Thesis For The Degree Of Doctor Of Philosophy

Geometry and Symmetry in Deep Learning

From Mathematical Foundations to Vision Applications

Oscar Carlsson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Division of Algebra and Geometry
Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2025

Geometry and Symmetry in Deep Learning: From Mathematical Foundations to Vision Applications
Oscar Carlsson
Gothenburg 2025
ISBN 978-91-8103-249-9

Acknowledgements, dedications, and similar personal statements in this thesis, reflect the author's own views.

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Doktorshavhandlingar vid Chalmers tekniska högskola
Ny serie nr 5707
ISSN 0346-718X

Division of Algebra and Geometry
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Telephone +46(0)31-772 10 00

Typeset with LaTeX
Printed by Chalmers Digitaltryck
Gothenburg, Sweden 2025

Geometry and Symmetry in Deep Learning

From Mathematical Foundations to Vision Applications

Oscar Carlsson

Division of Algebra and Geometry
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Deep learning — particularly neural networks — has profoundly transformed both industry and academia. However, designing and training effective networks remains challenging, often requiring extensive data and compute. This barrier limits applicability of deep learning in domains with scarce labelled data or limited computational budgets. One way to overcome these barriers is to bake prior knowledge — such as geometry or symmetry — directly into network architectures.

Geometric deep learning focuses on model and data design by leveraging the knowledge of problem specific geometry and symmetries. Encoding this into the pipeline can reduce sample complexity as the models do not need to learn these structures directly from the data. Two common examples of this is *equivariant* and *invariant* networks. Equivariant networks guarantee that when the input is transformed the output transforms in a predictable way. On the other hand, an invariant network is a network where the output does not change if the input is transformed.

In this thesis we study both applied and mathematical perspectives on parts of the geometric deep learning field. On the mathematical side I show a theory for equivariant CNNs on (bi)principal bundles and a novel framework for equivariant non-linear maps. On the applied side the I present a study of the effects of imposed equivariance on the data requirements and the increased data efficiency as well as the benefits of using a grid well suited for the underlying geometry of the data.

Keywords: geometric deep learning, induced representations, group theory, differential geometry, equivariant networks, gauge theory

Appended papers

This thesis is based on the following papers:

Paper I. Jan E. Gerken, Jimmy Aronsson*, **Oscar Carlsson***, Hampus Linander, Fredrik Ohlsson, Christoffer Petersson, and Daniel Persson "Geometric deep learning and equivariant neural networks". In: *Artificial Intelligence Review* (June 2023)

Paper II. Jan Gerken, **Oscar Carlsson**, Hampus Linander, Fredrik Ohlsson, Christoffer Petersson, and Daniel Persson "Equivariance versus Augmentation for Spherical Images". In: *Proceedings of the 39th International Conference on Machine Learning* (June 2022), pp. 7404-7421

Paper III. **Oscar Carlsson***, Jan E. Gerken*, Hampus Linander, Heiner Spieß, Fredrik Ohlsson, Christoffer Petersson, and Daniel Persson Daniel "HEALSWIN: A Vision Transformer on the Sphere". In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2024), pp. 6067-6077

Paper IV. Elias Nyholm*, **Oscar Carlsson***, Maurice Weiler, and Daniel Persson "Equivariant non-linear maps for neural networks on homogeneous spaces". *Submitted* (April 2025)

*: Equal contribution

Author contributions

In all projects O.C. was an active and contributing part to all discussions, conceptualisation, method development, analysis, and the writing of the papers. Below is a list of the easily identifiable contributions:

Paper I O.C. was the author of chapter 2, except section 2.6;

Paper II O.C. wrote some integral parts of the code base, e.g., the data loading pipeline. Additionally O.C. finalised the figures;

Paper III O.C. designed and performed all experiments relating to depth estimation and omnidirectional segmentation, analysed the results, and wrote all related text;

Paper IV O.C. wrote chapter 4 and the appendix.

Acknowledgements

With this thesis I conclude my PhD studies and through the past five years I have met many amazing people.

To start I have been part of a rather large research group, at least for mathematics. Firstly my supervisor Daniel Persson, my assistant supervisor Robert Berman and Christoffer Petersson filling the role as industrial supervisor. Additionally the group consists of Hampus Linander, Jan Gerken, Fredrik Ohlsson and Max Guillen. Not to forget my PhD comrades in the group: Elias Nyholm, Philipp Misof, Emma Andersdotter Svensson, and Jimmy Aronsson — who graduated two years before me. Additionally, a recent pleasant addition to the collaborator list is Maurice Weiler. I want to thank all above for invaulable discussions and insights, both on active collaborations and otherwise. Additionally I would like to thank my examinor Håkan Samuelsson for his support.

During my five years at the Department of Mathematical Sciences I also met and made friends with many fellow PhD students, some of which, in no particular order, are Mattias Byléhn, Erik Karlsson Nordling, Carl-Joar Karlsson, Selma Tabakovic, Edvin Wedin, Tobias Magnusson, Stepan Maximov, Rahim Nkunzimana, Robin van Haastrecht, Jan Gundelach, Georg Huppertz, Olof Zetterqvist, Kasper Bågmark, Kristian Holm, Erik Jansson, and Björn Müller, to name a few. There are also people at the department who have withstood my, not infrequent, questions. I would like to thank Marie Kühn, Alia Särkkä, Marija Cvijovic, and Elisabeth Eriksson for their patience.

I have also had the pleasure of being a PhD student in the *Wallenberg AI, Autonomous Systems and Software Program (WASP)* and through the common courses and activities I have met many new and amazing people. Some friends I have made from the WASP sphere is, without any particular order, Karl Bengtsson Bernander, Georg Bökman, Lucas Brynte, Johan Edstedt, Rita Laezza, Yara Lochman, Pavlo Melnyk, and Gabriel Arslan Waltersson.

During the writing of this thesis I had the pleasure of having many people in my circles outside of Chalmers showing interest in providing feedback on the thesis. As such I would like to thank Ida Rynefors, Stina and Henrik Carlsson, Bengt Carlsson, Emil Kindblom, Brunte, Johan Thorsell, Jesper Medin, and Anders Adlemo. Additionally I would like to thank Max Carnesten for fruitful discussions.

I would also like to thank the two student associations F-spexet and Chalmers Barockensemble for enhancing my life with all the amazing people therein, and requiring me to focus on something else from time to time.

Finally, I would like to thank the rest of my friends along with my family for all your love and support.

Oscar Carlsson, 2025

Contents

Part I

Introduction and motivation

1 Introduction

This chapter serves as brief introduction and motivation to my thesis. Section 1.1 presents my purpose in writing this thesis and section 1.4 details the outline of the thesis. Section 1.2 presents a technical motivation of my work from two different points of view: for the point of view of a person focussed on machine learning and for the point of view of a person focussed on mathematics. Finally section 1.3 gives a brief summary on the result of the appended papers.

1.1 Purpose of thesis

My intended purpose of this thesis is to present my research (**Paper I**–**Paper IV**) as well as to elaborate on details and concepts that were omitted in the papers — e.g., due to conferences having a stric page limit, usually around 8–9 — or presented too compactly. I also aim to present how my work fits into the larger field and any directly applicable takeaways from the different papers.

1.2 Why should you read this thesis?

Here are three different motivations for why you should read this thesis no matter if your background is in machine learning (section 1.2.1), mathematics (section 1.2.2), or neither (section 1.2.3).

1.2.1 Motivation from a machine learner's point of view

Geometry and symmetries can enter a machine learning problem in many different ways. Even if you have not thought about it in terms of geometry and inductive bias there is generally a trade-off between model flexibility and data efficiency. Briefly stated a stronger geometric biases yields higher sample efficiency but less architectural flexibility.

As an example compare a convolutional neural network (CNN) [47, 46, 83] to the vision transformer (ViT) [36]. One fundamental assumption — inductive bias — leading to the CNN architecture is that features, or patters, can appear at any position in the image with similar probability. Due to this assumption a CNN uses shared local filters across the entire image.The geometrical inductive bias forces the network to learn local pattern improving the data efficiency as compared to an non-biased network having no concept of locality and having to learn that patterns can appear in any position. This makes a CNN relatively data efficient.

The ViT on the other hand splits the input image into patches and merge these into tokens and then computes attention weights between each pair of tokens. This means that each layer of the ViT can create relations or patterns over the entire image and there is no inherent bias towards local patterns. Hence the ViT is more flexible as it lacks the locality bias. However, the downside of this flexibility is that more data is needed for the model to learn the relevant features [36]. With that said it is possible to reintroduce inductive biases through, e.g., the use of relative positional encoding [139, 114] that can guide the layers to attend to more local patterns while not prohibiting long range interaction.

For applications where data is limited, or generally expensive or complicated to obtain, one can use these types of inductive biases to increase data efficiency [17]. To achieve a well working model these assumptions must of course be reasonable considering the task and type of data.

Geometric inductive biases does not only concern the scale of patters — local vs global — but also their location. For the image example, as stated in the paragraphs above, one would expect that any patterns could occur at any part of the image with equal probability — not accounting for edge effects. Under this assumption a CNN is a useful choice as, due to the fact that the kernel is shared over the entire image, it acts identically on each feature independently of where it is placed in the image.

Now, other data have different symmetries. Take, for example, graphs embedded in $\mathbb{R}^3$ representing molecules [51]. A single molecule is equally likely to

occur in any orientation and translation. As such the distribution of molecules poses is E(3)-invariant [108, 165].

There are many ways to make sure that models are "compatible" with such G-invariant data, where G is a group describing the symmetry. Echoing the ViT above, one might take a more flexible model and train it with as many transformed examples as possible but this requires resources and time. Or one can modify the loss function to take this group action into account; a schematic example would be

$$\mathcal{L}_{\text{equivariance}}(\widehat{y}, x) = \mathcal{L}_{\text{standard}}(\widehat{y}, x) + \sum_{g \in G} \|g\mathcal{N}(x) - \mathcal{N}(gx)\|, \tag{1.1}$$

$$\mathcal{L}_{\text{invariance}}(\widehat{y}, x) = \mathcal{L}_{\text{standard}}(\widehat{y}, x) + \sum_{g \in G} \|\mathcal{N}(x) - \mathcal{N}(gx)\|, \tag{1.2}$$

depending on whether the network should commute with the group action:

$$\mathcal{N}(gx) = g\mathcal{N}(x), \tag{1.3}$$

that is, $\mathcal{N}$ is G-*equivariant* (eq. (1.1)), or if the network is G-*invariant* (eq. (1.2)):

$$\mathcal{N}(gx) = \mathcal{N}(x). \tag{1.4}$$

In either case $\widehat{y}$ is the ground truth for the input x. If the group G is continuous, or otherwise infinite, the sum is generally replaced by Monte Carlo integration. However, this approach also requires applying the network $\mathcal{N}$ on multiple transformed data points for each training step increasing the computational requirements.

Another approach is to canonicalise the data [70], that is to align each data point to some "standard" orientation, and then apply a standard network on the aligned data. An illustrative example of this is to rotate natural images so that the sky is upward and keeping track of the rotation that was needed, applying the network and then rotating the output of the network back to the original rotation. This approach does however rely on that the aligning never fails and if the alignment is done by a network than this network needs to learn the symmetry, which requires data and training.

If one wants robustness and theoretical guarantees, even for group transformations on out-of-distribution (OOD) data, then the best way is to build this group compatibility into the network from the start. In doing so you get the best possible theoretical guarantees for the behaviour of the network on transformed data, and due to this strong inductive bias one also gets higher data and parameter efficiency [12, 49, 40]. Of course, through building the group

structure into the network one does lose some flexibility.

All in all, the route one takes to deal with the geometric structures in the problem depends on the specific task and the data that is available, and the subfield of machine learning working with these geometrical inductive biases is called *geometric deep learning* [17].

In this thesis I will discuss some aspects of geometric deep learning applied to vision tasks: data efficiency through data augmentation or a manifestly group "compatible" network (chapter 6 and **Paper II**); and the importance of a well suited grid for spherical images (chapter 7 and **Paper III**).

On the mathematical side I additionally elaborate on the nature of equivariance and weight sharing (chapter 4 somewhat connecting to chapter 2 of **Paper I**); and construct a framework for general equivariant non-linear maps (chapter 5 and **Paper IV**).

1.2.2 Motivation from a mathematician's point of view

A large portion of current machine learning research is driven by empirical exploration [120].With that said, there are many different ways to approach machine learning problems in a mathematical way [163, 75, 138]. One such approach is to formalise the inputs as functions — so-called *feature maps* — from the domain on which the data lives to a "feature space", typically a vector space. In this setting the layers of a neural network become maps between function spaces. The overall network's properties, such as equivariance, are then directly inherited from the mathematical properties of these constituent maps.

Cohen et al. [29] took this approach when they formalised data on a homogeneous space $X \cong G/H$ as sections of an associated vector bundle $G \times_\rho V$ where (ρ, V) is an H-representation. This space of sections is isomorphic to the *induced representation* which is a space of functions

$$\mathrm{Ind}_H^G \rho = \left\{ f : G \to V \mid f(gh) = \rho(h^{-1}) f(g) \right\}, \tag{1.5}$$

together with the left regular G-action

$$[k \triangleright f](g) = f(k^{-1}g). \tag{1.6}$$

The H-equivariance condition in eq. (1.5) is called the *Mackey constraint* and is integral to the theory of induced representations [96, 95, 24].

This perspective allows the use of large swaths of mathematical knowledge relating to homogeneous spaces, group theory, and representation theory. In [29] Cohen et al. used this setting to derive a theory which, with some assumptions along with clarifications from [6, 7], covers all linear layers and found that they could be written on the form

$$[\phi f](g) = \int_G \widehat{\kappa}(g^{-1}g')f(g')\mathrm{d}g', \tag{1.7}$$

where the kernel $\widehat{\kappa} : G \to \mathrm{Hom}(V_{\mathrm{in}}, V_{\mathrm{out}})$ satisfies

$$\widehat{\kappa}(h_1 g h_2) = \rho_{\mathrm{out}}(h_1)\widehat{\kappa}(g)\rho_{\mathrm{in}}(h_2), \qquad \forall h_1, h_2 \in H. \tag{1.8}$$

Specifically, these maps are G-equivariant linear maps — i.e., *intertwiners* — from the induced representation $\mathrm{Ind}_H^G \rho_{\mathrm{in}}$ to $\mathrm{Ind}_H^G \rho_{\mathrm{out}}$.

One can partly use this for a non-homogeneous base space by constructing an associated vector bundle from a principal H-bundle P and an H-representation (ρ, V). Then the space of functions $f : P \to V$ satisfying

$$f(ph) = \rho(h^{-1})f(p) \tag{1.9}$$

is isomorphic to the space of sections $\Gamma(P \times_\rho V)$.

We used this approach to study local actions in chapter 2 of **Paper I**. Additionally we use the case with a homogeneous base space in **Paper IV** to extend the theory by Cohen et al. to encompass equivariant non-linear maps. Through this we unify several current day architectures — among others, convolutional neural networks and transformers — in a single framework.

On the more applied side we in **Paper II** also investigate the data efficiency one gets by restricting the hypothesis class of the networks to being equivariant networks as compared to approximating equivariance in network through data augmentation. And in **Paper III** we adapt the well performing SWIN model to the HEALPix grid and hence create a model that natively works on spherical data.

1.2.3 Non-technical motivation

Geometry and symmetries are everywhere around us. For example, a coffee cup stays a coffee cup even if it moved from one place to another or if it is rotated upside down. Mathematically speaking the class of the object — cup in this example — is invariant to translations and rotations — that is, it does

Figure 1.1: Picture of a coffee cup [154] with two "pixels" marked. For an untrained network that should detect cups in an image both of the marked areas are equally indicative of the presence of a cup in the image.

not change from a cup to something else due to any of these transformations. Humans have no problems dealing with such transformations as we have a very high level conceptualisation of objects around us and we separate the object from its position.

However, this is generally not true for neural networks. For a neural network to operate on something we need to represent the input in some way that computers can understand: using numbers. An image, for example, can be represented as an array of pixels, often arranged in a rectangular grid but it can be represented in other ways as well.

To exemplify: assume we have a *untrained* network constructed for the task of detecting if an image contains a cup or not. As the network has no conceptualisation of objects, any pixel in the pictures has initially the same amount of information on whether the image contains a cup; a pixel from the table can be as indicative to the network as one from the cup itself. See fig. 1.1. Only by showing many examples of many different cups in many different settings — training the network with a diverse dataset — the network can start to identify features that are indicative of a cup and separate these features from elements irrelevant for the task.

A similar effect appears when one adds transformations to the picture. Specifically, how to disentangle the content of the geometry. Here there are two important factors: how one represents the data in a way that is consistent with

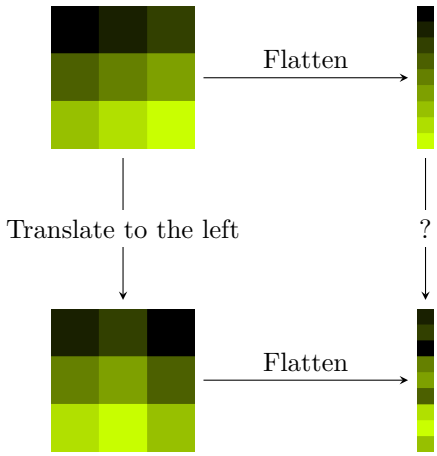


Figure 1.2: Different representations of a simple image. Left: a 2-dimensional, normal, representation of an image, note that it is clear which pixels are adjacent to each other. Right: a 1-dimensional, flattened, representation of the same image, note that the connectivity — the adjacency of pixels — is not immediately clear. Bottom: A shift to the left is well represented in the 2-dimensional representation (left), while in the 1-dimensional representation (right), this shift looks chaotic.

the geometry and transformations, and the type — or architecture — of the network.

We stay in the image domain and start with the representation of the image. As mentioned above, an image is often represented as a 2-dimensional rectangular array (left side of fig. 1.2), but one can also represent an image in different ways. One such alternative is to "flatten" the 2-dimensional image to a 1-dimensional sequence of pixels (right side of fig. 1.2), and as long as one knows the dimensions of the original image there is no problem switching between the two representations. However, if we now move all pixels in the top left image of fig. 1.2 one step to the left (bottom left of fig. 1.2) it is clear that one representation is better suited to this transformation as the 2-dimensional representation moves predictably while the 1-dimensional representation transform in a seemingly random way (bottom right of fig. 1.2). Hence, if we want a network operating on the 1-dimensional representation to perform a task invariant to this transformation we need to show the network many examples of translated images. This is similar to how a normal network needs to be shown many examples of a coffee cup in different settings to separate the cup from the background, and this process is called data augmentation. As the transformation is "simpler", consistent with the representation, when viewed

in the 2-dimensional representation, the amount of additional examples needed for the network to learn the transformation should be smaller, depending on the type of network.

The other important factor is the type of network. Certain networks behave predictably on transformed data without seeing any augmented data. For example, so-called convolutional neural networks (CNNs) behave predictably when the input is translated. This allows a CNN to avoid having to relearn the appearence of a coffee cup at different positions in an image by looking at many translated cups; that comes "for free". Building this symmetry into the network is reasonable as we expect a cup in any position to still be a cup. This is an example of an inductive bias, which are when one uses a priori knowledge of how we expect the model to work to design the model.

Note that just having a CNN is not a carte blanche to perform well on translated data: if we apply a CNN to a 1-dimensional representation of an image we will no longer get the predictable behaviour under translations that we had for the normal 2-dimensional representation. This highlights that both factors are needed: a representation of the data which is compatible with the geometry and the transformations, and a model which behaves predictably when the data is transformed.

This thesis touches both parts of this from both a mathematical and an applied angle. The mathematical part of this thesis works with building a framework for both linear — chapter 4 of this thesis and chapter 2 of **Paper I** — and non-linear maps — chapter 5 of this thesis and **Paper IV** — using objects called fibre bundles and tools from differential geometry as well as group and representation theory. The applied part of this thesis firstly examines whether augmenting the data with a transformation not native to the representation of the data can bring a standard model up to the level of one whose architecture already encodes these transformations — see chapter 6 of this thesis and **Paper II**. Secondly, this thesis examines the potential benefits of choosing a representation of the data which is compatible with the underlying geometry and modifies a model to work with this representation — chapter 7 of this thesis and **Paper III**.

1.3 Short summary of papers and results

In this section I provide a short summary of the included papers and their main result.

1.3.1 Paper I

This paper presents a review of different methods in geometric deep learning (GDL) from a mathematical view on gauge freedoms in neural networks (Chapter 2 in **Paper I**) to equivariance in linear layers on homogeneous spaces (Chapter 3) and spherical convolutions (Chapter 6). The paper recounts some of the main approaches available in the literature and expands on details that were either missing or not clear in the respective works.

1.3.2 Paper II

This paper extends the spherical CNN by Cohen et al. [28] and studies the data efficiency of this equivariant network when compared to data augmentation for an ordinary CNN. Explicitly, this paper investigates if the performance of the inductive bias provided by the more sophisticated (extended) spherical CNN can be matched by a simpler network only through data augmentation. We find that this is possible for certain tasks but not for all tasks implying that the inductive bias granted by the equivariant spherical model cannot fully be learnt, or matched, by a simpler network, and hence that these sophisticated methods might be a good choice for more intricate tasks with a strong geometric prior.

1.3.3 Paper III

This paper presents a vision transformer called HEAL-SWIN that natively work with spherical images, such as images from fisheye cameras which are common in drones and current day vehicles. We show that this model outperforms one of the, at the time, premier architectures and performs well at different tasks.

1.3.4 Paper IV

This paper takes inspiration from previous papers — mainly [29] — which provided a mathematical foundation for linear layers for data on homogeneous spaces and extended this theory to provide a framework to model equivariant non-linear layers for data on homogeneous spaces. Additionally, we show that several of the most used architectures appear as special cases of this general framework and through this unify, e.g., CNNs and transformers.

1.4 Outline

This thesis contains several independent parts and sections. Firstly, chapter 2 presents a very general and high level introduction to the history and important concepts of artificial intelligence and machine learning. Chapter 2 can easily be skipped for a reader well versed in the topic.

Part II deals with the mathematical elements of my work and is split into the following chapters: Chapter 3 introduces some of the necessary mathematical concepts for the later chapters and can be skipped for readers with background in group and representation theory and fibre bundles. Chapter 4 expands on chapter 2 of **Paper I**. Specifically, chapter 4 presents an alternate viewpoint of the equivariant map in chapter 2 of **Paper I**. Chapter 5 gives a short introduction to **Paper IV** which sets up a mathematical framework for general non-linear equivariant maps.

Part III deals with the applied aspects of my work and is split into the following chapters: Chapter 6 presents an introduction and motivation to **Paper II** which investigates how data augmentation stack up against manifest equivariance. Chapter 7 gives an introduction to the HEAL-SWIN model which is the centre piece of **Paper III**.

Finally, part IV shortly states the conclusions of this thesis and future research directions.

2 History and general knowledge

Machine learning (ML) and artificial intelligence (AI) is an immense field with an already exceptional impact on society, both technically and socially, and its influence will probably only increase with time. As a consequence it is important to have a good understanding of what is going on, how it can affect society, generally how it works, some limits, as well as current challenges and problems. This chapter aims at answering some of these questions from a very high level as well as setting the stage for the rest of this thesis.

Readers familiar with the history and central concepts of AI and machine learning can safely skip this chapter.

2.1 Brief history of AI and machine learning

The history leading up to the current state in machine learning and AI is colourful and starts earlier than one probably would have guessed. In this section I will give a brief summary on the general history of AI and machine learning, for the interested reader, please see e.g. Crevier [32] for a good summary and analysis up to the early 1990's.

Interestingly, but not very surprisingly, the meaning and use of the term AI has changed over time. Marvin Minsky defined in 1968 AI as "the science of making machines do things that would require intelligence if done by men." [105, p. V] and this definition has held to today. Early on this was realised through knowledge bases and hard coded rules that the system would combine and use to arrive at conclusions. These were called *expert systems*, and did not use or need any training data. This is contrasted to the machine learning used

in modern day which uses few, if any, hard coded rules and learns essentially everything from data. As such I will try to give a brief, or maybe not so brief, historical summary concerning both tracks leading up to current day.

The start of the track leading up to the methods of modern day started with the *artificial neuron*. This was a mathematical model of a neuron proposed by Warren McCulloch and Water Pitts in 1943 in which the neuron fires if the number of *excitatory* inputs was bigger than some threshold and no inputs were *inhibitory* [100]. This concept was picked up and refined later to become the basic building block of the later appearing perceptron and fully connected neural networks.An important work building on these artificial neurons was the paper by Herbert Robbins and Sutton Monro in 1951 that published a method for stochastic approximation [124]. This method was later the foundation for the *stochastic gradient descent* used everywhere in modern machine learning.

However, AI was not really considered a separate field of research until 1956 when Claude Shannon, John McCarthy, Nathaniel Rochester and Marvin Minsky arranged the workshop *Dartmouth Summer Research Project on Artificial Intelligence* [99]. This workshop is currently considered as one of, if not the, founding event for the field of artificial intelligence [144]. Machine learning as a term is younger than AI and was coined by Arthur Samuel in 1959 [132, 20], who was an IBM employee at the time.

The next step for machine learning was taken by Frank Rosenblatt who invented the "perceptron" in 1958. The perceptron consists of several artificial neurons organised in *layers* where the output of one layer is used as the input of the next layer [125]. Rosenblatt also introduced the terminology "back-propagating error correction" in 1961 but on a high level without any implementation details [126]. Concurrent with Rosenblatts perceptron, and progressing the rules based AI, McCarthy started developing LISP in 1958 [32, p. 60] with the intention of creating a programming language for AI. Later, in the middle 1980s, there was a large industry in producing microcomputers called "LISP machine"s that were specialised in running LISP programs [32, p. 200].

Towards the end of the 1960's, specifically in 1967, Shun-ichi Amari published the first multilayered neural network, i.e. sending information through several processing stages — layers — that was trained with *stochastic gradient descent* [2]. Following this, in 1969, Marvin Minsky and Seymour Papert published a book *Perceptrons: An Introduction to Computational Geometry* [106] on the limitations of these early perceptron networks. In particular, they showed that perceptrons could not reproduce the simple `XOR`-function. This reduced the general interest in continued research and funding connected to neural networks dwindled [129, p. 22]. However, in the following year, Seppo Lin-

nainmaa, a master student at the time, published Fortran code for the modern version of *automatic differentiation* method [92, 136].The automatic differentiation method allows for automatically update a networks weights based on observed behaviours and became a universal cornerstone in machine learning algorithms. Now automatic differentiation is known as *backpropagation* and is a cornerstone in the training of almost all modern neural networks.

Inspired by the work of the British on breaking the German codes during the second world war, people saw translation between languages or computers interpreting natural language as a task not much more complicated than breaking a cipher [32, p. 110]. As a consequence *The Defense Advanced Research Project Agency* (DARPA) poured several million dollars into various AI related language projects during the late 1960s and the early 1970s. One such project attempted to construct a hands free interface to military equipment using natural language. The project seemed to have some promise on a small vocabulary of around 1000 words, but under more scrutiny a large part of this was due to a quite restricted grammar. DARPA was quite disappointed by this and cut almost all AI funding in 1974 [32, p. 117].

This setback, along with the Minsky's book pointing out the limitations of the perceptrons and a critical report in 1973 by Sir James Lighthill which stated "In no part of the field (of AI) have the discoveries made so far produced the major impact that was then promised" [90], caused the interest and funding to cool off further resulting in the first, so called, *AI-winter* lasting between approximately 1974 and 1980. Although funding was scarce during this period a lot of researchers continued working on AI-related projects.

At the end of the first AI-winter, in 1979, Kunihiko Fukushima published the *Neocognitron* ([170, original, in Japanese] [46, English translation]) which recognised handwritten Japanese characters using a learned convolutional kernel, superseding Fukushima's previous work where the kernels were hand-designed. The Neocognitron later inspired the convolutional neural networks [84].

General interest in AI grew again an in the early 1980's and on the machine learning side John Hopfield — who got the Nobel Prize in 2024 — introduced *Hopfield networks* [63]. These are a type of *recurrent neural networks* which uses loops to create a semblance of memory. However, this network type was already published by Amari in 1972 [3] and was not cited by Hopfield. Due to this some people call these networks *Amari-Hopfield networks*. About 14 years later the modern day use of backpropagation of errors was popularised in 1986 by Rumlehart, Hinton — who got the Nobel Prize with Hopfield in 2024 — and Williams, who used Linnainmaa's algorithm to train a network [128].

On the side of the rules based AI systems so called *expert systems* saw a lot of interest and development in the early half of the 1980's. The purpose of expert systems was to replace an expert, often in either configuring or diagnosing other systems. These were often constructed by using a database of rules and knowledge which the system would use in recommending actions [32, p. 197–201].

The increased engagement and development resulted in heavy investment into research and development. Roger Schank and Marvin Minsky saw this and trying to cool the general sentiments warned in 1986 that the general enthusiasm and optimism in and on the industry was too high. Specifically they stated that the expert systems pushing this AI boom was based on old programming methods fuelled by increased computing power from newer hardware and would not continue to meet the expectations. This was unfortunately realised as the databases grew and the maintenance needed to keep the systems up to date became unmanageable [32, p. 203&204].

Following the optimism, Apple and IBM in 1987 produced their own microcomputers with the purpose to challenge the dedicated LISP machines. As the LISP machines outperformed by the new computers by Apple and IBM meant that there were little sense in staying with the more expensive and specialised LISP machines. This caused a market collapse and multiple companies lost millions of dollars [32, p. 210] resulting in the *second AI-winter* lasting approximately between 1987 and 2000.

Up to this point there loosely existed two branches of AI: the formal one of the expert systems and the data driven one of neural networks. However, by the late 1980's the interest in the formal systems had faded. A key limitation of this approach is the so-called *qualification problem* [32, p. 120]: in practical settings, rule-based systems struggle to accommodate an infinite range of unforeseen scenarios as this would require defining rules for all possible cases.

This limitation became evident in complex, real-world applications. For instance, there was an early optimistic belief that complexity was not in the mind but rather in the environment [143, p. 51] and hence any problem could be solved if one just discovered the correct set of rules. Based on this belief the U.S. military started a project in 1983 with the goal of developing an autonomous vehicle capable of navigating terrain at speed, use camouflage and cover natively [32, p. 210]. As we know today, the problem of autonomous vehicles is not so easily solved.

The reoccurring realisation that the complexity does not only lie in the environment were one of the factors pointing to the modern data driven approach where neural networks learn patters and actions from data or by interacting

with environments. To this end, in 1989 Christopher Watkins introduced *reinforcement learning* [157] which allows models to learn through trial and error through interaction with some environment.

Machine learning, the data driven part of the field of AI was dormant until around 2009 when the *ImageNet* dataset was released [34] and started the, still ongoing, *AI-boom*. An annual competition was held on ImageNet and in 2012 this competition was won by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton with the deep convolutional network *AlexNet* [80] beating the runner up in top-5 error rate by 10.8 percentage points. They conclude that the reason for this success was that the network consisted of many layers; making it a *deep neural network* (DNN). This was computationally expensive, but using GPUs made it feasible. Hence the field of deep learning was born.

All this laid the groundwork for the recent successes within the AI field. Just to name a few: In 2016 Google DeepMind's *AlphaGo* [140] — a Go-playing neural network trained through self-play (reinforcement learning) — beat Lee Sedol by 4 to 1; at the time Lee Sedol was regarded as the strongest Go player in the world; Continuing their highly impactful work DeepMind published *AlphaFold* 2 and 3 during 2020-2024 [69, 1] which has had a huge impact in medicine and biology: shortening the process of determining the structure from taking years to only seconds with high accuracy [35, 79, 87]; DeepMind have also made great progress in using machine learning models for weather prediction resulting in the *GraphCast* model which can generate a 10 day prognosis on a single GPU in under a minute — for comparison the otherwise leading traditional model (ECMWF HRES) takes hours on a supercomputer and GraphCast delivered more accurate predictions for over 90% of the predicted weather variables. In addition to these advancements, one should not forget the immense development in large language models — and other types of generative models — that has happened the past year. As it stands, the field of machine learning and AI has never been larger and progress seems to only be accelerating.

In his book from 1993, Crevier [32, p. 215] writes "It is not yet clear whether this headway of neural networks is a victory or a defeat for artificial intelligence as a field of scientific inquiry." and contrasts the neural network approach to that of the symbolic manipulation of "conventional AI". With the benefit of hindsight of the last 30 years in hand I feel pretty confident in saying that, yes, this approach has been a victory for the field of artificial intelligence.

However, with that said, in this thesis I will discuss the benefits of not just learning everything from data but what can be gained through injecting prior knowledge of geometry or symmetries in the problem. In a sense this would be a moderated approach of the modern data driven approach.

2.2 General introduction to AI and machine learning

This section deals with introducing AI and machine learning on a high level.

2.2.1 Terminology and fundamental concepts

In this section I will, with minimal technical details, introduce some general terminology and fundamental concepts that will help readers new to this material orient themselves in the general discourse and will serve as a good foundation when introducing more abstract concepts. As this introduction is focussed on introducing the central concepts it is, by necessity, light on technical details. Hence everything presented here are simplifications and generalisations, for more details on the material presented here please see [129, 53].

The hierarchy of AI

One can immediately see that Minsky's first definition of AI: "AI is the science of making machines do things that would require intelligence if done by men." covers a large swath of different systems depending on your interpretation; everything from the first rule-based expert systems to modern large language models. In modern day there is a trend of calling all "intelligent" software systems for AI, which, while this aligns with Minsky, it removes nuance from our terminology and discussion. Call these systes AI without thinking can misleadingly suggest such systems have general human-like intelligence, whereas until very recently most AI systems were narrow — trained for a single specific task. Although there is no exact definition, in modern nomenclature, a human-level intelligent system proficient at a wide range of tasks would be referred to as *artificial general intelligence* (AGI).

In reality, most software called "AI" today is a *machine learning model*, or just in ML circles just "model". A model is a system whose *parameters* — the numbers defining how a model functions — are learnt from data as opposed to explicitly coded by hand. In this thesis I will generally use the short form "model", but in general discussions I may still refer to AI in the broader sense. The training process results in models which, often, fall under Minsky's definition of AI but without the mysterious shimmer.

I will now discuss the hierarchy of AI in more detail, see fig. 2.1.

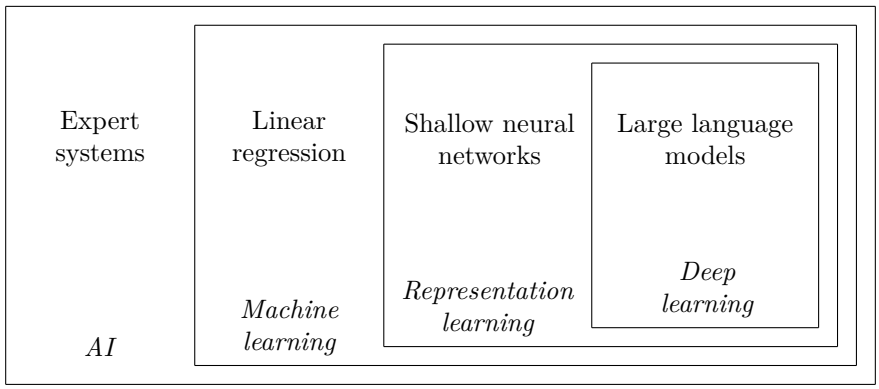


Figure 2.1: The hierarchy of artificial intelligence. *AI* includes all programs and algorithms doing something that otherwise would "require intelligence". *Machine learning* (ML) is the subset of AI these where parameters of the model are updated based on data. *Representation learning* is the subset of ML where the algorithm decides what to focus on, often with no human input. *Deep learning* (DL) is the subset of representation learning where the model processes the input data over several stages (layers). Reproduced with inspiration from [53, fig 1.4].

Specifically AI is an umbrella term covering all possible "intelligent" programs. An example of this would be the expert systems of the 1980's. The next level down is *machine learning* which covers all algorithms where a model uses *parameters*, a set of numbers, which are automatically updated during the training process. Examples of simple machine learning includes *linear regression* which deals with fitting straight lines to data. In this setting the learning means finding the best parameter values for the line. A special case on linear regression is the methods of least squares which uses properties of the problem to find optimal parameters in just one step of linear algebra.

Sometimes, especially early on, these machine learning models use human designed *features* for learning. A feature is some specified combination of the input data.[1] For example, if one is interested in predicting house prices, one input could be the area and one input could be the number of rooms. With these one could create a feature by dividing the area by the number of rooms to get the average room area, then the model combines these features through the parameters to get a prediction of the house. However, this requires that humans choose and design what features we think are relevant.

This leads us to the next step in the hierarchy which is *feature learning*. For models performing feature learning it is left up to the model to find the, hope-

[1] Often one calls each element in the input data an *input feature*.

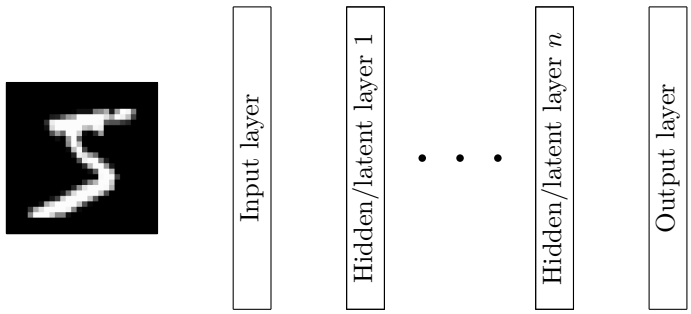


Figure 2.2: Schematic figure of a deep neural network (dnn). The input data is fed to the input layer which process this data in some way before sending it to the next layer. The layers between the input and output are generally called *hidden, or latent, layers* which each performs more processing until the data reaches the *output layer*. How one design and interpret the output layer depends on what the model is used for, but an easy example is how certain the model is that the input data is certain classes.

fully relevant, combinations of the input data to form the features. Often a machine learning model is built using several *layers* where, roughly speaking, each layer processes the output of the previous layer. See fig. 2.2. In this nomenclature the input is referred to as the *input layer* and the output as the *output layer*; all layers between the input and the output are called *hidden layers*. The representations learnt in these hidden layers are often called *latent representations*. The final step in the AI hierarchy is *Deep learning* in which the model consists of many hidden layers.

A common example of a machine learning model is a *neural network* which exists in several forms. A simple example is a *fully connected neural network*. Simply speaking a fully connected neural network encodes input data as a set of numbers, called input *nodes*. Then these numbers, nodes, are processed in the first layer by being combined and mixed. The exact mix of these nodes is specified by a set of *weights*, or *parameters*, and the result are sent as the input to the next layer. See fig. 2.3 for a visual representation of a simple neural network.

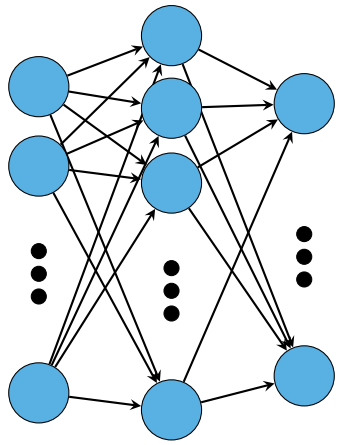


Figure 2.3: Visualisation of layers in a fully connected neural network. Each arrow has a number associated to it, the weight, and all arrows coming into a node determine how the data in the previous layer is mixed.

General information on machine learning models

This section will introduce some general concept relating to machine learning models.

For a general understanding of machine learning models it's necessary to have a rudimentary knowledge of the broad types of models. Roughly speaking, there are two large categories of models *discriminative models* and *generative models*. A *discriminative model*[2], or a classifier, learns to predict the probability of the input being a certain class or having some property

$$p(y_{\text{class}}|x_{\text{input}}), \tag{2.1}$$

that is "What is the probability of y_{class} given x_{input}?". One can then use this to categorise data. For example, a discriminative model for image classification could assign probabilities to the possible contents of the image. A real world example of this is a model that looks at images taken from cameras in vehicles and tries to classify the objects in the image.

Opposed to a discriminative model which uses the input to infer its properties

[2]The name comes from the model being able to separate — or discriminate — between different data.

or class, a *generative model* learns the joint probability

$$p(y_{\text{class}}, x_{\text{input}}), \tag{2.2}$$

that is "What is the probability of having both y_{class} and x_{input}?". Often one uses this to select the y_{class} and then generate the "input" x_{input} [13]. An example of this is the well known ChatGPT which iteratively generates the next word given by the most likely word, roughly, in a sequence started by a given text prompt.

A discriminative model is often trained with *supervised learning* which assumes one knows the true $\widehat{y}_{\text{class}}$ — the, so-called, *ground truth* — for each x_{input}. During training the model guesses the output y_{class} for the input x_{input} and then one compares the guessed y_{class} to ground truth $\widehat{y}_{\text{class}}$. How well the predicted y_{class} aligns with the true $\widehat{y}_{\text{class}}$ is measured by some *loss function.*A common choice is the distance between $\widehat{y}_{\text{class}}$ and y_{class}, that is how far the model was from being correct. The model weights are then updated to make the previous guess more accurate.

During training the models performance is tracked by some, so-called, *metric*. A common choice is to keep track how often the model guess correctly, that is, its *accuracy*.

On the generative side things are more fuzzy. One way to train generative models is giving it data and have the model recreate the data — this is especially common for autoencoders. While in the supervised learning for discriminative models the choice of loss has clear candidates, the same does not hold for generative models. If trained in the way mentioned above, one can take the loss as the distance between the input and the output, and update the weights to minimise the error. But if there are many "correct" outputs it is much harder to pick a good loss function [48].

Often a model is built from several parts. One common setup is to have two general parts: one *encoder* and one *decoder*. Intuitively, the encoder part takes the input and learns usable features of these data which are the latent representation. The decoder then takes this latent representation and computes the output based on this.

To illustrate this, consider the following scenario: I describe an image and describe it to after which you proceed to draw it. In that scenario I *encode* the information in the image into words — the latent representation — and you *decode* the words into an image. Hence, the purpose of the encoder is to extract the relevant higher level, or conceptual, information from the more detailed inputs.

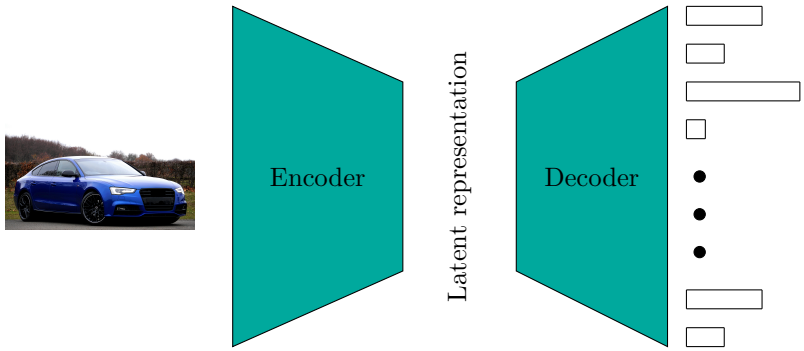


Figure 2.4: A schematic visualisation of an encoder and decoder. An input image is fed to the network and the *encoder* computes a (learned) latent representation. After this the decoder takes the latent representation and computes an output from it. For a discriminative model the output is some prediction on the input — e.g., predicting the model of the car. A generative model takes the latent representation and generates an output that could reasonably come from the training data. By adding noise to the latent representation, one can generate variations on the training data. Image of car from [15].

A discriminative model for classifying, e.g., car models in images can also be put into this context: I describe the input image to you and you guess what car is in the image. Again I form the encoder encoding the highly detailed image into abstract concepts and you decode my description into a guess of the car model. See fig. 2.4.

Data and learning algorithms

The process of training a machine learning model, or making the model learn, means using some algorithm to change the weights of the model in order to improve the model performance with respect to some metric.

To train a model one generally needs data in some form. Data exists in two broad categories: *Labelled data* for which each element in the dataset has a so called *ground truth* or *labels* which states what is "correct" for that data point; and *Unlabelled data* where these ground truth, or labels, are not available. The set of all data points one has access to is generally called a *dataset*.

A model is usually trained on a part of the dataset, called the *training data*, and its performance is usually tested after training on the *test data*. Importantly, the test data should not be available to the model during training since then

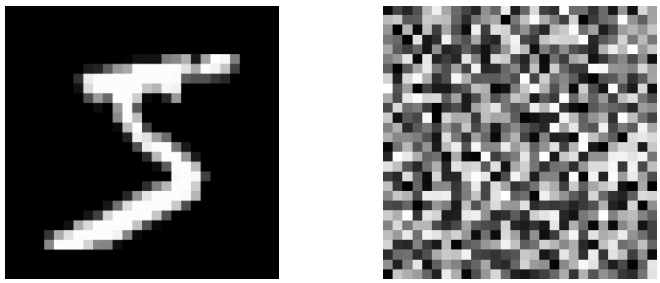


Figure 2.5: A sample image from the MNIST dataset (left) and a random 28×28 image (right).

the model can learn the test data eliminating its usefulness as a test. Often, a part of the available data is used to validate that the training progresses and that the model learns reasonable things. This dataset is called *validation data*.

One generally important concept when it comes to data is the *data distribution*. On a high level the distribution of a dataset is the spread of data in that dataset. For example, a dataset consisting of dogs and cats can look widely different depending on what breeds are included on each side; in this example the spread of breeds affects the distribution. Loosely speaking, one wants the distribution of the training data to be similar to the test data, or wherever the model would be used. Any data outside the distribution of the training data is called *out-of-distribution*. If the training data in the mentioned example only consists of images of golden retrievers for the dogs and sphynxs for the cats, then one would not expect a model trained on this dataset to be very good at classifying other breeds as dogs or cats. This is because they would not have been seen by the model during training and are as such out-of-distribution.

Finally, an intrinsically geometric property of data is the concept of *effective, or intrinsic, dimension*. This is most easily explained through an example: The *MNIST dataset* [85] consists of 60 000 scanned samples of handwritten digits where each sample is a picture of 28×28 pixels; see fig. 2.5. Naively, one could think that this means that the dataset is $28 \times 28 = 784$-dimensional as each of the 784 pixels could be varied independently. However, if one were to randomise all 784 pixels independently the result would never be a handwritten digit. This is because in any image of a digit no pixel cannot be fully independent of the other. In some sense this shows that not the entire 784-dimensional space are valid data points, in fact the dimensionality of the valid data is much smaller. A study published at *ICLR 2021* [116] estimated the intrinsic dimension of MNIST to be around 12. Hence, around 12 independent coordinates in the latent representation are enough to encode all relevant information of an

MNIST image. This around 12-dimensional space containing the relevant data of the MNIST dataset is called the *data manifold* and is an example of the *manifold hypothesis* [42] which claims that most data encountered naturally is rather low-dimensional. The manifold hypothesis and exactly how this relates to training and model generalisation is under debate, especially when it comes to natural language [117], but it seems reasonable for image data[53].

Once data is available one can train a model on the data. During *training*, or *learning*, the model parameters are updated from feedback on how it performs based on the chosen loss function. When the model weights does not change significantly any more the training is said to have *converged*. There are several broad categories of learning algorithms: *supervised learning*, *unsupervised learning*, *reinforcement learning*, and *evolutionary algorithms*. Which one chooses depends a lot on what type of data one has access to.

When training a model using *supervised learning* one compares the model output for a given input to the ground truth for that data point. Hence supervised learning requires labelled data. Then you update the weights of the model so the previous guess is improved. For example, in the case of supervised learning for image classification you want to update the weights of the model so that the likelihood the model gives to the correct class increases.

Unsupervised learning means that you update the model's weights so that it finds pattern and relationships between the data points without previous knowledge of the data or its meaning. This is most useful when the dataset lacks labels. An example of this is *clustering* where one updates the weights of the model so that data points with similar features are grouped together. Specifically, without labels, one could cluster images of dogs and cats so that the model outputs similar values for all cat images and clearly different values for all dog images.

Reinforcement learning (RL) is a completely different process where the model takes the position of some kind of agent and taking actions in some environment. The actions, or the result of the actions, are then assessed against some metric and the model weights are updated based on the assessment. One benefit of using RL to train models is that using human curated data always, either directly or indirectly, constrains the data to a human perspective and often limiting the model, and by using the model generate its own data by interaction with the environment it is possible to lessen the human bias. This was, for example, observed as after DeepMind's Go-playing model *AlphaGo* [140] — which was initially trained on human games — beat Lee Sedol 4-1, DeepMind trained the model *AlphaGo Zero* [141] using only self play. AlphaGo Zero started with only the rules of the game and after playing only against

itself it surprisingly and convincingly beat all previous models trained using human games. A version of reinforcement learning is *reinforcement learning with human feedback* (RLHF) in which, in addition to the normal feedback and grading of the models actions, a human also gives feedback to the model. This is generally done with the intent of aligning the model with humans: making it more understandable and human in its actions. Although one can make RL models more "human" by using HF this can also (re)introduce human biases into the model. With that said, even for pure RL without HF the metrics and environments are designed by people and hence might inject biases or priorities unknowingly.

Based on this some people see RL as a fundamental cornerstone in future machine learning models as this allows the models to escape the limitations of human data. Silver and Sutton from DeepMind calls this the *Era of Experience* [142].

Evolutionary algorithms is again a different training process, although more akin to RL than supervised or unsupervised learning. Generally speaking this is a wide class of training algorithms and when using an evolutionary algorithm to train a model one generates a set of models with different parameter values and check how they perform. Then the worst performing models are deleted and variations, mutations, on the best models are created. One can also mix the different models to create "offspring" through other avenues than mutation. If one does this, it is called a *genetic algorithm*. Through this, the training tries to mimic natural selection.

Recently DeepMind utilised a version of evolutionary algorithms to let their new model *AlphaEvolve* discover new algorithms in mathematics and coding [109].

This thesis will only discuss the case of supervised learning for discriminative models.

When training a model, its weights are updated to capture and represent patterns in the training data. How well a model *generalises* is a measure on how the model performs on unseen data. During training, however, it is possible for the model to reproduce the interesting properties of the training set perfectly; in some sense, memorising the training set. If not dealt with correctly, this could lead to *overfitting* where the performance on unseen data degrades due to the model only learning the training set. There are methods to counter this, these are however out of scope and will not be discussed here, even in brief. For a while people used this as an argument against scaling up networks, as more parameters just makes it easier for the network to memorise the training data and this is an example of the *bias-variance trade-off*. However, this does not

hold as when scaling model sizes over a certain threshold the models generalise better again [14]; this is called the *double descent* [134]. The double descent effect is what is underpinning the enormous large language models of today.

To keep track of the models performance and warn of overfitting one can use different *metrics*. Metrics are different functions used to keep track of the models performance. The first and often most intuitive metric, at least for classification tasks, is model *accuracy*. The accuracy of a model is defined as the total correct classifications divided by the total number of predictions the model has made. This seems like a perfectly good metric, but all metric have pitfalls.

When it comes to the accuracy metric this is clear when the dataset is imbalanced. For example, if one has the task of predicting if a patient has a medical condition from measurement data and the dataset contains 99% healthy people, then the model can achieve a very high accuracy (99%) by simply ignoring all sick patients. A model is never perfect, so observing this behaviour from an accuracy score of 99% would be essentially impossible to separate form a model that actually identifies all sick patients.

This brings up the concepts of *true and false positives* and *true and false negatives* when it comes to binary classification — that is classifying something into two classes, e.g., into true and false. A *true positive* is when the model predicts something being true and it is true. A *true negative* is when the model predicts something being false and it is false. A *false positive* (type I error) is when the model predicts something being true (e.g., a patient has a condition) when it is false (they do not). A *false negative* (type II error) is when the model predicts something being false (e.g., a patient not having a condition) when it is true (they actually do). Depending on the application one has to decide on if type I errors or type II errors are worse and select a metric that measures what matters.

To aid in this, one often uses many metrics to keep track of different aspects of the models performance. The *precision* metric checks how many of the models predicted positives are actually positives, for example, how many of patients the model predicted as sick are actually sick. The pitfall for this metric is that if the model predicts no-one as sick, or only the extremely clear cases, then the model will have perfect precision. Then there is the *recall* metric which checks how many of the actually positive points are predicted positive by the model, and this metric can be optimised by predicting everything as positive. These are just some examples, there are many more metrics to track the models performance based on what the model tries to do.

The above discussion shows that one should never rely on a single metric, but

rather use a combination and always be aware of how the different metrics can fail to capture unwanted behaviour.

Often training a model from the ground up is not feasible — either due to lack of data, compute resources, or other reasons — and in that case one can take a model already trained, hopefully on a similar task or dataset, and *fine tune* the model. When fine tuning a trained model you take the model and train it on data for the specific application you want to use the model for. The concept is that the majority of the training is already done on a more general task so the changes needed to specify the model to a more specific task would not be that big.

A different version of this is *transfer learning* in which one only updates the weights of the final layers. The underlying concept is that a lot of the earlier layers should work with more concrete information, e.g. colours and edges in the case of images, so one can use these layers as they are and only update the later ones when the information is more semantically meaningful for the application. As one trains a smaller part of the network less data is needed for this.

2.2.2 Geometric deep learning

For more details on this section see [19] or the later proto-book *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges* [17] and its associated course [18].

In many cases a problem, or dataset, contains some symmetry. An example of this is identifying cells in microscopy images where the rotation of the image will not affect the cell types; see fig. 2.6a. Another example is when working with graphs where the relative position of each node is important, e.g. molecules where each node is an atom and each edge is a bond; see fig. 2.6b. In that case moving or rotating the entire molecule will not affect its properties.

These are examples of data with a transformation — group — symmetry and if we has a task of determining the cell types or some property of the molecule, these tasks would be invariant to the group symmetry. The field of *geometric deep learning* (GDL) [17] is a subfield of machine learning which works with incorporating group symmetries — or other geometrical aspects — into the training pipeline. This could be by modifying the models, finding better ways to represent data, or through some other method taking advantage of the known geometrical properties.

In both of these examples — molecules and images of cells — the symmetry

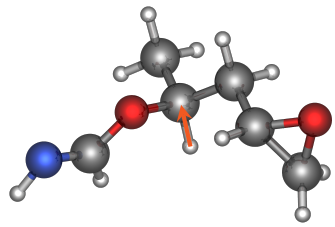
stems from the fact that when working with the data we are making some choice in how we represent the data: the orientation of the image or the global offset or rotation we give to the molecule. This choice is arbitrary and does not affect the underlying system, although it changes how the system looks in the chosen coordinates. In physics this appears rather often and is referred to as a, in this case global, *gauge symmetry*.

Just because the underlying system is independent to this choice of representation does not automatically mean that models are.This is because the system needs to be expressed in coordinates, or some other numerical representation, to be passed to a network and this numerical representation changes with the change of coordinates. Due to these — in some sense arbitrary — changes in the numerical representations a single system might look completely different to a network. Now, if the network is not consistent under the changes coming from changing coordinates one can get vastly different output for the same system. And worse, these outputs might have no relation to each other. The natural question is then: "If we know that this choice of orientation does not matter for the underlying system, then how does we make sure that the models respect this symmetry?"

The set of assumptions that the developer uses or builds into the system is called *inductive bias*. For example, that the underlying system is unchanged if we pick a different coordinate system. There are different ways of introducing an inductive bias for these symmetries.

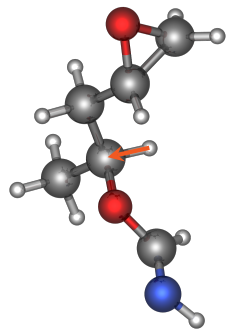
The first approach is to let the model learn everything from the data without further restrictions. Often, in the case of the examples presented here, this amounts to moving or rotating the data, including the ground truth where needed — for example when we want to detect both the type of cell, which does not change with rotations, and its position, which does change with rotations — through all relevant angles. This is an example of what is called *data augmentation* in which one transforms the training data so that the model experiences different representations of the data, and in this way makes the training set larger. One benefit of this is that it is very easy to implement as it requires no modification of the model, only the dataset, or possibly the training, needs to be altered. The downside however, is that the training set becomes much larger — as all transformed samples are added to the dataset — and in turn the model needs more training time and resources in order to converge.

(a) Example of cells. Image from [97].



(b) Example of a molecule, visualised from [65].

(c) Example of cells. Image from [97].



(d) Example of a molecule, visualised from [65].

Figure 2.6: Examples of data points from data which exhibit some symmetries. Note that the cell positions just move (the marked positions) under rotation, but the force noted in the molecule (the arrow) is both moved and rotated. In both of these cases the underlying symmetry comes from the fact that there is no preferred orientation of these systems, how the features, e.g. position and force, looks depends on how we view the data point and will hence change, e.g. move and rotate, when we change how we view the data.

Additionally, models trained with data augmentation can only be expected to respect the symmetry existing in the (augmented) training distribution, and one have no guarantee for how the models behaves on out-of-distribution data. For example, this means that if we augment the data only with 90° rotations, then we only knows that the model respects the symmetry of these 90° rotations. Any object that exists in the training data but viewed at 45° can give a drastically different response than if viewed at 0° or 90°. Furthermore, you have no guarantee that the model will respect this symmetry on any object outside the training distribution. E.g. if you train your model to be invariant to 90° rotations — e.g., an image of a dog rotated by 90° yields the same output — by augmenting the dataset, then the model might still give completely different outputs for a car depending on the rotation of the image.

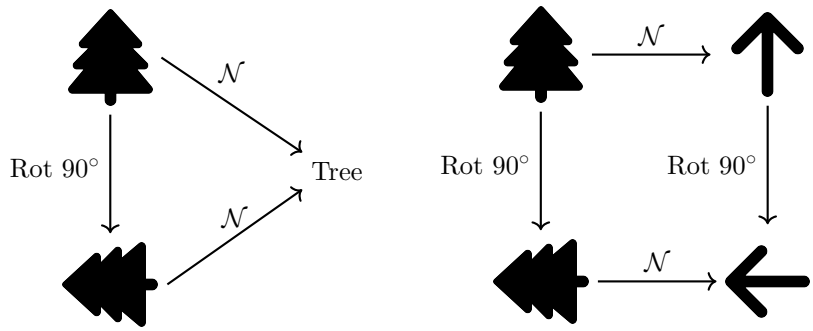
Another approach, which is the one relevant to this thesis, is when one encodes this symmetry into the model itself and thus gets theoretical guarantees that the symmetry will always be respected no matter if the model is trained or not. This also gives guarantees for out of distribution data, that is, one has a guarantee that the model will be consistent under transformations even for data it has never seen which is not true if one uses data augmentation.

However, this approach has downsides as well. The main one is that requiring this property puts constraints on the possible models one can construct and actually understanding how or why they work is theoretically demanding. Additionally, by their nature, models respecting the built in symmetry are less expressive and they might be more difficult to train [89, 155, 77]. There are however methods to improve the performance and training of such models, see e.g. [113, 115].

In some sense one can view this balance as a trade-off between building the model or algorithm based on prior knowledge (inductive bias) and relying on huge amounts of data to arrive at similar final outcomes.

Equivariant neural networks

Equivariant neural networks are a subfield of geometric deep learning dealing with the case where the output of the model transforms "predictably" when the input transforms. An example of this is a network that operates on images and predicts which direction in the image is "up"; see fig. 2.7b for a visualisation. In that case, if the input image is rotated, then the direction the model predicts should rotate the same way. That is, it does not matter whether you apply the rotation to the input image first and then the model, or apply the model first, and then rotate the output: the two different methods agrees. This would



(a) Example of an *invariant* network $\mathcal{N}$ that predicts the class of the input.

(b) Example of an *equivariant* network $\mathcal{N}$ that predicts the direction of the tree.

Figure 2.7: Visualising what equivariance and invariance means for a neural network $\mathcal{N}$. Compare this to fig. 2.6 where the part of a network that predicts the positions of cells needs to be equivariant to translations and rotations, while the part that classifies the type of cells needs to be invariant as the type of the cells are not changed by a translation or rotation; obviously the per cell classification needs to move with the cells.

be an *equivariant model*, and this property that the order of transforming the data and applying the model does not matter is called *equivariance*.

A special example of equivariance is when the output of the model does not change at all when the input is transformed, in that case the model is said to be *invariant*, and this property is called *invariance*.

Models on curved spaces and sampling

Another area of where geometric deep learning is useful, without any symmetry requirements, is when dealing with data on curved spaces. The go to example is data living on the sphere, e.g. weather data.

When working with continuous data in machine learning settings one must always perform some type of *sampling*. That is, decide on a set of discrete points where one measures the values of the continuous field, as most models take a finite set of numbers as input. Here we will only discuss two sampling methods for data on the sphere: the *Driscoll-Healy grid* and the *HEALPix grid*.

Often, the go to sampling for data on the sphere is to sample on the *Driscoll-Healy grid* [37], which is essentially a grid consisting of points as constant

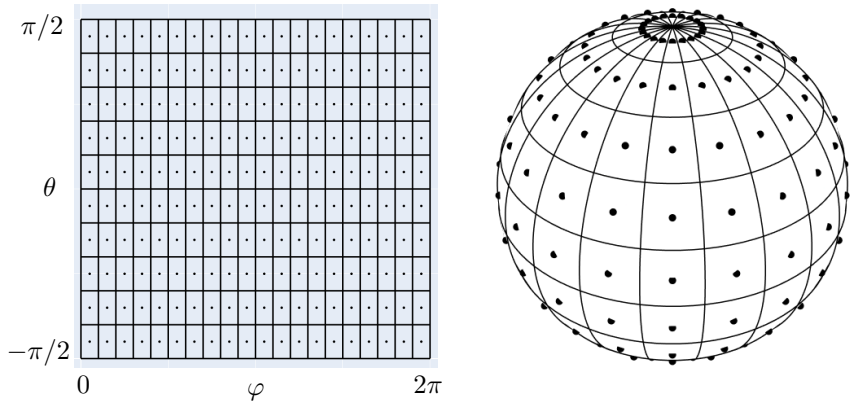
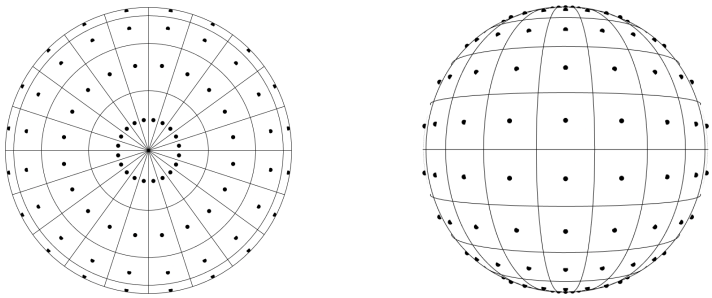


Figure 2.8: The Driscoll-Healy grid [37], viewed in angular space (left) and on the sphere (right). Each dot is a sample point. Note that when viewed in a latitude-longitude format, the sample points are equally distributed in a rectangular patter, while due to the curvature of the sphere the samples points are denser in the polar regions.

latitude and longitude. See fig. 2.8 for a visualisation. This grid is very easy to implement and work with, as the sample points look like a regular rectangular grid when plotted on a latitude-longitude grid (left). It however has a severe disadvantage as, due to the curvature of the sphere, the sample points are dense close to the poles while being sparse around the equator; compare fig. 2.9a and fig. 2.9b. Hence, there is a trade off in the density of the sample points: A reasonable density of samples for the polar region results in sparse sampling around the equator, and the converse that a reasonable sample density at the equator results in the poles being oversampled relative to the equator.

This is a downside as, for example, *spherical images* in latitude-longitude format appear stretched at the poles, and this could lead to that a model might overemphasize polar regions if not handled carefully. See fig. 2.10 for a visualisation of this distortion close to the poles.

An alternative for sampling spherical data is the *HEALPix grid* [55], visualised in fig. 2.11, where each sample point covers the same area on the sphere (right). This means that there is no implicit bias in the different areas of the sphere as the density of sample points is uniform. However, as a consequence, it cannot be visualised as a rectangular grid (left panel of fig. 2.11) which means that most models needs to be adapted to work for data sampled on the HEALPix grid. Selecting a good sampling method, or otherwise dealing with the curvature of the space, for your pipeline is part of the larger geometrical deep learning field.



(a) Driscoll-Healy pole view **(b)** Driscoll-Healy equator view

Figure 2.9: Comparison of sampling points in the polar region and around the equator in the Driscoll-Healy grid.

Figure 2.10: Due to the non-equal area in the sampling in the Dricoll-Healy grid we get the distortions visible at the top and bottom of this omnidirectional image. Image from the *Stanford 2D3D-S* dataset [5].

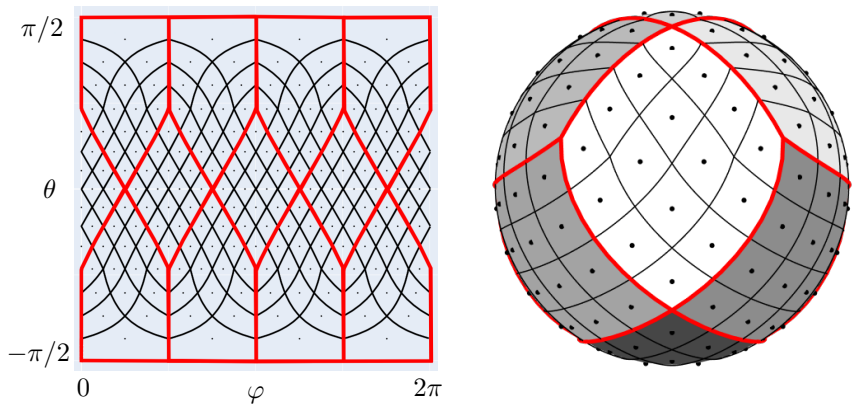


Figure 2.11: Visualisation of the HEALPix grid [55] made using `healpy` [169]. Note that the sample points are equally spaced out on the sphere (right). As a consequence, when viewed in a latitude-longitude format the sample points do not form a nice grid structure.

2.3 Impact on society by AI and ML

This section concerns the impact of AI and machine learning on society as a whole. I do not think that any of the work in **Paper I-Paper IV** will have any large societal impact or any ethical consequences. With that said, I think it is always important to be aware of the larger picture, and hence I chose to include this section in my thesis.

Not everything regarding machine learning and AI are good, and there is a lot to be written on both the benefits and drawbacks than I can provide here. The impact of AI on society is not my field of study, but however, for those reading this as their high level introduction to machine learning and AI I would be amiss without mentioning some of its impacts on society. Furthermore, I want to heavily emphasise that this is by no means an exhaustive coverage, nor curated by or ordered by any kind of metric.

The main sources for this section is the report *Energy and AI* from 2025 by the *International Energy Agency* (IEA) [66] and the report *Assessing Potential Future Artificial Intelligence Risks, Benefits and Policy Imperatives* from 2024 by the *Organisation for Economic Co-operation and Development* (OECD) [111]. The interested reader is referred to these reports for a more thorough breakdown.

Benefits

Here follows short texts on some areas where AI models are likely to be beneficial to society.

Accelerating science AI has a huge potential of accelerating science from finding relations in data — which, among other things, helped mathematicians prove new theorems in knot theory [33] and helped discovering a faster algorithm for matrix multiplication [41] — to generating new proteins in biomedicine — which has accelerated the mapping of proteins by a factor of approximately 45 000 [66, p. 17] — which fills a critical role in the design of new drugs.

Recently, more advances was made by DeepMind with AlphaEvolve where they find new efficiency gains in mathematical and coding algorithms [109].

Education AI, and large language models in particular, has a great possibility of enhancing education. Both in helping designing lesson plans [25] and

offloading other administrative tasks [23], to being a personalised additional mentor for both teachers and students [152]. Further, in areas with fewer schools LLMs could by acting as a digital teacher help offload teachers with too many students to give each proper attention. Of course, this final point relies on that the students are actively using LLMs in good faith, i.e., properly reflecting on the output with a critical eye, otherwise this could have adverse effects of over reliance on the AI models.

A study [50] found using a survey a statistically significant negative correlation between self reported use of AI models and the respondents self estimated skills in critical thinking. Additionally, a different study [148] found the respondents worried of negative effects on critical thinking from over reliance on AI models.

In order to avoid these negative potential effects the education systems should be restructured so that students and teachers have a healthy relation to these models. Several reports have been made on this [31, 104, 103, 152].

Healthcare AI can greatly assist in healthcare by, for example, offloading doctors in administrative work and help in finding symptoms underlying a diagnosis in images or other data [146, p. 9]. Additionally, it can help patients with keeping track of biometrics, with automatic warnings if values, or a combination thereof, lies outside the ordinary for the individual. It can also aid in making medicine more available to remote locations by self consultation or automatic screening for symptoms [146, p. 11]. A study tested GPT-4 in a healthcare setting and found that doctors with GPT assisting them in making a diagnosis fared only slightly better than doctors with conventional diagnostic resources. However, the LLM alone did better than both the previous groups [52]. A different study also found that GPT-4 outperformed physicians [64]. This indicates that there is more to be gained through better integration of LLMs in diagnostic reasoning.

Even though AI models can be hugely helpful in healthcare applications as it is, at least partly, one important part that needs further development is *explainability* or *interpretability* [130]. A model is called explainable, or interpretable, if the output of the model can be explained from the input in "a human understandable way" [131]. For example, if a model sets a diagnosis, a human could probe an explainable model to see what made it come to the diagnosis or decision it did.

This is important as this shows how well the models can be trusted, and in addition, this allows for human doctors to more easily learn from the models if they have learnt a pattern of symptoms not known to humans.

An additional challenge in healthcare application is that positive data, that is data for some condition, is often scarce. Hence it is difficult to get a well balanced training set, which needs to be compensated by other techniques, to make sure that the model are not making too many errors, either *false positives* — where the model states that a patient has a condition when they do not — or *false negatives* — where the model states that a patient does not have a condition when they do. To achieve zero errors is never feasible, so often one needs to decide which of false positives and false negatives are worse, and work to minimise those.

Assisting in decision making and optimising By the fact that AI models can sift through huge amount of data and finding relevant trends or correlations, that a human would have difficulties in observing, AI models can greatly improve planning and decision making [111, p. 14 & 16]. This enters at all levels, e.g. from helping at a personal level to optimising companies, energy consumption, and grid management [111, p. 18]. The IEA estimates that if existing AI applications gained enough adoption throughout industry this could "lead to energy savings equivalent to more than the total energy consumption of Mexico today" [66, p. 16].

On a separate note, recent advancement has resulted in a weather prediction model that can, at times, outperform our current best model for a fraction of the compute power, at least after training, making predictions more energy efficient and accurate [82].

Challenges and drawbacks

Here follows texts on some of the challenges and potential drawbacks of AI models. Note that none of the ones listed here are unavoidable but are areas which require work to improve the overall impact of AI.

Environmental impacts There is no denying that the rapid development for, and running of, machine learning and AI models puts requirements on the energy grid. Currently, a large AI-focussed data centre annually consumes electricity corresponding to around 100 000 households [66, p. 38] and the energy consumption of data centres are set to more than double by 2030 [66, p. 14]. One major drawback is that this energy consumption is very geographically focussed, meaning that local infrastructure needs to deal with this increased demand. Furthermore, the IEA estimates that the emissions from data centres will make up around 1.5% of the total energy sector emissions up to 2035 and

emissions related to data centres are one of the fastest growing emission source. With that said, IEA estimate that application of AI could result in efficiency gains causing a reduction of around equivalent to around 5% of energy-related emissions by 2035 [66, p. 18].

Outside of energy consumption, the dependency on critical minerals for data centres will increase, and sources of these are often geographically concentrated with China being the singularly largest source [66, p. 213-214]. This gives large amounts of control to specific countries. For example, due to China's export restrictions on gallium the price of gallium doubled between mid 2023 and end of 2024 [66, p. 214]. However, AI methods can also be used for optimising exploration and extraction of critical minerals reducing the societal and environmental impact of such activities [66, p. 120].

Another environmental impact of large scale computing in the modern day is the water usage. A study estimates that if OpenAI's model GPT-3 was trained on Microsoft's U.S. data centre would consume around 700 000 litres of available fresh water, not including any inference water usage [88].

Data collection, selection, bias, and privacy As mentioned previously, the modern usage of machine learning is extremely data hungry [93] but unfortunately acquiring high quality data is expensive [8]. For labelled data one needs to get labels for each data point, and this can be done in two ways: either one trains a model on a smaller set of labelled data, or use another algorithm, and use this to predict labels for the rest of the data [158, 26], this result is an example of *semi-supervised learning*; or one pays people to manually label each data point which takes a lot of time. Additionally it is well documented in the case of medical data that the opinions of experts can differ markedly [135, 147]. This means that in order to have high quality data with reliable labels one needs to consult multiple experts which drives up the cost of the dataset.

As mentioned previously, using the reinforcement learning paradigm, where the model learns itself through some interaction with either an environment or itself, allows for less, or even zero, human data. Using other training methods the models are generally limited to the distribution of the training data, which is often either generated or curated by humans, but using RL allows the model to transcend these limits and find new approaches humans have not thought about.

This was observed as the Go-playing model AlphaGo Zero [141] learnt Go only through self play, and was, at the time, significantly stronger than all humans and other AI models. Recently, a paper was uploaded to *ArXiv*, a site for scientists to share research in a wide variety of fields, presenting a learning

scheme for LLMs to learn mathematics and coding through RL by self proposing relevant problems and then solving them [167]. Silver and Sutton from DeepMind in their recent position paper "Welcome to the Era of Experience" [142] presented the position that future strong AI models will learn mostly through interaction with an environment and hence generating their own data.

As machine learning models are good at finding patterns and trends in data this puts heavy emphasis on that the data used to train the model does not contain any unwanted patterns the model can use as a proxy. A paper discussing explainability [123] showed an example of this by knowingly training a bad network to classify huskies and wolves and selected all pictures containing wolves to have snow. Due to this the network classified pictures of huskies over snow, or generally a light background, as wolves. Examination showed that the network really detected presence of snow and used that as an easier proxy for the task than actually using the animal in the picture.

That example is rather benign, but several other, more severe mistakes due to data, and how one collects data, exists. A machine learning model used in the USA to predict the health risk of individuals was found to be racially biased without having any features in the training data related to this. The cause of this is that they used health care costs as a proxy for health and African American patients tended to pay more than white patients for similar treatments and hence they were thought to be healthier, due to having spent more on healthcare. Another severe example is the *COMPAS* model which was used in the USA for predicting recidivism, i.e. the likelihood of a person repeating an offence. A study [4], their data analysis is available on github [119], found that COMPAS assigned higher risk of recidivism to African American defendants than was observed, while under estimating the recidivism rate for white defendants. Another group found that the COMPAS models fails certain group fairness metrics [58], while another study found that the COMPAS model is no more accurate than then people with no experience in criminal justice and a simple model with just two input features — as compared to COMPASs' 137 input features — is essentially equivalent. All this although the training data for COMPAS involved no racial features.

This behaviour stems from that machine learning models finds trends and pattern in used data, wherever that pattern comes from and independently of the source of the pattern. Hence if the existing data, e.g. prison recidivism, is skewed, knowingly or not, *from collection* there is a high risk that the model will itself perpetrate this bias. From this it is clear that machine learning models are likely to fall prey for confusing correlation and causation. E.g. if one trains a model to predict the risk of lung cancer and the training dataset contains the health information that patients have bad teeth the model might

use having bad teeth as a large sign of lung cancer, although both can stem from an underlying source, e.g. smoking. Thus one must be very clear on if the data is skewed from collection and what patters and biases exist in the training data. Unfortunately, it is not always easy to detect if the underlying training data has a bias or not so one must always be aware of the possibility.

When it comes to collecting the huge amounts of data needed for current day LLMs and foundational models some companies resort to scraping data from the internet without much consideration of the source [9, 38]. This results in issues regarding intellectual properties [112] and privacy [110]. There are studies showing that, in certain circumstances, it is possible to reverse engineer the data a model has been trained on [60, 21, 22, 107], further exacerbating the importance of keeping track of the training data.

Furthermore, as AI models allows for processing huge amounts of data without supervision, this allows for autonomous surveillance of public spaces. Many countries already use AI models for this purpose and with growing software, as well as hardware, capabilities this will probably increase [111, p. 24].

Trust, hallucinations, and misinformation Anyone who has used a large language model, such as ChatGPT, has encountered *hallucinations* which are statements the model claims are true but, in fact, are not. This means that one cannot use the output of a LLM without double checking all the statements. If not read with this in mind it is easy to accidentally fall for, or spread, misinformation. Unfortunately, this becomes an issue as many people has shifted to using LLMs instead of a conventional search engine. An Adobe survey of 1000 people in the U.S. found that around 77% have used LLMs — specifically ChatGPT — as search engines, and around 24% of respondents used LLMs first before conventional search engines [39]. Additionally, around 30% trusted the response from ChatGPT more than the ones from conventional search engines.

These hallucinations occur due to that the models are generally trained at predicting the next, or a missing, token and then getting human feedback on whether the output is a "good response". No part in training chain actually requires that the statements made are true, but rather emphasises that the responses are acceptable to the human giving feedback.

Additionally, generative models can be used to generate text or images explicitly intended to spread misinformation, such as fabricated news stories, or reports and papers [111, p. 20 & 21]. As the models gets better, it will be even more difficult to separate the actual information from one generated by AI. Moreover, just the presence of deep fakes or fake news can, and already has [27], be used to discredit real and factual news. A study in California Law

Review in 2019 called this *the Liar's Dividend* [27] as this can be used to avoid accountability by claiming that any evidence is generated by AI models. More research needs to really know the impact of this, additionally society at large needs to raise peoples AI-litteracy and make everyone aware of the potential for misuse — both intentional and unintentional.

Inequalities and consolidation of resources The general market for LLMs are already small with some of the main providers being OpenAI (the ChatGPT models), Google (the Gemini models), Anthropic (the Claude models), xAI (the Grok models), and DeepSeek (the DeepSeek models). Additionally, a lot of these companies use user input, and other interactions with their models, to obtain more training data and thus further improve their models. This, combined with the fact that it requires significant resources to train a competitive model from the ground up [98, p. 62] sets the stage for these companies to maintain their position by, for example, buying all start-ups that seem competitive. Hence this could result in a consolidation of resources to these companies [111, p. 22].

In addition to this, countries with successful AI companies are likely to get a large advantage over countries lacking these tools leading to inequalities and consolidation of power between countries [111, p. 25].

Making these models open source would reduce this risk but would increase the risk of other entities modifying the models for nefarious use cases, so it is not clear cut how to reduce this centralisation of power without exposing other risks.

Alignment and agentic AI One goal people often cite is to, at some point, obtain so called *agentic AI*. This is a fuzzy, speculative, field and many people have strong opinions making it divisive [111, p. 19, 34, &42]. Roughly, agentic AI means an algorithm that can make decisions, planning, and work towards a goal without further — or with minimal — further human instruction or guidance. These could be either fully digital or connected to robots for interaction with the physical world. This would be very helpful in many cases, from home assistance robots and autonomous robots working at disaster scenes to fully functioning AI assistants checking in with the human on what administrative tasks they can assist with. Obviously this range includes putting agentic AI into offensive scenarios, from digital and physical attacks to subterfuge and espionage.

Any use of agentic AI requires research and development of *alignment* methods, see [68] and [111, p. 34] for general discussions. An AI model is considered

aligned if it works towards the intended goals without deviating to unintended, either instrumental or terminal, goals. Along these lines, researchers has observed deceitful behaviour in the chain of thought of certain models [59, 102, 56], and OpenAI have detected instances of models modifying tasks and code to fool the evaluation and get an easier task [11]. The exact cause of these events is not known, e.g. if — or to what extent — they occur due to texts about this being present in the training data or due to other causes. In any case, developing AI, both agentic and not, too hastily could result in negative consequences and care should be taken to avoid this [111, p. 21]. This a field of research that requires more attention.

Part II

Mathematical foundations

3 Mathematical background

In this chapter I will present the needed mathematical background to read **Paper I** and **Paper IV**. Readers with very familiar with group and representation theory as well as differential geometry can skip this chapter and revisit it later if needed.

3.1 Group and representation theory

For more details on group and representation theory see [127, 72, 67].

The study of symmetry in mathematics is encompassed in the field of *group theory* where the main subject is the *group*.

Definition 3.1.1 (Group)**.** A *group* is a pair $(G, \odot)$ where G is a set and $\odot : G \times G \to G$ is a map such that

1. there exists an element $e \in G$ for which $e \odot g = g \odot e = g$ for all elements $g \in G$;
2. for all elements $g \in G$ there exists an element denoted g^{-1} such that $g \odot g^{-1} = g^{-1} \odot g = e$; and
3. for all elements $g, k, h \in G$ it holds that $g \odot (k \odot h) = (g \odot k) \odot h$.

If the map $\odot$ is symmetric, that is $g \odot k = k \odot g$ for all elements $g, k \in G$ then the group is called *abelian*. An operation with this property is called *commutative*. Generally one drops the $\odot$ and simply writes $gk := g \odot k$.

If two sets G and H are such that H is a subset of G, and both are groups with the same group operation, then H is called a *subgroup* of G and this is denoted $H \leqslant G$.

Example 3.1.3: The group $\mathrm{GL}(n, \mathbb{R})$

One clear example of a group is the set of $n \times n$ invertible real matrices with the standard matrix product as the group product. This group is typically denotes $\mathrm{GL}(n, \mathbb{R})$, but if the real part is understood from context one generally just writes $\mathrm{GL}(n)$.

Remark 3.1.2. There is a special type of group called a *Lie group*, see definition 3.2.4. In this thesis, unless otherwise stated all groups are assumed to be Lie groups.

Letting groups acting on objects and spaces is a powerful notion.

Definition 3.1.4 (Group action on a space)**.** Given a group G and a space X the left (right) action of G on X is a map $\triangleright : G \times X \to X$ (or for a right action $\triangleleft : X \times G \to X$) satisfying that

1. for all elements $x \in X$ it holds that $e \triangleright x = x$;
2. for all elements $g, k \in G$ and $x \in X$ it holds $g \triangleright (k \triangleright x) = (gk) \triangleright x$;

and the corresponding properties for the right action. The space X with a left (right) G-action is called a *(left) G-space X*. When the action is obvious I drop the explicit $\triangleright$ and just write gx for $g \in G$ and $x \in X$.

With the action of a group G on a space X this allows the fundamental concept of an *G-orbit of a point $x \in X$*. The G-orbit of x is the set of points $\{g \triangleright x \mid g \in G\}$, that is, all the points x can be moved to by elements of G.

A more abstract object that will be useful later for (bi)principal bundles it the torsor.

Definition 3.1.5 (Torsor)**.** Given a group H, a left (right) H-space X is a left (right) torsor if the action of H on X satisfies that

1. H acts freely: for each $h \in H$ it holds that $\forall z \in X$, $h \triangleright x = x$ implies that $h = e$.
2. H acts transitively: $\forall x, y \in X$ there exists $h \in H$ such that $y = h \triangleright x$.

A right torsor H-torsor is defined identically, but with a right action.

Example 3.1.6: Symmetry as a group

The set of transformations leaving an equilateral triangle, see fig. 3.1 below, the same are rotations by 120° and reflections about the symmetry lines. To be explicit, if we label the corners as x, y, and z, a rotation of 120° maps the corners $x \mapsto y \mapsto z \mapsto x$ while a 240° maps $x \mapsto z \mapsto y \mapsto x$. Hence, if we do not label the corners we see that the triangle is left the same after rotating by 0°, 120° or 240°. Due to this, any combination of these rotations will also leave the triangle the same. As for the reflections, if we reflect the triangle around l_x we see that $x \mapsto x$, $y \mapsto z$, and $z \mapsto y$. If we again drop the labels the triangle is the same after the reflection. Similar things hold for reflecting about l_y and l_z. Hence, setting G as the set $\{0°, 120°, 240°, l_x, l_y, l_z\}$ and defining $g \odot k := g \circ k$ as doing transformation k first and then g then one can see that this will result in a transformation in G. Or expressed differently, the *composition* of two symmetry preserving actions is a symmetry preserving action.

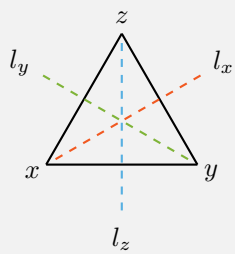


Figure 3.1: Figure displaying symmetry of triangle.

The symmetry of an object is generally described as a group. In that case the set G is the set of transformations keeping the object the same and the map $\odot : G \times G \to G$ is the composition of these transformations.

The above example deals with a group, the set of symmetry transformations, acting on an object, the triangle.

There are two important special cases of G-spaces: *homogeneous spaces*, see definition 3.1.7 and example 3.1.11, and G-actions on vector spaces through a *representation*, see definition 3.1.9 and example 3.1.12.

Definition 3.1.7 (Homogeneous space)**.** A G-space X is called *homogeneous* if for all $x, y \in X$ there exists a $g \in G$ such that $y = gx$. That is, the group G

acts transitively on X.

Remark 3.1.8. Using the orbit nomenclature, if X is a homogeneous G space this means that the G-orbit of any point $x \in X$ is the entire space X. In words this means that one can get to any point from any point using the group action.

For examples of homogeneous spaces, see example 3.1.11.

The other interesting special case of a G-space is when the underlying space is a *vector space*. In that case the group G acts on the vector space X through a *representation*.

Definition 3.1.9 (Representation)**.** A *representation* of a group G is a pair (ρ_V, V_ρ) of a vector space V_ρ and a map $\rho_V : G \to \mathrm{Aut}(V_\rho)$, where $\mathrm{Aut}(V_\rho)$ is the set of linear invertible maps $V_\rho \to V_\rho$. The map ρ_V needs to be a *homomorphism*, that is

$$\rho_V(gk) = \rho_V(g) \circ \rho_V(k), \quad \forall g, k \in G. \tag{3.1}$$

If the map or the space is obvious from context either can be referred to as a representation where the other element is implicit. Additionally, if there is no risk of confusion I will drop the subscripts of either the map, the vector space, or both.

To see some examples of representations, see example 3.1.12.

Remark 3.1.10. In the case that the vector space is finite dimensional $V \cong \mathbb{R}^n$ then $\mathrm{Aut}(V)$ is isomorphic to $\mathrm{GL}(n)$, that is, the space of invertible $n \times n$ matrices. Hence any group element g can, in this context, be represented as an invertible $n \times n$ matrix $\rho(g)$ when acting on vectors in V. To ease notation, I will denote the action of g on v as $\rho(g)v$ instead of $\rho(g)(v)$.

Note that any given group can act on different spaces, and even with a specific group and space there are multiple different representations specifying exactly how the group acts. In example 3.1.12 I provide some examples of different representations of groups.

The interesting part of this definition is that the map ρ needs to be a homomorphism. For a given vector space V this restricts the possible representations and, additionally, it opens the possibility to study special subspaces, so called *invariant subspaces*. Intuitively, a subspace is invariant if any vector in the subspace stays in the subspace when acted on by G.

Example 3.1.11: Some examples of homogeneous spaces

Here are some examples of homogeneous spaces which will come into play later in this thesis.

- *The plane:* The plane $\mathbb{R}^2$ is a homogeneous space under the group of two-dimensional translations. Specifically, a translation $z \in \mathbb{R}^2$ acts on $x \in \mathbb{R}^2$ by $z \triangleright x = x + z$. Hence, if $x, y \in \mathbb{R}^2$ then we know that there exists some translation z such that $y = x + z = z \triangleright x$.
- *The sphere:* The (two-dimensional) sphere S^2 is a homogeneous space under the group $SO(3)$ which is the group of three dimensional rotations. This group acts on points on the sphere by rotating while keeping the distance to the origin the same.
- *The space G/H:* This is the general form of a homogeneous space, and any homogeneous space can be viewed through this lens. If G is a group and $H \subset G$ is a subgroup then the coset space G/H is a homogeneous space under the group G. The coset space is the space of all points of the form $[g] := gH = \{gh \mid h \in H\}$ on which G acts as $k[g] = [kg]$. The $[g]$ notation is the *equivalence class* of g in that two group elements g, g' are considered the same if they are related by the (right) action of some $h \in H$. That is, $gh = g'$ for some $h \in H$.

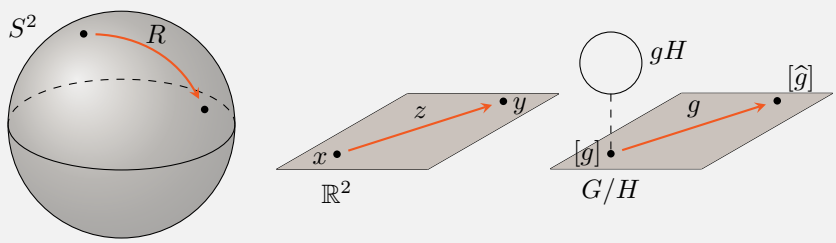


Figure 3.2: *Left panel:* Visualisation of the sphere with a rotation $R \in SO(3)$ linking two points. *Middle panel:* Visualisation of the plane with a translation $z \in \mathbb{R}^2$ linking two points. *Right panel:* Visualisation of the homogeneous space G/H and a group element linking two points, as well as the coset $[g] = gH$. Note that G/H is generally not flat, it is only shown as flat here for convenience.

Example 3.1.12: Some examples of representations

A representation of planar rotations: When the group of planar rotations act on the space of planar vectors $V = \mathbb{R}^2$ a rotation by an angle θ is generally mapped to the matrix

$$\rho(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}. \tag{3.2}$$

The trivial representation: If V is any vector space and G is any group the trivial representation $\rho : G \to \mathrm{Aut}(V)$ is defined as

$$\rho(g)(v) = v. \tag{3.3}$$

That is, this representation leaves all vectors unchanged.

Regular representation: If we have a vector space V of functions $f : X \to Y$ where X is a left G-space then the left regular representation of G allows G to act on functions through

$$[\rho_L(g)(f)](x) = f(g^{-1} \triangleright x). \tag{3.4}$$

In the case that X is a right G-space the corresponding right regular representation is

$$[\rho_R(g)(f)](x) = f(x \triangleleft g). \tag{3.5}$$

Since the left regular representation is very ubiquitous I will often just denote it $[gf]$, or possibly $[g \triangleright f]$, instead of $[\rho_L(g)f]$.

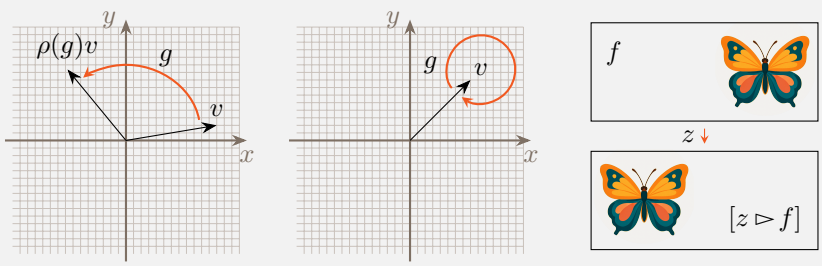


Figure 3.3: *Left panel:* the group of planar rotations acting on a planar vector through the representation in eq. (3.2). *Middle panel:* the same group of planar rotations, or any other group, acting on a planar vector through the trivial representation. *Right panel:* the group of translations on the plane acting on a function $f : \mathbb{R}^2 \to \mathbb{R}^3$, that assigns an RGB-value to each point, through the left regular representation $[z \triangleright f](x) = f(x - z)$.

Definition 3.1.13 (Invariant subspace)**.** Given a G-representation (ρ, V) a subspace $W \subset V$ is called an *invariant subspace* if

$$\rho(g)w \in W \quad \forall w \in W,\ g \in G. \tag{3.6}$$

All representations have two trivial invariant subspaces: the set containing the origin $\{0\}$ and the entire vector space V. This begs the natural question: for a given representation, how many non-trivial invariant subspaces are there and what are they? Representations with no non-trivial subspaces are special and are called *irreducible*.

Definition 3.1.14 (Irreducible and reducible representations)**.** A representation V is *irreducible*, often called an *irrep*, if the only invariant subspaces are $\{0\}$ and V, else it is called *reducible*.

For an example of a decomposition of a representation into irreps, see example 3.1.19.

As was mentioned in example 3.1.12 groups can act on functions. There is a way to construct a function space from a group action on a vector space.

Definition 3.1.15 (Induced representation)**.** Given a subgroup H to a group G and a H-representation (ρ, V). Then the *induced representation* is the function space

$$\mathrm{Ind}_H^G \rho := \left\{ f : G \to V \mid f(gh) = \rho(h^{-1})f(g),\ \forall h \in H,\ g \in G \right\}, \tag{3.7}$$

equipped with the left regular G-representation

$$[gf](k) = f(g^{-1}k). \tag{3.8}$$

For brevity, if the spaces are clear from context, we will denote $\mathrm{Ind}_H^G \rho$ by $\mathcal{I}_\rho$.

Remark 3.1.16. The constraint $f(gh) = \rho(h^{-1})f(g)$ introduced in eq. (3.7) is called the *Mackey constraint* or *Mackey condition*.

This construction will be relevant later as this is how we will treat our data mathematically.

Another definition that will be relevant is the definition *equivariance*.

Definition 3.1.17 (Equivariance)**.** Given two G-spaces X and Y with actions $\triangleright_X$ and $\triangleright_Y$ respectively. Then a map $T : X \to Y$ is *equivariant* if for all $x \in X$ and $g \in G$

$$T(g \triangleright_X x) = g \triangleright_Y T(x). \tag{3.9}$$

Remark 3.1.18. A special case of an equivariant map is when the domain and codomain are representations, so that $g \triangleright_X x = \rho_X(g)x$ and similarly for Y, and T is linear. In that case T is called an *intertwiner*.

Example 3.1.19: Example of irrep

To see a representations decomposed into irreps let $(\rho, \mathbb{R}^3)$ be the representation of planar rotations on three-dimensional vectors by rotations around the z-axis. Here there are two invariant subspaces: the xy-plane and the z-axis. The plane is invariant as the rotations keeps all vectors in plane, and the z-axis is invariant as no vectors on the z-axis are affected by the rotation, see fig. 3.4. Specifically, this representation is given by the map

$$\theta \mapsto \rho(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.10}$$

This allow us to decompose the representation $(\rho, \mathbb{R}^3)$ into

$$\rho = \rho_{\mathbb{R}^2} \oplus \mathrm{id}_{\mathbb{R}}, \qquad \mathbb{R}^3 = \mathbb{R}^2 \oplus \mathbb{R}, \tag{3.11}$$

where $\rho_{\mathbb{R}^2}$ acts on the xy-parts of vectors by the standard rotation matrix and id_R acts on the z-component and leaving it as it is.

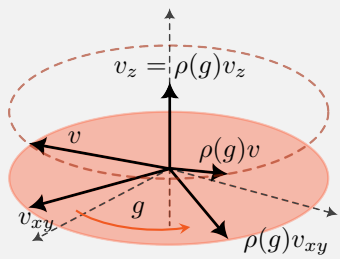


Figure 3.4: Planar rotations acting on vectors in $\mathbb{R}^3$ leaving the xy-plane and the z-axis invariant. Note that a vector v that lies outside either the xy-plane or the z-axis is not mapped back into the same non-trivial subspace.

3.2 Differential geometry and gauge theory

For more details on differential geometry, fibre bundles, and gauge theory see [61, 86, 145, 74, 73, 122].

Another relevant field is *differential geometry* which deals with calculus on, potentially, curved spaces. This formalism is what will allow us to talk about data on curved spaces.

The central object in differential geometry is the *manifold* which will be the spaces our data is defined on.

Definition 3.2.1 (Manifold)**.** An n-dimensional *manifold* is a space $\mathcal{M}$ such that for each point $x \in \mathcal{M}$ there exists an open neighbourhood U of x which is homeomorphic to an open subset $\widehat{U}$ of $\mathbb{R}^n$.

Remark 3.2.2. There are more technical assumptions on a manifold: it is assumed to be Hausdorff and second countable.

The intuition for a n-dimensional manifold is that it is an object that "locally looks like" $\mathbb{R}^n$. For an open set $U \subset \mathcal{M}$ which is mapped homeomorphically to $\varphi_U(U) \subset \mathbb{R}^n$ the pair (U, φ_U) is called a *coordinate chart*. As multiple charts can cover the same area on the manifold we need to know how these overlapping charts relate to each other.

Let (U_A, φ_A) and (U_B, φ_B) be two coordinate charts such that $U_A \cap U_B \neq \varnothing$, then the *transition map* is the map $\varphi_A \circ \varphi_B^{-1} : \varphi_B(U_A \cap U_B) \to \varphi_A(U_A \cap U_B)$. Explicitly, going through each step:

$$\mathbb{R}^n \supset \varphi_B(U_A \cap U_B) \xrightarrow{\varphi_B^{-1}} U_A \cap U_B \xrightarrow{\varphi_A} \varphi_A(U_A \cap U_B) \subset \mathbb{R}^n. \tag{3.12}$$

That is, $\varphi_A \circ \varphi_B^{-1}$ changes the coordinates of the overlapping area $U_A \cap U_B$ from the coordinates (given by) φ_B to the coordinates (given by) φ_A. Note that the transition map $\varphi_{AB} := \varphi_A \circ \varphi_B^{-1}$ is a map from (an open subset of) $\mathbb{R}^n$ to (an open subset of) $\mathbb{R}^n$, this means that one can view $\varphi_A \circ \varphi_B^{-1}$ as a standard multi variable function.

Now we want to be able to assign local coordinates in a way that we can cover $\mathcal{M}$ such that every point has, at least one, coordinate chart including that point. Any such collection of coordinate charts $\mathcal{A} = \{(U_a, \varphi_a)\}_{a \in I}$, where I is an index set, such that $\bigcup_{a \in I} U_a = \mathcal{M}$ is called an *atlas*. If all transition functions $\varphi_{ab} := \varphi_a \circ \varphi_b^{-1}$ are smooth, as functions $\mathbb{R}^n$ to $\mathbb{R}^n$, then the manifold $\mathcal{M}$ is a *smooth manifold*. For an example of the circle as both a smooth and non-smooth manifold, see example 3.2.6.

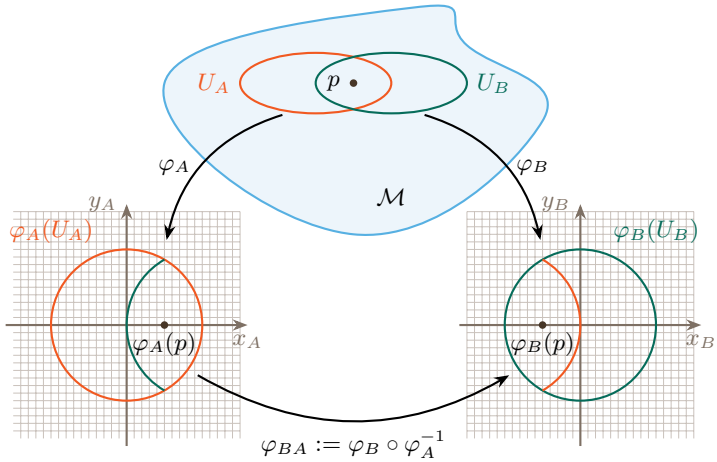


Figure 3.5: Visualisation of two coordinate charts (U_A, φ_A) and (U_B, φ_B) and a transition map showing the change of local coordinates in an open set around $p \in U \subset U_A \cap U_B \subset \mathcal{M}$.

Smooth manifolds allow us to formalise the concepts of *tangent vectors*. Namely, let $p \in U \subset \mathcal{M}$ be a point on the manifold such that (U, φ_U) be a local chart and let $\gamma : I \to \mathcal{M}$ — where $I \subset \mathbb{R}$ is an open set containing 0 — be a smooth curve in $\mathcal{M}$ such that $\gamma(0) = p$. This means that the composition $\varphi_U \circ \gamma$ is a map from $\mathbb{R}$ to $\mathbb{R}^n$ and the derivative $(\varphi_U \circ \gamma)'(0)$ is a vector in $\mathbb{R}^n$. Now, there are many curves that result in the same vector, hence we group all curves generating the same vector at p into equivalence classes $[\gamma]_p$. Specifically, if γ_1 and γ_2 are two curves passing through p at $t = 0$ we let

$$\gamma_1 \sim \gamma_2 \qquad \text{iff} \qquad (\varphi_U \circ \gamma_1)'(0) = (\varphi_U \circ \gamma_2)'(0). \tag{3.13}$$

With this we get a one-to-one mapping between vectors X_p and equivalence classes of curves $[\gamma]_p$. Each equivalence class at p is then a tangent vector to $\mathcal{M}$ at p.

Using this structure one tend to define an action of tangent vectors on smooth functions $f : \mathcal{M} \to V$ as

$$(X_p)(f) = \frac{d}{dt}(f \circ \gamma)(t)\bigg|_{t=0}, \tag{3.14}$$

where γ is any curve from the equivalence class of X_p.

Additionally, an important concept for smooth manifolds are *diffeomorphisms*.

A diffeomorphism $F : \mathcal{M} \to \mathcal{N}$ is an invertible smooth map between two smooth manifolds $\mathcal{M}$ and $\mathcal{N}$ such that the inverse is smooth as well. The set of diffeomorphisms from a manifold $\mathcal{M}$ to itself form a group under composition denoted $\mathrm{Diff}(\mathcal{M})$.

Not only points can be moved by diffeomorphisms, but tangent vectors can be pushed forward and functions can be pulled back.

Definition 3.2.3 (Pushforward and pullback)**.** Let $F : \mathcal{M} \to \mathcal{N}$ be a diffeomorphism between two smooth manifolds.

The differential of F, denoted dF is a map $dF : T\mathcal{M} \to T\mathcal{N}$ between the tangent bundles of $\mathcal{M}$ and $\mathcal{N}$. The *pushforward* of $X_p \in T_p\mathcal{M}$ by F is denoted $(dF)_p(X_p)$ and acts on a smooth function $f_\mathcal{N} : \mathcal{N} \to V$ through

$$(dF)_p(X_p)(f_\mathcal{N}) = \frac{d}{dt}(f_\mathcal{N} \circ F \circ \gamma)(t)\Big|_{t=0}. \tag{3.15}$$

The *pullback* of a function $f_\mathcal{N} : \mathcal{N} \to V$ by F, denoted F^*f is a map $F^*f : \mathcal{M} \to V$ defined as

$$(F^*f)(p) = (f \circ F)(p). \tag{3.16}$$

A very special smooth manifold is the Lie group which is used to describe essentially all continuous, or smooth, transformations.

Definition 3.2.4 (Lie group)**.** A *Lie group* G is a group which is also a smooth manifold such that the group multiplication

$$m : G \times G \to G, \qquad m(k, g) = kg, \tag{3.17}$$

and the inverse

$$\iota : G \to G, \qquad \iota(g) = g^{-1}, \tag{3.18}$$

are smooth maps for all $k, g \in G$.

Example 3.2.5: Example of Lie groups

A simple example of a Lie group is the group of planar rotations, or the group of translations of the plane.

As a Lie group G is a smooth manifold it makes sense to consider tangent vectors and tangent spaces. Specifically, we can identify the tangent space T_eG with the so-called Lie algebra $\mathfrak{g}$.The Lie algebra $\mathfrak{g}$ is a vector space and can intuitively be thought of as encoding the structure of the group G where

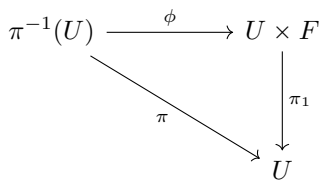
the tangent vectors $X \in \mathfrak{g}$ correspond to infinitesimal transformations. To obtain the tangent space at an other $g \in G$ note that $L_g := m(g, \bullet) : G \to G$ is a diffeomorphism and hence we can get T_gG at any $g \in G$ through the pushforward of T_eG. Specifically

$$T_gG = (dL_g)_e(T_eG) = (dL_g)(\mathfrak{g}), \tag{3.19}$$

as we can identify $\mathfrak{g}$ with T_eG. One important point is that there exists a map $\exp : \mathfrak{g} \to G$ which intuitively integrates an infinitesimal transformation X to obtain a finite transformation $g \in G$. This will be important when we discuss connections.

In order to discuss data on manifolds we also need the concepts of a *fibre bundle*. Intuitively, a fibre bundle is a construction that allows to attach a copy of a certain space to each point of another space.

Definition 3.2.7 (Fibre bundle)**.** A *fibre bundle* is a tuple $(E, \pi, \mathcal{M}, F)$ where E is the *total space*, $\mathcal{M}$ is the *base space* — often taken to be a smooth manifold — the map $\pi : E \to \mathcal{M}$ is a continuous surjective map called the projection, and F is a space called the *typical fibre*. The tuple needs to satisfy that for all $x \in \mathcal{M}$ we have $\pi^{-1}(\{x\}) =: F_x \cong F$ and that it is *locally trivial*. Locally trivial means: for all open set U in the atlas of $\mathcal{M}$ there exists a diffeomorphism $\phi : \pi^{-1}(U) \to U \times F$ satisfying that $\pi_1 \circ \phi = \pi$. Graphically this means that the following diagram commutes



This is visualised in fig. 3.7. Often a bundle is denoted simply by the total space E if the other components are obvious from context, or as $\pi : E \to \mathcal{M}$. If needed, the typical fibre is sometimes denoted as a subscript to the total space: E_F.

Intuitively, one can view ϕ as assigning coordinates to the fibre bundle locally, and as a manifold locally looks like $\mathbb{R}^n$ a fibre bundle locally looks like $U \times F \cong \widehat{U} \times F$ where $\widehat{U}$ is a subset of $\mathbb{R}^n$. Hence, in the case that F is a finite dimensional vector space we can through a local trivialisation ϕ assign coordinates $\phi(p) = (x^i, y^\alpha)$. Here x^i, $i = 1, \ldots, n$ are coordinates in the base space, and y^α, $\alpha = 1, \ldots, \dim(F)$, are coordinates in the fibre. The direction along the fibre is called the *vertical direction* and all other directions are called *horizontal*. This

Example 3.2.6: Smooth and non-smooth manifold

The smoothness of a manifold is entirely dependent on the compatibility of the atlas of coordinate charts. To illustrate this let our manifold $\mathcal{M}$ be the circle S^1 and lets equip this with two different atlases, first a smooth atlas and then a non-smooth atlas.
Smooth atlas: Define the four charts below all mapping the open set $(-1, 1)$ to the circle.

$$\begin{aligned} \varphi_1 &= \text{project top half } U_1 \text{ onto the } x \text{ axis} \\ \varphi_2 &= \text{project bottom half } U_2 \text{ onto the } x \text{ axis} \\ \varphi_3 &= \text{project right half } U_3 \text{ onto the } y \text{ axis} \\ \varphi_4 &= \text{project left half } U_4 \text{ onto the } y \text{ axis} \end{aligned} \tag{3.20}$$

To check that this is a smooth atlas we need to check that wherever two charts overlap, $U_i \cap U_j \neq \varnothing$, the transition map $\varphi_{ij} = \varphi_i \circ \varphi_j^{-1}$ is a smooth map from $(-1, 1) \subset \mathbb{R}$ to $(-1, 1) \subset \mathbb{R}$. For example, U_1 and U_3 overlap in the first quadrant and through these maps we get that the transition function from chart 1 to chart 3 is

$$\varphi_{31}(x) = \sqrt{1 - x^2}, \tag{3.21}$$

which is a smooth function on $(-1, 1)$.

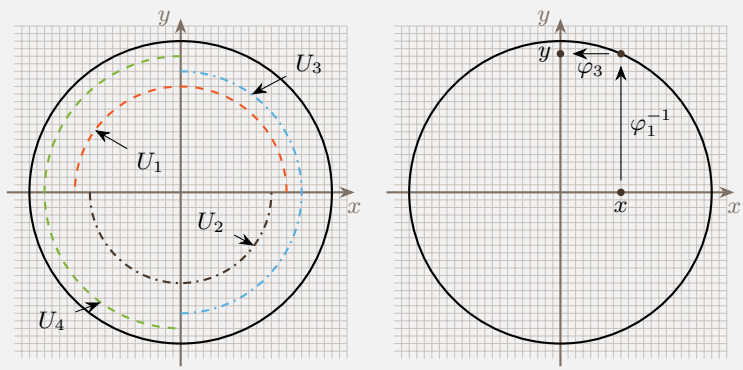


Figure 3.6: Visualisation of the four open sets $U_1, \ldots, U_4$ on S^1 used in the charts defined in eq. (3.20).
Non-smooth atlas: If we want to make this into a non-smooth atlas we can simply compose all of the charts $\varphi_1, \ldots, \varphi_4$ with the non-smooth map

$$h(t) = \begin{cases} t, & t < 0 \\ 2t, & t \geqslant 0. \end{cases} \tag{3.22}$$

This ensures that all charts $\varphi_1, \ldots, \varphi_4$ are still invertible, now mapping into $(-1, 2)$, and that their compositions $\varphi_{ij} = \varphi_i \circ \varphi_j^{-1}$ are non-differentiable at 0.

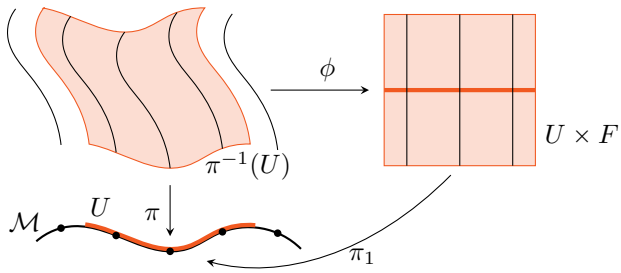


Figure 3.7: A fibre bundle $\pi : E \to \mathcal{M}$ with an open set $U \subset \mathcal{M}$ and a local trivialisation of $\pi^{-1}(U)$.

is visualised in fig. 3.7. Note that the bundle is only locally trivial, meaning that it locally looks like a simple product $U \times F$ while globally having a more complicated structure. A *trivial bundle* is a bundle where $E = \mathcal{M} \times F$.

Remark 3.2.8. Technically, a local trivialisation $\phi : \pi^{-1}(U) \to U \times F$ does not assign any coordinates on either the manifold or the fibre. However, a chart (U, φ_U) of local coordinates on the manifold can be used to assign coordinates to the manifold part of the local trivialisation. To achieve coordinates on the fibre one must additionally choose a basis e_α for the typical fibre. The full chain to achieve coordinates on both the manifold and the fibre is then for a point $p \in \pi^{-1}(U) \subset E$

$$p \xrightarrow{\phi} (x, y) \overset{(\varphi, \mathrm{id}_F)}{\longrightarrow} (x^i, y) \overset{(\mathrm{id}_{\mathbb{R}^n}, e_\alpha)}{\longrightarrow} (x^i, y^\alpha). \tag{3.23}$$

Although this is good to keep in mind it is sometimes easier to think of a local trivialisation as assigning coordinates on the bundle and hence being the analogue to the coordinate charts on a manifold.

Definition 3.2.9 (Types of bundles)**.** The typical fibre used determines the type of bundle here are some quick definitions:

- *Vector bundle:* A vector bundle is a fibre bundle $(E, \pi, \mathcal{M}, V)$ where the typical fibre is a vector space.
- *Tangent vector bundle:* The tangent vector bundle $(T\mathcal{M}, \pi, \mathcal{M}, \mathbb{R}^n)$ is a bundle where the fibre at each point p is the tangent space $T_p\mathcal{M} \cong \mathbb{R}^n$.
- *Principal H-bundle:* A principal H-bundle is a bundle $(P, \pi, \mathcal{M}, H)$ where H is a Lie group and H acts freely and transitively on the fibre, typically from the right. Note that this means that each fibre is isomorphic to the group H but is not a group in itself, rather each fibre is a H-torsor.

Example 3.2.11: Examples of bundles

- *Vector bundle:* A trivial example of a vector bundle is that $\mathbb{R}^3$ can be viewed as a vector bundle $(\mathbb{R}^3, \pi, \mathbb{R}, \mathbb{R}^2)$ where we attach a copy of $\mathbb{R}^2$ to each point along the base space $\mathbb{R}$.
- *Tangent vector bundle:* If the base manifold is S^2, the sphere, then each tangent space T_pS^2 would be the tangent plane to the sphere at p. The tangent bundle TS^2 is then the collection of all tangent planes.
- *Principal H-bundle:* Let the base space be $\mathbb{R}^3$ and the fibre at each point be a (right) GL(3)-torsor. This forms a (right) principal GL(3)-bundle. Intuitively one can think of the fibre over a point p as all possible coordinate frames for three-dimensional vectors at p.
- *Associated vector bundle:* Let the principal bundle P be the bundle just introduced and let $V = \mathbb{R}^3$ be the normal representation of GL(3). Then each for each element $[p, v]$ in the total space $P \times_\rho V$ one can view v as being the coordinates of the geometric vector, or coordinate free vector, v in the basis p. The invariance to the action of GL(3) enforced by the associated vector bundle then means that the equivalence class $[p, v]$ represents the geometric vector itself.

- *Associated (vector) bundle:* Given a left H-space V and a principal H-bundle P the associated bundle is constructed from the space $P \times V$ where one then identifies points under the H-action:

$$(p, v) \sim (p', v') \quad \text{iff} \quad \exists h \in H : p = p' \triangleleft h,\ v = h^{-1} \triangleright v'. \tag{3.24}$$

  This means that in each equivalence class we have $[p \triangleleft h, v] = [p, h \triangleright v]$. To denote that we have imposed this H-equivalence to the total space we denote the total space by $P \times_H V$. In the case that V is a vector space with a representation ρ, then we denote the total space by $P \times_\rho V$ and the bundle is called a associated vector bundle.

Remark 3.2.10. Technically one can define a principal H-bundle for any group — that is, H need not be a Lie group — but here we will assume that all principal bundles use Lie groups.

For any vector bundle a local trivialisation yields two sets of coordinates — one set for the base space and one set for the fibre — and hence it is natural to think about coordinate transformations on each set of coordinates. Coordinates in the base space can be affected in two ways: *actively* or *passively*. An *active transformation* on a manifold $\mathcal{M}$ is performed with a diffeomorphism $\phi : \mathcal{M} \to \mathcal{M}$ actively moving points around on the manifold. Hence, assuming a chart (U_a, φ_a) then a point $x \in U_a$ would have coordinates $\varphi_a(x) \in \mathbb{R}^n$ and if $x' = \phi(x) \in U_a$ then the moved point x' would have coordinates $\varphi_a(x')$ *in the same choice of coordinates*. A *passive transformation* on a manifold $\mathcal{M}$ is performed through a change of coordinates

$$\varphi_b(x) = \varphi_{ba}(\varphi_a(x)). \tag{3.25}$$

Of course, one could have that $\varphi_b(x) = \varphi_a(\phi(x))$ in which case it seems that both processes resulted in the same outcome, same coordinates, but one approach is *active* and the other is *passive*.

Specifically, for the spatial coordinates in a local trivialisation we get the following. For a fibre bundle $\pi : E \to \mathcal{M}$ with typical fibre F with coordinates φ_a on the subset $U_a \cap U_b$ will have local coordinates

$$(\varphi_a, \mathrm{id})\,(\phi(p)) = \left(x^i_a, y^\alpha\right). \tag{3.26}$$

Changing coordinates in the base manifold $\varphi_{ba} : \varphi_a \to \varphi_b$ will result in a purely horizontal change in the bundle coordinates:

$$(x^i_a, y^\alpha) \overset{(\varphi_{ba}, \mathrm{id})}{\longmapsto} (\varphi_{ba}(x^i_a), y^\alpha) = (x^i_b, y^\alpha). \tag{3.27}$$

Note that during this process, the point in the base manifold stays the same — just with different coordinates — and the coordinates y^α in the fibre are unchanged. This is visualised in fig. 3.8.

Importantly, the above discussion assumes we have coordinates in the fibre, this requires a choice of basis in the fibre. Let $\omega(x)$ be a choice of basis in the fibre over x with the α:th basis vector denoted $\omega_\alpha(x)$. With this any vector in that fibre can be expanded in coordinates y^α through $y = y^\alpha \omega_\alpha(x)$ (where any index repeated as a super- and a subscript is summed as by the *Einstein summation convention*). A change of coordinates in the fibre over x can be represented by an invertible matrix A such that the new coordinates and basis are

$$\begin{aligned} y^\alpha &\mapsto y^\beta = A^\beta{}_\alpha y^\alpha \\ \omega_\alpha(x) &\mapsto \omega_\beta(x) = (A^{-1})^\alpha{}_\beta \omega_\alpha(x), \end{aligned} \tag{3.28}$$

where $A^\alpha{}_\sigma (A^{-1})^\sigma{}_\beta = \delta^\alpha{}_\beta$ and $\delta^\alpha{}_\beta$ is 1 when $\alpha = \beta$ and 0 otherwise. Through

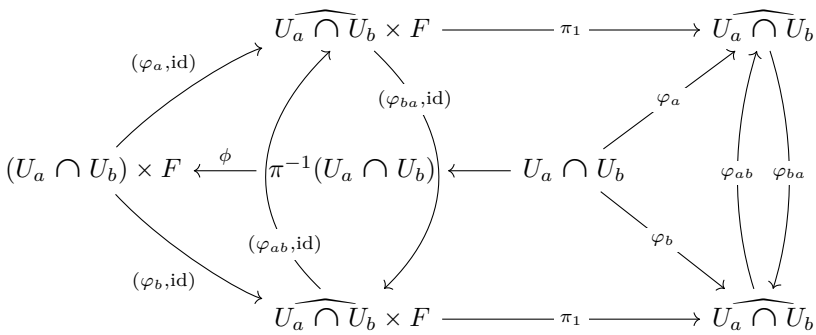


Figure 3.8: Diagram visualisation of the change of base manifold coordinates.

this we see that the vector y is independent of the choice of coordinates:

$$\begin{aligned} y &= y^\alpha \omega_\alpha(x) \\ &= A^\alpha{}_\beta y^\beta (A^{-1})^\sigma{}_\alpha \omega_\sigma(x) = (A^{-1})^\sigma{}_\alpha A^\alpha{}_\beta y^\beta \omega_\sigma(x) = \delta^\sigma{}_\beta y^\beta \omega_\sigma(x) \\ &= y^\beta \omega_\beta(x). \end{aligned} \tag{3.29}$$

The above was an example of a local change of coordinates in the fibre and this can be formalised through *gauge transformations*.

Definition 3.2.12 (Gauge transformation [62, p. 268])**.** Given a principal H-bundle $\pi : P \to \mathcal{M}$ a *gauge transformation* is a smooth local map $s : U \to H$ where $U \subset \mathcal{M}$ is an open subset of the base space. H is called the *gauge* or *structure group* and if the map s is constant over U it is called a *rigid gauge transformation*.

Example 3.2.13

Let F be a n-dimensional vector space, then the matrix A performing a change of coordinates is an invertible $n \times n$-matrix; i.e. an element of $\mathrm{GL}(n)$: the group of invertible $n \times n$-matrices. Hence the principal bundle in question is a principal $\mathrm{GL}(n)$-bundle and the gauge transformation is a local map $s : U \to \mathrm{GL}(n)$.

In the local coordinates from the local trivialisation above, and assuming a choice of fibre basis ω_α, we obtain coordinates for a point $p \in E$: $\phi(p) = (x^i, y^\alpha)$.

Performing a gauge transformation through the local map s then amounts to

$$(x^i, y^\alpha) \mapsto (x^i, s(\pi(p))^\beta{}_\alpha y^\alpha). \tag{3.30}$$

This is a vertical transformation as it only changes the coordinates in the fibre.

A different viewpoint on the gauge transformations are through the lens of local trivialisations. Since a local trivialisation maps $\pi^{-1}(U)$ to $U \times F$ two different trivialisations ϕ_a and ϕ_b can both trivialise $\pi^{-1}(U_a \cap U_b)$. Now, let $\phi_a : \pi^{-1}(U_a \cap U_b) \to (U_a \cap U_b) \times F_a$ and $\phi_b : \pi^{-1}(U_a \cap U_b) \to (U_a \cap U_b) \times F_b$, where $F_a = F_b = F$ are simply a relabelling of the typical fibre to ease bookkeeping. Hence we get the map

$$(U_a \cap U_b) \times F_a \xrightarrow{\phi_a^{-1}} \pi^{-1}(U_a \cap U_b) \xrightarrow{\phi_b} (U_a \cap U_b) \times F_b. \tag{3.31}$$

Letting $\phi_{ab} := \phi_a \circ \phi_b^{-1}$ we get that this map acts as

$$\phi_{ab}(x, y) = (x, t_{ab}(x)y) \tag{3.32}$$

where $t_{ab}(x) : F_b \to F_a$ and since F_a and F_b are the same space we get that $t_{ab}(x)$ is a fibre automorphism. Specifically, it needs to be a homeomorphism. The more common use case of this viewpoint is when the typical fibre F is a H-space, for some topological group H. Then the fibre automorphism $t_{ab}(x)$ is an element of the group H and t_{ab} can be viewed as a local smooth map $t_{ab} : U_a \cap U_b \to H$, which is the same thing as the gauge transformation defined above in definition 3.2.12. This is visualised in fig. 3.9.

The map t_{ab} generally needs to satisfy three conditions

$$t_{aa}(x) = 1 \tag{3.33}$$
$$t_{ab}(x) = t_{ba}(x)^{-1} \tag{3.34}$$
$$t_{ac}(x) = t_{ab}(x)t_{bc}(x) \tag{3.35}$$

where the third is known as the *cocycle condition*.

Depending on how the bundle is constructed, the active transformation by a diffeomorphism might induce a vertical transformation in addition to the horizontal transformations.

As I have hinted, we are going to view data as "living on a manifold". Specifically, the data will live locally in the fibres, and to formalise this we need the notion of a *section*.

Definition 3.2.14 (Section)**.** A (cross) section s of a bundle $\pi : E \to \mathcal{M}$ is a

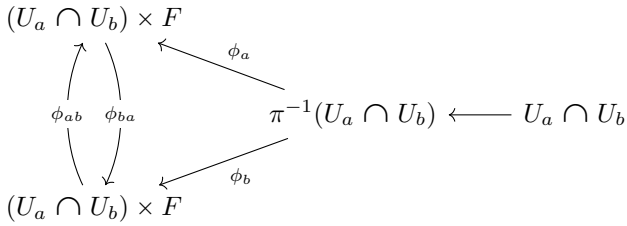


Figure 3.9: Fibre automorphism due to a change in local trivialisations.

Example 3.2.16: Local symmetry

A toy example of local, gauge, symmetries relating to physics is the following: Let $\pi : E \to \mathcal{M}$ be a vector bundle where the fibre is $\mathbb{C}$ and let P be a principal U(1)-bundle. That is, the group acts by multiplication by $\exp(i\theta)$ for θ in $[0, 2\pi)$.
Then, let $\phi \in \Gamma(E)$ be a section of E, and construct the object $\overline{\phi(x)}\phi(x)$. Transforming this locally through $\phi(x) \mapsto [s \triangleright \phi](x) = s(x)\phi(x)$ where s is a map $\mathcal{M} \to \mathrm{U}(1)$, we see that $\overline{\phi(x)}\phi(x)$ is invariant:

$$\overline{\phi(x)}\phi(x) \mapsto \overline{s(x)\phi(x)}s(x)\phi(x) = \overline{\phi(x)s(x)}s(x)\phi(x) = \overline{\phi(x)}\phi(x), \tag{3.36}$$

since $\overline{s(x)}s(x) = 1$. This shows that $\overline{\phi(x)}\phi(x)$ has a local U(1)-symmetry.

map $s : \mathcal{M} \to E$ satisfying $(\pi \circ s)(x) = \mathrm{id}_{\mathcal{M}}(x) = x$. The space of sections of the bundle E is denoted $\Gamma(E)$, and if the bundle is a vector bundle the space of sections is a vector space.

Remark 3.2.15. Depending on the type of bundle global continuous sections might exist, or they might not. For example, a non-trivial principal bundle has no continuous global sections.

Sections, and objects constructed from sections can have group symmetries on a local level (example 3.2.16), a global level (example 3.2.18), or both.

Whereas one can have a local structure without any additional constraints or actions on the base manifold $\mathcal{M}$, to get a global structure one must equip the manifold with a group action so that one can move points in $\mathcal{M}$ through this action. To formalise this we need *equivariant bundles*.

Example 3.2.18: Examples of data with global structure

An example of data with a global structure is vector fields on homogeneous spaces. A specific instance of this, under some assumptions, are wind fields on Earth. We can rotate the Earth through elements of SO(3) and as we do the vectors follow along and are rotated accordingly. Specifically, assuming $f : U \to \mathbb{R}^2$ assigns a tangent vector, horizontal wind, to each point $p \in U \subset S^2$, we get that this function is transformed as

$$[gf](p) = \rho(g)f(g^{-1}p), \tag{3.38}$$

where g rotates the points in U and $\rho(g)$ ensures that the tangent vector is rotated as well and not just moved. This is shown in fig. 3.10.

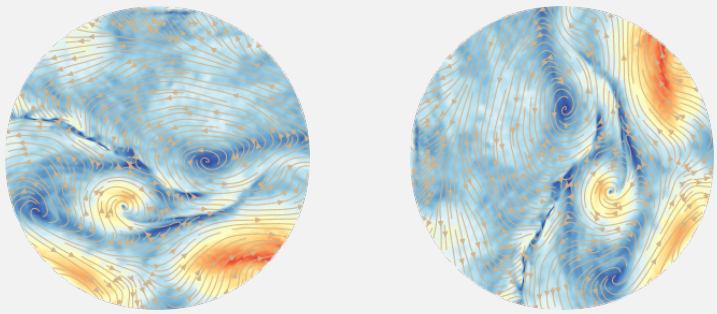


Figure 3.10: A rotation performed on wind data from 2023-01-07. Data form [101].

Definition 3.2.17 (Equivariant bundle)**.** An *equivariant bundle* is a bundle where both the total space and the base space are G-spaces and the projection is equivariant, i.e.

$$\pi \circ (g \rhd_E) = (g \rhd_{\mathcal{M}}) \circ \pi, \qquad \forall g \in G. \tag{3.37}$$

However, to work with these local sections one needs to choose local coordinates $(U_\alpha, \varphi_\alpha)$ and while doable, this is bothersome. What one can do instead is — if one is working with sections of an associated bundle — to lift the sections defined locally to global maps on a corresponding principal bundle.

Theorem 3.2.19 (Theorem §10.12 in [76])**.** *Let P be an principal H-bundle and $\mathcal{C}^\infty(P,S)^H$ be the space of smooth maps $f : P \to S$ satisfying the Mackey constraint $f(ph) = h^{-1} \rhd_S f(p)$ for all $p \in P$ and $h \in H$. Further, let $\mathcal{C}^\infty(P \times_H$*

S) be the space of smooth sections of the associated bundle $P \times_H S$. Then there is an isomorphism $\mathcal{C}^\infty(P,S)^H \cong \mathcal{C}^\infty(P \times_H S)$.

This allows to work with the lifted functions in a "coordinate free" way as the local coordinates are given through transformations by elements in H.

However, this is more powerful when considering equivariant bundles over homogeneous spaces. The homogeneous structure of the base space $X \cong G/H$ induces a group action on sections $f \in \Gamma(G \times_\rho V)$ of the associated vector bundle $G \times_\rho V$ through

$$[\rho_\Gamma(k)f](g) = \rho_{G\times_\rho V}(g) f(k^{-1}g), \tag{3.39}$$

where $\rho_{G\times_\rho V}$ acts on elements of the associated vector bundle as

$$\rho_{G\times_\rho V}(k)[g,v] = [kg,v]. \tag{3.40}$$

Now we can state a corollary to theorem 3.2.19.

Corollary 3.2.20. *The space of smooth sections of an associated vector bundle $\Gamma(G \times_\rho V)$ is isomorphic to the subspace of smooth functions in the induced representation $\mathrm{Ind}_H^{G,\infty} \rho$.*

In this way we are free to work with sections in local coordinates or with global elements of the induced representation. Due to the simplicity and the baked in symmetry of the Mackey constraint we will generally work in the lifted setting. This link between the induced representation and the sections can be visualised through the *Cartan mixing diagram*.

Definition 3.2.21 (Cartan mixing diagram [151, p. 33])**.** Given a principal H-bundle $\alpha : P \to \mathcal{M}$ and a left H-space X, Cartan's mixing diagram is the commutative diagram

$$\begin{array}{ccccc} P & \xleftarrow{\pi_1} & P \times X & \xrightarrow{\pi_2} & X \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \beta} & & \downarrow{\scriptstyle \gamma} \\ \mathcal{M} & \xleftarrow[\tau_\alpha]{} & P \times_H X & \xrightarrow[\tau_\beta]{} & H\backslash X \end{array} \tag{3.41}$$

where

$$\tau_\alpha([p,x]) = \alpha(p), \qquad \tau_\gamma([p,x]) = \gamma(x) = Hx$$

and β is the quotient map of $P \times X$ with respect to the diagonal right H-action

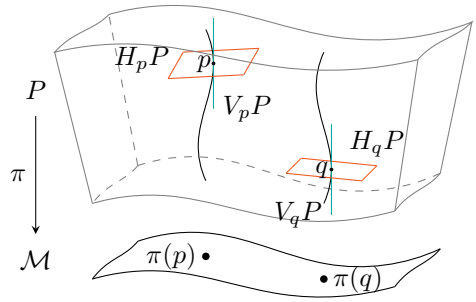


Figure 3.11: A visualisation of the vertical and a horizontal space in a principal bundle. Note that the vertical spaces are tangential to the fibres while the horizontal spaces are, without a connection, arbitrary.

$(p, x)h = (ph, h^{-1}x)$. Note that τ_γ is well-defined since all representatives (p, x) in the same equivalence class map to the same object Hx in $H\backslash X$. Moreover, by symmetry, if γ is a principal H-bundle, then τ_β is a fibre bundle with fibre P.

Several times I have emphasised the local nature of the fibre bundles, and for good reason: due to the potential curvature there is no a priori way to easily relate objects in different fibres — that is, over different points in the base manifold — to each other. Formulated differently, there is no clear way how to move objects between fibres. To achieve this we need the concept of how the fibres connect to each other, i.e. if you move in the base manifold, how would this motion in the base space induce a motion in the fibres of the total space. This concept is called the *connection* of the bundle or manifold, and the type of connection which is relevant depends on the bundle in question.

A connection is generally formulated in terms of *vertical* and *horizontal* subspaces.

Definition 3.2.22 (Vertical and horizontal subspace)**.** Given a bundle $\pi : E \to \mathcal{M}$ the *vertical subspace* $V_eE \subset T_eE$ at $e \in E$ is the kernel $\ker(d\pi_e)$ of the pushforward $d\pi : T_eE \to T_{\pi(e)}\mathcal{M}$. The *horizontal subspace* H_eE is any subspace of T_eE such that $H_eE \cap V_eE = \{0\}$ and

$$T_eE = V_eE \oplus H_eE. \tag{3.42}$$

Note that nothing in this definition states how one arrives at which horizontal subspace to choose, that is the purpose of the connection. I will now define a special formulation of connections that work for any smooth bundle but here

stated specifically for principal H-bundles.

Definition 3.2.23 (Ehresmann connection)**.** Let $\pi : P \to \mathcal{M}$ be a principal H-bundle where the right action of $h \in H$ is denoted $R_h : P \to P$. Then an *Ehresmann connection* is a choice of horizontal subspaces H_pP for each $p \in P$ such that the pushforward of the right action R_h maps horizontal spaces to horizontal spaces, that is

$$(dR_h)_p(H_pP) = H_{R_h(p)}P, \qquad \forall h \in H,\ p \in P. \tag{3.43}$$

We can be more specific and utilise that H is a Lie group. Specifically, as we have $\exp : \mathfrak{h} \to H$ we can generate vertical tangent vectors from Lie algebra elements: Let $A \in \mathfrak{h}$ be a Lie algebra element, then $(X^A)_p \in V_pP$ is a vertical tangent vector at p defined through

$$X^A_p := \frac{d}{dt}(p \triangleleft \exp(tA))\bigg|_{t=0}. \tag{3.44}$$

With this we can define the connection one-form.

Definition 3.2.24 (Connection one-form)**.** The *connection one-form* is a map $\omega_p : T_pP \to \mathfrak{h}$ satisfying

$$\omega_p(X^A_p) = A, \forall p \in P\ A \in \mathfrak{h}, \tag{3.45}$$

and

$$(R_h)^*\omega = \mathrm{Ad}_{h^{-1}} \circ \omega, \tag{3.46}$$

which when acting on $X_p \in T_pP$ expands to

$$\omega_{R_h(p)}((dR_h)_p(X_p)) = \mathrm{Ad}_{h^{-1}}(\omega_p(X_p)). \tag{3.47}$$

Equation (3.46) is the specific case of eq. (3.43). With such a map ω we can define the horizontal subspace as

$$H_pP := \ker(\omega_p). \tag{3.48}$$

Additionally, the presence of ω allows for a definition of a metric on the principal bundle P given that the base manifold $\mathcal{M}$ has a metric $g_{\mathcal{M}}$. The metric on P can be set to

$$g_P = \pi^* g_{\mathcal{M}} + k\omega, \tag{3.49}$$

such that

$$g_{P,p}(X_p, Y_p) = g_{\mathcal{M},\pi(p)}(d\pi_p(X_p), d\pi_p(Y_p)) + k(\omega_p(X_p), \omega_p(Y_p)). \tag{3.50}$$

Here k is an Ad invariant metric on $\mathfrak{h}$. This means that g_P is dR_h invariant:

$$dR_h(g_{P,p})(X_p, Y_p) = g_{P,R_h(p)}(dR_h(X_p), dR_h(Y_p)). \tag{3.51}$$

4 Differential geometry and equivariance

In this chapter I provide an improved version of the theory presented in chapter 2 of **Paper I**. Specifically, I lift the entire framework naturally to the total space of a (bi)principal bundle and this allows a direct connection between the lifted framework and the normal CNN as well as the group equivariant CNNs on homogeneous spaces [29].

It has been stated in several works that weight sharing and equivariance are deeply connected [159, 164, 30, 78]. Most of these works have approached this connection in different ways, from representation theory to the definition of a CNN. However, I find that taking the approach of differential geometry is very useful and as such I want to first show how weight sharing appears in a differential geometry context. Using this approach I will then present a convolutional-like layer which is equivariant to the fibre preserving action of a principal bundle.

4.1 Example: the ordinary CNN

A standard result is that a convolutional layer is translation equivariant. To formulate this mathematically, let $f : \mathbb{R}^2 \to V_{\text{in}}$ be our data — e.g., for an image $V_{\text{in}} = \mathbb{R}^3$ where the three numbers represent RGB values — and $[L_z f](x) = f(x - z)$ be the left regular representation translating functions

over $\mathbb{R}^2$. With this notation we can define a *convolutional layer*[1] as

$$[\phi f](x) = \int_{\mathbb{R}^2} [L_x \widehat{\kappa}](y) f(y) \mathrm{d}y = \int_{\mathbb{R}^2} \widehat{\kappa}(y - x) f(y) \mathrm{d}y, \tag{4.1}$$

where $\widehat{\kappa} : \mathbb{R}^2 \to \mathrm{Hom}(V_{\text{in}}, V_{\text{out}})$ is the learnable convolution kernel.

That this is equivariant with respect to translation, i.e.

$$[L_z[\phi f]](x) = [\phi[L_z f]](x), \tag{4.2}$$

is straightforward to show:

$$\begin{aligned}
[L_z[\phi f]](x) &= [\phi f](x - z) && \text{(Def. of } L_z\text{)} \\
&= \int_{\mathbb{R}^2} \widehat{\kappa}(y - (x - z)) f(y) \mathrm{d}y && \text{(Def. of } \phi\text{)} \\
&= \int_{\mathbb{R}^2} \widehat{\kappa}(y + z - x) f(y) \mathrm{d}y \\
&= \int_{\mathbb{R}^2} \widehat{\kappa}(y' - x) f(y' - z) \mathrm{d}y' && (y' = y + z) \\
&= \int_{\mathbb{R}^2} \widehat{\kappa}(y' - x) [L_z f](y') \mathrm{d}y' && \text{(Def. of } L_z\text{)} \\
&= [\phi[L_z f]](x). && \text{(Def. of } \phi\text{)}
\end{aligned} \tag{4.3}$$

The convolutional layer in eq. (4.1) is a special case of the more general linear map

$$[\widetilde{\phi} f](x) = \int_{\mathbb{R}^2} \kappa(x, y) f(y) \mathrm{d}y. \tag{4.4}$$

To obtain the convolutional layer in eq. (4.1) one imposes the weight sharing condition, which is an inductive bias

$$\kappa(x, y) = \kappa(x - z, y - z), \qquad \forall z \in \mathbb{R}^2 \tag{4.5}$$

and this allows the definition of a one-argument kernel $\widehat{\kappa}(y - x) := \kappa(0, y - x)$ as through eq. (4.5) we get $\kappa(0, y - x) = \kappa(x, y)$. One can also derive this weight sharing constraint from assuming that $\widetilde{\phi}$ is translation equivariant, hence they are equivalent.

As stated I now want to show how this weight sharing constraint appears when approaching using differential geometry. To do this, we must first introduce

[1] This is technically a *cross correlation* but in machine learning contexts this is called a convolutional layer.

some new viewpoints.

Instead of starting from a general kernel $\kappa : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathrm{Hom}(V_{\text{in}}, V_{\text{out}})$ depending on two points x, y, we will let $\kappa : \mathbb{R}^2 \times T\mathbb{R}^2 \to \mathrm{Hom}(V_{\text{in}}, V_{\text{out}})$. That is, the kernel depends on a point x in the base manifold ($\mathbb{R}^2$) and a tangent vector $X_x \in T_x\mathbb{R}^2$. Intuitively, this tangent vector is the relative position of x and y. [2] Let now $\mathcal{T}_{X_x}(x)$ be the map that moves x along the curve generated by the tangent vector X_x for unit time. In other words, $\mathcal{T}_{X_x}(x)$ is the unit time *geodesic flow*. Since $\mathbb{R}^2$ is flat, this curve is a straight line. See fig. 4.1 for a visualisation of these concepts.

Remark 4.1.1. There is a useful fact for the geodesic flow. Stated in general terms: if $\varphi : \mathcal{M} \to \mathcal{N}$ is a diffeomorphism and $\mathcal{T}_{X_x}$ is the flow of $X_x \in T_x\mathcal{M}$ for any amount of time, then

$$\varphi(\mathcal{T}_{X_x}(x)) = \mathcal{T}_{(d\varphi)_x(X_x)}(\varphi(x)), \tag{4.6}$$

where $(d\varphi)_x(X_x) \in T_{\varphi(x)}\mathcal{N}$ is the pushforward of $X_x \in T_x\mathcal{M}$ by φ. This can be depicted as a commutative diagram:

$$\begin{array}{ccc} \mathcal{M} & \xrightarrow{\mathcal{T}_{X_x}} & \mathcal{M} \\ {\scriptstyle\varphi}\downarrow & & \downarrow{\scriptstyle\varphi} \\ \mathcal{N} & \xrightarrow[\mathcal{T}_{(d\varphi)_x(X_x)}]{} & \mathcal{N} \end{array} \tag{4.7}$$

This fact will come in useful at several points, specifically in the special case where the diffeomorphism $\varphi : \mathcal{M} \to \mathcal{M}$ is the translation operator. For a visualisation of the special case $\varphi : \mathbb{R}^2 \to \mathbb{R}^2$ — where the manifold is flat — see fig. 4.2. On the other hand, for the special case of the right fibre preserving translation $R_h : P \to P$ in a principal H-bundle, see fig. 4.3.

With these conceptual, and notational, changes we can now define the integral map in this context:

Definition 4.1.2. Let $f : \mathbb{R}^2 \to V_{\text{in}}$ be the input feature map. Then let ϕf be the map

$$[\phi f](x) = \int_{T_x\mathbb{R}^2} \kappa(x, X_x) f(\mathcal{T}_{X_x}(x)) \mathrm{d}X_x, \tag{4.8}$$

where $\kappa : \mathbb{R}^2 \times T\mathbb{R}^2 \to \mathrm{Hom}(V_{\text{in}}, V_{\text{out}})$ is the integration kernel.

[2] This holds only for flat spaces like $\mathbb{R}^n$.

(a) Figure of two points **(b)** Figure of point and relative vector

Figure 4.1: Two ways of viewing the inputs to the kernel κ. The left panel shows the standard approach of viewing the input as two points, while the right panel shows the approach taken here where we select one point as the "base point" x and obtain the other point y by translating x by a vector X_x. That is, $y = \mathcal{T}_{X_x}(x)$.

Note that this map is not yet equivariant. In this setup, it is now natural to view a translation by z as a diffeomorphism $z : \mathbb{R}^2 \to \mathbb{R}^2$ acting as $z(x) = x - z$. This also generates a pushforward $dz_x : T_x\mathbb{R}^2 \to T_{z(x)}\mathbb{R}^2$.

We can now define the translation action on maps $f : \mathbb{R}^2 \to V$ as $[L_z f](x) = (f \circ z)(x) = f(z(x)) = f(x - z)$, and check what condition needs to hold to make $\widetilde{\phi}$ equivariant:

$$[\phi[L_z f]](x) \overset{!}{=} [L_z[\phi f]](x). \tag{4.9}$$

Theorem 4.1.3. *The map ϕ defined in definition 4.1.2 is translation equivariant, that is*

$$[L_z[\phi f]](x) = [\phi[L_z f]](x), \tag{4.10}$$

iff the kernel $\kappa : \mathbb{R}^2 \times T\mathbb{R}^2 \to \mathrm{Hom}(V_{in}, V_{out})$ satisfies

$$\kappa(x, X_x) = \kappa(z(x), dz_x(X_x)), \qquad \forall z \in \mathbb{R}^2. \tag{4.11}$$

That is, equivariance is equivalent to sharing the kernel weights over $\mathbb{R}^2$.

Proof. Starting with the right hand side of eq. (4.9) we note that $[L_z[\phi f]](x) = [\phi f](z(x))$ and

$$[\phi f](z(x)) = \int_{T_{z(x)}\mathbb{R}^2} \kappa(z(x), X_{z(x)}) f(\mathcal{T}_{X_{z(x)}}(z(x)))\mathrm{d}X_{z(x)}. \tag{4.12}$$

Now, noting that the pushforward $dz_x : T_x\mathbb{R}^2 \to T_{z(x)}\mathbb{R}^2$ is an isomorphism,

we can use it to change from integrating over $T_x\mathbb{R}^2$ to integrating over $T_{z(x)}\mathbb{R}^2$ by setting $X_{z(x)} = dz_x(X_x)$. Additionally, the pushforward dz_x is an isometry and hence we do not get any additional volume factor in the integral. This yields

$$\begin{aligned}\int_{T_{z(x)}\mathbb{R}^2} \kappa(z(x), X_{z(x)}) f(\mathcal{T}_{X_{z(x)}}(z(x)))\mathrm{d}X_{z(x)} \\ = \int_{T_x\mathbb{R}^2} \kappa(z(x), dz_x(X_x)) f(\mathcal{T}_{dz_x(X_x)}(z(x)))\mathrm{d}X_x.\end{aligned} \tag{4.13}$$

On the left hand side of eq. (4.9) we have

$$\begin{aligned}[\phi[L_z f]](x) &= \int_{T_x\mathbb{R}^2} \kappa(x, X_x)[L_z f](\mathcal{T}_{X_x}(x))\mathrm{d}X_x \\ &= \int_{T_x\mathbb{R}^2} \kappa(x, X_x) f(z(\mathcal{T}_{X_x}(x)))\mathrm{d}X_x.\end{aligned} \tag{4.14}$$

This is already close to eq. (4.13) and we now need to relate $\mathcal{T}_{dz_x(X_x)}(z(x))$ and $z(\mathcal{T}_{X_x}(x))$. As was stated in remark 4.1.1 we have

$$\mathcal{T}_{dz_x(X_x)}(z(x)) = z(\mathcal{T}_{X_x}(x)), \tag{4.15}$$

which is visualised in fig. 4.2. Using this in eq. (4.13) we arrive at that the following equality must hold for ϕ to be equivariant:

$$\int_{T_x\mathbb{R}^2} \kappa(z(x), dz_x(X_x)) f(z(\mathcal{T}_{X_x}(x)))\mathrm{d}X_x \overset{!}{=} \int_{T_x\mathbb{R}^2} \kappa(x, X_x) f(z(\mathcal{T}_{X_x}(x)))\mathrm{d}X_x. \tag{4.16}$$

Since this needs to hold for all f we can extract the point wise condition

$$\kappa(z(x), dz_x(X_x)) \overset{!}{=} \kappa(x, X_x), \tag{4.17}$$

which needs to hold for all translations z. □

This is just spatial weight sharing and allows us, with the choice of some point x_0 at the origin, to define a one-argument kernel

$$\widehat{\kappa} : T_{x_0}\mathbb{R}^2 \to \mathrm{Hom}(V_{\mathrm{in}}, V_{\mathrm{out}}), \tag{4.18}$$

in terms of the two-argument kernel using eq. (4.17)

$$\widehat{\kappa}(X) := \kappa(x_0, X), \tag{4.19}$$

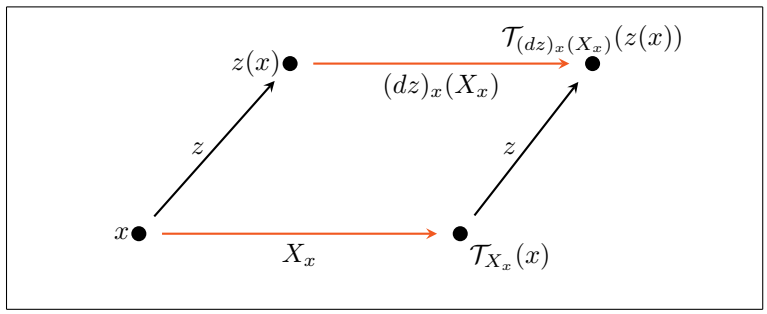


Figure 4.2: A visualisation of the commutativity of the flow and pushforward on $\mathbb{R}^2$. This is a special case of remark 4.1.1 where manifold is flat.

which can then be moved to any other point on $\mathbb{R}^2$ by a translation.

Hence, we again land back in weight sharing but through the lens of differential geometry, specifically sharing the kernel between different tangent spaces. This notion will be useful in the next section where I present a similar derivation for a convolutional-like layer equivariant to vertical transformations.

4.2 Equivariance in biprincipal bundles

The example in the previous section explored ordinary convolutional layers from a differential geometry point of view. One can view the diffeomorphism — the translation of the plane — used there as a purely horizontal transformation. [3]

But can one formulate something similar for the purely vertical transformations that are common in principal bundles? As we will see in this section: yes, it is possible to formulate a similar statement for functions defined on principal bundles, or more specifically biprincipal bundles.

In this section we will assume that the base space $\mathcal{M}$ is a Riemannian manifold, that is, $\mathcal{M}$ is a smooth manifold with a smoothly varying metric $g_{\mathcal{M}}$.

First, we introduce the biprincipal bundle, and to do this we need the definition of a torsor, repeated here for convenience.

Definition 4.2.1 (Torsor)**.** Given a group H, a left (right) H-space X is a left (right) torsor if the action of H on X satisfies that

[3] As we will see in section 4.2.1 the translation of the plane is also naturally viewed as a vertical transformation of the principal bundle $\pi : \mathbb{R}^2 \to \{0\}$.

1. H acts freely: for each $h \in H$ it holds that $\forall z \in X$, $h \triangleright x = x$ implies that $h = e$.
2. H acts transitively: $\forall x, y \in X$ there exists $h \in H$ such that $y = h \triangleright x$.

A right torsor H-torsor is defined identically, but with a right action.

In essence, if X is a H-torsor that means that X is isomorphic to H as a set. Intuitively, X is H but without an origin where only relative elements are available. In a principal H-bundle P the fibres are H-torsors — usually taken as right H-torsors — and only when given a choice of local trivialisation, e.g., a choice of origin, can the fibres be identified with the actual group H. The notion of a principal H-bundle can be generalised slightly to yield a biprincipal bundle. But first we need the notion of a bitorsor.

Definition 4.2.2 (Bitorsor [57, 16])**.** A (G, H)-*bitorsor* X is simultaneously a left G-torsor and a right H-torsor such that their actions commute:

$$R_h(L_g(x)) = L_g(R_h(x)), \quad \forall x \in X. \tag{4.20}$$

Remark 4.2.3. As the X is both a G and a H-torsor, this means that G and H are isomorphic as groups through a non-canonical isomorphism which depends on a choice of origin $x_0 \in X$.

With the definition of the bitorsor we can now define the biprincipal bundle.

Definition 4.2.4 (Biprincipal (G, H)-bundle [16])**.** A *biprincipal* (G, H)*-bundle* is a fibre bundle $\pi : P \to \mathcal{M}$ where each fibre P_x over $x \in \mathcal{M}$ is a (G, H)-bitorsor.

In this section we will have $\pi : P \to \mathcal{M}$ be a biprincipal (G, H)-bundle assumed to be a smooth manifold.

Now with the biprincipal (G, H)-bundle introduced we turn to the data. We are going to assume that our feature vectors lies in a H-representation (ρ, V). In this setting we let feature maps be sections of the associated vector bundle $P \times_\rho V = (P \times V)/H$ where $\pi : P \to \mathcal{M}$ is a biprincipal (G, H)-bundle. Note, that since P has a defined left G action we also have a G action on the equivalence classes of the associated vector bundle:

$$g[p, v] = [gp, v], \tag{4.21}$$

and while the equivalence class is invariant to H, this left action maps between different equivalence classes although the fibre is preserved.

By theorem 3.2.19, which is formally formulated for principal bundles can be extended to the biprincipal case, we can equivalently work with functions on the (bi)principal bundle $f : P \to V$ satisfying the Mackey condition

$$f(ph) = \rho(h^{-1})f(p), \qquad \forall h \in H. \tag{4.22}$$

For the sake of notational neatness, let $\mathcal{I}_\rho$ denote the space of such functions. Additionally we can equip the vector space V with a norm making it a Banach space and we will restrict our feature maps f to be smooth and square Bochner-integrable.

The biprincipal nature of P yields a natural G-action on the space of $\mathcal{I}_\rho$ as

$$[g \rhd f](p) = f(L_{g^{-1}}(p)), \tag{4.23}$$

where $L_{g^{-1}}(p) = g^{-1}p$.

Now we present the map ϕ inspired by eq. (4.8) which acts on input feature maps $f \in \mathcal{I}_{\rho_{\text{in}}}$. Initially we only know that the output ϕf is a map $\phi f : P \to V_{\text{out}}$ and we will later investigate what is needed to obtain $\phi f \in \mathcal{I}_{\rho_{\text{out}}}$.

Definition 4.2.5. For $f \in \mathcal{I}_{\rho_{\text{in}}}$ let $\phi : \mathcal{I}_{\rho_{\text{in}}} \to \text{Map}(P, V_{\text{out}})$ be a map defined as

$$[\phi f](p) = \int_{B_R(p)} \kappa(p, X_p) f(\mathcal{T}_{X_p}(p)) \mathrm{d}X_p, \tag{4.24}$$

where $\kappa : P \times TP \to \text{Hom}(V_{\text{in}}, V_{\text{out}})$ is an integration kernel, and $\mathcal{T}_{X_p}(p)$ is the map which transports p along the curve in P defined by the flow of the tangent vector X_p for unit time. Additionally, the integration domain $B_R(p) := \left\{X_p \in T_pP \mid \sqrt{(g_P)_p(X_p, X_p)} < R\right\}$ is an open ball in the tangent space T_pP at $p \in P$. The metric g_P is the metric introduced at the end of section 3.2 and $\mathrm{d}X_p$ is the volume form in T_pP with respect to this metric.

Note, as stated in section 3.2, this metric is invariant to the right translation by H on the fibres, but this does not hold for the left action. In general, while dR_h preserves the horizontal subspaces, dL_g mixes the vertical and horizontal parts of the tangent vectors and is not an isometry.

To ensure that ϕ is a continuous and bounded map we will additionally assume that κ is smooth in both p and X_p as well as $\kappa(p, X_p)$ is bounded for each p and $X_p \in T_pP$ — this is ensured as V_{in} and V_{out} are assumed to be finite dimensional vector spaces. Since V_{in} and V_{out} are finite-dimensional, and κ is assumed smooth in both p and X_p, the point wise boundedness of $\kappa(p, X_p)$

implies that

$$\sup_{p\in P}\sup_{X_p\in T_pP}\|\kappa(p,X_p)\|_{\mathrm{Hom}(V_{\mathrm{in}},V_{\mathrm{out}})}<\infty. \tag{4.25}$$

Opposed to the previous section where we only needed to deal with the translation — and hence weight sharing — of the layer, here we additionally have the Mackey condition to take into account. Hence, to make sure that $\phi f \in \mathcal{I}_{\rho_{\mathrm{out}}}$ we have the following lemma.

Lemma 4.2.6. *Let ϕ be defined as in definition 4.2.5, then $[\phi f] \in \mathcal{I}_{\rho_{out}}$ iff the kernel satisfies*

$$\kappa(R_h(p),(dR_h)_p(X_p)) = \rho_{out}(h^{-1})\kappa(p,X_p)\rho_{in}(h), \qquad \forall h\in H. \tag{4.26}$$

Proof. First assume that $\phi f \in \mathcal{I}_{\rho_{\mathrm{out}}}$, that is

$$[\phi f](R_h(p)) = \rho_{\mathrm{out}}(h^{-1})[\phi f](p). \tag{4.27}$$

Now we use the definition of ϕ to get

$$[\phi f](R_h(p)) = \int_{B_R(R_h(p))} \kappa(R_h(p),X_{R_h(p)})f(\mathcal{T}_{X_{R_h(p)}}(R_h(p)))\mathrm{d}X_{R_h(p)}, \tag{4.28}$$

then, as in the previous section, we change our integration domain through the pushforward $(dR_h)_p : H_pP \to H_{R_h(p)}P$ and since $(dR_h)_p$ is an isomorphism — even an isometry and hence $\mathrm{d}X_p = \mathrm{d}X_{R_h(p)}$ — we can write $X_{R_h(p)} = (dR_h)_p(X_p)$. And since dR_h is an isometry we have $(dR_h)_p^{-1}(B_R(R_h(p))) = B_R(p)$ which yields

$$[\phi f](R_h(p)) = \int_{B_R(p)} \kappa(R_h(p),(dR_h)_p(X_p))f(\mathcal{T}_{(dR_h)_p(X_p)}(R_h(p)))\mathrm{d}X_p. \tag{4.29}$$

Recall now remark 4.1.1 and note that, similar to the flat case of the ordinary CNN in section 4.1, we have

$$\mathcal{T}_{(dR_h)_p(X_p)}(R_h(p)) = R_h(\mathcal{T}_{X_p}(p)), \tag{4.30}$$

see figure fig. 4.3 for a visualisation. Using this we get

$$[\phi f](R_h(p)) = \int_{B_R(p)} \kappa(R_h(p),(dR_h)_p(X_p))f(R_h(\mathcal{T}_{X_p}(p)))\mathrm{d}X_p, \tag{4.31}$$

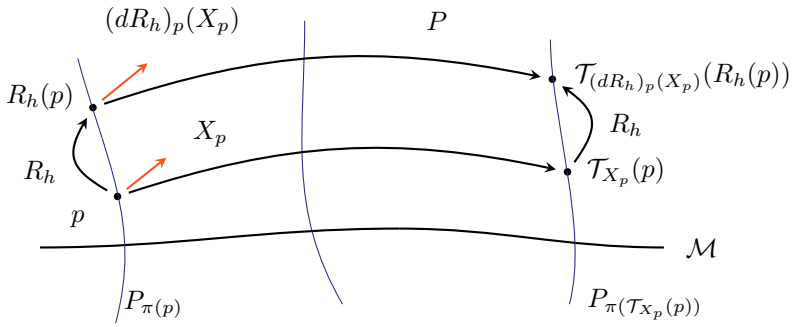


Figure 4.3: The commutativity of the pushforward and the (geodesic) flow. This is a special case of remark 4.1.1 for when the diffeomorphism is the fibre preserving right action $R_h : P \to P$.

which, since $f \in \mathcal{I}_{\rho_{\text{in}}}$, becomes

$$[\phi f](R_h(p)) = \int_{B_R(p)} \kappa(R_h(p), (dR_h)_p(X_p))\rho_{\text{in}}(h^{-1}) f(\mathcal{T}_{X_p}(p))\mathrm{d}X_p. \tag{4.32}$$

Now going from the other side of eq. (4.27) we know that $[\phi f](R_h(p)) = \rho_{\text{out}}(h^{-1})[\phi f](p)$, this means that

$$\begin{aligned}[\phi f](R_h(p)) &= \rho_{\text{out}}(h^{-1})[\phi f](p) \\ &= \int_{B_R(p)} \rho_{\text{out}}(h^{-1})\kappa(p, X_p) f(\mathcal{T}_{X_p}(p))\mathrm{d}X_p,\end{aligned} \tag{4.33}$$

after moving $\rho_{\text{out}}(h^{-1})$ inside the integral.

Finally, as eq. (4.32) and eq. (4.33) need to be equal for all f we obtain that

$$\kappa(R_h(p), (dR_h)_p(X_p))\rho_{\text{in}}(h^{-1}) = \rho_{\text{out}}(h^{-1})\kappa(p, X_p), \tag{4.34}$$

and multiplying both sides with $\rho_{\text{in}}(h)$ from the right results in the wanted condition:

$$\kappa(R_h(p), (dR_h)_p(X_p)) = \rho_{\text{out}}(h^{-1})\kappa(p, X_p)\rho_{\text{in}}(h). \tag{4.35}$$

To obtain the other direction, simply follow this proof backwards. □

This shows that ϕ is actually a map between $\mathcal{I}_{\rho_{\text{in}}}$ to $\mathcal{I}_{\rho_{\text{out}}}$. To get equivariance requires a bit more work.

First, we use the fact that P is a biprincipal (G, H)-bundle to define the left group action of G acting on $\mathcal{I}_\rho$:

$$[g \rhd f](p) = f(g^{-1}p). \tag{4.36}$$

Equivariance of ϕ then means

$$[g \rhd [\phi f]](p) = [\phi[g \rhd f]](p). \tag{4.37}$$

We can formulate the requirement for equivariance as a lemma:

Lemma 4.2.7. *Let ϕ be defined as in definition 4.2.5 and let $\widehat{G} \leqslant G$ be any subgroup of G. Then ϕ is equivariant iff the kernel satisfies*

$$\begin{aligned} &\mathbf{1}_{B_R(p)}(X_p)\kappa(p, X_p) \\ &= \mathbf{1}_{\widetilde{B}_R(p)}(X_p)\kappa(L_{g^{-1}}(p), (dL_{g^{-1}})_p(X_p)) \left|\det((dL_{g^{-1}})_p)\right|, \end{aligned} \tag{4.38}$$

for all $g \in \widehat{G}$, where $\widetilde{B}_R(p) = (dL_{g^{-1}})_p^{-1}(B_R(L_{g^{-1}}(p)))$.

That is, to obtain $\widehat{G}$-equivariance the kernel needs to be shared along the $\widehat{G}$-orbit in the fibre.

Proof. Assume that ϕ is $\widehat{G}$-equivariant, that is

$$[g \rhd [\phi f]](p) = [\phi[g \rhd f]](p), \quad \forall g \in \widehat{G}. \tag{4.39}$$

First, expand the left hand side $[g \rhd [\phi f]](p) = [\phi f](L_{g^{-1}}(p))$ and hence

$$\begin{aligned} [g \rhd [\phi f]](p) = \int_{B_R(L_{g^{-1}}(p))} &\kappa(L_{g^{-1}}(p), X_{L_{g^{-1}}(p)}) \\ &f(\mathcal{T}_{X_{L_{g^{-1}}(p)}}(L_{g^{-1}}(p)))\mathrm{d}X_{L_{g^{-1}}(p)}. \end{aligned} \tag{4.40}$$

Now we use $(dL_{g^{-1}})_p : T_pP \to T_{L_{g^{-1}}(p)}P$ to change coordinates as $X_{L_{g^{-1}}(p)} = (dL_{g^{-1}})_p(X_p)$. Here, as $dL_{g^{-1}}$ is not an isometry, we get that $\mathrm{d}X_{L_{g^{-1}}(p)} = \left|\det((dL_{g^{-1}})_p)\right| \mathrm{d}X_p$. Note that the new integration domain becomes

$$(dL_{g^{-1}})_p^{-1}(B_R(L_{g^{-1}}(p))), \tag{4.41}$$

which we will denote as $\widetilde{B}_R(p)$ for brevity. Together this yields

$$\begin{aligned}[g \rhd [\phi f]](p) = \int_{\widetilde{B}_R(p)} & \kappa(L_{g^{-1}}(p), (dL_{g^{-1}})_p(X_p)) \\ & f(\mathcal{T}_{(dL_{g^{-1}})_p(X_p)}(L_{g^{-1}}(p))) \left|\det((dL_{g^{-1}})_p)\right| \mathrm{d}X_p.\end{aligned} \tag{4.42}$$

As before, we use the fact, stated in remark 4.1.1, that

$$\mathcal{T}_{(dL_{g^{-1}})_p(X_p)}(L_{g^{-1}}(p)) = L_{g^{-1}}(\mathcal{T}_{X_p}(p)), \tag{4.43}$$

and hence

$$\begin{aligned}[g \rhd [\phi f]](p) = \int_{\widetilde{B}_R(p)} & \kappa(L_{g^{-1}}(p), (dL_{g^{-1}})_p(X_p)) \\ & f(L_{g^{-1}}(\mathcal{T}_{X_p}(p))) \left|\det((dL_{g^{-1}})_p)\right| \mathrm{d}X_p.\end{aligned} \tag{4.44}$$

We now note that $f(L_{g^{-1}}(\mathcal{T}_{X_p}(p))) = [g \rhd f](\mathcal{T}_{X_p}(p))$ and hence

$$\begin{aligned}[g \rhd [\phi f]](p) = \int_{\widetilde{B}_R(p)} & \kappa(L_{g^{-1}}(p), (dL_{g^{-1}})_p(X_p)) \\ & [g \rhd f](\mathcal{T}_{X_p}(p)) \left|\det((dL_{g^{-1}})_p)\right| \mathrm{d}X_p.\end{aligned} \tag{4.45}$$

To compare this with coming from the other direction we expand this to an integral over the entire T_pP by adding the indicator function $\mathbf{1}_{(dL_{g^{-1}})_p(B_R(p))}$ to arrive at

$$\begin{aligned}[g \rhd [\phi f]](p) = \int_{T_pP} & \mathbf{1}_{\widetilde{B}_R(p)}(X_p) \kappa(L_{g^{-1}}(p), (dL_{g^{-1}})_p(X_p)) \\ & [g \rhd f](\mathcal{T}_{X_p}(p)) \left|\det((dL_{g^{-1}})_p)\right| \mathrm{d}X_p.\end{aligned} \tag{4.46}$$

Now we turn the attention to the right hand side on eq. (4.39) and expand

$$[\phi[g \rhd f]](p) = \int_{B_R(p)} \kappa(p, X_p)[g \rhd f](\mathcal{T}_{X_p}(p))\mathrm{d}X_p. \tag{4.47}$$

Then, as in the final step of $[g \rhd [\phi f]]$ extend this integral to T_pP by adding the indicator function $\mathbf{1}_{B_R(p)}$. This gives

$$[\phi[g \rhd f]](p) = \int_{T_pP} \mathbf{1}_{B_R(p)}(X_p)\kappa(p, X_p)[g \rhd f](\mathcal{T}_{X_p}(p))\mathrm{d}X_p. \tag{4.48}$$

Then, as the left and right hand side of eq. (4.39) must be equal for all feature

maps f we must have that eq. (4.46) and eq. (4.48) are equal for all f. This allows us to extract the point wise equality

$$\begin{aligned}&\mathbf{1}_{B_R(p)}(X_p)\kappa(p, X_p)\\&= \mathbf{1}_{\widetilde{B}_R(p)}(X_p)\kappa(L_{g^{-1}}(p), (dL_{g^{-1}})_p(X_p))\left|\det((dL_{g^{-1}})_p)\right|,\end{aligned} \tag{4.49}$$

for all $g \in \widehat{G}$. To get $\widehat{G}$-equivariance from eq. (4.49) simply perform this proof backwards. □

Now we can finalise this section by collecting these results into a theorem.

Theorem 4.2.8 (Equivariance in biprincipal bundle)**.** *Let $\pi : P \to \mathcal{M}$ be a biprincipal (G, H)-bundle and let ϕ be defined as in definition 4.2.5, then*

1. *$\phi : \mathcal{I}_{\rho_{in}} \to \mathcal{I}_{\rho_{out}}$ iff κ satisfies lemma 4.2.6, that is*

$$\kappa(R_h(p), (dR_h)_p(X_p)) = \rho_{out}(h^{-1})\kappa(p, X_p)\rho_{in}(h), \qquad \forall h \in H, \tag{4.50}$$

   and;

2. *the map ϕ is $\widehat{G} \leqslant G$-equivariant, that is $[g \rhd [\phi f]](p) = [\phi[g \rhd f]](p)$, iff the kernel satisfies the weight sharing in lemma 4.2.7:*

$$\begin{aligned}&\mathbf{1}_{B_R(p)}(X_p)\kappa(p, X_p)\\&= \mathbf{1}_{\widetilde{B}_R(p)}(X_p)\kappa(L_{g^{-1}}(p), (dL_{g^{-1}})_p(X_p))\left|\det((dL_{g^{-1}})_p)\right|,\end{aligned} \tag{4.51}$$

   for all $g \in \widehat{G}$, where $\widetilde{B}_R(p) = (dL_{g^{-1}})_p^{-1}(B_R(L_{g^{-1}}(p)))$.

Proof. The proof follows immediately from lemma 4.2.6 and lemma 4.2.7. □

There is one final important detail, namely the role of the indicator functions in lemma 4.2.7. Since the condition eq. (4.38) should hold for all $g \in G$ we get problems when G is non-compact because through this we can reduce the domain of the indicator function on the RHS to arbitrarily small domain. Taking the infimum of both domains we arrive at the kernel can only be defined for $X_p = 0$. This would reduce the map fully to a pointwise map of the features and is the same as the 1×1 GM-convolutions in [159, p. 208].

Naively we could try to avoid this by initially, in definition 4.2.5, integrating over the entire tangent space T_pP instead of over the open ball $B_R(p)$. This would seemingly remove the support constraint resulting in a constraint

$$\kappa(p, X_p) = \kappa(L_{g^{-1}}(p), (dL_{g^{-1}})_p(X_p))\left|\det((dL_{g^{-1}})_p)\right|, \qquad \forall g \in \widehat{G}. \tag{4.52}$$

However, as κ is bounded the only solution for eq. (4.52) that works for all $g \in \widehat{g}$ is that $\kappa = 0$.

One solution that allows for spatially extended kernels is that resctrict G such that L_g is an isometry for all g. In that case we get that $B_R(L_{g^{-1}}(p)) = (dL_{g^{-1}})_p(R_R(p))$ remains the same integration domain. This yields a "proper" weight sharing which does not collapse:

$$\kappa(p, X_p) = \kappa(L_{g^{-1}}(p), (dL_{g^{-1}})_p(X_p)), \qquad \forall g \in \widehat{G}. \tag{4.53}$$

In [159, p. 225] this is referred to as isometry equivariance.

4.2.1 Relation to the ordinary CNN case

In section 4.1 I walked through the formulation of a normal CNN in this differential geometry setting, however there is one significant difference to the formulation in theorem 4.2.8. For the standard CNN case our feature maps have a trivial representation $\rho = \mathrm{id}$ and hence the biprincipal framework is not needed — just a simple principal bundle suffices. Specifically, a normal CNN is obtained through a principal bundle $\pi : \mathbb{R}^2 \to \{0\}$.

4.2.2 Relation to equivariant CNNs on homogeneous spaces

This formulation stated above has some close connections to the fibre bundle perspective of the equivariant CNN on homogeneous spaces [29]. That formulation of equivariant CNNs is based on viewing a homogeneous space G/H as the base space of a principal H-bundle where the right H-action preserves fibres and the left G-action moves between fibres. This is visualised in fig. 4.4.

In this way one of course attains both a left G and a right H action on the total space as in the biprincipal bundle case. In the biprincipal case, however, we need that G and H are isomorphic and that both G and H-actions leave the fibres invariant.

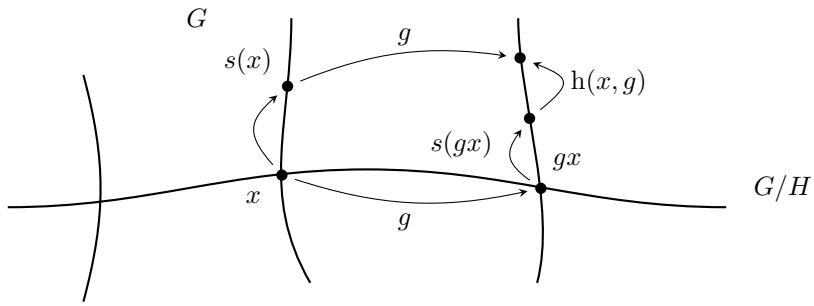


Figure 4.4: A visualisation of of the left G action moving between the fibres and the map $\mathrm{h} : G/H \times G \to H$ encoding the twist in the fibres, essentially determining the connection. Figure is figure 3 in **Paper IV**.

4.3 Coordinate independence and spatially local actions

For more details on coordinate independent networks I refer the interested reader to Weiler's PhD thesis [159]. I would, however, like to elaborate on a specific aspect. Namely, discuss the effects of local transformations on networks consisting of several layers or local transformations of foreground elements.

As mentioned before, there are two viewpoints on transformations: *active transformations* in which the data, or underlying manifold, is actively moved around and *passive transformations* in which a change of local coordinates makes it look like the manifold is transformed.

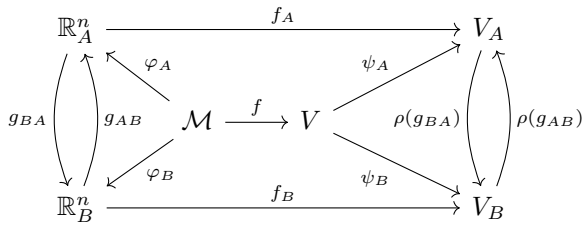
A cornerstone on this discussion is the concept of coordinate independent network, specifically a coordinate independent CNN ϕ. To illustrate coordinate independence, let $\pi : E \to \mathcal{M}$ be a vector bundle with typical fibre V being a representation ρ of the structure group G of $\mathcal{M}$. Also let $U \subset \mathcal{M}$ be an open set, and (φ_A, ψ_A) and (φ_B, ψ_B) be two local trivialisations assigning coordinates $\varphi_{A/B}$ on U and fibre coordinates $\psi_{A/B}$ on V. Additionally, assume that the coordinate representations $\varphi_A(U)$ and $\varphi_B(U)$ of U are related by some group transformation

$$\varphi_A(U) = g_{AB}\varphi_B(U), \tag{4.54}$$

and similarly for the fibre coordinates

$$\psi_A(V) = \rho(g_{AB})\psi_B(V). \tag{4.55}$$

Now, if $f : \mathcal{U} \to V$ is a section of E we can express f in the different local trivialisations and compare how they relate. The coordinate transformations are illustrated in the commutative diagram below.



(4.56)

By following these maps we arrive at

$$f_A(x_A) = \rho(g_{AB})f_B(g_{BA}x_A), \tag{4.57}$$

showing how to change the coordinate representation of the feature map f. A coordinate independent CNN is a map which is consistent under a change of coordinates on the base manifold as described above.

Let us first discuss local active transformations and a assume that ϕ is a coordinate independent network which, for the sake of argument, computes a gradient of the data in its receptive field. Further assume that the "background" is uniform so that a local transformation only affects the foreground objects of interest. Then ϕ will be equivariant to all group actions which are limited to a single receptive field. See the leftmost, green, patch centred at P_1 in the top and bottom row in fig. 4.5 which is acted on by a local active transformation contained completely in the patch, while the rightmost, red, patch centred at P_3 is completely untouched. This contained transformation causes a similar transformation of the feature computed at P_1.

However, when a patch includes differently transformed objects, then this generally cannot be represented as the action of a group element and hence we do not know a priori how the output on the partially transformed patch relate to the output on the non-transformed version. This is visualised in the middle, blue, patch centred at P_2 in fig. 4.5; for easier comparison, these patches are also extracted and shown in fig. 4.6.

The case of local passive transformation is a bit more nuanced. A passive transformation appears when one assigns different local coordinates to the same

region. Let (φ_A, U_{P_1}) and (φ_B, U_{P_2}) be two local coordinate charts charts such that $P_3 \in U_{P_3} \subset U_{P_1} \cap U_{P_2}$. See the top panel of fig. 4.7. This means that both in the chart intersection U_3 we have two possible coordinate systems, and the coordinate independent network ϕ will be equivariant to the change of local coordinates by the cart transition map $\varphi_{AB} := \varphi_A \circ \varphi_B^{-1}$. The centre of the middle panel of fig. 4.7 shows the data in their local coordinates and the chart transition map on their overlap. If however we apply ϕ at P_1 and P_2 and the receptive field extends beyond where the charts overlap then there are no chart transition map relating which we can use to map the output at P_1 to the output at P_2. Hence we need to express the output at P_1 and P_2 relative to the local coordinates φ_A and φ_B respectively, as seen on the outer segments of the middle panel of fig. 4.7. Now, if we naively applied the next coordinate independent layer we would mix feature vectors expressed in different local coordinates, generally one coordinate chart per point, and this would lead to inconsistencies. To avoid this one can contract the feature vectors with their basis to arrive at the coordinate free formulation, lower panel in fig. 4.7, which can then be fed to the next layer without inconsistencies.

As previously stated, for more details on this, see [159].

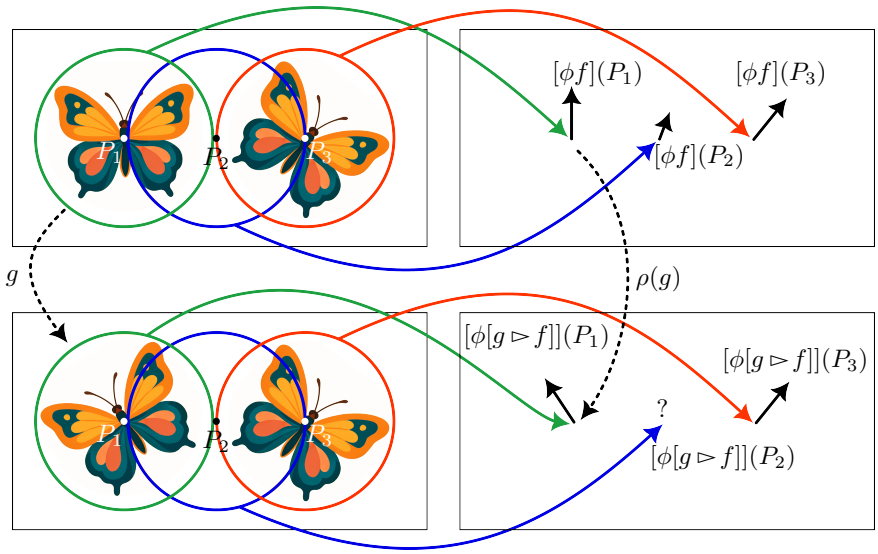


Figure 4.5: Illustration of an active local transformation and the coordinate independent network ϕ applied to three different positions: P_1, P_2 and P_3 with the receptive field shown as a circle centred at each position. For the leftmost and rightmost position the receptive field completely covers a foreground element with no additional details, while the receptive field for the middle position intersects with two foreground elements in different orientations. When an active local transformation is applied on the leftmost object $g \rhd P_1$, since this is a clear group action g, the output transforms accordingly $[\phi[g \rhd f]](P_1) = \rho(g)[\phi f](P_1)$. Similarly, the network on P_3 is unaffected. For P_2 however, there is no group element relating P_2 before and after the transformation, hence the output of $[\phi f](P_2)$ after the transformation is unknown. The situation is similar if foreground elements are transformed while a non-uniform background remains untransformed. In that case there is no group element relating the patch before the transformation to the same patch after the transformation. Note that this is a schematic picture, the feature vectors does not need to be tangent vectors, but it is easier to visualise.

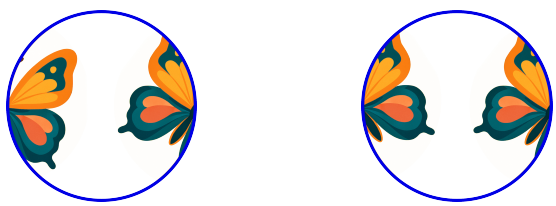


Figure 4.6: Illustration of the middle patch of fig. 4.5 showing that no element of the structure group maps one to the other. Hence the outputs of a coordinate independent network on each patch are not related by some group action.

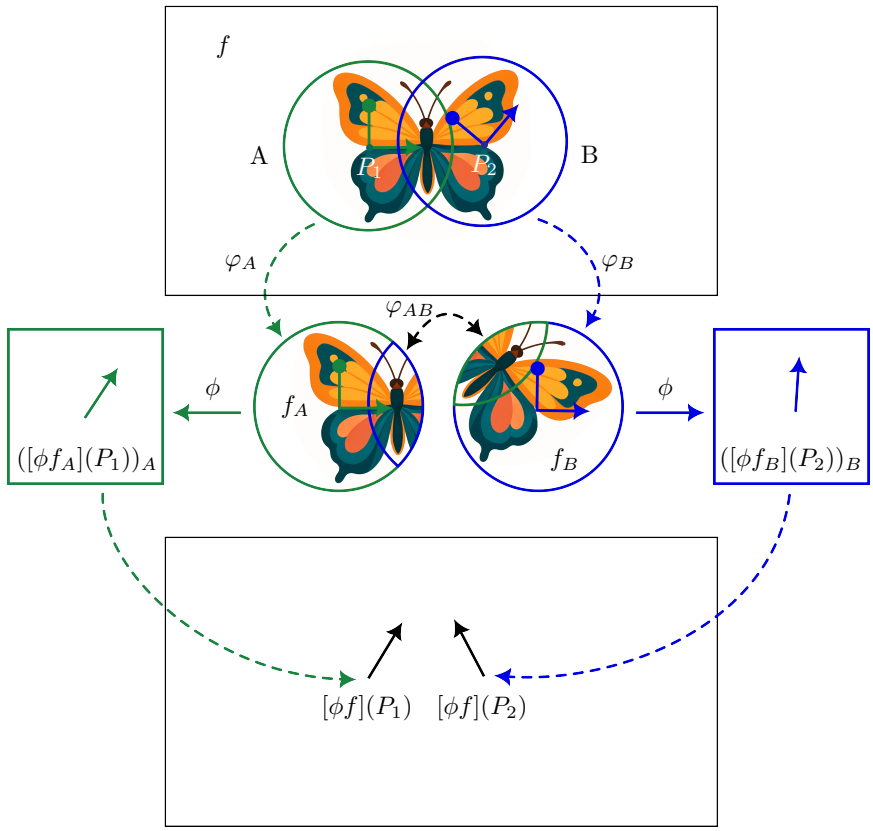


Figure 4.7: Illustration of a passive local transformation — that is, a change in local coordinates — and a coordinate independent network ϕ. *Top panel* shows the data and two different local coordinate charts A and B centred at P_1 and P_2, respectively. *Middle panel* shows the data expressed in these local coordinates — the two circles in the middle — and the application of ϕ in each coordinate chart yielding $[\phi f]_A(P_1)$ and $[\phi f]_B(P_2)$ which are expressed in their respective coordinate systems — the outer squares. *Bottom panel* shows these feature vectors combined with their local basis to yield the geometrical, coordinate free, vectors. Note that if one does not contract the computed feature vectors with their basis to obtain the coordinate free vectors, and instead passes the computed feature map $[\phi f_{U_P}]_{U_P}(P)$ — where (φ_{U_P}, U_P) is a coordinate chart centred at P — to the next layer, one will lose all coordinate independence properties as feature vectors computed in different local coordinates will be mixed.

4.4 Takeaway

In this chapter I have shown some examples of how equivariance is fundamentally connected to weight sharing on a mathematical level. When turning to applications, however, it is not enough to know this high level connection but one must also find an efficient implementation to the task at hand. Due to the wide variety of tasks, data — including how data is represented and stored — and models, most efficient implementations are specific to the task and the intended model design. With that said, there are several published models implementing equivariance for specific cases and data [150, 45, 133, 12, 44, 160, 78] — note that the cited works form by no means an exhaustive list — and these published models should be the first thing to check when approaching a problem where equivariance might be useful.

While equivariance can be very useful, one should also keep in mind the limitations, e.g., the impact of local transformations as discussed in section 4.3. One of those limitations is that equivariant models are, by necessity, less expressive than non-equivariant models as they are restricted to a subset of the hypothesis space and as such if the task, or data, does not have the geometrical inductive bias then one should generally not use equivariant networks. However, for problems where there are theoretical reasons to impose the geometrical inductive bias that equivariance gives, equivariant models can provide a much more parameter and data efficient model compared to non-equivariant models.

5 Framework for non-linear maps

This chapter will briefly introduce and discuss section 4 of **Paper IV** and parts of chapter 5. **Paper IV** deals in general with introducing a framework for non-linear layers between induced representations and putting this new framework into context with the theory presented for linear, convolutional, layers in [29].

5.1 Brief derivation

In the previous chapter we discussed the case of gauge equivariance on a general Riemannian manifold. Another interesting case is when the base space is an G-homogeneous space X. If one picks an arbitrary point $x_0 \in X$ as the origin, then there is a subgroup $H = \mathrm{Stab}_G(x_0) \subset G$ that stabilises x_0. That is, for all $h \in H$ we have $h \rhd x_0 = x_0$. Then one can describe X and the quotient G/H, or more detailed as a principal H-bundle $\pi : G \to G/H$.

If the stabiliser H acts through a representation ρ on some vector space V, then we can induce a G-representation as the left regular representation on the space of functions

$$\mathrm{Ind}_H^G \rho := \left\{ f : G \to V \mid f(gh) = \rho(h^{-1}) f(g) \right\}. \tag{5.1}$$

That is, G acts on $f \in \mathrm{Ind}_H^G \rho$ as $[k \rhd f](g) = f(k^{-1}g)$. This space is the induced representation and, for notational ease, we will denote it as $\mathcal{I}_\rho$. The induced representation was discussed briefly in section 3.1. Specifically, since $\pi : G \to X$ is a principal H-bundle we can use this and the H-action on V to create the associated vector bundle $G \times_\rho V$. We will view data as sections of this associated bundle, which, through corollary 3.2.20 we can work with as

element of the corresponding induced representation.

Maps mapping data to data will then map between induced representations $\Phi : \mathcal{I}_{\rho_{\text{in}}} \to \mathcal{I}_{\rho_{\text{out}}}$. As mentioned was the linear case covered in [29] with some edge cases clarified in [6] but no attempts have been made for a framework covering non-linear maps. In **Paper IV** we introduce such a framework inspired by [29] and based on the integral map defined below.

Definition 5.1.1. For $f \in \mathcal{I}_{\rho_{\text{in}}}$ let $\Phi : \mathcal{I}_{\rho_{\text{in}}} \to \mathcal{I}_{\rho_{\text{out}}}$ be a map defined as

$$[\Phi f](g) = \int_G \omega(f, g, g')\mathrm{d}g', \tag{5.2}$$

where $\omega : \mathcal{I}_{\rho_{\text{in}}} \times G \times G \to V_{\rho_{\text{out}}}$ satisfies

$$\omega(f, gh, g') = \rho_{\text{out}}(h^{-1})\omega(f, g, g'), \quad \forall h \in H. \tag{5.3}$$

Remark 5.1.2. This condition on ω is intuitively the same as the condition in the integral kernel presented in lemma 4.2.6, and is separate from the condition necessary for the group equivariance of Φ.

Remark 5.1.3. Note that in this case we include the Mackey constraint on ω as opposed to section 4.2 where we could derive it. For the biprincipal CNN case we could derive this constraint because the dependence on the feature map f was clearly stated allowing us to use the linear structure and localised feature maps f to extract point wise equalities on the kernel behaviour. Here the dependence on f is a priori unknown and specified by ω which means that our previous tools do not work. Hence we assume the Mackey property eq. (5.3) already in the definition.

The map Φ is universal in that any non-linear map can be represented in this way by some suitable choice of ω.

Theorem 5.1.4 (Theorem 4.5 in **Paper IV**)**.** *Any map $\lambda : \mathcal{I}_\rho \to \mathcal{I}_\sigma$ between induced representations can be written as an integral*

$$\lambda[f](g) = [\Phi f](g) := \int_G \omega(f, g, g')\mathrm{d}g', \tag{5.4}$$

with a suitable choice of $\omega \in \Omega$.

There are some freedom in choosing the integration domain as is elaborated on in remark 5.1.7, however, as it is easier to conceptualise I will continue to use G as the integration domain

Intuitively one can view $\omega(f, g, g')$ as computing some information, or message, based on the input feature map f and the two points $g,\ g'$ in the group G which is then aggregated to g to get the output at g.

Now, the construction of the induced representation yields the natural G-action

$$(kf)(g) = f(k^{-1}g). \tag{5.5}$$

Note that this is essentially the same group action as defined in the principal bundle case with a non-homogeneous base space, eq. (4.23). The main distinction is that due to the homogeneous structure this G-action moves point across the base space.

If we want to make eq. (5.2) equivariant with respect to this action, that is

$$[\Phi(kf)](g) = (k[\Phi f])(g), \tag{5.6}$$

we need to, in analogy to lemma 4.2.7, share this ω in some way over G. In this case the weight sharing appears as

$$\omega(f, g, g') = \omega(k^{-1}f, k^{-1}g, g'), \tag{5.7}$$

for all $k \in G$. Hence, we can choose $k = g$ which yields

$$\omega(f, g, g') = \omega(g^{-1}f, e, g'). \tag{5.8}$$

This allows us to define a reduced map

$$\widehat{\omega}(f, g') := \omega(f, e, g'). \tag{5.9}$$

Finally this results in the following theorem for equivariant non-linear maps:

Theorem 5.1.5 (Non-linear equivariant map)**.** *If the map $\widehat{\omega} : \mathcal{I}_{\rho_{in}} \times G \to V_{out}$ satisfies*

$$\widehat{\omega}(h^{-1}f, g') = \rho_{out}(h^{-1})\widehat{\omega}(f, g'), \qquad \forall h \in H \leqslant G, \tag{5.10}$$

then the map

$$[\Phi f](g) = \int_G \widehat{\omega}(g^{-1}f, g')\mathrm{d}g', \tag{5.11}$$

is a G-equivariant map between the induced representations $\mathcal{I}_{\rho_{in}}$ and $\mathcal{I}_{\rho_{out}}$.

This formulation is general in the space of equivariant non-linear maps.

Theorem 5.1.6 (Theorem 4.12 in **Paper IV**)**.** *Given any equivariant operator*

$\lambda : \mathcal{I}_\rho \to \mathcal{I}_\sigma$ there exist a choice of $\hat{\omega}$ such that $\Phi = \lambda$. This is given by

$$\hat{\omega}(g^{-1}f, g') = \delta(g')\lambda[g^{-1}f](e), \tag{5.12}$$

where $\delta(g')$ is the Dirac delta distribution.

Remark 5.1.7. Note that nowhere in this setting are we really required to use G as the integration domain. We can equally well consider the equivariant map as

$$[\Phi f](g) = \int_X \widehat{\omega}(g^{-1}f, x)\mathrm{d}x, \tag{5.13}$$

where X is any space instead of the form in remark 5.1.7. With that said, however, if we set $X = G$ there is a natural change of integration variable $g' \mapsto g^{-1}\widehat{g}$ since through this change of coordinates we get

$$[\Phi f](g) = \int_G \widehat{\omega}(g^{-1}f, g^{-1}\widehat{g})\mathrm{d}\widehat{g}. \tag{5.14}$$

This form allows for a natural interpretation where $g^{-1}\widehat{g}$ encodes the relative position between where the feature map is centred and the position on which the $\widehat{\omega}$ is conditioned. As we will discuss in section 5.2 this allows for relative positional bias in, e.g., transformer architectures.

Remark 5.1.8. If the integration domain G is compact and the integral is normalised, then one does not need to consider the Dirac delta distribution and simply sets $\widehat{\omega}(g^{-1}f, g') = \lambda[g^{-1}f](e)$. With this one achieves

$$[\Phi f](g) = \int_G \widehat{\omega}(g^{-1}f, g')\mathrm{d}g' = \int_G \lambda[g^{-1}f](e)\mathrm{d}g' = \lambda[g^{-1}f](e)\int_G \mathrm{d}g' = \lambda[f](g). \tag{5.15}$$

Intuitively, if one sees $\widehat{\omega}$ as a some kind of filter bank then the operation in eq. (5.11) can be viewed as the following process: First, centre the feature map at g followed by computing the match of this centred feature map and this filter bank conditioned on the integration variable g'. Note that the filter bank may in itself depend on the centred feature map $g^{-1}f$.

5.2 Instances of this framework

From the previous section we have that if

$$[\Phi f](g) = \int_G \widehat{\omega}(g^{-1}f, g')\mathrm{d}g', \tag{5.16}$$

where $\widehat{\omega} : \mathcal{I}_{\rho_{\text{in}}} \times G \to V_{\text{out}}$ satisfies

$$\widehat{\omega}(hg^{-1}f, g') = \rho_{\text{out}}(h)\widehat{\omega}(g^{-1}f, g'), \quad \forall h \in H, \tag{5.17}$$

then Φ will be equivariant to the left regular action of G on the feature maps:

$$[\Phi(kf)](g) = (k[\Phi f])(g). \tag{5.18}$$

This begs the question of what forms $\widehat{\omega}$ can take and whether any known architectures can fit in this framework. **Paper IV** presents five different known architectures that emerges from different choices of $\widehat{\omega}$, they are summarised in fig. 5.1.

When constructing the $\widehat{\omega}$ to form the layer one can evaluate the g-centred feature map $g^{-1}f$ at either the identity or at g'. Initially, it might seem weird to evaluate at g' as $[g^{-1}f](g') = f(gg')$ which has no clear interpretation. However, as mentioned in remark 5.1.7, if we perform the change of variables $g' \mapsto g^{-1}\widehat{g}$ in the integral so that

$$[\Phi f](g) = \int_G \widehat{\omega}(g^{-1}f, g^{-1}\widehat{g})\mathrm{d}\widehat{g}, \tag{5.19}$$

we get that

$$f(gg') = f(gg^{-1}\widehat{g}) = f(\widehat{g}). \tag{5.20}$$

Hence, with this change of integration variable we see that we can evaluate f at g or at $\widehat{g}$. In addition to evaluating the feature map at different points we can evaluate any other dependencies in the $\widehat{\omega}$ at g' before the change of integration variable, or at $g^{-1}\widehat{g}$ after. The feature to evaluate any other part of $\widehat{\omega}$ at $g^{-1}\widehat{g}$ is what allows relative positional bias, and similar structures to be encoded in this framework.

From this we see that to construct an $\widehat{\omega}$ we have access to the following ingredients:

- the feature map f evaluated at g;
- the feature map f evaluated at integration variable $\widehat{g}$;
- any function ψ evaluated at the relative group element $g^{-1}\widehat{g}$.

These ingredients can be used in any combination as long as eq. (5.17) is satisfied.

In addition to the architectures presented in fig. 5.1 one can easily obtain certain non-linearities used for equivariant models. An example of this is the

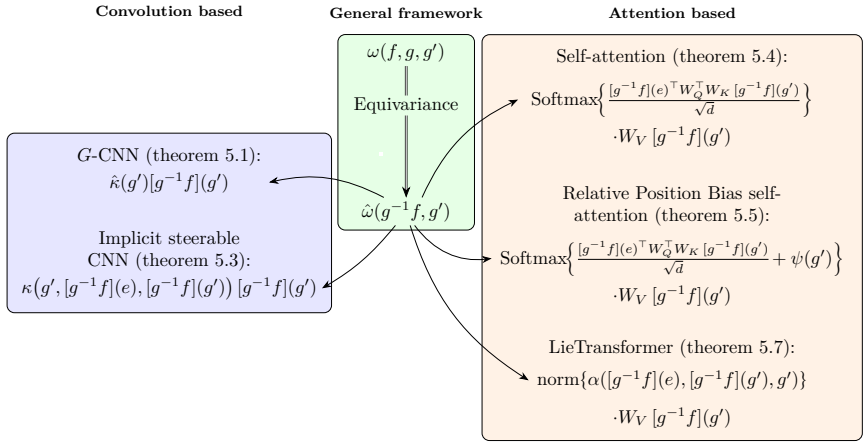


Figure 5.1: Specialisations of the general framework based on the integrand $\hat{\omega} : \mathcal{I}_\rho \times G \to V_\sigma$ to well-established architectures for different choices of the map $\hat{\omega}$. The G-CNN case is obtained when $\hat{\omega}$ factorises in a kernel $\hat{\kappa} : G \to \mathrm{Hom}(V_\rho, V_\sigma)$, which is independent of the input features, and the input feature $g^{-1}f : G \to V_\rho$. The other cases also appears through different factorisations as some type of kernel contracted with the feature $g^{-1}f$. However, in all cases the kernel is also dependent on the input feature $g^{-1}f$, which separates them from the G-CNN case and makes them inherently non-linear in f. Figure is figure 2 in **Paper IV**, all references are to theorems in **Paper IV**.

norm based non-linearity for H being an orthogonal group [150, 118]. This non-linearity scales a vector by a non-linearity acting on the norm of the vector and is obtained by taking

$$\widehat{\omega}(g^{-1}f, g') = \sigma(\|[g^{-1}f](e)\|)[g^{-1}f](e)\delta(g'), \tag{5.21}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is any non-linear function. This works as $\|\rho_{\mathrm{in}}(h^{-1})[g^{-1}f](e)\| = \|[g^{-1}f](e)\|$ In the case where the integration domain is compact with a normalised measure we can discard the $\delta(g')$ factor as mentioned in remark 5.1.8.

5.3 Takeaway

In this chapter I gave a short introduction to the framework for equivariant non-linear maps shown in full in **Paper IV** and showed the ingredients available when constructing a new equivariant layer.

This framework also opens up for a future classification of different architectures depending on the mathematical structure of their layers. If this would result in anything useful is hard to say but it could be an interesting research endeavour.

Part III

Vision applications

6 Data augmentation vs Equivariance

This chapter will discuss benefits and drawbacks of data augmentation contra equivariance in the context of **Paper II**. Specifically **Paper II** investigated how well normal CNNs can perform with data augmentation of SO(3) rotations on spherical images compared to the spherical CNN by Cohen et al. [28]. For the semantic segmentation experiments we extended the available layers for the spherical CNN with a layer projecting a feature map f on the group SO(3) to a feature map on the sphere.

The main motivation for **Paper II** was that, at the time we wrote the paper, it was common to compare manifestly equivariant models to models trained with data augmentation by comparing four modalities: training on normal or transformed data, and then testing on normal or transformed data. This is schematically visualised in table 6.1.

However, these comparisons were almost always made without consideration of how the models would benefit from the fact that datasets expands during data augmentation. Hence there were no studies on the limiting behaviours

Table 6.1: Table 1 from [28] where they compare a planar, normal, CNN to the spherical CNN on the spherical MNIST dataset. The split "X/Y" means trained on "X" and tested on "Y". "R" means that the data in question was randomly rotated, while "NR" stands for non-rotated data. Note that the paper does not go into detail on the relative sizes of the training datasets, and does not investigate how the performance is affected by the dataset size.

	NR/NR	R/R	NR/R
planar	0.98	0.23	0.11
spherical	0.96	0.95	0.94

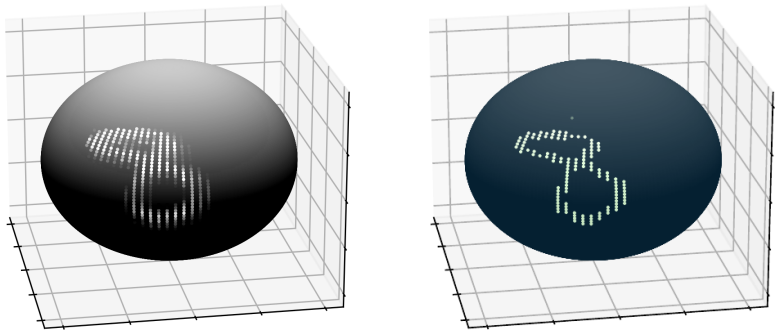


Figure 6.1: Sample from the spherical MNIST dataset used for semantic segmentation. Left: input data. Right: segmentation mask. Figure is figure 1 from **Paper II**.

of models under data augmentation. To fill this knowledge gap we wanted to study if non-equivariant models could reach the performance of the manifestly equivariant models through more data augmentation.

The tasks considered in **Paper II** are classification and semantic segmentation on MNIST. To increase the complexity of the segmentation task multi-digit version of MNIST was made where each data point includes four digits.

6.1 Equivariance

As mentioned before, a map — or neural network — $\mathcal{N}_\theta : X \to Y$ is equivariant if it commutes with the group action on the input and output spaces as

$$\mathcal{N}_\theta(g \rhd_X x) = g \rhd_Y (\mathcal{N}_\theta(x)). \tag{6.1}$$

Formulated differently it means that this diagram commutes

$$\begin{array}{ccc} X & \xrightarrow{\mathcal{N}_\theta} & Y \\ {\scriptstyle g\rhd_X}\downarrow & & \downarrow{\scriptstyle g\rhd_Y} \\ X & \xrightarrow[\mathcal{N}_\theta]{} & Y \end{array}$$

There are several ways one can imbue this property to a network, either through data augmentation or through by making the network manifestly equivariant by its construction. Outside of these methods there are several others. To

mention some, there are *loss guided* methods, which modifies the loss function to guide the network to being equivariant and *canonicalisation* methods [149, 71], which transforms all incoming data to some canonical state, stores the transformation, operates on the canonicalised input, and then transform the result back. These other methods are however out of scope to this thesis and will not be covered here.

6.2 Dataset and augmentation

In **Paper II** we performed classification and semantic segmentation — per pixel classification — on MNIST as well as semantic segmentation on a multi-digit MNIST variant.

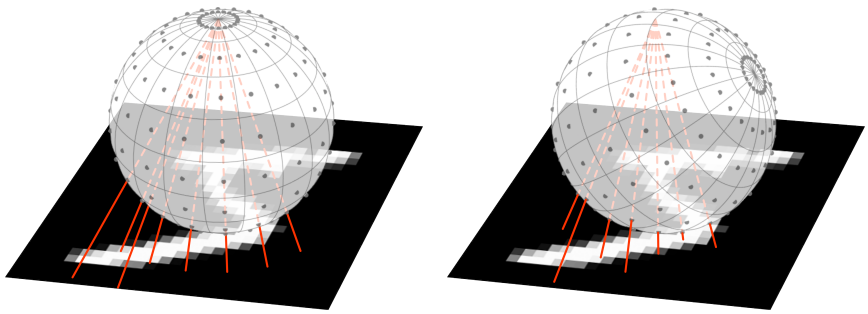
To obtain a sample on the sphere with n MNIST digits we perform the following process: First we select a resolution of the spherical Driscoll-Healy grid, visualised in fig. 2.8, and generate the coordinates of the grid points. Then n digits is randomly placed on a 60×60 canvas below the south pole of the sphere. At this point, if the sample should be rotated, the grid points in the Driscoll-Healy grid are rotated by a random element in SO(3). After an eventual rotation, the grid points in the Driscoll-Healy grid are stereographically projected onto the 60×60 canvas. Finally, after the grid points are projected onto the canvas, the canvas is sampled bilinearly onto the projected grid points. This process is visualised in fig. 6.2.

In the case of semantic segmentation the ground truth mask is sampled onto the projected points by the nearest neighbour. As the classification ground truth, relevant only for canvases with a single digit, is only a single number for the entire sample it is kept unmodified for the spherical version. All grid points on the sphere landing outside the canvas after stereographic projection are given the background value of 0.

6.2.1 Data augmentation

I will first present a general discussion on data augmentation. Let $\mathcal{D}$ be a labelled dataset consisting of pairs $(X_i, Y_i)_{i \in I}$ of input data $X_i \in X$ and the corresponding label $Y_i \in Y$. If there is a well defined G-action on X and Y *the augmented dataset* is usually defined as

$$\widehat{\mathcal{D}} = \{(g \rhd_X X_i, g \rhd_Y Y_i) \mid \forall (X_i, Y_i) \in \mathcal{D},\ \forall g \in G\} = G\mathcal{D}, \tag{6.2}$$



(a) non-rotated projection **(b)** rotated projection

Figure 6.2: Schematic projection of MNIST digit onto the Driscoll-Healy grid. Rays extend from the nort pole of the sphere through the grid points on the sphere. Then the image is sampled where these rays intersect it and this samled value is assigned to the pixel the ray passed through. Note that in our data pipeline the digit is randomly placed in a 60×60 canvas, which, for simplicity, is not shown here.

and is the G-orbit of the original dataset $\mathcal{D}$. While easy to write down, a full augmentation of the dataset in this way is only possible for finite groups as the total dataset size increases by a factor $|G|$. A network that fully learns an augmented dataset will be equivariant on $\widehat{\mathcal{D}}$ since

$$\mathcal{N}_\theta(g \rhd_X X_i) \overset{1}{=} g \rhd_Y Y_i \overset{2}{=} g \rhd_Y (\mathcal{N}_\theta(X_i)), \tag{6.3}$$

where 1 follows from the assumption that the network has learnt the transformed data and 2 from that the network learnt the non-transformed data.

If the group G is not finite it is not possible to perform a full augmentation. In these cases one often performs augmentation live during training where each pair $(X_i, Y_i) \in \mathcal{D}$ is randomly transformed when loaded.

In **Paper II** we augmented the samples as they were loaded during training.

6.3 Models

In this section I briefly introduce the different models used in **Paper II**. For the exact architecture details I refer the reader to the paper and its appendix.

6.3.1 Equivariant model: the spherical CNN

In **Paper II** the equivariant model was a spherical CNN developed by Cohen et al. [28]; denoted S2CNN. This model works with spherical data modelled as functions $f : S^2 \to \mathbb{R}^k$ where k is the number of channels. The group action they consider is $[L_R f](x) = f(R^{-1}x)$ for rotations $R \in \mathrm{SO}(3)$. That is, they only work with scalar fields.

To construct a SO(3)-equivariant network the input function is lifted in the first layer to a function on SO(3) through

$$[\psi \star f](R) = \int_{S^2} \sum_{i=1}^{k} \psi_i(R^{-1}x) f_i(x) \mathrm{d}x, \tag{6.4}$$

where $\psi : S^2 \to \mathbb{R}^k$ is a filter assumed to have local support. In this way $[\psi \star f] : \mathrm{SO}(3) \to \mathbb{R}$ is a function on the entire group SO(3) and so is all features in subsequent layers. To get multiple channels one simply uses multiple filters.

The map between two hidden layers, that is mapping functions on SO(3) to functions on SO(3), is defined as

$$[\psi \star f](R) = \int_{\mathrm{SO}(3)} \sum_{i=1}^{k} \psi_i(R^{-1}Q) f_i(Q) \mathrm{d}Q, \tag{6.5}$$

where $\mathrm{d}Q$ is the left invariant Haar measure on SO(3).

To compute these maps one transforms the features f and the filters ψ to the spectral domain by projecting them onto either the basis of spherical harmonics $Y_m^l(x)$ — for features and filters on S^2 — or onto the basis of Wigner D-matrices $D_{mn}^l(R)$ for features on SO(3). Then the correlation can be computed by simply performing a matrix product between the transformed features $\widehat{f}$ and the transformed filter $\widehat{\psi}$ through

$$\widehat{\psi \star f} = \widehat{f} \cdot \widehat{\psi}^{\dagger}. \tag{6.6}$$

In order to apply any non-linearity $\sigma : \mathbb{R} \to \mathbb{R}$ one first needs to transform the feature map back to the spatial domain, SO(3), by computing

$$[\psi \star f](R) = \sum_{l=0}^{b} (2l+1) \sum_{m=-l}^{l} \sum_{n=-l}^{l} \widehat{\psi \star f}_{mn}^{l} D_{mn}^{l}(R), \tag{6.7}$$

where b is the maximum frequency, called the *bandwidth* and is related to the resolution of the spatial grid. After transforming the features back to a function on SO(3) the non-linearity can be applied point wise. That is, the i:th channel in output f_{out} is computed $f_{\text{out},\,i}(R) = \sigma([\psi_i \star f](R))$ for each filter ψ_i used in the layer.

Through this method one can concatenate several layers where each layer is the following sequence. First lift the input feature map on S^2 to a function on SO(3) through eq. (6.4). Then for each hidden layer one transforms the input function on SO(3) to the spectral domain, perform the convolution as a matrix product, transform the result back to the spatial domain and apply the non-linearity. The final transformations back to the spatial domain is only needed to apply a point wise non-linearity and should be omitted if no non-linearity is used.

The above scheme describes the workings of the encoder, essentially, all layers except the final one. To obtain a useful output f_{out} one needs to adapt the final layer, the decoder, to the task. For the classification task we need SO(3) invariance which can be achieved by integrating the output of the final hidden layer f_{final} over SO(3)

$$f_{\text{out}} = \int_{\text{SO(3)}} f_{\text{final}}(R)\mathrm{d}R. \tag{6.8}$$

However, for the segmentation task we still need an output feature map on S^2 that maintains equivariance to rotations of the initial feature map. The spherical CNN by Cohen et al. [28] did not include such a layer as they were focussed on classification. As such we needed to extend the possible layers with a layer that mapped feature maps on SO(3) to feature maps on S^2.

To obtain this, we note that the sphere S^2 is a homogeneous space and can be viewed as the quotient SO(3)/SO(2). Additionally, let $g_x \in \text{SO(3)}$ be a representative of x in the coset space SO(3)/SO(2). Then there are two options, either the output feature map $f_{\text{out}} : S^2 \to \mathbb{R}^k$ is computed as

$$f_{\text{out}}(x) = \int_{\text{SO(3)}} \kappa(g^{-1}g_x) f_{\text{final}}(g)\mathrm{d}g, \tag{6.9}$$

where κ is SO(2) invariant: $\kappa(gh) = \kappa(g)$ for all $h \in \text{SO(2)}$; or using the projection

$$f_{\text{out}}(x) = \int_{\text{SO(2)}} f_{\text{final}}(g_x h)\mathrm{d}h, \tag{6.10}$$

representing taking an average over the subgroup SO(2). The second approach

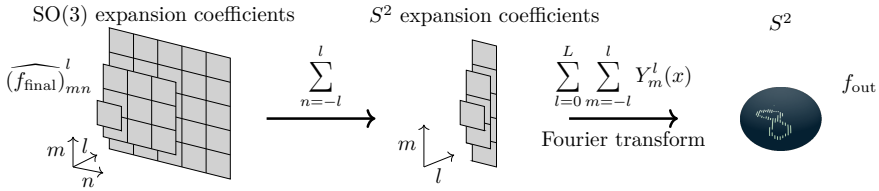


Figure 6.3: Projection of feature map from SO(3) to a feature map on S^2. This projection is an extention of the existing S2CNN. Figure modified from Figure 3 in **Paper II**.

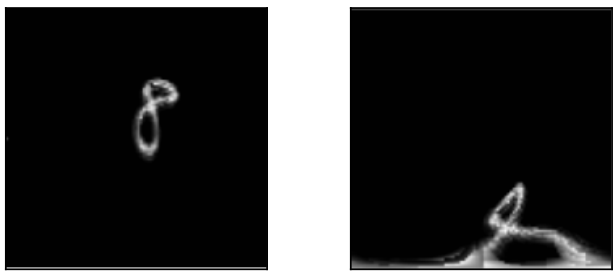


Figure 6.4: Example of the input data in the Driscoll-Healy grid for a digit projected onto the grid centre (left) and onto the pole of the sphere (right). Figure is figure 13 from **Paper II**.

is the one used in **Paper II** and is explicitly computed

$$f_{\text{out}}(x) = \sum_{l=0}^{b} \sum_{m=-l}^{l} \sum_{n=-l}^{l} (f_{\text{final}})^l_{mn} Y^l_m(x). \tag{6.11}$$

This final projection is visualised in fig. 6.3.

6.3.2 Non-equivariant model

The non-equivariant model in **Paper II** is an ordinary CNN. For the CNN to operate on the spherical images in the Driscoll-Healy grid, we naively treat this grid as a flat space. As a consequence, digits close to the poles will be heavily distorted; see fig. 6.4. However, when training on non-rotated images, we counteract this by placing all digits on the equator, minimising the distortions.

Table 6.2: Training times for the S2CNN classification model and non-equivariant CNN model at matched accuracy on rotated spherical images. The spherical CNN model is trained on non-rotated images whereas the CNN is trained on an augmented dataset with rotated images. A single Nvidia T4 16GB was used for training. Table is Table 3 in **Paper II**.

Model	Accuracy	Training time
150k S2CNN	97.64%	15h
5M CNN	97.49%	26h

6.4 Comparison of data augmentation and equivariance

In **Paper II** we studied the performance of manifestly equivariant models and models trained with data augmentation on classification and semantic segmentation tasks. In this section I will briefly recount the observed properties and results.

6.4.1 Classification

For the classification task, we observed that the CNN models could reach the performance of the manifestly equivariant spherical CNN through data augmentation. The metric in question is overall accuracy on the test set. See the left panel of fig. 6.5. However, the training time required for the CNN to learn the augmented data was much longer as is shown in table 6.2.

As the classification task only need the models to output a single class for each input image, it is enough for the model to find some key global feature and use this to classify the image. This gives an interpretation of the performance curves of the left panel of fig. 6.5 as the capability of the models to detect these key features.

6.4.2 Semantic segmentation

For the semantic segmentation task, however, the CNN model saturated well below the performance of the manifestly equivariant spherical CNN, even with data augmentation. See fig. 6.5.

Note that since the background is both a class of its own and a large majority

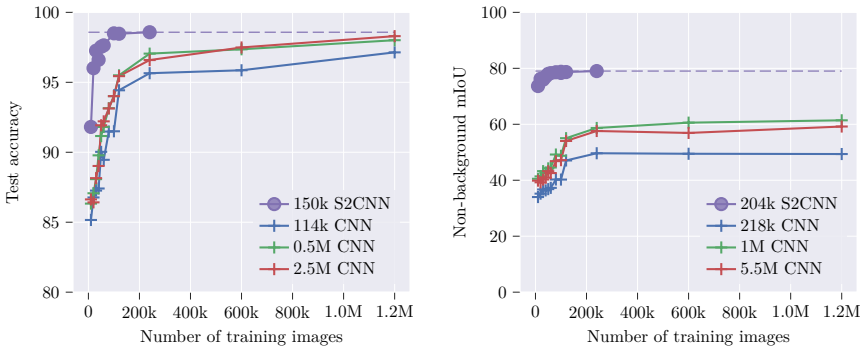


Figure 6.5: *Non-equivariant performance saturation for segmentation.* Left: For classification of spherical MNIST, the non-equivariant models reach the test accuracy of the equivariant models for very large amounts of data augmentation. Right: For semantic segmentation of one-digit spherical MNIST, the non-background mIoU of the non-equivariant models saturates well below the performance of the equivariant model even for moderately high amounts of data augmentation. Figure is figure 10 from **Paper II**.

of the pixels we compute and present the mean IoU for the non-background classes.

For classification task, to achieve good performance it is enough for the models to detect some key feature and use this to assign a single class to the image. This would naively suggest that a similar approach can be taken for single-digit semantic segmentation: Use some key global feature to detect the class and then assign this class to each pixel with intensity above some threshold. This is however not supported when comparing the results between the single-digit semantic segmentation and the multi-digit semantic segmentation.

In the multi MNIST digit experiments we created blank canvases and added four random MNIST digits to the canvas, after a random individual rotation for each digit. Interestingly, each model saturated at a similar performance level for both single-digit and multi-digit semantic segmentation. Compare the right panel of fig. 6.5 showing the single-digit semantic segmentation performance saturation to fig. 6.6 showing the multi-digit version of the same task. If the models performed the naive approach described in the previous paragraph one would expect much worse performance on the multi-digit version, but this is not what we observe. This indicates that the models really rely on local features to perform the segmentation, and that the equivariant model is better at detecting these across different orientations.

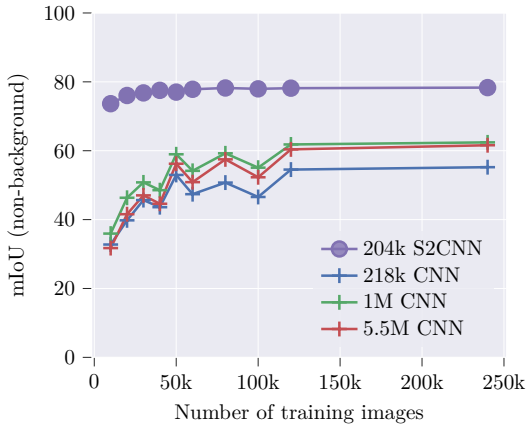


Figure 6.6: *Segmentation on four digit Spherical MNIST.* Performance of equivariant and non-equivariant models in semantic segmentation on four-digit spherical MNIST for various amounts of data augmentation. Figure is figure 7 in **Paper II**.

While the equivariant model is much more data efficient, it is also much slower in processing the inputs, largely due to custom implementation of operations in the Fourier domain as well as frequent transitions back and forth between the spatial and Fourier domain. Detailed times are in table 6.3.

Table 6.3: Runtime and latency throughput for the equivariant spherical CNN (around 204k parameters) and the non-equivariant CNN (around 200k parameters) both on an Nvidia T4 16GB GPU performing semantic segmentation. Latency is the time elapsed for a single forward pass of a batch of images through the model on the GPU, while throughput is the number of samples per second for the given batch size. For both cases, to speed up training and throughput, we chose the largest batch size that fit into the GPU memory. This table is a merge of Table 1 and 2 from **Paper II**.

Model	Batch size	Latency (ms)	Throughput (N/s)
S2CNN	1	111 ± 0.6	9.0 ± 0.04
	7	479 ± 2.2	14.6 ± 0.07
CNN	1	5.93 ± 0.24	169 ± 5.8
	60	87.98 ± 0.17	682 ± 1.3

6.5 Takeaway

The main takeaway from **Paper II** is that one should carefully consider the parameters relevant for your task at hand. Assuming there is some relevant symmetry present in the data or task, consider using an equivariant model if, in rough order of importance: the group action is "complicated" when viewed in your data representation, for example the spherical rotations are complicated when viewed from the plane, see fig. 6.4; you require that the group transformations are respected as much a possible; you do not have access to a lot of data; or your training time is limited.

On the other hand, you should consider a non-equivariant model if, in rough order of importance: the throughput is important, e.g., using a model in time-critical places like autonomous vehicles; the group action is simple for the model to learn without enforcing it; or you have access to a lot of data and training resources. Of course the points regarding the latency and throughput of the model depends heavily on the implementation, and equivariant models which rely on other previously optimised procedures, like convolutions, will suffer less — or not at all — from the additional structure required for equivariance.

7 HEAL-SWIN: a spherical vision transformer

This chapter will discuss elements of **Paper III**, specifically focussing on the motivation behind the HEAL-SWIN model and its construction. For the specific architecture and experimental results I refer the reader to **Paper III** although I will present a short summary of experiments we performed.

7.1 Problem: Images on curved space

There are many tasks where one could make use of machine learning models that are inherently spherical. Some examples include weather prediction [82, 91, 81] and spherical images. Spherical images are the focus of this chapter and appear in many places: any type of localisation using omnidirectional images [166, 162], and fisheye images which is the focus of **Paper III**.

A fisheye image, an example of which is shown in fig. 7.2, is projected on the camera sensor by a lens that is symmetrical with respect to rotations about the optical central axis. Due to this construction, and a large field of view, fisheye images are heavily distorted. One way around this distortions is to map the image to the sphere. That is, assign each pixel to the point on the unit sphere that corresponds to the incoming light ray for that pixel. Through this one obtains an image on the sphere.

Specifically, a fisheye camera is determined by its *mapping function*, f. This map sends an incoming light ray with direction (θ, φ), where θ is the zenithal angle with respect to the optical centre line of the lens and φ is the azimuthal angle, to a point (r, φ) on the camera sensor where $r = f(\theta)$. In this way, the mapping function provides a one-to-one map between pixels in the image and

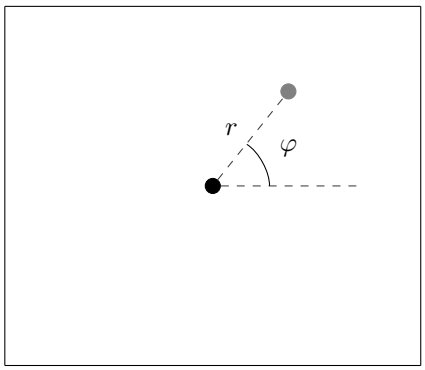


Figure 7.1: Schematic figure of camera sensor where r is the distance from a pixel, marked with a grey dot, to the optical centre line, marked with a black dot, and φ is the azimuthal angle.

Figure 7.2: Example image from the WoodScape dataset [121]. Note the that the distortions increase with the distance to the image centre, which here is the optical axis.

directions of incoming light rays. Figure 7.1 shows a schematic representation of the camera sensor, and fig. 7.3 shows a cross section of a fisheye camera visualising a simplified light path for the incoming light rays.

Since the sphere is a curved surface and the plane is flat there will always be distortions if one represents spherical data on a flat space, e.g. the plane. Unfortunately most developed methods work naturally with flat data and needs to be modified to work on spherical data.

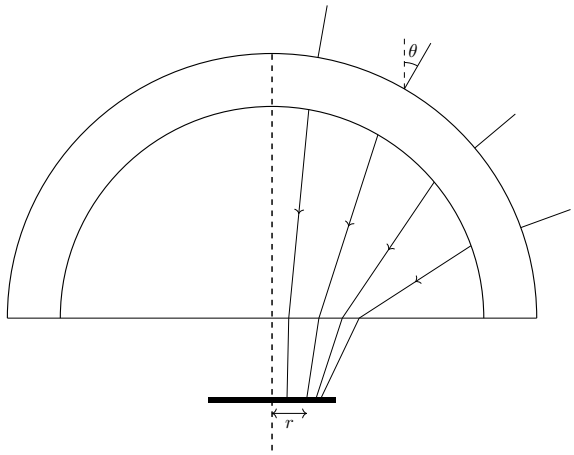


Figure 7.3: Schematic of a fisheye camera with the black rectangle at the bottom representing the camera sensor. The dashed line represents the optical axis and the lens is symmetric to rotations around this axis. The angle θ is the angle of an incoming light ray relative to the optical axis and $r = f(\theta)$ is the distance on the camera sensor between the optical axis and the pixel hit by the light ray with angle θ, given by the mapping function f. Note the compression of the incoming light rays at the edge of the camera sensor.

7.2 Short introduction to vision transformers

Generally people view the start of the deep learning revolution as when the deep machine learning model AlexNet [80] won the ImageNet 2012 challenge; beating the runner up by 10.8 percentage points [43]. The ImageNet challenge consists of classifying the content of an image and AlexNet used predominantly convolutional layers. Since then, networks using mostly convolutional layers have been the go-to for image based tasks.

Although convolutional neural networks have performed well for vision tasks, each convolutional layer acts only locally as the computation of a new feature at a given point only depends on the nearby input features. This means that CNNs can have issues capturing long range interactions [156]. Stacking many layers means that the receptive field increases, but so does also the influx of information from other features making it harder to fine specific relations between distant pixels.

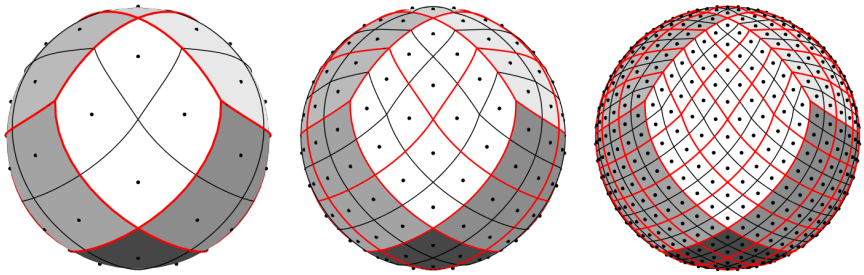
To overcome this limitation one can use the *self-attention* layer which can capture relations over any distance. Roughly speaking this happens as the

self-attention mechanism [10, 153] constructs a so-called *attention matrix* of size $n \times n$, where n is the number of pixels. The value of the attention matrix A_{ij} at (i, j) encodes the degree the attention mechanism "believes" that pixel i is "connected" to pixel j. In this way the attention mechanism can capture relations over any range.

This aside, the attention mechanism has several downsides. One downside is that attention is computationally heavy: the number of operations scales quadratically with respect to the input length [153]. To examplify why this is downside, let us compare input sizes for text and images. When a transformer works with texts, the input text if first split into so-called *tokens* and these tokens are then fed to the attention mechanism. How a text is split into tokens varies between languages, but for English OpenAI states a rule of thumb that one token on average corresponds to 0.75 of a word [161]. OpenAI claims that some of their models can work with up to 128 000 tokens jointly over input and output. This would correspond to around 90 000 words, which is about a shorter average book. For most use cases this context length is totally fine; for images this is not nearly enough since on the image side, at present time, a low resolution image has at least 12 000 000 pixels. If each pixel were to be considered as a token this would be over 90 times more than the current limit of OpenAI's models. Hence, to actually allow transformers to work efficiently on images one needs to deal with this scaling issue.

In 2020 the Vision transformer (ViT) was introduced. The ViT solved the scaling issue by first batching patches of nearby pixels together and embedding these patches as the tokens. This reduced the computational requirements drastically.

Another downside of transformers for images is that transformers have none of the inductive biases that the convolutional networks have. E.g., CNNs are translationally equivariant by construction while transformers do not satisfy this at all. To counteract this transformers generally need to train on a lot more data. However, the SWIN (Shifted WINdow) transformer [94] was released in 2021 and used smaller attention windows, hence reducing computational costs, that are reused at shifted locations. This shifting means that weights going into the computation of the attention matrix is shared over different areas of the input image, akin to how the kernel in a convolutional layer is shared over all points. Additionally, in addition to the shifting, SWIN performs patch merging, similar to pooling used in CNNs, allowing to construct hierarchical features as well as forming long range connections over the image which would otherwise be difficult due to the local attention windows.



(a) $n_{\text{side}} = 2$ **(b)** $n_{\text{side}} = 4$ **(c)** $n_{\text{side}} = 8$

Figure 7.4: The HEALPix grid at different n_{side} values. This also illustrates the hierarchical construction of the HEALPix grid where the grid for the next n_{side} value is obtained through a subdivision of the previous grid. The red outline shows which pixels are subdivided from the same parent pixel in the previous stage. Note that $n_{\text{side}} = 2$ is the first subdivision of the 12 base pixels, here shaded in different shades of grey.

7.3 HEAL-SWIN

In this section I present the underlying parts of the HEAL-SWIN model put forth in **Paper III** leading up to introducing the full HEAL-SWIN model at the end of this section.

Roughly speaking the HEAL-SWIN model joins the SWIN transformer with the HEALPix grid.

7.3.1 The HEALPix grid

The HEAL part of the HEAL-SWIN model represent the use of the HEALPix grid.

This grid was developed by astrophysicists in the late 1990's [54] and the HEALPix name is an abbreviation of *H*ierarchical *E*qual *A*rea iso*L*atitude *Pix*elation. As the "hierarchical" part in the name suggests the HEALPix grid is constructed recursively by repeated subdivision of 12 curvilinear quadrilateral base pixels. Often the resolution of a HEALPix grid is given in terms of the n_{side} parameter where $n_{\text{side}} = 2^k$ and $k = 0, 1, \ldots$ is the number of subdivisions. This subdivision process is shown in fig. 7.4. The resolution of a HEALPix grid, if all 12 base pixels are used, is then only dependent of the subdivision order k, where $k = 0, 1 \ldots$, and is $R_k = 12 \cdot n_{\text{side}}^2 = 12 \cdot 2^{2k}$

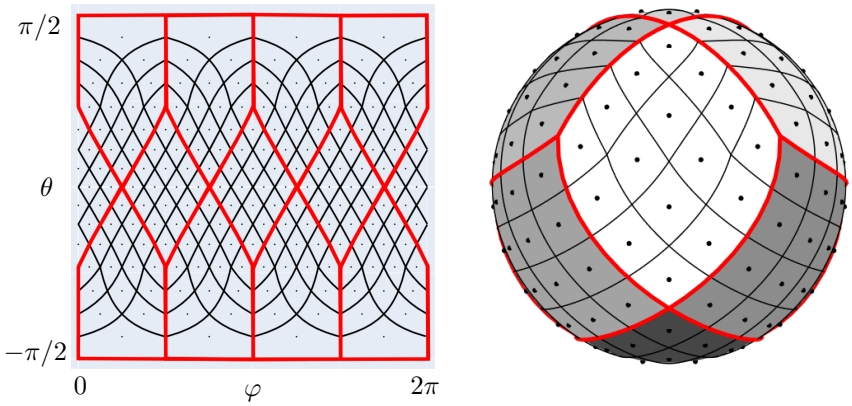


Figure 7.5: The HEALPix grid with $n_{\text{side}} = 4$ and a total resolution of 192 pixels. The 12 base pixels are outlined in red and on the right shaded in different shades of grey. Left panel: The HEALPix grid viewed in the normal angular coordinates on the sphere: zenithal and azimuthal angles. Note that in this view the pixels close to the poles ($\theta = \pi/2$ or $-\pi/2$) appear to take more space, but this is due to the inherent distortions in this coordinate system. Right panel: The HEALPix grid viewed on the sphere. Note that the polar pixels are the same size as any other pixel.

Importantly, each pixel in the HEALPix grid has the same area. This means that every part of the sphere gets equal pixel density and the spatial resolution is uniform. A uniform resolution is highly relevant when one want to resolve small details that might, or will, appear at any position on the sphere, where the Driscoll-Healy grid is more heavily sampled at the poles. Compare fig. 7.5 and fig. 7.6.

Additionally, an important point is that the HEALPix grid cannot easily be stored in any 2-dimensional array. Instead the HEALPix grid is stored in a 1-dimensional index array and use one of two possible representations to map from this index array to coordinates on the sphere: *ring* or *nested*. In the ring representation, fig. 7.7a, the pixels are numbered starting at the north pole and then following the iso-latitude rings, one at a time, down to the south pole. The nested representation on the other hand, fig. 7.7b, utilises the hierarchical structure of the HEALPix grid to number the pixels in a tree like structure where. This one-dimensional nature of how the HEALPix grid is stored means that to adapt models designed for 2-dimensional data one must modify them appropriately. Generally this is done with a mapping which extracts spatially nearby pixels from the index array.

Finally, another advantage that the HEALPix grid has over the Driscoll-Healy

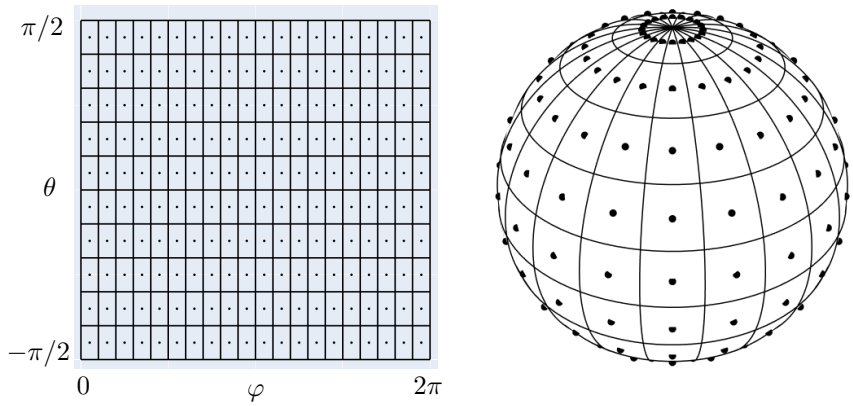
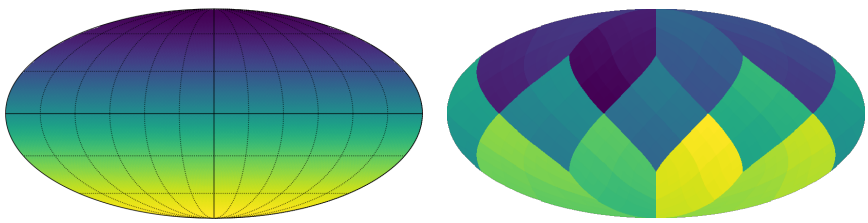


Figure 7.6: Driscoll-Healy with a resolution of 200 pixels. Note that, as opposed to the HEALPix grid in fig. 7.5, the pixels look regular and uniform in the angular view (left panel) while it is clear that the size of the pixels varies with the latitude when viewed on the sphere (right). As a consequence the poles are oversampled as compared to the equator making a uniform spatial resolution impossible.



(a) Ring representation **(b)** Nested representation

Figure 7.7: Different representations of the HEALPix grid, figure taken from the HEALPix documentation . The colour each pixel represents its index value in the different representations. For an explicit example of the indexing in the nested representation, see fig. 7.9. Images modified from [168].

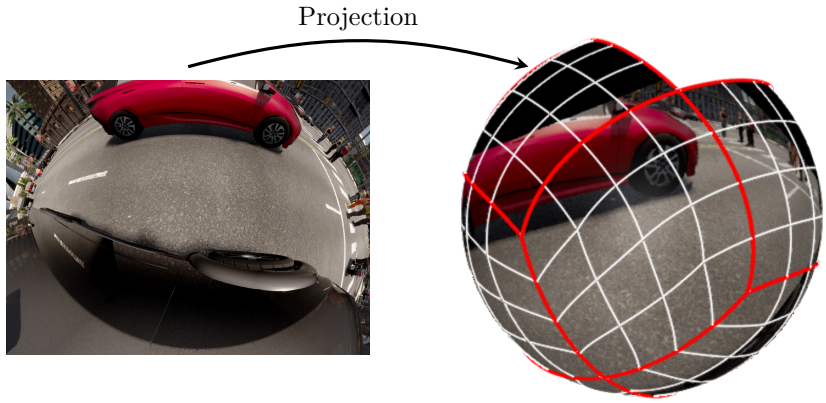


Figure 7.8: Projection of a (synthetic) fisheye image, left, to the sphere, right. Image from the SynWoodScape dataset [137]. Note that the HEALPix grid on the right only uses 8 out of the 12 base pixels.

grid, apart from the equal area sampling, is that it is easy to use only a part of the grid by extracting specific base pixels. To extract base pixel i, zero indexed, in the nested scheme this is simply done by retrieving the indices at positions $n_{\text{side}}^2 \cdot i$ to $n_{\text{side}}^2 \cdot (i+1) - 1$. Of course one would ideally rotate the grid so that the kept base pixels as well as possible with the data. This is used in some of the experiments in **Paper III** in which we use the first 8 base pixels.

To sample an fisheye image onto the HEALPix grid one first maps each pixel, in polar coordinates (r, φ), to the angular coordinates of the incoming light ray, (θ, φ), using the mapping function $\theta = f(r)$. Then the angular positions of the HEALPix sampling points are calculated and the spherical image is resampled onto these points. For RGB data bilinear resampling is reasonable, but for dealing with distance data or semantic segmentation labels using nearest neighbour is better. An example of this is shown in fig. 7.8.

7.3.2 The HEAL-SWIN transformer

The HEAL-SWIN transformer is the modification of the SWIN transformer to work on images (or other data) in the HEALPix grid. This means that the key features of the SWIN transformer must be adapted to work on the 1-dimensional index array of the HEALPix grid. Specifically, three elements needs to be adapted: the patch merging, the attention windows, and the shifting of the attention windows.

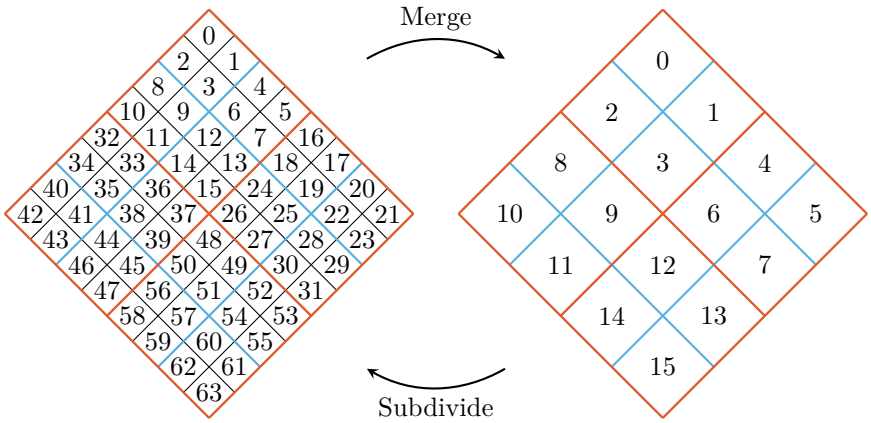


Figure 7.9: Merging and subdivision explicit in the nested grid index scheme. Note that through this scheme one is guaranteed that all pixels sharing the integer part of $i/2^{2k}$, where i is the pixel index and $k \geqslant 1$, is guaranteed to form a square.

Initially nearby pixels in the input high-resolution image on the HEALPix grid is merged into patches — visualised in fig. 7.9 — similar to the ViT, to reduce the overall computational complexity. This merging utilises the nested representation of the HEALPix grid since in this representation pixels are grouped by the parent pixel. Specifically, through this one gets that all pixels $\{p_i\}$ agreeing on

$$\left\lfloor \frac{p_i}{2^{2k}} \right\rfloor, \quad k \geqslant 1, \tag{7.1}$$

where p_i is the pixel index, are guaranteed to form a square in the HEALPix grid with side length 2^k.

The same procedure is used to group these patches into attention windows, specifically, a window is assumed to consist of a number of pixels equal to a power of 4 since this is guaranteed to yield square windows by eq. (7.1). This ensures that the windows align well with the HEALPix grid. After the pixels for a window are gathered attention is performed over these pixels. See fig. 7.10.

Moreover, the patch merging reappear at later stages in the model to again lower the spatial resolution in order to extract more semantic information and spatial relations between pixels further away in the input image. This process is visualised in fig. 7.11 and in fig. 7.9 with explicit indexing in the nested representation.

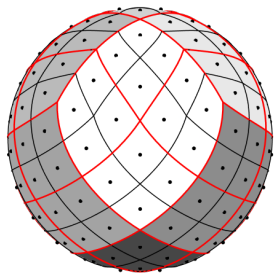


Figure 7.10: Attention windows of size 4, outlined in red, in a HEALPix grid with $n_{\text{side}} = 4$.

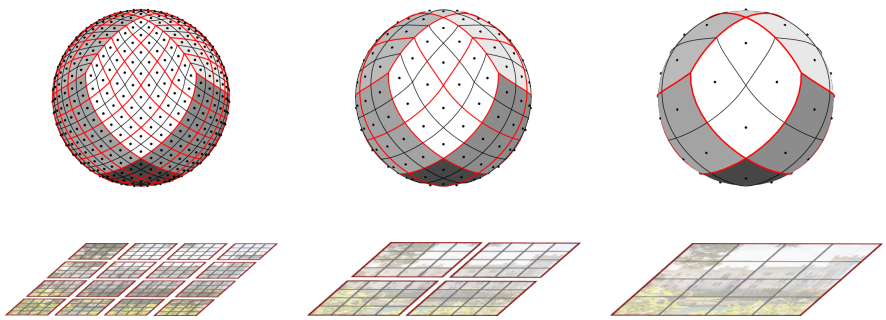


Figure 7.11: Patch merging in HEAL-SWIN (top row) and SWIN (bottom row) from a higher resolution on the left to a lower on the right. In each step the patches in a red outlined window are merged to form the patches in the next step on the right. Note the hierarchical similarities between the patch merging in the HEAL-SWIN to the one in the SWIN.

Finally, to actually share the attention weights over different parts of the image we need to shift the windows. Or equivalently, as we do in the HEAL-SWIN, keep the windows fixed and shift the data. This takes the form of an map $S : I \to I$ where I is the HEALPix index array.

There are two natural shifting strategies connected to the two different representations of the HEALPix array: *spiral* shifting for the ring representation or *grid* shifting for the nested representation. Both shifting schemes are visualised in fig. 7.13. The spiral shifting scheme, fig. 7.13a, is easily implemented as a roll on the index array. This induces what looks like a spiral motion in the image, especially near the poles where the same movement in the index array causes a larger angular movement due to the shorter iso-latitude line compared to close to the equator. The grid shifting on the nested representation, fig. 7.13b, is the closer analogue to the shifting in SWIN which is performed by shifting along the two axes of the grid. In HEAL-SWIN this is a bit more intricate and visualised in fig. 7.12.

In both shifting schemes some areas are shifted "out of the HEALPix grid", e.g., when rolling the index array in the ring representation pixels shifted out at the end of the array. Additionally "empty space" is created at the other end. To keep all data the pixels that are shifted out are copied into the areas lacking data. Doing so naively would allow the attention in the affected windows create connections immediately over the edge of the image essentially changing the topology of the image. To prevent this the copied pixels are masked out in the attention and locality is preserved. The copied, and hence masked out areas, are highlighted in yellow in fig. 7.13.

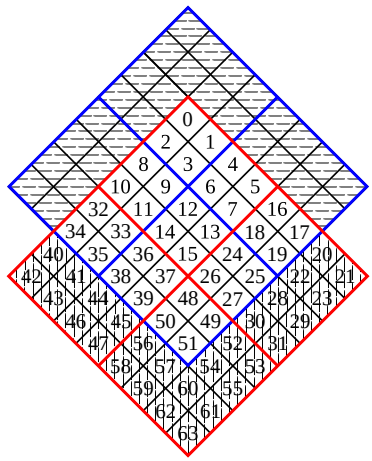
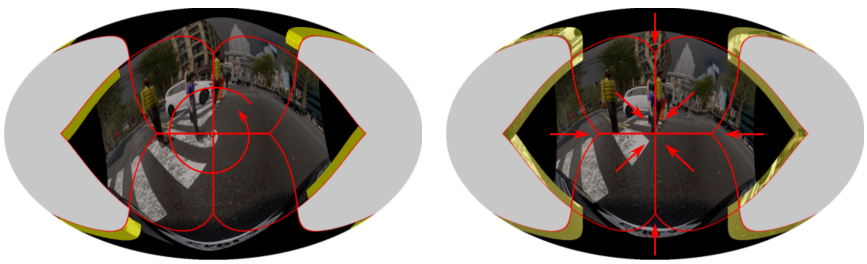


Figure 7.12: Grid shifting scheme for window size 16: The windows before the shift are framed in red and the patches are numbered in the nested scheme. After a shift by half a window size, the patches are divided into the windows framed in blue, so that e.g. patch 0 becomes patch 12 after the shift. The hashed regions are masked in the attention layer. The patches hashed horizontally correspond to the pixels marked in yellow in Figure 7.13 (left). They are filled with patches hashed vertically which correspond to the pixels lost in the centre of Figure 7.13 (left). Figure is figure 3 in **Paper III**.



(a) Spiral shifting **(b)** Grid shifting

Figure 7.13: Different shifting configurations for the HEAL-SWIN model. Note that for these figures only 8 of the 12 base pixels are used, the unused base pixels are filled with grey for this visualisation. The yellow highlighted regions are regions that needs to be filled with the data that is shifted out of the grid. Specifically, in the grid shifting scheme (right) pixels are moved towards the centre, and as they are moved into the centre they are copied into the yellow areas. Note that the shifting is greatly exaggerated here to make it easily visually noticeable. Figure is figure 4 in **Paper III**.

7.4 Experiments

In **Paper III** we performed several experiments using the normal SWIN and our new HEAL-SWIN. Specifically we performed semantic segmentation — per pixel classification task — depth estimation and classification. The semantic segmentation experiments were done on three different datasets: WoodScape [121], SynWoodScape [137], and the Stanford 2D3D-S dataset [5]. The depth estimation experiment was only performed on the SynWoodScape dataset, and the classification was performed on spherical MNIST — see chapter 6 for more information. Note that images from the two WoodScape datasets were projected on only 8/12 of the base pixels to minimise tracking unnecessary background information. The Stanford 2D3D-S dataset, however, is fully omnidirectional, except for the polar regions of the RGB images where data is missing.

To compare the models for the semantic segmentation tasks the SWIN output was projected to the HEALPix before computing the per class mIoU. The HEALPix grid is well-suited for this for two main reasons. First, it is the native domain for fisheye imagery. Second, because the direction of each incoming light ray is inherently known, the representation is ideal for downstream tasks or for building an internal world model. For an overview of the results on the semantic segmentation on SynWoodScape, see fig. 7.15 with more details in **Paper III**.

The depth estimation experiment is a bit different as the model was tasked for an RGB image input to estimate, for each pixel, the distance along the corresponding light ray to its intersection with some object. The ground truth for this consisted of greyscale images where the pixel value corresponded to the ground truth distance. For this experiment we initially tried to perform the same method as for the semantic segmentation, that is resample the SWIN prediction to the HEALPix grid and then perform a distance measurement in that grid. Unfortunately this approach yielded large errors, primarily originating from high contrast edges which are very sensitive for how they are sampled. To circumvent this issue we decided to compare the models outputs and the ground truths as point clouds in 3-dimensional space.

To create these point clouds we utilise that we know the light ray direction for each pixel. Then we create a point in space along that ray at the specified distance — either specified by the ground truth or the prediction. See fig. 7.14. This method avoids the resampling errors that would otherwise affect the predicted distances.

The predicted point cloud is then compared to the ground truth point cloud

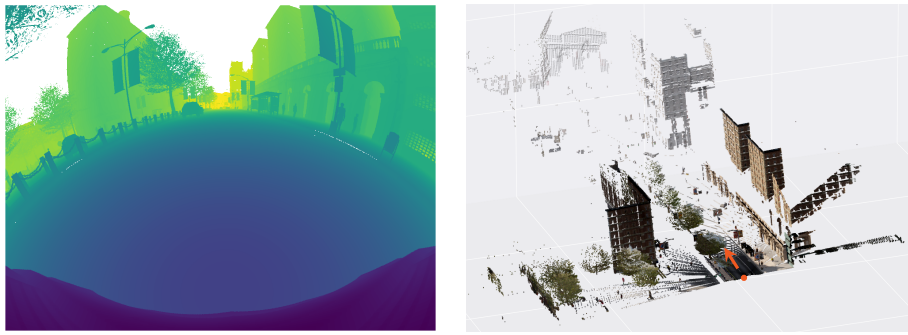


Figure 7.14: Depth-map ground truth (left) and corresponding point cloud (right). The red arrow indicates the orientation of the camera. Figure is figure 7 in **Paper III**.

by computing the so-called *Chamfer distance*:

$$\mathrm{CD}(P_{\mathrm{pred}}, P_{\mathrm{gt}}) = \frac{1}{|P_{\mathrm{pred}}|} \sum_{p \in P_{\mathrm{pred}}} \min_{p' \in P_{\mathrm{gt}}} d(p, p')^2 + \frac{1}{|P_{\mathrm{gt}}|} \sum_{p \in P_{\mathrm{gt}}} \min_{p' \in P_{\mathrm{pred}}} d(p, p')^2 . \tag{7.2}$$

For an overview of the results on the depth estimation, see fig. 7.15 with more details in **Paper III**.

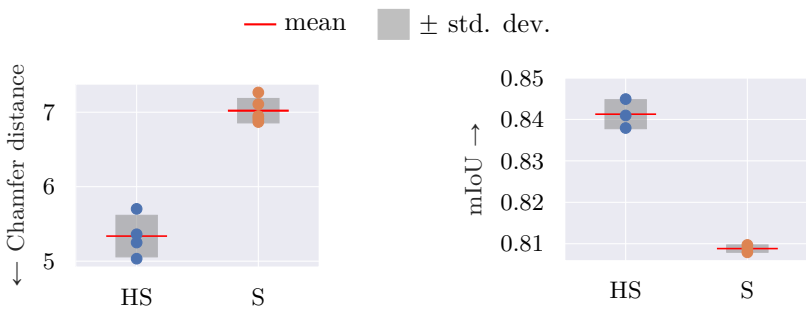


Figure 7.15: Chamfer distance (lower is better) and mIoU (higher is better) for HEAL-SWIN (HS) and SWIN (S). The dataset used for both of these was SynWoodScape. Figure is figure 2 in **Paper III**.

7.5 Takeaway

The takeaway from this paper and project would simply be to carefully think on how you represent your data. Especially, if there are any underlying geometrical inductive biases that you would like to include, think on if there are any ways to represent the data that natively takes this geometry into account. It is easer to let the data representation, in this case the grid, work the geometry for you instead of imposing geometric constraints on a grid that would otherwise destroy that geometry.

Part IV

Conclusions and outlook

8 Conclusions and outlook

In this chapter I present some conclusions from this thesis as well as some paths for future research.

8.1 Conclusions

Many different problems have some geometry or group symmetry present and utilising this as an inductive bias can be very beneficial. On the applied side the benefits can include data efficiency and reduced training time for an equivalent performance — see **Paper II** and chapter 6— or utilising a grid more suitable to the underlying geometry resulting in general increased performance — see **Paper III** and chapter 7.

While one should always be aware of the geometrical inductive biases that can be implemented for a task, using them blindly is not generally advisable. Often equivariant models can have additional overhead which can be problematic if the task the model is used for is time-critical. Additionally, if the group action is easy for the model to learn, which depends on how the data is represented, then using data augmentation — or other methods, like guiding the learning through a modification of the loss function — might be enough and allow for larger expressiveness. Hence one should always consider all parameters of the task when deciding on what type of model to be used.

On the mathematical side this thesis presents a close knit approach to equivariance in CNNs based on differential geometry — see **Paper I** and chapter 4 — and a novel framework for equivariant non-linear maps between induced representations on homogeneous spaces — see **Paper IV** and chapter 5. This shows that it is possible to approach equivariance in a systematic way compatible with many different architectures.

8.2 Outlook

There are a bunch of relevant threads from **Paper IV**. The most immediate is to use this framework to construct new manifestly equivariant layers. Of course, designing new layers could be interesting from a mathematical point of view but for this to be relevant in applications these need also be made efficient. To do this would probably require deep knowledge of CUDA or similar frameworks for coding efficient code on GPUs.

Another track from **Paper IV** is to examine the possibility of feature maps f not just defined on G but with domain $G \times G$. Such maps would natively encode features connecting different group elements and could probably be used to model edge features. From the mathematical side having the domain of the features be $G \times G$ makes sense from the lens of functional analysis. See the appendix of **Paper IV** for more details on the functional analysis version of the framework in **Paper IV**. Additionally, the functional analysis approach to this problem could be elaborated and expanded, there is definitely more to say on that side.

On the applied side some research avenues are: Is it possible to use the geometric structure of the HEALPix grid to implement equivariance? On the data side I would personally be interested in a study in data efficiency for equivariant transformers, and probing the limits of this, along the lines of **Paper II**. Additionally, the same type of project would be interesting for transformers that natively deal with data on curved surfaces: how much data is needed by different approaches to data on curved surfaces in order to reach the same performance, if that is even possible?

Mathematical concepts

Bibliography

[1] Josh Abramson, Jonas Adler, Jack Dunger, et al. "Accurate structure prediction of biomolecular interactions with AlphaFold 3". In: *Nature* 630.8016 (June 2024), pp. 493–500 (cit. on p. 17).

[2] Shun'Ichi Amari. "A Theory of Adaptive Pattern Classifiers". In: *IEEE Transactions on Electronic Computers* EC-16.3 (1967), pp. 299–307 (cit. on p. 14).

[3] Shun'Ichi Amari. "Learning Patterns and Pattern Sequences by Self-Organizing Nets of Threshold Elements". In: *IEEE Transactions on Computers* C-21.11 (1972), pp. 1197–1206 (cit. on p. 15).

[4] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. *Machine Bias: There's software used across the country to predict future criminals. And it's biased against blacks.* ProPublica. May 2016 (cit. on p. 40).

[5] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. *Joint 2D-3D-Semantic Data for Indoor Scene Understanding.* 2017. arXiv: 1702.01105 [cs.CV] (cit. on pp. 34, 125).

[6] Jimmy Aronsson. "Homogeneous Vector Bundles and G-equivariant Convolutional Neural Networks". In: *Sampling Theory, Signal Processing, and Data Analysis* 20.2 (July 11, 2022), p. 10 (cit. on pp. 7, 92).

[7] Jimmy Aronsson. "Mathematical Foundations of Equivariant Neural Networks". PhD thesis. Chalmers University of Technology, May 2023 (cit. on p. 7).

[8] Santiago Andrés Azcoitia, Costas Iordanou, and Nikolaos Laoutaris. *What Is the Price of Data? A Measurement Study of Commercial Data Marketplaces.* 2021. arXiv: 2111.04427 [cs.CY] (cit. on p. 39).

[9] Stefan Baack. "A Critical Analysis of the Largest Source for Generative AI Training Data: Common Crawl". In: *The ACM Conference on Fairness, Accountability, and Transparency (FAccT).* FAccT '24. June 2024, pp. 2199–2208 (cit. on p. 41).

[10] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *3rd International Conference on Learning Representations (ICLR)*. 2015 (cit. on p. 116).

[11] Bowen Baker, Joost Huizinga, Leo Gao, et al. *Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation*. 2025. arXiv: 2503.11926 [cs.AI] (cit. on p. 43).

[12] Simon Batzner, Albert Musaelian, Lixin Sun, et al. "E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials". In: *Nature Communications* 13.1 (May 2022), p. 2453 (cit. on pp. 5, 90).

[13] Ivan Belcic. *What is a generative model?* https://www.ibm.com/think/topics/generative-model. 2024 (cit. on p. 22).

[14] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. "Reconciling modern machine-learning practice and the classical bias-variance trade-off". In: *Proc Natl Acad Sci USA* 116.32 (July 2019), pp. 15849–15854 (cit. on p. 27).

[15] Mike Bird. *Picture of a car*. https://www.pexels.com/ (cit. on p. 23).

[16] Lawrence Breen. *Notes on 1- and 2-gerbes*. Nov. 2006. arXiv: math/0611317 [math.CT] (cit. on p. 77).

[17] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. 2021. arXiv: 2104.13478 [cs.LG] (cit. on pp. 4, 6, 28).

[18] Michael M. Bronstein, Joan Bruna, Taco S. Cohen, and Petar Veličković. *GDL Course*. https://geometricdeeplearning.com/lectures/. 2021 (cit. on p. 28).

[19] Michael M. Bronstein, Joan Bruna, Yann LeCun, et al. "Geometric Deep Learning: Going beyond Euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (July 2017), pp. 18–42 (cit. on p. 28).

[20] Rodney Brooks. *[FoR&AI] Machine Learning Explained*. https://rodneybrooks.com/forai-machine-learning-explained/. Aug. 2017 (cit. on p. 14).

[21] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, et al. "The secret sharer: evaluating and testing unintended memorization in neural networks". In: *Proceedings of the 28th USENIX Conference on Security Symposium*. SEC'19. USA: USENIX Association, 2019, pp. 267–284 (cit. on p. 41).

[22] Nicholas Carlini, Florian Tramer, Eric Wallace, et al. *Extracting Training Data from Large Language Models*. 2021. arXiv: 2012.07805 [cs.CR] (cit. on p. 41).

[23] Carnegie Learning. *The State of AI in Education 2025: Key Findings from a National Survey*. Tech. rep. Accessed: 2025-05-15. Carnegie Learning, 2025 (cit. on p. 37).

[24] T. Ceccherini-Silberstein, A. Machí, F. Scarabotti, and F. Tolli. "Induced Representations and Mackey Theory". In: *Journal of Mathematical Sciences* 156.1 (Jan. 1, 2009), pp. 11–28 (cit. on p. 6).

[25] Ismail Celik, Muhterem Dindar, Hanni Muukkonen, and Sanna Järvelä. "The Promises and Challenges of Artificial Intelligence for Teachers: a Systematic Review of Research". In: *TechTrends* 66.4 (July 2022), pp. 616–630 (cit. on p. 36).

[26] Xieyuanli Chen, Benedikt Mersch, Lucas Nunes, et al. "Automatic Labeling to Generate Training Data for Online LiDAR-Based Moving Object Segmentation". In: *IEEE Robotics and Automation Letters* 7.3 (July 2022), pp. 6107–6114 (cit. on p. 39).

[27] Bobby Chesney and Danielle Citron. "Deep Fakes: A Looming Challenge for Privacy, Democracy, and National Security". In: *California Law Review* 107 (2019), pp. 1753–1820 (cit. on pp. 41, 42).

[28] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. "Spherical CNNs". In: *6th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2018 (cit. on pp. 11, 101, 105, 106).

[29] Taco S. Cohen, Mario Geiger, and Maurice Weiler. "A General Theory of Equivariant CNNs on Homogeneous Spaces". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019 (cit. on pp. 6, 7, 11, 71, 84, 91, 92).

[30] Taco S. Cohen and Max Welling. *Group Equivariant Convolutional Networks*. 2016. arXiv: `1602.07576 [cs.LG]` (cit. on p. 71).

[31] European Commission, European Education, and Culture Executive Agency. *AI report*. By the European Digital Education Hub's Squad on artificial intelligence in education. Publications Office of the European Union, 2023 (cit. on p. 37).

[32] Daniel Crevier. *AI: The Tumultuous History of the Search for Artificial Intelligence*. BasicBooks, 1993 (cit. on pp. 13–17).

[33] Alex Davies, Petar Veličković, Lars Buesing, et al. "Advancing mathematics by guiding human intuition with AI". In: *Nature* 600.7887 (Dec. 2021), pp. 70–74 (cit. on p. 36).

[34] Jia Deng, Wei Dong, Richard Socher, et al. "ImageNet: A large-scale hierarchical image database". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255 (cit. on p. 17).

[35] Dev Desai, Shiv V Kantliwala, Jyothi Vybhavi, et al. “Review of AlphaFold 3: Transformative Advances in Drug Design and Therapeutics”. In: *Cureus* (July 2024) (cit. on p. 17).

[36] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. “An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: International Conference on Learning Representations. Oct. 2, 2020 (cit. on p. 4).

[37] J.R. Driscoll and D.M. Healy. “Computing Fourier Transforms and Convolutions on the 2-Sphere”. In: *Advances in Applied Mathematics* 15.2 (1994), pp. 202–250 (cit. on pp. 32, 33).

[38] Duncan Evans. *Meta executives confirm tech giant scraping public data since 2007*. https://www.news.com.au/. 2024 (cit. on p. 41).

[39] Adobe Express. *How ChatGPT is changing the way we search*. `https://www.adobe.com/express/learn/blog/chatgpt-as-a-search-engine`. 2025 (cit. on p. 41).

[40] Francesco Farina and Emma Slade. “Data efficiency in graph networks through equivariance”. In: *International Conference on Machine Learning (ICML), Workshop on Subset Selection in Machine Learning: From Theory to Practice*. 2021 (cit. on p. 5).

[41] Alhussein Fawzi, Matej Balog, Aja Huang, et al. “Discovering faster matrix multiplication algorithms with reinforcement learning”. In: *Nature* 610.7930 (Oct. 2022), pp. 47–53 (cit. on p. 36).

[42] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. *Testing the Manifold Hypothesis*. 2013. arXiv: `1310.0425 [math.ST]` (cit. on p. 25).

[43] Li Fei-Fei, Jia Deng, Olga Russakovsky, et al. *Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)*. `https://image-net.org/challenges/LSVRC/2012/results.html`. 2012 (cit. on p. 115).

[44] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. “Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data”. In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research (PMLR). July 2020, pp. 3165–3176 (cit. on p. 90).

[45] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. “SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 1970–1981 (cit. on p. 90).

[46] Kunihiko Fukushima. “Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position”. In: *Biological Cybernetics* 36.4 (Apr. 1980), pp. 193–202 (cit. on pp. 4, 15).

[47] Kunihiko Fukushima. “Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements”. In: *IEEE Transactions on Systems Science and Cybernetics* 5.4 (1969), pp. 322–333 (cit. on p. 4).

[48] Zijun Gao and Yan Sun. *Statistical Inference for Generative Model Comparison*. 2025. arXiv: `2501.18897 [stat.ML]` (cit. on p. 22).

[49] Mario Geiger and Tess Smidt. *e3nn: Euclidean Neural Networks*. 2022. arXiv: `2207.09453 [cs.LG]` (cit. on p. 5).

[50] Michael Gerlich. “AI Tools in Society: Impacts on Cognitive Offloading and the Future of Critical Thinking”. In: *Societies* 15.1 (2025) (cit. on p. 37).

[51] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, et al. “Neural Message Passing for Quantum Chemistry”. In: *Proceedings of the 34th International Conference on Machine Learning, (ICML)*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1263–1272 (cit. on p. 4).

[52] Ethan Goh, Robert Gallo, Jason Hom, et al. “Large Language Model Influence on Diagnostic Reasoning: A Randomized Clinical Trial”. In: *JAMA Network Open* 7.10 (Oct. 2024) (cit. on p. 37).

[53] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016 (cit. on pp. 18, 19, 25).

[54] K. M. Górski and et al. “Analysis issues for large CMB data sets”. In: *Evolution of Large Scale Structure : From Recombination to Garching*. Jan. 1999, p. 37 (cit. on p. 117).

[55] K. M. Górski, E. Hivon, A. J. Banday, et al. “HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere”. In: *Astrophysical Journal* 622 (Apr. 2005), pp. 759–771 (cit. on pp. 33, 35).

[56] Ryan Greenblatt, Carson Denison, Benjamin Wright, et al. *Alignment faking in large language models*. 2024. arXiv: `2412.14093 [cs.AI]` (cit. on p. 43).

[57] A. Grothendieck. “SGA 7 Exposé VII: Biextensions de faisceaux de groupes”. In: *Groupes de Monodromie en Géométrie Algébrique*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1972, pp. 133–217 (cit. on p. 77).

[58] F. Gursoy and I. A. Kakadiaris. “Equal Confusion Fairness: Measuring Group-Based Disparities in Automated Decision Systems”. In: *IEEE International Conference on Data Mining Workshops (ICDMW)*. 2022, pp. 137–146 (cit. on p. 40).

[59] Thilo Hagendorff. “Deception abilities emerged in large language models”. In: *Proceedings of the National Academy of Science* 121.24 (June 2024) (cit. on p. 43).

[60] Niv Haim, Gal Vardi, Gilad Yehudai, et al. “Reconstructing Training Data From Trained Neural Networks”. In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., 2022, pp. 22911–22924 (cit. on p. 41).

[61] Mark Hamilton. *Mathematical gauge theory*. en. 1st ed. Universitext. Cham, Switzerland: Springer International Publishing, Jan. 2018 (cit. on p. 55).

[62] Mark Hamilton. *Mathematical gauge theory*. 1st ed. Universitext. Springer International Publishing, Jan. 2018 (cit. on p. 63).

[63] J J Hopfield. “Neural Networks and Physical Systems with Emergent Collective Computational Abilities”. In: *Proceedings of the National Academy of Sciences of the United States of America* 79.8 (Apr. 1982), pp. 2554–2558. pmid: 6953413 (cit. on p. 15).

[64] John Michael Hoppe, Matthias K Auer, Anna Strüven, et al. “ChatGPT With GPT-4 Outperforms Emergency Department Physicians in Diagnostic Accuracy: Retrospective Analysis”. In: *Journal of Medical Internet Research* 26 (July 2024), e56110 (cit. on p. 37).

[65] Addison Howard, inversion, and Lars Bratholm. *Predicting Molecular Properties*. https://kaggle.com/competitions/champs-scalar-coupling. Kaggle. 2019 (cit. on p. 30).

[66] International Energy Agency. *Energy and AI*. Tech. rep. Licence: CC BY 4.0. Paris: International Energy Agency, 2025 (cit. on pp. 36, 38, 39).

[67] Gordon James and Martin Liebeck. *Representations and Characters of Groups*. Cambridge University Press, Oct. 2001 (cit. on p. 47).

[68] Alexandra Jonker and Alice Gomstyn. *What is AI alignment?* https://www.ibm.com/think/topics/ai-alignment. 2024 (cit. on p. 42).

[69] John Jumper, Richard Evans, Alexander Pritzel, et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873 (Aug. 2021), pp. 583–589 (cit. on p. 17).

[70] Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, et al. "Equivariance with Learned Canonicalization Functions". In: *International Conference on Machine Learning (ICML)*. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 15546–15566 (cit. on p. 5).

[71] Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, et al. "Equivariance with Learned Canonicalization Functions". In: *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202. Proceedings of Machine Learning Research (PMLR). July 2023, pp. 15546–15566 (cit. on p. 103).

[72] Anthony W Knapp. *Representation theory of semisimple groups*. Princeton Mathematical Series. Princeton, NJ: Princeton University Press, Jan. 2002 (cit. on p. 47).

[73] Shoshichi Kobayashi and Katsumi Nomizu. *Foundations of differential geometry: V. 1*. Tracts in Pure & Applied Mathematics. Nashville, TN: John Wiley & Sons, Jan. 1966 (cit. on p. 55).

[74] Shoshichi Kobayashi and Katsumi Nomizu. *Foundations of differential geometry: V. 2*. en. Tracts in Pure & Applied Mathematics. Nashville, TN: John Wiley & Sons, Jan. 1969 (cit. on p. 55).

[75] Kathlén Kohn, Thomas Merkh, Guido Montúfar, and Matthew Trager. "Geometry of Linear Convolutional Networks". In: *SIAM Journal on Applied Algebra and Geometry* 6.3 (2022), pp. 368–406 (cit. on p. 6).

[76] Ivan Kolar, Peter W. Michor, and Jan Slovak. *Natural Operations in Differential Geometry*. Springer Science & Business Media, Mar. 9, 2013. 440 pp. (cit. on p. 66).

[77] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. "Clebsch-Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS)*. 2018, pp. 10138–10147 (cit. on p. 31).

[78] Risi Kondor and Shubhendu Trivedi. "On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. Vol. 80. Proceedings of Machine Learning Research (PMLR). 2018, pp. 2752–2760 (cit. on pp. 71, 90).

[79] Oleg Kovalevskiy, Juan Mateos-Garcia, and Kathryn Tunyasuvunakool. "AlphaFold two years on: Validation and impact". In: *Proceedings of the National Academy of Sciences* 121.34 (2024) (cit. on p. 17).

[80] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012 (cit. on pp. 17, 115).

[81] Thorsten Kurth, Shashank Subramanian, Peter Harrington, et al. "FourCastNet: Accelerating Global High-Resolution Weather Forecasting Using Adaptive Fourier Neural Operators". In: *Proceedings of the Platform for Advanced Scientific Computing Conference*. PASC '23. Association for Computing Machinery, 2023 (cit. on p. 113).

[82] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, et al. "Learning skillful medium-range global weather forecasting". In: *Science* 382.6677 (2023), pp. 1416–1421 (cit. on pp. 38, 113).

[83] Y. LeCun, B. Boser, J. S. Denker, et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551 (cit. on p. 4).

[84] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444 (cit. on p. 15).

[85] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]* 2 (2010). Available: `http://yann.lecun.com/exdb/mnist` (cit. on p. 24).

[86] John Lee. *Introduction to Smooth Manifolds*. en. 2nd ed. Graduate Texts in Mathematics. New York, NY: Springer, May 2012 (cit. on p. 55).

[87] Marc F. Lensink, Guillaume Brysbaert, Nessim Raouraoua, et al. "Impact of AlphaFold on structure prediction of protein complexes: The CASP15-CAPRI experiment". In: *Proteins: Structure, Function, and Bioinformatics* 91.12 (Oct. 2023), pp. 1658–1683 (cit. on p. 17).

[88] Pengfei Li, Jianyi Yang, Mohammad A. Islam, and Shaolei Ren. *Making AI Less "Thirsty": Uncovering and Addressing the Secret Water Footprint of AI Models*. 2025. arXiv: `2304.03271 [cs.LG]` (cit. on p. 39).

[89] Yi-Lun Liao and Tess Smidt. *Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs*. Feb. 27, 2023. arXiv: `2206.11990 [physics]`. (Visited on 06/26/2023). Pre-published (cit. on p. 31).

[90] James Lighthill. *A Report on Artificial Intelligence*. London: UK Science Research Council, 1973 (cit. on p. 15).

[91] Hampus Linander, Christoffer Petersson, Daniel Persson, and Jan E. Gerken. *PEAR: Equal Area Weather Forecasting on the Sphere*. 2025. arXiv: `2505.17720 [cs.LG]` (cit. on p. 113).

[92] Seppo Linnainmaa. “The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors”. University of Helsinki, 1970 (cit. on p. 15).

[93] Yang Liu, Jiahuan Cao, Chongyu Liu, et al. *Datasets for Large Language Models: A Comprehensive Survey*. 2024. arXiv: 2402.18041 [cs.CL] (cit. on p. 39).

[94] Ze Liu, Yutong Lin, Yue Cao, et al. “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 9992–10002 (cit. on p. 116).

[95] George W. Mackey. “Induced Representations of Locally Compact Groups II. The Frobenius Reciprocity Theorem”. In: *Annals of Mathematics* 58.2 (1953), pp. 193–221. JSTOR: 1969786 (cit. on p. 6).

[96] George W. Mackey. “On Induced Representations of Groups”. In: *American Journal of Mathematics* 73.3 (1951), pp. 576–592 (cit. on p. 6).

[97] Chloe Martin-King, Ali Nael, Louis Ehwerhemuepha, et al. *CIL:55790, Homo sapiens*. Dataset, Cell Image Library (CIL). 2024 (cit. on p. 30).

[98] Nestor Maslej, Loredana Fattorini, Erik Brynjolfsson, et al. *The AI Index 2023 Annual Report*. Annual Report. Stanford, CA: AI Index Steering Committee, Institute for Human-Centered AI, Stanford University, Apr. 2023 (cit. on p. 42).

[99] John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon. “A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955”. In: *AI Magazine* 27.4 (Dec. 2006), p. 12 (cit. on p. 14).

[100] Warren S. McCulloch and Walter Pitts. “A Logical Calculus of the Ideas Immanent in Nervous Activity”. In: *Bulletin of Mathematical Biophysics* 5 (1943), pp. 115–133 (cit. on p. 14).

[101] C. Mears, T. Lee, L. Ricciardulli, et al. *RSS Cross-Calibrated Multi-Platform (CCMP) 6-Hourly Ocean Vector Wind Analysis on 0.25° Grid*. Tech. rep. Version 3.0. Santa Rosa, CA: Remote Sensing Systems, 2022 (cit. on p. 66).

[102] Alexander Meinke, Bronson Schoen, Jérémy Scheurer, et al. *Frontier Models are Capable of In-context Scheming*. 2025. arXiv: 2412.04984 [cs.AI] (cit. on p. 43).

[103] Fengchun Miao and Mutlu Cukurova. *AI Competency Framework for Teachers*. Tech. rep. United Nations Educational, Scientific and Cultural Organization (UNESCO), Aug. 2024 (cit. on p. 37).

[104] Fengchun Miao, Kelly Shiohira, and Natalie Lao. *AI Competency Framework for Students*. Tech. rep. United Nations Educational, Scientific and Cultural Organization (UNESCO), Aug. 2024 (cit. on p. 37).

[105] Marvin Minsky, ed. *Semantic Information Processing*. Cambridge, MA: The MIT Press, 1968 (cit. on p. 13).

[106] Marvin Minsky and Seymour Papert. *Perceptrons*. en. London, England: MIT Press, June 1969 (cit. on p. 14).

[107] Milad Nasr, Nicholas Carlini, Jonathan Hayase, et al. *Scalable Extraction of Training Data from (Production) Language Models*. 2023. arXiv: 2311.17035 [cs.LG] (cit. on p. 41).

[108] Kieran Nehil-Puleo, Co D. Quach, Nicholas C. Craven, et al. "E(n) Equivariant Graph Neural Network for Learning Interactional Properties of Molecules". In: *The Journal of Physical Chemistry B* 128.4 (Feb. 2024), pp. 1108–1117 (cit. on p. 5).

[109] Alexander Novikov, Ngâln Vũ, Marvin Eisenberger, et al. *AlphaEvolve: A coding agent for scientific and algorithmic discovery*. Google DeepMind. May 2025 (cit. on pp. 26, 36).

[110] OECD. *AI, data governance and privacy: Synergies and areas of international co-operation*. Tech. rep. 22. Paris: OECD Publishing, 2024 (cit. on p. 41).

[111] OECD. *Assessing Potential Future Artificial Intelligence Risks, Benefits and Policy Imperatives*. Tech. rep. 27. Paris: OECD Publishing, 2024 (cit. on pp. 36, 38, 41–43).

[112] OECD. *Intellectual property issues in artificial intelligence trained on scraped data*. Tech. rep. 33. Paris: OECD Publishing, 2025 (cit. on p. 41).

[113] Tycho F. A. van der Ouderaa, David W. Romero, and Mark van der Wilk. *Relaxing Equivariance Constraints with Non-stationary Continuous Filters*. 2022. arXiv: 2204.07178 [cs.LG] (cit. on p. 31).

[114] Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. "A Decomposable Attention Model for Natural Language Inference". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*. The Association for Computational Linguistics, 2016, pp. 2249–2255 (cit. on p. 4).

[115] Stefanos Pertigkiozoglou, Evangelos Chatzipantazis, Shubhendu Trivedi, and Kostas Daniilidis. *Improving Equivariant Model Training via Constraint Relaxation*. 2025. arXiv: 2408.13242 [cs.LG] (cit. on p. 31).

[116] Phil Pope, Chen Zhu, Ahmed Abdelkader, et al. "The Intrinsic Dimension of Images and Its Impact on Learning". In: *International Conference on Learning Representations (ICLR) Spotlight Track*. Jan. 2021 (cit. on p. 24).

[117] Phillip Pope, Chen Zhu, Ahmed Abdelkader, et al. "The Intrinsic Dimension of Images and Its Impact on Learning". In: *9th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2021 (cit. on p. 25).

[118] Adrien Poulenard and Leonidas J. Guibas. "A Functional Approach to Rotation Equivariant Non-Linearities for Tensor Field Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Computer Vision Foundation / IEEE, 2021, pp. 13174–13183 (cit. on p. 96).

[119] ProPublica. *Machine Bias: compas-analysis*. `https://github.com/propublica/compas-analysis/tree/master`. 2017 (cit. on p. 40).

[120] Ali Rahimi and Ben Recht. *Reflections on Random Kitchen Sinks*. `https://archives.argmin.net/2017/12/05/kitchen-sinks/`. 2017 (cit. on p. 6).

[121] Saravanabalagi Ramachandran, Ganesh Sistu, John McDonald, and Senthil Yogamani. *Woodscape Fisheye Semantic Segmentation for Autonomous Driving – CVPR 2021 OmniCV Workshop Challenge*. 2021. arXiv: `2107.08246` (cit. on pp. 114, 125).

[122] Simon Rea. *Lecture notes from Fredric Schuller's Geometric anatomy of theoretical physics*. `https://github.com/sreahw/schuller-geometric`. 2017 (cit. on p. 55).

[123] Marco Tulio Ribeiro, Sameer Singh, and Carlos 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*. KDD '16. Association for Computing Machinery, 2016, pp. 1135–1144 (cit. on p. 40).

[124] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (Sept. 1951), pp. 400–407 (cit. on p. 14).

[125] F. Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". In: *Psychological Review* 65.6 (1958), pp. 386–408 (cit. on p. 14).

[126] Frank Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. VG-II96-G-8. Cornell University, Mar. 15, 1961 (cit. on p. 14).

[127] Joseph J. Rotman. *An Introduction to the Theory of Groups*. Springer New York, 1995 (cit. on p. 47).

[128] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Representations by Back-Propagating Errors". In: *Nature* 323.6088 (Oct. 1986), pp. 533–536 (cit. on p. 15).

[129] Stuart Russell and Peter Norvig. *Artificial intelligence: A modern approach, global edition*. en. 4th ed. London, England: Pearson Education, May 2021 (cit. on pp. 14, 18).

[130] Zahra Sadeghi, Roohallah Alizadehsani, Mehmet Akif CIFCI, et al. "A review of Explainable Artificial Intelligence in healthcare". In: *Computers and Electrical Engineering* 118 (2024), p. 109370 (cit. on p. 37).

[131] Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, et al. *Explainable AI: Interpreting, explaining and visualizing deep learning*. en. Cham, Switzerland: Springer Nature, Sept. 2019 (cit. on p. 37).

[132] A. L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM Journal of Research and Development* 3.3 (July 1959), pp. 210–229 (cit. on p. 14).

[133] Víctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. "E(n) Equivariant Graph Neural Networks". In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research (PMLR). 2021, pp. 9323–9332 (cit. on p. 90).

[134] Rylan Schaeffer, Zachary Robertson, Akhilan Boopathy, et al. "Double Descent Demystified: Identifying, Interpreting & Ablating the Sources of a Deep Learning Puzzle". In: *The Third Blogpost Track at ICLR 2024*. 2024 (cit. on p. 27).

[135] Mike Schaekermann, Graeme Beaton, Minahz Habib, et al. "Understanding Expert Disagreement in Medical Data Analysis through Structured Adjudication". In: *Proc. ACM Hum.-Comput. Interact.* 3.CSCW (Nov. 2019) (cit. on p. 39).

[136] Jürgen Schmidhuber. "Deep Learning in Neural Networks: An Overview". In: *Neural Networks* 61 (Jan. 1, 2015), pp. 85–117 (cit. on p. 15).

[137] Ahmed Rida Sekkat, Yohan Dupuis, Varun Ravi Kumar, et al. "SynWoodScape: Synthetic Surround-view Fisheye Camera Dataset for Autonomous Driving". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2022 (cit. on pp. 120, 125).

[138] Vahid Shahverdi, Giovanni Luca Marchetti, and Kathlén Kohn. "On the Geometry and Optimization of Polynomial Convolutional Networks". In: *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*. Vol. 258. Proceedings of Machine Learning Research. PMLR, May 2025, pp. 604–612 (cit. on p. 6).

[139] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. "Self-Attention with Relative Position Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics, June 2018, pp. 464–468 (cit. on p. 4).

[140] David Silver, Aja Huang, Chris J. Maddison, et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (Jan. 2016), pp. 484–489 (cit. on pp. 17, 25).

[141] David Silver, Julian Schrittwieser, Karen Simonyan, et al. "Mastering the game of Go without human knowledge". In: *Nature* 550.7676 (Oct. 2017), pp. 354–359 (cit. on pp. 25, 39).

[142] David Silver and Richard S. Sutton. *Welcome to the Era of Experience*. Google DeepMind. 2025 (cit. on pp. 26, 40).

[143] Herbert A Simon. *The sciences of the artificial*. The MIT Press, 2019 (cit. on p. 16).

[144] Grace Solomonoff. *The Meeting of the Minds That Launched AI - IEEE Spectrum*. URL: `https://spectrum.ieee.org/dartmouth-ai-workshop` (visited on 01/28/2025) (cit. on p. 14).

[145] Stephen Bruce Sontz. *Principal bundles*. en. 2015th ed. Universitext. Cham, Switzerland: Springer International Publishing, Apr. 2015 (cit. on p. 55).

[146] Steering Committee for Human Rights in the fields of Biomedicine and Health (CDBIO). *Report on the application of artificial intelligence in healthcare and its impact on the 'patient–doctor' relationship*. Tech. rep. CDBIO(2023)7 Rev3. Council of Europe, Sept. 2024 (cit. on p. 37).

[147] Aneeta Sylolypavan, Derek Sleeman, Honghan Wu, and Malcolm Sim. "The impact of inconsistent human annotations on AI driven clinical decision making". en. In: *NPJ Digit Med* 6.1 (Feb. 2023), p. 26 (cit. on p. 39).

[148] Katarzyna Szmyd and Ewelina Mitera. "The Impact of Artificial Intelligence on the Development of Critical Thinking Skills in Students". In: *European Research Studies* XXVII.Issue 2 (2024), pp. 1022–1039 (cit. on p. 37).

[149] Kai Sheng Tai, Peter Bailis, and Gregory Valiant. "Equivariant Transformer Networks". In: *Proceedings of the 36th International Conference on Machine Learning, (ICML)*. Vol. 97. Proceedings of Machine Learning Research (PMLR). 2019, pp. 6086–6095 (cit. on p. 103).

[150] Nathaniel Thomas, Tess Smidt, Steven Kearnes, et al. *Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds*. 2018. arXiv: `1802.08219 [cs.LG]` (cit. on pp. 90, 96).

[151] Loring W. Tu. *Introductory Lectures on Equivariant Cohomology*. Princeton University Press, 2020 (cit. on p. 67).

[152] U.S. Department of Education, Office of Educational Technology. *Artificial Intelligence and Future of Teaching and Learning: Insights and Recommendations*. Tech. rep. Washington, DC: U.S. Department of Education, Office of Educational Technology, 2023 (cit. on p. 37).

[153] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*. 2017, pp. 5998–6008 (cit. on p. 116).

[154] Jason Villanueva. *Picture of coffee mug*. `https://www.pexels.com/` (cit. on p. 8).

[155] Rui Wang, Elyssa F. Hofgard, Hang Gao, et al. "Discovering Symmetry Breaking in Physical Systems with Relaxed Group Convolution". In: *41st International Conference on Machine Learning (ICML)*. OpenReview.net, 2024 (cit. on p. 31).

[156] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. "Non-local Neural Networks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7794–7803 (cit. on p. 115).

[157] Christopher Watkins. "Learning from Delayed Rewards". 1989 (cit. on p. 17).

[158] Silvan Weder, Hermann Blum, Francis Engelmann, and Marc Pollefeys. "LABELMAKER: Automatic Semantic Label Generation from RGB-D Trajectories". In: *International Conference on 3D Vision (3DV)*. IEEE, 2024, pp. 334–343 (cit. on p. 39).

[159] Maurice Weiler, Patrick Forré, Erik Verlinde, and Max Welling. *Equivariant and Coordinate Independent Convolutional Networks. A Gauge Field Theory of Neural Networks*. 2023 (cit. on pp. 71, 83–85, 87).

[160] Maurice Weiler, Mario Geiger, Max Welling, et al. "3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS)*. 2018, pp. 10402–10413 (cit. on p. 90).

[161] *What Are Tokens and How to Count Them? | OpenAI Help Center*. https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them (cit. on p. 116).

[162] N. Winters, J. Gaspar, G. Lacey, and J. Santos-Victor. "Omni-directional vision for robot navigation". In: *Proceedings IEEE Workshop on Omnidirectional Vision (Cat. No.PR00704)*. 2000, pp. 21–28 (cit. on p. 113).

[163] David Woodruff and Taisuke Yasuda. "Sharper Bounds for ℓ_p Sensitivity Sampling". In: *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202. Proceedings of Machine Learning Research. PMLR, July 2023, pp. 37238–37272 (cit. on p. 6).

[164] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. "Harmonic Networks: Deep Translation and Rotation Equivariance". In: *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*. IEEE Computer Society, 2017, pp. 7168–7177 (cit. on p. 71).

[165] Yuzhi Xu, Daqian Bian, Cheng-Wei Ju, et al. "Pretrained E(3)-equivariant message-passing neural networks with multi-level representations for organic molecule spectra prediction". In: *npj Computational Materials* 11.1 (July 2025), p. 203 (cit. on p. 5).

[166] Yasushi YAGI. "Omnidirectional Sensing and Its Applications". In: *IEICE TRANSACTIONS on Information* E82-D.3 (Mar. 1999), pp. 568–579 (cit. on p. 113).

[167] Andrew Zhao, Yiran Wu, Yang Yue, et al. *Absolute Zero: Reinforced Self-play Reasoning with Zero Data*. 2025. arXiv: 2505.03335 [cs.LG] (cit. on p. 40).

[168] Andrea Zonca, Leo Singer, Daniel Lenz, et al. *healpy tutorial*. https://healpy.readthedocs.io/en/latest/tutorial.html. 2021 (cit. on p. 119).

[169] Andrea Zonca, Leo Singer, Daniel Lenz, et al. "healpy: equal area pixelization and spherical harmonics transforms for data on the sphere in Python". In: *Journal of Open Source Software* 4.35 (Mar. 2019), p. 1298 (cit. on p. 35).

[170] 福島 邦彦. "位置ずれに影響されないパターン認識機構の神経回路モデル　ネオコグニトロン ". In: 電子情報通信学会論文誌 *A* J62-A.10 (Oct. 25, 1979), pp. 658–665 (cit. on p. 15).