 Tabular Benchmark

**Table 2:** Time (↓) in seconds required for training and prediction on the OpenML benchmark.

Dataset	Gaussian Process						Boosting			
	ARD			Ours			RFM		NGBBoost	
	RBF	Laplace	deep KL	RBF	Lap.	Lap. full	RFM	RFM-diag	CatBoost	
cpu-act	13.23 ±0.76	10.84 ±0.59	2.87 ±0.06	10.59 ±0.30	10.77 ±0.56	8.88 ±0.30	10.92 ±0.36	10.31 ±0.24	23.03 ±3.33	27.37 ±4.10
pol	12.47 ±0.17	17.30 ±0.50	4.49 ±0.05	13.77 ±0.25	17.08 ±0.45	10.56 ±0.68	17.84 ±0.50	17.54 ±0.08	18.92 ±0.17	24.16 ±3.76
elevators	22.38 ±0.53	20.37 ±0.20	5.24 ±0.05	20.79 ±0.19	20.39 ±0.24	12.35 ±0.12	21.73 ±0.32	21.45 ±0.03	23.83 ±0.15	23.50 ±2.59
isolet	6.69 ±0.08	11.42 ±0.73	3.04 ±0.05	7.85 ±0.08	11.86 ±0.49	6.47 ±0.15	12.84 ±0.75	10.79 ±0.16	830.81 ±3.03	258.70 ±17.49
wine	6.80 ±0.15	11.15 ±0.56	2.67 ±0.03	6.86 ±0.17	10.95 ±1.32	5.69 ±0.22	11.23 ±0.60	9.43 ±0.80	9.97 ±1.11	21.83 ±3.37
Ailerons	14.13 ±0.26	15.66 ±1.03	4.22 ±0.06	14.05 ±0.15	15.46 ±0.67	9.45 ±0.41	16.45 ±1.11	14.96 ±0.23	29.33 ±0.17	27.14 ±2.65
houses	15.65 ±0.24	32.46 ±1.29	6.53 ±0.06	15.75 ±0.19	32.30 ±0.65	19.35 ±0.38	34.40 ±1.05	33.79 ±0.23	31.97 ±0.20	27.83 ±2.63
houses-16H	20.87 ±6.45	39.30 ±0.62	7.48 ±0.30	22.29 ±7.01	39.13 ±0.13	23.73 ±0.41	41.80 ±0.98	41.43 ±0.06	72.42 ±5.58	32.03 ±3.35
Bra-houses	17.35 ±6.80	12.31 ±0.57	3.27 ±0.04	9.72 ±0.72	12.48 ±1.06	6.86 ±0.32	12.80 ±0.82	11.57 ±0.55	11.51 ±0.52	23.42 ±2.40
bike	12.42 ±0.08	22.44 ±0.14	5.56 ±0.02	12.80 ±0.07	22.51 ±0.21	13.39 ±0.19	24.37 ±1.67	23.53 ±0.07	12.81 ±0.16	20.11 ±2.73
house-sales	24.78 ±3.06	35.02 ±0.16	6.84 ±0.05	22.93 ±0.40	35.05 ±0.23	21.27 ±0.53	37.50 ±0.52	37.13 ±0.08	39.21 ±0.24	29.78 ±3.38
sulfur	7.69 ±0.07	12.25 ±1.49	3.38 ±0.12	7.84 ±0.09	12.62 ±1.49	6.18 ±0.19	12.99 ±1.21	10.85 ±0.33	11.36 ±4.35	25.19 ±4.00
Miami2016	10.26 ±1.44	15.49 ±0.54	4.37 ±0.05	10.18 ±0.06	15.46 ±0.64	8.97 ±0.33	16.65 ±0.79	15.32 ±0.25	38.62 ±0.23	30.22 ±4.17
superconduct	15.88 ±0.08	34.60 ±0.66	6.78 ±0.04	17.04 ±0.07	34.47 ±0.58	28.85 ±33.30	71.69 ±83.62	37.35 ±0.98	262.93 ±1.43	61.23 ±5.36
california	16.38 ±0.25	32.44 ±0.82	6.49 ±0.13	16.13 ±0.25	32.34 ±0.93	19.49 ±0.11	34.37 ±0.77	33.98 ±0.48	35.96 ±0.26	28.92 ±3.32
fifa	13.99 ±0.62	24.25 ±0.12	5.58 ±0.04	14.59 ±0.15	24.40 ±0.30	14.50 ±0.08	26.26 ±1.38	25.72 ±0.07	14.29 ±0.65	25.10 ±3.56

**Table 3:** OpenML dataset: cpu-act (8192 samples; 21 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	$4.1983 \pm 0.3682$	$2.8004 \pm 0.0687$	$0.0407 \pm 0.0030$	$29.0699 \pm 2.8943$	$13.2314 \pm 0.7587$
GP-Laplace	$3.8907 \pm 0.2875$	$2.5481 \pm 0.0244$	$0.0310 \pm 0.0029$	$15.5996 \pm 0.1433$	$10.8411 \pm 0.5874$
deep Kernel Learning	$2.5007 \pm 0.0820$	$2.6724 \pm 0.0315$	$0.0892 \pm 0.1866$	$20.1186 \pm 0.7163$	$2.8708 \pm 0.0596$
GP-ARD-RBF	$3.3840 \pm 0.2889$	$2.6651 \pm 0.0397$	$0.0430 \pm 0.0019$	$23.5181 \pm 1.1309$	$10.5879 \pm 0.0596$
GP-ARD-Laplace	$2.7398 \pm 0.1460$	$2.2968 \pm 0.0173$	$0.0279 \pm 0.0029$	$11.7734 \pm 0.0615$	$10.7739 \pm 0.5581$
GP-ARD-Laplace-full	$7.1210 \pm 1.0766$	$3.7143 \pm 0.0963$	$0.0376 \pm 0.0037$	$59.3472 \pm 3.7178$	$8.8822 \pm 0.2975$
<b>GP-RFM-Laplace</b>	$2.3291 \pm 0.1149$	$2.2050 \pm 0.0131$	$0.0206 \pm 0.0032$	$10.0737 \pm 0.1053$	$10.9167 \pm 0.3581$
<b>GP-RFM-Laplace-diag</b>	$2.1579 \pm 0.0447$	$2.1661 \pm 0.0091$	$0.0247 \pm 0.0028$	$9.8702 \pm 0.0921$	$10.3116 \pm 0.2368$
NGBoost	$2.4774 \pm 0.0915$	$2.3276 \pm 0.1373$	$0.0077 \pm 0.0057$	$8.5031 \pm 0.7630$	$23.0311 \pm 3.3259$
CatBoost-Ensemble	$2.5011 \pm 0.1260$	$2.1699 \pm 0.0295$	$0.0081 \pm 0.0042$	$8.0882 \pm 0.0825$	$27.3657 \pm 4.0987$

**Table 4:** OpenML dataset: pol (15000 samples; 26 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	$8.1744 \pm 0.2212$	$3.7279 \pm 0.0083$	$0.0402 \pm 0.0012$	$60.3733 \pm 0.5259$	$12.4684 \pm 0.1706$
GP-Laplace	$6.4119 \pm 0.0992$	$3.4294 \pm 0.0064$	$0.0281 \pm 0.0021$	$40.8013 \pm 0.2545$	$17.3032 \pm 0.4992$
deep Kernel Learning	$3.3911 \pm 0.3008$	$3.4039 \pm 0.0125$	$0.0462 \pm 0.0017$	$44.8599 \pm 0.9984$	$4.4946 \pm 0.0514$
GP-ARD-RBF	$3.2847 \pm 0.0940$	$3.0713 \pm 0.0065$	$0.0462 \pm 0.0006$	$33.9202 \pm 0.3687$	$13.7665 \pm 0.0514$
GP-ARD-Laplace	$3.5131 \pm 0.1060$	$2.8353 \pm 0.0070$	$0.0330 \pm 0.0021$	$22.9351 \pm 0.1555$	$17.0843 \pm 0.4549$
GP-ARD-Laplace-full	$6.3502 \pm 0.2260$	$4.4107 \pm 0.0210$	$0.2220 \pm 0.3442$	$127.1110 \pm 1.5457$	$10.5559 \pm 0.6841$
<b>GP-RFM-Laplace</b>	$3.3721 \pm 0.1531$	$2.7280 \pm 0.0144$	$0.0301 \pm 0.0024$	$19.6895 \pm 0.1998$	$17.8368 \pm 0.5048$
<b>GP-RFM-Laplace-diag</b>	$4.7450 \pm 0.1514$	$3.1021 \pm 0.0125$	$0.0269 \pm 0.0021$	$29.3767 \pm 0.2548$	$17.5354 \pm 0.0763$
NGBoost	$12.0208 \pm 0.2321$	$3.5523 \pm 0.0134$	$0.0150 \pm 0.0023$	$42.2595 \pm 0.4839$	$18.9153 \pm 0.1735$
CatBoost-Ensemble	$4.9089 \pm 0.1442$	$2.0865 \pm 0.0324$	$0.0136 \pm 0.0041$	$10.6416 \pm 0.1790$	$24.1609 \pm 3.7644$

**Table 5:** OpenML dataset: elevators (16599 samples; 16 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	0.0022 $\pm$ 0.0000	-4.4642 $\pm$ 0.0313	0.0400 $\pm$ 0.0017	0.0166 $\pm$ 0.0008	22.3781 $\pm$ 0.5291
GP-Laplace	0.0022 $\pm$ 0.0000	-4.6729 $\pm$ 0.0090	0.0259 $\pm$ 0.0023	0.0105 $\pm$ 0.0001	20.3693 $\pm$ 0.1979
deep Kernel Learning	0.0019 $\pm$ 0.0000	-4.8461 $\pm$ 0.0123	0.0109 $\pm$ 0.0031	0.0081 $\pm$ 0.0002	5.2406 $\pm$ 0.0463
GP-ARD-RBF	0.0022 $\pm$ 0.0001	-4.5271 $\pm$ 0.0177	0.0404 $\pm$ 0.0014	0.0151 $\pm$ 0.0004	20.7872 $\pm$ 0.0463
GP-ARD-Laplace	0.0021 $\pm$ 0.0000	-4.7496 $\pm$ 0.0092	0.0244 $\pm$ 0.0030	0.0097 $\pm$ 0.0001	20.3881 $\pm$ 0.2434
GP-ARD-Laplace-full	0.0026 $\pm$ 0.0002	-4.3103 $\pm$ 0.1588	0.0397 $\pm$ 0.0044	0.0181 $\pm$ 0.0033	12.3529 $\pm$ 0.1221
<b>GP-RFM-Laplace</b>	0.0019 $\pm$ 0.0000	-4.8622 $\pm$ 0.0085	0.0154 $\pm$ 0.0025	0.0081 $\pm$ 0.0001	21.7328 $\pm$ 0.3192
<b>GP-RFM-Laplace-diag</b>	0.0020 $\pm$ 0.0000	-4.7850 $\pm$ 0.0086	0.0205 $\pm$ 0.0024	0.0092 $\pm$ 0.0001	21.4464 $\pm$ 0.0321
NGBoost	0.0036 $\pm$ 0.0001	-4.4798 $\pm$ 0.0151	0.0047 $\pm$ 0.0030	0.0115 $\pm$ 0.0001	23.8299 $\pm$ 0.1544
CatBoost-Ensemble	0.0023 $\pm$ 0.0000	-4.7321 $\pm$ 0.0183	0.0451 $\pm$ 0.0042	0.0068 $\pm$ 0.0001	23.4951 $\pm$ 2.5942

**Table 6:** OpenML dataset: isolet (7797 samples; 613 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	7.5039 $\pm$ 0.0491	3.4345 $\pm$ 0.0065	0.0500 $\pm$ 0.0000	29.4934 $\pm$ 0.1699	6.6856 $\pm$ 0.0760
GP-Laplace	7.5039 $\pm$ 0.0491	3.4344 $\pm$ 0.0065	0.0500 $\pm$ 0.0000	29.5290 $\pm$ 0.0978	11.4205 $\pm$ 0.7336
deep Kernel Learning	3.1464 $\pm$ 0.2413	2.6187 $\pm$ 0.0857	0.0227 $\pm$ 0.0090	11.6625 $\pm$ 3.1630	3.0425 $\pm$ 0.0533
GP-ARD-RBF	7.5039 $\pm$ 0.0491	3.4345 $\pm$ 0.0065	0.0500 $\pm$ 0.0000	29.4947 $\pm$ 0.1697	7.8513 $\pm$ 0.0533
GP-ARD-Laplace	7.5039 $\pm$ 0.0490	3.4345 $\pm$ 0.0065	0.0500 $\pm$ 0.0000	29.5526 $\pm$ 0.1406	11.8644 $\pm$ 0.4897
GP-ARD-Laplace-full	7.5039 $\pm$ 0.0490	3.4345 $\pm$ 0.0065	0.0500 $\pm$ 0.0000	29.5522 $\pm$ 0.0904	6.4689 $\pm$ 0.1459
<b>GP-RFM-Laplace</b>	2.5696 $\pm$ 0.0979	2.3408 $\pm$ 0.0373	0.0057 $\pm$ 0.0043	9.9181 $\pm$ 0.3743	12.8427 $\pm$ 0.7516
<b>GP-RFM-Laplace-diag</b>	3.1757 $\pm$ 0.0768	2.5661 $\pm$ 0.0167	0.0102 $\pm$ 0.0042	14.2781 $\pm$ 0.2057	10.7894 $\pm$ 0.1638
NGBoost	4.1270 $\pm$ 0.0689	2.7053 $\pm$ 0.0157	0.0041 $\pm$ 0.0028	14.5239 $\pm$ 0.1299	830.8086 $\pm$ 3.0250
CatBoost-Ensemble	3.4882 $\pm$ 0.0693	2.5194 $\pm$ 0.0215	0.0395 $\pm$ 0.0072	10.1068 $\pm$ 0.1052	258.7036 $\pm$ 17.4887

**Table 7:** OpenML dataset: wine-quality (6497 samples; 11 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	0.6684 $\pm$ 0.0190	1.0380 $\pm$ 0.0237	0.0152 $\pm$ 0.0060	3.1303 $\pm$ 0.1727	6.8032 $\pm$ 0.1487
GP-Laplace	0.6086 $\pm$ 0.0172	0.9493 $\pm$ 0.0152	0.0176 $\pm$ 0.0051	2.9316 $\pm$ 0.0346	11.1491 $\pm$ 0.5616
deep Kernel Learning	0.6958 $\pm$ 0.0129	1.0632 $\pm$ 0.0235	0.0178 $\pm$ 0.0139	2.5838 $\pm$ 0.1521	2.6719 $\pm$ 0.0273
GP-ARD-RBF	0.6699 $\pm$ 0.0208	1.0434 $\pm$ 0.0258	0.0172 $\pm$ 0.0065	3.1711 $\pm$ 0.1556	6.8614 $\pm$ 0.0273
GP-ARD-Laplace	0.6114 $\pm$ 0.0175	0.9496 $\pm$ 0.0155	0.0165 $\pm$ 0.0063	2.8888 $\pm$ 0.0613	10.9450 $\pm$ 1.3206
GP-ARD-Laplace-full	0.6129 $\pm$ 0.0166	0.9488 $\pm$ 0.0166	0.0136 $\pm$ 0.0061	2.8646 $\pm$ 0.0338	5.6883 $\pm$ 0.2169
<b>GP-RFM-Laplace</b>	0.6105 $\pm$ 0.0170	0.9494 $\pm$ 0.0151	0.0168 $\pm$ 0.0053	2.8991 $\pm$ 0.0597	11.2258 $\pm$ 0.6049
<b>GP-RFM-Laplace-diag</b>	0.6132 $\pm$ 0.0161	0.9523 $\pm$ 0.0154	0.0170 $\pm$ 0.0058	2.8860 $\pm$ 0.0737	9.4255 $\pm$ 0.8003
NGBoost	0.6981 $\pm$ 0.0153	1.0351 $\pm$ 0.0256	0.0135 $\pm$ 0.0082	2.5237 $\pm$ 0.0325	9.9671 $\pm$ 1.1130
CatBoost-Ensemble	0.6910 $\pm$ 0.0159	1.0276 $\pm$ 0.0273	0.0201 $\pm$ 0.0124	2.4433 $\pm$ 0.1155	21.8260 $\pm$ 3.3698

**Table 8:** OpenML dataset: Ailerons (13750 samples; 33 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	$0.0002 \pm 0.0000$	$-7.1758 \pm 0.0275$	$0.0341 \pm 0.0028$	$0.0010 \pm 0.0000$	$14.1305 \pm 0.2561$
GP-Laplace	$0.0002 \pm 0.0000$	$-7.3105 \pm 0.0092$	$0.0122 \pm 0.0029$	$0.0007 \pm 0.0000$	$15.6566 \pm 1.0279$
deep Kernel Learning	$0.0002 \pm 0.0000$	$-7.3094 \pm 0.0166$	$0.0181 \pm 0.0132$	$0.0006 \pm 0.0000$	$4.2194 \pm 0.0552$
GP-ARD-RBF	$0.0002 \pm 0.0000$	$-7.1859 \pm 0.0146$	$0.0364 \pm 0.0021$	$0.0010 \pm 0.0000$	$14.0530 \pm 0.0552$
GP-ARD-Laplace	$0.0002 \pm 0.0000$	$-7.3346 \pm 0.0094$	$0.0108 \pm 0.0031$	$0.0007 \pm 0.0000$	$15.4564 \pm 0.6676$
GP-ARD-Laplace-full	$0.0002 \pm 0.0000$	$-6.7233 \pm 0.0045$	$0.0382 \pm 0.0016$	$0.0016 \pm 0.0000$	$9.4539 \pm 0.4092$
<b>GP-RFM-Laplace</b>	$0.0002 \pm 0.0000$	$-7.3728 \pm 0.0094$	$0.0061 \pm 0.0031$	$0.0007 \pm 0.0000$	$16.4483 \pm 1.1083$
<b>GP-RFM-Laplace-diag</b>	$0.0002 \pm 0.0000$	$-7.3641 \pm 0.0091$	$0.0073 \pm 0.0032$	$0.0007 \pm 0.0000$	$14.9629 \pm 0.2323$
NGBoost	$0.0002 \pm 0.0000$	$-7.4229 \pm 0.0113$	$0.0042 \pm 0.0025$	$0.0006 \pm 0.0000$	$29.3339 \pm 0.1665$
CatBoost-Ensemble	$0.0002 \pm 0.0000$	$-7.4136 \pm 0.0131$	$0.0071 \pm 0.0037$	$0.0006 \pm 0.0000$	$27.1441 \pm 2.6503$

**Table 9:** OpenML dataset: houses (20640 samples; 8 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	$0.2555 \pm 0.0037$	$0.1620 \pm 0.0054$	$0.0346 \pm 0.0010$	$1.4874 \pm 0.0096$	$15.6545 \pm 0.2366$
GP-Laplace	$0.2528 \pm 0.0038$	$0.0727 \pm 0.0067$	$0.0216 \pm 0.0020$	$1.2362 \pm 0.0080$	$32.4463 \pm 1.2917$
deep Kernel Learning	$0.2647 \pm 0.0057$	$0.0919 \pm 0.0216$	$0.0061 \pm 0.0088$	$1.0388 \pm 0.0241$	$6.5323 \pm 0.0603$
GP-ARD-RBF	$0.2425 \pm 0.0038$	$0.1618 \pm 0.0059$	$0.0388 \pm 0.0012$	$1.5535 \pm 0.0107$	$15.7538 \pm 0.0603$
GP-ARD-Laplace	$0.2087 \pm 0.0035$	$-0.0989 \pm 0.0064$	$0.0290 \pm 0.0020$	$1.1205 \pm 0.0086$	$32.3049 \pm 0.6497$
GP-ARD-Laplace-full	$0.2530 \pm 0.0034$	$0.1745 \pm 0.0049$	$0.0351 \pm 0.0020$	$1.5020 \pm 0.0041$	$19.3467 \pm 0.3817$
<b>GP-RFM-Laplace</b>	$0.2190 \pm 0.0032$	$-0.0740 \pm 0.0066$	$0.0232 \pm 0.0020$	$1.0871 \pm 0.0087$	$34.3996 \pm 1.0498$
<b>GP-RFM-Laplace-diag</b>	$0.2245 \pm 0.0033$	$-0.0377 \pm 0.0055$	$0.0257 \pm 0.0020$	$1.1525 \pm 0.0099$	$33.7937 \pm 0.2298$
NGBoost	$0.2826 \pm 0.0036$	$0.0747 \pm 0.0127$	$0.0041 \pm 0.0024$	$1.0640 \pm 0.0051$	$31.9747 \pm 0.2022$
CatBoost-Ensemble	$0.2345 \pm 0.0028$	$-0.1249 \pm 0.0191$	$0.0387 \pm 0.0032$	$0.6976 \pm 0.0049$	$27.8312 \pm 2.6283$

**Table 10:** OpenML dataset: house-16H (22784 samples; 16 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	$0.6324 \pm 0.0281$	$0.8378 \pm 0.0294$	$0.0406 \pm 0.0016$	$2.8425 \pm 0.2764$	$20.8717 \pm 6.4480$
GP-Laplace	$0.6096 \pm 0.0274$	$0.7220 \pm 0.0241$	$0.0325 \pm 0.0022$	$2.2368 \pm 0.0525$	$39.2961 \pm 0.6178$
deep Kernel Learning	$0.6585 \pm 0.0551$	$0.9472 \pm 0.0505$	$0.0281 \pm 0.0137$	$2.4594 \pm 0.2951$	$7.4795 \pm 0.3042$
GP-ARD-RBF	$0.6352 \pm 0.0253$	$0.8382 \pm 0.0234$	$0.0408 \pm 0.0017$	$2.8555 \pm 0.2688$	$22.2921 \pm 0.3042$
GP-ARD-Laplace	$0.6077 \pm 0.0269$	$0.7171 \pm 0.0242$	$0.0328 \pm 0.0021$	$2.2158 \pm 0.0520$	$39.1299 \pm 0.1318$
GP-ARD-Laplace-full	$0.6168 \pm 0.0257$	$0.8972 \pm 0.0261$	$0.1304 \pm 0.2719$	$2.7582 \pm 0.0506$	$23.7313 \pm 0.4123$
<b>GP-RFM-Laplace</b>	$0.6063 \pm 0.0260$	$0.6899 \pm 0.0241$	$0.0318 \pm 0.0025$	$2.1696 \pm 0.0602$	$41.8046 \pm 0.9823$
<b>GP-RFM-Laplace-diag</b>	$0.6179 \pm 0.0283$	$0.7098 \pm 0.0315$	$0.0338 \pm 0.0024$	$2.2318 \pm 0.0699$	$41.4268 \pm 0.0609$
NGBoost	$0.6031 \pm 0.0280$	$0.5686 \pm 0.0467$	$0.0078 \pm 0.0029$	$1.5809 \pm 0.0292$	$72.4206 \pm 5.3819$
CatBoost-Ensemble	$0.5956 \pm 0.0308$	$0.5125 \pm 0.0586$	$0.0041 \pm 0.0035$	$1.4476 \pm 0.0891$	$32.0270 \pm 3.3470$

**Table 11:** OpenML dataset: Brazilian-houses (10692 samples; 8 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	$0.1049 \pm 0.0356$	$-0.9903 \pm 0.6726$	$0.1723 \pm 0.2795$	$0.6436 \pm 0.4977$	$17.3515 \pm 6.7992$
GP-Laplace	$0.0622 \pm 0.0281$	$-1.4862 \pm 0.0729$	$0.0451 \pm 0.0016$	$0.3345 \pm 0.0393$	$12.3126 \pm 0.5714$
deep Kernel Learning	$0.0493 \pm 0.0228$	$-0.6394 \pm 0.0444$	$0.0841 \pm 0.1507$	$0.8156 \pm 0.0364$	$3.2681 \pm 0.0353$
GP-ARD-RBF	$0.0646 \pm 0.0318$	$-2.1872 \pm 0.0338$	$0.0459 \pm 0.0017$	$0.1688 \pm 0.0076$	$9.7154 \pm 0.0353$
GP-ARD-Laplace	$0.0537 \pm 0.0284$	$-1.8204 \pm 0.1340$	$0.0473 \pm 0.0011$	$0.2136 \pm 0.0055$	$12.4825 \pm 1.0629$
GP-ARD-Laplace-full	$0.0981 \pm 0.0216$	$0.2656 \pm 0.0432$	$0.0493 \pm 0.0004$	$2.0113 \pm 0.0849$	$6.8555 \pm 0.3227$
<b>GP-RFM-Laplace</b>	$0.0414 \pm 0.0181$	$-2.1078 \pm 0.0600$	$0.0488 \pm 0.0005$	$0.1406 \pm 0.0049$	$12.8021 \pm 0.8225$
<b>GP-RFM-Laplace-diag</b>	$0.0404 \pm 0.0184$	$-2.0733 \pm 0.0696$	$0.0486 \pm 0.0007$	$0.1566 \pm 0.0009$	$11.5711 \pm 0.5537$
NGBoost	$0.0529 \pm 0.0243$	$-2.1812 \pm 0.1506$	$0.0194 \pm 0.0049$	$0.1061 \pm 0.0022$	$11.5133 \pm 0.5235$
CatBoost-Ensemble	$0.0541 \pm 0.0318$	$-2.6618 \pm 0.2337$	$0.0398 \pm 0.0042$	$0.0833 \pm 0.0126$	$23.4218 \pm 2.4005$

**Table 12:** OpenML dataset: Bike-Sharing-Demand (17379 samples; 6 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	$110.0164 \pm 1.5813$	$6.1724 \pm 0.0069$	$0.0135 \pm 0.0026$	$544.7001 \pm 3.7165$	$12.4240 \pm 0.0800$
GP-Laplace	$108.5130 \pm 1.7637$	$6.1472 \pm 0.0096$	$0.0095 \pm 0.0037$	$520.2480 \pm 16.0873$	$22.4437 \pm 0.1403$
deep Kernel Learning	$103.8203 \pm 2.7178$	$6.0799 \pm 0.0479$	$0.0185 \pm 0.0093$	$445.8881 \pm 54.9197$	$5.5567 \pm 0.0227$
GP-ARD-RBF	$99.5496 \pm 1.1839$	$6.0465 \pm 0.0084$	$0.0089 \pm 0.0042$	$446.8315 \pm 9.7015$	$12.8033 \pm 0.0227$
GP-ARD-Laplace	$100.2501 \pm 1.1656$	$6.0335 \pm 0.0096$	$0.0176 \pm 0.0031$	$417.9397 \pm 2.5062$	$22.5115 \pm 0.2084$
GP-ARD-Laplace-full	$102.6751 \pm 1.4195$	$6.0678 \pm 0.0126$	$0.0047 \pm 0.0045$	$455.0795 \pm 11.7388$	$13.3893 \pm 0.1907$
<b>GP-RFM-Laplace</b>	$100.4792 \pm 1.2527$	$6.0351 \pm 0.0102$	$0.0192 \pm 0.0038$	$415.9705 \pm 4.3815$	$24.3658 \pm 1.6734$
<b>GP-RFM-Laplace-diag</b>	$100.4778 \pm 1.1564$	$6.0343 \pm 0.0097$	$0.0199 \pm 0.0041$	$414.1059 \pm 4.0362$	$23.5265 \pm 0.0726$
NGBoost	$104.1888 \pm 1.2904$	$5.6200 \pm 0.0110$	$0.0114 \pm 0.0028$	$337.1860 \pm 2.0218$	$12.8066 \pm 0.1632$
CatBoost-Ensemble	$100.3143 \pm 1.1865$	$5.5759 \pm 0.0120$	$0.0025 \pm 0.0022$	$310.3412 \pm 2.2997$	$20.1121 \pm 2.7261$

**Table 13:** OpenML dataset: house-sales (21613 samples; 15 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	$0.2215 \pm 0.0051$	$0.0370 \pm 0.0182$	$0.0401 \pm 0.0019$	$1.3997 \pm 0.0306$	$24.7769 \pm 3.0601$
GP-Laplace	$0.2034 \pm 0.0027$	$-0.1850 \pm 0.0078$	$0.0115 \pm 0.0027$	$0.9016 \pm 0.0043$	$35.0244 \pm 0.1595$
deep Kernel Learning	$0.1944 \pm 0.0034$	$-0.2193 \pm 0.0168$	$0.0056 \pm 0.0044$	$0.7751 \pm 0.0265$	$6.8426 \pm 0.0507$
GP-ARD-RBF	$0.2028 \pm 0.0037$	$-0.0021 \pm 0.0137$	$0.0416 \pm 0.0008$	$1.3650 \pm 0.0191$	$22.9284 \pm 0.0507$
GP-ARD-Laplace	$0.1808 \pm 0.0020$	$-0.3033 \pm 0.0060$	$0.0127 \pm 0.0020$	$0.8016 \pm 0.0031$	$35.0499 \pm 0.2341$
GP-ARD-Laplace-full	$0.1995 \pm 0.0068$	$-0.0552 \pm 0.1645$	$0.1210 \pm 0.2763$	$1.2464 \pm 0.3343$	$21.2724 \pm 0.5262$
<b>GP-RFM-Laplace</b>	$0.1755 \pm 0.0016$	$-0.3151 \pm 0.0065$	$0.0177 \pm 0.0019$	$0.8292 \pm 0.0085$	$37.4970 \pm 0.5236$
<b>GP-RFM-Laplace-diag</b>	$0.1731 \pm 0.0023$	$-0.3198 \pm 0.0100$	$0.0201 \pm 0.0026$	$0.8333 \pm 0.0174$	$37.1304 \pm 0.0767$
NGBoost	$0.2029 \pm 0.0020$	$-0.2679 \pm 0.0083$	$0.0026 \pm 0.0015$	$0.7642 \pm 0.0033$	$39.2063 \pm 0.2366$
CatBoost-Ensemble	$0.1963 \pm 0.0021$	$-0.3144 \pm 0.0085$	$0.0136 \pm 0.0090$	$0.6892 \pm 0.0028$	$29.7815 \pm 3.3777$

**Table 14:** OpenML dataset: sulfur (10081 samples; 6 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	$0.0183 \pm 0.0027$	$-2.4195 \pm 0.0201$	$0.0466 \pm 0.0012$	$0.1279 \pm 0.0043$	$7.6919 \pm 0.0673$
GP-Laplace	$0.0159 \pm 0.0031$	$-2.8279 \pm 0.0344$	$0.0395 \pm 0.0019$	$0.0770 \pm 0.0027$	$12.2545 \pm 1.4937$
deep Kernel Learning	$0.0259 \pm 0.0038$	$-2.4438 \pm 0.2366$	$0.0284 \pm 0.0112$	$0.0809 \pm 0.0091$	$3.3750 \pm 0.1205$
GP-ARD-RBF	$0.0182 \pm 0.0028$	$-2.4195 \pm 0.0195$	$0.0467 \pm 0.0012$	$0.1280 \pm 0.0034$	$7.8406 \pm 0.1205$
GP-ARD-Laplace	$0.0169 \pm 0.0041$	$-2.8275 \pm 0.0379$	$0.0394 \pm 0.0015$	$0.0775 \pm 0.0017$	$12.6240 \pm 1.4911$
GP-ARD-Laplace-full	$0.0182 \pm 0.0047$	$-2.6309 \pm 0.1468$	$0.0418 \pm 0.0016$	$0.0952 \pm 0.0117$	$6.1771 \pm 0.1899$
<b>GP-RFM-Laplace</b>	$0.0171 \pm 0.0044$	$-2.8000 \pm 0.0486$	$0.0395 \pm 0.0022$	$0.0781 \pm 0.0028$	$12.9883 \pm 1.2052$
<b>GP-RFM-Laplace-diag</b>	$0.0181 \pm 0.0043$	$-2.7366 \pm 0.0560$	$0.0407 \pm 0.0018$	$0.0826 \pm 0.0032$	$10.8533 \pm 0.3308$
NGBoost	$0.0256 \pm 0.0042$	$-2.5867 \pm 0.4146$	$0.0145 \pm 0.0141$	$0.0606 \pm 0.0053$	$11.3634 \pm 4.3508$
CatBoost-Ensemble	$0.0244 \pm 0.0046$	$-2.8813 \pm 0.0822$	$0.0141 \pm 0.0162$	$0.0484 \pm 0.0050$	$25.1911 \pm 4.0039$

**Table 15:** OpenML dataset: MiamiHousing2016 (13932 samples; 13 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	$0.1803 \pm 0.0047$	$0.0057 \pm 0.0041$	$0.0439 \pm 0.0011$	$1.4023 \pm 0.0078$	$10.2637 \pm 1.4351$
GP-Laplace	$0.1655 \pm 0.0041$	$-0.3567 \pm 0.0117$	$0.0168 \pm 0.0036$	$0.8053 \pm 0.0073$	$15.4859 \pm 0.5375$
deep Kernel Learning	$0.1691 \pm 0.0048$	$-0.3507 \pm 0.0236$	$0.0056 \pm 0.0043$	$0.7240 \pm 0.0214$	$4.3708 \pm 0.0464$
GP-ARD-RBF	$0.1785 \pm 0.0045$	$0.0044 \pm 0.0036$	$0.0441 \pm 0.0010$	$1.4056 \pm 0.0073$	$10.1812 \pm 0.0464$
GP-ARD-Laplace	$0.1486 \pm 0.0032$	$-0.4600 \pm 0.0089$	$0.0216 \pm 0.0024$	$0.7449 \pm 0.0049$	$15.4611 \pm 0.6357$
GP-ARD-Laplace-full	$0.1670 \pm 0.0081$	$-0.1694 \pm 0.1773$	$0.0348 \pm 0.0094$	$1.1301 \pm 0.2839$	$8.9669 \pm 0.3332$
<b>GP-RFM-Laplace</b>	$0.1485 \pm 0.0027$	$-0.4744 \pm 0.0087$	$0.0178 \pm 0.0028$	$0.7233 \pm 0.0063$	$16.6506 \pm 0.7884$
<b>GP-RFM-Laplace-diag</b>	$0.1451 \pm 0.0027$	$-0.5000 \pm 0.0085$	$0.0188 \pm 0.0025$	$0.7064 \pm 0.0063$	$15.3197 \pm 0.2462$
NGBoost	$0.1997 \pm 0.0038$	$-0.3802 \pm 0.0143$	$0.0055 \pm 0.0037$	$0.7269 \pm 0.0061$	$38.6213 \pm 0.2251$
CatBoost-Ensemble	$0.1835 \pm 0.0039$	$-0.5299 \pm 0.0164$	$0.0141 \pm 0.0190$	$0.5927 \pm 0.0063$	$30.2160 \pm 4.1676$

**Table 16:** OpenML dataset: superconduct (21263 samples; 79 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	$11.9302 \pm 0.2640$	$4.2074 \pm 0.0045$	$0.0363 \pm 0.0014$	$91.1963 \pm 0.5064$	$15.8843 \pm 0.0842$
GP-Laplace	$9.4970 \pm 0.2163$	$4.0099 \pm 0.0127$	$0.0281 \pm 0.0017$	$72.7501 \pm 0.7574$	$34.5985 \pm 0.6591$
deep Kernel Learning	$14.5717 \pm 0.4363$	$4.0989 \pm 0.0298$	$0.0097 \pm 0.0029$	$58.0927 \pm 1.2015$	$6.7805 \pm 0.0431$
GP-ARD-RBF	$11.8767 \pm 0.2608$	$4.2020 \pm 0.0046$	$0.0364 \pm 0.0017$	$90.8332 \pm 0.5023$	$17.0431 \pm 0.0431$
GP-ARD-Laplace	$9.5881 \pm 0.2008$	$3.9596 \pm 0.0106$	$0.0291 \pm 0.0020$	$68.9498 \pm 0.6899$	$34.4697 \pm 0.5762$
GP-ARD-Laplace-full	$11.2755 \pm 0.8971$	$4.3354 \pm 0.1772$	$0.0887 \pm 0.1950$	$111.7846 \pm 21.7467$	$28.8458 \pm 33.2999$
<b>GP-RFM-Laplace</b>	$13.2086 \pm 7.0422$	$4.0457 \pm 0.1059$	$0.0409 \pm 0.0362$	$93.3640 \pm 34.0538$	$71.6949 \pm 83.6162$
<b>GP-RFM-Laplace-diag</b>	$10.3174 \pm 0.2215$	$4.0254 \pm 0.0094$	$0.0294 \pm 0.0022$	$75.3887 \pm 1.3068$	$37.3528 \pm 0.9843$
NGBoost	$13.2014 \pm 0.1743$	$3.6477 \pm 0.0177$	$0.0064 \pm 0.0028$	$43.5723 \pm 0.3922$	$262.9271 \pm 1.4310$
CatBoost-Ensemble	$10.9449 \pm 1.0811$	$3.4594 \pm 0.1136$	$0.0236 \pm 0.0174$	$29.7679 \pm 6.6060$	$61.2334 \pm 5.3609$

**Table 17:** OpenML dataset: california (20640 samples; 8 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	0.1614 $\pm$ 0.0025	-0.3129 $\pm$ 0.0082	0.0325 $\pm$ 0.0013	0.9172 $\pm$ 0.0064	16.3839 $\pm$ 0.2506
GP-Laplace	0.1591 $\pm$ 0.0026	-0.3960 $\pm$ 0.0148	0.0201 $\pm$ 0.0017	0.7693 $\pm$ 0.0112	32.4422 $\pm$ 0.8164
deep Kernel Learning	0.1645 $\pm$ 0.0039	-0.3389 $\pm$ 0.0859	0.0144 $\pm$ 0.0147	0.7538 $\pm$ 0.1859	6.4947 $\pm$ 0.1318
GP-ARD-RBF	0.1478 $\pm$ 0.0016	-0.3194 $\pm$ 0.0052	0.0389 $\pm$ 0.0011	0.9708 $\pm$ 0.0049	16.1321 $\pm$ 0.1318
GP-ARD-Laplace	0.1233 $\pm$ 0.0020	-0.6350 $\pm$ 0.0067	0.0246 $\pm$ 0.0022	0.6407 $\pm$ 0.0045	32.3389 $\pm$ 0.9311
GP-ARD-Laplace-full	0.1528 $\pm$ 0.0024	-0.4045 $\pm$ 0.0317	0.0242 $\pm$ 0.0063	0.7753 $\pm$ 0.0627	19.4896 $\pm$ 0.1069
<b>GP-RFM-Laplace</b>	0.1297 $\pm$ 0.0018	-0.5954 $\pm$ 0.0056	0.0224 $\pm$ 0.0021	0.6503 $\pm$ 0.0048	34.3716 $\pm$ 0.7675
<b>GP-RFM-Laplace-diag</b>	0.1261 $\pm$ 0.0017	-0.6142 $\pm$ 0.0052	0.0242 $\pm$ 0.0016	0.6513 $\pm$ 0.0047	33.9779 $\pm$ 0.4784
NGBoost	0.1674 $\pm$ 0.0019	-0.4479 $\pm$ 0.0110	0.0027 $\pm$ 0.0015	0.6248 $\pm$ 0.0044	35.9571 $\pm$ 0.2644
CatBoost-Ensemble	0.1443 $\pm$ 0.0017	-0.5852 $\pm$ 0.0235	0.0396 $\pm$ 0.0040	0.4275 $\pm$ 0.0043	28.9227 $\pm$ 3.3222

**Table 18:** OpenML dataset: fifa (18063 samples; 5 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	0.8447 $\pm$ 0.0112	1.2718 $\pm$ 0.0083	0.0194 $\pm$ 0.0032	4.1114 $\pm$ 0.0515	13.9853 $\pm$ 0.6240
GP-Laplace	0.8367 $\pm$ 0.0106	1.2434 $\pm$ 0.0092	0.0132 $\pm$ 0.0023	3.6669 $\pm$ 0.0112	24.2460 $\pm$ 0.1180
deep Kernel Learning	0.7928 $\pm$ 0.0096	1.1871 $\pm$ 0.0126	0.0059 $\pm$ 0.0029	3.0820 $\pm$ 0.0194	5.5823 $\pm$ 0.0410
GP-ARD-RBF	0.8226 $\pm$ 0.0091	1.2295 $\pm$ 0.0068	0.0108 $\pm$ 0.0027	3.8480 $\pm$ 0.0312	14.5917 $\pm$ 0.0410
GP-ARD-Laplace	0.8203 $\pm$ 0.0102	1.2215 $\pm$ 0.0091	0.0131 $\pm$ 0.0043	3.7242 $\pm$ 0.1273	24.4010 $\pm$ 0.2973
GP-ARD-Laplace-full	0.8060 $\pm$ 0.0097	1.1942 $\pm$ 0.0095	0.0026 $\pm$ 0.0022	3.3354 $\pm$ 0.0318	14.5039 $\pm$ 0.0771
<b>GP-RFM-Laplace</b>	0.8093 $\pm$ 0.0135	1.2056 $\pm$ 0.0194	0.0068 $\pm$ 0.0049	3.5477 $\pm$ 0.2081	26.2624 $\pm$ 1.3755
<b>GP-RFM-Laplace-diag</b>	0.7982 $\pm$ 0.0101	1.1864 $\pm$ 0.0092	0.0043 $\pm$ 0.0042	3.3570 $\pm$ 0.0129	25.7180 $\pm$ 0.0734
NGBoost	0.7746 $\pm$ 0.0096	1.0939 $\pm$ 0.0116	0.0136 $\pm$ 0.0030	2.8662 $\pm$ 0.0161	14.2918 $\pm$ 0.6475
CatBoost-Ensemble	0.7768 $\pm$ 0.0101	1.0964 $\pm$ 0.0178	0.0147 $\pm$ 0.0033	2.8305 $\pm$ 0.0502	25.0984 $\pm$ 3.5605

## C.2 UCI benchmark

**Table 19:** UCI dataset: Concrete Compression Strength (1030 samples; 8 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	5.1649 $\pm$ 0.8539	3.3152 $\pm$ 0.0541	0.0417 $\pm$ 0.0098	39.9684 $\pm$ 2.1881	5.6247 $\pm$ 0.1457
GP-Laplace	4.9342 $\pm$ 0.6294	3.1955 $\pm$ 0.1154	0.0350 $\pm$ 0.0104	30.6059 $\pm$ 0.8027	4.2635 $\pm$ 0.3242
deep Kernel Learning	5.9734 $\pm$ 0.6219	3.2103 $\pm$ 0.1078	0.0257 $\pm$ 0.0266	23.3185 $\pm$ 1.2344	1.5215 $\pm$ 0.0246
GP-ARD-RBF	5.0771 $\pm$ 0.8361	3.2424 $\pm$ 0.0776	0.0413 $\pm$ 0.0097	36.5128 $\pm$ 2.0837	4.4744 $\pm$ 0.0246
GP-ARD-Laplace	4.8375 $\pm$ 0.7156	3.0636 $\pm$ 0.1369	0.0261 $\pm$ 0.0156	25.1250 $\pm$ 0.6902	4.2398 $\pm$ 0.4143
GP-ARD-Laplace-full	5.8577 $\pm$ 0.5434	3.5410 $\pm$ 0.0146	0.0471 $\pm$ 0.0054	48.5448 $\pm$ 0.3850	4.8013 $\pm$ 0.6353
<b>GP-RFM-Laplace</b>	4.9390 $\pm$ 0.6834	3.0209 $\pm$ 0.0878	0.0243 $\pm$ 0.0157	24.1509 $\pm$ 0.8528	4.4505 $\pm$ 0.4256
<b>GP-RFM-Laplace-diag</b>	5.3889 $\pm$ 0.8233	3.0977 $\pm$ 0.1039	0.0199 $\pm$ 0.0133	25.2850 $\pm$ 0.6942	4.3315 $\pm$ 0.1873
NGBoost	5.6672 $\pm$ 0.6433	3.0846 $\pm$ 0.1400	0.0298 $\pm$ 0.0280	18.9717 $\pm$ 1.2642	3.1179 $\pm$ 0.3985
CatBoost-Ensemble	5.3957 $\pm$ 0.5575	3.0866 $\pm$ 0.1788	0.0414 $\pm$ 0.0344	17.0637 $\pm$ 1.8528	13.1103 $\pm$ 0.9603

**Table 20:** UCI dataset: Energy Efficiency (768 samples; 8 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	0.5372 $\pm$ 0.0881	0.8400 $\pm$ 0.0745	0.0338 $\pm$ 0.0142	2.8408 $\pm$ 0.0700	0.8882 $\pm$ 0.0090
GP-Laplace	1.6181 $\pm$ 0.1648	1.9621 $\pm$ 0.0585	0.0279 $\pm$ 0.0160	8.3290 $\pm$ 0.0656	1.3786 $\pm$ 0.0936
deep Kernel Learning	0.6141 $\pm$ 0.1244	1.9590 $\pm$ 0.0368	0.0500 $\pm$ 0.0000	10.8165 $\pm$ 0.3824	0.5040 $\pm$ 0.0166
GP-ARD-RBF	0.4482 $\pm$ 0.0466	0.6581 $\pm$ 0.0641	0.0249 $\pm$ 0.0108	2.2212 $\pm$ 0.0304	0.9271 $\pm$ 0.0166
GP-ARD-Laplace	0.5804 $\pm$ 0.0655	1.1556 $\pm$ 0.0268	0.0481 $\pm$ 0.0046	4.3960 $\pm$ 0.0471	1.5367 $\pm$ 0.3320
GP-ARD-Laplace-full	2.6116 $\pm$ 0.2181	2.9738 $\pm$ 0.0099	0.0500 $\pm$ 0.0000	28.6954 $\pm$ 0.1360	0.7235 $\pm$ 0.0093
<b>GP-RFM-Laplace</b>	0.4965 $\pm$ 0.0431	0.9515 $\pm$ 0.0279	0.0481 $\pm$ 0.0046	3.4807 $\pm$ 0.0345	1.3571 $\pm$ 0.2191
<b>GP-RFM-Laplace-diag</b>	0.4849 $\pm$ 0.0441	0.8953 $\pm$ 0.0309	0.0474 $\pm$ 0.0052	3.2305 $\pm$ 0.0402	1.0165 $\pm$ 0.0221
NGBoost	0.5141 $\pm$ 0.0463	0.6078 $\pm$ 0.1680	0.0274 $\pm$ 0.0148	1.9866 $\pm$ 0.2474	2.7234 $\pm$ 0.2241
CatBoost-Ensemble	0.5588 $\pm$ 0.1193	0.5794 $\pm$ 0.2291	0.0373 $\pm$ 0.0203	1.7964 $\pm$ 0.6647	5.9525 $\pm$ 0.7011

**Table 21:** UCI dataset: Kin8nm (8192 samples; 8 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	$0.0756 \pm 0.0023$	$-0.7730 \pm 0.0057$	$0.0499 \pm 0.0003$	$0.6547 \pm 0.0019$	$7.8375 \pm 0.0981$
GP-Laplace	$0.0761 \pm 0.0025$	$-1.0735 \pm 0.0148$	$0.0376 \pm 0.0033$	$0.4172 \pm 0.0014$	$12.5205 \pm 0.5642$
deep Kernel Learning	$0.0722 \pm 0.0058$	$-1.1991 \pm 0.0704$	$0.0157 \pm 0.0082$	$0.3139 \pm 0.0172$	$3.3887 \pm 0.0272$
GP-ARD-RBF	$0.0747 \pm 0.0023$	$-0.7759 \pm 0.0056$	$0.0499 \pm 0.0003$	$0.6541 \pm 0.0019$	$7.8674 \pm 0.0272$
GP-ARD-Laplace	$0.0722 \pm 0.0022$	$-1.1377 \pm 0.0147$	$0.0381 \pm 0.0036$	$0.3853 \pm 0.0012$	$12.0966 \pm 0.4330$
GP-ARD-Laplace-full	$0.0833 \pm 0.0062$	$-0.8490 \pm 0.1591$	$0.0457 \pm 0.0044$	$0.5756 \pm 0.1107$	$7.4953 \pm 0.3022$
<b>GP-RFM-Laplace</b>	$0.0657 \pm 0.0018$	$-1.2620 \pm 0.0109$	$0.0321 \pm 0.0042$	$0.3282 \pm 0.0026$	$13.2497 \pm 0.6537$
<b>GP-RFM-Laplace-diag</b>	$0.0755 \pm 0.0016$	$-1.0997 \pm 0.0106$	$0.0381 \pm 0.0036$	$0.3948 \pm 0.0012$	$11.9409 \pm 0.1580$
NGBoost	$0.1819 \pm 0.0038$	$-0.3626 \pm 0.0185$	$0.0082 \pm 0.0053$	$0.6536 \pm 0.0051$	$22.3981 \pm 0.2512$
CatBoost-Ensemble	$0.1388 \pm 0.0032$	$-0.6557 \pm 0.0219$	$0.0081 \pm 0.0051$	$0.4712 \pm 0.0029$	$24.8655 \pm 1.6878$

**Table 22:** UCI dataset: Naval Plant Maintenance (11934 samples; 16 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	$0.0002 \pm 0.0001$	$-6.1782 \pm 0.8216$	$0.1608 \pm 0.2245$	$0.0023 \pm 0.0000$	$11.5056 \pm 1.2598$
GP-Laplace	$0.0003 \pm 0.0000$	$-5.7682 \pm 0.0052$	$0.0500 \pm 0.0002$	$0.0049 \pm 0.0000$	$17.4098 \pm 0.2460$
deep Kernel Learning	$0.0005 \pm 0.0001$	$-4.1528 \pm 0.2374$	$0.0500 \pm 0.0000$	$0.0253 \pm 0.0070$	$4.6008 \pm 0.0534$
GP-ARD-RBF	$0.0002 \pm 0.0000$	$-6.4786 \pm 0.0103$	$0.1825 \pm 0.3225$	$0.0023 \pm 0.0000$	$10.9200 \pm 0.0534$
GP-ARD-Laplace	$0.0002 \pm 0.0000$	$-5.9669 \pm 0.0043$	$0.0500 \pm 0.0000$	$0.0040 \pm 0.0000$	$17.2876 \pm 0.1934$
GP-ARD-Laplace-full	$0.0005 \pm 0.0001$	$-4.1346 \pm 0.2305$	$0.0500 \pm 0.0000$	$0.0255 \pm 0.0047$	$10.4092 \pm 0.4880$
<b>GP-RFM-Laplace</b>	$0.0003 \pm 0.0000$	$-5.9195 \pm 0.0361$	$0.0500 \pm 0.0000$	$0.0041 \pm 0.0002$	$19.8079 \pm 0.52316$
<b>GP-RFM-Laplace-diag</b>	$0.0002 \pm 0.0000$	$-5.8776 \pm 0.0038$	$0.0500 \pm 0.0000$	$0.0044 \pm 0.0000$	$18.3550 \pm 0.0997$
NGBoost	$0.0059 \pm 0.0001$	$-3.9178 \pm 0.0138$	$0.0479 \pm 0.0013$	$0.0234 \pm 0.0002$	$37.5817 \pm 5.8329$
CatBoost-Ensemble	$0.0016 \pm 0.0001$	$-5.4828 \pm 0.0322$	$0.0444 \pm 0.0023$	$0.0059 \pm 0.0002$	$29.2242 \pm 2.6337$

**Table 23:** UCI dataset: Combined Cycle Power Plant (9568 samples; 4 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	$3.9517 \pm 0.1233$	$3.1728 \pm 0.0070$	$0.0484 \pm 0.0010$	$33.5592 \pm 0.0525$	$9.2673 \pm 0.8142$
GP-Laplace	$3.4041 \pm 0.1247$	$2.7214 \pm 0.0198$	$0.0310 \pm 0.0037$	$17.8956 \pm 0.0945$	$14.7721 \pm 0.7598$
deep Kernel Learning	$3.9615 \pm 0.1201$	$2.8441 \pm 0.0209$	$0.0362 \pm 0.0040$	$19.6622 \pm 0.5423$	$3.7840 \pm 0.0196$
GP-ARD-RBF	$3.9448 \pm 0.1227$	$3.1725 \pm 0.0069$	$0.0484 \pm 0.0010$	$33.5615 \pm 0.0524$	$9.2302 \pm 0.0196$
GP-ARD-Laplace	$2.6824 \pm 0.1770$	$2.7131 \pm 0.0176$	$0.0438 \pm 0.0023$	$20.8760 \pm 0.1848$	$15.0723 \pm 0.6198$
GP-ARD-Laplace-full	$3.5091 \pm 0.2767$	$2.8893 \pm 0.2700$	$0.0824 \pm 0.1988$	$23.0570 \pm 0.6857$	$11.6361 \pm 15.4711$
<b>GP-RFM-Laplace</b>	$3.2018 \pm 0.1364$	$2.6849 \pm 0.0214$	$0.0343 \pm 0.0034$	$17.7328 \pm 0.1058$	$15.2813 \pm 0.5343$
<b>GP-RFM-Laplace-diag</b>	$3.2550 \pm 0.1277$	$2.6963 \pm 0.0200$	$0.0342 \pm 0.0039$	$17.8465 \pm 0.0886$	$14.0767 \pm 0.1701$
NGBoost	$3.8750 \pm 0.1488$	$2.7677 \pm 0.0749$	$0.0074 \pm 0.0041$	$14.5355 \pm 0.6960$	$11.9584 \pm 2.2816$
CatBoost-Ensemble	$3.3635 \pm 0.3802$	$2.6596 \pm 0.1451$	$0.0220 \pm 0.0156$	$11.1089 \pm 1.9365$	$25.3693 \pm 4.5353$

**Table 24:** UCI dataset: Wine Quality Red (1599 samples; 11 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	0.6026 $\pm$ 0.0440	0.9114 $\pm$ 0.0648	0.0194 $\pm$ 0.0099	2.4992 $\pm$ 0.0667	3.2311 $\pm$ 0.0717
GP-Laplace	0.5517 $\pm$ 0.0433	0.8475 $\pm$ 0.0521	0.0197 $\pm$ 0.0095	2.6585 $\pm$ 0.0346	4.1511 $\pm$ 0.1734
deep Kernel Learning	0.6440 $\pm$ 0.0587	0.9905 $\pm$ 0.1014	0.0266 $\pm$ 0.0190	2.4026 $\pm$ 0.1845	1.4830 $\pm$ 0.0280
GP-ARD-RBF	0.6033 $\pm$ 0.0412	0.9202 $\pm$ 0.0553	0.0172 $\pm$ 0.0095	2.6270 $\pm$ 0.1211	3.3238 $\pm$ 0.0280
GP-ARD-Laplace	0.5635 $\pm$ 0.0435	0.8547 $\pm$ 0.0528	0.0181 $\pm$ 0.0117	2.6115 $\pm$ 0.0285	3.9449 $\pm$ 0.2618
GP-ARD-Laplace-full	0.5747 $\pm$ 0.0416	0.8811 $\pm$ 0.0519	0.0203 $\pm$ 0.0115	2.6116 $\pm$ 0.0423	2.3756 $\pm$ 0.2144
<b>GP-RFM-Laplace</b>	0.5569 $\pm$ 0.0435	0.8500 $\pm$ 0.0558	0.0184 $\pm$ 0.0094	2.6239 $\pm$ 0.0531	4.0593 $\pm$ 0.2478
<b>GP-RFM-Laplace-diag</b>	0.5617 $\pm$ 0.0488	0.8618 $\pm$ 0.0649	0.0200 $\pm$ 0.0126	2.6260 $\pm$ 0.0683	4.0605 $\pm$ 0.4318
NGBoost	0.6212 $\pm$ 0.0445	0.9278 $\pm$ 0.0741	0.0209 $\pm$ 0.0132	2.2635 $\pm$ 0.1561	2.6412 $\pm$ 0.6679
CatBoost-Ensemble	0.6134 $\pm$ 0.0507	0.8973 $\pm$ 0.1082	0.0291 $\pm$ 0.0208	2.0483 $\pm$ 0.0399	15.4885 $\pm$ 2.2552

**Table 25:** UCI dataset: Yacht Hydrodynamics (308 samples; 6 covariates)

	RMSE ( $\downarrow$ )	NLL ( $\downarrow$ )	CE (95%) ( $\downarrow$ )	IL (95%) ( $\downarrow$ )	Time ( $\downarrow$ )
GP-RBF	0.8173 $\pm$ 0.2675	1.0579 $\pm$ 0.0704	0.0366 $\pm$ 0.0164	3.7188 $\pm$ 0.1813	0.7283 $\pm$ 0.0648
GP-Laplace	2.9169 $\pm$ 0.8372	2.5045 $\pm$ 0.1010	0.0310 $\pm$ 0.0190	15.6999 $\pm$ 0.3866	1.1829 $\pm$ 0.2183
deep Kernel Learning	0.7553 $\pm$ 0.3034	2.3936 $\pm$ 0.0231	0.0500 $\pm$ 0.0000	16.8325 $\pm$ 0.4511	0.4360 $\pm$ 0.0097
GP-ARD-RBF	0.5183 $\pm$ 0.2543	0.8530 $\pm$ 0.2459	0.0368 $\pm$ 0.0159	2.8902 $\pm$ 0.0785	0.8103 $\pm$ 0.0097
GP-ARD-Laplace	1.1895 $\pm$ 0.5616	1.9742 $\pm$ 0.0777	0.0366 $\pm$ 0.0164	10.1187 $\pm$ 0.1586	1.4123 $\pm$ 0.3088
GP-ARD-Laplace-full	4.6409 $\pm$ 1.4430	3.4261 $\pm$ 0.0448	0.0468 $\pm$ 0.0097	44.0111 $\pm$ 0.5874	0.7122 $\pm$ 0.0339
<b>GP-RFM-Laplace</b>	1.0447 $\pm$ 0.3585	1.4985 $\pm$ 0.2114	0.0323 $\pm$ 0.0162	4.7886 $\pm$ 0.1456	1.2185 $\pm$ 0.2006
<b>GP-RFM-Laplace-diag</b>	1.0187 $\pm$ 0.3369	1.5235 $\pm$ 0.1761	0.0310 $\pm$ 0.0169	5.2000 $\pm$ 0.2562	0.8918 $\pm$ 0.0676
NGBoost	0.7487 $\pm$ 0.2946	0.7046 $\pm$ 0.4721	0.0402 $\pm$ 0.0364	2.1121 $\pm$ 0.9981	2.0757 $\pm$ 0.2763
CatBoost-Ensemble	1.2838 $\pm$ 0.7056	0.4008 $\pm$ 0.6086	0.1053 $\pm$ 0.0749	1.5808 $\pm$ 1.3508	5.0563 $\pm$ 1.0566

# Paper II

## Title

Invertible Kernel PCA with Random Fourier Features

## Authors

Daniel Gedon, Antônio H. Ribeiro, Niklas Wahlström, Thomas B. Schön

## Edited version of

D. Gedon, A. H. Ribeiro, N. Wahlström, and T. B. Schön. “Invertible Kernel PCA with Random Fourier Features.” In: *IEEE Signal Processing Letters*. 2023



# Invertible Kernel PCA with Random Fourier Features



## Abstract

Kernel principal component analysis (kPCA) is a widely studied method to construct a low-dimensional data representation after a nonlinear transformation. The prevailing method to reconstruct the original input signal from kPCA—an important task for denoising—requires us to solve a supervised learning problem. In this paper, we present an alternative method where the reconstruction follows naturally from the compression step. We first approximate the kernel with random Fourier features. Then, we exploit the fact that the nonlinear transformation is invertible in a certain subdomain. Hence, the name *invertible kernel PCA (ikPCA)*. We experiment with different data modalities and show that ikPCA performs similarly to kPCA with supervised reconstruction on denoising tasks, making it a strong alternative.

## 1 Introduction

Principal Component Analysis (PCA) involves finding a projection matrix  $\mathbf{P}$  that transforms a given input  $\mathbf{x} \in \mathbb{R}^p$  into a lower-dimensional representation  $\mathbf{z} = \mathbf{P}\mathbf{x} \in \mathbb{R}^d$ , with  $d < p$ . Conversely, given a lower-dimensional representation, the original input space can be reconstructed with the inverse transformation  $\hat{\mathbf{x}} = \mathbf{P}^\top \mathbf{z}$ . The data  $\mathbf{x}$  is often assumed to lie on a low-dimensional manifold. In such cases, PCA is beneficial since it enables the extraction of the most important features or directions of maximum variability in the data. The algorithm is optimal [SB14] in the sense that there is no reconstruction matrix  $\mathbf{U}$  and reduction matrix  $\mathbf{V}$  such that the average distance between the original and reconstructed vector  $\|\hat{\mathbf{x}} - \mathbf{U}\mathbf{V}\mathbf{x}\|_2$  is smaller than for  $\mathbf{U} = \mathbf{P}^\top$  and  $\mathbf{V} = \mathbf{P}$ . Importantly, the matrix  $\mathbf{P}$  serves both as a tool for dimensionality reduction and as a means for reconstructing the original input through its transpose  $\mathbf{P}^\top$ .

Kernel PCA (kPCA) builds upon traditional PCA by enabling the study of the principal components after a nonlinear transformation [SSM97]. This allows for the generalization of the assumption that the data lies on a low-dimensional linear manifold to cases where this manifold is nonlinear. Traditional PCA

might not be capable of retrieving useful low-dimensional representations  $\mathbf{z}$  in this scenario, but kPCA might succeed by using PCA after a nonlinear transformation  $\Phi$  of the input  $\mathbf{x}$  into a (possibly infinite-dimensional) feature space  $\mathcal{F}$

$$\mathbf{z} = \mathbf{P}\Phi(\mathbf{x}). \quad (1)$$

kPCA is indeed a natural and valuable idea. However, while the dimensionality reduction can be easily computed it is far from obvious how to obtain a reconstructed  $\hat{\mathbf{x}}$  from  $\mathbf{z}$ .

This inverse reconstruction problem is known as the pre-image problem. Solutions are proposed based on gradient descent [Bur96], nonlinear optimization [Mik+98] or distance constraints in feature space [KT03]. The most widely disseminated solution by Bakır et al. [BWS04], is to apply (1) to construct a data set  $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{z}_i)\}_{i=1}^n$  consisting of original inputs and their low-dimensional representations. The goal is to find a nonlinear function  $f$  that maps  $\mathbf{z}_i$  back to  $\mathbf{x}_i$ . This approach is available, for instance, in scikit-learn [Ped+11] or the multivariate statistics package for Julia [Bez+17]. However, there are drawbacks to this approach: Unlike PCA, reconstruction is not an immediate by-product of kPCA and instead requires solving a supervised learning (SL) problem. Here, we denote this combination as kPCA+SL. Moreover, since the function  $f$  needs to be nonlinear, the supervised problem of finding the map between  $\mathbf{z}$  and  $\mathbf{x}$  usually results in a non-convex optimization problem. Indeed, direct nonlinear approaches—such as autoencoders [BK88; HZ93] and variational autoencoders [KW14; RMW14]—that concurrently implement dimensionality reduction and reconstruction, can yield significantly improved performance over kPCA+SL. While deep autoencoders are popular components of generative models, they require solving a non-convex optimization problem. Contrarily, kernel methods and PCA are well-understood and widely adopted preprocessing steps.

We propose a new formulation of kPCA that provides the reconstruction method as a direct by-product. The method works for any translational-invariant kernel. As we will discuss in Section 2, any such kernel can be approximated by a feature map of the type  $\Phi(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$  with a nonlinearity  $\sigma$  and  $\mathbf{W} \in \mathbb{R}^{r \times p}$ . Here, the dimensionality is reduced by the following sequence of computations

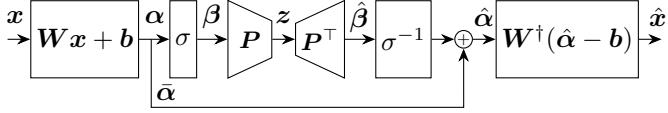
$$\boldsymbol{\alpha} = \mathbf{W}\mathbf{x} + \mathbf{b}, \quad (2a)$$

$$\boldsymbol{\beta} = \sigma(\boldsymbol{\alpha}), \quad (2b)$$

$$\mathbf{z} = \mathbf{P}\boldsymbol{\beta}. \quad (2c)$$

The method we propose involves inverting the operations step-by-step, as depicted in Figure 1. If a particular operation  $\sigma(\boldsymbol{\alpha})$  cannot be inverted, we decompose the vector  $\boldsymbol{\alpha}$  into two components  $\boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}$  and  $\bar{\boldsymbol{\alpha}}$ , such that  $\boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}$  belongs to a domain where  $\sigma$  is invertible. We can use PCA to compress and

decompress the first component, while the second component is bypassed. In this way, we avoid any nonlinear supervised problem and the reconstruction follows directly.



**Figure 1:** Illustration of our invertible kernel PCA method.

## 2 Background

Let,  $k(\mathbf{x}, \mathbf{y})$  be a positive semidefinite kernel

$$k(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle = \sum_i \phi_i(\mathbf{x})\phi_i(\mathbf{y}), \quad (3)$$

where  $\Phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots)$  denotes a sequence of values that maps the input into the feature space  $\mathcal{F}$ .

### 2.1 Kernel PCA

For a set of observations  $\{\mathbf{x}_i\}_{i=1}^n$  the empirical covariance matrix in  $\mathcal{F} \times \mathcal{F}$  is given by

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^n \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_i)^\top. \quad (4)$$

The spectral decomposition of this matrix yields

$$\hat{\Sigma} = \sum_i \lambda_i \mathbf{v}_i \mathbf{v}_i^\top, \quad (5)$$

such that  $\lambda_1 \geq \lambda_2 \geq \dots$ . We define the projection into the first  $d$  components as  $\mathbf{P} = [\mathbf{v}_1 \ \dots \ \mathbf{v}_d]^\top$ . For which we can obtain the lower dimensional representation  $\mathbf{z} = \mathbf{P}\Phi(\mathbf{x})$ .

### 2.2 Infinite dimensional feature maps

In practice, the kernel trick enables working with feature spaces of infinite dimension. The method we propose here is however intended for finite-dimensional feature spaces. Hence, when dealing with infinitely dimensional

feature maps, we resort to approximations. Specifically, to work with finite-dimensional features, we use  $r$  components of the feature map denoted as  $\tilde{\Phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \dots, \phi_r(\mathbf{x}))$  to approximate the kernel, meaning that we can write  $k(\mathbf{x}, \mathbf{y}) \approx \langle \tilde{\Phi}(\mathbf{x}), \tilde{\Phi}(\mathbf{y}) \rangle$ .

We follow the development of [RR08] using *random Fourier features* to approximate a translation-invariant kernel, i.e. kernels of the form  $k(\mathbf{x}, \mathbf{y}) = g(\mathbf{x} - \mathbf{y})$ . Bochner theorem guarantees that this kernel is continuous and positive semidefinite iff  $g(\delta)$  is the Fourier transform of a probability distribution  $p(\omega)$ , possibly re-scaled.

Take as an example the Gaussian kernel  $k(\mathbf{x}, \mathbf{y}) = \exp \frac{-\|\mathbf{x} - \mathbf{y}\|_2^2}{2}$  which allows for the decomposition (3) only when considering an infinite dimensional feature space. To approximate this features space with random Fourier features, let  $\mathbf{W} \in \mathbb{R}^{r \times p}$  be a matrix with random i.i.d. entries drawn from the distribution  $p(\omega)$  and let  $\mathbf{b} \in \mathbb{R}^r$  be a vector drawn i.i.d. from  $\mathcal{U}(-\pi, \pi)$ . Then, we obtain

$$\tilde{\Phi}(\mathbf{x}) = \sqrt{2} \sin(\mathbf{W}\mathbf{x} + \mathbf{b}), \quad (6)$$

where  $\sin$  is applied element-wise. It is proved in [RR08] that,  $\langle \tilde{\Phi}(\mathbf{x}), \tilde{\Phi}(\mathbf{y}) \rangle$  converges uniformly to  $k(\mathbf{x}, \mathbf{y})$ . Moreover, the convergence is exponentially fast in  $r$ .

### 3 Invertible kernel PCA

Let us consider feature maps of the type

$$\Phi(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}), \quad (7)$$

where  $\mathbf{W} \in \mathbb{R}^{r \times p}$ ,  $\mathbf{b} \in \mathbb{R}^r$  and  $\sigma$  is a nonlinearity applied element-wise. The discussion in the previous section motivates how these feature maps can be used to approximate the space associated with any translational-invariant kernel. Next, we detail how to invert the operations, given that the dimensionality reduction was computed according to (2). One of the key challenges is the fact that the activation function  $\sigma$  is in general non-invertible. We describe our solution to deal with these problems next.

#### 3.1 Non-invertible activation functions

In most cases of interest, the nonlinear function  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ ,  $\sigma : \alpha \mapsto \beta$  is non-invertible in the entire domain  $\mathbb{R}$  but might be invertible in a subdomain  $\mathcal{X} \subset \mathbb{R}$ .

This holds for almost all activation functions. Denote  $\sigma_{\mathcal{X}}$  as  $\sigma$  restricted to  $\mathcal{X}$ , then the inverse  $\sigma_{\mathcal{X}}^{-1}$  is well-defined. Let  $\bar{\alpha} = \alpha - \sigma_{\mathcal{X}}^{-1} \circ \sigma(\alpha)$ . Consider two examples: First, for the ReLU activation function  $\beta = \sigma(\alpha) = \max(\alpha, 0)$ , the invertible domain is  $\mathcal{X} = [0, \infty)$ . Thus,  $\sigma_{\mathcal{X}}^{-1}(\beta) = \beta$  and  $\bar{\alpha} = \min(\alpha, 0)$ . Second, for  $\beta = \sigma(\alpha) = \sin \alpha$ , as used in random Fourier features, we have that  $\sigma$  is invertible in  $\mathcal{X} = (-\pi/2, \pi/2]$ . Thus,  $\sigma_{\mathcal{X}}^{-1}(\beta) = \arcsin \beta$  and  $\bar{\alpha} = (-1)^k \alpha + \pi k$  for some  $k \in \mathbb{Z}$ .

## 3.2 ikPCA

The reconstruction method inverts the operations in (2) step-by-step. We can write

$$\hat{\beta} = \mathbf{P}^\top \mathbf{z}, \quad (8a)$$

$$\hat{\alpha} = \sigma_{\mathcal{X}}^{-1}(\hat{\beta}) + \bar{\alpha}, \quad (8b)$$

$$\hat{x} = \arg_x \min \|Wx + b - \hat{\alpha}\|_2^2 + \lambda \|x\|_2^2. \quad (8c)$$

The first step (8a) inverts the dimensionality reduction, and is motivated by the same reasoning as PCA: the projection matrix  $\mathbf{P}$  is such that the reconstruction error  $\|\hat{\beta} - \mathbf{P}^\top \mathbf{z}\|_2$  is minimal. The second step (8b), inverts the nonlinear function  $\sigma$  on the subdomain  $\mathcal{X}$  and adds the bypassed non-invertible part  $\bar{\alpha}$ . Finally, the last step (8c) inverts the linear map  $x \mapsto Wx + b$  by solving a Ridge regression problem. Notice that for  $\lambda \rightarrow 0^+$  the last step reduces to  $\hat{x} = W^\dagger(\hat{\alpha} - b)$ , where  $W^\dagger$  is the pseudo-inverse of  $W$ , as in Figure 1.

## 4 Numerical examples

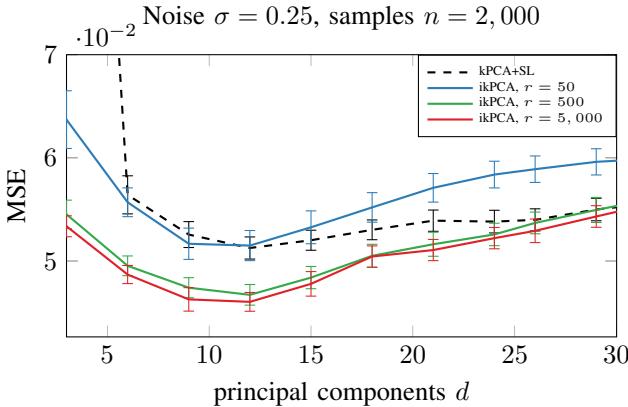
In this section, we outline the experiments to evaluate the performance of the proposed ikPCA method. We focus on the task of denoising inputs of various modalities. Quantitatively, we evaluate the mean square error (MSE) between the de-noised test signal and the true non-noisy test signal. We compare ikPCA with PCA and kPCA+SL due to their structural similarity. To ensure a fair comparison, we did not consider denoising autoencoders, which present hierarchical, deep models. In the discussion, we detail how our methodology could be extended to neural networks and a setup that could be better compared with autoencoders.

In all experiments, we consider the Gaussian kernel (or its random Fourier feature approximation) and present the results in terms of mean and standard deviation over 20 random runs. Parameters of the methods which are fixed

in an experiment were optimized through hyperparameter grid search. For reproducibility, we release our code publicly<sup>1</sup>.

## 4.1 Synthetic toy data: s-curve

We generated synthetic 3-dimensional data points in the shape of the letter ‘S’ using the s-curve toy problem. For training and testing, we generate  $n = 2,000$  data points and add Gaussian noise with  $\sigma = 0.25$ . For kPCA+SL we set the kernel width  $\gamma = 1$  and the reconstruction Ridge strength  $\lambda = 1$ ; for ikPCA we set  $\gamma = 0.5$  and  $\lambda = 1$ . The results are presented in Figure 2. Our proposed ikPCA method is capable of denoising the data in a comparable manner to kPCA+SL. For this problem, we observe that as few as  $r = 50$  random Fourier features were sufficient to match the performance of kPCA+SL. This provides ikPCA with a computational advantage over kPCA+SL, which needs to invert a  $n \times n$  matrix.



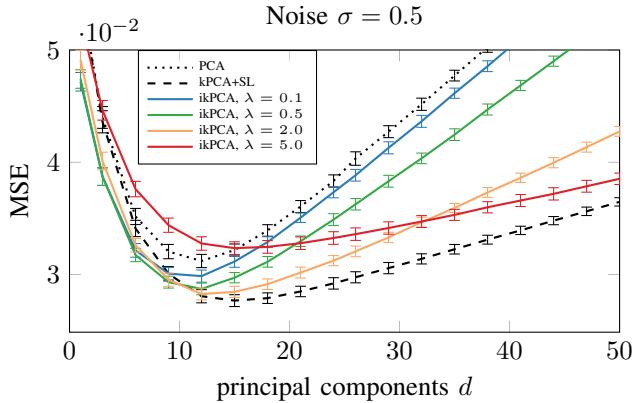
**Figure 2:** S-curve toy example. Reconstruction MSE for a different number of random features chosen for ikPCA.

## 4.2 USPS Images

We utilize the USPS data set which contains handwritten digits in a greyscale format of size  $16 \times 16$  and add Gaussian noise with  $\sigma = 0.5$ . We use  $n = 1,000$  images for training and 400 images for testing. For kPCA+SL we set  $\gamma = 5 \cdot 10^{-3}$  and  $\lambda = 10^{-2}$ ; for ikPCA we use 30,000 random Fourier features and set  $\gamma = 10^{-4}$ . In Figure 3, we vary the regularization parameter  $\lambda$  of ikPCA. Again, our findings show that ikPCA performs similarly to

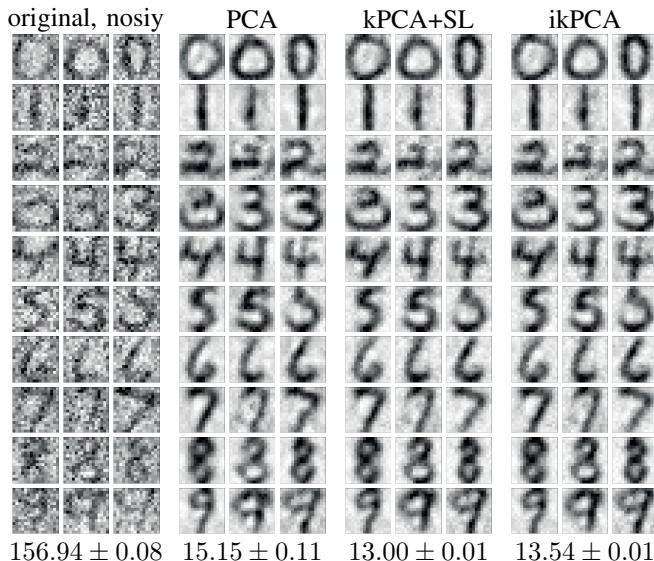
<sup>1</sup>Code is available at [https://github.com/dgedon/invertible\\_kernel\\_PCA](https://github.com/dgedon/invertible_kernel_PCA)

kPCA+SL for an optimal number of principal components  $d$ . The figure further suggests that Ridgeless reconstruction behaves comparably in performance to PCA. However, excessive regularization negatively affects the overall reconstruction performance.



**Figure 3:** USPS data. Effect of regularization parameter  $\lambda$ .

Figure 4 displays image denoising results for noise scale  $\sigma = 0.25$ . For ikPCA we chose the optimal regularization  $\lambda = 1.3$ . The number of principal components  $d$  is chosen for each method such that the MSE is minimised. All methods demonstrate visually comparable image denoising capabilities, which is supported by the difference in MSE from Figure 3.



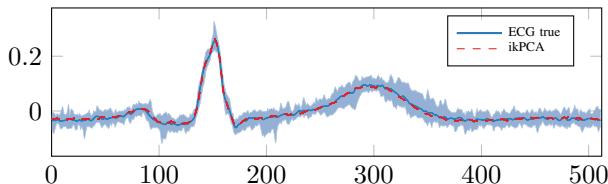
**Figure 4:** USPS reconstruction. MSE below images as  $10^{-2}$ .

### 4.3 Electrocardiogram

The electrocardiogram (ECG) is a routine, medical test that records the heart’s electrical activity, typically used to diagnose various heart conditions. Noise measured during the recording can complicate the diagnosis. Several methods have been proposed to de-noise the ECG. For comparisons, two approaches have been suggested: (1) artificially adding noise to the signal and comparing with the original one itself [Sam+07; Xio+16; Chi+19], or (2) de-noise the existing signal and comparing it to the mean beat as the noise-free reference [Cas+07; Joh+09]. We choose the latter approach to account for real-world noise scenarios.

We utilize ECGs from the China Physiological Signal Challenge 2018 (CPSC)<sup>2</sup> which contains data between 6 and 60 seconds long [Liu+18]. From the 918 ECGs with no abnormalities, we selected the longest recordings and focused on a single lead in this example. To extract the beats, we first remove baseline wander with a high-pass filter. Then, we identify the R-peaks [Xie+23], resample the interval between each peak to 512 samples and finally locate the R-peak at the 150th sample following the preprocessing approach of Johnstone et al. [Joh+09].

We extracted 70 beats from the selected ECG; 49 for training and 21 for testing. Applying kPCA+SL with  $\gamma = 10$  and  $\lambda = 15$ , and ikPCA with  $\gamma = 5 \cdot 10^{-5}$  and  $\lambda = 10$ , along with the minimum of 512 random Fourier features  $r$ , we achieved perfect signal denoising using only the first component, as shown in Figure 5. Quantitatively over 500 simulations, the MSE for ikPCA was  $2.6 \pm 0.8 \cdot 10^{-5}$ , similar to that of kPCA+SL, while PCA had a slightly higher MSE.



**Figure 5:** Denoising of ECG beats from lead I. The blue area marks the min/max values of the 21 test beats. The red dashed lines show all test reconstructions with ikPCA.

<sup>2</sup>Data is available at <http://2018.icbeb.org/Challenge.html>

## 5 Computational considerations

The computational complexity is not increased by adding the reconstruction stage for ikPCA. The reason is that the cost of obtaining the reconstruction is smaller than the cost of the kPCA decomposition (whenever the input dimension  $p$  is lower than the number of samples  $n$ ). However, our method requires the kernel map to be approximated by  $r$  random Fourier features. When  $r < n$  this might reduce the computational cost, but when  $r > n$  the computational cost is increased by a factor of  $r/n + 1$  compared to that of kPCA.

**kPCA computational cost** Some kernels have closed forms that can be computed in  $\mathcal{O}(1)$  operations. The cost for kPCA is then dominated by the inversion of the Gram matrix  $\mathbf{K}$  which requires  $\mathcal{O}(n^3)$  operations. The Gram matrix being the matrix with entry  $(i, j)$  equal to  $k(\mathbf{x}_i, \mathbf{x}_j)$ .

**Computation cost of PCA in the feature space** In ikPCA we approximate the kernels with finite,  $r$ -dimensional features  $\Phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \dots, \phi_r(\mathbf{x}))$ , and perform PCA on the covariance matrix of the features  $\hat{\Sigma} = \frac{1}{n} \sum_i^n \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_i)^\top$ . Computing the entries of the matrix and its spectral decomposition requires  $\mathcal{O}(r^2 n + r^3)$  operations. Hence, for  $r < n$ , approximating the kernel and computing the spectral decomposition of  $\hat{\Sigma}$  might be computationally more efficient than working directly with the Gram Matrix  $\mathbf{K}$  as in kPCA.

However, if  $r > n$  this advantage is diminished and it can be efficient to work with the Gram matrix  $\mathbf{K} = \Phi(\mathbf{X})^\top \Phi(\mathbf{X})$  instead of  $\hat{\Sigma}$ .  $\mathbf{K}$  has the same (nonzero) eigenvalues as  $n\hat{\Sigma}$ , and its eigenvectors multiplied by  $\Phi(\mathbf{X})$  yield the eigenvectors of  $n\hat{\Sigma}$ . The cost in this formulation is  $\mathcal{O}(rn^2 + n^3)$ . Therefore, the cost is a factor of  $r/n + 1$  times higher than the cost obtained for kernels with a closed-form solution.

According to Claim 1 in [RR08],  $r = \Omega\left(\frac{p}{\epsilon^2} \log \frac{D}{\epsilon}\right)$  random Fourier features are required to ensure an approximation error smaller than  $\epsilon$  on a space of diameter  $D$ . Thus,  $r$  grows linearly with the input dimension  $p$ . In the case of the s-curve example,  $r \ll n$ , whereas in the USPS example,  $r > n$ , due to large  $p$  and our method's computational advantage is lost. For high-dimensional data like the latter, Nyström approximations [WS00] could be used and be more efficient in terms of  $r$  [Yan+12].

**Cost of reconstruction** For ikPCA the cost is dominated by solving the optimization (8c). Here, we require computing the SVD of  $\mathbf{W}$  a single time with  $\mathcal{O}(p^3 + p^2 r)$ . The cost of solving the reconstruction is then  $\mathcal{O}(pr + dr)$  for each new  $\hat{\alpha}$ . To compare, the reconstruction in Figure 4 takes 0.001 sec for PCA, 0.030 sec on an Intel i9-CPU for kPCA, and 1.867 sec for ikPCA which is due to  $r = 40.000$ .

## 6 Conclusion and discussion

We propose an invertible version of kPCA+SL. While the traditional approach solves a supervised problem to map back from the latent space to the input space, our method obtains this mapping naturally. We approximate the kernel transformation with random Fourier features  $\Phi(x) = \sigma(\mathbf{W}x + \mathbf{b})$ . Although the nonlinear function  $\sigma$  might not be invertible, we observe that it can be inverted in a subdomain. We can exploit this observation by decomposing its input into invertible and non-invertible parts and bypassing the second. We show the effectiveness of our approach for denoising in three examples: an s-curve toy problem, the USPS image data set and ECGs.

We compare our method with symmetric kPCA+SL. Symmetry implies here that the kernel for compression and reconstruction are defined identically, which is motivated by implementations in common frameworks [Ped+11; Bez+17]. However, the method in [BWS04] is not limited to this by design. Conversely, ikPCA is required to have a symmetric setup due to the natural inversion of the nonlinear transformation in the reconstruction. While our method aligns well with kPCA+SL in the numerical experiments we presented, it remains uncertain how it would compare against a well-tuned non-symmetric kPCA+SL.

Despite the simplicity of our method, there is a wide array of possible extensions. To extend the representational power, we can stack multiple layers of  $\Phi(x) = \sigma(\mathbf{W}x + \mathbf{b})$  transformations in a hierarchical way. Hence, we obtain a structure which is closer to that of a deep autoencoder. This may allow drawing further connections between the theoretically well-established kernel regime and neural networks. In a similar direction, we can view the random Fourier features in our method as an untrained, single-layer neural network. Extending our method to trained neural networks would allow input reconstruction without re-training. Finally, we experiment with underparameterized data (USPS example with  $p/n \approx 0.25$ ) and overparameterized data (ECG example with  $p/n \approx 10$ ). This fact, combined with the use of a high number of random Fourier features, raises questions about overparameterization and benign overfitting of denoising models [Rad+18; Bar+20].

**Acknowledgments** This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation; by Kjell och Märta Beijer Foundation; and by the Swedish Research Council (VR) via the project Physics-informed machine learning (registration number: 2021-04321). The computations were enabled by the supercomputing resource Berzelius provided by National Supercomputer Centre at Linköping University and the Knut and Alice Wallenberg foundation.

## References

- [Bar+20] P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler. “Benign overfitting in linear regression.” In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30063–30070 (cit. on p. II-10).
- [Bez+17] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. “Julia: A fresh approach to numerical computing.” In: *SIAM Review* 59.1 (2017), pp. 65–98 (cit. on pp. II-2, II-10).
- [BK88] H. Bourlard and Y. Kamp. “Auto-association by multilayer perceptrons and singular value decomposition.” In: *Biological cybernetics* 59.4-5 (1988), pp. 291–294 (cit. on p. II-2).
- [Bur96] C. J. C. Burges. “Simplified Support Vector Decision Rules.” In: *International Conference on Machine Learning*. 1996, pp. 71–77 (cit. on p. II-2).
- [BWS04] G. H. Bakir, J. Weston, and B. Schölkopf. “Learning to find pre-images.” In: *Advances in neural information processing systems* 16 (2004), pp. 449–456 (cit. on pp. II-2, II-10).
- [Cas+07] F. Castells, P. Laguna, L. Sörnmo, A. Bollmann, and J. M. Roig. “Principal component analysis in ECG signal processing.” In: *EURASIP Journal on Advances in Signal Processing* 2007 (2007), pp. 1–21 (cit. on p. II-8).
- [Chi+19] H.-T. Chiang, Y.-Y. Hsieh, S.-W. Fu, K.-H. Hung, Y. Tsao, and S.-Y. Chien. “Noise reduction in ECG signals using fully convolutional denoising autoencoders.” In: *IEEE Access* 7 (2019), pp. 60806–60813 (cit. on p. II-8).
- [HZ93] G. E. Hinton and R. Zemel. “Autoencoders, minimum description length and Helmholtz free energy.” In: *Advances in neural information processing systems* 6 (1993) (cit. on p. II-2).
- [Joh+09] I. M. Johnstone, A. Y. Lu, B. Nadler, D. M. Witten, T. Hastie, R. Tibshirani, and J. O. Ramsay. “On Consistency and Sparsity for Principal Components Analysis in High Dimensions.” In: *Journal of the American Statistical Association* 104.486 (2009), pp. 682–703 (cit. on p. II-8).
- [KT03] J. T. Kwok and I. W. Tsang. “The pre-image problem in kernel methods.” In: *Proceedings of the 20th International Conference on Machine Learning*. 2003, pp. 408–415 (cit. on p. II-2).
- [KW14] D. P. Kingma and M. Welling. “Auto-Encoding Variational Bayes.” In: *International Conference on Learning Representations*. 2014 (cit. on p. II-2).

- [Liu+18] F. Liu, C. Liu, L. Zhao, X. Zhang, X. Wu, X. Xu, Y. Liu, C. Ma, S. Wei, Z. He, J. Li, and E. N. Yin Kwee. “An Open Access Database for Evaluating the Algorithms of Electrocardiogram Rhythm and Morphology Abnormality Detection.” In: *Journal of Medical Imaging and Health Informatics* (2018), pp. 1368–1373 (cit. on p. II-8).
- [Mik+98] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. “Kernel PCA and de-noising in feature spaces.” In: *Advances in neural information processing systems* 11 (1998) (cit. on p. II-2).
- [Ped+11] F. Pedregosa, G. Varoquaux, et al. “Scikit-learn: Machine Learning in Python.” In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on pp. II-2, II-10).
- [Rad+18] A. Radhakrishnan, K. Yang, M. Belkin, and C. Uhler. “Memorization in overparameterized autoencoders.” In: *arXiv preprint arXiv:1810.10333* (2018) (cit. on p. II-10).
- [RMW14] D. J. Rezende, S. Mohamed, and D. Wierstra. “Stochastic backpropagation and approximate inference in deep generative models.” In: *International conference on machine learning*. PMLR. 2014, pp. 1278–1286 (cit. on p. II-2).
- [RR08] A. Rahimi and B. Recht. “Random Features for Large-Scale Kernel Machines.” In: *Advances in Neural Information Processing Systems* 20. 2008, pp. 1177–1184 (cit. on pp. II-4, II-9).
- [Sam+07] R. Sameni, M. B. Shamsollahi, C. Jutten, and G. D. Clifford. “A nonlinear Bayesian filtering framework for ECG denoising.” In: *IEEE Transactions on Biomedical Engineering* 54.12 (2007), pp. 2172–2185 (cit. on p. II-8).
- [SB14] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014 (cit. on p. II-1).
- [SSM97] B. Schölkopf, A. Smola, and K.-R. Müller. “Kernel principal component analysis.” In: *International conference on artificial neural networks*. Springer. 1997, pp. 583–588 (cit. on p. II-1).
- [WS00] C. Williams and M. Seeger. “Using the Nyström method to speed up kernel machines.” In: *Advances in neural information processing systems* 13 (2000) (cit. on p. II-9).
- [Xie+23] C. Xie, L. McCullum, A. Johnson, T. Pollard, B. Gow, and B. Moody. *Waveform Database Software Package (WFDB) for Python*. 2023 (cit. on p. II-8).

- [Xio+16] P. Xiong, H. Wang, M. Liu, S. Zhou, Z. Hou, and X. Liu. “ECG signal enhancement based on improved denoising auto-encoder.” In: *Engineering Applications of Artificial Intelligence* 52 (2016), pp. 194–202 (cit. on p. II-8).
- [Yan+12] T. Yang, Y.-F. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou. “Nyström method vs random Fourier features: A theoretical and empirical comparison.” In: *Advances in neural information processing systems* 25 (2012) (cit. on p. II-9).



## Appendix

### A Additional results on s-curve data set

The s-curve is generated by the following set of equations where the variable  $t$  is often used as a label<sup>3</sup>. For our purposes, we do not require labels but are only concerned with inputs  $\mathbf{x}$ . Additive Gaussian noise  $\mathbf{v} \sim \mathcal{N}(0, \sigma \mathbf{I}_3)$  is added to  $\mathbf{x}$ .

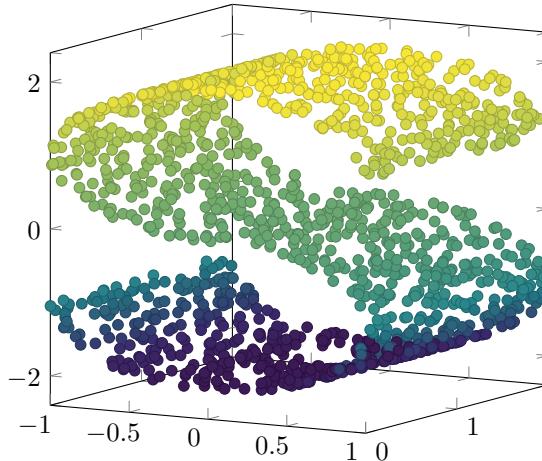
$$t \sim \mathcal{U}\left(-\frac{3}{2}\pi, \frac{3}{2}\pi\right), \quad (9a)$$

$$x_1 = \sin t, \quad (9b)$$

$$x_2 \sim \mathcal{U}(0, 2), \quad (9c)$$

$$x_3 = \text{sign}(t) (\cos t - 1). \quad (9d)$$

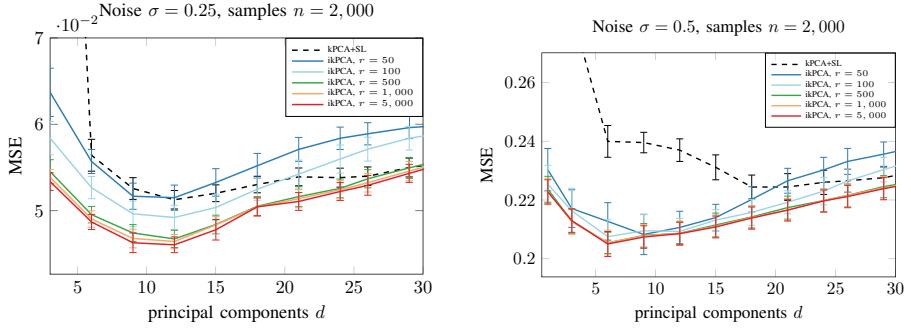
Figure 6 shows a visualisation of the s-curve data set. Figure 7 is an extension of Figure 2 for a larger set of random Fourier features  $r$  and for a second level of additive noise. For this data set, fewer random Fourier features are necessary than training data points. Hence, our method is numerically faster. Already  $r = 500$  features (less than 1/4 of the number of samples  $n = 2,000$ ) are sufficient for the optimal performance curve. We observe that for larger noise values, ikPCA even outperforms kPCA+SL and PCA and that the effect of  $r$  is less pronounced.



**Figure 6:** Visualization of the s-curve data set. The colour indicates the regression label  $t$ .

---

<sup>3</sup>See also [https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make\\_s\\_curve.html](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_s_curve.html)



**Figure 7:** Effect of the number of random features components on reconstruction MSE. Figure 2 is a modified version of the left figure here.

## B Additional results on USPS data set

For the following plots, the hyperparameters for kPCA+SL (i.e. kernel width  $\gamma$  and regularization strength  $\lambda$ ) were selected such that the lowest reconstruction MSE was achieved. A grid search was utilized. For all results mean (and in error plots also standard deviation) over 20 random runs are presented.

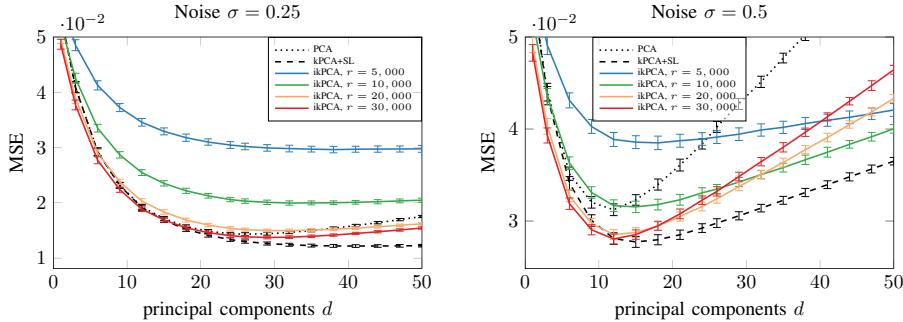
Figure 8 explores the effect of the number of random Fourier features  $r$  for this data set. We observe that generally more random Fourier features yield asymptotically better results. Furthermore, we note that our method ikPCA approaches kPCA+SL for  $r \rightarrow \infty$  as suggested by the approximation of the kernel.

Figure 9 explores the effect of the regularization parameter  $\lambda$  for the reconstruction in our ikPCA method. We observe that an optimal trade-off has to be found. For  $\lambda \rightarrow 0^+$ , ikPCA approaches the performance of PCA. Conversely, for large values of  $\lambda$ , the problem becomes over-regularized and does not generalize anymore.

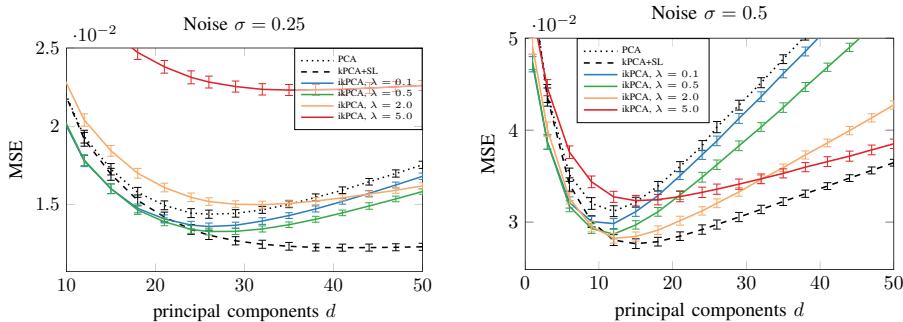
Figure 10 shows the combined effect of the additive noise level and the number of principal components  $d$  chosen for the latent space. The number of components with the lowest MSE for each method is shown in the left plot of Figure 11. We observe that a larger noise value leads to a lower number of optimal principal components  $d$ , which is justified as the noise level dominates a larger portion of singular values. Figure 11 subsequently shows the MSE values of all three methods when choosing the optimal number of principal components  $d$ . We observe that the MSE for optimal tuned methods in this data set is similar for all methods and noise levels.

Figure 12 is a reconstruction of USPS images for two different noise levels when choosing optimal hyperparameters for all methods. As the quantitative

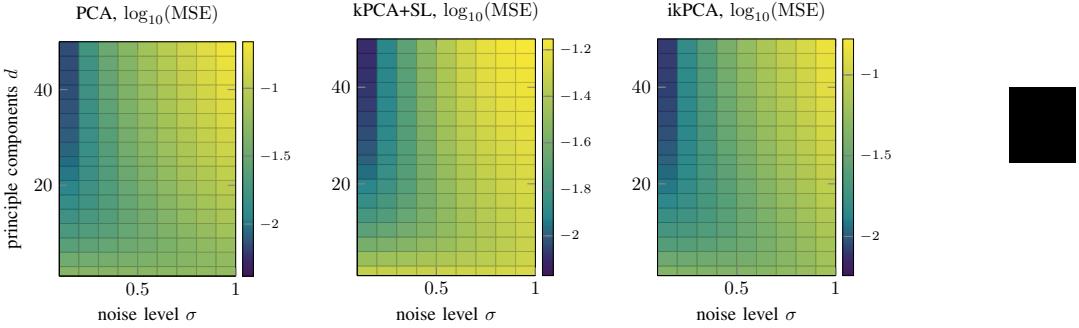
comparison in the right plot of Figure 11 suggests, the reconstructions are also qualitatively similar.



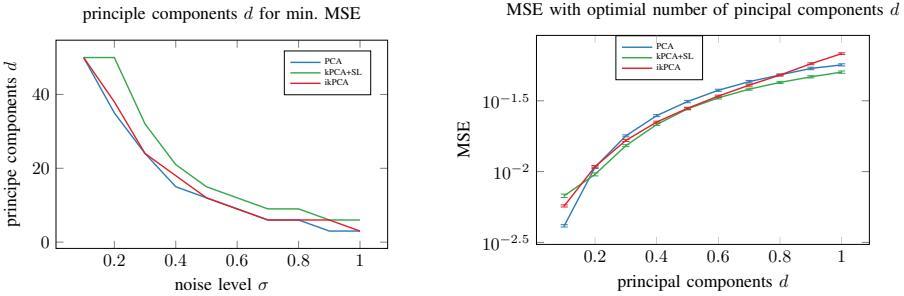
**Figure 8:** Effect of different number of random feature components on reconstruction MSE.



**Figure 9:** Effect of regularization parameter  $\lambda$  on reconstruction MSE. An optimal value has to be chosen. Right plot is a repetition of Figure 3.



**Figure 10:** Combined effect of noise and latent space dimension on reconstruction MSE.

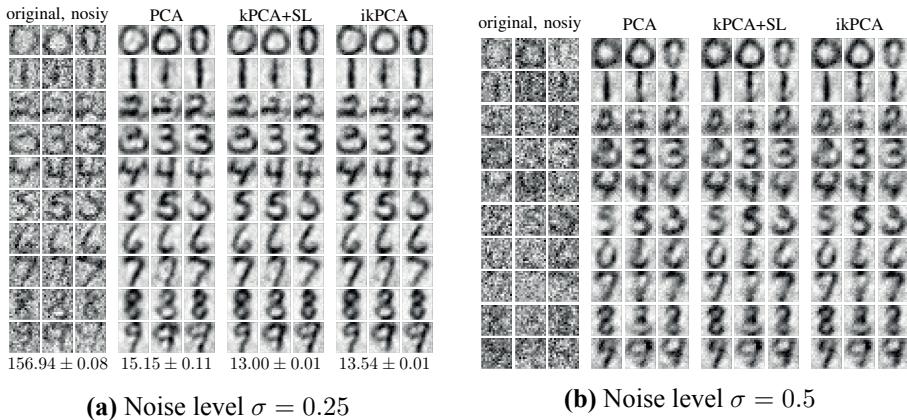


**Figure 11:** Effect of noise. (Left) The best number of components to achieve the lowest MSE for a certain noise level. (Right) MSE of the three methods for reconstruction choosing the optimal number of components.

## C Additional results on ECG data

Figure 13 shows two more examples of reconstructing ECG signals, complementing Figure 5. The same hyperparameters as in the main text are chosen. In the right plot, we can see that for PCA some reconstructions (red dashed lines) are not optimal, i.e. close to the ground truth line. This leads to a significantly higher MSE. Both kPCA+SL and ikPCA perform similarly.

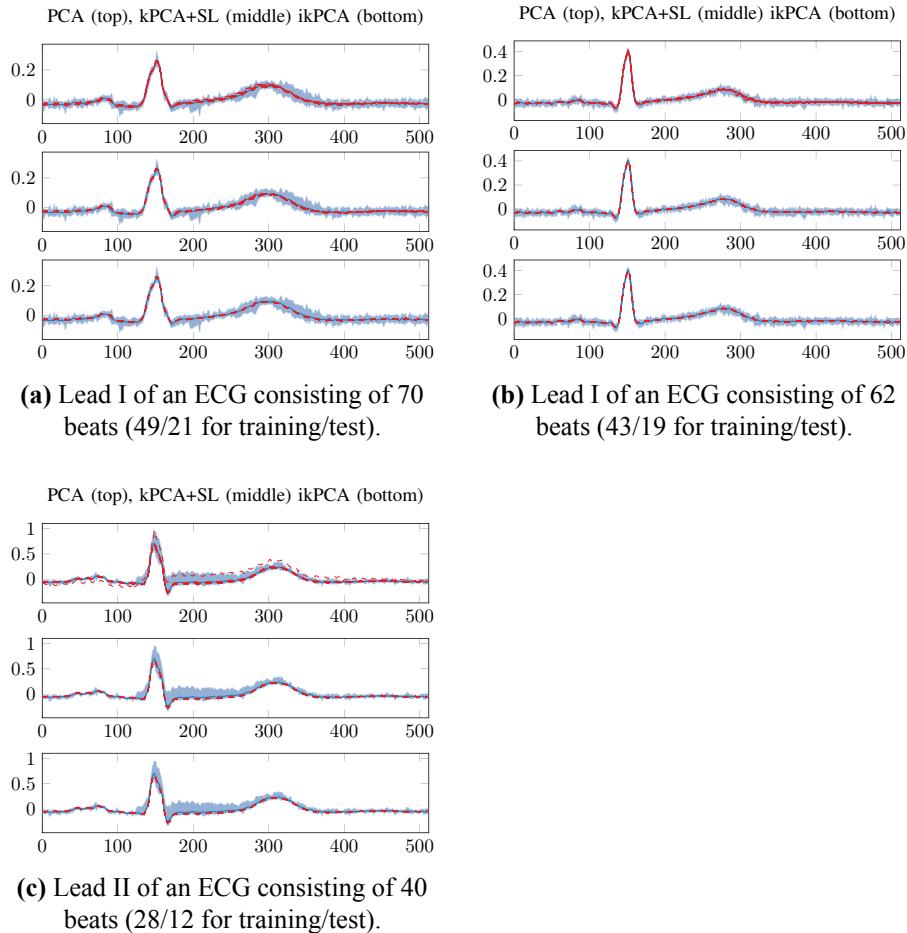
Table 1 compares the MSE values over 500 simulations with different train/test splits for the three ECG traces in Figure 13. We observe that ikPCA and kPCA+SL perform similarly in terms of MSE, while PCA has a slightly higher MSE. Hence, ECG denoising is not as good with a purely linear model.



**Figure 12:** Reconstruction with different methods. Optimal hyperparameters were chosen for each method to achieve the lowest MSE. Left plot is a repetition of Figure 4.

**Table 1:** Reconstruction MSE for different ECGs.

	PCA	kPCA+SL	ikPCA	[unit]
ECG (a)	$4.00 \pm 1.47$	$2.78 \pm 0.74$	$2.57 \pm 0.79$	$[10^{-5}]$
ECG (b)	$3.20 \pm 0.35$	$2.38 \pm 0.29$	$2.32 \pm 0.31$	$[10^{-5}]$
ECG (c)	$8.37 \pm 5.93$	$2.43 \pm 1.63$	$2.27 \pm 1.49$	$[10^{-4}]$



**Figure 13:** More ECG reconstruction results. Figure 13a is the same example as in Figure 5.



# Paper III

**Title**

No Double Descent in Principal Component Regression: A High-Dimensional Analysis

**Authors**

Daniel Gedon, Antônio H. Ribeiro, Thomas B. Schön

**Edited version of**

D. Gedon, A. H. Ribeiro, and T. B. Schön. “No Double Descent in Principal Component Regression: A High-Dimensional Analysis.” Submitted. 2024



# No Double Descent in Principal Component Regression: A High-Dimensional Analysis

## Abstract

Understanding the generalization properties of large-scale models necessitates incorporating realistic data assumptions into the analysis. Therefore, we consider Principal Component Regression (PCR)—combining principal component analysis and linear regression—on data from a low-dimensional manifold. We present an analysis of PCR when the data is sampled from a spiked covariance model, obtaining fundamental asymptotic guarantees for the generalization risk of this model. Our analysis is based on random matrix theory and allows us to provide guarantees for high-dimensional data. We additionally present an analysis of the distribution shift between training and test data. The results allow us to disentangle the effects of (1) the number of parameters, (2) the data-generating model and, (3) model misspecification on the generalization risk. The use of PCR effectively regularizes the model and prevents the interpolation peak of the double descent. Our theoretical findings are empirically validated in simulation, demonstrating their practical relevance.

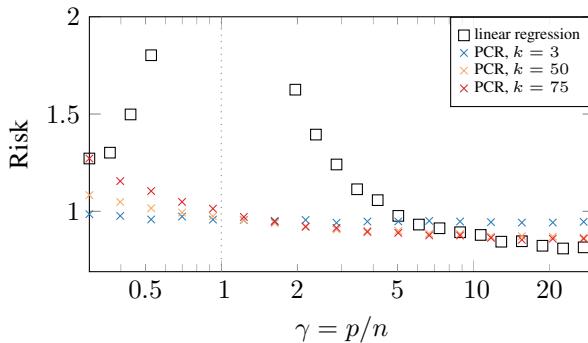
## 1 Introduction

The study of overparameterized models with more features than training data points offers a natural route to gain theoretical understanding when it comes to the successes of large-scale models with good generalization properties [NTS15; Zha+17]. The observation that the generalization error often decreases in the overparameterized regime and the framing as ‘double descent’ [Bel+19] boosted research in this direction even if generalization of large models was already studied before [BM02; DR17; BHM18; ASS20].

Principal Component Regression (PCR) is a widely adopted model where the input data is projected onto the principal components of the data and then a linear model is fitted to the projected data [Pea01; Jol02]. It is a simple but effective model that is used in many real-world applications for its interpretable regularization effect. Examples include exploratory statistical research [Mas65], econometrics [Gew96], genetics [WA08], robotics [VS00]

and many more. These real-world datasets are often high-dimensional but lie on a low-dimensional manifold [TSL00]. In this work, we therefore consider the case where PCR is applied to data which is sampled from a spiked covariance model [Joh01] which is a widely adopted model for high-dimensional data [BAP05; BS06]. The spiked covariance model assumes that the data is sampled from a low-dimensional subspace. Thus, the feature covariance matrix is given by a base covariance  $C_0$  representing noise and some spikes for the  $d$ -dimensional data subspace, yielding  $C = C_0 + \sum_{i=1}^d v_i \lambda_i v_i^\top$ .

While the double descent is well studied for linear regression models [Bar+20; Has+22; MM22], the effect of the double descent for PCR is not well understood. To visualize PCR under high-dimensional inputs for a real-world data example, we use the Diverse MAGIC wheat data set [Sco+21]. Here, we sub-sample the genotypes uniformly for a varying number of features  $p$  while keeping the number of samples  $n$  fixed. Figure 1 shows the risk of PCR and full linear regression on this data set. While in this example there is no reason to assume that the data is sampled from a spiked covariance model which is a linear data generator, we can observe that (1) linear regression has a double descent curve and (2) PCR effectively regularizes the model and for sufficiently many principal components, the risk approaches the linear regression risk for small and large number of features. Therefore, even though our analysis focuses on the linear case, it can describe the qualitative behaviour of PCR on real-world data.



**Figure 1: Risk on real-world data.** PCR and full regression on diverse MAGIC wheat genetics data set.

Our analysis is based on random matrix theory which allows us to provide asymptotic guarantees for high-dimensional data. Random matrix theory is a well-established tool in the analysis of overparameterized models [PW17; Has+22]. We apply its results from the spiked covariance model to the generalization risk of PCR. Therefore, we extend the results of unsupervised feature analysis from PCA to the supervised learning setting of PCR.

Our main contribution is an analysis of the *asymptotic generalization risk* of PCR on data sampled from a spiked covariance model. We show the connection of the risk of PCR to the number of parameters, data assumptions and model misspecification. We further provide an analysis of the *distribution shift* between training and test data, a scenario which is often encountered in practice. Our theoretical findings are empirically *validated through simulation*.

## 2 Background and problem

Throughout the paper we use bold capital letters  $\mathbf{X}$  to denote matrices, bold lower-case letters  $\mathbf{x}$  for vectors, and lower-case letters  $x$  for scalars. The identity matrix of size  $p$  is denoted by  $\mathbf{I}_p$ . Estimated values are denoted by hats, e.g.  $\hat{\mathbf{C}}$  is the estimate of the covariance matrix  $\mathbf{C}$ .

### 2.1 Data generating process.

Let the eigendecomposition of a covariance matrix be  $\mathbf{C} = \mathbf{V}\Lambda\mathbf{V}^\top$ , where  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_p)$  is a matrix of sorted eigenvalues and  $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_p]$  is a matrix of eigenvectors. The singular value decomposition (SVD) of a data matrix  $\mathbf{X} \in \mathbb{R}^{n \times p}$  with  $n$  samples and  $p$  features is denoted by  $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$ , where  $\mathbf{S} = \text{diag}(s_1, \dots, s_p)$  is a matrix of sorted singular values and  $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_p]$  and  $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_p]$  are matrices of left and right singular vectors, respectively.

Take a base covariance  $\mathbf{C}_0$  and a low-rank perturbation covariance  $\mathbf{C}_z$  with  $d$  spiked eigenvalues  $\lambda_1, \dots, \lambda_d$  and corresponding eigenvectors  $\mathbf{v}_1, \dots, \mathbf{v}_d$ .

The *spiked covariance model* is then defined by  $\mathbf{C} = \mathbf{C}_0 + \begin{bmatrix} \mathbf{C}_z & 0 \\ 0 & 0 \end{bmatrix}$  [Joh01].

We assume that the base covariance is  $\mathbf{C}_0 = \mathbf{I}_p$  and the eigenvectors  $\mathbf{v}_i$  are the canonical basis vectors  $\mathbf{e}_i$ . Furthermore, we let the eigenvalues be  $\lambda_i = \exp(-i\alpha)$ , which describes an exponentially decaying spectrum with decay rate  $\alpha \geq 0$ . Our analysis does not require this eigenvalue decay or a specific rate  $\alpha$ . However, we consider this spectrum in our experiments because fast-decaying eigenvalues occur in many real-world examples. As the data-generating model, we consider the latent factor model which connects the spiked covariance model to regression outcomes.

**Definition 1.** The *latent factor model* is the linear model  $\mathbf{x}_i = \mathbf{W}\mathbf{r}_w\mathbf{z}_i + \mathbf{e}_i$ , and  $y_i = \boldsymbol{\theta}^\top \mathbf{z}_i + \varepsilon_i$ . With latent variables  $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_z)$ , feature noise  $\mathbf{e}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)$ , outcome noise  $\varepsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ , feature matrix  $\mathbf{W} \in$

$\mathbb{R}^{p \times d}$  such that  $\mathbf{W}^\top \mathbf{W} = \mathbf{I}_d$ . Let  $r_w^2 = \frac{p}{\text{Tr}(\Lambda_z)} \rho_x$  to control the feature signal-to-noise-ratio (SNR)  $\rho_x = \frac{\mathbb{E}[\|\mathbf{W} r_w \mathbf{z}\|_2^2]}{\mathbb{E}[\|\mathbf{e}\|_2^2]}$ , label noise  $\varepsilon_i \sim \mathcal{N}(\mathbf{0}, \sigma_\varepsilon^2)$  and let  $\boldsymbol{\theta} = 1_d \frac{r_\theta}{\sqrt{\text{Tr}(\Lambda_z)}}$  to control the outcome SNR  $\rho_y = \frac{\mathbb{E}[\|\boldsymbol{\theta}^\top \mathbf{z}\|_2^2]}{\mathbb{E}[\|\varepsilon\|_2^2]} = \frac{r_\theta^2}{\sigma_\varepsilon^2}$ .

This ensures the population covariance follows the spiked covariance as  $\mathbb{E}[\mathbf{x}_i^\top] = \mathbf{C}$ . With this definition  $\mathbf{C} = \Lambda$  as the population eigenvectors are the canonical basis vectors  $\mathbf{V} = \mathbf{I}_p$ . Thus, the feature matrix equals the first  $d$  population eigenvectors  $\mathbf{W} = \mathbf{V}_d$ .

The latent factor model can equivalently be expressed as a linear model directly between features and outcome by  $y_i = \boldsymbol{\beta}^\top \mathbf{x}_i + \nu_i$ . With  $\mathbf{x}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$ ,  $\nu_i \sim \mathcal{N}(0, \sigma_\nu^2)$ ,  $\boldsymbol{\beta} = r_w \mathbf{W} \mathbf{C}_z (\mathbf{I}_d + r_w^2 \mathbf{C}_z)^{-1} \boldsymbol{\theta}$ , and  $\sigma_\nu^2 = \sigma_\varepsilon^2 + \boldsymbol{\theta}^\top (\mathbf{I}_d + r_w^2 \mathbf{C}_z)^{-1} \mathbf{C}_z \boldsymbol{\theta}$ . For details, see Appendix C.1. The latent factor model and equivalent direct model are depicted in Figure 2 (left) as the data generating models.



**Figure 2: Model definitions.** *Left:* Data-generating models: Latent factor model (orange) and equivalent direct model (green). *Right:* Estimation models: PCR (red) and full regression model (blue).

## 2.2 Random matrix theory background

Let  $\hat{\lambda}_i$  denote the eigenvalues of the sample covariance matrix  $\hat{\mathbf{C}} = \frac{1}{n} \mathbf{X}^\top \mathbf{X}$ , and denote the empirical measure of eigenvalues  $\hat{\mu}_p = \frac{1}{p} \sum_{i=1}^p \delta_{\hat{\lambda}_i}$  where  $\delta$  is the Dirac measure. We use results from random matrix theory [Tao23; AGZ10] which allow us to state asymptotic distributions to which the empirical measure converges.

**Marčenko-Pastur.** With  $d = 0$  spikes we get the isotropic case. The empirical measure converges to the Marčenko-Pastur distribution [MP67]. Informally, the Marčenko-Pastur distribution states that the sample eigenvalues concentrate close to the value 1 for small  $\gamma := \frac{p}{n}$  and spread for large  $\gamma$ .

**Theorem 1** (Marčenko and Pastur [MP67]). Let  $n, p \rightarrow \infty$ , such that  $\frac{p}{n} \rightarrow \gamma \in (0, \infty)$  and sample  $\mathbf{x}_i \sim \mathcal{N}(0, \mathbf{I}_p)$ . Then, almost surely the empirical measure  $\hat{\mu}_p$  converges weakly to the Marčenko-Pastur distribution with

density  $f(x)$

$$\begin{cases} f^{MP}(x) = \frac{1}{2\pi\gamma x} \sqrt{(\gamma_+ - x)(x - \gamma_-)}, & \gamma \leq 1 \\ F(dx) = (1 - 1/\gamma)\delta_0(dx) + f^{MP}(x)dx, & \gamma > 1, \end{cases}$$

with  $\delta_0$  as the unit point mass at 0, upper and lower boundaries of  $f^{MP}(x)$  as  $\gamma_{\pm} = (1 \pm \sqrt{\gamma})^2$ .

**Eigenvalue shift.** Let  $d = 1$  spikes, i.e.  $\lambda_1 > 1$ . The distribution of the sample eigenvalue  $\hat{\lambda}_1$  changes with  $\lambda_1$ , transitioning at the critical point  $1 + \sqrt{\gamma}$ . While generally the bulk of the distribution stays Marčenko-Pastur, there are two cases of interest for the spike:

- $\lambda_1 \in [1, 1 + \sqrt{\gamma}]$ : The spike follows limiting Tracy-Widom  $n^{2/3} \frac{\hat{\lambda}_1 - \mu(\gamma)}{\sigma(\gamma)} \xrightarrow{\mathcal{D}} TW_1$ , with  $\mu(\gamma) = (1 + \sqrt{\gamma})^2$  and  $\sigma(\gamma) = (1 + \sqrt{\gamma})^{4/3}\gamma^{-1/6}$  [Joh01; BAP05; BV13].
- $\lambda_1 > 1 + \sqrt{\gamma}$ : The spike follows a Normal  $n^{1/2} \frac{\hat{\lambda}_1 - \mu(\lambda, \gamma)}{\sigma(\lambda, \gamma)} \xrightarrow{\mathcal{D}} \mathcal{N}(0, 1)$ , with  $\mu(\lambda, \gamma) = \gamma \frac{\lambda}{\lambda-1} + \lambda$  and  $\sigma^2(\lambda, \gamma) = 2\lambda^2(1 - \frac{\gamma}{(\lambda-1)^2})$  [BS06; Pau07; YJ18].

This highlights an upward shift of the spike sample distribution mean. Hence, the sample eigenvalue  $\hat{\lambda}_1$  will separate from the bulk of the Marčenko-Pastur distribution for  $\lambda_1 > 1 + \sqrt{\gamma}$ . See Figure 3 for an illustrative example.

This is extendable to  $d > 1$  with a spike multiplicity of one [BAP05; Pau07]. For  $d > 1$  spikes with multiplicity one, we assume that the sample eigenvalues will be distributed according to the spiked covariance model distribution with a bulk of Marčenko-Pastur and  $d$  normally distributed spikes. Let the fraction of population eigenvalues above the critical point be  $\frac{d}{p} \rightarrow \phi \in (0, 1)$ , then the mass of the distribution related to the spike is given by  $\phi$ .

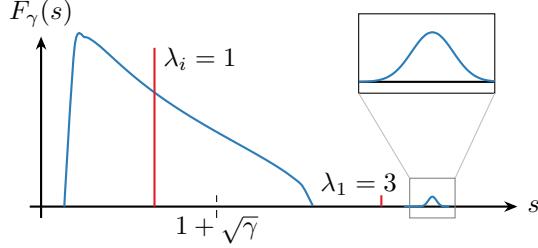
**Stieltjes transform.** Following Anderson et al. [AGZ10] we define the Stieltjes transform of a measure.

**Definition 2** (Stieltjes [Sti94]). Let  $\mu$  be a positive, finite measure on  $\mathbb{R}$ , then the *Stieltjes transform* of the measure is given by  $\varphi_\mu(z) := \int_{\mathbb{R}} \frac{\mu(d\sigma)}{\sigma - z}$  with  $z \in \mathbb{C} \setminus \mathbb{R}_+$ .

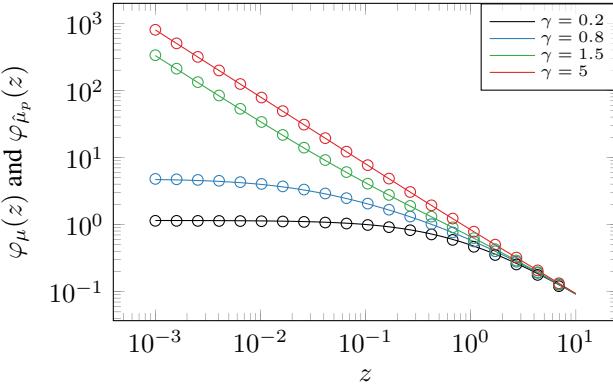
The utility of the Stieltjes transform is that if for a sequence of measures  $\{\mu_1, \mu_2, \dots\}$  the Stieltjes transform converges pointwise  $\varphi_{\mu_p} \rightarrow \varphi_\mu$ , then  $\mu_p \rightarrow \mu$  weakly, see Anderson et al. [AGZ10]. Moreover, the Stieltjes transform of the empirical measure  $\widehat{\mu}_p$  is

$$\varphi_{\widehat{\mu}_p}(z) = \frac{1}{p} \text{Tr}[(\hat{\mathbf{C}} - z\mathbf{I}_p)^{-1}]. \quad (1)$$

The Stieltjes transform can be used to prove different limiting distributions for the empirical measure [BS10] or Bach [Bac23] for an application-oriented presentation of these results. Here, we provide numerical verification for the results above of the eigenvalue shift but with  $d > 1$ : In Figure 4 we illustrate that  $\varphi_{\hat{\mu}_p}(z)$  and the Stieltjes transform of the distribution it converges to  $\varphi_\mu(z)$  are close for large values of  $p$ .



**Figure 3: Eigenvalue spectrum for spiked covariance model.** Sample distribution of  $d = 1$  spike (blue) of population eigenvalues (red,  $\lambda_1 = 3$ , others  $\lambda_i = 1$ ) for  $\gamma = 0.3$ ,  $n = 500$ . The spike has a normally distributed sample eigenvalue with  $\mu = 3.45$ .



**Figure 4: Stieltjes transform  $\varphi(z)$  of eigenvalue distribution.** Solid lines denote numerical evaluation of the Stieltjes transform Definition 2; ‘ $\circ$ ’ markers denote the empirical transform (1). We use  $d = 10$  spikes here.

**Eigenvector shift.** In the spiked covariance model, the top- $d$  eigenvectors can be inconsistent. Let  $\mathbf{v}_i \in \mathbb{R}^p$  be unit population eigenvectors with sample eigenvectors  $\hat{\mathbf{v}}_i \in \mathbb{R}^p$ . As  $p/n \rightarrow \gamma$ ,

$$(\mathbf{v}_i^\top \hat{\mathbf{v}}_i)^2 \rightarrow \begin{cases} \frac{1-\gamma/(\lambda_i-1)^2}{1+\gamma/(\lambda_i+1)} & \text{for } \lambda_i > 1 + \sqrt{\gamma} \\ 0 & \text{for } \lambda_i \in [1, 1 + \sqrt{\gamma}] \end{cases} \quad (2)$$

This implies that the angle between population  $\mathbf{v}_i$  and sample eigenvectors  $\hat{\mathbf{v}}_i$  grows as the spike eigenvalue  $\lambda_i$  decreases. Eigenvectors below the crit-

ical point resemble isotropic, randomly placed vectors within a hypersphere [Pau07; Joh+09]. An overview of high-dimensional PCA is given in Johnstone and Paul [JP18].

## 2.3 Problem formulation

As the data generator, we use the latent factor model and as the estimation model, we use PCR, both of which are visualized in Figure 2. As a baseline estimation model, we choose the unregularized full regression model. We consider the case where the number of spikes  $d$  is fixed but unknown. This implies that the signal is concentrated on a low-dimensional (linear) manifold. With fixed  $d$ , we analyse the effect of the number of features by presenting results for the risk in the low-dimensional  $\gamma < 1$  and high-dimensional  $\gamma > 1$  regime.

**PCR model.** Let  $\hat{C}$  be the sample covariance matrix and  $\hat{\mathbf{V}} = [\hat{\mathbf{v}}_1, \dots, \hat{\mathbf{v}}_k]$  be the matrix of sample eigenvectors truncated to the first  $k \leq p$  principal components. The *PCR model* is defined as the linear model  $\hat{y}_i = \hat{\boldsymbol{\theta}}^\top \hat{\mathbf{z}}_i$  with  $\hat{\boldsymbol{\theta}} = (\hat{\mathbf{Z}}^\top \hat{\mathbf{Z}})^+ \hat{\mathbf{Z}}^\top \mathbf{y}$  and  $\hat{\mathbf{Z}} = \mathbf{X} \hat{\mathbf{V}}$ .

**Full regression model.** Let the linear model  $\hat{y}_i = \hat{\boldsymbol{\beta}}^\top \mathbf{x}_i$  with  $\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^+ \mathbf{X}^\top \mathbf{y}$  be the unregularized least squares estimator. We denote this as the *full regression model*<sup>1</sup> to contrast it with the PCR model. Note that the full regression model is not low-rank. Furthermore, it is a special case of PCR with  $k = p$  and orthogonal features.

**Risk.** Let  $\hat{\boldsymbol{\theta}}$  be a parameter estimator. Then, the *risk* is the mean squared error of the predictions  $R(\hat{\boldsymbol{\theta}}) = \mathbb{E}_{(\mathbf{x}_0, y_0) \sim \mathcal{D}} [(y_0 - \hat{y}_0(\mathbf{x}_0))^2]$ , with data distribution  $\mathcal{D}$ .

## 3 Analysis of in-distribution risk

### 3.1 Risk of PCR

The PCR model jointly estimates SVD components and parameters  $\boldsymbol{\theta}$ . We first choose the number of principal components  $k$  for truncation in the SVD and then estimate the parameters  $\hat{\boldsymbol{\theta}}$ . Hence, let  $\hat{\mathbf{S}}$ ,  $\hat{\mathbf{U}}$ ,  $\hat{\mathbf{V}}$  be the first  $k$  singular values and left, right singular vectors, respectively, then the least squares estimator of  $\hat{\boldsymbol{\theta}}$  on the latent space  $\mathbf{Z}$  projection is given by

$$\hat{\boldsymbol{\theta}} = \hat{\mathbf{V}}^\top \boldsymbol{\beta} + \hat{\mathbf{S}}^{-1} \hat{\mathbf{U}}^\top \boldsymbol{\nu}. \quad (3)$$

---

<sup>1</sup>We use  $\boldsymbol{\beta}$  when modelling the interaction between  $\mathbf{x}$  and  $y$  directly and we use  $\boldsymbol{\theta}$  when modelling the interaction of a latent variable with the outcome, e.g.  $\mathbf{z}$  and  $y$ .

To compute the expected risk of PCR, we let  $\boldsymbol{\Pi} = \mathbf{I}_p - \hat{\mathbf{V}}\hat{\mathbf{V}}^\top$  be the projection matrix onto the subspace orthogonal to the principal components of  $\hat{\mathbf{C}}$ . Then, we obtain

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\nu}} [R(\hat{\boldsymbol{\theta}})] &= \boldsymbol{\beta}^\top \boldsymbol{\Pi} \mathbf{C} \boldsymbol{\Pi} \boldsymbol{\beta} \\ &\quad + \frac{\sigma_\nu^2}{n} \text{Tr} (\hat{\mathbf{V}}^\top \mathbf{C} \hat{\mathbf{V}} \hat{\mathbf{V}}^\top \hat{\mathbf{C}}^{-1} \hat{\mathbf{V}}) + \sigma_\nu^2.\end{aligned}\tag{4}$$

The terms can be interpreted as squared bias, variance and irreducible error. The result highlights that the variance term contains a projection of the covariance onto its estimated  $k$ -dimensional subspace.

For asymptotic results, we generalize the eigenvector shift in the spiked covariance model from the expression  $(\mathbf{v}_i^\top \hat{\mathbf{v}}_i)^2$  in (2) to products of eigenvectors  $(\mathbf{V}^\top \hat{\mathbf{V}})^2$  where we have cross products with  $i \neq j$ . Let  $\hat{\mathbf{V}} = [\hat{\mathbf{v}}_1, \dots, \hat{\mathbf{v}}_k]$  and  $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_k]$  be the estimated and true eigenvectors truncated to the first  $k \leq p$  principal components, respectively. Define  $\hat{\mathbf{V}}_d \in \mathbb{R}^{d \times k}$  and  $\mathbf{V}_{d,d} \in \mathbb{R}^{d \times d}$  as the matrices where the eigenvectors are truncated to the first  $d$  eigenvector elements. Then, the eigenvector shift product is

$$\mathbf{P}_k = \begin{cases} \text{diag}((\mathbf{v}_1^\top \hat{\mathbf{v}}_1)^2, \dots, (\mathbf{v}_k^\top \hat{\mathbf{v}}_k)^2, 0, \dots, 0), & k < d, \\ \text{diag}((\mathbf{v}_1^\top \hat{\mathbf{v}}_1)^2, \dots, (\mathbf{v}_d^\top \hat{\mathbf{v}}_d)^2), & k = d, \\ \text{diag}((\mathbf{v}_1^\top \hat{\mathbf{v}}_1)^2 + c_1^2, \dots, (\mathbf{v}_d^\top \hat{\mathbf{v}}_d)^2 + c_d^2), & k > d, \end{cases}\tag{5}$$

with the correction factor  $c_i^2 = (k-d) \frac{1-(\mathbf{v}_i^\top \hat{\mathbf{v}}_i)^2}{p-d}$  for  $k > d$ . Note that  $\mathbf{P}_k$  is a diagonal matrix with entries depending on the choice of  $k$ . The proofs for (3)-(5) and all following theorems are in Appendix C.

**Asymptotic PCR risk.** With the generalized eigenvector shift, we can find asymptotic expressions for the risk (4). The *asymptotic squared bias* term is given by

$$\text{Bias}_\gamma(\hat{\boldsymbol{\theta}})^2 = \bar{\boldsymbol{\beta}}^\top (\boldsymbol{\Lambda}_d - \boldsymbol{\Lambda}_d \mathbf{P}_k - \mathbf{P}_k \boldsymbol{\Lambda}_d + \mathbf{P}_k + \mathbf{P}_k r_w^2 \mathbf{C}_z \mathbf{P}_k) \bar{\boldsymbol{\beta}}\tag{6}$$

with  $\bar{\boldsymbol{\beta}} = \mathbf{W}^{-1} \boldsymbol{\beta} = r_w \mathbf{C}_z (\mathbf{I}_d + r_w^2 \mathbf{C}_z)^{-1} \boldsymbol{\theta}$  and  $\boldsymbol{\Lambda}_d \in \mathbb{R}^{d \times d}$  as truncation of the population eigenvalue matrix  $\boldsymbol{\Lambda}$  to the first  $d$  dimensions. The *asymptotic variance* term is

$$\begin{aligned}\text{Var}_\gamma(\hat{\boldsymbol{\theta}}) &= \frac{\sigma_\nu^2}{n} \left( \text{Tr} \left[ (\mathbf{P}_k r_w^2 \mathbf{C}_z + \mathbf{I}_k) \frac{1}{\mu(\boldsymbol{\Lambda}, \gamma)} \right] \right. \\ &\quad \left. + (p-d) \int_{s_c}^{(1+\sqrt{\gamma})^2} \frac{1}{s} dF_\gamma(s) \right)\end{aligned}\tag{7}$$

with  $\mu(\boldsymbol{\Lambda}, \gamma)$  as diagonal matrix with entries  $\mu(\lambda_i, \gamma)$  as mean of the spike eigenvalue distribution,  $F_\gamma$  as the Marčenko-Pastur distribution and  $s_c$  the value in  $\mathbb{R}$  which satisfies  $\max \left( \frac{k-d}{p-d}, 0 \right) = \int_{s_c}^{(1+\sqrt{\gamma})^2} dF_\gamma(s)$ .

**Theorem 2** (Asymptotic PCR risk). In the asymptotic limit  $n, p \rightarrow \infty$ , such that  $\frac{p}{n} \rightarrow \gamma \in (0, \infty)$ , the expected risk of PCR will converge almost surely to

$$\mathbb{E}_{\nu} [R(\hat{\theta})] \rightarrow \text{Bias}_{\gamma}(\hat{\theta})^2 + \text{Var}_{\gamma}(\hat{\theta}) + \sigma_{\nu}^2.$$

The theorem implies that the squared bias is a scaled version of the  $d$ -dimensional subspace of the eigenvalues  $\Lambda$ . Moreover, in the variance, we see that for  $k \leq d$   $s_c = (1 + \sqrt{\gamma})^2$  meaning that the integral term is zero. Hence, the variance is a scaled version of the inverse mean of the spike eigenvalue distribution  $\mu(\lambda_i, \gamma)$ . The results hold for eigenvalues below and above the phase transition threshold.

### 3.2 Risk of baseline methods

As a reference, we state the null predictor, i.e.  $\hat{\theta} = 0$ . In this case, the expected null risk becomes  $\mathbb{E}_{\nu} [R(\hat{\theta})] = \beta^T C \beta + \sigma_{\nu}^2$ , which has zero variance and contains an unprojected squared bias term. Let us restate the full regression risk from [Has+22, Lemma 1], which is a special case of (4) for  $k = p$

$$\mathbb{E}_{\nu} [R(\hat{\beta})] = \beta^T \Pi C \Pi \beta + \frac{\sigma_{\nu}^2}{n} \text{Tr}(C \hat{C}^{-1}) + \sigma_{\nu}^2. \quad (8)$$

Below we have the asymptotic result for full regression with the spiked covariance model. Here, the bias is zero for  $\gamma < 1$  while the variance term remains unchanged from the asymptotic PCR result. This result is a special case of the PCR result in Theorem 2 for  $k = p$ .

**Theorem 3** (Asymptotic full regression risk). In the asymptotic limit  $n, p \rightarrow \infty$ , such that  $\frac{p}{n} \rightarrow \gamma \in (0, \infty)$ , the expected risk of the full regression model will converge almost surely to

$$\mathbb{E}_{\nu} [R(\hat{\beta})] \rightarrow \text{Bias}_{\gamma}(\hat{\beta})^2 + \text{Var}_{\gamma}(\hat{\beta}) + \sigma_{\nu}^2$$

with the asymptotic squared bias term as  $\text{Bias}_{\gamma}(\hat{\beta})^2 = 0$  for  $\gamma < 1$  and  $\text{Bias}_{\gamma}(\hat{\beta})^2$  as in Theorem 2 with  $k = p$  for  $\gamma \geq 1$ ; and the variance term  $\text{Var}_{\gamma}(\hat{\beta})$  equal to the definition of the variance in Theorem 2 with  $k = p$ .

## 4 Analysis of covariate-shifted risk

In this section, we change the data generator from Definition 1 to one inspired by Emami et al. [Ema+20] to include covariate shift. Specifically, we introduce a shift in the latent factors between training and test time. Let us assume that the eigenvectors  $\mathbf{V}$  of the training or source covariance  $\mathbf{C}_S = \mathbf{V}\Lambda_S\mathbf{V}^\top$  and test covariance  $\mathbf{C}_T = \mathbf{V}\Lambda_T\mathbf{V}^\top$  are the same but the eigenvalues  $\Lambda_S, \Lambda_T$  differ. This covariate shift relates to scenarios where the underlying structure remains consistent, represented by the unchanged eigenvectors. The variations in eigenvalues reflect real-world situations where the magnitude of certain factors changes without altering their relationship.

**Definition 3.** Let the covariance matrices of the latent factors be  $\mathbf{C}_{z,S} = \text{diag}(\lambda_{1,S}, \dots, \lambda_{d,S})$  and  $\mathbf{C}_{z,T} = \text{diag}(\lambda_{1,T}, \dots, \lambda_{d,T})$  for the training and test data, respectively. Then, we have  $\mathbf{z}_S \sim \mathcal{N}(0, \mathbf{C}_{z,S})$ , and  $\mathbf{x}_{i,S} = \mathbf{W}_w \mathbf{z}_{i,S} + \mathbf{e}_{i,S}$ , and  $y_{i,S} = \boldsymbol{\theta}^\top \mathbf{z}_{i,S} + \varepsilon_{i,S}$  for the training data and similarly for the test data. The spike eigenvalues are sampled i.i.d. from a zero-mean Normal distribution with  $\text{Cov}(u_S, u_T) = \sigma_\ell^2 \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$  such that  $\lambda_{S,i} = \exp(\alpha u_{S,i})$ ,  $\lambda_{T,i} = \exp(\alpha u_{T,i})$ .

Note that, we assume the number of spikes  $d$  is equal. By choosing  $\sigma_\ell^2, \rho$ , we can control the covariate shift between the training and test data. Specifically, there are two scenarios without covariate shifts. First, with  $\sigma_\ell^2 = 0$ , we have  $\lambda_{i,S} = \lambda_{i,T} = 1$ . Second, with  $\sigma_\ell^2 > 0$  but  $\rho = 1$ , we have  $\lambda_{i,S} = \lambda_{i,T} \neq 1$ . Finally, we can introduce covariate shift with  $\sigma_\ell^2 > 0$  and  $\rho \in ]0, 1[$  which defines correlated but different latent factors.

### 4.1 Risk of PCR

We use the parameter estimator (3) but write it as  $\hat{\boldsymbol{\theta}} = \hat{\mathbf{V}}^\top \boldsymbol{\beta} + \hat{\mathbf{S}}_S^{-1} \hat{\mathbf{U}}^\top \boldsymbol{\nu}$ , with  $\hat{\mathbf{S}}_S$  as the training data singular values to highlight the explicit dependence on the training data. Also, we update the definition for the test risk under covariate shift as  $R(\hat{\boldsymbol{\theta}}) = \mathbb{E}_{(\mathbf{x}_0, y_0) \sim \mathcal{D}_T} [(y_0 - \hat{y}_0(\mathbf{x}_0))^2]$  with  $\mathcal{D}_T$  as the test data distribution.

We can generalize the PCR risk from (4) under covariate shift as in Definition 3. Let the projection matrix  $\Phi = \hat{\mathbf{V}} \hat{\mathbf{V}}^\top$  onto the subspace spanned by the first  $k$  principal components of the training data. Then, the expected risk of PCR

under covariate shift is given by

$$\begin{aligned}\mathbb{E}_{\nu_T} [R(\hat{\theta})] &= (\beta_T - \Phi\beta)^\top C_T (\beta_T - \Phi\beta) \\ &\quad + \frac{\sigma_\nu^2}{n} \text{Tr} \left( \hat{V}^\top C_T \hat{V} \hat{V}^\top \hat{C}_S^{-1} \hat{V} \right) + \sigma_T^2,\end{aligned}\tag{9}$$

with  $\beta_T = r_w \mathbf{W} \mathbf{C}_{z,T} (\mathbf{I}_d + r_w^2 \mathbf{C}_{z,T})^{-1} \boldsymbol{\theta}$  and  $\sigma_T^2 = \sigma_\varepsilon^2 + \boldsymbol{\theta}^\top (\mathbf{I}_d + r_w^2 \mathbf{C}_{z,T})^{-1} \mathbf{C}_{z,T} \boldsymbol{\theta}$ .

This result resembles (4) but also shows that the introduction of covariate shift complicates the analysis because we have to deal with quantities from training and test distribution. Therefore, when inspecting the asymptotic result as stated below, we have to treat factors in the squared bias term individually according to the contribution from training and test data.

**Covariate-shifted asymptotic PCR risk.** For asymptotic expressions of (9), we can define the *asymptotic squared bias* term as

$$\text{Bias}_{\gamma,T}(\hat{\theta})^2 = [\bar{\beta}_T^\top \quad \bar{\beta}^\top] \begin{bmatrix} \Lambda_{d,T} & -\Lambda_{d,T} \mathbf{P}_k \\ -\mathbf{P}_k \Lambda_{d,T} & \mathbf{P}_k + \mathbf{P}_k r_w^2 \mathbf{C}_{z,T} \mathbf{P}_k \end{bmatrix} \begin{bmatrix} \bar{\beta}_T \\ \bar{\beta} \end{bmatrix} \tag{10}$$

with  $\bar{\beta} = \mathbf{W}^{-1} \beta$ ,  $\bar{\beta}_T = \mathbf{W}^{-1} \beta_T$  and  $\Lambda_{d,T} \in \mathbb{R}^{d \times d}$  as the truncation of  $\Lambda_T$  to the first  $d$  dimensions. The *asymptotic variance* term is

$$\begin{aligned}\text{Var}_{\gamma,T}(\hat{\theta}) &= \frac{\sigma_\nu^2}{n} \left( \text{Tr} \left[ (\mathbf{P}_k r_w^2 \mathbf{C}_{z,T} + \mathbf{I}_k) \frac{1}{\mu(\Lambda, \gamma)} \right] \right. \\ &\quad \left. + (p-d) \int_{s_c}^{(1+\sqrt{\gamma})^2} \frac{1}{s} dF_\gamma(s) \right)\end{aligned}\tag{11}$$

with  $\mu(\Lambda, \gamma)$ ,  $F_\gamma$  and  $s_c$  as described in the non-covariate shifted variance term (7). The main difference in the variance for covariate shifts lies in the term  $\mathbf{C}_{z,T}$  which is the shifted covariance of the spikes that is used here instead of the covariance from the training data.

**Theorem 4** (Covariate-shifted asymptotic PCR risk). In the asymptotic limit  $n, p \rightarrow \infty$ , such that  $\frac{p}{n} \rightarrow \gamma \in (0, \infty)$ , the expected risk of PCR under covariate shift will converge almost surely to

$$\mathbb{E}_{\nu_T} [R(\hat{\theta})] \rightarrow \text{Bias}_{\gamma,T}(\hat{\theta})^2 + \text{Var}_{\gamma,T}(\hat{\theta}) + \sigma_T^2.$$

## 4.2 Risk of baseline methods

Similar to Section 3.2, the results of the baseline models are special cases of the main PCR results with  $k = p$ . However, due to the covariate shift the squared bias term will not diminish in the asymptotic case.

Let us start with the null predictor as a reference. For  $\hat{\boldsymbol{\theta}} = 0$  the expected null risk under covariate shift becomes  $\mathbb{E}_{\nu_T} [R(\hat{\boldsymbol{\theta}})] = \boldsymbol{\beta}_T^\top \mathbf{C}_T \boldsymbol{\beta}_T + \sigma_T^2$ . As a generalization of the expected risk of full regression from (8), we obtain under covariate shift

$$\begin{aligned}\mathbb{E}_{\nu} [R(\hat{\boldsymbol{\beta}})] &= (\boldsymbol{\beta}_T - \hat{\mathbf{V}} \hat{\mathbf{V}}^\top \boldsymbol{\beta})^\top \mathbf{C}_T (\boldsymbol{\beta}_T - \boldsymbol{\beta}) \\ &\quad + \frac{\sigma_\nu^2}{n} \text{Tr}(\mathbf{C}_T \hat{\mathbf{C}}^{-1}) + \sigma_T^2.\end{aligned}\tag{12}$$

Finally, we can extend the asymptotic results from Theorem 3 to obtain the following result under covariate shift.

**Theorem 5** (Covariate-shifted asymptotic full regression risk). In the asymptotic limit  $n, p \rightarrow \infty$ , such that  $\frac{p}{n} \rightarrow \gamma \in (0, \infty)$ , the expected risk of the full regression model under covariate shift will converge almost surely to

$$\mathbb{E}_{\nu} [R(\hat{\boldsymbol{\beta}})] \rightarrow \text{Bias}_{\gamma, T}(\hat{\boldsymbol{\beta}})^2 + \text{Var}_{\gamma, T}(\hat{\boldsymbol{\beta}}) + \sigma_T^2$$

with the asymptotic squared bias term as

$$\text{Bias}_{\gamma, T}(\hat{\boldsymbol{\beta}})^2 = (\bar{\boldsymbol{\beta}}_T - \bar{\boldsymbol{\beta}})^\top \boldsymbol{\Lambda}_{d, T} (\bar{\boldsymbol{\beta}}_T - \bar{\boldsymbol{\beta}})$$

for  $\gamma < 1$  and as in Theorem 4 for  $\gamma \geq 1$  with  $k = p$ . The variance term  $\text{Var}_{\gamma}(\hat{\boldsymbol{\beta}})$  is equal to the definition of the variance in Theorem 4 with  $k = p$ .

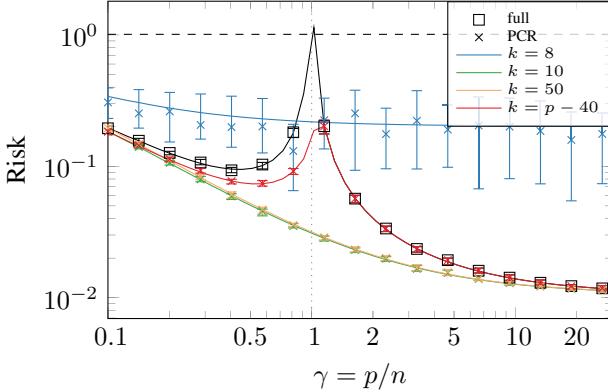
## 5 Numerical results

### 5.1 Simulation of in-distribution data

**Setup.** For the data-generating process, we choose the parameters  $\alpha = 0.1$ ,  $\sigma_\varepsilon = 0.1$ ,  $r_\theta = 1$  and  $\rho_x = 1$ . While we select one set of parameters for visualization, we have rigorously validated our results across numerous combinations. We choose  $d = 10$  spikes and vary  $k$  to see the effect of model misspecification. For our simulations, we choose  $n = 500$  and set  $p$  accordingly to fulfill  $\gamma = \frac{p}{n}$ . We vary  $\gamma \in [0.1, 30]$ , i.e. from low-dimensional  $\gamma < 1$  to high-dimensional  $\gamma > 1$ . We compute the risk  $\mathbb{E}_{\nu} [R(\boldsymbol{\theta})]$  and present median values of the simulation results from 50 realizations as well as 25%, 75% quantiles.

**Main result.** The main result for the risk analysis is in Figure 5. We can observe the following: (1) Theory and simulation align well and therefore support

our analysis. (2) For misspecified models with  $k < d$  we remove important directions and therefore the risk rises heavily; misspecifications with  $k \gg d$  diminish the regularizing effect of PCR by including many noise directions and therefore get close to the full regression solution. (3) For appropriately chosen  $k^2$  PCR does not suffer from the interpolation peak and therefore shows an optimal regularizing effect. (4) For  $k \geq d$ , the PCR model matches the full regression solution in the limit of small and large  $\gamma$ .



**Figure 5: Risk on in-distribution data: Simulation vs. analysis.** The different marks denote finite sample risk; solid lines denote analytical results. Null risk is given by the dashed line.

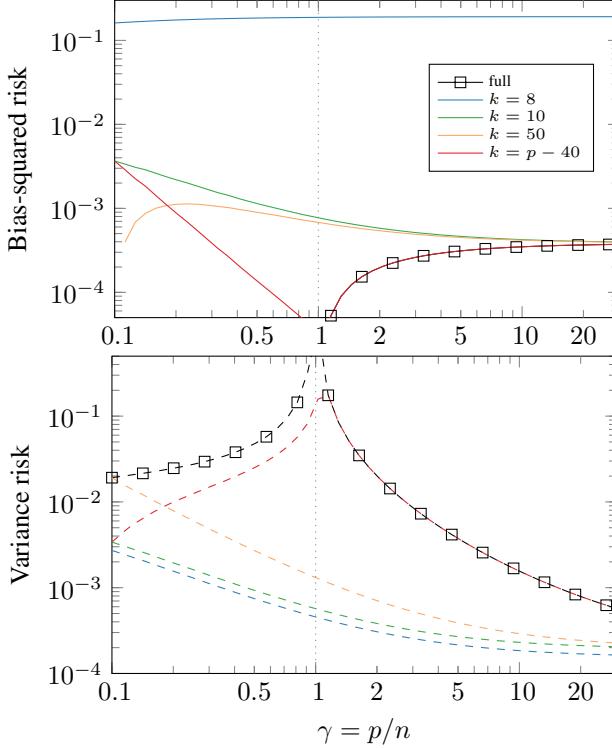
**Bias-variance decomposition.** We split the risk according to Theorem 2 and 3 in bias and variance terms to analyse them independently. The results are shown in Figure 6 where we can observe that the bias term is dominant for  $k < d$ . According to Theorem 2 for  $k < d$  the least amount of risk is subtracted because  $P_k$  contains many zeros. For appropriately chosen  $k$ , the bias decreases with large  $\gamma$ . The variance term increases with larger  $k$  as supported by Theorem 2 because more noise directions are considered.

## 5.2 Simulation of covariate-shifted data

**Setup.** We choose the same setup as for in-distribution data with  $\alpha = 1$ . For the eigenvalue strength, we have to specify  $\sigma_\ell^2$  and  $\rho$  to control the correlation between training and test data. We focus on the effect of correlation between training and test data through the choice of  $\rho_\ell$ .

**Main result.** In Figure 7 we present results for low correlated covariate eigenvalues with  $\rho_\ell = 0.1$  and highly correlated eigenvalues with  $\rho_\ell = 0.9$ . We

<sup>2</sup>I.e.  $k = d + \Delta$  with  $\Delta$  a small non-negative integer such that a limited number of noise directions are considered.

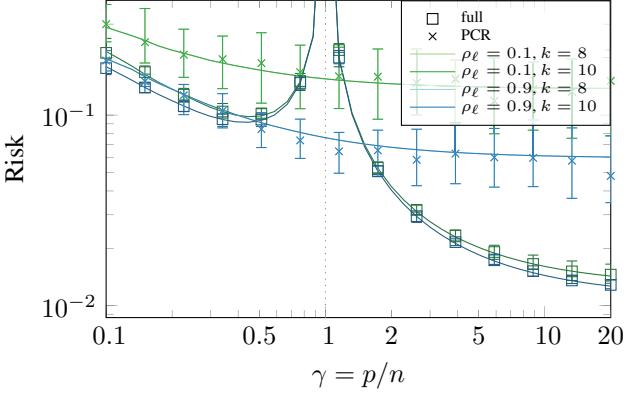


**Figure 6: Bias-variance decomposition of in-distribution risk: analysis.** Squared bias (top) and variance (bottom).

notice both scenarios have qualitatively similar behaviour which follows the observations from the in-distribution results in Figure 5. The main difference is that highly correlating train and test data leads to lower risk. Since test data in this scenario is generated roughly from the same distribution as the training data, the model is less misspecified which leads to lower risk.

## 6 Related work

**Overparameterization.** The ‘double descent’ phenomenon [Bel+19] for the generalization curve of overparameterized models was already discovered and analysed in early works [KH91; GBD92; Opp95]. While it has been observed in deep state-of-the-art models [dAs+20; Nak+21], most theoretical studies focus on simple models. Examples are found for linear regression [Bar+20; Mut+20; Has+22], ensembles [LJB20; Lou+21], classification [Ger+20; WMT21; DKT22], random features [Bel+19; MM22] or small neural networks trained using gradient descent [ASS20; FCB22; Cao+22]. Practical



**Figure 7: Risk on covariate-shifted data: Simulation vs. analysis.** Different marks denote finite sample risk; solid lines denote analytical results. In green tones are results for low-correlated features  $\rho_\ell = 0.1$ ; and in blue tones for high-correlated features  $\rho_\ell = 0.9$ .

model regularizers such as Ridge are analysed [TB23], and we extend the analysis to the commonly used PCR model. We broaden our analysis to distribution shifts, a relatively underexplored area in the context of overparameterization [TAP21a; TAP21b; Ema+20].

**PCR analysis.** Using PCA [Pea01; Jol02] is common—discussions focus on the choice of principle components [BF83] or high-dimensional data [LLP12]. Since PCA acts on the eigenvectors of the covariance matrix, it can be viewed as a spectral regularizer. Finite sample risk bounds were analysed for general spectral regularization including PCR [BPR07; DFH17] and adaptive PCR under the same latent factor model as ours but using concentration inequalities [Bin+21]. Hucker and Wahl [HW23] provide high probability bounds for the PCR risk. Furthermore, PCR is investigated in Xu and Hsu [XH19] for general but fully known covariances  $C$  in the asymptotic regime. Wu and Xu [WX20] extend it by showing that the misalignment of true and estimated eigenvectors affects the risk. Huang et al. [HHV22] use misalignment bounds [Lou17] to remove the known covariance assumption and obtain non-asymptotic risk bounds. We extend and complement existing analyses to the case real-world sample covariances  $\hat{C}$  to obtain asymptotic risk guarantees for the latent factor model using random matrix theory.

**Spiked covariance model.** The spiked covariance model was introduced in Johnstone [Joh01] for high-dimensional covariance matrices where data is generated from a low-dimensional subspace. It is a popular model for covariance estimation [DGJ18; DLS20] and has been used to analyse the performance of PCA [JP18; BS10]. We combine the spiked covariance model with the latent factor model to obtain a regression model with low-dimensional latent factors

and high-dimensional covariates. Then, we apply asymptotic results from random matrix theory for the eigenvalue shift of the spikes [BAP05; BS06], and the eigenvector misalignment [Pau07; Joh+09] to obtain asymptotic risk guarantees for PCR.

## 7 Conclusion

**Guide to choosing  $k$ .** Deriving precise guidelines from our analysis is challenging. From Theorems 2 and 4, we can see that choosing a larger number of principal components  $k$  increases the variance due to noise contributions. From Figures 5 and 7, we deduce that choosing  $k$  too small increases the risk due to discarding signal components. In practice, for  $\gamma = \frac{p}{n}$  small  $< 0.5$  or large  $> 2$ ,  $k$  has little effect on the risk.

**Summary.** We present asymptotic results for the generalization risk of PCR under the spiked covariance model based on random matrix theory. Furthermore, we consider the case where the training and test distribution vary. Our analysis generalizes the asymptotic result for linear regression from Hastie et al. [Has+22] which is a special case of PCR without dimensionality reduction ( $k = p$ ). In the non-asymptotic regime, Huang et al. [HHV22] show similar results, thus independently supporting our findings. Selecting the correct number of principal components  $k$  is crucial for the risk as Theorems 2 and 4 suggest.

While our results that PCA mitigates the interpolation peak due to its regularizing behaviour may not surprise, we provide formal guarantees for the risk of a commonly used model on real-world data structures. Practitioners can now rely on fundamental guarantees for model development, but more research is needed for general data structures.

**Limitations and future work.** In this paper, we limit the theoretical analysis to the supervised case where data is sampled from the spiked covariance model with linear regressors. Our analysis is based on random matrix theory and therefore only holds in the asymptotic regime. For finite sample risk bounds, we refer to Bing et al. [Bin+21].

Extending our analysis to more general covariance matrices  $C$  is an important extension and the tools we developed could be exploited since the Stieltjes transformation holds in more general settings. We believe the phenomenon we study with decaying eigenvalues is the most relevant for generalization analysis. This is extensively studied e.g. in Bartlett et al. [Bar+20]. One way to include the effect of gradient-based training could be to generalize our findings to spectral regularization techniques such as Landweber iteration [BPR07]. Another interesting avenue is the extension into semi-supervised settings [WL07].

Scenarios, where the PCA is trained on a large unlabeled dataset and then used for regression on a small labeled dataset, are related to the common idea of pre-training large models and might shed light on the generalization behaviour of such models.

**Acknowledgments** This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. We would like to thank Parthe Pandit for essential discussions about the covariate shift part of the paper.

## References

- [AGZ10] G. W. Anderson, A. Guionnet, and O. Zeitouni. *An introduction to random matrices*. Cambridge university press, 2010 (cit. on pp. III-4, III-5).
- [ASS20] M. S. Advani, A. M. Saxe, and H. Sompolinsky. “High-dimensional dynamics of generalization error in neural networks.” In: *Neural Networks* 132 (2020), pp. 428–446 (cit. on pp. III-1, III-14).
- [Bac23] F. Bach. “High-dimensional analysis of double descent for linear regression with random projections.” In: *arXiv preprint arXiv:2303.01372* (2023) (cit. on p. III-6).
- [BAP05] J. Baik, G. B. Arous, and S. Péché. “Phase transition of the largest eigenvalue for nonnull complex sample covariance matrices.” In: *The Annals of Probability* 33.5 (2005), pp. 1643–1697 (cit. on pp. III-2, III-5, III-16).
- [Bar+20] P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler. “Benign overfitting in linear regression.” In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30063–30070 (cit. on pp. III-2, III-14, III-16).
- [Bel+19] M. Belkin, D. Hsu, S. Ma, and S. Mandal. “Reconciling modern machine-learning practice and the classical bias–variance trade-off.” In: *Proceedings of the National Academy of Sciences* 116.32 (2019), pp. 15849–15854 (cit. on pp. III-1, III-14).
- [BF83] L. Breiman and D. Freedman. “How many variables should be entered in a regression equation?” In: *Journal of the American Statistical Association* 78.381 (1983), pp. 131–136 (cit. on p. III-15).

Paper III – No Double Descent in Principal Component Regression: A High-Dimensional Analysis

- [BHM18] M. Belkin, D. J. Hsu, and P. Mitra. “Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate.” In: *Advances in neural information processing systems* 31 (2018) (cit. on p. III-1).
- [Bin+21] X. Bing, F. Bunea, S. Strimas-Mackey, and M. Wegkamp. “Prediction under latent factor regression: Adaptive PCR, interpolating predictors and beyond.” In: *The Journal of Machine Learning Research* 22.1 (2021), pp. 7994–8043 (cit. on pp. III-15, III-16).
- [BM02] P. L. Bartlett and S. Mendelson. “Rademacher and Gaussian complexities: Risk bounds and structural results.” In: *Journal of Machine Learning Research* 3.Nov (2002), pp. 463–482 (cit. on p. III-1).
- [BPR07] F. Bauer, S. Pereverzev, and L. Rosasco. “On regularization algorithms in learning theory.” In: *Journal of complexity* 23.1 (2007), pp. 52–72 (cit. on pp. III-15, III-16).
- [BS06] J. Baik and J. W. Silverstein. “Eigenvalues of large sample covariance matrices of spiked population models.” In: *Journal of multivariate analysis* 97.6 (2006), pp. 1382–1408 (cit. on pp. III-2, III-5, III-16).
- [BS10] Z. Bai and J. W. Silverstein. *Spectral analysis of large dimensional random matrices*. Vol. 20. Springer, 2010 (cit. on pp. III-6, III-15).
- [BV13] A. Bloemendal and B. Virág. “Limits of spiked random matrices I.” In: *Probability Theory and Related Fields* 156 (2013), pp. 795–825 (cit. on p. III-5).
- [Cao+22] Y. Cao, Z. Chen, M. Belkin, and Q. Gu. “Benign overfitting in two-layer convolutional neural networks.” In: *Advances in neural information processing systems* 35 (2022), pp. 25237–25250 (cit. on p. III-14).
- [dAs+20] S. d’Ascoli, M. Refinetti, G. Biroli, and F. Krzakala. “Double trouble in double descent: Bias and variance (s) in the lazy regime.” In: *International Conference on Machine Learning*. PMLR. 2020, pp. 2280–2290 (cit. on p. III-14).
- [DFH17] L. H. Dicker, D. P. Foster, and D. Hsu. “Kernel ridge vs. principal component regression: Minimax bounds and the qualification of regularization operators.” In: *Electronic Journal of Statistics* 11.1 (2017), pp. 1022–1047 (cit. on p. III-15).
- [DGJ18] D. L. Donoho, M. Gavish, and I. M. Johnstone. “Optimal shrinkage of eigenvalues in the spiked covariance model.” In: *Annals of statistics* 46.4 (2018), p. 1742 (cit. on p. III-15).

- [DKT22] Z. Deng, A. Kammoun, and C. Thrampoulidis. “A model of double descent for high-dimensional binary linear classification.” In: *Information and Inference: A Journal of the IMA* 11.2 (2022), pp. 435–495 (cit. on p. III-14).
- [DLS20] E. Dobriban, W. Leeb, and A. Singer. “Optimal prediction in the linearly transformed spiked model.” In: *The Annals of Statistics* 48.1 (2020), pp. 491–513 (cit. on p. III-15).
- [DR17] G. K. Dziugaite and D. M. Roy. “Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data.” In: *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017 (cit. on p. III-1).
- [Ema+20] M. Emami, M. Sahraee-Ardakan, P. Pandit, S. Rangan, and A. Fletcher. “Generalization error of generalized linear models in high dimensions.” In: *International Conference on Machine Learning*. PMLR. 2020, pp. 2892–2901 (cit. on pp. III-10, III-15).
- [FCB22] S. Frei, N. S. Chatterji, and P. Bartlett. “Benign overfitting without linearity: Neural network classifiers trained by gradient descent for noisy linear data.” In: *Conference on Learning Theory*. PMLR. 2022, pp. 2668–2703 (cit. on p. III-14).
- [GBD92] S. Geman, E. Bienenstock, and R. Doursat. “Neural Networks and the Bias/Variance Dilemma.” In: *Neural Computation* 4.1 (1992), pp. 1–58 (cit. on p. III-14).
- [Ger+20] F. Gerace, B. Loureiro, F. Krzakala, M. Mézard, and L. Zdeborová. “Generalisation error in learning with random features and the hidden manifold model.” In: *International Conference on Machine Learning*. PMLR. 2020, pp. 3452–3462 (cit. on p. III-14).
- [Gew96] J. Geweke. “Bayesian reduced rank regression in econometrics.” In: *Journal of econometrics* 75.1 (1996), pp. 121–146 (cit. on p. III-1).
- [Has+22] T. Hastie, A. Montanari, S. Rosset, and R. J. Tibshirani. “Surprises in high-dimensional ridgeless least squares interpolation.” In: *The Annals of Statistics* 50.2 (2022), pp. 949–986 (cit. on pp. III-2, III-9, III-14, III-16, III-25, III-34, III-36).
- [HHV22] N. ( Huang, D. W. Hogg, and S. Villar. “Dimensionality Reduction, Regularization, and Generalization in Overparameterized Regressions.” In: *SIAM Journal on Mathematics of Data Science* 4.1 (2022), pp. 126–152 (cit. on pp. III-15, III-16).

Paper III – No Double Descent in Principal Component Regression: A High-Dimensional Analysis

- [HW23] L. Hucker and M. Wahl. “A note on the prediction error of principal component regression in high dimensions.” In: *Theory of Probability and Mathematical Statistics* 109 (2023), pp. 37–53 (cit. on p. III-15).
- [Joh+09] I. M. Johnstone, A. Y. Lu, B. Nadler, D. M. Witten, T. Hastie, R. Tibshirani, and J. O. Ramsay. “On Consistency and Sparsity for Principal Components Analysis in High Dimensions.” In: *Journal of the American Statistical Association* 104.486 (2009), pp. 682–703 (cit. on pp. III-7, III-16).
- [Joh01] I. M. Johnstone. “On the distribution of the largest eigenvalue in principal components analysis.” In: *Annals of statistics* 29.2 (2001), pp. 295–327 (cit. on pp. III-2, III-3, III-5, III-15).
- [Jol02] I. T. Jolliffe. *Principal component analysis for special types of data*. Springer, 2002 (cit. on pp. III-1, III-15).
- [JP18] I. M. Johnstone and D. Paul. “PCA in high dimensions: An orientation.” In: *Proceedings of the IEEE* 106.8 (2018), pp. 1277–1292 (cit. on pp. III-7, III-15).
- [KH91] A. Krogh and J. Hertz. “A simple weight decay can improve generalization.” In: *Advances in neural information processing systems* 4 (1991) (cit. on p. III-14).
- [LJB20] D. LeJeune, H. Javadi, and R. Baraniuk. “The implicit regularization of ordinary least squares ensembles.” In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 3525–3535 (cit. on p. III-14).
- [LLP12] Y. K. Lee, E. R. Lee, and B. U. Park. “Principal component analysis in very high-dimensional spaces.” In: *Statistica Sinica* (2012), pp. 933–956 (cit. on p. III-15).
- [Lou+21] B. Loureiro, G. Sicuro, C. Gerbelot, A. Pacco, F. Krzakala, and L. Zdeborová. “Learning Gaussian Mixtures with Generalized Linear Models: Precise Asymptotics in High-dimensions.” In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 10144–10157 (cit. on p. III-14).
- [Lou17] A. Loukas. “How close are the eigenvectors of the sample and actual covariance matrices?” In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2228–2237 (cit. on p. III-15).
- [Mas65] W. F. Massy. “Principal components regression in exploratory statistical research.” In: *Journal of the American Statistical Association* 60.309 (1965), pp. 234–256 (cit. on p. III-1).

- [MM22] S. Mei and A. Montanari. “The generalization error of random features regression: Precise asymptotics and the double descent curve.” In: *Communications on Pure and Applied Mathematics* 75.4 (2022), pp. 667–766 (cit. on pp. III-2, III-14).
- [MP67] V. A. Marčenko and L. A. Pastur. “The distribution of eigenvalues in certain sets of random matrices.” In: *Mathematics of the USSR-Sbornik* 1.4 (1967), pp. 457–483 (cit. on p. III-4).
- [Mut+20] V. Muthukumar, K. Vodrahalli, V. Subramanian, and A. Sahai. “Harmless interpolation of noisy data in regression.” In: *IEEE Journal on Selected Areas in Information Theory* 1.1 (2020), pp. 67–83 (cit. on p. III-14).
- [Nak+21] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. “Deep double descent: Where bigger models and more data hurt.” In: *Journal of Statistical Mechanics: Theory and Experiment* 2021.12 (2021), p. 124003 (cit. on p. III-14).
- [NTS15] B. Neyshabur, R. Tomioka, and N. Srebro. “In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning.” In: *ICLR (Workshop)*. 2015 (cit. on p. III-1).
- [Opp95] M. Opper. “Statistical mechanics of learning: Generalization.” In: *The handbook of brain theory and neural networks* (1995), pp. 922–925 (cit. on p. III-14).
- [Pau07] D. Paul. “Asymptotics of sample eigenstructure for a large dimensional spiked covariance model.” In: *Statistica Sinica* (2007), pp. 1617–1642 (cit. on pp. III-5, III-7, III-16).
- [Pea01] K. Pearson. “LIII. On lines and planes of closest fit to systems of points in space.” In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572 (cit. on pp. III-1, III-15).
- [PW17] J. Pennington and P. Worah. “Nonlinear random matrix theory for deep learning.” In: *Advances in neural information processing systems* 30 (2017) (cit. on p. III-2).
- [Sco+21] M. F. Scott, N. Fradgley, A. R. Bentley, T. Brabbs, F. Corke, K. A. Gardner, R. Horsnell, P. Howell, O. Ladejobi, I. J. Mackay, et al. “Limited haplotype diversity underlies polygenic trait architecture across 70 years of wheat breeding.” In: *Genome biology* 22.1 (2021), pp. 1–30 (cit. on pp. III-2, III-24).
- [Sti94] T.-J. Stieltjes. “Recherches sur les fractions continues.” In: *Annales de la Faculté des sciences de Toulouse: Mathématiques*. Vol. 8. 1894, J1–J122 (cit. on p. III-5).
- [Tao23] T. Tao. *Topics in random matrix theory*. Vol. 132. American Mathematical Society, 2023 (cit. on p. III-4).

Paper III – No Double Descent in Principal Component Regression: A High-Dimensional Analysis

- [TAP21a] N. Tripuraneni, B. Adlam, and J. Pennington. “Covariate shift in high-dimensional random feature regression.” In: *arXiv preprint arXiv:2111.08234* (2021) (cit. on p. III-15).
- [TAP21b] N. Tripuraneni, B. Adlam, and J. Pennington. “Overparameterization improves robustness to covariate shift in high dimensions.” In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 13883–13897 (cit. on p. III-15).
- [TB23] A. Tsigler and P. L. Bartlett. “Benign overfitting in ridge regression.” In: *J. Mach. Learn. Res.* 24 (2023), pp. 123–1 (cit. on p. III-15).
- [TSL00] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. “A global geometric framework for nonlinear dimensionality reduction.” In: *science* 290.5500 (2000), pp. 2319–2323 (cit. on p. III-2).
- [VS00] S. Vijayakumar and S. Schaal. “Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space.” In: *Proceedings of the seventeenth international conference on machine learning (ICML 2000)*. Vol. 1. 2000, pp. 288–293 (cit. on p. III-1).
- [WA08] K. Wang and D. Abbott. “A principal components regression approach to multilocus genetic association studies.” In: *Genet. Epidemiol.* 32.2 (2008), pp. 108–118 (cit. on p. III-1).
- [WL07] L. Wasserman and J. Lafferty. “Statistical analysis of semi-supervised regression.” In: *Advances in Neural Information Processing Systems* 20 (2007) (cit. on p. III-16).
- [WMT21] K. Wang, V. Muthukumar, and C. Thrampoulidis. “Benign overfitting in multiclass classification: All roads lead to interpolation.” In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 24164–24179 (cit. on p. III-14).
- [WX20] D. Wu and J. Xu. “On the Optimal Weighted  $\ell_2$  Regularization in Overparameterized Linear Regression.” In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 10112–10123 (cit. on p. III-15).
- [XH19] J. Xu and D. J. Hsu. “On the number of variables to use in principal component regression.” In: *Advances in neural information processing systems* 32 (2019) (cit. on p. III-15).
- [YJ18] J. Yang and I. M. Johnstone. “Edgeworth correction for the largest eigenvalue in a spiked pca model.” In: *Statistica Sinica* 28.4 (2018), p. 2541 (cit. on p. III-5).

- [Zha+17] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. “Understanding deep learning requires rethinking generalization.” In: *International Conference on Learning Representations*. 2017 (cit. on p. III-1).

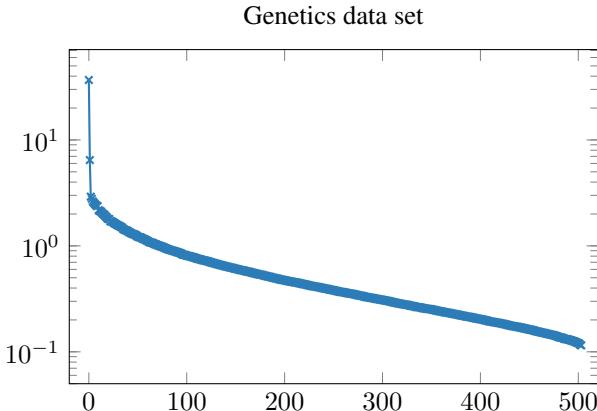


## Appendix

### A Experiment details: genetics example

**Background.** The Diverse MAGIC Wheat data set<sup>3</sup> is based on 16 founding wheat varieties which were listed between 1935 and 2004. These varieties were interbred to obtain new wheat varieties. From the resulting wheat types, the genome of a total of 502 kinds of wheat were sequenced. This genome sequence consists of approximately 1.1 million single nucleotide polymorphisms. Furthermore, phenotypes of the 502 wheat types were analysed, see Scott et al. [Sco+21].

**Data processing.** The genotypes consist of binary features. The binary variables represent equality or difference to a reference genotype. The phenotypes are real-values variables. We choose the phenotype column named 'HET\_2' in this example. Missing values for both, genotype and phenotype are replaced with the mean value of the variable. We select a subset of genotypes as inputs randomly at uniform to obtain the necessary  $p$  features. Then, we normalize both, genotype and phenotype by z-transformation.



**Figure 8: Eigenvalue distribution of the Diverse MAGIC Wheat genetics data set.**

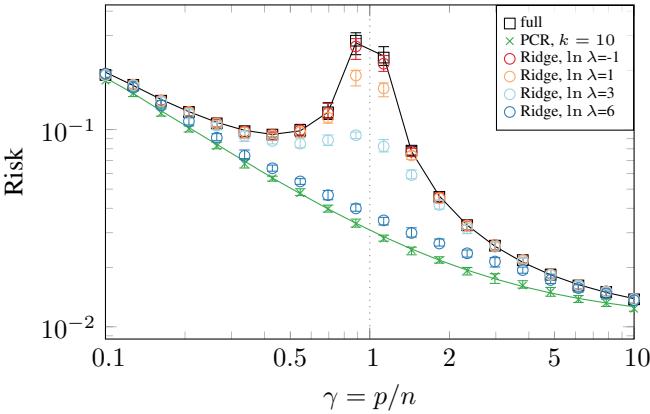
**Data analysis.** In Figure 8 we plot the eigenvalue distribution for the Diverse MAGIC Wheat data set. We observe that the eigenvalue distribution is heavy-tailed. While it has two dominant eigenvalues, it does not depict a clear example of a low-dimensional latent manifold. Therefore, using the PCR model will discard some useful information. We also note, that there is no reason to assume a linear relationship between the features and the outcome which is in contrast to our synthetically generated data.

<sup>3</sup><http://mtweb.cs.ucl.ac.uk/mus/www/MAGICdiverse/>

## B Additional numerical results

### B.1 Ridge regression

We compare the PCR model with different values for Ridge regression for in-distribution data. The results are depicted in Figure 9. Here, we compare with the optimal  $k = d$  PCR model. Note that solid lines indicate analytical solutions while markers indicate simulations. Analytical solutions for Ridge regression could be obtained, see Hastie et al. [Has+22]. We can observe that Ridge regression shows similar regularizing behaviour as regularization through PCR. However, the exact shape of the risk curve varies and for the case we present, PCR with  $k = d$  shows the lowest risk for all  $\gamma$ .



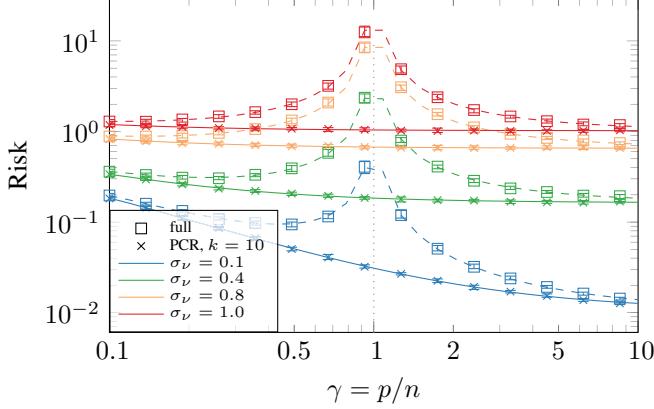
**Figure 9: Risk PCR vs. Ridge regularisation.** We show median, 25%, and 75% quantiles over 50 seeds for all simulation results.

### B.2 Varying output noise $\sigma_\nu$

In Figure 10, we compare the PCR model under varying output noise  $\sigma_\nu$  for in-distribution data. Results from Theorem 2 are compared with simulations. The figure shows that our analysis holds for a suitable range of noise values and that the risk decreases with lower noise variance.

## C Proofs

In this section, we will follow the structure of the main paper to prove all results. For clarity, we will repeat the results before providing the proof.



**Figure 10: Risk PCR for varying  $\sigma_\nu$ : Simulation vs. analysis.** We show median, 25%, and 75% quantiles over 50 seeds for all simulation results. Dashed lines are analytical solutions for the full regression; solid lines are analytical solutions for the PCR.

## C.1 Data generator

In this section, we derive the linear model from the latent factor model in Definition 1. Let us first restate the definition:

**Definition 1.** The *latent factor model* is the linear model  $\mathbf{x}_i = \mathbf{W}r_w\mathbf{z}_i + \mathbf{e}_i$ , and  $y_i = \boldsymbol{\theta}^\top \mathbf{z}_i + \varepsilon_i$ . With latent variables  $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_z)$ , feature noise  $\mathbf{e}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)$ , outcome noise  $\varepsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ , feature matrix  $\mathbf{W} \in \mathbb{R}^{p \times d}$  such that  $\mathbf{W}^\top \mathbf{W} = \mathbf{I}_d$ . Let  $r_w^2 = \frac{p}{\text{Tr}(\mathbf{A}_z)} \rho_x$  to control the feature signal-to-noise-ratio (SNR)  $\rho_x = \frac{\mathbb{E}[\|\mathbf{W}r_w\mathbf{z}\|_2^2]}{\mathbb{E}[\|\mathbf{e}\|_2^2]}$ , label noise  $\varepsilon_i \sim \mathcal{N}(\mathbf{0}, \sigma_\varepsilon^2)$  and let  $\boldsymbol{\theta} = \mathbf{1}_d \frac{r_\theta}{\sqrt{\text{Tr}(\mathbf{A}_z)}}$  to control the outcome SNR  $\rho_y = \frac{\mathbb{E}[\|\boldsymbol{\theta}^\top \mathbf{z}\|_2^2]}{\mathbb{E}[\|\varepsilon\|_2^2]} = \frac{r_\theta^2}{\sigma_\varepsilon^2}$ .

We aim to derive the linear model  $y_i = \boldsymbol{\beta}^\top \mathbf{x}_i + \nu_i$ , i.e. to find the expressions for

$$\mathbf{x}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C}), \quad (\text{C-1})$$

$$\nu_i \sim \mathcal{N}(0, \sigma_\nu^2) \quad \text{with} \quad \sigma_\nu^2 = \sigma_\varepsilon^2 + \boldsymbol{\theta}^\top (\mathbf{I}_d + r_w^2 \mathbf{C}_z)^{-1} \mathbf{C}_z \boldsymbol{\theta}, \quad (\text{C-2})$$

$$\boldsymbol{\beta} = r_w \mathbf{W} \mathbf{C}_z (\mathbf{I}_d + r_w^2 \mathbf{C}_z)^{-1}. \quad (\text{C-3})$$

*Proof.* The covariance matrix of  $(\mathbf{x}_i, y_i)$  under the linear model is given by

$$\begin{bmatrix} \mathbb{E}[\mathbf{x}_i \mathbf{x}_i^\top] & \mathbb{E}[y_i \mathbf{x}_i^\top] \\ \mathbb{E}[y_i^\top \mathbf{x}_i] & \mathbb{E}[y_i y_i^\top] \end{bmatrix} = \begin{bmatrix} \mathbf{C} & \boldsymbol{\beta}^\top \mathbf{C} \\ \mathbf{C} \boldsymbol{\beta} & \boldsymbol{\beta}^\top \mathbf{C} \boldsymbol{\beta} + \sigma_\nu^2 \end{bmatrix}. \quad (\text{C-4})$$

We can compare this with the covariance under the latent factor model

$$\begin{bmatrix} \mathbb{E}[\mathbf{W} \mathbf{z}_i \mathbf{z}_i^\top \mathbf{W}^\top r_w^2 + \mathbf{e}_i \mathbf{e}_i^\top] & \mathbb{E}[\boldsymbol{\theta}^\top \mathbf{z}_i \mathbf{z}_i^\top \mathbf{W}^\top r_w + \varepsilon_i \mathbf{e}_i^\top] \\ \mathbb{E}[r_w \mathbf{W} \mathbf{z}_i \mathbf{z}_i^\top \boldsymbol{\theta} + \mathbf{e}_i \varepsilon_i^\top] & \mathbb{E}[\boldsymbol{\theta}^\top \mathbf{z}_i \mathbf{z}_i^\top \boldsymbol{\theta} + \varepsilon_i \varepsilon_i^\top] \end{bmatrix} = \quad (\text{C-5})$$

$$\begin{bmatrix} \mathbf{W} \mathbf{C}_z \mathbf{W}^\top r_w^2 + \mathbf{I}_p & \boldsymbol{\theta}^\top \mathbf{C}_z \mathbf{W}^\top r_w \\ r_w \mathbf{W} \mathbf{C}_z \boldsymbol{\theta} & \boldsymbol{\theta}^\top \mathbf{C}_z \boldsymbol{\theta} + \sigma_\varepsilon^2 \end{bmatrix}. \quad (\text{C-6})$$

Comparing element (1, 1), we directly observe that  $\mathbf{C} = \mathbf{W} \mathbf{C}_z \mathbf{W}^\top r_w^2 + \mathbf{I}_p$  which is our spiked covariance model.

From element (2, 1), we get the following when using  $\mathbf{C}$  as well as  $\mathbf{W} \in \mathbb{R}^{p \times d}$  such that  $\mathbf{W}^\top \mathbf{W} = \mathbf{I}_d$

$$\mathbf{C}\boldsymbol{\beta} = r_w \mathbf{W} \mathbf{C}_z \boldsymbol{\theta} \quad (\text{C-7})$$

$$\boldsymbol{\beta} = \begin{bmatrix} (\mathbf{C}_z r_w^2 + \mathbf{I}_d)^{-1} & 0 \\ 0 & \mathbf{I}_{p-d} \end{bmatrix} r_w \mathbf{W} \mathbf{C}_z \boldsymbol{\theta} \quad (\text{C-8})$$

$$= (\mathbf{I}_p + \mathbf{W} \mathbf{C}_z \mathbf{W}^\top r_w^2)^{-1} r_w \mathbf{W} \mathbf{C}_z \boldsymbol{\theta}. \quad (\text{C-9})$$

Using the push-through or Woodbury matrix identity, we obtain the result for  $\boldsymbol{\beta}$ .

Finally from element (2, 2), we get the following expression

$$\sigma_\nu^2 = \boldsymbol{\theta}^\top \mathbf{C}_z \boldsymbol{\theta} - \boldsymbol{\beta}^\top \mathbf{C} \boldsymbol{\beta} + \sigma_\varepsilon^2. \quad (\text{C-10})$$

Using (C-9) for  $\boldsymbol{\beta}^\top$  with the result for  $\boldsymbol{\beta}$  and  $\mathbf{C}$ , we obtain

$$\sigma_\nu^2 = \boldsymbol{\theta}^\top (\mathbf{C}_z - \mathbf{C}_z \mathbf{C}_z r_w^2 (\mathbf{I}_d + r_w^2 \mathbf{C}_z)^{-1}) \boldsymbol{\theta} + \sigma_\varepsilon^2 \quad (\text{C-11})$$

$$= \boldsymbol{\theta}^\top \mathbf{C}_z (\mathbf{I}_d - \mathbf{C}_z r_w^2 (\mathbf{I}_d + r_w^2 \mathbf{C}_z)^{-1}) \boldsymbol{\theta} + \sigma_\varepsilon^2 \quad (\text{C-12})$$

Using the identity  $(\mathbf{I} + \mathbf{P})^{-1} = \mathbf{I} - (\mathbf{I} + \mathbf{P})^{-1} \mathbf{P}$ , we obtain

$$\sigma_\nu^2 = \sigma_\varepsilon^2 + \boldsymbol{\theta}^\top \mathbf{C}_z (\mathbf{I}_d + r_w^2 \mathbf{C}_z)^{-1} \boldsymbol{\theta}. \quad (\text{C-13})$$

Using the push-through or Woodbury identity yields the result for the variance.  $\square$

## C.2 Analysis of risk

### Risk of PCR

**PCR parameter estimator.** Let us first re-state the parameter estimator for PCR:

$$\hat{\boldsymbol{\theta}} = \hat{\mathbf{V}}^\top \boldsymbol{\beta} + \hat{\mathbf{S}}^{-1} \hat{\mathbf{U}}^\top \boldsymbol{\nu}. \quad (3)$$

*Proof.* We consider the unregularized linear regression solution between the latent variables  $\hat{\mathbf{Z}}$  and the outcome  $\mathbf{y}$ :

$$\hat{\boldsymbol{\theta}} = (\hat{\mathbf{Z}}^\top \hat{\mathbf{Z}})^+ \hat{\mathbf{Z}}^\top \mathbf{y} \quad (\text{C-14})$$

$$= (\hat{\mathbf{S}}_k^\top \hat{\mathbf{U}}^\top \hat{\mathbf{U}} \hat{\mathbf{S}}_k)^+ \hat{\mathbf{S}}_k^\top \hat{\mathbf{U}}^\top \mathbf{y} \quad (\text{C-15})$$

with  $\hat{\mathbf{U}}^\top \hat{\mathbf{U}} = \mathbf{I}$  and  $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\nu}$

$$\hat{\boldsymbol{\theta}} = (\hat{\mathbf{S}}_k^\top \hat{\mathbf{S}}_k)^+ \hat{\mathbf{S}}_k^\top \hat{\mathbf{U}}^\top (\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\nu}) \quad (\text{C-16})$$

$$= (\hat{\mathbf{S}}_k^\top \hat{\mathbf{S}}_k)^+ \hat{\mathbf{S}}_k^\top \hat{\mathbf{S}} \hat{\mathbf{V}}^\top \boldsymbol{\beta} + (\hat{\mathbf{S}}_k^\top \hat{\mathbf{S}}_k)^+ \hat{\mathbf{S}}_k^\top \hat{\mathbf{U}}^\top \boldsymbol{\nu} \quad (\text{C-17})$$

Where we used  $\mathbf{X} = \hat{\mathbf{U}} \hat{\mathbf{S}} \hat{\mathbf{V}}^\top$ . Now we combine the singular value matrices. We indicate the dimensions of combined matrices. Note that  $\hat{\mathbf{S}}_k$  and  $\hat{\mathbf{S}}$  are of different sizes.

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \begin{bmatrix} \frac{1}{\hat{\sigma}_1^2} & \mathbf{0} \\ \mathbf{0} & \ddots & \frac{1}{\hat{\sigma}_k^2} \end{bmatrix}_{k \times k} \begin{bmatrix} \hat{\sigma}_1^2 & \mathbf{0} \\ \mathbf{0} & \ddots & \hat{\sigma}_k^2 \end{bmatrix}_{k \times p} \begin{bmatrix} \hat{\mathbf{V}}^\top \boldsymbol{\beta} \\ \mathbf{0} \end{bmatrix} + \\ &\quad + \begin{bmatrix} \frac{1}{\hat{\sigma}_1} & \mathbf{0} \\ \mathbf{0} & \ddots & \frac{1}{\hat{\sigma}_k} \end{bmatrix}_{k \times n} \begin{bmatrix} \hat{\mathbf{U}}^\top \boldsymbol{\nu} \\ \mathbf{0} \end{bmatrix} \end{aligned} \quad (\text{C-18})$$

$$= [\mathbf{I}_k \quad \mathbf{0}] \hat{\mathbf{V}}^\top \boldsymbol{\beta} + [\hat{\mathbf{S}}_{kk}^{-1} \quad \mathbf{0}] \hat{\mathbf{U}}^\top \boldsymbol{\nu} \quad (\text{C-19})$$

Summarizing the matrices by truncating  $\hat{\mathbf{V}}^\top$  and  $\hat{\mathbf{U}}^\top$  yields the following solution for the regression parameter estimation

$$\hat{\boldsymbol{\theta}} = \hat{\mathbf{V}}_k^\top \boldsymbol{\beta} + \hat{\mathbf{S}}_{kk}^{-1} \hat{\mathbf{U}}_k^\top \boldsymbol{\nu}. \quad (\text{C-20})$$

This concludes the proof as we defined  $\hat{\mathbf{V}} := \hat{\mathbf{V}}_k$ ,  $\hat{\mathbf{S}} := \hat{\mathbf{S}}_{kk}$  and  $\hat{\mathbf{U}} := \hat{\mathbf{U}}_k$ .  $\square$

**Expected risk of PCR.** Let us first re-state the expected risk result for PCR:

$$\mathbb{E}_{\boldsymbol{\nu}} [R(\hat{\boldsymbol{\theta}})] = \boldsymbol{\beta}^\top \boldsymbol{\Pi} \mathbf{C} \boldsymbol{\Pi} \boldsymbol{\beta} + \frac{\sigma_\nu^2}{n} \text{Tr} (\hat{\mathbf{V}}^\top \mathbf{C} \hat{\mathbf{V}} \hat{\mathbf{V}}^\top \hat{\mathbf{C}}^{-1} \hat{\mathbf{V}}) + \sigma_\nu^2. \quad (4)$$

*Proof.* We define the risk as the expectation over the mean squared error and then use  $y_0 = \beta^\top \mathbf{x}_0 + \nu_0$ , as well as  $\hat{y}(\mathbf{x}_0) = \hat{\theta}^\top \hat{\mathbf{z}}$  and  $\hat{\mathbf{z}} = \hat{\mathbf{V}}^\top \mathbf{x}_0$  to obtain

$$R(\hat{\theta}) = \mathbb{E}_{(\mathbf{x}_0, y_0)} [(y_0 - \hat{y}(\mathbf{x}_0))^2] \quad (\text{C-21})$$

$$= \mathbb{E}_{\mathbf{x}_0} [(\beta^\top \mathbf{x}_0 + \nu_0 - \hat{y}(\mathbf{x}_0))^2] \quad (\text{C-22})$$

$$= \mathbb{E}_{\mathbf{x}_0} [(\beta^\top \mathbf{x}_0 + \nu_0 - \hat{\theta}^\top \hat{\mathbf{z}})^2] \quad (\text{C-23})$$

$$= \mathbb{E}_{\mathbf{x}_0} [(\beta^\top \mathbf{x}_0 + \nu_0 - \hat{\theta}^\top \hat{\mathbf{V}}^\top \mathbf{x}_0)^2] \quad (\text{C-24})$$

$$= \mathbb{E}_{\mathbf{x}_0} \left[ \left( (\beta - \hat{\mathbf{V}}\hat{\theta})^\top \mathbf{x}_0 + \nu_0 \right)^2 \right] \quad (\text{C-25})$$

$$= (\beta - \hat{\mathbf{V}}\hat{\theta})^\top \mathbf{C}(\beta - \hat{\mathbf{V}}\hat{\theta}) + \nu_0 \nu_0^\top \quad (\text{C-26})$$

Define the orthogonal projector  $\Phi = \hat{\mathbf{V}}\hat{\mathbf{V}}^\top$  and define another orthogonal projector with  $\Pi = \mathbf{I}_p - \Phi$ . For simplicity, let us rephrase the following

$$\beta - \hat{\mathbf{V}}\hat{\theta} = \beta - \hat{\mathbf{V}}(\hat{\mathbf{V}}^\top \beta + \hat{\mathbf{S}}^{-1}\hat{\mathbf{U}}^\top \nu) \quad (\text{C-27})$$

$$= \beta - \hat{\mathbf{V}}\hat{\mathbf{V}}^\top \beta - \hat{\mathbf{V}}\hat{\mathbf{S}}^{-1}\hat{\mathbf{U}}^\top \nu \quad (\text{C-28})$$

$$= (\mathbf{I}_p - \Phi)\beta - \hat{\mathbf{V}}\hat{\mathbf{S}}^{-1}\hat{\mathbf{U}}^\top \nu \quad (\text{C-29})$$

$$= \Pi\beta - \hat{\mathbf{V}}\hat{\mathbf{S}}^{-1}\hat{\mathbf{U}}^\top \nu \quad (\text{C-30})$$

Now let us use this expression to take the expectation of the risk w.r.t. the noise. This yields

$$\mathbb{E}_\nu [R(\hat{\theta})] = \beta^\top \Pi \mathbf{C} \Pi \beta + \mathbb{E}_\nu [\text{Tr}(\nu^\top \hat{\mathbf{U}} \hat{\mathbf{S}}^{-1} \hat{\mathbf{V}}^\top \mathbf{C} \hat{\mathbf{V}} \hat{\mathbf{S}}^{-1} \hat{\mathbf{U}}^\top \nu)] + \mathbb{E}_\nu [\nu \nu^\top] \quad (\text{C-31})$$

Here we made use of the Trace since the expression is scalar. Hence, we can use the cyclic property of the Trace and pull the expectation inside

$$\mathbb{E}_\nu [R(\hat{\theta})] = \beta^\top \Pi \mathbf{C} \Pi \beta + \text{Tr}(\hat{\mathbf{V}}^\top \mathbf{C} \hat{\mathbf{V}} \hat{\mathbf{S}}^{-1} \hat{\mathbf{U}}^\top \mathbb{E}_\nu [\nu \nu^\top] \hat{\mathbf{U}} \hat{\mathbf{S}}^{-1}) + \mathbb{E}_\nu [\nu \nu^\top] \quad (\text{C-32})$$

with  $\mathbb{E}_\nu [\nu \nu^\top] = \sigma_\nu^2$  and  $\hat{\mathbf{U}}^\top \hat{\mathbf{U}} = \mathbf{I}$

$$\mathbb{E}_\nu [R(\hat{\theta})] = \beta^\top \Pi \mathbf{C} \Pi \beta + \sigma_\nu^2 \text{Tr}(\hat{\mathbf{V}}^\top \mathbf{C} \hat{\mathbf{V}} \hat{\mathbf{S}}^{-2}) + \sigma_\nu^2 \quad (\text{C-33})$$

Using  $\hat{\mathbf{S}}^{-2} = \frac{1}{n} \hat{\mathbf{V}}^\top \hat{\mathbf{C}}^+ \hat{\mathbf{V}}$  we obtain (4) which concludes the proof.  $\square$

**Eigenvector shift product.** Let us first re-state the result which generalizes the eigenvector shift to matrices:

$$\mathbf{P}_k = \begin{cases} \text{diag}((\mathbf{v}_1^\top \hat{\mathbf{v}}_1)^2, \dots, (\mathbf{v}_k^\top \hat{\mathbf{v}}_k)^2, 0, \dots, 0) & \text{for } k < d, \\ \text{diag}((\mathbf{v}_1^\top \hat{\mathbf{v}}_1)^2, \dots, (\mathbf{v}_d^\top \hat{\mathbf{v}}_d)^2) & \text{for } k = d, \\ \text{diag}((\mathbf{v}_1^\top \hat{\mathbf{v}}_1)^2 + c_1^2, \dots, (\mathbf{v}_d^\top \hat{\mathbf{v}}_d)^2 + c_d^2) & \text{for } k > d, \end{cases} \quad (5)$$

where  $\mathbf{V}_{d,d} \in \mathbb{R}^{d \times d}$  and  $\hat{\mathbf{V}}_d \in \mathbb{R}^{d \times k}$  are the matrices where the eigenvectors are truncated to the first  $d$  elements in each eigenvector; and with the correction factor  $c_i^2 = (k - d) \frac{1 - (\mathbf{v}_i^\top \hat{\mathbf{v}}_i)^2}{p-d}$  for  $k > d$ .

Let us give an informal justification of (5): Define  $\hat{\mathbf{V}}_d \in \mathbb{R}^{d \times k}$  and  $\mathbf{V}_{d,d} \in \mathbb{R}^{d \times d}$  as the matrices where the eigenvectors are truncated to the first  $d$  eigenvector elements. Then, the expression  $\mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}_d^\top \mathbf{V}_{d,d} \in \mathbb{R}^{d \times d}$  describes the matrix equivalent of the eigenvector shift (2) which is defined only for  $(\mathbf{v}_i^\top \hat{\mathbf{v}}_j)^2$  with  $i = j$ . These are the diagonal terms of the matrix expression. For off-diagonal values with  $i \neq j$  we have that  $(\mathbf{v}_i^\top \hat{\mathbf{v}}_j)^2 \rightarrow 0$  because as  $p \rightarrow \infty$  the estimates eigenvectors  $\hat{\mathbf{v}}_j$  are randomly placed in a  $p$ -dimensional space. Hence  $P((\mathbf{v}_i^\top \hat{\mathbf{v}}_j)^2 > \epsilon) \rightarrow 0$  which means that the expression  $\mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}_d^\top \mathbf{V}_{d,d}$  yields a diagonal matrix.

Dependent on the choice of  $k$ , there are three different results:

1. For  $k < d$ : since  $\hat{\mathbf{V}}_d \in \mathbb{R}^{d \times k}$ , we have that  $\hat{\mathbf{V}}_d \hat{\mathbf{V}}_d^\top$  will only have non-zero values for first  $k$  diagonal entries, yielding

$$\mathbf{P}_k = \mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}_d^\top \mathbf{V}_{d,d} = \text{diag}\left((\mathbf{v}_1^\top \hat{\mathbf{v}}_1)^2, \dots, (\mathbf{v}_k^\top \hat{\mathbf{v}}_k)^2, 0, \dots, 0\right). \quad (\text{C-34})$$

2. For  $k = d$ : with the same reasoning as in the previous case we obtain

$$\mathbf{P}_k = \mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}_d^\top \mathbf{V}_{d,d} = \text{diag}\left((\mathbf{v}_1^\top \hat{\mathbf{v}}_1)^2, \dots, (\mathbf{v}_d^\top \hat{\mathbf{v}}_d)^2\right). \quad (\text{C-35})$$

3. For  $k > d$ : Let us rewrite the matrix product as

$$\mathbf{P}_k = \mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}_d^\top \mathbf{V}_{d,d} = \mathbf{V}_{d,d}^\top \begin{bmatrix} \hat{\mathbf{V}}_{d,:d} & \hat{\mathbf{V}}_{d,d:} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{V}}_{d,:d}^\top \\ \hat{\mathbf{V}}_{d,d:}^\top \end{bmatrix} \mathbf{V}_{d,d} \quad (\text{C-36})$$

$$= \mathbf{V}_{d,d}^\top \left( \hat{\mathbf{V}}_{d,:d} \hat{\mathbf{V}}_{d,:d}^\top + \hat{\mathbf{V}}_{d,d:} \hat{\mathbf{V}}_{d,d:}^\top \right) \mathbf{V}_{d,d} \quad (\text{C-37})$$

$$= \mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_{d,:d} \hat{\mathbf{V}}_{d,:d}^\top \mathbf{V}_{d,d} + \mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_{d,d:} \hat{\mathbf{V}}_{d,d:}^\top \mathbf{V}_{d,d} \quad (\text{C-38})$$

where the first term if equal to  $\mathbf{P}_k$  for  $k = d$ . Let us write the second term element-wise as  $\mathbf{v}_i^\top \hat{\mathbf{v}}_{i,d:} \hat{\mathbf{v}}_{i,d:}^\top \mathbf{v}_i$  which is equal to  $\hat{\mathbf{v}}_{i,d:} \hat{\mathbf{v}}_{i,d:}^\top$  since  $\mathbf{v}_{ij} = 0$  if  $i \neq j$ . Hence, we need to know the expected value of the elements  $\hat{\mathbf{v}}_{ij}$  for  $j > d$ . To identify this, we can note that  $1 = \hat{\mathbf{v}}_i^\top \hat{\mathbf{v}}_i$  which we can expand as  $1 = \sum_{j=1}^d \hat{\mathbf{v}}_{ij}^\top \hat{\mathbf{v}}_{ij} + \sum_{j=d+1}^p \hat{\mathbf{v}}_{ij}^\top \hat{\mathbf{v}}_{ij}$ . Again, we can expand the first sum with  $\mathbf{v}$  as  $\mathbf{v}_{ij} = 1$  only for  $i = j$ . Hence, we obtain

$$1 = (\mathbf{v}_i^\top \hat{\mathbf{v}}_i)^2 + \sum_{j=d+1}^p \hat{\mathbf{v}}_{ij}^\top \hat{\mathbf{v}}_{ij} \quad (\text{C-39})$$

Hence, assuming that  $\mathbf{v}_{ij}$  are uniformly distributed, we get that  $\hat{\mathbf{v}}_{ij}^\top \hat{\mathbf{v}}_{ij} = \frac{1 - (\mathbf{v}_i^\top \hat{\mathbf{v}}_i)^2}{p-d}$ . Finally, we see that we have to sum  $k - d$  of these elements in the expression which leads to the second term in (C-38) yielding

$$c_i^2 = (k - d) \frac{1 - (\mathbf{v}_i^\top \hat{\mathbf{v}}_i)^2}{p - d}. \quad (\text{C-40})$$

Finally, we obtain in the case for  $k > d$

$$\mathbf{P}_k = \mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}_d^\top \mathbf{V}_{d,d} = \text{diag} \left( (\mathbf{v}_1^\top \hat{\mathbf{v}}_1)^2 + c_1^2, \dots, (\mathbf{v}_d^\top \hat{\mathbf{v}}_d)^2 + c_d^2 \right). \quad (\text{C-41})$$

**Asymptotic PCR risk.** In order to prove Theorem 2, we re-state the results for the squared bias and the variance terms first: The *asymptotic squared bias* term is given by

$$\text{Bias}_\gamma(\hat{\boldsymbol{\theta}})^2 = \bar{\boldsymbol{\beta}}^\top (\boldsymbol{\Lambda}_d - \boldsymbol{\Lambda}_d \mathbf{P}_k - \mathbf{P}_k \boldsymbol{\Lambda}_d + \mathbf{P}_k + \mathbf{P}_k r_w^2 \mathbf{C}_z \mathbf{P}_k) \bar{\boldsymbol{\beta}} \quad (6)$$

with  $\bar{\boldsymbol{\beta}} = \mathbf{W}^{-1} \boldsymbol{\beta} = r_w \mathbf{C}_z (\mathbf{I}_d + r_w^2 \mathbf{C}_z)^{-1} \boldsymbol{\theta}$ , and  $\boldsymbol{\Lambda}_d \in \mathbb{R}^{d \times d}$  as the truncation of the population eigenvalue matrix  $\boldsymbol{\Lambda}$  to the first  $d$  dimensions. The *asymptotic variance* term is

$$\begin{aligned} \text{Var}_\gamma(\hat{\boldsymbol{\theta}}) &= \frac{\sigma_\nu^2}{n} \left( \text{Tr} \left[ (\mathbf{P}_k r_w^2 \mathbf{C}_z + \mathbf{I}_k) \frac{1}{\mu(\boldsymbol{\Lambda}, \gamma)} \right] \right. \\ &\quad \left. + (p - d) \int_{s_c}^{(1+\sqrt{\gamma})^2} \frac{1}{s} dF_\gamma(s) \right) \end{aligned} \quad (7)$$

with  $\mu(\boldsymbol{\Lambda}, \gamma)$  as diagonal matrix with entries  $\mu(\lambda_i, \gamma)$  as mean of the spike eigenvalue distribution,  $F_\gamma$  as the Marčenko-Pastur distribution and  $s_c$  the value in  $\mathbb{R}$  which satisfies  $\max \left( \frac{k-d}{p-d}, 0 \right) = \int_{s_c}^{(1+\sqrt{\gamma})^2} dF_\gamma(s)$ .

**Theorem 2** (Asymptotic PCR risk). In the asymptotic limit  $n, p \rightarrow \infty$ , such that  $\frac{p}{n} \rightarrow \gamma \in (0, \infty)$ , the expected risk of PCR will converge almost surely to

$$\mathbb{E}_\nu \left[ R(\hat{\boldsymbol{\theta}}) \right] \rightarrow \text{Bias}_\gamma(\hat{\boldsymbol{\theta}})^2 + \text{Var}_\gamma(\hat{\boldsymbol{\theta}}) + \sigma_\nu^2.$$

*Proof.* In the following, we split the proof into the squared bias term and the variance term.

**Squared bias term.** Let us start with the squared bias term from (4) given by  $\text{Bias}(\hat{\boldsymbol{\theta}})^2 = \boldsymbol{\beta}^\top \boldsymbol{\Pi} \mathbf{C} \boldsymbol{\Pi} \boldsymbol{\beta}$ , and with  $\boldsymbol{\beta} = r_w \mathbf{W} \mathbf{C}_z (\mathbf{I}_d + r_w^2 \mathbf{C}_z)^{-1} \boldsymbol{\theta}$  from the equivalence of the spiked covariance model to a data generator directly between features and outcomes. Further, we have that  $\mathbf{W} = \mathbf{V}_d$  with the

eigenvectors defined as  $\mathbf{V} = \mathbf{I}_p$ . Hence, since  $\mathbf{V}_d \in \mathbb{R}^{p \times d}$ , we know that the last  $p - d$  rows will be zeros only. Therefore, we can write  $\mathbf{W} = \mathbf{V}_d = \begin{bmatrix} \mathbf{V}_{d,d} \\ 0 \end{bmatrix}$  where  $\mathbf{V}_{d,d} \in \mathbb{R}^{d \times d}$  are the eigenvectors truncated to the first  $d$ -elements. Further, we use  $\bar{\beta} = \mathbf{W}^{-1}\beta$  to write

$$\text{Bias}(\hat{\theta})^2 = \bar{\beta}^\top \begin{bmatrix} \mathbf{V}_{d,d}^\top & 0 \end{bmatrix} \mathbf{\Pi C \Pi} \begin{bmatrix} \mathbf{V}_{d,d} \\ 0 \end{bmatrix} \bar{\beta} \quad (\text{C-42})$$

Using  $\Pi = (\mathbf{I}_p - \hat{\mathbf{V}}\hat{\mathbf{V}}^\top)$  and  $\mathbf{C} = \mathbf{V}\Lambda\mathbf{V}^\top$  to expand  $\mathbf{\Pi C \Pi}$  we obtain

$$\begin{aligned} \text{Bias}(\hat{\theta})^2 &= \bar{\beta}^\top \begin{bmatrix} \mathbf{V}_{d,d}^\top & 0 \end{bmatrix} (\mathbf{V}\Lambda\mathbf{V}^\top - \mathbf{V}\Lambda\mathbf{V}^\top\hat{\mathbf{V}}\hat{\mathbf{V}}^\top - \\ &\quad \hat{\mathbf{V}}\hat{\mathbf{V}}^\top\mathbf{V}\Lambda\mathbf{V}^\top + \hat{\mathbf{V}}\hat{\mathbf{V}}^\top\mathbf{V}\Lambda\mathbf{V}^\top\hat{\mathbf{V}}\hat{\mathbf{V}}^\top) \begin{bmatrix} \mathbf{V}_{d,d} \\ 0 \end{bmatrix} \bar{\beta} \end{aligned} \quad (\text{C-43})$$

$$\begin{aligned} &= \bar{\beta}^\top (\Lambda_d - \Lambda_d \mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}_d^\top \mathbf{V}_{d,d} - \mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}_d^\top \mathbf{V}_{d,d} \Lambda_d + \\ &\quad \mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}^\top \mathbf{V}\Lambda\mathbf{V}^\top \hat{\mathbf{V}}\hat{\mathbf{V}}^\top \mathbf{V}_{d,d}) \bar{\beta} \end{aligned} \quad (\text{C-44})$$

where  $\Lambda_d$  are the first  $d$  eigenvalues and  $\hat{\mathbf{V}}_d \in \mathbb{R}^{d \times k}$  are the first  $k$  estimated eigenvectors truncated to the first  $d$  elements in each vector. The latter happens due to the multiplication with the  $\begin{bmatrix} \mathbf{V}_{d,d}^\top & 0 \end{bmatrix}$  from left and its transpose from right. For the last term, we can expand  $\Lambda = \mathbf{I}_p + r_w^2 \mathbf{W} \mathbf{C}_z \mathbf{W}^\top$ . Then we obtain

$$\mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}^\top \mathbf{V}\Lambda\mathbf{V}^\top \hat{\mathbf{V}}\hat{\mathbf{V}}^\top \mathbf{V}_{d,d} = \mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}^\top \mathbf{V} (\mathbf{I}_p + r_w^2 \mathbf{W} \mathbf{C}_z \mathbf{W}^\top) \quad (\text{C-45})$$

$$\begin{aligned} &= \mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}^\top \mathbf{V}_{d,d} + \mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}^\top \mathbf{V}_{d,d} \\ &= (r_w^2 \mathbf{C}_z) \mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}^\top \mathbf{V}_{d,d} \end{aligned} \quad (\text{C-46})$$

With this expression we observe that all terms in (C-44) contain the term  $\mathbf{V}_{d,d}^\top \hat{\mathbf{V}}_d \hat{\mathbf{V}}^\top \mathbf{V}_{d,d}$  which is the same expression as we have for the generalization of the eigenvector shift in (5). Hence, for  $p, n \rightarrow \infty$  such that  $p/n \rightarrow \gamma$ , we obtain

$$\text{Bias}_\gamma(\hat{\theta})^2 \rightarrow \bar{\beta}^\top (\Lambda_d - \Lambda_d \mathbf{P}_k - \mathbf{P}_k \Lambda_d + \mathbf{P}_k + \mathbf{P}_k r_w^2 \mathbf{C}_z \mathbf{P}_k) \bar{\beta} \quad (\text{C-47})$$

which is the expression for the squared bias term.

**Variance term.** Let us start with the variance term from (4) given by  $\text{Var}(\hat{\theta}) = \frac{\sigma_\nu^2}{n} \text{Tr}(\hat{\mathbf{V}}^\top \mathbf{C} \hat{\mathbf{V}} \hat{\mathbf{V}}^\top \hat{\mathbf{C}}^{-1} \hat{\mathbf{V}})$ . Here, we follow a similar approach by expanding the covariance terms and exploiting the multiplication of eigen-

vectors. For the trace term, we thus obtain

$$\hat{\mathbf{V}}^\top \mathbf{C} \hat{\mathbf{V}} \hat{\mathbf{V}}^\top \hat{\mathbf{C}}^{-1} \hat{\mathbf{V}} = \hat{\mathbf{V}}^\top \mathbf{V} \Lambda \mathbf{V}^\top \hat{\mathbf{V}} \hat{\Lambda}^{-1} \quad (\text{C-48})$$

$$= \hat{\mathbf{V}}^\top \mathbf{V} \left( \mathbf{W} r_w^2 \mathbf{C}_z \mathbf{W}^\top + \mathbf{I}_p \right) \mathbf{V}^\top \hat{\mathbf{V}} \hat{\Lambda}^{-1} \quad (\text{C-49})$$

$$= \hat{\mathbf{V}}^\top \mathbf{V} \left( \begin{bmatrix} \mathbf{V} \\ 0 \end{bmatrix} r_w^2 \mathbf{C}_z \begin{bmatrix} \mathbf{V}^\top & 0 \end{bmatrix} + \mathbf{I}_p \right) \mathbf{V}^\top \hat{\mathbf{V}} \hat{\Lambda}^{-1} \quad (\text{C-50})$$

$$= \left( \hat{\mathbf{V}}^\top \mathbf{V} d r_w^2 \mathbf{C}_z \mathbf{V}_d^\top \hat{\mathbf{V}} + \mathbf{I}_k \right) \hat{\Lambda}^{-1} \quad (\text{C-51})$$

Here, we can notice that the expression  $\hat{\mathbf{V}}^\top \mathbf{V}_d$  can give us part of the eigen-vector shift equation from (5) to obtain

$$\hat{\mathbf{V}}^\top \mathbf{C} \hat{\mathbf{V}} \hat{\mathbf{V}}^\top \hat{\mathbf{C}}^{-1} \hat{\mathbf{V}} \rightarrow \left( \begin{bmatrix} \mathbf{P}_k^{1/2} \\ 0 \end{bmatrix} r_w^2 \mathbf{C}_z \begin{bmatrix} \mathbf{P}_k^{1/2} & 0 \end{bmatrix} + \mathbf{I}_k \right) \hat{\Lambda}^{-1} \quad (\text{C-52})$$

since both  $\mathbf{C}_z$  and  $\mathbf{P}_k$  are diagonal, we can write

$$\hat{\mathbf{V}}^\top \mathbf{C} \hat{\mathbf{V}} \hat{\mathbf{V}}^\top \hat{\mathbf{C}}^{-1} \hat{\mathbf{V}} \rightarrow \begin{bmatrix} \mathbf{P}_k r_w^2 \mathbf{C}_z + \mathbf{I}_d & 0 \\ 0 & \mathbf{I}_{k-d} \end{bmatrix} \hat{\Lambda}^{-1}. \quad (\text{C-53})$$

Including this into the full variance term, and considering each term in the trace individually, we obtain

$$\text{Var}_\gamma(\hat{\theta}) = \frac{\sigma_\nu^2}{n} \left( \sum_{i=1}^{\min(d,k)} (\mathbf{P}_k r_w^2 \mathbf{C}_z + \mathbf{I}_k)_{[i]} \frac{1}{\hat{\lambda}_i} + \sum_{i=\min(d,k)+1}^k \frac{1}{\hat{\lambda}_i} \right) \quad (\text{C-54})$$

where  $\mathbf{A}_{[i]}$  denotes the  $i$ th diagonal element of  $\mathbf{A}$ . Here, we notice that the first term corresponds to the spike eigenvalues since the sum goes only over the top  $\min(d, k)$  eigenvalues and the second summation goes over the remaining eigenvalues including all noise terms. Hence, the first term can be written as

$$\sum_{i=1}^{\min(d,k)} (\mathbf{P}_k r_w^2 \mathbf{C}_z + \mathbf{I}_k)_{[i]} \frac{1}{\hat{\lambda}_i} = \text{Tr} \left[ (\mathbf{P}_k r_w^2 \mathbf{C}_z + \mathbf{I}_k) \frac{1}{\mu(\Lambda, \gamma)} \right] \quad (\text{C-55})$$

where  $\mu(\Lambda, \gamma)$  is a diagonal matrix with entries  $\mu(\lambda_i, \gamma)$  as mean of the spike eigenvalue distribution. For the second term, we can write the sum over the eigenvalues  $\hat{\lambda}$  as the integral over the spectral measure  $F_{\hat{\mathbf{C}}_{MP}}$  of the covariance for the Marčenko-Pastur distribution  $\hat{\mathbf{C}}_{MP}$ .

$$\sum_{i=\min(d,k)+1}^k \frac{1}{\hat{\lambda}_i} = (p - d) \int_{s_c}^{(1+\sqrt{\gamma})^2} \frac{1}{s} dF_{\hat{\mathbf{C}}_{MP}}(s) \quad (\text{C-56})$$

Paper III – No Double Descent in Principal Component Regression: A High-Dimensional Analysis

Since this term is only over the noise terms, the integral upper bound is given by the upper bound of the Marčenko-Pastur distribution  $(1 + \sqrt{\gamma})^2$ . The integral lower bound  $s_c$  corresponds to the  $p - k$  largest eigenvalue. There is a scaling factor of  $p - d$  as this is the number of eigenvalues corresponding to the part of the eigenvalue distribution. Now, we know that in the limit  $p, n \rightarrow \infty$  such that  $p/n \rightarrow \gamma$  the spectral measure will almost surely converge to the Marčenko-Pastur distribution  $F_\gamma$ . Therefore we obtain

$$\sum_{i=\min(d,k)+1}^k \frac{1}{\hat{\lambda}_i} \rightarrow (p - d) \int_{s_c}^{(1+\sqrt{\gamma})^2} \frac{1}{s} dF_\gamma(s) \quad (\text{C-57})$$

There are two steps to solve this integral. First, we need to find out the lower integral bound  $s_c$  and second, solve the integral itself. For  $s_c = -\infty$ , one can use the closed-form solution of the Stieltjes transformation  $\varphi(z)$  of the Marčenko-Pastur distribution and evaluate it at  $z = 0$ . However, there is no known closed-form solution for general  $s_c$ . We therefore solve this part numerically.

*Step 1 obtain the lower bound  $s_c$ :* We can view the spectral measure as  $F_{\hat{C}_{MP}}$  as a series of  $(p - d)$  impulses at  $s_i$  with magnitude  $1/p$  (as the sum is normalized to 1). Since we only consider the  $k - d$  largest eigenvalues (the remaining ones are considered in the first term of the trace for the spike part of the covariance), we know that their sum is  $(k - d)/(p - d)$ , see Figure 11a. This sum is the same as the integral from  $s_c$  over the Marčenko-Pastur distribution, see Figure 11b. Therefore we can find the lower integral bound  $s_c$  by solving the following numerically

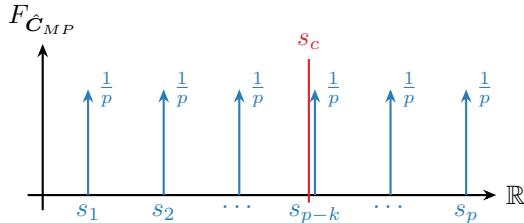
$$\max \left( \frac{k - d}{p - d}, 0 \right) = \int_{s_c}^{(1+\sqrt{\gamma})^2} dF_\gamma(s). \quad (\text{C-58})$$

Here, the max is necessary as we could have  $k \leq d$ . Then  $s_c$  will become  $(1 + \sqrt{\gamma})^2$  which means that the Marčenko-Pastur part will not contribute to the variance term.

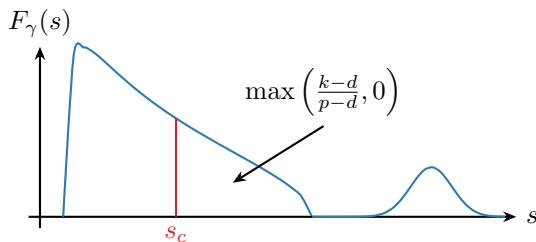
*Step 2 solve integral of interest:* Given the lower bound  $s_c$ , we can solve the integral numerically from  $s_c$  to the upper bound  $(1 + \sqrt{\gamma})^2$ . Therefore, we obtain a solution for the second term, which is not based on data but on the properties of our data matrix, especially  $\gamma$ ,  $d$  and  $k$ . This concludes the full proof for the asymptotics of the parameter norm.  $\square$

## Risk of baseline methods

Let us first re-state the risk for full regression. Note that this is a known result from Hastie et al. [Has+22, Lemma 1]. Hence, we refer to the original



(a)



(b)

**Figure 11:** (a) spectral measure impulses and lower integral bound of integral  $s_c$ . (b) Illustration of spiked covariance distribution for  $\gamma = 0.3$  with specific lower integration bound.

reference for the proof.

$$\mathbb{E}_\nu [R(\hat{\beta})] = \beta^\top \Pi C \Pi \beta + \frac{\sigma_\nu^2}{n} \text{Tr}(C \hat{C}^{-1}) + \sigma_\nu^2. \quad (8)$$

**Asymptotic full regression risk.** For the asymptotic full regression risk, we have  $k = p$ . We first re-state the theorem:

**Theorem 3** (Asymptotic full regression risk). In the asymptotic limit  $n, p \rightarrow \infty$ , such that  $\frac{p}{n} \rightarrow \gamma \in (0, \infty)$ , the expected risk of the full regression model will converge almost surely to

$$\mathbb{E}_\nu [R(\hat{\beta})] \rightarrow \text{Bias}_\gamma(\hat{\beta})^2 + \text{Var}_\gamma(\hat{\beta}) + \sigma_\nu^2$$

with the asymptotic squared bias term as  $\text{Bias}_\gamma(\hat{\beta})^2 = 0$  for  $\gamma < 1$  and  $\text{Bias}_\gamma(\hat{\beta})^2$  as in Theorem 2 with  $k = p$  for  $\gamma \geq 1$ ; and the variance term  $\text{Var}_\gamma(\hat{\beta})$  equal to the definition of the variance in Theorem 2 with  $k = p$ .

*Proof.* For the *asymptotic variance* term, we cannot simplify the results from Theorem 2 except for the case that  $\hat{V}\hat{V}^\top = \mathbf{I}$  here because the eigenvectors are not truncated.

For the *asymptotic squared bias* term, we know in the full regression model that  $\Pi$  is a projection matrix onto the null space of  $\mathbf{X}$ . Hence, we have equally to the asymptotic result from Hastie et al. [Has+22, Theorem 1] that

$$\text{Bias}_\gamma(\hat{\beta})^2 \rightarrow \begin{cases} 0 & \gamma < 1 \\ \boldsymbol{\beta}^\top \Pi C \Pi \boldsymbol{\beta} & \gamma \geq 1 \end{cases} \quad (\text{C-59})$$

which concludes the proof as we cannot simplify the expression further, given the non-isotropic covariance matrix.  $\square$

### C.3 Analysis under covariate shift

#### Risk of PCR

**Expected covariate shifted risk of PCR.** Let us first re-state the expected risk result for PCR under covariate shift:

$$\mathbb{E}_{\nu_T} [R(\hat{\theta})] = (\boldsymbol{\beta}_T - \Phi \boldsymbol{\beta})^\top \mathbf{C}_T (\boldsymbol{\beta}_T - \Phi \boldsymbol{\beta}) \quad (\text{C-60})$$

$$+ \frac{\sigma_\nu^2}{n} \text{Tr} \left( \hat{\mathbf{V}}^\top \mathbf{C}_T \hat{\mathbf{V}} \hat{\mathbf{V}}^\top \hat{\mathbf{C}}_S^{-1} \hat{\mathbf{V}} \right) + \sigma_T^2, \quad (9)$$

with  $\boldsymbol{\beta}_T = r_w \mathbf{W} \mathbf{C}_{z,T} (\mathbf{I}_d + r_w^2 \mathbf{C}_{z,T})^{-1} \boldsymbol{\theta}$  and  $\sigma_T^2 = \sigma_\varepsilon^2 + \boldsymbol{\theta}^\top (\mathbf{I}_d + r_w^2 \mathbf{C}_{z,T})^{-1} \mathbf{C}_{z,T} \boldsymbol{\theta}$ . (9)

*Proof.* First, we estimate the parameters which is done using source/training data which is equal to the non-covariate shifted case. Then, we define the risk as the expectation over the mean squared error over test data and follow a similar derivation as for (4):

$$R(\hat{\theta}) = \mathbb{E}_{(\mathbf{x}_T, y_T) \sim \mathcal{D}_T} [(y_T - \hat{y}_T(\mathbf{x}_T))^2] \quad (\text{C-61})$$

$$= \mathbb{E}_{\mathbf{x}_T} \left[ (\boldsymbol{\beta}_T^\top \mathbf{x}_T + \nu_T - \hat{y}_T(\mathbf{x}_T))^2 \right] \quad (\text{C-62})$$

$$= \mathbb{E}_{\mathbf{x}_T} \left[ \left( (\boldsymbol{\beta}_T - \hat{\mathbf{V}} \hat{\theta})^\top \mathbf{x}_T + \nu_T \right)^2 \right] \quad (\text{C-63})$$

$$= (\boldsymbol{\beta}_T - \hat{\mathbf{V}} \hat{\theta})^\top \mathbf{C}_T (\boldsymbol{\beta}_T - \hat{\mathbf{V}} \hat{\theta}) + \nu_T \nu_T^\top \quad (\text{C-64})$$

We consider again the orthogonal projector  $\Phi = \hat{\mathbf{V}} \hat{\mathbf{V}}^\top$  to rewrite the following:

$$\boldsymbol{\beta}_t - \hat{\mathbf{V}} \hat{\theta} = \boldsymbol{\beta}_T - \Phi \boldsymbol{\beta} - \hat{\mathbf{V}} \hat{\mathbf{S}}^{-1} \hat{\mathbf{U}}^\top \nu. \quad (\text{C-65})$$

Finally, using the previous two results following the same derivation as for the non-covariate shifted risk, we can write the expected risk w.r.t. the noise as

$$\mathbb{E}_{\nu_T} [R(\hat{\theta})] = (\beta_T - \Phi\beta)^\top C_T (\beta_T - \Phi\beta) \quad (C-66)$$

$$+ \frac{\sigma_\nu^2}{n} \text{Tr} \left( \hat{V}^\top C_T \hat{V} \hat{V}^\top \hat{C}_S^{-1} \hat{V} \right) + \sigma_T^2, \quad (C-67)$$

which concludes the proof.  $\square$

**Covariate shifted asymptotic PCR risk.** In order to prove Theorem 4, we re-state the results for the squared bias and the variance terms first: The *asymptotic squared bias* term is given by

$$\text{Bias}_{\gamma,T}(\hat{\theta})^2 = [\bar{\beta}_T^\top \bar{\beta}^\top] \begin{bmatrix} \Lambda_{d,T} & -\Lambda_{d,T} P_k \\ -P_k \Lambda_{d,T} & P_k + P_k r_w^2 C_{z,T} P_k \end{bmatrix} \begin{bmatrix} \bar{\beta}_T \\ \bar{\beta} \end{bmatrix} \quad (10)$$

with  $\bar{\beta} = W^{-1}\beta$ ,  $\bar{\beta}_T = W^{-1}\beta_T$  and  $\Lambda_{d,T} \in \mathbb{R}^{d \times d}$  as the truncation of  $\Lambda_T$  to the first  $d$  dimensions. The *asymptotic variance* term is

$$\text{Var}_{\gamma,T}(\hat{\theta}) = \frac{\sigma_\nu^2}{n} \left( \text{Tr} \left[ (P_k r_w^2 C_{z,T} + I_k) \frac{1}{\mu(\Lambda, \gamma)} \right] \right. \quad (C-68)$$

$$\left. + (p-d) \int_{s_c}^{(1+\sqrt{\gamma})^2} \frac{1}{s} dF_\gamma(s) \right) \quad (11)$$

with  $\mu(\Lambda, \gamma)$  as diagonal matrix with entries  $\mu(\lambda_i, \gamma)$  as mean of the spike eigenvalue distribution,  $F_\gamma$  as the Marčenko-Pastur distribution and  $s_c$  the value in  $\mathbb{R}$  which satisfies  $\max \left( \frac{k-d}{p-d}, 0 \right) = \int_{s_c}^{(1+\sqrt{\gamma})^2} dF_\gamma(s)$ .

**Theorem 4** (Covariate-shifted asymptotic PCR risk). In the asymptotic limit  $n, p \rightarrow \infty$ , such that  $\frac{p}{n} \rightarrow \gamma \in (0, \infty)$ , the expected risk of PCR under covariate shift will converge almost surely to

$$\mathbb{E}_{\nu_T} [R(\hat{\theta})] \rightarrow \text{Bias}_{\gamma,T}(\hat{\theta})^2 + \text{Var}_{\gamma,T}(\hat{\theta}) + \sigma_T^2.$$

*Proof.* We split the proof into the squared bias and variance term.

**Squared bias term.** We start with the squared bias term  $\text{Bias}_T(\beta)^2 = (\beta_T - \Phi\beta)^\top C_T (\beta_T - \Phi\beta)$  from (9) and multiply out the terms while using the definition of  $\beta = W\bar{\beta}$  and similarly for the  $\beta_T$

$$\begin{aligned} \text{Bias}_T(\beta)^2 &= \bar{\beta}_T^\top W^\top C_T W \bar{\beta}_T \\ &\quad - \bar{\beta}_T^\top W^\top C_T \hat{V} \hat{V}^\top W \bar{\beta} \\ &\quad - \bar{\beta}^\top W^\top \hat{V} \hat{V}^\top C_T W \bar{\beta}_T \\ &\quad + \bar{\beta}^\top W^\top \hat{V} \hat{V}^\top C_T \hat{V} \hat{V}^\top W \bar{\beta} \end{aligned} \quad (C-69)$$

where we can use the same results about the asymptotic eigenvector shifts  $\mathbf{P}_k$  when expanding the covariance  $\mathbf{C}_T = \mathbf{U}\Lambda_T\mathbf{V}^T$ . Then, we can summarize the terms into

$$\text{Bias}_{\gamma,T}(\boldsymbol{\beta})^2 \rightarrow [\bar{\boldsymbol{\beta}}_T^\top \quad \bar{\boldsymbol{\beta}}^\top] \begin{bmatrix} \Lambda_{d,T} & -\Lambda_{d,T}\mathbf{P}_k \\ -\mathbf{P}_k\Lambda_{d,T} & \mathbf{P}_k + \mathbf{P}_k r_w^2 \mathbf{C}_{z,T} \mathbf{P}_k \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{\beta}}_T \\ \bar{\boldsymbol{\beta}} \end{bmatrix} \quad (\text{C-70})$$

Which yields the result for the squared bias term.

**Variance term.** We start with the variance term  $\text{Var}_T(\boldsymbol{\beta}) = \frac{\sigma_\nu^2}{n} \text{Tr}(\hat{\mathbf{V}}^\top \mathbf{C}_T \hat{\mathbf{V}} \hat{\mathbf{C}}_S^{-1} \hat{\mathbf{V}})$  from (9). Let us focus on the Trace part first

$$\hat{\mathbf{V}}^\top \mathbf{C}_T \hat{\mathbf{V}} \hat{\mathbf{V}}^\top \hat{\mathbf{C}}^{-1} \hat{\mathbf{V}} = \hat{\mathbf{V}}^\top \mathbf{V} \Lambda_T \mathbf{V}^\top \hat{\mathbf{V}} \hat{\Lambda}^{-1} \quad (\text{C-71})$$

$$= \hat{\mathbf{V}}^\top \mathbf{V} \left( \mathbf{W} r_w^2 \mathbf{C}_{z,T} \mathbf{W}^\top + \mathbf{I}_p \right) \mathbf{V}^\top \hat{\mathbf{V}} \hat{\Lambda}^{-1} \quad (\text{C-72})$$

$$= \left( \hat{\mathbf{V}}^\top \mathbf{V} d r_w^2 \mathbf{C}_{z,T} \mathbf{V}^\top \hat{\mathbf{V}} + \mathbf{I}_k \right) \hat{\Lambda}^{-1} \quad (\text{C-73})$$

Again, we utilize the results from the eigenvector shift equation from (5) to obtain

$$\hat{\mathbf{V}}^\top \mathbf{C}_T \hat{\mathbf{V}} \hat{\mathbf{V}}^\top \hat{\mathbf{C}}^{-1} \hat{\mathbf{V}} \rightarrow \left( \begin{bmatrix} \mathbf{P}_k^{1/2} \\ 0 \end{bmatrix} r_w^2 \mathbf{C}_{z,T} \begin{bmatrix} \mathbf{P}_k^{1/2} & 0 \end{bmatrix} + \mathbf{I}_k \right) \hat{\Lambda}^{-1} \quad (\text{C-74})$$

Here, we notice that the main difference to the non-covariate shifted result lies in the first part where we use  $\mathbf{C}_{z,T}$  instead of  $\mathbf{C}_z$ . Therefore, using the same arguments as in the proof for Theorem 2 we obtain the final result for the variance term as

$$\begin{aligned} \text{Var}_{\gamma,T}(\boldsymbol{\beta}) &\rightarrow \frac{\sigma_\nu^2}{n} \left( \text{Tr} \left[ (\mathbf{P}_k r_w^2 \mathbf{C}_{z,T} + \mathbf{I}_k) \frac{1}{\mu(\Lambda, \gamma)} \right] \right. \\ &\quad \left. + (p-d) \int_{s_c}^{(1+\sqrt{\gamma})^2} \frac{1}{s} dF_\gamma(s) \right) \end{aligned} \quad (\text{C-75})$$

which concludes the proof.  $\square$

## Risk of baseline methods

**Theorem 5** (Covariate-shifted asymptotic full regression risk). In the asymptotic limit  $n, p \rightarrow \infty$ , such that  $\frac{p}{n} \rightarrow \gamma \in (0, \infty)$ , the expected risk of the full regression model under covariate shift will converge almost surely to

$$\mathbb{E}_\nu \left[ R(\hat{\boldsymbol{\beta}}) \right] \rightarrow \text{Bias}_{\gamma,T}(\hat{\boldsymbol{\beta}})^2 + \text{Var}_{\gamma,T}(\hat{\boldsymbol{\beta}}) + \sigma_T^2$$

with the asymptotic squared bias term as

$$\text{Bias}_{\gamma,T}(\hat{\beta})^2 = (\bar{\beta}_T - \bar{\beta})^\top \Lambda_{d,T}(\bar{\beta}_T - \bar{\beta})$$

for  $\gamma < 1$  and as in Theorem 4 for  $\gamma \geq 1$  with  $k = p$ . The variance term  $\text{Var}_{\gamma}(\hat{\beta})$  is equal to the definition of the variance in Theorem 4 with  $k = p$ .

*Proof.* The variance term and the squared bias term for  $\gamma \geq 1$  are equal to the proof for Theorem 4. For the bias-square term for  $\gamma < 1$  we have that  $\text{Bias}_T(\beta)^2 = (\beta_T - \Phi\beta)^\top C_T(\beta_T - \Phi\beta)$ . Now, we can follow our derivation for the squared bias term of Theorem 2 with  $P_k = I$  since we do not truncate the eigenvectors  $\hat{V}$  in this full regression case. Hence, we expand  $C = U\Lambda V^\top$  to obtain

$$\text{Bias}_{\gamma,T}(\hat{\beta})^2 \rightarrow (\bar{\beta}_T - \bar{\beta})^\top \Lambda_{d,T}(\bar{\beta}_T - \bar{\beta}) \quad (\text{C-76})$$

since  $\Phi = \hat{V}\hat{V}^\top = I$  for  $\gamma < 1$ . This concludes the proof.  $\square$



# Paper IV

## Title

Deep State Space Models for Nonlinear System Identification

## Authors

Daniel Gedon, Niklas Wahlström, Thomas B. Schön, Lennart Ljung

## Edited version of

D. Gedon, N. Wahlström, T. B. Schön, and L. Ljung. “Deep State Space Models for Nonlinear System Identification.” In: *19th IFAC Symposium on System Identification (SYSID)*. Padova, Italy, 2021



# Deep State Space Models for Nonlinear System Identification

## Abstract

Deep state space models (SSMs) are an actively researched model class for temporal models developed in the deep learning community which have a close connection to classic SSMs. The use of deep SSMs as a black-box identification model can describe a wide range of dynamics due to the flexibility of deep neural networks. Additionally, the probabilistic nature of the model class allows the uncertainty of the system to be modelled. In this work a deep SSM class and its parameter learning algorithm are explained in an effort to extend the toolbox of nonlinear identification methods with a deep learning based method. Six recent deep SSMs are evaluated in a first unified implementation on nonlinear system identification benchmarks.

## 1 Introduction

System identification is a well-established area of automatic control, see [ÅE71; Lju99]. A wide range of identification methods have been developed for parametric and non-parametric models as well as for grey-box and black-box models. Contrary, the field of machine learning and deep learning has emerged as the new standard in many disciplines to model highly complex systems, see [GBC16]. Deep learning can identify and capture patterns as a black-box model. It has been shown to be useful for high dimensional and nonlinear problems emerging in diverse areas such as image analysis, time series modelling, speech recognition and text classification. This paper provides one step to combine the areas of system identification and deep learning by showing the usefulness of deep SSMs applied to nonlinear system identification. It helps to bridge the gap between the fields and to learn from each others advances.

Nowadays, a wide range of system identification algorithms for parametric models are available. Parametric models such as SSMs can include pre-existing knowledge about the structure of the system and can result in precise identification. For automatic control this is a popular model class and a variety of identification algorithms is available, e.g. [SWN11].

In deep learning recent advances in the development of deep SSMs have been made, e.g. [BO15; Chu+15; Fra+16]. The class of deep SSMs has three main advantages. (1) Compared with SSMs it is more flexible due to the use of Neural Networks (NNs). (2) In many cases it can be more expressive for temporal data than feedforward NNs because of its recurrent structure including hidden states. (3) Deep SSMs can capture the output uncertainty. These advantages have been exploited for the generation of handwritten text by [LB05] and speech by [Pra+13]. The examples have highly nonlinear dynamics and require accurate uncertainty quantification to generate new realistic sequences. Our main contributions are:

- Bring the communities of system identification and deep learning closer by rigorously elaborating a deep learning model class and its learning algorithm, while applying it to nonlinear system identification problems. It extends the toolbox of possible identification approaches with a new class of deep learning models. This paper complements the work by [And+19], where deterministic NNs are applied to nonlinear system identification.
- In system identification there is a clear separation of model structure and parameter estimation. In this paper the same distinction between model structure (Section 2) and parameter learning (Section 3) is taken as a future guideline for deep learning.
- Six deep SSMs are compared in a unified implementation for nonlinear system identification (Section 4). The model advantages are highlighted by showing that a maximum likelihood estimate is obtained and additionally the uncertainty is captured which is beneficial in robust control or system analysis.

## 2 Deep state space models for sequential data

Sequence modeling is an active topic in deep learning as motivated by the temporal nature of the physical environment. A dynamic model is required to encode the system dynamics. The model is identified from observed input-output pairs  $\{(\mathbf{u}_t, \mathbf{y}_t)\}_{t=1}^T$  to predicted outputs  $\hat{\mathbf{y}}_t$ . An SSM is obtained if the computations are performed via a latent variable  $\mathbf{h}$  that incorporates past information:

$$\mathbf{h}_t = f_\theta(\mathbf{h}_{t-1}, \mathbf{u}_t, \mathbf{y}_t), \quad (1a)$$

$$\hat{\mathbf{y}}_t = g_\theta(\mathbf{h}_t), \quad (1b)$$

where  $\theta$  are unknown parameters. If the functions  $f_\theta(\cdot)$  and  $g_\theta(\cdot)$  are described by deep mappings such as deep NNs, the resulting model is referred to as a deep SSM.

Another deep learning research direction is that of *generative models* involving generative adversarial networks (GANs) by [Goo+14] and Variational Autoencoders (VAEs) by [KW14], which are used to learn representations of the data and generate new instances from the same distribution, such as realistic images. Extending VAEs to sequential models such as in [Fra18] yields the subclass of deep SSM models which are studied in this paper. The building blocks for these models are Recurrent NNs (RNNs) and VAEs.

## 2.1 Recurrent neural networks

RNNs are useful in modeling sequences of variable length. Models with external inputs  $\mathbf{u}_t$  and outputs  $\mathbf{y}_t$  at each time step are considered. RNNs make use of a hidden state  $\mathbf{h}_t = f_\theta(\mathbf{h}_{t-1}, \mathbf{u}_t)$ . The function parameters are learned by unfolding the RNN and using backpropagation through time. The most notable types of RNNs for long-term dependencies are Long Short-Term Memory (LSTM) networks by [HS97] and Gated Recurrent Units (GRUs) by [Cho+14], which yield empirically similar results. GRUs are used in this paper due to their structural simplicity.

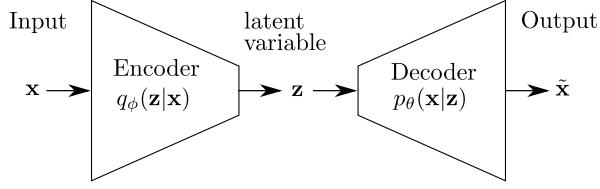
## 2.2 Variational autoencoders

A VAE embeds a representation of the data distribution of  $\mathbf{x}$  in a low dimensional latent variable  $\mathbf{z}$  via an inference network (encoder). A decoder network uses  $\mathbf{z}$  to generate new data  $\tilde{\mathbf{x}}$  of approximately the same distribution as  $\mathbf{x}$ . The conceptual idea of a VAE is visualized in Figure 1 and can be viewed as a latent variable model through  $\mathbf{z}$ .

Within VAEs it is generally assumed that the data  $\mathbf{x}$  can be modelled by a normal distribution. The decoder is chosen accordingly as  $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}^{\text{dec}}, \boldsymbol{\sigma}^{\text{dec}})$ . The parameters for this distribution are given by  $[\boldsymbol{\mu}^{\text{dec}}, \boldsymbol{\sigma}^{\text{dec}}] = \text{NN}_\theta^{\text{dec}}(\mathbf{z})$  as a deep NN with parameters  $\theta$ , input  $\mathbf{z}$  and outputs  $\boldsymbol{\mu}^{\text{dec}}$  and  $\boldsymbol{\sigma}^{\text{dec}}$ . The generative model is characterized by the joint distribution  $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$ , where the multivariate normal distribution  $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}^{\text{prior}}, \boldsymbol{\sigma}^{\text{prior}})$  is used as prior. The prior parameters are usually chosen as  $[\boldsymbol{\mu}^{\text{prior}}, \boldsymbol{\sigma}^{\text{prior}}] = [\mathbf{0}, \mathbf{I}]$ .

For the data embedding in  $\mathbf{z}$ , the distribution of interest is the posterior  $p(\mathbf{z}|\mathbf{x})$  which is intractable in general. It is approximated by a parametric distribution  $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}^{\text{enc}}, \boldsymbol{\sigma}^{\text{enc}})$ . The distribution parameters are encoded by a deep NN  $[\boldsymbol{\mu}^{\text{enc}}, \boldsymbol{\sigma}^{\text{enc}}] = \text{NN}_\phi^{\text{enc}}(\mathbf{x})$ . This network is optimized by variational inference of the variational parameters  $\phi$  shared over all data points, [BKM17].

Notably, there exists a connection between the VAE and linear dimension reduction methods such as PCA. In [Row98] it is shown that the PCA corre-



**Figure 1:** Conceptual idea of the VAE.

sponds to a linear Gaussian model. Specifically, the VAE can be viewed as a nonlinear generalization of the probabilistic PCA.

### 2.3 Combining RNNs and VAEs into deep SSMs

To obtain a deep SSM we combine RNNs with VAEs, see Figure 2 for concrete examples. The RNN can be viewed as a special case of classic SSMs with Dirac delta functions as state transition distribution  $\tilde{p}(\mathbf{h}_t|\mathbf{h}_{t-1})$  compare with (1a) or [Fra18]. The VAE can be used to approximate the output distributions of the dynamics from the RNN output, see (1b). A temporal extension of the VAE is required for the studied class of deep SSMs. The parameters of the VAE prior are updated sequentially with the output  $\mathbf{z}_t$  of the RNN as  $[\mu_t^{\text{prior}}, \sigma_t^{\text{prior}}] = \text{NN}_\theta^{\text{prior}}(\mathbf{z}_{t-1}, \mathbf{u}_t)$ . The state transition distribution is given by  $p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_t) = \mathcal{N}(\mathbf{z}_t|\mu_t^{\text{prior}}, \sigma_t^{\text{prior}})$ . Note that compared with the VAE prior, the parameters  $\mu_t^{\text{prior}}, \sigma_t^{\text{prior}}$  are now not static but dependent on previous time steps and describe the recurrence of the model. Similarly the output distribution is given as  $p_\theta(\mathbf{y}_t|\mathbf{z}_t) = \mathcal{N}(\mathbf{y}_t|\mu_t^{\text{dec}}, \sigma_t^{\text{dec}})$  with  $[\mu_t^{\text{dec}}, \sigma_t^{\text{dec}}] = \text{NN}_\theta^{\text{dec}}(\mathbf{z}_t)$ . The joint distribution of the deep SSM is

$$p_\theta(\mathbf{y}_{1:T}, \mathbf{z}_{1:T} | \mathbf{u}_{1:T}, \mathbf{z}_0) = \prod_{t=1}^T p_\theta(\mathbf{y}_t | \mathbf{z}_t) p_\theta(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{u}_t). \quad (2)$$

Similar to the VAE, this expression describes the generative process. It can be further decomposed with a clear separation between the RNN and the VAE which yields the most simple form within the studied class of deep SSMs, the so-called VAE-RNN from [Fra18]. The model consists of stacking a VAE on top of an RNN as shown in Figure 2. Notice the clear separation between model parameter learning in the inference network with the available data  $\{(\mathbf{u}_t, \mathbf{y}_t)\}_{t=1}^T$  and the output prediction  $\mathbf{y}_t$  in the generative network. The joint true posterior can be factorized according to the graphical model as

$$p_\theta(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}, \mathbf{h}_{1:T} | \mathbf{u}_{1:T}, \mathbf{h}_0) = p_\theta(\mathbf{y}_{1:T} | \mathbf{z}_{1:T}) p_\theta(\mathbf{z}_{1:T} | \mathbf{h}_{1:T}) \tilde{p}(\mathbf{h}_{1:T} | \mathbf{u}_{1:T}, \mathbf{h}_0), \quad (3)$$

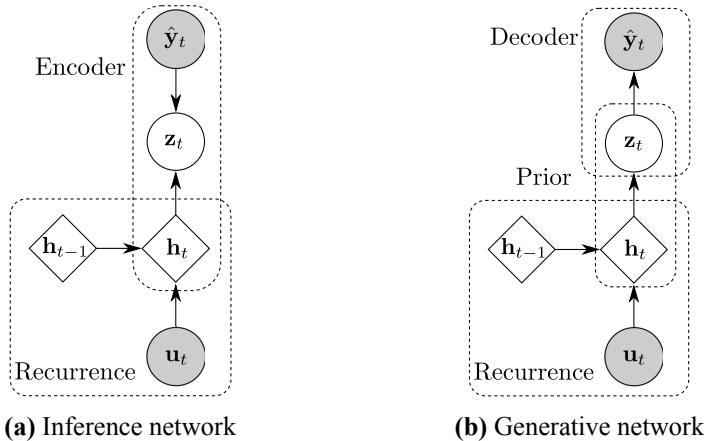
with prior given by  $p_\theta(\mathbf{z}_t|\mathbf{h}_t) = \mathcal{N}(\mathbf{z}_t|\boldsymbol{\mu}_t^{\text{prior}}, \boldsymbol{\sigma}_t^{\text{prior}})$  with  $[\boldsymbol{\mu}_t^{\text{prior}}, \boldsymbol{\sigma}_t^{\text{prior}}] = \text{NN}_\theta^{\text{prior}}(\mathbf{h}_t)$  only depending on the recurrent state  $\mathbf{h}_t$ . The approximate posterior can be chosen to mimic the same factorization

$$q_\phi(\mathbf{z}_{1:T}, \mathbf{h}_{1:T} | \mathbf{y}_{1:T}, \mathbf{u}_{1:T}, \mathbf{h}_0) = q_\phi(\mathbf{z}_{1:T} | \mathbf{y}_{1:T}, \mathbf{h}_{1:T}) \tilde{p}(\mathbf{h}_{1:T} | \mathbf{u}_{1:T}, \mathbf{h}_0). \quad (4)$$

In this paper we consider variations in this class of the deep SSM next to the VAE-RNN, specifically:

- Variational RNN (VRNN) by [Chu+15]: Based on VAE-RNN but the recurrence additionally uses the previous latent variable  $\mathbf{z}_{t-1}$  for  $p_\theta(\mathbf{h}_t) = p_\theta(\mathbf{h}_t | \mathbf{h}_{t-1}, \mathbf{u}_t, \mathbf{z}_{t-1})$ .
- VRNN-I by [Chu+15]: Same as VRNN but a static prior is used  $[\boldsymbol{\mu}^{\text{prior}}, \boldsymbol{\sigma}^{\text{prior}}] = [\mathbf{0}, \mathbf{I}]$  in every time step.
- Stochastic RNN (STORN) by [BO15]: Based on the VRNN-I. In the inference network STORN additionally makes use of a forward running RNN with input  $\mathbf{y}_t$ , latent variable  $\mathbf{d}_t$  and output  $\mathbf{z}_t$ . Hence,  $\mathbf{z}_t$  is characterized by  $p_\theta(\mathbf{z}_t) = \int p_\theta(\mathbf{z}_t | \mathbf{d}_t) p_\theta(\mathbf{d}_t | \mathbf{d}_{t-1}, \mathbf{y}_t) d\mathbf{d}_t$ .

Graphical models for these extensions are provided in Appendix A. For VRNN and VRNN-I an additional version using Gaussian mixtures as output distribution (VRNN-GMM) is studied. More methods are available in literature, see e.g. [Ali+17; Fra+16].



**Figure 2:** Graphical model of the VAE-RNN model. Round blocks indicate probabilistic variables and rectangular blocks deterministic variables. Shaded blocks indicate observed variables.

### 3 Model parameter learning

#### 3.1 Cost function for the VAE

The parameter learning method of the deep SSMs is based on the method used for VAEs. The VAE parameters  $\theta$  are learned by maximum likelihood  $\mathcal{L}(\theta) = \sum_{i=1}^N \log p_\theta(x_i) = \sum_{i=1}^N \mathcal{L}_i(\theta)$  with  $N$  data points  $\{x_i\}_{i=1}^N$ . Performing variational inference with shared parameters for all data results in the following using Jensen's inequality

$$\begin{aligned}\mathcal{L}_i(\theta) &= \log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\ &= \log \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[ \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \\ &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[ \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] = \tilde{\mathcal{L}}_i(\theta, \phi).\end{aligned}\quad (5)$$

The expression  $\tilde{\mathcal{L}}_i(\theta, \phi)$  is referred to as the evidence lower bound (ELBO) and can be rewritten using the Kullback-Leibler (KL) divergence

$$\tilde{\mathcal{L}}_i(\theta, \phi) = \mathbb{E}_{q_\phi} [\log p_\theta(\mathbf{x}|\mathbf{z})] - \text{KL}(q_\phi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z})), \quad (6)$$

where the expectation is w.r.t.  $q_\phi(\mathbf{z}|\mathbf{x})$ . The first term encourages the reconstruction of the data by the decoder. The KL-divergence in the second term is a measure of closeness between the two distributions and can be interpreted as a regularization term. Approximate posteriors  $q_\phi(\mathbf{z}|\mathbf{x})$  far away from the prior  $p_\theta(\mathbf{z})$  are penalized. The ELBO is then given by  $\tilde{\mathcal{L}}(\theta, \phi) = \sum_{i=1}^N \tilde{\mathcal{L}}_i(\theta, \phi)$  which is maximized instead of the intractable log-likelihood  $\mathcal{L}(\theta)$ .

#### 3.2 Cost function for deep SSMs

A temporal extension of the VAE parameter learning is required for the studied deep SSMs. A similar derivation for the ELBO of the VAE as in (5) leads for the generic deep SSM to

$$\tilde{\mathcal{L}}(\theta, \phi) = \mathbb{E}_{q_\phi} \left[ \log \frac{p_\theta(\mathbf{y}_{1:T}, \mathbf{z}_{1:T} | \mathbf{u}_{1:T}, \mathbf{z}_0)}{q_\phi(\mathbf{z}_{1:T} | \mathbf{y}_{1:T}, \mathbf{u}_{1:T}, \mathbf{z}_0)} \right], \quad (7)$$

where the expectation is w.r.t. the approximate distribution  $q_\phi(\mathbf{z}_{1:T} | \mathbf{y}_{1:T}, \mathbf{u}_{1:T}, \mathbf{z}_0)$ . The factorization of the true joint posterior distribution from (2) can be applied which yields an ELBO as the sum over all time steps. Note that in this generic scheme  $q_\phi(\cdot)$  can be factorized as  $\prod_{t=1}^T q_\phi(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{y}_{t:T}, \mathbf{u}_{t:T})$ , which requires a smoothing step since  $\mathbf{z}_t$  depends on all inputs and outputs for all time

steps  $t = 1, \dots, T$ . If there exists a similar factorization for the approximate posterior as in (4), then an expression similar to (6) for the VAE in can be obtained.

In the VAE-RNN a solution for parameter learning is obtained by a clear separation between the RNN and the VAE. Note that here no smoothing step for the variational distribution is necessary since the states  $\mathbf{z}_{1:T}$  are independent given  $\mathbf{h}_{1:T}$  as can be seen by d-separation in Figure 2. The same factorization as in (4) can be used. The ELBO for the VAE-RNN is written as

$$\tilde{\mathcal{L}}(\theta, \phi) = \mathbb{E}_{q_\phi} \left[ \log \frac{p_\theta(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}, \mathbf{h}_{1:T} | \mathbf{u}_{1:T}, \mathbf{h}_0)}{q_\phi(\mathbf{z}_{1:T}, \mathbf{h}_{1:T} | \mathbf{y}_{1:T}, \mathbf{u}_{1:T}, \mathbf{h}_0)} \right], \quad (8)$$

where the expectation is taken w.r.t. the approximate posterior  $q_\phi(\mathbf{z}_{1:T}, \mathbf{h}_{1:T} | \mathbf{y}_{1:T}, \mathbf{u}_{1:T}, \mathbf{h}_0)$ . Applying the posterior factorizations in (3) and (4) to the ELBO in (8) and taking the expectation w.r.t.  $q_\phi(\mathbf{z}_t | \mathbf{y}_t, \mathbf{h}_t)$  yields

$$\begin{aligned} \tilde{\mathcal{L}}(\theta, \phi) &= \sum_{t=1}^T \mathbb{E}_{q_\phi} \left[ \log \frac{p_\theta(\mathbf{y}_t | \mathbf{z}_t) p_\theta(\mathbf{z}_t | \mathbf{h}_t)}{q_\phi(\mathbf{z}_t | \mathbf{y}_t, \mathbf{h}_t)} \right] \\ &= \sum_{t=1}^T \mathbb{E}_{q_\phi} [\log p_\theta(\mathbf{y}_t | \mathbf{z}_t)] - \text{KL}(q_\phi(\mathbf{z}_t | \mathbf{y}_t, \mathbf{h}_t) || p_\theta(\mathbf{z}_t | \mathbf{h}_t)), \end{aligned} \quad (9)$$

which is of the same form as the VAE ELBO in (6), but with a temporal extension summing over all time steps.

## 4 Numerical experiments

All six models described in Section 2 are evaluated. The model hyperparameters are the dimension of the hidden state  $\mathbf{z}_t$  denoted by  $z_{\text{dim}}$ , the dimension of the RNN hidden state  $\mathbf{h}_t$  denoted by  $h_{\text{dim}}$  and the number of layers within the RNN networks  $n_{\text{layer}}$ . For STORN the dimension of  $\mathbf{d}_t$  is chosen equal to that of  $\mathbf{h}_t$ . The VRNN-GMM uses five Gaussian mixtures in the output distribution. The encoder and decoder are modelled as 3-layer NN and the features of  $\mathbf{y}_t$ ,  $\mathbf{u}_t$ ,  $\mathbf{z}_t$  are extracted with 2-layer NNs.

For parameter learning, hyperparameter and model selection, the data is split in training and validation data. A separate test data set is used for evaluating the final performance. The ADAM optimizer with default parameters is used with early stopping and batch normalization, see [KB15]. The initial learning rate of  $10^{-3}$  is decreased if the validation loss plateaus. Note that the optimization parameters are not fine-tuned for any for the experiments whereas the sequence length for training is considered to be a design parameter.

Three experiments are conducted: (1) a linear Gaussian system, (2) the nonlinear Narendra-Li Benchmark from [NL96], and (3) the Wiener-Hammerstein (WH) process noise benchmark from [SN17]. The first two experiments are considered to show the power of deep SSMs for uncertainty quantification with known true uncertainty, while the last experiment serves as a more complex real world example. The identified models are evaluated in open loop. The initial state is not estimated. The generated output sequences are compared with the true test data output. As performance metric, the root mean squared error (RMSE) is considered,  $\sqrt{\frac{1}{T} \sum_{t=1}^T (\hat{y}_t - y_t)^2}$  with  $\hat{y}_t = \mu_t^{\text{dec}}$  such that a fair comparison with maximum likelihood estimation methods can be made. To quantify the quality of the uncertainty estimate, the negative log-likelihood (NLL) per time step is used,  $\frac{1}{T} \sum_{t=1}^T -\log \mathcal{N}(y_t | \mu_t^{\text{dec}}, \sigma_t^{\text{dec}})$ , describing how likely it is that the true data point falls in the model output distribution. PyTorch code is available [https://github.com/dgedon/DeepSSM\\_SysID](https://github.com/dgedon/DeepSSM_SysID).

## 4.1 Toy problem: Linear Gaussian system

Consider the following linear system with process noise  $\mathbf{v}_k \sim \mathcal{N}(0, 0.5 \cdot \mathbf{I})$  and measurement noise  $\mathbf{w}_k \sim \mathcal{N}(0, 1)$

$$\mathbf{x}_{k+1} = \begin{bmatrix} 0.7 & 0.8 \\ 0 & 0.1 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} -1 \\ 0.1 \end{bmatrix} \mathbf{u}_k + \mathbf{v}_k, \quad (10a)$$

$$\mathbf{y}_k = [1 \ 0] \mathbf{x}_k + \mathbf{w}_k. \quad (10b)$$

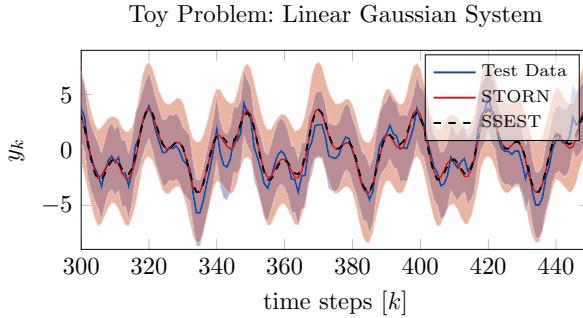
The models are trained and validated with 2 000 samples and tested on 5 000 samples. The same number of layers in the NNs is taken for all models but with different number of neurons per layer. A grid search for the selection of the best architecture is performed with  $h_{\text{dim}} = \{50, 60, 70, 80\}$  and  $z_{\text{dim}} = \{2, 5, 10\}$ . Here  $n_{\text{layer}} = 1$  is chosen due to the simplicity of the experiment. For all models the architecture with the lowest RMSE value is presented.

The deep SSMs are compared with two methods. First, a linear model is identified using SSEST from the system identification toolbox, [Lju18] with the true system order of 2. SSEST also estimates the output variance, which is used as baseline. Second, the true system matrices as best possible linear model are run in open loop without noise.

The results are listed in Table 1; the models are listed with increasing complexity. For the deep SSMs the values are averaged over 50 identified models and for the baseline methods over 500 identifications, since these methods are computationally less expensive. The results indicate that the deep SSMs can reach an accuracy close to the state of the art methods. Note that SSEST assumes a linear model, whereas the deep SSMs fit a flexible, nonlinear model. The table also shows that a more complex deep SSM yields more accurate results. An

**Table 1:** Results for linear Gaussian toy problem.

Model	RMSE	NLL	$(h_{\text{dim}}, z_{\text{dim}})$
VAE-RNN	1.56	1.95	(80,10)
VRNN-Gauss-I	1.48	1.82	(50,5)
VRNN-Gauss	1.47	1.85	(80,2)
VRNN-GMM-I	1.45	1.80	(70,10)
VRNN-GMM	1.43	1.79	(50,5)
STORN	1.43	1.79	(60,5)
SSEST	1.41	1.78	-
True lin. model	1.34	-	-



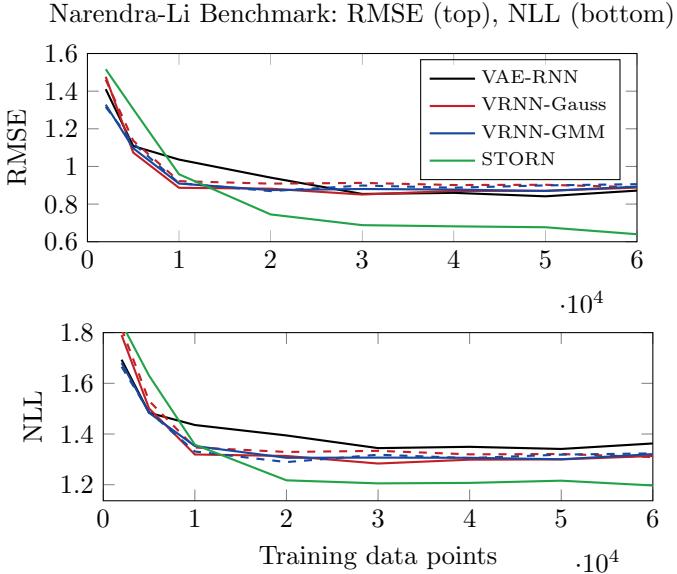
**Figure 3:** Toy problem: results of open loop run for test data, STORN (both with  $\mu \pm 3\sigma$ ) and SSEST. Shaded area depicts uncertainty.

open loop plot with mean and confidence interval of  $\pm 3$  standard deviation for the identified models by STORN and SSEST (only mean) is given in Figure 3 and compared to the ground truth. The uncertainty is captured well, but it is conservatively overestimated.

## 4.2 Narendra-Li benchmark

The dynamics of the Narendra-Li benchmark are given by [NL96] with additional measurement noise from [Ste99]. The benchmark is designed as a highly nonlinear but non-physical, fictional system. For more details, see the appendix.

This benchmark is evaluated for a varying number of training samples in [2 000; 60 000]. For each identification 5 000 validation samples and the same 5 000 test samples are used. A gridsearch is performed to choose architecture parameters, revealing, that in general it is advantageous to have larger networks.



**Figure 4:** Narendra-Li benchmark: RMSE and NLL for varying number of training data points. VRNN-Gauss-I and VRNN-GMM-I with dashed lines.

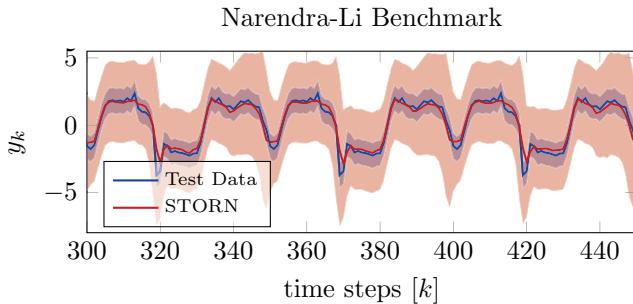
Hence, for comparability all models are run with  $h_{\text{dim}} = 60$ ,  $z_{\text{dim}} = 10$  and  $n_{\text{layer}} = 1$ . No batch normalization is applied.

The results are plotted in Figure 4 and show averaged RMSE and NLL values over 30 identified models for varying training data sizes. Generally, more training data yields more accurate estimates, both in terms of RMSE and NLL. After a specific amount of training data, the identification results stop to improve. This plateau indicates that the chosen model is saturated. Larger models could be more flexible to decrease the values even further. Specifically, the STORN model outperforms the other models, all of which show similar performance. This is due to the enhanced flexibility in STORN via the use of a second recurrent network in the inference, allowing for the learned state representations  $\mathbf{z}_t$  to be more accurate.

The lowest RMSE values of each model are in Table 2 compared with results from literature. The methods compared against do not estimate uncertainty, therefore NLL cannot be provided. Table 2 also includes the required number of samples to obtain the given performance. The table indicates that deep SSMs require more samples for learning than classic models which is in line with general deep learning experience. Despite the performance gap, we believe this research to be of interest in areas where many datapoints are available and deep SSM can provide an accurate black-box model. One reason for the performance gap can be that gray-box models from literature are compared

**Table 2:** Results for the Narendra-Li benchmark.

Model	RMSE	NLL	Samples
VAE-RNN	0.84	1.34	50 000
VRNN-Gauss-I	0.89	1.31	60 000
VRNN-Gauss	0.85	1.28	30 000
VRNN-GMM-I	0.87	1.29	20 000
VRNN-GMM	0.87	1.30	50 000
STORN	0.64	1.20	60 000
Multivariate adapt. reg. splines	0.46	-	2 000
Adapt. hinging hyperplanes	0.31	-	2 000
Model-on-demand	0.46	-	50 000
Direct weight optimization	0.43	-	50 000
Basis function expansion	0.06	-	2 000



**Figure 5:** Narendra-Li benchmark: Time evaluation of true system and STORN with uncertainties.

with deep SSMs which are black-box models. The results indicate that in particular STORN reaches RMSE values close to gray-box models.

An open-loop run for identified STORN model compared with the true data is given in Figure 5. Mean value and  $\pm 3$  standard deviations are shown. The figure highlights: First, the complex nonlinear dynamics are identified well. Second, the uncertainty bounds are captured but are much more conservative than the true bounds.

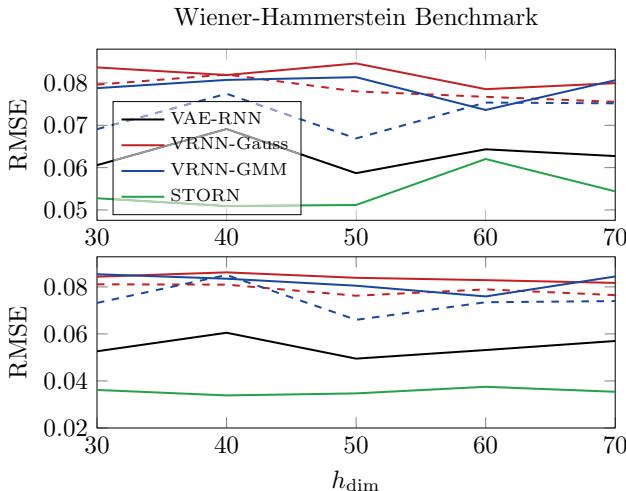
### 4.3 Wiener-Hammerstein process noise cenchmark

The WH benchmark with process noise by [SN17] provides measured input-output data from an electric circuit. The system can be described by a nonlinear WH model which has a nonlinearity between two linear dynamic systems.

Process noise enters before the nonlinearity making the benchmark particularly difficult.

The training data consist of 8 192 samples where the input is a faded multisine realization. The validation data are taken from the same data set but a different realization. The test data set consists of 16 384 samples, one multisine realization and one swept sine. Preliminary tests indicate that a longer training sequence length yield more accurate results, hence a length of 2 048 points is used. This benchmark is evaluated for varying sizes of the deep SSM layers. Here  $h_{\text{dim}} = \{30, 40, 50, 60\}$  with constant  $z_{\text{dim}} = 3$  and  $n_{\text{layer}} = 3$ .

The resulting RMSE values for the multisine and swept sine test sequence are presented in Figure 6. The lowest RMSE values of the plot are in Table 3 compared to state of the art methods from the literature. The values are presented as averages over 20 identified models. The plot indicates that the influence of  $h_{\text{dim}}$  is rather limited. Larger values and therefore larger NNs in general tend to result in more accurate identification results. Again, STORN yields the best results, while also the very simple VAE-RNN identifies this complex benchmark well. The jagged behaviour of the plot may arise since the chosen identification data set only consists of two realizations. Therefore the randomness over the multiple experiments originates mainly from random initialization of the weights in the NNs. The difference to results from literature in Table 3 could result because these methods are gray-box models and incorporate system knowledge.



**Figure 6:** WH benchmark: RMSE of for multisine (top) and swept sine (bottom) test signal for varying  $h_{\text{dim}}$ .

**Table 3:** Results in RMSE for WH benchmark.

Model	swept sine	multisine
VAE-RNN	0.050	0.059
VRNN-Gauss-I	0.076	0.076
VRNN-Gauss	0.082	0.079
VRNN-GMM-I	0.066	0.067
VRNN-GMM	0.076	0.074
STORN	0.034	0.051
NOBF	$\approx 0.2$	<0.3
NFIR	<0.05	<0.05
NARX	<0.05	$\approx 0.05$
PNLSS	0.022	0.038
Best Linear Approx.	-	0.035
ML	-	0.016
SMC	0.014	0.015

## 5 Conclusion and future work

This paper provides an introduction to deep SSMs as an extension to SSMs using highly flexible NNs and elaborates the parameter learning method based on variational inference. Six deep SSMs are implemented and applied to three system identification problems to benchmark their potential. The results indicate that the class of deep SSMs is competitive to classic identification methods. Therefore, the toolbox of nonlinear identification methods is extended by a new model class based on deep learning. Deep SSMs also estimate the uncertainty in the dynamics by its probabilistic nature, which appears to be as conservative as established uncertainty quantification methods. This conservative behavior is in line with the existing literature on variational inference of deep learning models.

This study concerns a subclass of deep SSMs based on variational inference methods. Future work should study a broader class of deep SSMs and more nonlinear system identification benchmarks should be considered. It is of high interest to use deep SSM in automatic control like e.g. model predictive control and to elaborate how to exploit the latent state variables.

**Acknowledgments** This research was partially supported by the *Wallenberg AI, Autonomous Systems and Software Program (WASP)* funded by Knut and Alice Wallenberg Foundation and the Swedish Research Council, contracts 2016-06079 and 2019-04956.

## References

- [ÅE71] K. J. Åström and P. Eykhoff. “System Identification—A Survey.” In: *Automatica* 7.2 (1971), pp. 123–162 (cit. on p. IV-1).
- [Ali+17] A. G. Alias Parth Goyal, A. Sordoni, M.-A. Côté, N. R. Ke, and Y. Bengio. “Z-Forcing: Training Stochastic Recurrent Networks.” In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 6713–6723 (cit. on p. IV-5).
- [And+19] C. Andersson, A. H. Ribeiro, K. Tiels, N. Wahlström, and T. B. Schön. “Deep Convolutional Networks in System Identification.” In: *Proceedings of the 58th IEEE Conference on Decision and Control*. Nice, France, 2019 (cit. on p. IV-2).
- [BKM17] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. “Variational Inference: A Review for Statisticians.” In: *Journal of the American Statistical Association* 112.518 (2017), pp. 859–877 (cit. on p. IV-3).
- [BO15] J. Bayer and C. Osendorfer. “Learning Stochastic Recurrent Networks.” In: *arXiv:1411.7610* (2015) (cit. on pp. IV-2, IV-5).
- [Cho+14] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches.” In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 103–111 (cit. on p. IV-3).
- [Chu+15] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. “A Recurrent Latent Variable Model for Sequential Data.” In: *Advances in Neural Information Processing Systems 28*. 2015, pp. 2980–2988 (cit. on pp. IV-2, IV-5).
- [Fra+16] M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther. “Sequential Neural Models with Stochastic Layers.” In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Barcelona, Spain, 2016, pp. 2207–2215 (cit. on pp. IV-2, IV-5).
- [Fra18] M. Fraccaro. “Deep Latent Variable Models for Sequential Data.” PhD thesis. DTU Compute, 2018 (cit. on pp. IV-3, IV-4).
- [GBC16] I. Goodfellow, Y. Bengio, and A. C. Courville. *Deep Learning*. MIT Press, 2016 (cit. on p. IV-1).
- [Goo+14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. “Generative Adversarial Nets.” In: *Advances in Neural Information Processing Systems 27*. 2014, pp. 2672–2680 (cit. on p. IV-3).

- [HS97] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory.” In: *Neural Comput.* 9.8 (1997), pp. 1735–1780 (cit. on p. IV-3).
- [KB15] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization.” In: *3rd International Conference on Learning Representations, (ICLR)*. San Diego, CA, USA, 2015 (cit. on p. IV-7).
- [KW14] D. P. Kingma and M. Welling. “Auto-Encoding Variational Bayes.” In: *Proceedings of the International Conference on Learning Representations (ICLR)*. Banff, Canada, 2014 (cit. on p. IV-3).
- [LB05] M. Liwicki and H. Bunke. “IAM-OnDB - an on-Line English Sentence Database Acquired from Handwritten Text on a Whiteboard.” In: *Eighth International Conference on Document Analysis and Recognition (ICDAR ’05)*. 2005, 956–961 Vol. 2 (cit. on p. IV-2).
- [Lju18] L. Ljung. *System Identification Toolbox: The Manual*. 9th edition 2018. Natick, MA, USA: The MathWorks Inc., 2018 (cit. on p. IV-8).
- [Lju99] L. Ljung. “System identification: theory for the user.” In: *PTR Prentice Hall, Upper Saddle River, NJ* (1999) (cit. on p. IV-1).
- [NL96] K. S. Narendra and S.-M. Li. “Neural networks in control systems.” In: Hillsdale, NJ, USA: Lawrence Erlbaum Associates, 1996. Chap. 11, pp. 347–394 (cit. on pp. IV-8, IV-9, IV-19).
- [Pra+13] K. Prahallad, A. Vadapalli, N. K. Elluru, G. V. Mantena, B. Pulu-gundla, P. Bhaskararao, H. A. Murthy, S. J. King, V. Karaikos, and A. W. Black. “The Blizzard Challenge 2013 - Indian Language Tasks.” In: 2013 (cit. on p. IV-2).
- [Row98] S. T. Roweis. “EM Algorithms for PCA and SPCA.” In: *Advances in Neural Information Processing Systems 10*. 1998, pp. 626–632 (cit. on p. IV-3).
- [SN17] M. Schoukens and J. P. Noel. “Wiener-Hammerstein Benchmark with Process Noise.” In: *20th IFAC World Congress*. 20th IFAC World Congress 50.1 (2017), pp. 448–453 (cit. on pp. IV-8, IV-11).
- [Ste99] A. Stenman. *Model on Demand: Algorithms, Analysis and Applications*. Linköping Studies in Science and Technology Dissertation 571. Linköping University, 1999 (cit. on pp. IV-9, IV-19).
- [SWN11] T. B. Schön, A. Wills, and B. Ninness. “System Identification of Nonlinear State-Space Models.” In: *Automatica* 47.1 (2011), pp. 39–49 (cit. on p. IV-1).

## Appendix

### A Graphical models for deep SSMs

In Section 2 the focus lies on the most simple deep SSM, namely the VAE-RNN. The loss function is then derived based on the graphical model for the VAE-RNN in Figure 1. Here the graphical models for all other studied deep SSMs are presented and shortly explained.

#### A.1 VRNN

The graphical model for the VRNN is shown in Figure 7. Note that the VRNN-Gauss and VRNN-GMM use the same network architecture. The difference lies in the output distribution, which is either Gaussian or a Gaussian mixture. In the VRNN the recurrence additionally makes use of the previous latent variable  $\mathbf{z}_{t-1}$ . In the generative network also the hidden state  $\mathbf{h}_t$  has a direct influence on  $\mathbf{y}_t$ . Both of these features give more flexibility for the network output distribution. The joint true posterior of the VRNN can be factorized according to the generative network in Figure 7 as

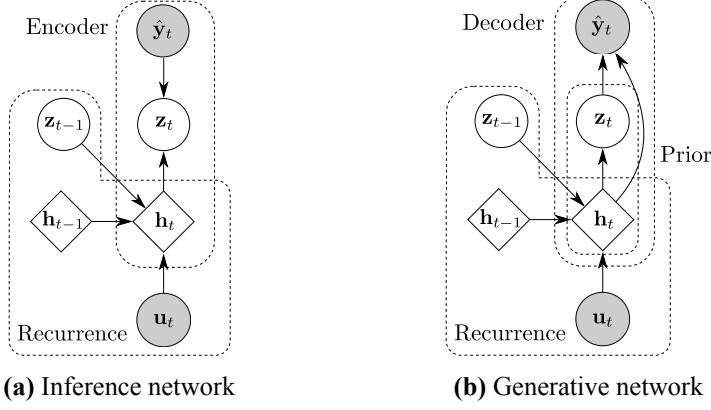
$$\begin{aligned} p_{\theta}(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}, \mathbf{h}_{1:T} | \mathbf{u}_{1:T}, \mathbf{h}_0) = & p_{\theta}(\mathbf{y}_{1:T} | \mathbf{z}_{1:T}, \mathbf{h}_{1:T}) \times \\ & \times p_{\theta}(\mathbf{z}_{1:T} | \mathbf{h}_{1:T}) \tilde{p}(\mathbf{h}_{1:T} | \mathbf{z}_{1:T}, \mathbf{u}_{1:T}, \mathbf{h}_0). \end{aligned} \quad (11)$$

The joint approximate posterior of the VRNN factorization according to the inference network as

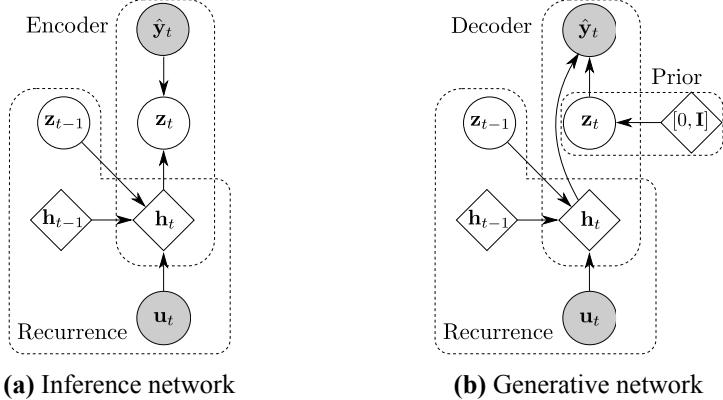
$$q_{\phi}(\mathbf{z}_{1:T}, \mathbf{h}_{1:T} | \mathbf{y}_{1:T}, \mathbf{u}_{1:T}, \mathbf{h}_0) = q_{\phi}(\mathbf{z}_{1:T} | \mathbf{y}_{1:T}, \mathbf{h}_{1:T}) \tilde{p}(\mathbf{h}_{1:T} | \mathbf{z}_{1:T}, \mathbf{u}_{1:T}, \mathbf{h}_0). \quad (12)$$

#### A.2 VRNN-I

The VRNN-I is a simple modification of the VRNN and its graphical model is shown in Figure 8. The difference to the VRNN is that the prior in the generative network is static and does not change temporally by the recurrent network. Similarly, here the VRNN-Gauss-I and VRNN-GMM-I only differ in the output distribution but not in the network structure. For the VRNN-I the same true and approximate joint posterior distributions as for the VRNN above apply with the difference in the true posterior that the prior is static  $p_{\theta}(\mathbf{z}_{1:T} | [0, \mathbf{I}])$ .



**Figure 7:** Graphical model for VRNN.



**Figure 8:** Graphical model for VRNN-I. Note that the inference network is equal to the VRNN inference network.

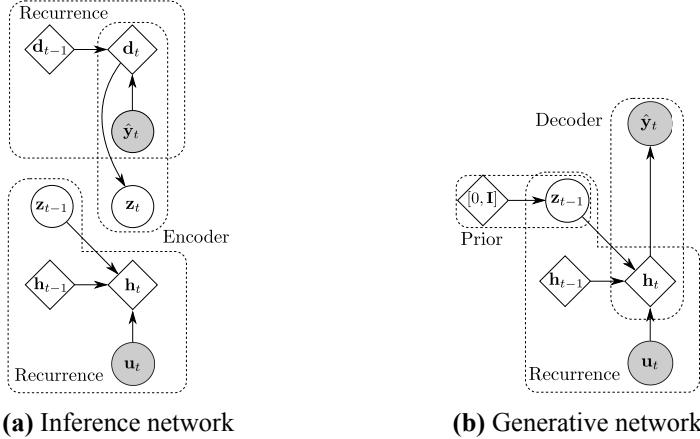
### A.3 STORN

The graphical model for STORN is given in Figure 9. The main difference to the other models is the additional forward running recurrent network in the inference network. This recurrence is implemented as a GRU with the same hidden layer dimension  $d_{\text{dim}}$  as the other recurrence with  $h_{\text{dim}}$ . This additional recurrence helps to encode the output distribution more precisely. Note also that in the generative network a static prior is used, similar to the VRNN-I. The joint true posterior of STORN can be factorized according to Figure 9 as

$$p_{\theta}(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}, \mathbf{h}_{1:T} | \mathbf{u}_{1:T}, \mathbf{h}_0) = p_{\theta}(\mathbf{y}_{1:T} | \mathbf{h}_{1:T}) \times \\ \times p_{\theta}(\mathbf{z}_{1:T} | [0, \mathbf{I}]) \tilde{p}(\mathbf{h}_{1:T} | \mathbf{z}_{1:T}, \mathbf{u}_{1:T}, \mathbf{h}_0). \quad (13)$$

The joint approximate posterior of STORN factorization as

$$q_\phi(\mathbf{z}_{1:T}, \mathbf{h}_{1:T}, \mathbf{d}_{1:T} | \mathbf{y}_{1:T}, \mathbf{u}_{1:T}, \mathbf{h}_0, \mathbf{d}_0) = q_\phi(\mathbf{z}_{1:T} | \mathbf{d}_{1:T}, \mathbf{h}_{1:T}) \times \tilde{p}(\mathbf{d}_{1:T} | \mathbf{y}_{1:T}, \mathbf{d}_0) \tilde{p}(\mathbf{h}_{1:T} | \mathbf{y}_{1:T}, \mathbf{u}_{1:T}, \mathbf{h}_0). \quad (14)$$



**Figure 9:** Graphical model for STORN.

## B Toy problem: Linear Gaussian system

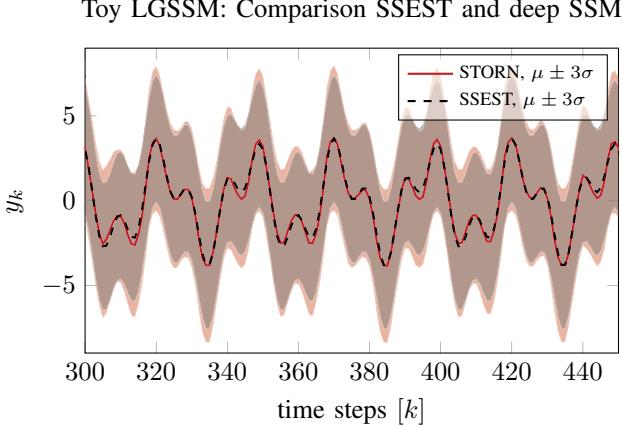
For identification of the linear Gaussian toy problem an excitation input signal with uniform random noise in the range  $[-2.5; 2.5]$  is used for the training and validation signals. The presented results are averaged over 50 Monte Carlo identifications. For each of these identifications the training and validation sequences are drawn from a new realization with the same statistical properties. For the test data the input is given by

$$u_k = \sin\left(\frac{2k\pi}{10}\right) + \sin\left(\frac{2k\pi}{25}\right). \quad (15)$$

The same test data set is used for all identified systems in order to obtain comparable performance measures.

The numerical results from Figure 3 show that the uncertainty quantification is conservative compared to the true uncertainty bounds of the system. Here an additional figure is provided to compare state of the art uncertainty quantification as calculated by SSEST with the uncertainty quantification given by a deep SSM. In Figure 10 this comparison is shown for the same time sequence as previously in Figure 3. It indicates that the uncertainty quantification of

STORN is comparable with the one of SSEST. Fine tuning of the hyperparameter of STORN could yield an uncertainty bound which tens towards the one of SSEST. In this experiment no fine tuning is performed.



**Figure 10:** LGSSM toy problem: Comparison between SSEST and STORN for their uncertainty estimation.

## C Narendra-Li benchmark

The true dynamics of the Narendra-Li Benchmark are given by [NL96] with the following second order model

$$\begin{bmatrix} x_{k+1}^{(1)} \\ x_{k+1}^{(2)} \end{bmatrix} = \begin{bmatrix} \frac{x_k^{(1)}}{1+(x_k^{(1)})^2} + 1 \sin(x_k^{(2)}) \\ x_k^{(2)} \cos(x_k^{(2)}) + x_k^{(1)} \exp(-\frac{(x_k^{(1)})^2+(x_k^{(2)})^2}{8}) + \frac{u_k^3}{1+u_k^2+0.5 \cos(x_k^{(1)}+x_k^{(2)})} \end{bmatrix},$$

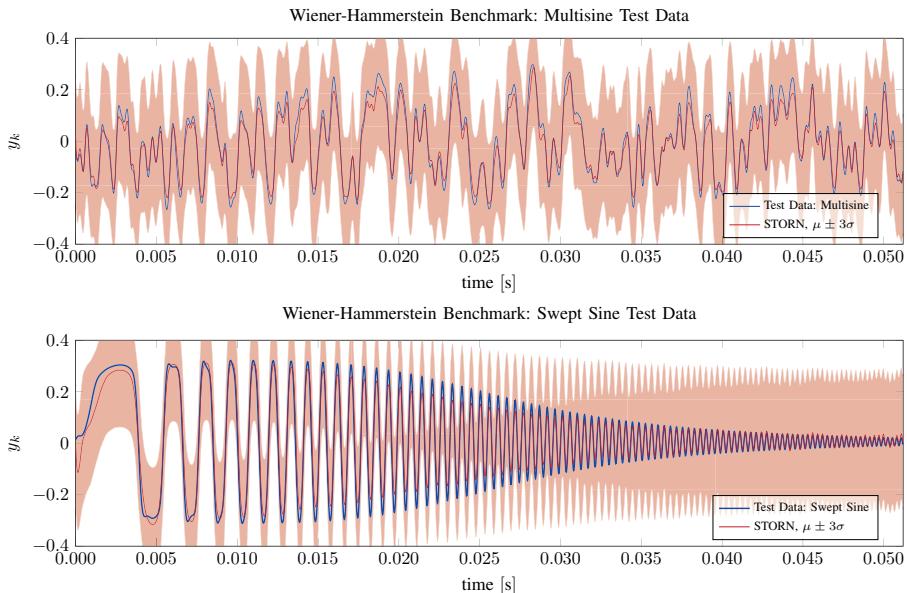
$$y_k = \frac{x_k^{(1)}}{1+0.5 \sin(x_k^{(2)})} + \frac{x_k^{(2)}}{1+0.5 \sin(x_k^{(1)})} + e_k.$$

Additional measurement noise is added to the original problem by [Ste99] of  $e_k \sim \mathcal{N}(0, 0.1)$  to make the problem more challenging. The same procedure for the excitation signals as for the linear Gaussian toy problem is used. Namely a training and validation data set where the input is uniform random noise in the range  $[-2.5; 2.5]$  and for the test data set the input sequence is defined by  $u_k = \sin(\frac{2k\pi}{10}) + \sin(\frac{2k\pi}{25})$ .

## D Wiener-Hammerstein process noise benchmark

In Section 4.3 all studied deep SSMs are compared for their performance on the test data sets of the Wiener-Hammerstein process noise benchmark. Additionally, here a time evaluation is shown in Figure 11 for both test data sets. Note that only the first 51.2 [ms] of the total  $\approx 20.97$  [ms] are shown to have well visible plots. The figure indicates an accurate identification of the complex system dynamics, which can represent the dynamics on two different test data sets. The uncertainty bounds are similarly conservative to the ones in the Narendra-Li benchmark. Tests with identifications on available data sets with more samples yield tighter uncertainty bounds but are not presented here since it would not be comparable with the comparison methods from literature.

All comparison methods in Table 3 use the same amount of training samples (8192), except from PNLSS which uses 9 realization with each consisting of 8192 samples.



**Figure 11:** Wiener Hammerstein benchmark: Time evaluation for multisine and swept sine test data set of best results from Table 3, i.e. STORN with  $h_{\text{dim}} = 40$ ,  $z_{\text{dim}} = 3$ ,  $n_{\text{layers}} = 3$ .

# Paper V

## Title

First Steps Towards Self-Supervised Pretraining of the 12-Lead ECG

## Authors

Daniel Gedon, Antônio H. Ribeiro, Niklas Wahlström, Thomas B. Schön

## Edited version of

D. Gedon, A. H. Ribeiro, N. Wahlström, and T. B. Schön. “First Steps Towards Self-Supervised Pretraining of the 12-Lead ECG.” in: *Computing in Cardiology (CinC)*. Online, 2021



# First Steps Towards Self-Supervised Pretraining of the 12-Lead ECG

## Abstract

Self-supervised learning is a paradigm that extracts general features which describe the input space by artificially generating labels from the input without the need for explicit annotations. The learned features can then be used by transfer learning to boost the performance on a downstream task. Such methods have recently produced state of the art results in natural language processing and computer vision. Here, we propose a self-supervised learning method for 12-lead electrocardiograms (ECGs). For pretraining the model we design a task to mask out subsegements of all channels of the input signals and try to predict the actual values. As the model architecture, we use a U-ResNet containing an encoder-decoder structure. We test our method by self-supervised pretraining on the CODE dataset and then transfer the learnt features by finetuning on the PTB-XL and CPSC benchmarks to evaluate the effect of our method in the classification of 12-leads ECGs. The method does provide modest improvements in performance when compared to not using pretraining. In future work we will make use of these ideas in smaller dataset, where we believe it can lead to larger performance gains.

## 1 Introduction

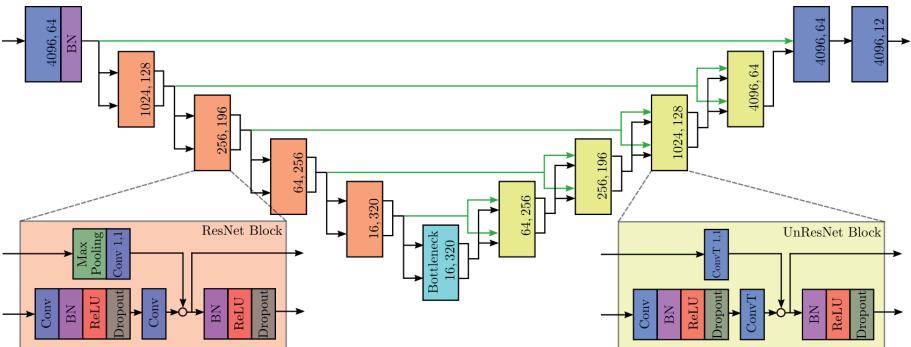
Supervised learning has been used for some of the most successful applications of deep learning. However, it requires a large amount of labeled training data to be successful [BCV13] which is restricting its applicability especially when data is scarce and annotating data is expensive, which is true for ECGs. Self-supervised learning, on the other hand, use unlabeled data, such as images or raw text, to *generate a supervision signal from the data itself* and then learns the most relevant features for the given task without the need for explicit labels [Zha18]. It opens up for interesting new possibilities, since the feature representations learned from the data can then be used with transfer learning to learn a downstream task on a new dataset with a smaller amount of labeled

data. Improved performance can be obtained with this method compared with training from scratch.

Self-supervised learning has been successfully applied to a number of different fields, such as natural language processing [Dev+19], computer vision [Hje+19], speech recognition and for audio-visual data. This learning paradigm has been particularly successful in areas where a large corpus of unlabeled data is easily available to improve the performance on downstream tasks with a small labeled dataset.

The ECG is a highly utilized diagnostic tool, where accurate labeling of abnormalities for training deep learning algorithms requires expensive hours of specialized doctors. Here, we are motivated by the availability of large quantities of unlabeled ECG data and the recent progress of automated analysis of ECGs to present a method of applying self-supervised methods to ECGs. For this, we: 1) define a self-supervised learning task and pretraining procedure which can learn generalisable features of ECG data and 2) develop and show that a ResNet based architecture can successfully be used in combination with our learning task. We illustrate our development by first pretraining on the CODE training dataset [Rib+20b] and, then, we use transfer learning with the ECG benchmarks: PTB-XL [Wag+20] and the dataset made available during the 2018 China Physiological Signal Challenge, the CPSC dataset [Liu+18].

## 2 Self-Supervised learning task and model architecture



**Figure 1: U-ResNet Architecture:** Output dimension of each block are indicated. The input and output to the architecture are the ECG traces with a length of 4096 samples and 12 channels. In the decoder the input channels are concatenated.

Our self-supervised learning task is inspired by the completion task of BERT [Dev+19], which is a popular model developed in the context of natural

language processing. The model architecture takes inspiration from the context encoder [Pat+16]. The sequential and bi-directional nature of the problem is the reason for us to choose a BERT-like self-supervision task. The convolution based architecture is motivated by the correlations between nearby samples in the ECG.

**Pretraining task description:** Our self-supervised pretraining task requires the model to reconstruct the input from a corrupted version of it. Specifically, we modify the input to the network by masking out the true values for random subsequences of  $N$  samples from the ECGs across all channels. The masked input is fed to the model which has to predict the true values for the masked subsequences. In total we mask out  $P$  percent of the input samples in this way. Similar to [Dev+19], 90% of the masked input subsequences are replaced by zeros and the remaining 10% are left unchanged. We use an MSE reconstruction loss.

An important design parameter is the number of subsequent samples  $N$  in a subsequence that are masked out. If the value of  $N$  is too small, then the task might reduce to a simple interpolation problem due to the significant temporal correlation in the ECG signal. If on the other hand,  $N$  is too large the patch might become too hard to reconstruct, rendering the task ineffective for self-supervision.

**Model architecture:** We use a convolutional ResNet architecture adapted from image classification [He+16] to unidimensional signals as described in [Rib+20b]. Due to the convolutional model, a fixed input size is required. An extension for variable input size for this model is possible [Rib+20a]. For pre-training we choose an encoder-decoder architecture based on [Pat+16] where the decoder mirrors the encoder using transposed convolutions. The bottleneck layer is a channel-wise fully connected layer. To improve the reconstruction ability of the network, we append the architecture with U-Net based skip-connections [RFB15]. Details are shown in Figure 1.

**Training on downstream task:** Once a model is trained on the pretraining task, we transfer learn the pretrained model to a new labeled dataset as downstream classification task. Specifically, we use only the encoder, exclude the bottleneck layer, remove the U-Net skip connections and add a linear classifier.

### 3 Experiments

In this section we present the results of our experiments. We show that our pretraining task and model architecture works and generalises to new datasets. The results for the highly competitive ECG classification performance indi-

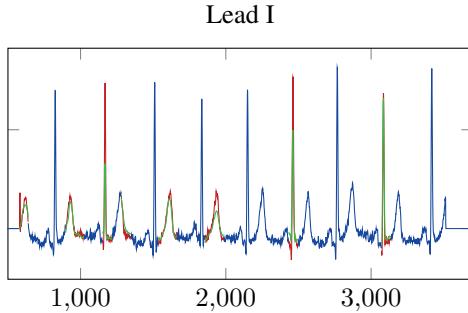
cates that our approach is an interesting and promising path forward compared to randomly initialized models.

We use three datasets for our experiments. (1) CODE obtained by the Tele-health Network of Minas Gerais [Alk+12] which contains 2,322,513 ECGs from 1,676,384 patients out of 811 counties in Minas Gerais, Brazil with a length between 7 and 10 seconds. CODE also includes a test dataset with 827 ECGs. (2) CPSC 2018 [Liu+18] with 6,877 ECGs from 11 hospitals in China with lengths varying between 6 and 60 seconds including labels for 8 different anomalies. (3) PTB-XL [Wag+20] with 21,837 ECGs from 18,885 patients from 51 different sites with a length of 10 seconds including labels for 71 different anomalies. The anomalies are not mutually exclusive. Hence, for the classification task we use a sigmoid output layer together with a binary cross-entropy loss.

For self-supervised pretraining we use the CODE training data [Rib+20b] and for the downstream task training we classify the anomalies in CPSC and PTB-XL. For CPSC we use a 70%-30% training/validation split (using the validation split also for reporting the final results), and for PTB-XL we take the predefined 80%-10%-10% training/validation/test split. We apply a high-pass filter to the input ECGs: an elliptic filter with a cutoff frequency at 0.8Hz and an attenuation of 40dB, applied in the forward and reverse direction to obtain zero-phase distortion. The filter removes biases and low frequency trends. In sequence, we resample all ECGs to a sampling frequency of 400 Hz and use zero padding if necessary to obtain a signal of a fixed length corresponding to 4096 samples. ECGs with longer traces are split over multiple batches and the individual classification predictions are combined by averaging the logits. For details on the hyperparameter of the model and the (pre-)training, see Appendix B.

### 3.1 Pretraining

The pretraining reconstruction task requires two parameters, the number of subsequently masked samples  $N$  and the percentage of masked samples  $P$  in total. For the first, we choose  $N = 64 \doteq 0.16$  sec which cover approximately one ECG segment, i.e. a characteristic subpart of the ECG. Therefore, the model is required to reconstruct complete segments and to learn the characteristics of ECG segments which can then be utilized in downstream tasks. For the second parameter, we choose  $P = 30\%$  in order to make the problem hard enough, but still feasible. We train the model for 10 epochs on a single GPU. As an example, Figure 2 illustrates that the model can reconstruct complete segments and even partially cover the peaks. More examples with four leads of all used datasets are given in Appendix C.1.



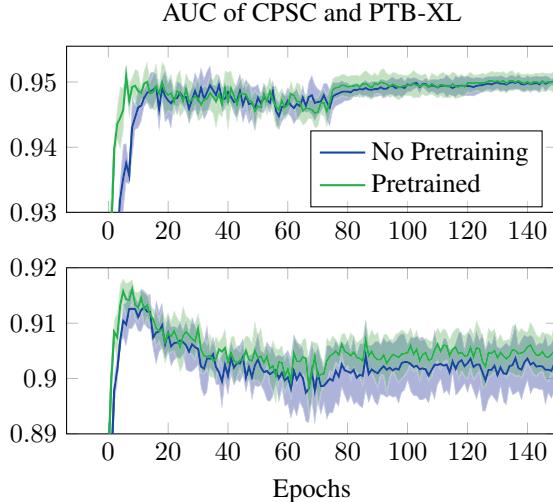
**Figure 2: ECG Reconstruction:** Examples for lead I of one CODE test ECG. Inputs to the model are blue where the gaps ( $N = 64$ ,  $P = 30\%$ ) are filled with zeros. The ground truth which should be reconstructed is red and the model output is green.

**Table 1:** Average MSE loss for the reconstruction on different datasets when removing blocks from the U-ResNet.

blocks	CODE train	CODE test	CPSC	PTB-XL
all	1.20	1.80	17.40	22.01
-1	2.54	3.06	19.63	19.66
-2	4.58	5.90	22.46	18.12
-3	19.72	19.70	32.12	28.92

To quantify how well the model generalises, we analyse the average MSE reconstruction loss of the obtained model for all datasets. Note that for the CODE training dataset, which we use for pretraining we report results on the first 100,000 ECGs only. The results are listed in Table 1. In the first row we see that the model generalizes reasonably well to the CODE test set, but not to the other test sets. Visual inspections of the ECGs show that while CODE is a rather homogeneous dataset, CPSC and PTB-XL have many outliers accounting for this difference in loss.

We additionally inspect the importance of depth for the reconstruction. We remove up to three ResNet blocks in the encoder/decoder, replace the bottleneck layer with an identity and try to reconstruct without re-training. Due to the U-ResNet skip connections, the model could be able to reconstruct the ECG only from the first blocks, which limits the usefulness of deeper ResNet blocks. We observe that the loss increases sharply when removing layers. We therefore conclude that the model learns useful representations in deeper layers, confirming our choice of architecture.



**Figure 3: AUC evolution:** AUC on the validation set during training for CPSC (top) and PTB-XL (bottom). Mean and standard deviation over 6 models are shown.

### 3.2 ECG classification

The main results of the paper are summarized in Table 2. We compare multiple performance metrics on the classification of the CPSC and the PTB-XL dataset with and without pretraining. The results in the table correspond to the model with the largest area under the ROC curve (AUC). Plots for the evolution of the AUC on the validation set over the training epochs are given in Figure 3. The improvements we obtained are modest, but we believe the procedure has the potential to open interesting new possibilities in the low data regime.

### 3.3 Diversity of predictions

For many real world applications, an ensemble of models outperforms standalone models in terms of accuracy [Zho12] while it provides uncertainty bounds. It is theoretically proven that a high performing ensemble model requires the base models to be accurate and diverse in terms of their errors on the input [KV95]. Diversity in deep neural networks is achieved by initialising the network randomly and shuffling the training examples. For the pre-trained model we always initialize at the same point. We analyse the diversity of the trained models initialized with our pretraining method measured by the pairwise disagreement [Zho12] and compare to randomly initialised models. Details and complete results are given in Appendix C.2. We observe, that the models from our method on CPSC are only slightly less diverse than purely randomly initialised models while achieving higher AUC values. For PTB-

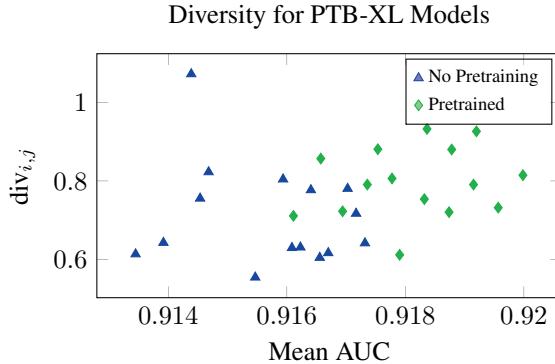
**Table 2:** Classification performance metrics with and without pretraining. ‘PT’ indicates with pretraining. For  $F_\beta$  and  $G_\beta$  we use  $\beta = 2$ , see Appendix D for metric definitions.

(a) Metrics for CPSC			
Model	$F_\beta$	$G_\beta$	AUC
Ours	.775 ± .004	.533 ± .016	.953 ± .001
Ours + PT	<b>.780 ± .013</b>	<b>.538 ± .019</b>	<b>.954 ± .001</b>

(b) Metrics for PTB-XL			
Model	$F_{\max}$	AUC	
ResNet1d [Str+20]	.767 ± .008	.919 ± .008	
Ours	.667 ± .037	.917 ± .004	
Ours + PT	.638 ± .034	<b>.919 ± .003</b>	

XL we even notice, that our method produces models with higher AUC and greater diversity, see Figure 4. The results indicate that our method is well suited for ensemble based models.



**Figure 4:** Diversity plot of models for PTB-XL. Pairwise diversity over the mean between the AUC of two models.

## 4 Conclusion and future work

In this paper, we introduce a simple completion based pretraining task for ECGs based on ResNet models. We illustrate the use of the method in the 12-lead ECG classification of the benchmarks: CPSC and PTB-XL. Our performance improvement is limited, nonetheless, we still believe this line of research to be interesting and relevant. This work should be seen as initial

exploratory step that can yield interesting future work. Continuing in this direction can lead to more general ECG models that can be used across many datasets to solve tasks for which data is scarce.

**Acknowledgments** This research was supported by the Brazilian research agency CAPES, the *Wallenberg AI, Autonomous Systems and Software Program (WASP)* funded by Knut and Alice Wallenberg Foundation, the *Kjell och Märta Beijer Foundation* and the project *AI4Research* at Uppsala University.

## References

- [Alk+12] M. B. Alkmim, R. M. Figueira, M. S. Marcolino, C. S. Cardoso, M. Pena de Abreu, L. R. Cunha, D. F. da Cunha, A. P. Antunes, A. G. de A Resende, E. S. Resende, and A. L. P. Ribeiro. “Improving patient access to specialized health care: the Telehealth Network of Minas Gerais, Brazil.” In: *Bulletin of the World Health Organization* (2012), pp. 373–378 (cit. on p. V-4).
- [BCV13] Y. Bengio, A. Courville, and P. Vincent. “Representation learning: a review and new perspectives.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013), pp. 1798–1828 (cit. on p. V-1).
- [Dev+19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Minneapolis, Minnesota, 2019, pp. 4171–4186 (cit. on pp. V-2, V-3).
- [FHL19] S. Fort, H. Hu, and B. Lakshminarayanan. “Deep Ensembles: A Loss Landscape Perspective.” In: *arXiv:1912.02757* (2019) (cit. on p. V-14).
- [GBD17] GBD 2016 Causes of Death Collaborators. “Global, regional, and national age-sex specific mortality for 264 causes of death, 1980–2016: a systematic analysis for the Global Burden of Disease Study 2016.” In: *Lancet (London, England)* (2017), pp. 1151–1210 (cit. on p. V-11).
- [Gus96] F. Gustafsson. “Determining the initial states in forward-backward filtering.” In: *IEEE Trans. Signal Process.* (1996), pp. 988–992 (cit. on p. V-11).

- [Han+19] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng. “Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network.” In: *Nature Medicine* (2019), pp. 65–69 (cit. on p. V-11).
- [He+16] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778 (cit. on p. V-3).
- [Hin18] G. Hinton. “Deep learning—a technology with the potential to transform health care.” In: *JAMA* (2018), pp. 1101–1102 (cit. on p. V-11).
- [Hje+19] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. “Learning Deep Representations by Mutual Information Estimation and Maximization.” In: *International Conference for Learning Representations (ICLR)*. 2019 (cit. on p. V-2).
- [KV95] A. Krogh and J. Vedelsby. “Neural Network Ensembles, Cross Validation, and Active Learning.” In: *Advances in Neural Information Processing Systems*. 1995, pp. 231–238 (cit. on p. V-6).
- [Liu+18] F. Liu, C. Liu, L. Zhao, X. Zhang, X. Wu, X. Xu, Y. Liu, C. Ma, S. Wei, Z. He, J. Li, and E. N. Yin Kwee. “An Open Access Database for Evaluating the Algorithms of Electrocardiogram Rhythm and Morphology Abnormality Detection.” In: *Journal of Medical Imaging and Health Informatics* (2018), pp. 1368–1373 (cit. on pp. V-2, V-4).
- [Pat+16] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. “Context Encoders: Feature Learning by Inpainting.” In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2536–2544 (cit. on p. V-3).
- [RFB15] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI*. 2015, pp. 234–241 (cit. on p. V-3).
- [Rib+20a] A. H. Ribeiro, D. Gedon, D. M. Teixeira, M. H. Ribeiro, A. L. P. Ribeiro, T. B. Schön, and W. Meira Jr. “Automatic 12-lead ECG classification using a convolutional network ensemble.” In: *Computing in Cardiology*. 2020 (cit. on p. V-3).

- [Rib+20b] A. H. Ribeiro, M. H. Ribeiro, G. M. M. Paixão, D. M. Oliveira, P. R. Gomes, J. A. Canazart, M. P. S. Ferreira, C. R. Andersson, P. W. Macfarlane, W. Meira Jr., T. B. Schön, and A. L. P. Ribeiro. “Automatic diagnosis of the 12-lead ECG using a deep neural network.” In: *Nature Communications* (11 2020), p. 1760 (cit. on pp. V-2, V-3, V-4, V-11).
- [Str+20] N. Strothoff, P. Wagner, T. Schaeffter, and W. Samek. *Deep Learning for ECG Analysis: Benchmarks and Insights from PTB-XL*. 2020 (cit. on p. V-7).
- [Wag+20] P. Wagner, N. Strothoff, R.-D. Bousseljot, D. Kreiseler, F. I. Lunze, W. Samek, and T. Schaeffter. “PTB-XL, a large publicly available electrocardiography dataset.” In: *Scientific Data* (2020) (cit. on pp. V-2, V-4).
- [Zha18] R. Zhang. “Image Synthesis for Self-Supervised Visual Representation Learning.” PhD thesis. University of California at Berkeley, 2018 (cit. on p. V-1).
- [Zho12] Z.-H. Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012 (cit. on pp. V-6, V-14).

# Appendix

## A Broader impact

Machine learning advances might bring great improvements to health care and clinical practice [Hin18] where automated ECG analysis is an area of particular interest. Globally, more than 17 million people die each year of cardiovascular diseases [GBD17] for which ECG is a major diagnostic tool. Making it more accessible and reliable can lower these fatalities, especially in low and middle-income countries, which concentrate 75% of deaths related to cardiovascular disease and where access to cardiologists with expertise in ECG diagnosis is rare. These numbers indicate that already a slight improvement in the state of the art can have a major positive effect on the lives of millions of people.

Recent successes in deep learning applied to the automated ECG classification [Han+19; Rib+20b] rely on supervised learning settings. The possibility of training with fewer labelled points might open up new possibilities, such as screening the population for rare cardiac diseases, for which there are too few examples available for a supervised approach.

## B Model and learning parameters

For the input high-pass filter, we use a cut-off frequency of  $f_c = 0.8$  Hz, a rejection band of  $f_{st} = 0.4$  Hz, a ripple in the bandpass of  $r_p = 0.2$  dB and an attenuation in the rejection band of  $r_s = 40$  dB with forward-backward filtering [Gus96]. The model architecture consists of four encoder and four decoder ResNet blocks with 1D convolutions. In the decoder, we use transposed convolution to upsample the sequence length. A dropout rate of 0.5 is used. The convolution parameters are listed in Table 3 which are in line with the dimensions in Figure 1.

For pre-training we mask out  $N = 64$  subsequent samples. A total of  $P = 30\%$  of the samples are masked out, excluding padded samples. The same mask is applied to all leads. We replace 90% of the masked subsequences with a value of zero and leave the remaining 10% as they are. We use a constant learning rate of  $10^{-4}$  and clip the gradient norm at 1.0. We pre-train for 10 epochs on a single Titan Xp GPU for approximately 85h.

For training, we use a learning rate of  $10^{-3}$  which is reduced by a factor of 10 at epochs 75 and 125. In total, we train for 150 epochs and store the model with the highest AUC. Values for other metrics are given for this stored model. We vary the seed for each training run which varies the random initialisation and the shuffling of the training data. However, the seed does not vary the

training/test data split for CPSC. The same seed is used for one model with pre-training and one without to ensure consistency in the comparison.

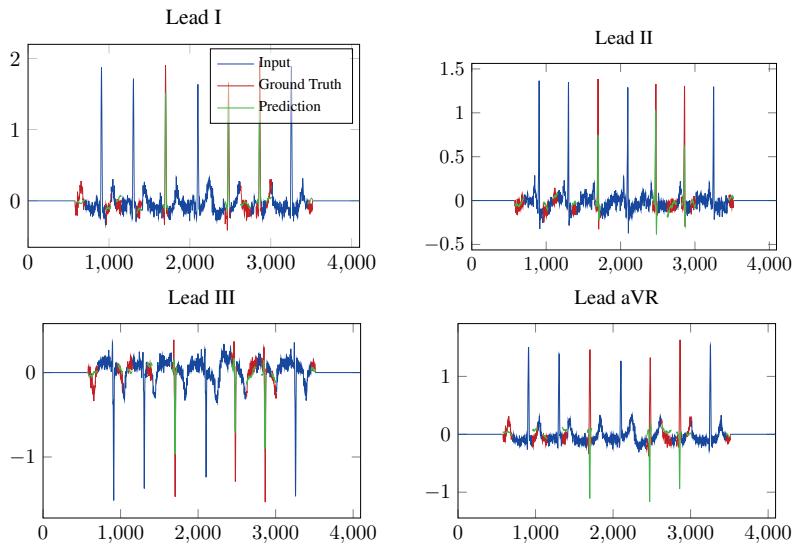
**Table 3:** Model parameter

Block	Conv.	in channels	out channels	kernel size	stride	padd.	out padd.
-	Enc. Conv	12	64	17	1	8	-
0	Conv1	64	128	17	4	8	-
0	Conv2	128	128	17	4	7	-
0	ConvSkip	64	128	1	1	0	-
1	Conv1	128	196	17	4	8	-
1	Conv2	196	196	17	4	7	-
1	ConvSkip	128	196	1	1	0	-
2	Conv1	196	256	17	4	8	-
2	Conv2	256	256	17	4	7	-
2	ConvSkip	196	256	1	1	0	-
3	Conv1	256	320	17	4	8	-
3	Conv2	320	320	17	4	7	-
3	ConvSkip	256	320	1	1	0	-
-	Bottleneck	320	320	1	1	0	-
0	Conv1	640	256	17	1	8	-
0	Conv2T	256	256	17	4	7	1
0	ConvSkip	640	256	1	4	0	3
1	Conv1	512	196	17	1	8	-
1	Conv2T	196	196	17	4	7	1
1	ConvSkip	512	196	1	4	0	3
2	Conv1	392	128	17	1	8	-
2	Conv2T	128	128	17	4	7	1
2	ConvSkip	392	128	1	4	0	3
3	Conv1	256	64	17	1	8	-
3	Conv2T	64	64	17	4	7	1
3	ConvSkip	256	64	1	4	0	3
-	Dec. Conv1	64	64	17	1	8	-
-	Dec. Conv2	64	12	17	1	8	-

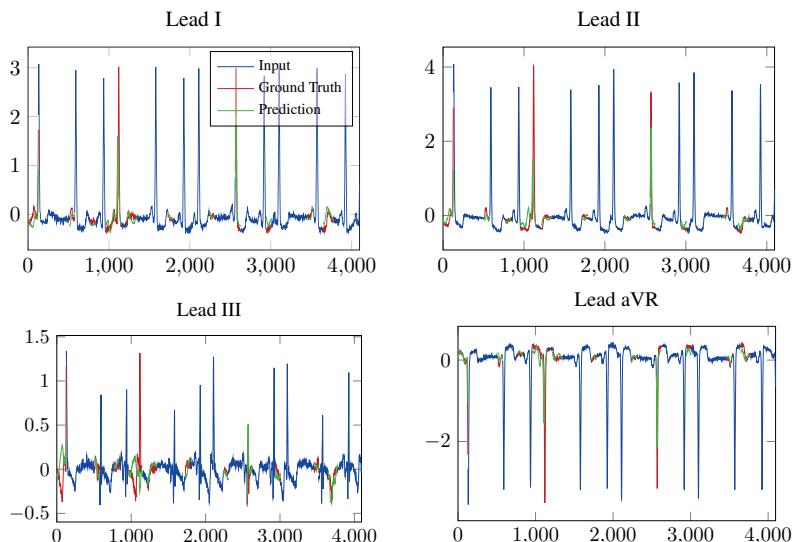
## C Additional results

### C.1 Pre-Training: Reconstruction results

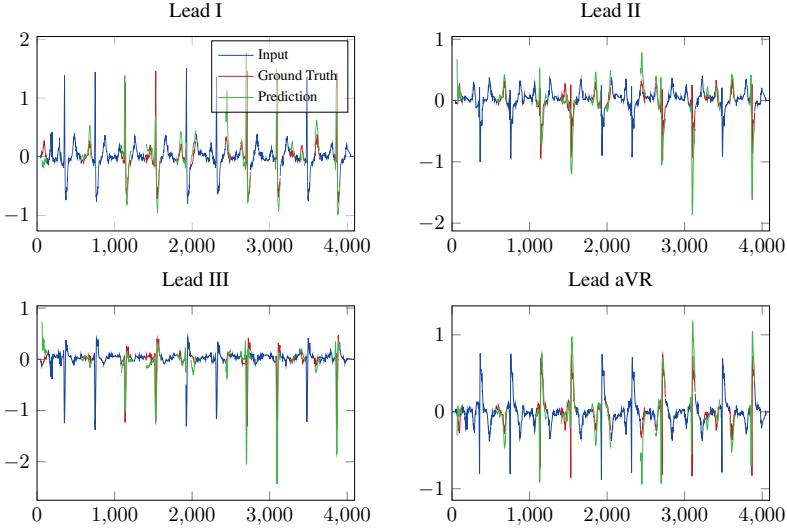
In Figure 5-7 we show additional randomly selected examples for ECG reconstruction. We show the first four leads. We can see that while the reconstruction of CODE test data is still fairly good, it becomes worse for CPSC and PTB-XL which is an indication that these datasets are slightly out of distribution. This is supported by the findings in Table 1.



**Figure 5:** Additional example for ECG reconstruction from CODE test set



**Figure 6:** Example for ECG reconstruction from CPSR

**Figure 7:** Example for ECG reconstruction from PTB-XL

## C.2 Diversity of Model with and without Pre-training

There exist various definitions for the measure of diversity [Zho12]. In this work, we consider the pairwise disagreement which measures the disagreement at the output of the model for the same input. It is defined for two functions  $f_i(\cdot)$  and  $f_j(\cdot)$  as

$$div_{i,j} = 1 - \frac{1}{N} \sum_{n=1}^N \mathbf{I}[f_i(x_n) == f_j(x_n)] \quad (1)$$

where we have  $N$  data points and where  $\mathbf{I}[\cdot]$  is the indicator function with output 1 if the argument is true and zero otherwise. This formulation can be interpreted as the fraction of prediction where the two models disagree. We extend this formulation to multi-class classification problems with  $K$  classes as

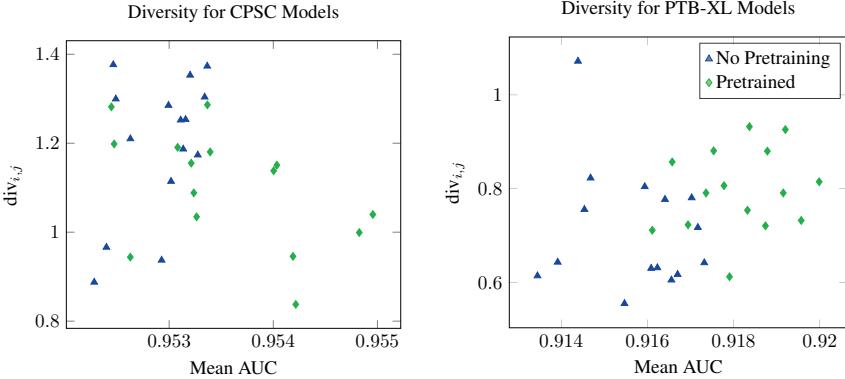
$$div_{i,j} = \frac{1}{K} \sum_{k=1}^K \left( 1 - \frac{1}{N} \sum_{n=1}^N \mathbf{I}[f_i(x_n)_k == f_j(x_n)_k] \right) \quad (2)$$

where  $f(\cdot)_k$  is the prediction of the function for class  $k$ . Since we have multiple base models, we compute  $div_{i,j}$  for each combination of the models. Hence, for  $M$  models we have a total combination of  $\binom{M}{2}$  diversity measures. To account for the higher accuracy, we scale the diversity by the mean of the AUC of each base learner, similar to [FHL19]. Hence, our final diversity

measure is given by

$$div_{i,j} = \frac{\frac{1}{K} \sum_{k=1}^K \left( 1 - \frac{1}{N} \sum_{n=1}^N \mathbf{I}[f_i(x_n)_k == f_j(x_n)_k] \right)}{1 - (\text{AUC}_i + \text{AUC}_j)/2}. \quad (3)$$

The diversity for each combination of models is shown in Figure 8. Each point represents the combination of two models. The x-axis shows the mean AUC of the combined models and the y-axis the diversity between the model’s predictions. An ideal method would produce models in the top right corner with high accuracy in terms of AUC and high diversity between the models. We can see, that for CPSC while the pre-trained model is slightly lower in terms of diversity, it is higher in accuracy. Specifically, the mean of all the diversity points is [0.953, 1.198] without pre-training and [0.954, 1.098] with pre-training. For the PTB-XL dataset, we see even, that the models from our methods are in general more diverse and more accurate in the given measures. Specifically, the mean of all diversity points is [0.9157, 0.7109] without pre-training and [0.9182, 0.7954] with pre-training. Hence, we conclude that our method produces sufficiently diverse and accurate base models compared with a randomly initialised model.



**Figure 8:** Diversity scatter plots of models for validation dataset of CPSC and PTB-XL. Plot for PTB-XL repeated from Figure 4 but with the same ratio as for CPSC for easy comparison.

## D Definition of performance metrics

To evaluate the performance on the classification task, we consider four different metrics, see Table 2. We define them as follows with TP as true positive, FP as false positive, TN as true negative and FN as false negative:

- $F_{max}$  as the threshold dependent  $F_1$  score where the threshold  $\tau$  is varied to maximize the  $F_1$  score:

$$F_{max} = \max_{\tau} \frac{2 \cdot \text{Precision}(\tau) \cdot \text{Recall}(\tau)}{\text{Precision}(\tau) + \text{Recall}(\tau)} \quad (4)$$

- $F_{\beta}$  as a generalisation of the  $F_1$  score:

$$F_{\beta} = (1 + \beta^2) \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}} \quad (5)$$

$$= (1 + \beta^2) \frac{\text{TP}}{(1 + \beta^2)\text{TP} + \text{FP} + \beta^2 \cdot \text{FN}} \quad (6)$$

- $G_{\beta}$  used in the PhysioNet 2020 challenge for the initial CPSC dataset:

$$G_{\beta} = \frac{\text{TP}}{\text{TP} + \text{FP} + \beta \cdot \text{FN}} \quad (7)$$

- AUC as the area under the ROC curve where the ROC curve plots the TPR (true positive rate) over the FPR (false positive rate):

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (8)$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \quad (9)$$

Note that we use  $\beta = 2$  for the  $F_{\beta}$  and  $G_{\beta}$  score which assigns more weight to recall over precision.

# Paper VI

## Title

Development and validation of deep learning ECG-based prediction of myocardial infarction in emergency department patients

## Authors

Stefan Gustafsson, Daniel Gedon, Erik Lampa, Antônio H. Ribeiro, Martin J. Holzmann, Thomas B. Schön, Johan Sundström

## Edited version of

S. Gustafsson, D. Gedon, E. Lampa, A. H. Ribeiro, M. J. Holzmann, T. B. Schön, and J. Sundström. “Development and validation of deep learning ECG-based prediction of myocardial infarction in emergency department patients.” In: *Scientific Reports*. Vol. 12. 2022

An early version of this paper appeared as:

S. Gustafsson, D. Gedon, at al. ResNet-based ECG Diagnosis of Myocardial Infarction in the Emergency Department. In: *Machine learning from ground truth: New medical imaging datasets for unsolved medical problems Workshop at NeurIPS*. 2021



# **Development and validation of deep learning ECG-based prediction of myocardial infarction in emergency department patients**

## **Abstract**

Myocardial infarction diagnosis is a common challenge in the emergency department. In managed settings, deep learning-based models and especially convolutional deep models have shown promise in electrocardiogram (ECG) classification, but there is a lack of high-performing models for the diagnosis of myocardial infarction in real-world scenarios. We aimed to train and validate a deep learning model using ECGs to predict myocardial infarction in real-world emergency department patients.

We studied emergency department patients in the Stockholm region between 2007 and 2016 that had an ECG obtained because of their presenting complaint. We developed a deep neural network based on convolutional layers similar to a residual network. Inputs to the model were ECG tracing, age, and sex; and outputs were the probabilities of three mutually exclusive classes: non-ST-elevation myocardial infarction (NSTEMI), ST-elevation myocardial infarction (STEMI), and control status, as registered in the SWEDEHEART and other registries. We used an ensemble of five models.

Among 492,226 ECGs in 214,250 patients, 5,416 were recorded with an NSTEMI, 1,818 a STEMI, and 485,207 without a myocardial infarction. In a random test set, our model could discriminate STEMIs/NSTEMIs from controls with a C-statistic of 0.991/0.832 and had a Brier score of 0.001/0.008. The model obtained a similar performance in a temporally separated test set of the study sample, and achieved a C-statistic of 0.985 and a Brier score of 0.002 in discriminating STEMIs from controls in an external test set.

We developed and validated a deep learning model with excellent performance in discriminating between control, STEMI, and NSTEMI on the presenting ECG of a real-world sample of the important population of all-comers to the

emergency department. Hence, deep learning models for ECG decision support could be valuable in the emergency department.

## 1 Introduction

Emergency department care costs are high [GP16] and rising [Lan+20] in developed societies. Based on limited data in a chaotic environment, emergency medicine doctors must make quick decisions about patients' probabilities for many diagnoses and risks. Diagnostic error is commonplace [Moo+17; Med+16], and there is a desperate need for emergency department decision support systems [Wri+19].

The emergency department handling of myocardial infarctions is especially precarious. In the United States, myocardial infarctions are missed in the range of 10–50,000 per year at emergency departments [Sha+21]. At the other end, less than half of those hospitalized for a suspected myocardial infarction are eventually diagnosed with the condition [CS18]. The electrocardiogram (ECG) has long been used for diagnosing myocardial infarctions. Some large myocardial infarctions are accompanied by an elevation of the ST-segment of the ECG, giving them the name ST-elevation myocardial infarctions (STEMIs). Other myocardial infarctions, the non-ST-elevation myocardial infarctions (NSTEMIs), are often inconspicuous to the human eye on the ECG, and rely on other means of diagnosis.

Artificial intelligence (AI) with deep learning based models has shown much recent promise in ECG classification [Sio+21], for common ECG diagnoses [Rib+20] as well as for traits with unclear ECG diagnostic criteria or those not usually thought of as ECG diagnoses [Tis+19; Coh+21; Rag+20]. Even ECGs that appear normal to the human eye carry information useful for AI algorithms [Rag+20; Att+19] AI is very promising in the diagnosis of myocardial infarction [Liu+21; Cho+20], but many studies have used limited [Mak+20; Al+20] or managed [Liu+21; Cho+20; Mak+20; Al+20; Zha+20] datasets. We postulate that AI ECG interpretation may be useful also in the most important population—all-comers to emergency departments.

Most of the studies above, as well as most studies on deep learning for arrhythmia classification [Ebr+20], detection of myocardial infarctions [XLC22] or other ECG-based tasks [Hon+20] use models based on convolutional layers. We employed a ResNet [He+16], which is a special convolutional neural network with skip connections. This structure allows use of deeper models with more layers, which was an essential breakthrough in deep learning. By deploying a standardized architecture, we aimed to ensure reproducibility and understanding of the model components. This is in contrast to prior work which use specialized, non-standard architectures [Liu+21; Cho+20]. Using a

large real-world sample of patients presenting at emergency departments, we developed and validated a deep learning model for diagnosis of NSTEMI and STEMI on the presenting ECG.

## 2 Results

Of the included total 492,226 ECGs from the emergency department visit, 5,416 (1.1%) were recorded with an NSTEMI, 1,818 (0.4%) with a STEMI and 484,992 (98.5%) without a myocardial infarction. Clinical characteristics of the study sample are presented in Table 1 and stratified for the data splits in Supplementary Table 5, and the patients' age and admission date distributions are shown in Supplementary Figure 8 and Figure 9.

**Table 1:** Clinical characteristics of the study sample. Patient characteristics of coronary care unit admissions included in the study, by control/NSTEMI/STEMI outcome. Data are medians (quartiles) or percent.

\*Prevalent disease based on any diagnosis position, inpatient and outpatient specialist care combined.

\*\*Combining troponin and high-sensitive troponin laboratory measurements from regional laboratory databases and the SWEDEHEART database. Maximum of all available measurements within the time window reported. Done separately for troponin I and T.

\*\*\*Primary diagnosis from inpatient specialist care and/or specialized outpatient care at time of the ED/CCU visit. STEMI, ST-elevation myocardial infarction; NSTEMI, non-ST-elevation myocardial infarction; ED, emergency department; NTproBNP, N-terminal pro-B-type natriuretic peptide.

	Control	NSTEMI	STEMI
Number of ECGs	484,992	5,416	1,818
<b>Clinical characteristics at ED visit</b>			
Age	65.0 (47.0,78.0)	71.0 (62.0,81.0)	66.0 (57.0,77.0)
Male	47.3	65.4	73.7
Year	2013 (2010,2015)	2013 (2011,2015)	2013 (2011,2015)
<b>Presenting complaint</b>			
Chest pain	21.5	71.3	70.1
Difficulty breathing	14.5	12.3	5.9
Dizziness	7.1	0.7	1.2
Heart problems	2.0	1.3	1.9
Circulatory arrest	0.1	1.1	3.7

Continued on next page

	Control	NSTEMI	STEMI
<b>Cardiovascular diagnoses prior to ED visit*</b>			
Myocardial infarction	8.4	26.1	16.2
Unstable angina	4.2	11.2	6.4
Ischemic heart disease	20.1	43.1	23.8
Stroke	9.1	11.2	7.4
Peripheral artery disease	7.0	12.4	7.2
Heart failure	15.8	20.6	9.6
Atrial fibrillation	19.9	15.8	8.7
Cardiovascular disease	59.0	70.8	52.2
<b>Drugs with &gt;=1 dispensation within one year prior to ED visit</b>			
Renin-angiotensin system inhibitors	33.0	51.8	36.9
Calcium channel blockers	17.9	29.3	21.1
Beta-receptor blockers	36.3	50.6	33.9
Mineralocorticoid receptor antagonists	6.3	6.2	3.0
Diuretics	27.6	34.2	19.4
Anti-arrhythmic drugs	1.7	0.4	0.6
Statins	23.7	41.9	24.3
Anticoagulants	12.8	8.8	5.6
Antiplatelets	27.8	48.8	28.9
<b>Cardiac enzymes within ED visit or coronary care unit hospitalization**</b>			
Troponin I measured	1.1	8.2	13.8
Max troponin I (ng/L)	29.7 (29.7,40.0)	2700.0 (610.0,10150.0)	18100.0 (3100.0,47150.0)
Troponin T measured	38.2	87.2	87.6
Max troponin T (ng/L)	9.9 (5.0,19.0)	266.0 (93.0,814.0)	1900.0 (519.0,4680.0)
NTproBNP measured	8.7	21.5	17.1
Max NTproBNP (ng/L)	1340 (286,4300)	2940 (784,8720)	3120 (772,9220)
<b>Main diagnoses at admission ***</b>			
Myocardial infarction	0.0	94.1	97.2
Unstable angina	0.2	4.1	1.6
Ischemic heart disease	1.3	94.6	97.2
Stroke	2.0	1.1	0.7
Peripheral artery disease	0.4	0.3	0.3
Heart failure	3.3	2.1	0.9
Atrial fibrillation	5.9	0.8	0.3
Cardiovascular disease	18.4	96.2	98.4
<b>Mortality after coronary care unit admission</b>			
30-day all-cause death	3.5	6.4	10.6
In-hospital all-cause death	2.6	5.5	9.2

We conducted a preliminary analysis in a subset of the study sample including only patients that were admitted to the coronary care unit (CCU), i.e. omitting the large pool of controls with an emergency department visit without being transferred to the CCU. From this preliminary analysis, we identified modifications to the previously published architecture [Rib+20] that could contribute to the performance of our model. The two changes that were most important were: (1) to extend our training dataset with repeated ECGs recordings during the same visit of a patient, which can be viewed as a form of data augmentation; and (2) the use of an ensemble-based model.

The performance of our model in the two test sets of the study sample (random and temporal) and a publicly available external database, PTB-XL, is described in Table 2 including model uncertainty over ten different initialization seeds. Additionally in Supplementary Table 7, we show bootstrapped data uncertainty with similar performance, details for this are in the Supplementary Methods, Appendix A. In the random test set, STEMIs could be discriminated with a C-statistic of 0.991 and the model had a Brier score of 0.001. For NSTEMIs, the model had a C-statistic of 0.832 and a Brier score of 0.008, with lower precision than STEMIs. The temporal test set, which contained data from emergency department visits that did not overlap in time of the development set, resulted in a C-statistic of 0.985 for STEMI and 0.867 for NSTEMI. Figure 1 illustrates receiver operating characteristics and precision-recall curves for multiple independently trained models on the two test splits of the study sample. Calibration was acceptable (calibration plots in Supplementary Figure 10, predicted probabilities in Supplementary Figure 11).

**Table 2:** Performance of the model in the random and temporal test set of the study sample as well as the external PTB-XL test set. Results of the model in the two study sample test sets and the publicly available PTB-XL dataset as comparison in the rightmost column. Sample size is given by each test set and outcome label together with proportion out of the given test set. Performance metrics are given as median (minimum–maximum) over ten trained models initiated with different seeds; each of the ten models is an ensemble consisting of five model members. Arrows indicate direction of better performance. We compute the metrics as class vs. all. Note that the class MI merges the classes STEMI and NSTEMI in one class. ECE is for multi-class calibration instead of class-wise calibration. STEMI, ST-elevation myocardial infarction; NSTEMI, non-ST-elevation myocardial infarction; AP, Average Precision or equivalently Area Under the Precision-Recall curve; ECE, Expected Calibration Error.

		<b>Random</b>	<b>Temporal</b>	<b>PTB-XL</b>
N (%)	Control	88,742 (98.9)	27,561 (98.7)	200 (72.7)
	STEMI	193 (0.2)	108 (0.4)	25 (27.3)
	NSTEMI	820 (0.9)	263 (0.9)	-

Continued on next page

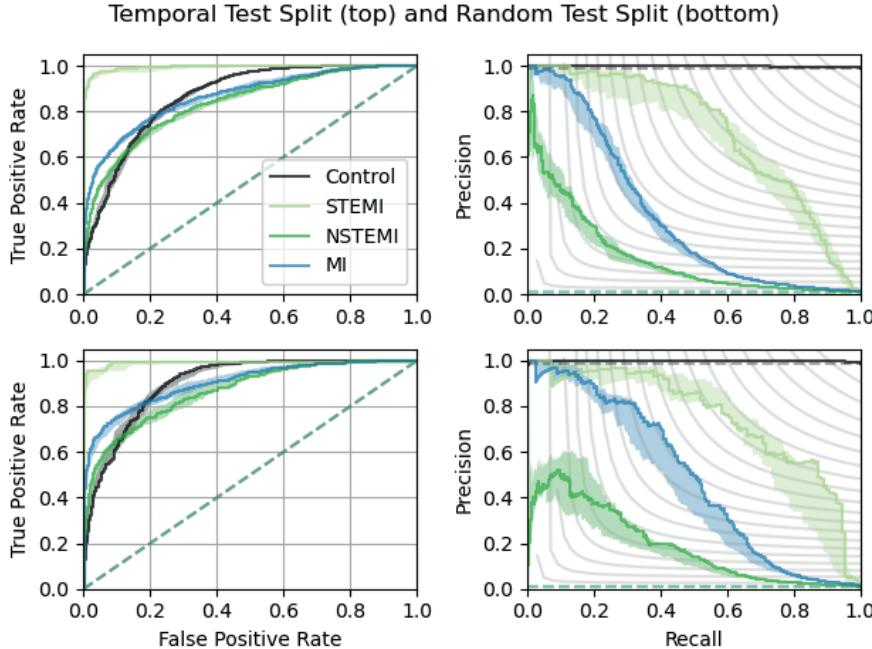
		<b>Random</b>	<b>Temporal</b>	<b>PTB-XL</b>
	MI	1,013 (1.1)	371 (1.3)	-
C-statistic ( $\uparrow$ )	Control	0.863 (0.860-0.870)	0.903 (0.896-0.909)	0.962 (0.953-0.97)
	STEMI	0.991 (0.988-0.994)	0.985 (0.983-0.987)	0.932 (0.904-0.959)
	NSTEMI	0.832 (0.828-0.841)	0.867 (0.859-0.876)	-
	MI	0.863 (0.860-0.870)	0.903 (0.896-0.909)	-
AP ( $\uparrow$ )	Control	0.998 (0.998-0.998)	0.998 (0.998-0.998)	0.955 (0.934-0.972)
	STEMI	0.692 (0.641-0.727)	0.744 (0.716-0.773)	0.954 (0.935-0.971)
	NSTEMI	0.160 (0.134-0.168)	0.184 (0.144-0.214)	-
	MI	0.330 (0.307-0.347)	0.466 (0.42-0.484)	-
Brier ( $\downarrow$ )	Control	0.009 (0.009-0.009)	0.009 (0.009-0.010)	0.145 (0.126-0.158)
	STEMI	0.001 (0.001-0.001)	0.002 (0.002-0.002)	0.184 (0.167-0.196)
	NSTEMI	0.008 (0.008-0.008)	0.008 (0.008-0.009)	-
	Multiclass	0.018 (0.018-0.018)	0.019 (0.019-0.020)	-
ECE ( $\downarrow$ )	Multiclass	0.417 (0.416-0.418)	0.415 (0.415-0.417)	0.277 (0.257-0.297)

To ensure that our model did not use any proxies for its predictions, we stratified the test datasets according to different possible confounders. Supplementary Figure 12 (C-statistic) illustrate results stratified according to test set records (1) without any filter applied based on ST-elevation label noise, (2) ST-elevation filter corresponding to results in main table (used in all following test set subsets); (3) stricter ST-elevation where “possible STEMI” (see Supplementary Methods, Appendix A) were removed from both STEMI and NSTEMI cases, (3) age tertiles, (4) sex, (5) ECG collected at the same day as the admission or not, (6) emergency department visit at Karolinska Hospital (main source of data) or another emergency department in the Stockholm region, (7) patients attending the CCU only, (8) ECGs recorded using the most common machine type (MAC55) or not, (9) ECGs recorded using the most common software (v237) or not. Apart from analyses restricting the test set controls to CCU controls only, we observed no major changes to the model performances, indicating that the predictions are not driven by these potential confounders.

As an additional test, we trained a model with only ECG traces as input, omitting age and sex. The results were almost identical to our main results indicating that the explicit addition of age and sex to the model was not crucial for our results. Furthermore, we compare our results with a model trained on data from patients at the coronary care unit alone. This dataset has 16,628 ECGs, i.e. a subset of 3.4% of our current dataset, omitting most of the control patients which were not admitted to the coronary care unit. The results from this model show that the larger dataset including all controls is important for our performance.

Inspecting Grad-CAM plots yielded new insights. Figure 2 illustrates four STEMIs correctly classified with high probability. In all four panels, the model focused on the ST-segment, which is the segment trained medical staff would focus on to detect a STEMI. But the model also used the down-sloping last part of the T-wave, which is a segment that medical staff would typically not focus on when diagnosing a STEMI. Figure 3 illustrates four correctly classified NSTEMIs. In all panels, the model had focused on the ST-segment, but had also used the last part of the PQ-segment, and the last part of the T-wave, which are segments not used by medical staff to detect a NSTEMI.

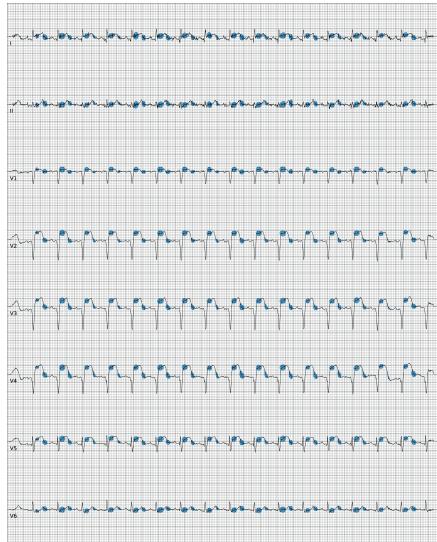
Characteristics of hospitalizations with misclassified ECGs are described in Supplementary Table 6. Among the misclassified ECGs, those misclassified as STEMI more often had perimyocarditis, valvular disease or cardiomyopathy; those misclassified as NSTEMI more often had valvular or congenital heart disease, pulmonary edema, gastric ulcer or dental traits.



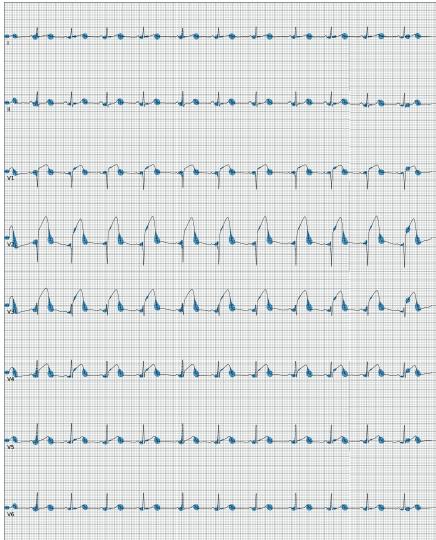
**Figure 1:** Receiver operating characteristics and precision-recall curves. Top row: Temporal test split. Bottom row: Random test split. Left column: Receiver operating characteristics curve. Right Column: Precision-recall curve. We show the curve with the median C-statistic or AP respectively as a solid line over-trained models with 10 seeds of our ensemble-based model. The shaded area shows the min and max values for all 10 models. The dashed lines indicate the curve corresponding to a random guess for the class in the given color according the legend. For the Precision-Recall curve this is a horizontal line at the value (number of positive examples)/(number of all examples). Note that the worst-case curves are overlapping for NSTEMI, STEMI and MI in the precision recall curves due to the small fraction of positives (see dashed line close to the bottom). In the Precision-Recall curves the grey curves depict iso-F1 curves. Each curve is class vs all, where MI specifically is STEMI and NSTEMI vs Control.



(a)



(b)



(c)



(d)

**Figure 2:** Four representative STEMIs correctly classified with high probability. Four Grad-CAM plots of STEMIs correctly classified with high probability, highlighting the parts in the ECG that the model focuses on for its prediction. Gradients corresponding to the activations in the first convolutional layer of the neural network are averaged to get the proportional importance of each channel, which is then used to compute a proportional mean of the activations. Positive values obtained were plotted as blue disks overlaid on top of the ECG, with size proportional to its magnitude.

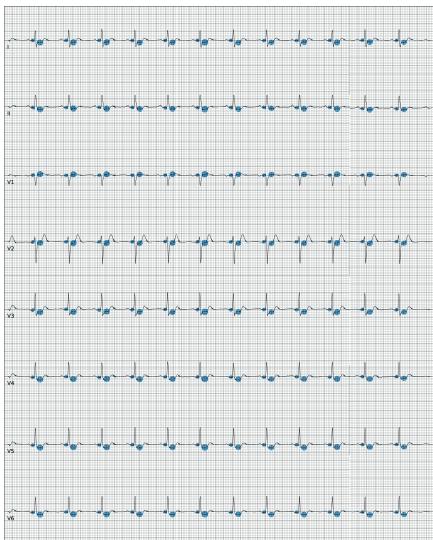
Paper VI – Development and validation of deep learning ECG-based prediction of myocardial infarction in emergency department patients



(a)



(b)



(c)



(d)

**Figure 3:** Four representative NSTEMIs correctly classified with high probability. Four Grad-CAM plots of NSTEMIs correctly classified with high probability, highlighting the parts in the ECG that the model focuses on for its prediction. Gradients corresponding to the activations in the first convolutional layer of the neural network are averaged to get the proportional importance of each channel, which is then used to compute a proportional mean of the activations. Positive values obtained were plotted as blue disks overlaid on top of the ECG, with size proportional to its magnitude.

### 3 Discussion

In a very large sample of all-comers to emergency departments, we developed and validated an AI model that can discriminate STEMI and NSTEMI from non-myocardial infarctions using routine 10-s ECGs.

The model achieved excellent performance for both STEMI and NSTEMIs. Doctors' ECG interpretation is often imprecise, with a reported accuracy of 0.69 overall for practicing physicians and 0.75 for cardiologists in controlled test settings [COP20], with similar numbers reported for STEMIs [McC+13; Soa+19]. Performance of doctors outside of such standardized settings is unknown but not likely better. In the context of the model's performance outside the study population, we do not observe a worse performance in the temporal test set when compared to the random test set. This suggests that our model works on data outside of the training data. In addition, it performs well in the discrimination of STEMIs in the external PTB-XL database test set. Unlike STEMI, diagnosis by humans of NSTEMIs from ECGs is almost by definition futile and has seen very little research.

This study is clinically important as it uses the most relevant sample possible. These consecutive all-comers represent the real-world ECG experience for emergency doctors, with ECGs—especially among the non-infarctions—that are far from the very clear specimens in managed online databases or heavily curated samples. Notably, in Sweden today and during the study period, pre-hospital ECGs are sent to coronary care units for immediate diagnosis, so the obvious STEMIs usually bypass the emergency department and transfer straight to the coronary intervention lab upon arrival to hospital, rendering the STEMIs in the present study the less obvious cases and the walk-ins. Hence, these are cases in great need of decision support. Further, we did not exclude difficult cases, comorbidities, or previous myocardial infarctions (except for technical reasons, we removed potentially linked hospitalizations for the same myocardial infarction, and LBBBs, which cannot per se identify an acute myocardial infarction from a single ECG, but need a prior ECG for comparison).

Other studies using deep learning models have also shown good performance but lower than our model's performance when tested in representative settings [Liu+21], with reports of very good and similar to our model's performance in managed settings [Liu+21; Cho+20]. Our study differs from the previous literature in that it is a multicenter study using real-world data of consecutive patients with very few exclusions, and with output labeling by many doctors. Descriptions of the clinical setting and the controls are sometimes unclear [Cho+20]. Past studies have seldom investigated NSTEMIs.

In general, it is difficult to determine which model performs best in practice since there exists no publicly available test dataset and therefore no direct comparison to other studies. However, in terms of the used models, while other

studies either use a more simple convolutional network without residual connections [Cho+20] or assume that each lead is independent by using a ResNet for each lead [Liu+21], we do not include this assumption in our ResNet and therefore present the most general model. We tried to keep the model as standardized as possible with minimal modifications from a standard ResNet architecture to improve reproducibility.

Our model performed slightly better in younger than older patients for NSTEMI classification, and slightly better for STEMI discrimination in men than in women. Younger NSTEMI patients might have fewer underlying diseases, potentially making the prediction simpler. Men contribute to around 2/3 of the myocardial infarctions, potentially making male infarctions easier to learn given more data. A stricter filter for label noise (based on a mismatch between original labels and updated ST-elevation annotations as described in the Supplementary material) shows that a stricter filter improved the result for STEMI whereas the opposite was seen for NSTEMI, but the differences were minor. As expected, the more pronounced the ST-elevation was, as evaluated by a senior cardiologist, the higher was the model’s predicted probability of STEMI as seen in Supplementary Figure 11. For NSTEMI, then difference in predicted probabilities was smaller when applying a stricter filter for label noise. Overall, the results were comparable across test set subsets and some differences in prediction results might be due to pure chance. A notable difference in the analyses restricting the test set controls to CCU controls only may be due to more underlying heart conditions in those controls compared with the large number of controls not admitted to the CCU.

The Grad-CAM plots in Figure 2 and Figure 3 provide important insights. The model recognizes the same ST-segment features that humans would. But the model also finds features that are novel, or that are imperceptible to the human eye. This shows an interesting way forward. We give the AI ECGs and the label, then the AI teaches us novel ways to read the ECGs. Variants of such model evaluation can likely give useful clinical and pathophysiological clues in many medical fields.

Our model’s misclassifications as STEMI follows known clinical and machine learning patterns, with per-imyocarditis as an important impostor [Tan+19]. The conditions over-represented in those misclassified as NSTEMI were logical to some extent, such as valvular or congenital heart disease and pulmonary edema; the gastric ulcer and dental traits more surprising.

Some important limitations are worth mentioning. Our dataset contains some label noise. The label was determined at discharge from the coronary care unit or emergency room when the whole care episode could be summarized. The ECGs in the test sets of this study may hence not always be the ones guiding the final diagnosis. We mitigate that to some extent by using multiple ECGs if available within the day before admission in the training set, but not in the

test sets. On the other hand, the hindsight allows for more stable labels for the episode as a whole, which is the ultimate goal of the classification. More information about the handling of label noise is provided in the Supplementary materials. Another important limitation is the lack of an external validation sample. We did hold out the 10% of the patients with their first admission in 2016 as a temporal test set; many circumstances in that set would be similar to those in the earlier training set, but a restructuring of the Stockholm region emergency department logistics which is our main data source during the data collection period did change the composition of the sample. Furthermore, we make use of the publicly available PTB-XL dataset which does include data containing STEMI but not NSTEMI. In this dataset, our model achieves good discriminative performance. No publicly available data repositories contain ECGs with NSTEMI labels to test this externally. While the calibration of our model was better than that of comparable models, there is still room for improvement; calibration is indeed an underappreciated property in general. We did not consider transferring learned features, only model architecture, from a previous study [Rib+20]. An exploration of potential improvements in model convergence speed and final performance boost by pre-training on a different ECG classification task with a dataset in a different context may be useful, but may also introduce model biases from the other dataset. Lastly, we did not compare the performance of this ECG model to troponin-based or other methods of diagnosing myocardial infarction. While this study shows that both STEMI and NSTEMI discrimination can be performed on a general, non-curated population of emergency department patients, further research is needed for clinical use. This includes among other points, (1) the performance improvement in terms of precision for the NSTEMI class to reduce the number of false-positive predictions; (2) the improvement of the model calibration for meaningful predicted probabilities; (3) clinical usefulness should be evaluated in a randomized trial; and (4) understanding of needs of training of medical staff to work in a collaborative environment with model predictions as decision support tools. Already at this stage, we believe that this study shows the power of deep models for the general, real-world population of patients for this task. With the use of a standardized model structure from the deep learning community, we ensure that future research can be boosted from our model to obtain a clinically useful model in the near future.

In conclusion, we developed and validated a deep learning model with excellent performance in discriminating between NSTEMI, STEMI and controls on the presenting ECG of a large real-world sample of general emergency department patients. Considering the high and rising emergency department care costs and the high numbers of missed myocardial infarctions at emergency departments, our model could be of clinical value for ECG decision support, with promise of further performance development.

## 4 Methods

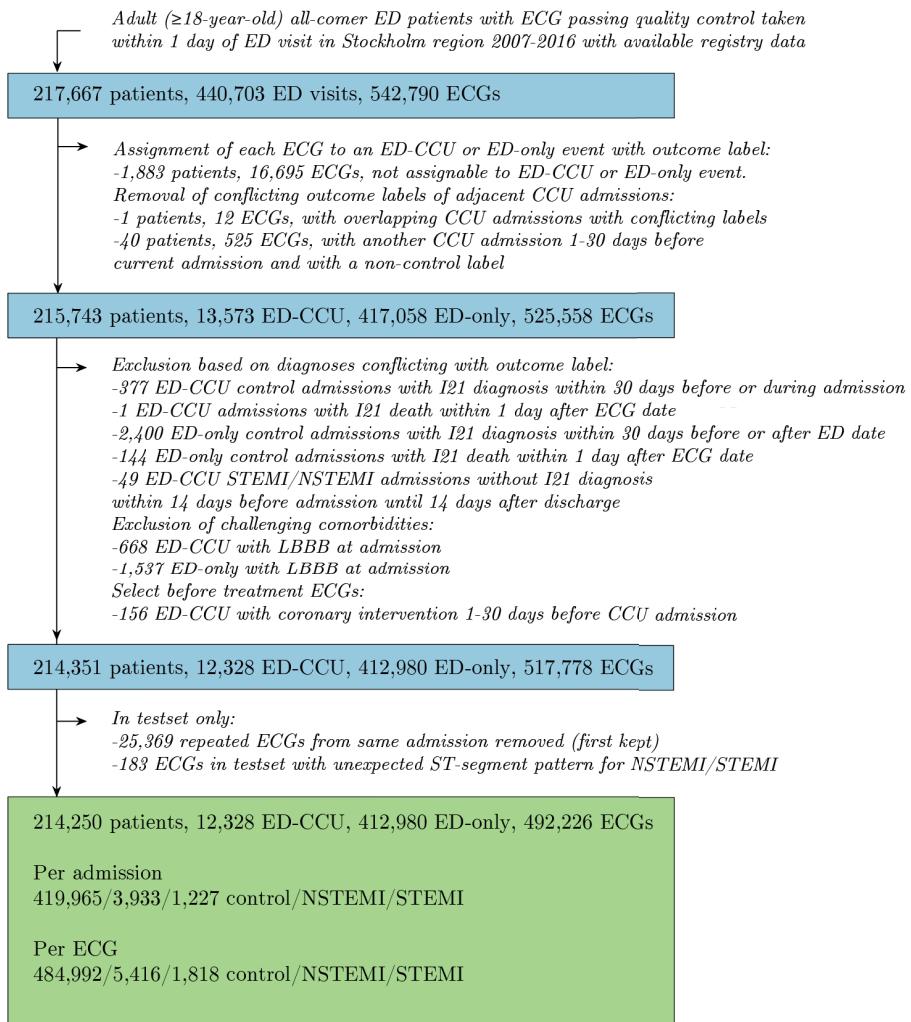
### 4.1 Study sample

We utilized a consecutive sample of adult all-comer patients attending emergency departments in the Stockholm region between 2007 and 2016, for whom a routine ECG was obtained upon their presenting complaint. Details of the data sources are described in the Supplementary Methods, Appendix A. The procedure and criteria used to define the study sample are described in Figure 4, with available predictor and outcome data described below and in Supplementary Methods, Appendix A. In total, 217,667 patients had at least one registered emergency department visit within the study period with at least one valid ECG recording within 1 day of visit. The emergency department visits were either followed by a coronary care unit (CCU) admission (NSTEMI/STEMI/ control) or had no subsequent CCU admission (control only). After applying the sequence of filters described in Figure 4 to ensure inclusion of at-event before-treatment ECGs, as well as confirming the outcome label, 214,250 patients with 492,226 ECGs were available for analysis, representing a total of 12,328 CCU admissions and 412,980 non-CCU visits. Out of these ECGs, 67,137 exams were repeated recordings, i.e. the same patient had multiple ECG exams in the same visit (used in training the model only). The study was approved by the Swedish Ethical Review Authority, application number 2020–01654. Informed consent was waived in this study by all applicable ethics review boards, i.e. the Region Stockholm Ethical Review Board and the Swedish Ethical Review Authority. All methods were performed in accordance with the relevant guidelines and regulations.

### 4.2 Data sources

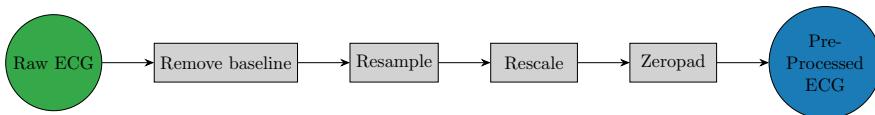
Data on the predictors and outcome of the trained model were available for all included patients from discharge records from the emergency departments, electronic health records, from linked hospitalizations, and from the SWEDEHEART registry with patient records joined on the Swedish personal identifier number of the patient. Data sources and definitions used are described in Supplementary Methods, Appendix A and Supplementary Table 3 and Table 4. We only included patients with complete data on the predictors and outcome in the analyses. Predictors. As predictors in the model, we used digital ECG data, age and sex, as in a previous study [Cho+20], to make the model as transportable and unbiased as possible. Standard 10-s 12-lead ECG recordings sampled at 250–500 Hz were used; 8 leads were used in the present study as 4 of the standard leads are linear combinations of these 8 and are hence redundant. All digital ECGs were available in the GE MUSE (v9) XML format. The baseline

was removed by filtering the tracing with a high-pass filter and all ECGs were then resampled



**Figure 4:** Derivation of the study sample. Inclusion/exclusion criteria applied to define the study sample. SWEDEHEART, Swedish Web-system for Enhancement and Development of Evidence-based care in Heart disease Evaluated According to Recommended Therapies; RIKS-HIA, Register of Information and Knowledge About Swedish Heart Intensive Care Admissions; CCU, coronary care unit; ED, emergency department; ED-CCU, ED visit with CCU admission and ED date within 0–1 days prior to CCU admission and ED-CCU admission; ED-only, ED visit without any CCU admission within ±30 days of ECG recording; NSTEMI, non-ST-elevation myocardial infarction; STEMI, ST-elevation myocardial infarction.

to 400 Hz and zero-padded to a fixed length of 4096 samples. ECGs where one or more required leads were missing or contained all-zero entries were removed. Data pre-processing steps are described in more detail in Figure 5. Age is normalized in all data to a zero-mean unit variance variable using the mean age and standard deviation from the training set and concatenated with sex as binary variable.



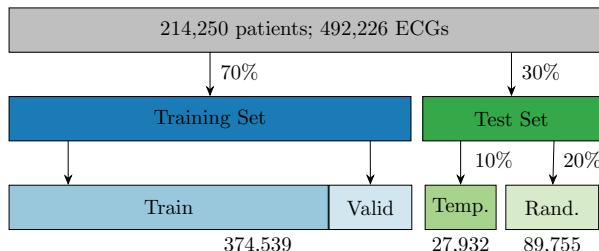
**Figure 5:** Data pre-processing of ECGs. From the raw input ECG we removed the baseline by filtering the tracing with a high-pass filter to remove biases and low-frequency trends. The filter is an elliptic filter with a cut-off frequency of 0.8 Hz and an attenuation of 40 dB, applied to the forward and reverse direction to obtain zero-phase distortion. We then resampled all ECGs to 400 Hz and zero-padded to a fixed length of 4096 samples, since the convolution-based model requires a fixed input size. For duplicated ECGs with identical data and collection time, the first copy was kept. ECGs where one or more required leads were missing or contained all-zero entries were removed. We used one-hot encoding for sex and normalized the age with the mean and standard deviation of the training dataset.

### 4.3 Outcome

For the mutually exclusive outcome labels NSTEMI/STEMI/control status we used the high-quality SWEDEHEART registry for the CCU admissions, primarily to define NSTEMI/STEMI of the myocardial infarction cases. In addition, diagnoses from the Swedish in-patient and cause-of-death registries were used to confirm a myocardial infarction (I21) for cases, or absence of a myocardial infarction for controls (for all patients). The SWEDEHEART Riks-HIA labels which were used to define NSTEMI/STEMI are the decision of a discharging physician that followed the entire patient journey during the hospitalization; this physician had access to other exams besides the ECG, such as coronary angiograms in some patients, and blood testing in all patients. This label is the accepted gold standard for all research using SWEDEHEART data; efforts to further minimize label noise are described in Supplementary Methods, Appendix A.

## 4.4 Training, validation, and test sets of the study sample

The patients fulfilling the inclusion criteria were divided in 70%/30% splits with records from the same patient always put in the same split. The 70% split was used for training and developing the model and the 30% split, without any patient overlap, for testing the model performance. We set aside a total of 10% of the training set for validation. The 30% test split was further divided into two splits containing 20% and 10% of the study sample, to allow us to test the model in two different scenarios. The 10% split contains patients with a first recorded admission date after 2016-01-01 or later, hence temporally separated from the training data set which included patients with a first recorded admission before this date. This way the 10% test split can be used to assess the model susceptibility to trends that change with time. We denote this split the temporal test split. The other 20% split was sampled at random from entries with an admission date before 2016-01-01, which is the same period the 70% training split is sampled from. We denote this split the random test split. The process is illustrated in Figure 6 and patient characteristics in the sets are presented in Supplementary Table 5. Our temporal test set can be considered the best possible external (temporal) validation available for the NSTEMI cases since there exists no other publicly available data set available containing ECGs and NSTEMI annotations.



**Figure 6:** Data splitting of the study sample. The dataset was split in a training set consisting of 70% and two test sets consisting of 10% and 20% of the complete data each. The last row in the figure lists the total number of ECGs in each data split. Repeated ECG recordings at the same visit of a patient were added to the training set only.

## 4.5 Test set of the external PTB-XL database

In addition to the study sample, an external validation dataset with 75 acute myocardial infarction cases with ST-elevation and 200 randomly selected controls was manually curated from the PTB-XL database [Wag+20; Gol+00]. The PTB-XL is a publicly available database of 21,837 10-s 12-lead ECGs

annotated with 71 different ECG statements, including cases of myocardial infarction. The inclusion criteria of the PTB-XL test set used in this study is described in the Supplementary Methods, Appendix A.

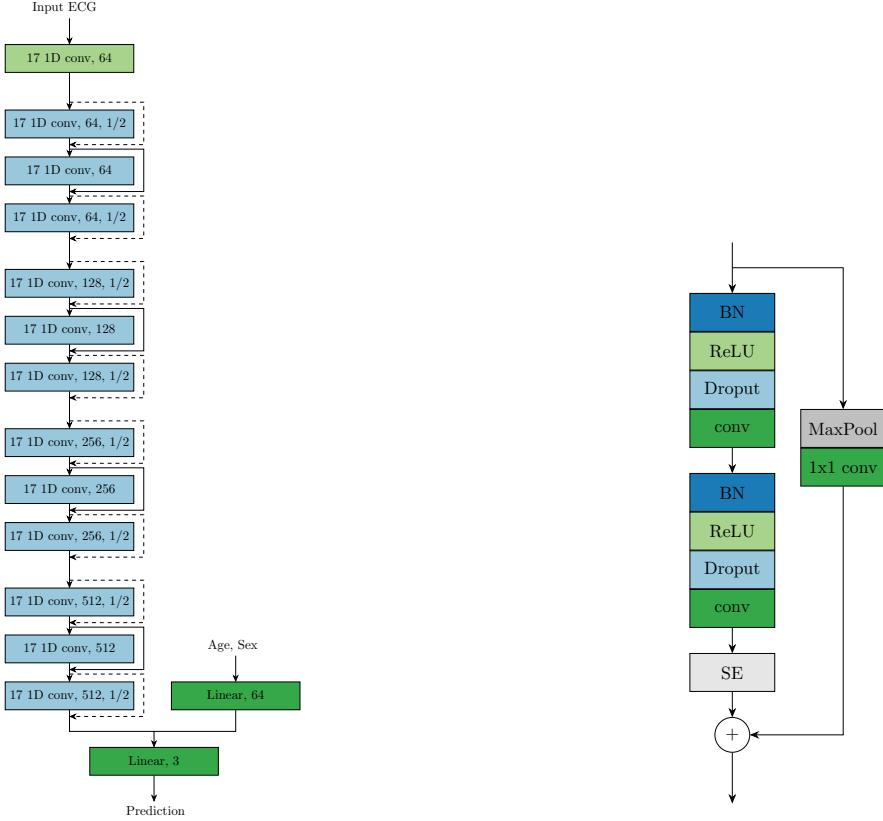
## 4.6 Model architecture

Our Deep Neural Network (DNN) model architecture is an extension of a previous model architecture [Rib+20], for which the DNN was trained to detect six types of ECG abnormalities [Alk+12]. The full model architecture is depicted and described in Figure 7. In the first part of the model, to process the raw ECGs, we used a neural network based on convolutional layers similar to a residual network (ResNet) that is commonly used in image classification, but adapted here to unidimensional signals. This architecture allows DNNs to be efficiently trained by including skip connections. We adapt a modification of layer arrangements within the residual block and a skip connection which is shown to be more effective [He+16].

In the second part of the model, we concatenate age with sex and pass it through a fully connected layer. The output of both model parts are flattened and concatenated to obtain one feature vector. The resulting features were used in the final linear classification layer which outputs the model prediction. The output of the trained model was the probabilities of the three mutually exclusive outcome classes NSTEMI/STEMI/control. Ensembles of neural network models improve predictive performance [HS90], and lead to better calibrated models. Therefore, we expanded our model as an ensemble of five model members. We followed the ensemble strategy of Lakshminarayanan et al. [LPB17] by using the same model structure and training method for each ensemble member. The only difference between the different ensemble members are the random initializations of the model parameters. This leads the model during the optimization process to reach different minima in the optimization loss landscape. Hence, the logits as outputs of the different ensemble members will be different. These logits were averaged to obtain the final prediction. A detailed description of all model hyperparameters is provided in the Supplementary Methods, Appendix A.

## 4.7 Model training and validation

The model was trained by minimizing the cross-entropy loss for 100 epochs. Details about the training hyperparameter and regularization terms are described in the Supplementary Methods, Appendix A. In the training dataset, we made use of the repeated ECG exams of patients who had multiple exams



**Figure 7:** Deep neural network model architecture. The left panel is a high-level model architecture for ECG classification consisting of one part to extract features from the ECG exam and one for features from phenotypes age and sex. The light green block contains a convolutional layer followed by a batch normalization for rescaling the output and a ReLU activation function. This layer is followed by four sets of each three residual blocks in light blue, i.e. 12 residual blocks in total. The name of the block indicates the filter size of the convolutions, the number of filters and the down-sampling factor (if applicable). Note that we downsample the signal by a factor of 1/2 in the beginning and ending of each set of residual blocks. The right panel illustrates the content of each residual block. Dropout is used after each nonlinear activation function as regularization. Only the first residual block does not contain the first batch normalization, ReLU and dropout layer since these layers are already applied after the initial convolution layer. We rearrange layers from the initial architecture [Rib+20] and extend it with Squeeze and Excite (SE) blocks. This operation helps to weight the channel-wise information. Downsampling residual blocks consist in the residual skip connection (dashed lines) of a Max Pooling operation followed by a convolutional layer with filter length 1 to match the dimensions with the main branch for the summation. The remaining skip connections (full lines) do not contain any operations since input and output dimensions of the residual block are equal.

at an emergency department visit, but not for validation and testing. We consider this as a form of data augmentation since these exams have the same label but are recorded at different times and therefore with a slightly different state of the patient and possibly different placement of the ECG leads. For validation and testing, only the first recorded ECG of an admission was used. Details are in the Supplementary Methods, Appendix A. Note that we improve the training procedure and model architecture from a previous study [Rib+20] with established methods [Bel+21]. The details for the model and training improvements as well as all hyperparameters for model training such as learning rate and regularization parameters are described in Supplementary Methods, Appendix A.

## 4.8 Hyperparameter search

We conducted a search over 6 hyperparameters and their respective values. This gives a total of 2025 possible combinations. Due to the high number of possible combinations, we ran a random search for 345 combinations. The remaining model and training hyperparameters were fixed in this search since they have been proven efficient in a previous study with a model architecture which the present study extends [Rib+20]. We evaluated the tested hyperparameters in the validation dataset. Note that during the search we only train a single model and not an ensemble of multiple models. Details of the hyperparameter search and its related computational cost are outlined in the Supplementary Methods, Appendix A.

## 4.9 Model discrimination and calibration

Performance of prediction models is often quantified using the C-statistic that measures discrimination, which is also used in the present study, i.e. the model’s ability to assign higher risk estimates/probabilities to patients who will experience the event than patients who will not. While discrimination is important, it tells us nothing about the reliability of the probability estimates. Calibration, i.e. if the model’s probability estimates reflect the ground truth empirical class frequencies, is a more important property of the model for clinical use. One of our main concerns was model calibration, since modern DNN architectures are generally poorly calibrated [Guo+17]. As metrics of model calibration, we focused on the Expected Calibration Error (ECE) and the Brier score. The ECE is the weighted absolute difference between the class membership and the estimated probability for that class averaged over 15 bins, while the Brier score measures the average squared error on the probability scale. Both metrics are applicable to both binary and multiclass

problems. We visualized the calibration of our model using calibration plots, also called reliability diagrams.

## 4.10 Evaluation of correct versus incorrect high-probability classifications

We investigated possible patterns for our models correct and incorrect classifications of STEMIs and NSTEMIs. First, we used Grad-CAM plots to highlight the parts of the ECG that the model focuses on—meaning the sections of the input ECG where the model puts its most weight on—to make its predictions [Sel+17]. This method generates the visualization in two steps: In a forward pass step, it computes the activations of the neural network in a given intermediary layer (in our case, the first convolutional layer). And in a backward step, it computes the gradients corresponding to these activations. The gradients are averaged to get the proportional importance of each channel, which is then used to compute a proportional mean of the activations. Positive values obtained were plotted as purple disks overlaid on top of the ECG, with size proportional to its magnitude, to generate the visualization. One senior cardiologist (JS) inspected the Grad-CAM plots from the ten cases with the highest estimated probability for each myocardial infarction class and selected four illustrative plots per class. We tested the over/underrepresentation of inpatient and specialized outpatient care diagnoses at the time of the visit among correctly classified versus misclassified patients with a predicted probability  $>0.5$  in pairwise independent tests to find underlying diagnoses where the model often made incorrect classifications.

## References

- [Al-Zaiti+20] S. Al-Zaiti, L. Besomi, Z. Bouzid, Z. Faramand, S. Frisch, C. Martin-Gill, R. Gregg, S. Saba, C. Callaway, and E. Sejdić. “Machine learning-based prediction of acute coronary syndrome using only the pre-hospital 12-lead electrocardiogram.” In: *Nature communications* 11.1 (2020), p. 3966 (cit. on p. VI-2).
- [Alkmim+12] M. B. Alkmim, R. M. Figueira, M. S. Marcolino, C. S. Cardoso, M. P. d. Abreu, L. R. Cunha, D. F. d. Cunha, A. P. Antunes, A. G. d. A. Resende, E. S. Resende, et al. “Improving patient access to specialized health care: the Telehealth Network of Minas Gerais, Brazil.” In: *Bulletin of the World Health Organization* 90 (2012), pp. 373–378 (cit. on p. VI-18).

Paper VI – Development and validation of deep learning ECG-based prediction of myocardial infarction in emergency department patients

- [Att+19] Z. I. Attia, P. A. Noseworthy, F. Lopez-Jimenez, S. J. Asirvatham, A. J. Deshmukh, B. J. Gersh, R. E. Carter, X. Yao, A. A. Rabinstein, B. J. Erickson, et al. “An artificial intelligence-enabled ECG algorithm for the identification of patients with atrial fibrillation during sinus rhythm: a retrospective analysis of outcome prediction.” In: *The Lancet* 394.10201 (2019), pp. 861–867 (cit. on p. VI-2).
- [Bel+21] I. Bello, W. Fedus, X. Du, E. D. Cubuk, A. Srinivas, T.-Y. Lin, J. Shlens, and B. Zoph. “Revisiting resnets: Improved training and scaling strategies.” In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 22614–22627 (cit. on pp. VI-20, VI-30).
- [Cho+20] Y. Cho, J.-m. Kwon, K.-H. Kim, J. R. Medina-Inojosa, K.-H. Jeon, S. Cho, S. Y. Lee, J. Park, and B.-H. Oh. “Artificial intelligence algorithm for detecting myocardial infarction using six-lead electrocardiography.” In: *Scientific reports* 10.1 (2020), p. 20495 (cit. on pp. VI-2, VI-11, VI-12, VI-14).
- [Coh+21] M. Cohen-Shelly, Z. I. Attia, P. A. Friedman, S. Ito, B. A. Es-sayagh, W.-Y. Ko, D. H. Murphree, H. I. Michelena, M. Enriquez-Sarano, R. E. Carter, et al. “Electrocardiogram screening for aortic valve stenosis using artificial intelligence.” In: *European heart journal* 42.30 (2021), pp. 2885–2896 (cit. on p. VI-2).
- [COP20] D. A. Cook, S.-Y. Oh, and M. V. Pusic. “Accuracy of physicians’ electrocardiogram interpretations: a systematic review and meta-analysis.” In: *JAMA internal medicine* 180.11 (2020), pp. 1461–1471 (cit. on p. VI-11).
- [CS18] C. A. Caulfield and J. R. Stephens. “Things We Do for No Reason: Hospitalization for the Evaluation of Patients with Low-Risk Chest Pain.” In: *Journal of Hospital Medicine* 13.4 (2018), pp. 277–279 (cit. on p. VI-2).
- [Ebr+20] Z. Ebrahimi, M. Loni, M. Daneshtalab, and A. Gharehbaghi. “A review on deep learning methods for ECG arrhythmia classification.” In: *Expert Systems with Applications: X* 7 (2020), p. 100033 (cit. on p. VI-2).
- [Gol+00] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. “PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals.” In: *circulation* 101.23 (2000), e215–e220 (cit. on pp. VI-17, VI-31).

- [GP16] J. E. Galarraga and J. M. Pines. “Costs of ED episodes of care in the United States.” In: *The American journal of emergency medicine* 34.3 (2016), pp. 357–365 (cit. on p. VI-2).
- [Guo+17] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. “On calibration of modern neural networks.” In: *International conference on machine learning*. PMLR. 2017, pp. 1321–1330 (cit. on pp. VI-20, VI-31, VI-41).
- [He+16] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778 (cit. on pp. VI-2, VI-18, VI-30).
- [Hon+20] S. Hong, Y. Zhou, J. Shang, C. Xiao, and J. Sun. “Opportunities and challenges of deep learning methods for electrocardiogram data: A systematic review.” In: *Computers in biology and medicine* 122 (2020), p. 103801 (cit. on p. VI-2).
- [Hot+08] T. Hothorn, K. Hornik, M. A. Van De Wiel, and A. Zeileis. “Implementing a class of permutation tests: the coin package.” In: *Journal of statistical software* 28.8 (2008), pp. 1–23 (cit. on p. VI-37).
- [HS90] L. K. Hansen and P. Salamon. “Neural network ensembles.” In: *IEEE transactions on pattern analysis and machine intelligence* 12.10 (1990), pp. 993–1001 (cit. on p. VI-18).
- [HSS18] J. Hu, L. Shen, and G. Sun. “Squeeze-and-excitation networks.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141 (cit. on p. VI-29).
- [Jer+10] T. Jernberg, M. F. Attebring, K. Hambraeus, T. Ivert, S. James, A. Jeppsson, B. Lagerqvist, B. Lindahl, U. Steneström, and L. Wallentin. “The Swedish Web-system for enhancement and development of evidence-based care in heart disease evaluated according to recommended therapies (SWEDEHEART).” In: *Heart* (2010) (cit. on p. VI-28).
- [Kul+19] M. Kull, M. Perello Nieto, M. Kängsepp, T. Silva Filho, H. Song, and P. Flach. “Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration.” In: *Advances in neural information processing systems* 32 (2019) (cit. on pp. VI-31, VI-41).
- [Lan+20] B. H. Lane, P. J. Mallow, M. B. Hooker, and E. Hooker. “Trends in United States emergency department visits and associated charges from 2010 to 2016.” In: *The American journal of emergency medicine* 38.8 (2020), pp. 1576–1581 (cit. on p. VI-2).

Paper VI – Development and validation of deep learning ECG-based prediction of myocardial infarction in emergency department patients

- [Liu+19] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han. “On the variance of the adaptive learning rate and beyond.” In: *arXiv preprint arXiv:1908.03265* (2019) (cit. on p. VI-30).
- [Liu+21] W.-C. Liu, C.-S. Lin, C.-S. Tsai, T.-P. Tsao, C.-C. Cheng, J.-T. Liou, W.-S. Lin, S.-M. Cheng, Y.-S. Lou, C.-C. Lee, et al. “A deep learning algorithm for detecting acute myocardial infarction.” In: *EuroIntervention* 17.9 (2021), pp. 765–773 (cit. on pp. VI-2, VI-11, VI-12).
- [LPB17] B. Lakshminarayanan, A. Pritzel, and C. Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles.” In: *Advances in neural information processing systems* 30 (2017) (cit. on p. VI-18).
- [Mak+20] H. Makimoto, M. Höckmann, T. Lin, D. Glöckner, S. Gerguri, L. Clasen, J. Schmidt, A. Assadi-Schmidt, A. Bejinariu, P. Müller, et al. “Performance of a convolutional neural network derived from an ECG database in recognizing myocardial infarction.” In: *Scientific reports* 10.1 (2020), p. 8445 (cit. on p. VI-2).
- [McC+13] J. M. McCabe, E. J. Armstrong, I. Ku, A. Kulkarni, K. S. Hoffmayer, P. D. Bhave, S. W. Waldo, P. Hsue, J. C. Stein, G. M. Marcus, et al. “Physician accuracy in interpreting potential ST-segment elevation myocardial infarction electrocardiograms.” In: *Journal of the American Heart Association* 2.5 (2013), e000268 (cit. on p. VI-11).
- [Med+16] L. Medford-Davis, E. Park, G. Shlamovitz, J. Suliburk, A. N. Meyer, and H. Singh. “Diagnostic errors related to acute abdominal pain in the emergency department.” In: *Emergency Medicine Journal* (2016) (cit. on p. VI-2).
- [Moo+17] P.-J. Moonen, L. Mercelina, W. Boer, and T. Fret. “Diagnostic error in the Emergency Department: follow up of patients with minor trauma in the outpatient clinic.” In: *Scandinavian journal of trauma, resuscitation and emergency medicine* 25 (2017), pp. 1–7 (cit. on p. VI-2).
- [Rag+20] S. Raghunath, A. E. Ulloa Cerna, L. Jing, D. P. VanMaanen, J. Stough, D. N. Hartzel, J. B. Leader, H. L. Kirchner, M. C. Stumpe, A. Hafez, et al. “Prediction of mortality from 12-lead electrocardiogram voltage data using a deep neural network.” In: *Nature medicine* 26.6 (2020), pp. 886–891 (cit. on p. VI-2).
- [Rib+20] A. H. Ribeiro, M. H. Ribeiro, G. M. M. Paixão, D. M. Oliveira, P. R. Gomes, J. A. Canazart, M. P. S. Ferreira, C. R. Andersson, P. W. Macfarlane, W. Meira Jr., T. B. Schön, and A. L. P. Ribeiro. “Automatic diagnosis of the 12-lead ECG using a deep neural net-

- work.” In: *Nature Communications* 11 (2020), p. 1760 (cit. on pp. VI-2, VI-5, VI-13, VI-18, VI-19, VI-20, VI-29).
- [Sel+17] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. “Grad-cam: Visual explanations from deep networks via gradient-based localization.” In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626 (cit. on p. VI-21).
- [Sha+21] A. L. Sharp, A. Baecker, N. Nassery, S. Park, A. Hassoon, M.-S. Lee, S. Peterson, S. Pitts, Z. Wang, Y. Zhu, et al. “Missed acute myocardial infarction in the emergency department-standardizing measurement of misdiagnosis-related harms using the SPADE method.” In: *Diagnosis* 8.2 (2021), pp. 177–186 (cit. on p. VI-2).
- [Sio+21] K. C. Siontis, P. A. Noseworthy, Z. I. Attia, and P. A. Friedman. “Artificial intelligence-enhanced electrocardiography in cardiovascular disease management.” In: *Nature Reviews Cardiology* 18.7 (2021), pp. 465–478 (cit. on p. VI-2).
- [Soa+19] W. E. Soares III, L. L. Price, B. Prast, E. Tarbox, T. J. Mader, and R. Blanchard. “Accuracy screening for ST elevation myocardial infarction in a task-switching simulation.” In: *Western Journal of Emergency Medicine* 20.1 (2019), p. 177 (cit. on p. VI-11).
- [SWE] SWEDEHEART. *variable list*. <https://www.ucr.edu/swedeheart/dokument-sh/variabellista>. Accessed: 2014-01-16 (cit. on p. VI-28).
- [Szu+17] K. Szummer, L. Wallentin, L. Lindhagen, J. Alfredsson, D. Erlinge, C. Held, S. James, T. Kellerth, B. Lindahl, A. Ravn-Fischer, et al. “Improved outcomes in patients with ST-elevation myocardial infarction during the last 20 years are related to implementation of evidence-based treatments: experiences from the SWEDEHEART registry 1995–2014.” In: *European heart journal* 38.41 (2017), pp. 3056–3065 (cit. on p. VI-28).
- [Szu+18] K. Szummer, L. Wallentin, L. Lindhagen, J. Alfredsson, D. Erlinge, C. Held, S. James, T. Kellerth, B. Lindahl, A. Ravn-Fischer, et al. “Relations between implementation of new treatments and improved outcomes in patients with non-ST-elevation myocardial infarction during the last 20 years: experiences from SWEDEHEART registry 1995 to 2014.” In: *European heart journal* 39.42 (2018), pp. 3766–3776 (cit. on p. VI-28).
- [Tan+19] A. Tanguay, J. Lebon, E. Brassard, D. Hébert, and F. Bégin. “Diagnostic accuracy of prehospital electrocardiograms interpreted remotely by emergency physicians in myocardial infarction patients.” In: *The American Journal of Emergency Medicine* 37.7 (2019), pp. 1242–1247 (cit. on p. VI-12).

Paper VI – Development and validation of deep learning ECG-based prediction of myocardial infarction in emergency department patients

- [Tis+19] G. H. Tison, J. Zhang, F. N. Delling, and R. C. Deo. “Automated and interpretable patient ECG profiles for disease detection, tracking, and discovery.” In: *Circulation: Cardiovascular Quality and Outcomes* 12.9 (2019), e005289 (cit. on p. VI-2).
- [Ugg+20] B. af Uggla, T. Djärv, P. L. Ljungman, and M. J. Holzmann. “Association between hospital bed occupancy and outcomes in emergency care: a cohort study in Stockholm region, Sweden, 2012 to 2016.” In: *Annals of Emergency Medicine* 76.2 (2020), pp. 179–190 (cit. on p. VI-28).
- [Wag+20] P. Wagner, N. Strothoff, R.-D. Bousseljot, D. Kreiseler, F. I. Lunze, W. Samek, and T. Schaeffter. “PTB-XL, a large publicly available electrocardiography dataset.” In: *Scientific data* 7.1 (2020), p. 154 (cit. on pp. VI-17, VI-31).
- [Wri+19] B. Wright, N. Faulkner, P. Bragge, and M. Graber. “What interventions could reduce diagnostic error in emergency departments? A review of evidence, practice and consumer perspectives.” In: *Diagnosis* 6.4 (2019), pp. 325–334 (cit. on p. VI-2).
- [XLC22] P. Xiong, S. M.-Y. Lee, and G. Chan. “Deep learning for detecting and locating myocardial infarction by electrocardiogram: A literature review.” In: *Frontiers in cardiovascular medicine* 9 (2022), p. 860032 (cit. on p. VI-2).
- [Zha+20] Y. Zhao, J. Xiong, Y. Hou, M. Zhu, Y. Lu, Y. Xu, J. Teliewubai, W. Liu, X. Xu, X. Li, et al. “Early detection of ST-segment elevated myocardial infarction by artificial intelligence with 12-lead electrocardiogram.” In: *International Journal of Cardiology* 317 (2020), pp. 223–230 (cit. on p. VI-2).

**Data availability** The data that support the findings of this study are available from the Swedish Board of Health and Welfare and the included healthcare regions, but restrictions apply to the availability of these data, which were used under license for the current study, and so are not publicly available. Data are however available from corresponding author (J.S.) upon reasonable request and with permission of the Swedish Board of Health and Welfare and the included healthcare regions. All code together with the parameter/hyperparameter estimates of the presented models is available upon request.

**Acknowledgments** We thank David Widmann for the discussions and his input towards analyzing and improving the calibration of our model.

**Author contributions** All authors participated in the study design, interpretation of the data, and critically reviewing the paper. S.G., D.G., E.L., A.H.R., and J.S. wrote the first draft, and M.J.H., and T.B.S. contributed to the writing of subsequent versions. Data was acquired and prepared for analysis by M.J.H., S.G., and A.H.R. Statistical analyses and preparation of tables and fig-

ures was performed by S.G., D.G., E.L., and A.H.R. with support from J.S. and T.B.S. M.J.H., J.S., T.B.S. were responsible for funding acquisition, project administration, and supervision. All authors had access to and could verify the underlying data.

**Funding** Open access funding provided by Uppsala University. The study was funded by The Kjell and Märta Beijer Foundation, Anders Wiklöf, the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation, and Uppsala University. This project has received funding from the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme (grant agreement n° 101054643). The computations were enabled by resources in project sens2020005 and sens2020598 provided by the Swedish National Infrastructure for Computing (SNIC) at UPPMAX, partially funded by the Swedish Research Council through grant agreement no. 2018-05973. The funders had no role in in study design; collection, analysis, and interpretation of data; writing of the report; or decision to submit the paper for publication.

**Competing interests** J.S. reports stock ownership in companies providing services to Itrim, Amgen, Janssen, Novo Nordisk, Pfizer, Takeda, AstraZeneca and Vifor Pharma, outside the submitted work. S.G. is employed at Sence Research AB. The other authors report no conflicts of interest.

## Supplementary material

### A Supplementary methods

#### A.1 Data sources

Adult patients ( $\geq 18$  years old) with available emergency department data from 6 emergency departments in the Stockholm region, Sweden, between 2003 and 2017 were collected. The sample was linked to national registries (the patient [inpatient and specialized outpatient], prescribed drug, and death registries), national quality registries (SWEDEHEART [Swedish Web-system for Enhancement and Development of Evidence-based care in Heart disease Evaluated According to Recommended Therapies; a Swedish nationwide quality register] sub-registries RIKS-HIA [Register of Information and Knowledge About Swedish Heart Intensive Care Admissions] and SCAAR [Swedish Coronary Angiography and Angioplasty Registry]), as well as a regional database of ECGs (Karolinska ECG database) and electronic health records as summarized in Supplementary Table 3. All data sources covered the time-period 2007-2016 or longer. Characteristics have been described for STEMI [Szu+17] and NSTEMI [Szu+18] patients in SWEDEHEART during the present study period, and the study sample has been partially described previously [Ugg+20]. All ECG recordings were available as exported XML files in the GE MUSE (v9) XML format, with each lead represented by a numerical signal vector together with meta-data of the recording.

The outcome label used, NSTEMI/STEMI/control, is the standard INFARCT-TYPE variable in SWEDEHEART RIKS-HIA [SWE], complemented by diagnoses from the Swedish in-patient and cause-of-death registries to confirm a myocardial infarction (I21) for cases, or absence of a myocardial infarction for controls.

The SWEDEHEART variable captures the view of the whole cycle of care by the attending cardiologist at time of discharge from the coronary care unit, who has access to all relevant patient data, including but not limited to single ECGs, continuous ECG monitoring, cardiac enzyme series and other lab data, and angiographic and echocardiographic results. Regular monitoring of the SWEDEHEART registry shows a data accuracy of around 96% [Jer+10].

#### A.2 Label noise

We ran a risk of label noise since the label (control/STEMI/NSTEMI) was determined at discharge from the hospital, when the whole care episode, which can include multiple ECGs, could be summarized. Hence, it is impossible for

us to identify which ECG determined the final diagnosis of the discharging physician. We try to include the correct ECGs by inclusion filters at-event before-treatment. However, in the end the ECG data of this study may not always be the ones guiding the final diagnosis. We mitigate that to some extent by using multiple ECGs, i.e. the repeated recordings if available within one day before or on admission in the training set, but not in the validation and test sets.

We evaluated the label noise by checking for possible/definite ST-elevation in ECGs with the original label STEMI or for the absence of definite ST-elevation in the ECGs with the label NSTEMI or control. All those ECGs were manually evaluated by a senior cardiologist (JS) who was blinded to the original labels. Given the large number of control ECGs, JS only manually evaluated 100 control ECGs selected at random and found no ECG with a possible/probable ST-elevation. All ECGs with the label STEMI/NSTEMI in the test set where checked and we found a fair amount of label noise. This does not necessarily imply that the data is wrongly labelled but that the ECG that we use based on our inclusion filters does not clearly show the characteristics for the label whereas another exam in the same hospitalization may show the characteristics.

To have a clear test set for which we report the model evaluation metrics, ECGs with a likely incorrect original label were removed. The manually evaluated ECGs were labelled as "definitely STEMI (0)", "likely STEMI (1)", "not STEMI (2)". If the original label was STEMI and our review's label was (2) we removed the ECG from the test set. Similarly, if the original label was NSTEMI and our review's label was (0), we removed the ECG. In this way we removed 117 (28%) STEMI and 66 (6%) NSTEMI ECGs from the test data set.

### A.3 Model architecture

Our model architecture is an extension of a previous study [Rib+20]. We extended the architecture with Squeeze and Excite (SE) blocks [HSS18] within each residual block. Additionally, we include the phenotypes age and sex in our model. Our final model has the following hyperparameter. We use 12 residual blocks with filter sizes {64;64;64;128;128;128;256;256;256;512;512;512} and down sampling factors {2;1;2;2;1;2;2;1;2;2;1;2} where all SE layers have a reduction factor of 8. The convolutional layers have a kernel size of 17. Age is normalized in all data to a zero-mean unit variance variable using the mean age and standard deviation from the training set and concatenated with sex as binary variable. The combined age and sex is then encoded in a linear layer with 64 output units before it gets concatenated with the flattened ResNet output. In the end we create an ensemble of 5 models which are independently trained from scratch.

## A.4 Model training

The model was trained by minimizing the cross-entropy loss using the Adam optimizer with default parameters and learning rate of 0.0005 for 100 epochs using an effective batch size of 1024 distributed over two GPUs. We used a cosine learning rate scheduler which reduces the learning rate according to a cosine function from the initial learning rate to a final value of  $10^{-6}$  over the epochs. Initially, we warmed up the learning rate linearly over 10 epochs. For regularization we use dropout with dropout probability of 0.5 within the ResNet blocks. Furthermore, we regularized with weight decay of  $10^{-3}$ . We tried label smoothing for additional regularization during the hyperparameter search but found an optimal value of 0.00.

## A.5 Model and training improvements

We extended the training procedure with multiple training and architecture options following previous recommendations of modern ResNet tuning to improve model performance [Bel+21; He+16]. We tested to include these options iteratively on a subset of the training data set with high-risk patients who were admitted to a coronary care unit, with increased number of training epochs up to 200 using 5-fold cross validation.

1. Training data set augmentation: include the repeated recordings in the train data set.
2. Learning rate scheduler: move from an initial multistep learning rate scheduler with fixed decrease by factor 10 at epoch 75, 125, 175 to a cosine learning rate scheduler with linear warmup [Liu+19].
3. Label smoothing.
4. Additional SE net layers.
5. Additional age and sex embedding.
6. Additional ensemble-based model with heuristically chosen five ensemble members.

Our observation was that each of the extensions improves our evaluation metrics. Therefore, our model for the hyperparameter search contains all of the extensions. Furthermore, we experimented with the SGD optimizer instead of ADAM without and with momentum of {0.7, 0.8, 0.9} and with increase of the network width by factors of {1, 1.5, 2}. However, these results did not improve our performance.

## A.6 Hyperparameter Search

For our random hyperparameter search we considered the following model hyperparameter and respective values: number of residual blocks {4, 8 ,12} with respective filter sizes and down sample factors, reduction factor of the SE-layer {4, 8, 16}. For the training hyperparameters we considered the following: batch size per GPU 256, 512, 1024, learning rate {0.05, 0.01, 0.005, 0.001, 0.0005}, weight decay { $10^{-3}$ ,  $10^{-4}$ , 0}, loss smoothing {0, 0.05, 0.1, 0.15, 0.2}.

We ran our model training on 2 Nvidia A100-PCIE-40Gb GPUs. We utilize distributed data parallel training with multiple parallel spawned processes. This effectively doubles our batch size and therefore increases training speed significantly. In this setup training of one model—not five models for the ensemble—takes around 4 hours for 100 epochs.

## A.7 Model calibration

We tried to improve upon the original calibration by temperature scaling [Guo+17], vector calibration [Guo+17], and Dirichlet calibration [Kul+19]. None of those methods succeeded in improving the model calibration. Further work is necessary to investigate the possibility to improve model calibration beyond our current setting.

## A.8 External validation set

The PTB-XL is a publicly available database of 21,837 10-second 12-lead ECGs annotated with 71 different ECG statements, including cases of myocardial infarction [Wag+20; Gol+00].

- 98 ECGs annotated as likely acute myocardial infarctions without electrode problems by at least one cardiologist where selected based on the following PTB-XL annotations (`infarction_stadium1 == "Stadium I" OR infarction_stadium2 == "Stadium I"`) AND `validated_by_human == TRUE` AND `isna(electrodes_problem)`.
- 200 ECGs annotated as likely controls without myocardial infarction and without electrode problems by at least one cardiologist where selected based on the following PTB-XL annotations (`isna(infarction_stadium1)` AND `isna(infarction_stadium2)`) AND `validated_by_human == TRUE` AND `isna(electrodes_problem)` AND `NORM == 100`.

## Paper VI – Development and validation of deep learning ECG-based prediction of myocardial infarction in emergency department patients

These ECGs were manually reviewed by a senior cardiologist (JS). All controls were confirmed and 75 myocardial infarction ECGs had a probable or definite ST-elevation in line with the existing PTB-XL annotation, which were included in the PTB-XL test set. The leads used in training were extracted and normalized as described in Figure 3. Age and sex was extracted from the PTB-XL database and age was normalized using the mean and standard deviation from the training dataset.

### A.9 Bootstrapped data uncertainty

In Table 2 we report model uncertainty from different initialization of our convolutional network. In Supplementary Figure 11 we additionally report data uncertainty based on bootstrapped data from the test data sets. For the bootstrap we chose the model with the median micro average precision on the validation data. Since we have 10 models, we cannot chose the median but chose the 6th best model in terms of micro average precision. Then we make 1000 samplings with replacement from each respective test data set and report the mean and 95% confidence interval (percentile bootstrap) for all metrics.

## B Supplementary tables

**Table 3:** Data sources used to define the study sample with patient records linked on the Swedish personal identifier number.

Database	Coverage	Type of data
Emergency department discharge records	Regional	Presented complaint
Electronic health records	Regional	ECGs, laboratory measurements
SWEDEHEART Riks-HIA	National	CCU admissions with STEMI/NSTEMI labels
SWEDEHEART SCAAR	National	Coronary interventions
National Patient Registry	National	Diagnoses, surgical interventions
Cause of death registry	National	Causes of death
Swedish prescribed drug registry	National	Dispensed drugs

**Table 4:** Definitions used for diagnoses, surgery, and pharmaceutical treatment.

Diagnosis/intervention/treatment	Codes
<i>Diagnosis (ICD10)</i>	
Myocardial infarction	I21
Unstable angina	I20.0
Ischemic heart disease	I20-I25
Left bundle branch block	I44.(6 7)
Stroke	I60-I64
Peripheral artery disease	I70-I74,I77.(3 6 8),I79
Heart failure	I50
Atrial fibrillation	I48
Cardiovascular disease	I
<i>Surgical codes (KVA)</i>	
PCI/CABG	FNG(00 02 05 10 96),FNC,FND,FNE
<i>Treatment (ATC)</i>	
Renin-angiotensin system inhibitors	C09
Calcium channel blockers	C08
Beta-receptor blockers	C07
Mineralocorticoid receptor antagonists	C03DA
Diuretics	C03
Anti-arrhythmic drugs	C01B
Statins	C10AA
Anticoagulants	B01A(A E F)
Antiplatelets	B01AC

**Table 5:** Clinical characteristics of the study sample, stratified by control/NSTEMI/STEMI and training/test set.

Patient characteristics of the study sample, by control/NSTEMI/STEMI status and by training and test sets. Data presented as median (interquartile range) or percentages.

\*Prevalent disease based on any diagnosis position, inpatient and outpatient specialist care combined.

\*\*Combining troponin and high-sensitive troponin laboratory measurements from regional laboratory databases and the SWEDHEART database. Maximum of all available measurements within the time window is reported separately for troponin I and T.

\*\*\*Primary diagnosis from inpatient specialist care and/or specialized outpatient care at time of the ED/CCU visit.

ED, emergency department; NTproBNP, N-terminal pro-B-type natriuretic peptide.

	Random test sets			Temporal test sets			Training sets		
	Control	NSTEMI	STEMI	Control	NSTEMI	STEMI	Control	NSTEMI	STEMI
Number of patients	88,742	820	193	27,561	263	108	368,689	4,333	1,517
<i>Clinical characteristics at ED visit</i>									
Age	64.0 (47.0,77.0)	71.0 (61.0,80.2)	66.0 (60.0,77.0)	54.0 (37.0,71.0)	70.0 (59.5,79.5)	64.0 (55.0,71.0)	65.0 (48.0,78.0)	72.0 (62.0,81.0)	67.0 (57.0,78.0)
Male	47.2	67.8	75.1	48.1	66.5	80.6	47.2	64.9	73.0
Year	2012 (2010,2014)	2013 (2011,2015)	2013 (2011,2014)	2016 (2011,2016)	2016 (2011,2016)	2016 (2011,2016)	2012 (2010,2014)	2013 (2011,2014)	2013 (2011,2014)
<i>Presenting complaint</i>									
Chest pain	21.5	71.5	67.4 7.3	22.3 12.5	74.1 10.6	80.6 3.7	21.4 14.8	71.1 12.2	69.7 5.9
Difficulty breathing	14.1	13.4							

Continued on next page

	Random test sets			Temporal test sets			Training sets	
	Dizziness	Heart problems	Circulatory arrest					
Myo-cardial infarction	8.5	24.5	12.4	3.3	14.4	9.3	8.8	27.0
Unstable angina	3.9	11.5	3.1	1.3	5.3	1.9	4.5	11.5
Ischemic heart disease	20.5	40.4	19.2	7.7	27.0	14.8	21.0	44.7
Stroke	9.3	11.5	5.2	3.8	3.0	3.7	9.4	11.6
Peripheral artery disease	7.0	12.2	5.7	3.6	5.7	7.4	7.2	12.8
Heart failure	15.6	20.4	5.7	4.8	9.9	2.8	16.7	21.3
Atrial fibrillation	19.0	16.3	5.2	7.7	9.9	0.9	21.1	16.1
Cardio-vascular disease	58.7	69.6	43.5	37.5	54.0	41.7	60.6	72.0
<i>Drugs with &gt;=1 dispensation within one year prior to ED visit</i>								
Renin-angio-tensin system inhibitors	32.7	51.8	33.7	23.9	51.0	30.6	33.7	51.8
Calcium channel blockers	17.4	28.7	23.3	13.0	27.0	13.9	18.3	29.5
Beta-receptor blockers	36.0	52.7	31.6	20.1	36.5	20.4	37.5	51.1
Mineralocorticoid receptor antagonists	6.3	7.7	4.1	2.1	3.4	0.9	6.6	6.0
Diuretics	27.4	36.1	21.2	12.3	23.6	5.6	28.8	34.5
Anti-arrhythmic drugs	1.6	0.5	0.0	0.4	0.4	0.0	1.8	0.4
Statins	23.9	42.2	20.2	15.4	29.3	24.1	24.2	42.6
Anticoagulants	12.4	10.6	1.0	6.8	4.9	1.9	13.3	8.7
Antiplatelets	27.9	49.0	26.9	13.4	30.8	16.7	28.9	49.8

Continued on next page

	Random test sets			Temporal test sets			Training sets		
<i>Cardiac enzymes within ED visit or coronary care unit hospitalization **</i>									
Troponin I measured	1.0	8.2	14.0	0.0	0.0	0.0	1.2	8.7	14.7
Max troponin I (ng/L)	29.7 (29.7, 40.0)	2200.0 (550, 6450)	21000.0 (5100, 48850)	-	-	-	29.7 (29.7, 40.0)	2950.0 (630, 10500)	18100.0 (2900, 45000)
Troponin T measured	36.4	86.3	86.0	37.5	98.9	100.0	38.7 (5.0, 20.0)	86.7 (96.0, 830.0)	86.9 (474.5, 4500)
Max troponin T (ng/L)	9.9 (5.0, 16.0)	228.5 (80.8, 777.8)	2315.0 (684, 5720)	5.0 (5.0, 12.0)	231.5 (103.5, 740.8)	2910.0 (1205, 6275)	9.9 (5.0, 20.0)	271.0 (96.0, 830.0)	1790.0 (474.5, 4500)
NTproBNP measured	7.8	20.1	12.4	6.9	24.0	26.9	9.1 (303, 4410)	21.6 (802, 9150)	17.0 (900.5, 9300)
Max NTproBNP (ng/L)	1230.0 (269, 4187.5)	2910.0 (965, 7770)	4150.0 (3492, 7630)	746.0 (130, 1935)	1180.0 (348.5, 6130)	1670.0 (610, 4410)	1400.0 (3020.0)	3020.0 (303, 4410)	3300.0 (802, 9150)
<i>Main cause of coronary care unit hospitalization ***</i>									
Myocardial infarction	0.0	93.4	95.9	0.0	96.6	98.1	0.0	94.0	97.4
Unstable angina	0.2	4.1	0.5	0.2	10.6	0.0	0.2	3.7	1.8
Ischemic heart disease	1.2	93.8	95.9	0.6	96.6	98.1	1.4	94.6	97.4
Stroke	2.2	0.6	0.5	1.9	0.8	0.9	2.0	1.2	0.7
Peripheral artery disease	0.4	0.2	0.0	0.4	0.4	0.0	0.5	0.3	0.3
Heart failure	3.1	2.2	1.0	1.4	3.0	0.9	3.5	2.1	0.9
Atrial fibrillation	4.9	1.2	0.0	3.0	0.4	0.0	6.4	0.8	0.4
Cardiovascular disease	16.8	95.5	99.0	12.1	97.7	100.0	19.3	96.3	98.2
<i>Mortality after coronary care unit admission</i>									
30-day all-cause death	3.3	6.1	13.5	2.5	4.9	7.4	3.6	6.5	10.5
In-hospital all-cause death	2.4	5.2	12.4	1.9	3.8	5.6	2.7	5.6	9.0

**Table 6:** Over-/underrepresented diagnoses among misclassified cases.

Over-/underrepresented diagnoses when comparing correct classifications with misclassifications for a given observed class, restricted to ECGs with a high predicted probability ( $\text{Pr} > 0.5$ ). Tested in an asymptotic general independence test with a two-sided alternative hypothesis[Hot+08]. All diagnoses at time of the visit (coronary care unit or emergency department only) are included (all diagnosis positions). All results with a false discovery rate (Benjamini-Hochberg)  $< 0.01$  are reported.

<b>ICD10 code</b>	<b>Odds ratio</b>	<b>Z-score</b>	<b>Observed class</b>	<b>Wrong predicted class</b>
K04	884.9	27.5	Control	NSTEMI
K27	601.6	22.7	Control	NSTEMI
J95	417.9	18.9	Control	NSTEMI
I05	301.3	16.2	Control	NSTEMI
I33	209.6	13.5	Control	NSTEMI
I40	195.5	27.8	Control	STEMI
I07	181.7	12.9	Control	STEMI
I51	126.0	15.0	Control	STEMI
J81	109.9	9.8	Control	NSTEMI
Q20	78.8	8.2	Control	NSTEMI
K26	59.9	7.2	Control	NSTEMI
I34	54.9	6.8	Control	NSTEMI
K25	44.7	6.2	Control	NSTEMI
I46	43.2	10.5	Control	STEMI
B95	37.6	5.6	Control	NSTEMI
K62	33.1	5.5	Control	STEMI
I71	31.8	5.1	Control	NSTEMI
I42	29.5	8.6	Control	STEMI
I47	26.3	4.6	Control	NSTEMI
I30	24.3	4.6	Control	STEMI
K92	23.9	4.4	Control	NSTEMI
E10	23.3	4.4	Control	NSTEMI
A04	21.9	4.4	Control	STEMI
A49	21.1	4.3	Control	STEMI
Z72	19.1	5.6	Control	STEMI
N18	12.9	4.1	Control	NSTEMI
I20	11.5	3.9	Control	NSTEMI
Z86	10.0	4.7	Control	STEMI
T81	0.1	-4.0	NSTEMI	Control
N17	0.1	-3.9	NSTEMI	Control
J09	0.02	-4.7	NSTEMI	Control
N13	0.02	-4.7	NSTEMI	Control

**Table 7:** Performance of the model showing bootstrapped data uncertainty

		<b>Random</b>	<b>Temporal</b>	<b>PTB-XL</b>
C-statistic ( $\uparrow$ )	Control	0.867 (0.854-0.880)	0.902 (0.884-0.921)	0.958 (0.920-0.984)
	STEMI	0.994 (0.990-0.997)	0.987 (0.963-0.998)	0.959 (0.921-0.985)
	NSTEMI	0.837 (0.823-0.852)	0.866 (0.842-0.889)	-
	MI	0.867 (0.854-0.880)	0.902 (0.884-0.921)	-
AP ( $\uparrow$ )	Control	0.998 (0.998-0.998)	0.998 (0.997-0.999)	0.967 (0.920-0.994)
	STEMI	0.714 (0.653-0.770)	0.757 (0.674-0.834)	0.939 (0.898-0.971)
	NSTEMI	0.160 (0.133-0.189)	0.191 (0.147-0.240)	-
	MI	0.343 (0.314-0.372)	0.483 (0.428-0.534)	-
Brier ( $\downarrow$ )	Control	0.009 (0.008-0.009)	0.009 (0.008-0.010)	0.150 (0.114-0.189)
	STEMI	0.001 (0.001-0.001)	0.002 (0.001-0.002)	0.189 (0.147-0.232)
	NSTEMI	0.008 (0.008-0.009)	0.008 (0.007-0.010)	-
	Multiclass	0.018 (0.017-0.019)	0.020 (0.017-0.022)	-
ECE ( $\downarrow$ )	Multiclass	0.417 (0.416-0.418)	0.416 (0.415-0.417)	0.281 (0.240-0.324)

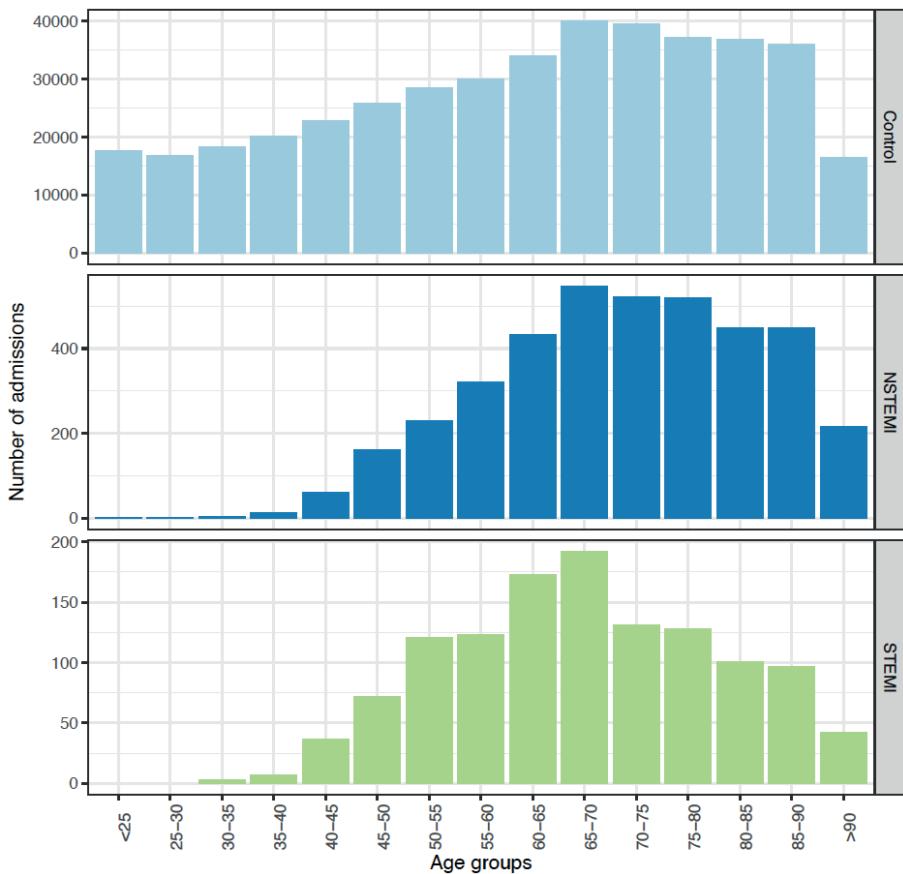
Results of the model in the two test sets of the study sample and the publicly available PTB-XL dataset as comparison in the rightmost column. We show the mean and 95% confidence interval from a bootstrap to highlight data uncertainty which is in contrast to Table 2 where we show model uncertainty using different seeds.

**Table 8:** Number and proportion of records in the random and temporal test set restricted to the strata shown in Supplementary Figure 12

<b>Subset</b>	<b>Random</b>	<b>Temporal</b>
No ST-filter	88,742 (98.7);273 (0.3);870 (1.0)	27,561 (98.5);145 (0.5);279 (1.0)
Moderate ST-filter*	88,742 (98.9);193 (0.2);820 (0.9)	27,561 (98.7);108 (0.4);263 (0.9)
Strict ST-filter	88,742 (99.1);152 (0.2);698 (0.8)	27,561 (98.9);82 (0.3);237 (0.9)
Age <54	29,706 (99.6);23 (0.1);87 (0.3)	13,420 (99.6);25 (0.2);33 (0.2)
Age [54,73)	30,399 (98.4);106 (0.3);382 (1.2)	8,553 (97.8);61 (0.7);127 (1.5)
Age >73	28,637 (98.6);64 (0.2);351 (1.2)	5,588 (97.8);22 (0.4);103 (1.8)
Male	41,913 (98.4);145 (0.3);556 (1.3)	13,261 (98.1);87 (0.6);175 (1.3)
Female	46,829 (99.3);48 (0.1);264 (0.6)	14,300 (99.2);21 (0.1);88 (0.6)
ECG same day	86,171 (99.0);182 (0.2);674 (0.8)	26,800 (98.8);105 (0.4);228 (0.8)
ECG 1d apart	2,571 (94.2);11 (0.4);146 (5.4)	761 (95.2);3 (0.4);35 (4.4)
Karolinska ED	74,061 (99.0);141 (0.2);617 (0.8)	19,701 (98.9);61 (0.3);153 (0.8)
Not Karolinska ED	14,681 (98.3);52 (0.3);203 (1.4)	7,860 (98.0);47 (0.6);110 (1.4)
CCU only	1,551 (60.5);193 (7.5);820 (32.0)	170 (31.4);108 (20.0);263 (48.6)
MAC55	69,916 (98.8);159 (0.2);690 (1.0)	22,848 (98.6);92 (0.4);229 (1.0)
Not MAC55	18,826 (99.1);34 (0.2);130 (0.7)	4,713 (99.0);16 (0.3);34 (0.7)
v237	65,021 (99.0);138 (0.2);529 (0.8)	14,530 (98.7);63 (0.4);128 (0.9)
Not v237	23,721 (98.6);55 (0.2);291 (1.2)	13,031 (98.6);45 (0.3);135 (1.0)

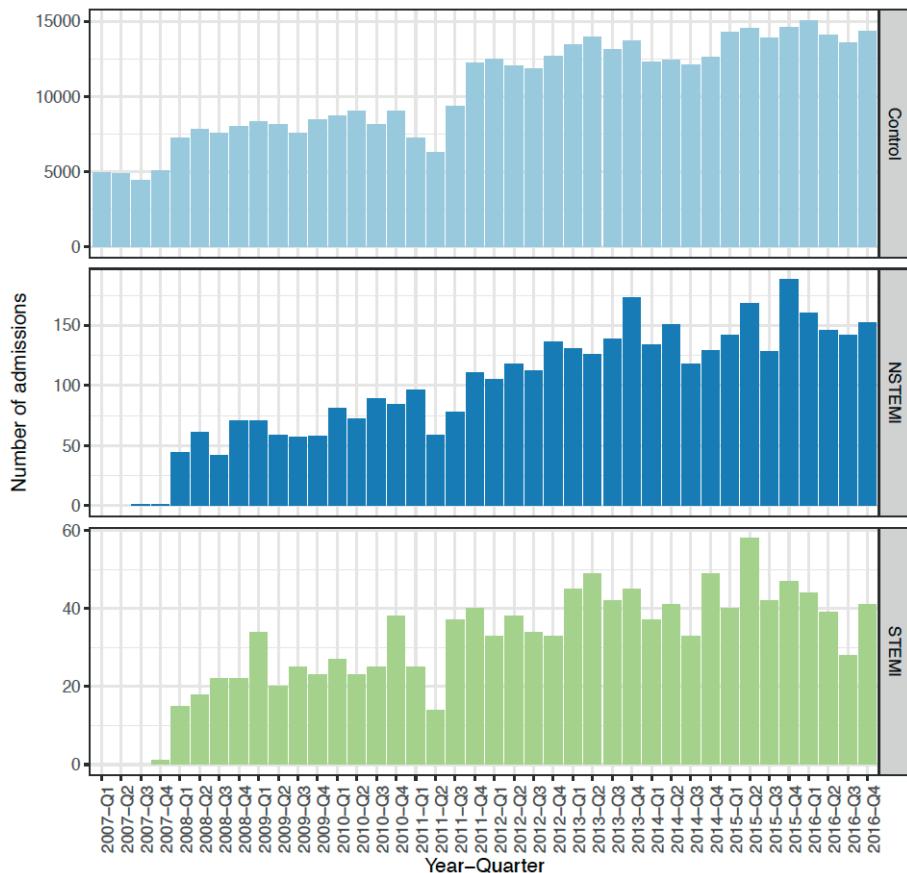
The numbers are presented in the order controls, STEMI, NSTEMI.

## C Supplementary figures



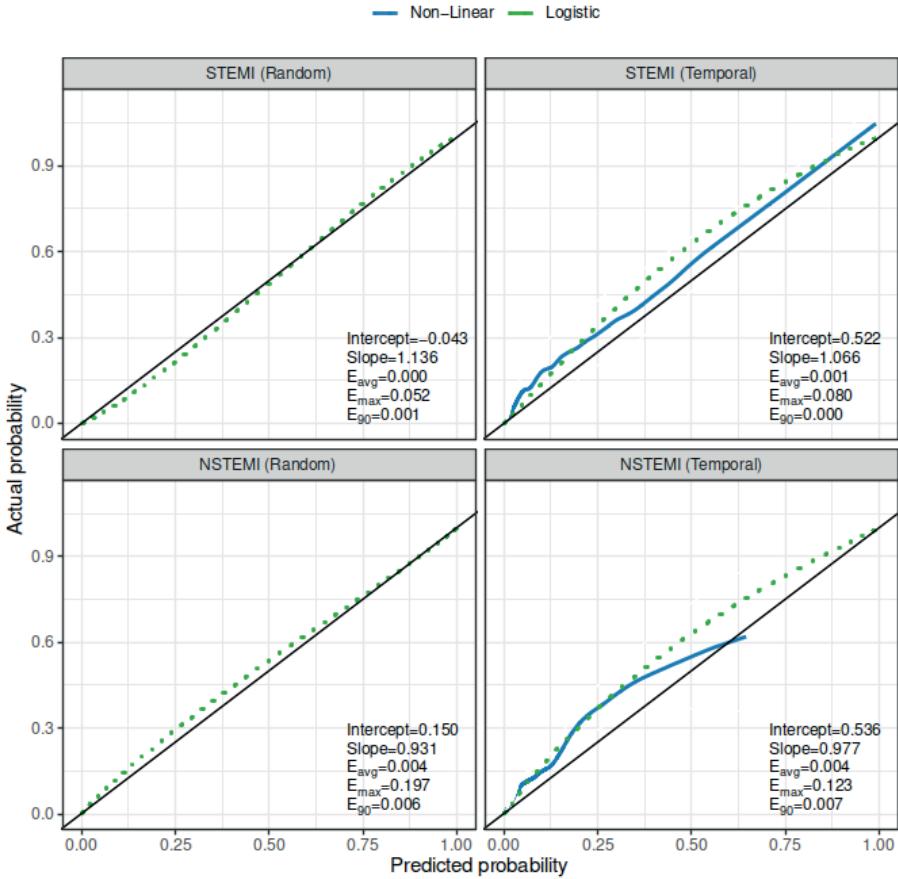
**Figure 8:** Distributions of age at time of the ECG recording, by control/NSTEMI/STEMI status.

Note the different y-scale in different panels.



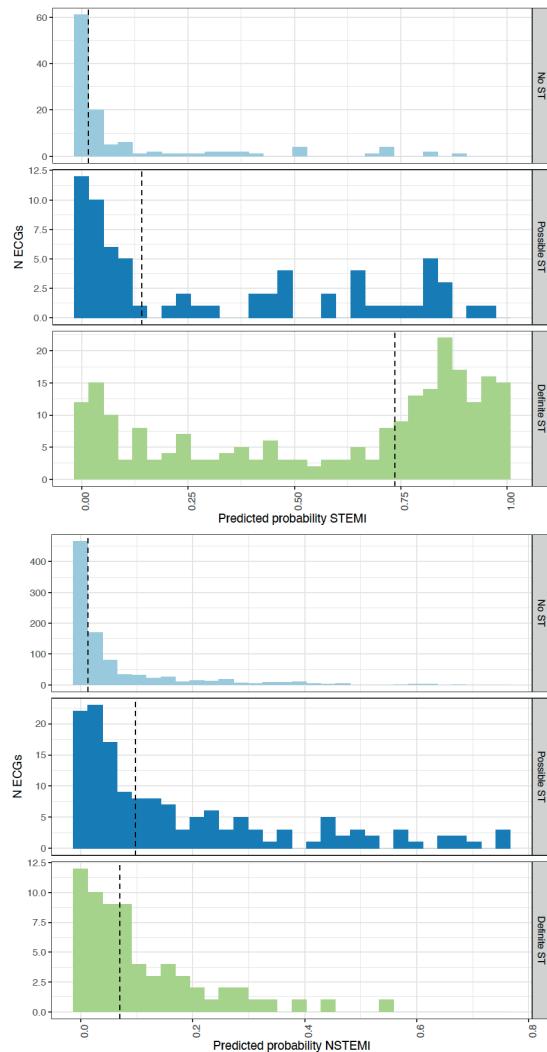
**Figure 9:** Distributions of ECG date (year and quarter bins), by control/NSTEMI/STEMI status.

Note the different y-scale in different panels.

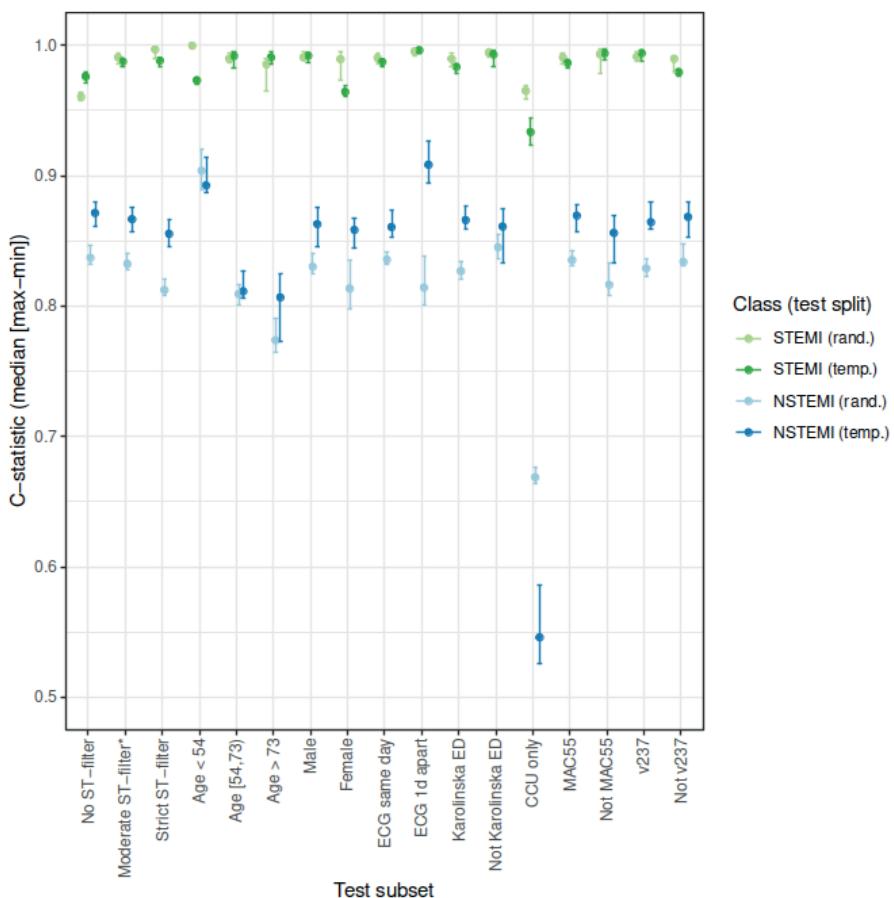


**Figure 10:** Calibration of the final model.

Calibration plot for NSTEMI vs all and STEMI vs all for both of our test sets (random and temporal). Blue solid lines are LOESS (local regression) smooths of the estimated probabilities versus the class membership accompanied by a logistic calibration curve in green (dotted) resulting from regressing the estimated logits on the class membership. The light-blue shaded areas are 95% bootstrap confidence intervals from a bootstrap with 2,000 replicates. The black solid lines indicate ideal calibration. Intercept denotes the intercept from the logistic models and should be as close to zero as possible. Slope denotes the slope of the logistic regression fit and should be as close to 1 as possible. A slope  $>1$  indicates underfitting and a slope  $<1$  indicates overfitting with the degree of under-/overfitting directly proportional to the absolute size of the slope.  $E_{avg}$ ,  $E_{90}$  and  $E_{max}$  correspond to the average absolute error, the 90th percentile of the absolute error and the maximum absolute error between the predicted probabilities and the LOESS fit. We tried to improve upon the original calibration by temperature scaling [Guo+17], vector calibration [Guo+17], and Dirichlet calibration [Kul+19]. None of those methods succeeded in improving the model calibration.

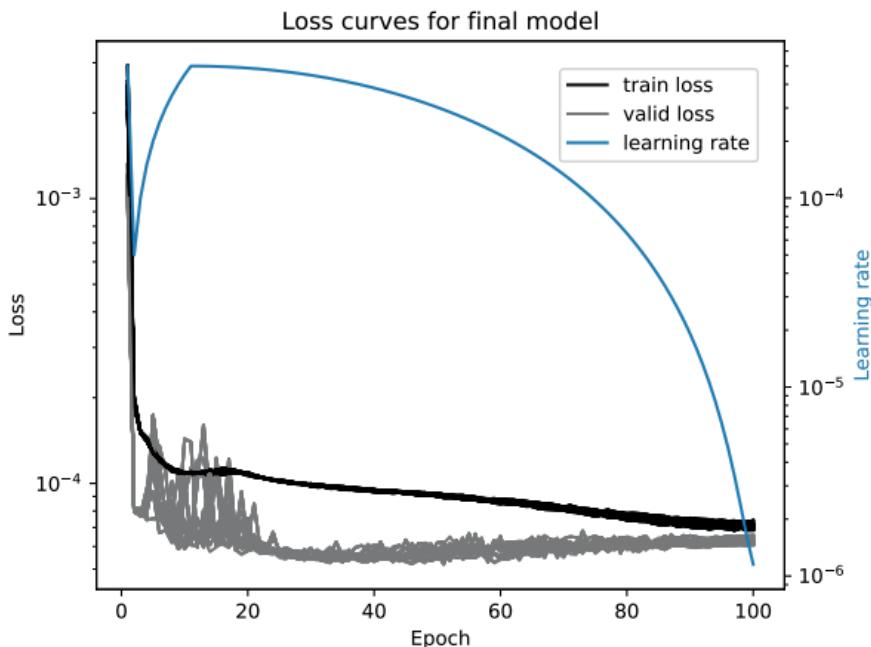


**Figure 11:** Predicted probabilities of STEMI for ECGs labeled as STEMI (top three panels), or for NSTEMI for ECGs labeled as NSTEMI (bottom three panels). A manual evaluation of the presence of a ST-elevation was performed for the ECGs in the test set as described in the methods (no/possible/definite ST-elevation). The results are stratified by these groups and the median predicted probability of each group is represented by a vertical dashed line.



**Figure 12:** C-statistics of STEMI vs rest (green) as well as NSTEMI vs rest (blue) in the temporal and random test sets, in subsets of patients.

The x-axis classes are given in the order: test set records without any filter applied based on ST-elevation label noise; ST-elevation filter corresponding to results in main table (used in all following test set subsets); stricter ST-elevation where “possible STEMI” is removed from all classes; age tertiles; sex; was the ECG collected at the same day as the admission or not ; did the patient visit the emergency department at Karolinska Hospital (main source of data) or another emergency department in the Stockholm region, patients attending the CCU only, ECGs recorded using the most common machine type (MAC55) or not, ECGs recorded using the most common software (v237) or not. Median (min-max) is presented from the model initiated with 10 random seeds.



**Figure 13:** Loss and learning rate over epochs.

The validation loss in grey and training loss in black for all models, i.e. all ensemble members. Note that the validation loss is lower here due to the use of weight decay regularization and dropout during training but not during validation which is standard practice. We select the model with the lowest validation loss. The learning rate is a cosine annealing learning rate scheduler with linear warm up for the first 10 epochs. The y-axis in log mode distorts the cosine annealing function but is better suited to see the actual learning rate value at any given epoch.



