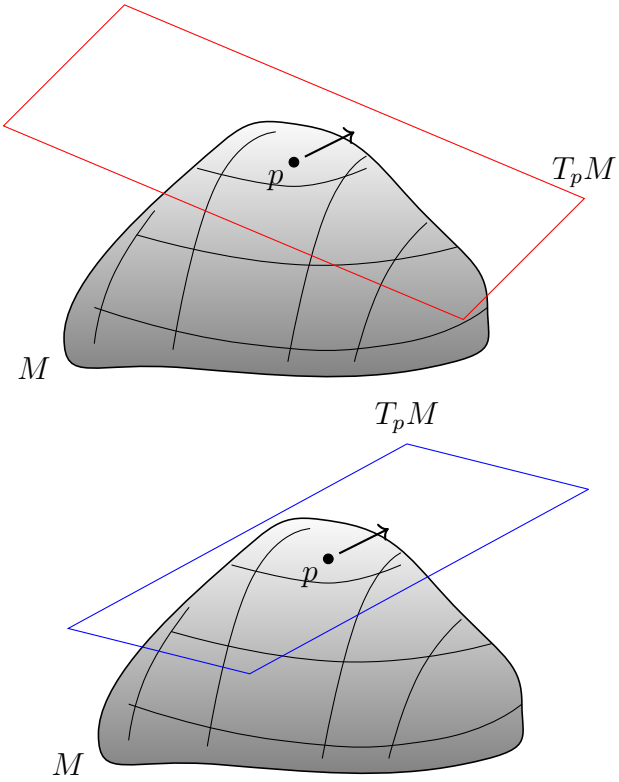
CHALMERS
UNIVERSITY OF TECHNOLOGY



Gauge equivariant convolutional neural networks

Master's thesis in Physics and Astronomy

OSCAR CARLSSON

Department of Mathematics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020:57118

Gauge equivariant convolutional neural networks

OSCAR CARLSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Algebra and Geometry
Chalmers University of Technology
Gothenburg, Sweden 2020

Gauge equivariant convolutional neural networks
OSCAR CARLSSON

Supervisor: Daniel Persson, Department of Mathematical Sciences, Algebra and Geometry
Industrial supervisor: Christoffer Petersson
Examiner: Robert Berman, Department of Mathematical Sciences, Algebra and Geometry

Master's Thesis 2020:57118
Department of Mathematical Sciences
Division of Algebra and Geometry
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Invariance of a tangent vector to a manifold under change of coordinates in the tangent space. See Chapter 3 for more details.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

Gauge equivariant convolutional neural networks
OSCAR CARLSSON
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

In this thesis we present a review of the current theory of group and gauge equivariant convolutional neural networks on homogeneous spaces and general smooth manifolds, with focus on the latter, formulated from a mathematical viewpoint. We also provide a new interpretation of layers in neural networks as maps between associated bundles. Furthermore we discuss the implementation of simple convolutional neural networks invariant under 90° rotations and reflections, build such networks, and test them to show the effect of the invariant construction. This testing shows that the addition of the group invariant structure allows the network to efficiently classify transformed data while only training on untransformed data.

Keywords: Convolutional neural networks, machine learning, manifolds, group, gauge, Python, Tensorflow, Keras.

Acknowledgements

Firstly, I would like to express my gratitude to my supervisor Daniel Persson for the support given during the work on this thesis as well as introducing me to this area combining mathematics, physics, and coding.

Secondly, I would like to direct thanks to Jan Gerken, Christoffer Petersson, Fredrik Ohlsson, and Jimmy Aronsson for the many interesting and enlightening discussions we have had.

Furthermore, I'm deeply grateful for the interest my family have expressed as well as the help they have given, be it feedback on language or allowing me to pester them with discussions.

Lastly, a big thanks to all my friends for providing support, knowingly or not, and willingness to discuss things, both large and small, and helping me to grow as a person.

Also, none of this work, or my academic career thus far, would be possible without the privilege of free higher education for Swedish citizens and the economic support by the Swedish Board of Student Finance.

Oscar Carlsson, Gothenburg, June 2020

Contents

List of Figures

List of Tables

1
Introduction

In this chapter we present a historical background, some context on current research, and how this ties in to the topics covered in this thesis. Furthermore we state the goals and main result presented, and end on a short description of the outline of the thesis.

1.1 Background and context

Understanding of the human brain and its efficiency has long been a goal in some areas of science and technology, and consciousness has been an ever persistent question for philosophers.

On the 1st of October 1950 an article by Alan Turing called "Computing Machinery and Intelligence"[1] was published in which he asks the monumental question "Can machines think?". This paper was foundational for the developments of what we call computers today and the question is still just as relevant. Already the following year the first machine that could update its parameters on its own, SNARC, was constructed by Marvin Minsky and Dean Edmonds [2]. Since then, the advancement of self learning machines has followed the development of computers, and today the amount of resources and time spent on research and development in this field is bigger than ever.

Artificial intelligence (AI) and machine learning (ML) is nowadays used in almost everything that uses a computer; from analysing data from CERN [3], analysing building energy consumption [4], and learning to play games, e.g. Go [5], to recognising and classifying objects in images.

One of the areas most covered in media have been the advancement of self driving vehicles. Since these vehicles need to build a model of their surroundings from data — be it radar, ordinary images, or other sources — and it would be impossible to cover every possible case if coding it by hand, machine learning is a very good option.

In order to fully construct a local map a self driving car needs to be able to recognise objects — "It is a/an <insert object>!" — and isolate where they are in images from a background — "It is there!". In the machine learning language one calls the task of recognising objects *classifying* and the task to isolate objects from a background is called *segmentation*. Immediately one comes to two closely related conclusions illustrated in Figure 1.1 and 1.2: if I want to recognise, e.g., a "5" then it should not matter where in the image the "5" is, and if I want to isolate where the "5" is from the background then if the "5" moves, i.e. translates, my choice of

Figure 1.1: This is an example of invariance: it is a "5" no matter where it is.

Figure 1.2: This is an example of equivariance: the selection of the "5" moves with it. Another way to think about this is that it does not matter if we find the "5" first and then move it, or if we first move the "5" and then find it.

"It is there!" should move with it. These two properties are very central and are called *invariance* — when it doesn't matter where the "5" is, it is still a "5" — and *equivariance* — when we want to know where it is and this needs to follow the "5".

Both of these properties are very appreciated in a self driving car: one would not want a traffic sign to turn into a tree just because it is not in its usual place, and the car should be able to follow a pedestrian over a crossing without losing track of them.

All of this is already in heavy usage, primarily due to *convolutional neural networks* (CNN) [6] which have the equivariance property built into their construction and can also be made invariant as well. Another common type of network are *fully connected* networks [7] — also called *feedforward* neural networks — and these can be trained somewhat to be invariant but this requires them to train on every possible object position beforehand which CNNs do not; this is called *data augmentation*. Mathematical details about CNNs and follu connected networks are covered in Chapter 2.

The careful reader might now want to ask: "Moving around the "5" is all well and good, but what about rotation? It's still a "5" even if it's rotated.". This is indeed very true, and the set of all translations and rotations in the plane together form a symmetry group of the plane and by the reasoning above we would like to achieve networks both invariant and equivariant under these symmetries.

Such networks have been designed in several different ways, one approach is using Gaussian kernels and complex convolutions as presented by Worrall et al. [8], and another approach, taken by Cohen et al. in a paper from 2016 [9], uses the 90° rotational symmetry of a square grid to create networks invariant under rotations of 90°. The second approach will be the basis for the experimental part of this thesis.

All of this is thus far concerning only flat "ordinary" images, but there also exists a plethora of non-flat data that this equivariance and invariance framework would be very useful for. A clear example is an image from a fisheye camera used in autonomous cars in order to increase the angular awareness of the car. In extreme cases a single camera can take images covering a bit more than a hemisphere. Of course we would like our networks to be able to identify and track objects if they move around and as such we need invariance and equivariance with respect to the symmetry of the hemisphere: rotating around an axis down its centre. But if the car would be tilted for some reason this should not change how the network operates so what we really need is equivariance and invariance of networks operating on fully spherical data.

This enters into an area called *geometric deep learning*, which covers all cases where the data is not defined on a nice flat grid, i.e. non-Euclidean data. LeCun et al. published a paper in 2017 [10] in which they give an overview on applicable theory for non-Euclidean data, e.g. data defined on graphs and on arbitrary surfaces, as well as some possible methods for implementations.

Thus far we have only discussed scenarios where the image content is moving around in the image and this is an active transformation — we actively move the object — but if we were to describe the object in a coordinate system we could just as well have transformed the coordinate system instead of moving the object, and in the end we would have the same coordinate representation of the object; that would be a passive transformation. This shift of philosophy, from active to passive transformations, allows for further generalisations.

When one talks about a global coordinate system one could just as well use a coordinate system in each point with the condition that all coordinate systems are the same. Then a global coordinate transformation would be the action of the transformation on each individual coordinate system such that they all transform together. Taking the viewpoint of a coordinate system in each point gives the question: would they all need to be the same coordinate system? The answer to this is no. On a general curved manifold one cannot assign a global coordinate system and hence one must use local coordinates.

In physics one heavily relies on the assumption that whatever coordinate system one chooses the laws of physics should work the same. For example, if two people are standing at different positions and setting up the same experiment only rotated in relation to each other they should get the same result. This is exactly the same as if one would use two different coordinate systems in two different points to describe an image; it looks different from each coordinate system, but the content is the same.

The next logical step is to ask if it then would be possible to change the coordinates locally, and by the reasoning above this is a very reasonable thing to strive after. In the physics example: if you rotate your experiment where you are it should not affect the result. In physics this principle has a special name: *The principle of covariance*, and is one of the most fundamental building blocks when constructing theories. This is illustrated in Figure 1.3. The principal of covariance is also one of the assumptions made by Einstein when developing the theory of general relativity and gives the possibility to always choose a locally flat coordinate system, which in turn gives rise to the fundamental and well known *principle of equivalence*. Furthermore,

in physics one calls the local change of coordinates as *change of gauge* — this refers to much more than just local coordinate changes in physics but that usage will suffice for us — and hence a network that would be equivariant to local coordinate changes would be a *gauge equivariant network*.

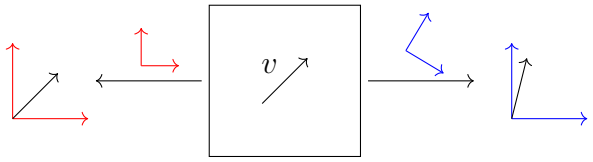


Figure 1.3: A schematic illustration of the *Principle of covariance*. The vector v looks different when described in the left (red) basis or the right (blue) basis, but it's still the same vector.

1.2 Thesis goal and main results

The goal of this thesis will be to study the theory of gauge equivariant convolutions on general smooth manifolds presented by Cheng et al. in a paper from 2019 [11], filling in the details and presenting this in a coherent way to ease future formulations of the theory in the language of differential forms.

This thesis also aims at presenting the theory for the case of networks equivariant and invariant under discrete group transformations such as those used by Cohen et. al. in [9].

As one of its main results, this thesis introduces a new way to view layers in a neural networks as part of a map between associated bundles. In this setting equivariance of the layer becomes equivalent to the map between the associated bundles being well defined on the equivalence classes. Furthermore we present a detailed description of the convolution proposed by Cheng et al. [11] and make some notes on its similarities to the theory used for homogeneous spaces in [9, 12].

On the experimental side we present a working example of a rotation invariant network used for classifying MNIST digits [13] by using code published by Cohen et al. on their `GitHub` [14, 15]. Furthermore, we do thorough tests on this to show the impact of the rotation invariance.

1.3 Thesis outline

The thesis has the following structure. In Chapter 2 we present neural networks, specifically CNNs, in a mathematical way and proceed to show equivariance of the standard convolutional layer with respect to translation as well as derive some constraints that the kernel will need to obey for the convolution to be equivariant also under rotations.

Chapter 3 will continue with introducing data on more general surfaces and presents a new viewpoint on layers in a neural network as maps between associated bundles where well definedness will require that the feature data transforms in an equivariant

way. Furthermore, in this chapter we also present the theory of Cheng et al. [11] where we fill in mathematical details not presented in their paper.

In Chapter 4 we will present the theory of Cohen et al. for the case of networks equivariant and invariant under discrete rotations and mirrorings of the plane. In addition we also present several tests of such networks to show the effect of this additional structure.

The thesis ends with Chapter 5 in which the results are discussed and future research directions and outlooks are presented and discussed.

The thesis contains several appendices with important content.

In Appendix A we present all underlying mathematical structure that the thorough reader that is not well versed in manifolds, differential geometry, and fibre bundles might find useful.

In Appendix B we present the code we used to run the tests, which results are presented in Chapter 4.3.1.

Appendix C contains tables of testing results that were not necessary in the main result and discussion to show the observed effects.

Appendix D lists all used definitions to help the reader find the appropriate definitions.

2

Introduction to fully connected and convolutional neural networks

In this chapter we aim to give a more technical introduction to fully connected and convolutional neural networks in the setting when the task is classification, as well as the concept of equivariance. For a general review of machine learning the interested reader can check out [16]. The definitions presented in this chapter are heavily inspired by that review.

2.1 A neural network: definition and training

A neural network $\mathcal{N}$ designed for a classification task takes some input data x and usually outputs an array of length C, where C denotes the number of different types of objects the network should classify. Each position in the array corresponds to a type of object and the network's output at each position is its confidence that the input is of that type. Hence the output array must sum to 1. Each instance of a network has some set of parameters θ that can be adjusted to change the network's action on the input data. This leads us to make the following definition.

Definition 2.1.1 (Neural network and terminology) A *neural network* N is a map $N : \Theta \times I \to O_N \subset \mathbb{R}^C$ where Θ is the parameter space, I is the space of inputs, such that the output $N(\theta, x) = N_\theta(x)$ in the output space O_N satisfies

$$\sum_k (N_\theta(x))_k = 1 \tag{2.1}$$

where $(N_\theta(x))_k$ denotes the k^{th} element of the output. This must hold for all inputs $x \in I$ and parameter choice $\theta \in \Theta$. Neural networks often have a layered structure so that the network can be decomposed into a sequence of maps between spaces

$$I \xrightarrow{L_0^{\theta_0}} V_1 \xrightarrow{L_1^{\theta_1}} \cdots \xrightarrow{L_{n-2}^{\theta_{n-2}}} V_{n-1} \xrightarrow{L_{n-1}^{\theta_{n-1}}} O_N \tag{2.2}$$

such that

$$N_\theta(x) = [\bigcirc_{i=0\ldots n} L_i^{\theta_i}](x) = [L_n^{\theta_n} \circ L_{n-1}^{\theta_{n-1}} \circ \cdots \circ L_0^{\theta_0}](x). \tag{2.3}$$

We will denote the intermediary space V_i as the *feature space at level* i and we will call the map L_i as the *layer at level* i. The number of *channels* at level i of a network is the dimensionality of the feature space V_i.

Remark 2.1.2 This terminology differs from the terminology normally used in machine learning. The normal nomenclature would be to call the intermediate spaces V_i layers. The choice to depart from the standard terminology is done to facilitate the discussion of the maps which this thesis is mostly focussed on.

The goal of the network is to learn dependencies in the data such that it can perform good predictions and this is usually done by training the network. There are several different ways to train networks, but for classification tasks the most used is supervised learning in which each data point x in the data set I must have a label y containing the *ground truth* of what x is. The set of all ground truths will be denoted T. Then one puts the data through the network and compares the output of the network with the ground truth. The comparison is done by a loss function:

Definition 2.1.3 (Loss function) A loss function (alt. cost function) for a network N is a smooth map $\mathcal{C} : O_N \times T \to \mathbb{R}$ which acts on the pairs $(N_\theta(x_i), y_i)$ where y_i is the ground truth corresponding to the data point x_i.

Example 2.1.4 Two examples of cost functions:

i) The method of least squares regression uses a cost function which just takes the square of the difference of the output to the ground truth

$$R^2 = \mathcal{C}(f_\theta(I), T) = \sum_{i \in 1,\dots,|I|} (f_\theta(x_i) - y_i)^2. \tag{2.4}$$

ii) A common loss function for classifying tasks is *categorical cross entropy* defined as

$$\mathcal{C}(N_\theta(x_i), y_i) = \sum_{j=1,\dots,C} (y_i)_j \log(N_\theta(x_i)_j) \tag{2.5}$$

where C denotes the number of classes.

Intuitively one can see $\mathcal{C}(N(\cdot, x), y)$ as a map from the parameter space Θ to $\mathbb{R}$ by

$$\theta \mapsto [\mathcal{C}(N(\cdot, x), y)](\theta) = \mathcal{C}(N(\theta, x), y) \in \mathbb{R}. \tag{2.6}$$

This allows for the interpretation that to minimise the loss of the network for pairs of input data and ground truth, one needs to find the minima of the surface created by the map $\mathcal{C}(N(\cdot, x), y) : \Theta \to \mathbb{R}$. As such this can be done by gradient descent, but finding the minima $\hat{\theta}$ for a single pair (x, y) does not imply that the same $\hat{\theta}$ minimises the loss also for (x', y'), and indeed this is not the case. If one would find a parameter set $\hat{\theta}$ that minimises the loss for a very specific set of data points — the extreme would be fitting to just one data point (x, y) — this parameter set would probably perform very poorly outside of this data set; this is called *overfitting*. The goal then becomes to find the parameter set $\hat{\theta}$ that minimises the loss for the training data but also generalises to data the network has not yet seen. This is called the *bias-variance tradeoff* and is an important subject, unfortunately this thesis will not focus more on training or the bias-variance tradeoff but the interested reader is encouraged to read [16] which gives a good overview of the topic.

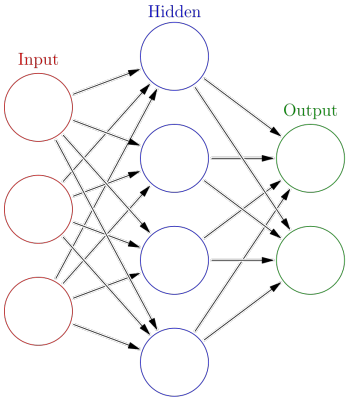


Figure 2.1: Schematic sketch of a simple neural network. The rings are called nodes. We will call each vertical stack of nodes a feature vector since they are an element of a feature space. The collection of arrows between feature vectors are called layers; this figure thus depicts two layers. Image source: [28]; used under licence CC BY-SA 3.0.

2.2 Fully connected networks

A common type of neural network is the fully connected neural network, also called feed forward neural network, and although this thesis will not focus on these we will still present rudimentary definitions for the sake of orientation. A schematic figure of a fully connected network can be found in Figure 2.1.

Definition 2.2.1 (Fully connected network) A *fully connected network* N_θ is a sequence of vector spaces V_i with $\dim V_i = n_i$, i.e n_i channels, and maps $L_i^{\theta_i} : V_i \to V_{i+1}$

$$I = V_0 \xrightarrow{L_0^{\theta_0}} V_1 \xrightarrow{L_1^{\theta_1}} \cdots \xrightarrow{L_{n-1}^{\theta_{n-1}}} V_n = O_N. \tag{2.7}$$

A vector in V_0 is an input feature vector, and vectors in V_n are output feature vectors.

Definition 2.2.2 (Fully connected layer) A layer $L_i^{\theta_i} : V_i \to V_{i+1}$ in an fully connected network acts on a vector $x \in V_i$ by

$$L_i^{\theta_i} = W_i x + b_i \tag{2.8}$$

where W_i is a $n_{i+1} \times n_i$ matrix, the elements of which are called *weights* of the layer and b_i is a vector in V_{i+1} called the *bias*. The weights W_i and biases b_i for a layer together make up the parameters for the layer θ_i.

2.3 Convolutional neural networks

We will talk about convolutional neural networks (CNNs) in terms of their application to image analysis. Since the data in the case of images is inherently two-dimensional we would like to get a feature vector, which contains numerical values for each feature we're looking for, at each point in the image. This concept is

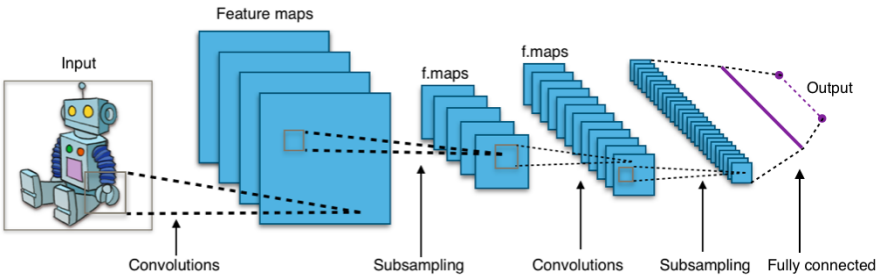


Figure 2.2: A schematic image of a typical CNN. Where an input image is convolved over to create the feature map. Note that this image use a different terminology: what in this image is called "feature maps" is what we call a single feature map. Image source: [29]; used under licence CC BY-SA 4.0.

encapsulated in feature maps. A schematic description of a CNN can be seen in Figure 2.2.

Definition 2.3.1 (Feature map) A *feature map* is a compactly supported map $f : \mathbb{R}^2 \to V$ where V is a feature vector space. We will sometimes refer to the feature maps in terms of their target space. We denote the set of feature maps between $\mathbb{R}^2$ and the feature vector space V as $\Gamma(\mathbb{R}^2, V) = \mathcal{C}_c^\infty(\mathbb{R}^2, V)$.

Definition 2.3.2 (Image) An *image* is a compactly supported map $I : \mathbb{R}^2 \to \mathbb{R}^n$ where $n = 1$ for greyscale data and $n = 3$ for RGB data. That is, an image is a feature map where V is either $\mathbb{R}$ or $\mathbb{R}^3$.

Remark 2.3.3 Note that in applications the numerical values for each component is restricted to some range; e.g. 8-bit precision where each component takes an integer value between 0 and 255.

Since we will use convolutions to map one feature map V_i into the next V_{i+1} we need a kernel.

Definition 2.3.4 (Kernel) A *kernel* is a compactly supported map $k : \mathbb{R}^2 \to \mathbb{R}^{m \times n}$.

Since we will perform convolutions on a discretised grid (the pixels of the image) we will "discretise" the image and kernel so that they are defined on $\mathbb{Z}^2$. This discretisation is only done when one thinks about the images and kernels as continuous functions, in application the image is never continuous (due to discrete light measurements) and therefore would not need to be "discretised". When one talks about *kernel size* one refers to the support of the kernel, which in applications is usually only non-zero on a subgrid of $\mathbb{Z}^2$ between 1×1 and 7×7.

Letting $\Gamma(A, B)$ denote the set of smooth maps between A and B, we can now define the convolutional layer

Definition 2.3.5 (Convolutional layer) A *continuous* convolutional layer L_i is a map between feature map spaces $\Gamma(\mathbb{R}^2, V_i)$ and $\Gamma(\mathbb{R}^2, V_{i+1})$ and maps a feature

map f_i to f_{i+1} pointwise as

$$f_{i+1}(x) = L_i(f_i)(x) = [k_i \star f_i](x) = \int_{y\in\mathbb{R}^2} k_i(y-x)f_i(y)dy \tag{2.9}$$

where the kernel maps

$$k_i : \mathbb{R}^2 \to \mathbb{R}^{n_{i+1}\times n_i}. \tag{2.10}$$

The discretised version of the convolution uses feature maps and kernels defined on $\mathbb{Z}^2$ and takes pointwise the form

$$f_{i+1}(x) = L_i(f_i)(x) = [k_i \star f_i](x) = \sum_{y\in\mathbb{Z}^2} k_i(y-x)f_i(y) \tag{2.11}$$

where the kernel maps

$$k_i : \mathbb{Z}^2 \to \mathbb{R}^{n_{i+1}\times n_i}. \tag{2.12}$$

Remark 2.3.6 The convolution defined here is actually a correlation and the convolution is really

$$[k_i \star f_i](x) = \int_{y\in\mathbb{R}^2} k_i(x-y)f_i(y)dy. \tag{2.13}$$

In machine learning the terminology is used interchangeably.

Remark 2.3.7 The Definition 2.3.5 is the mathematical definition for the continuous and discrete case. For convolutional layers in neural networks there are more parameters one has to give a convolutional layer when defining it in code: *stride* and *kernel size*. Above we mentioned that the kernel size is the support of the kernel, and stride refers to the sum in the convolution going over a subgrid to $\mathbb{Z}^2$. A (m,n)-strided convolution would be

$$[k_i \star f_i](x) = \sum_{y\in\{(a,b)\ \mid\ a=mz,\ b=nz' \text{ for } z,z'\in\mathbb{Z}^2\}} k_i(y-x)f_i(y). \tag{2.14}$$

2.3.1 Translation equivariance of convolutional layers

We now go on to discussing equivariance of the "ordinary" convolutional layer, firstly from a more abstract point of view. If the reader would like some rudimentary definitions in group theory, such can be found in Appendix A. A simple definition of equivariance is as follows.

Definition 2.3.8 (Equivariance) Let G be a group acting on a vector space V through a representation $\rho : G \to GL(V)$ and on V' through $\eta : G \to GL(V')$. Then a map $\psi : V \to V'$ is called G-equivariant if

$$\psi(\rho(g)v) = \eta(g)\psi(v). \tag{2.15}$$

This is diagrammatically represented as this diagram commuting:

$$\begin{array}{ccc} V & \xrightarrow{\psi} & V' \\ \downarrow{\scriptstyle \rho(G)} & & \downarrow{\scriptstyle \eta(G)} \\ V & \xrightarrow{\psi} & V' \end{array}$$

In the case of ordinary convolutions, the group is the translation group acting on $\Gamma(\mathbb{Z}^2, V_i)$. It acts on the feature maps as left translation $l_y[f_i](x) = f_i(x - y)$.

Lemma 2.3.9 (Translation equivariance) *A convolutional layer*

$$L_i : \Gamma(\mathbb{Z}^2, V_i) \to \Gamma(\mathbb{Z}^2, V_{i+1}) \tag{2.16}$$

is translation equivariant. □

Proof We first apply the convolution to the translated feature map and use the definitions

$$\begin{aligned} L_i(l_y f_i)(x) = [k_i \star l_y f_i](x) &= \sum_{z \in \mathbb{Z}^2} k_i(z - x)(l_y f_i)(z) \\ &= \sum_{z \in \mathbb{Z}^2} k_i(z - x) f_i(z - y). \end{aligned} \tag{2.17}$$

Next we shift our summation variable $z' = z - y$

$$\sum_{z \in \mathbb{Z}^2} k_i(z-x) f_i(z-y) = \sum_{z' \in \mathbb{Z}^2} k_i(z'+y-x) f_i(z') = \sum_{z' \in \mathbb{Z}^2} k_i(z'-(x-y)) f_i(z') \tag{2.18}$$

and finally proceed with using the definition of the convolution to arrive at the wanted result

$$\sum_{z' \in \mathbb{Z}^2} k_i(z' - (x - y)) f_i(z') = L_i(f_i)(x - y) = l_y[L_i(f_i)](x). \tag{2.19}$$

Hence

$$L_i(l_y f_i) = l_y[L_i f_i]$$

and translation equivariance is proven. ■

For convolutional layers this means that if the input image is translated by some vector then the output image would be the same as if the convolution was done first and then translation was applied.

2.3.2 Why convolutional neural networks?

Convolutional networks are really efficient when it comes to image analysis, and the main reasons for this is *weight sharing* and *locality*.

Weight sharing

Say that we would like to analyse an $n \times n$ greyscale image with a fully connected network and have an m-dimensional feature vector after the first layer. The input feature vector would have n^2 dimensions (n^2 nodes) and since every node in one feature vector is connected to every node in the next feature vector, this becomes $n^2 m$ parameters in the first layer.

Compare this to a convolutional neural network for the same image where in the layer we define a kernel $k_0 : \mathbb{Z}^2 \to \mathbb{R}^{m \times 1}$ which assigns a $m \times 1$ matrix to each point

in $\mathbb{Z}^2$; the 1 comes from the fact that the image is greyscale and thus has a one-dimensional feature vector at each point. If the kernel is of size 3×3, i.e. is non-zero on a 3 by 3 grid, then we have $3 \cdot 3 \cdot m$ parameters, and since this kernel is slid over the image these parameters are used for the whole image, and thus *shared*. This amounts to an immense reduction in parameters which is a big gain in efficiency.

Locality

If we were to analyse an image using a fully connected network then we would have to flatten the image; that is, convert from a 2D image to a 1D array. In doing so, some pixels that were spatially adjacent in the original image would end up not spatially adjacent in the flattened version. See Figure 2.3 where a subimage of a MNIST [13] digit is flattened.

Spatial locality is an important feature of reality, and we often recognise things by how they are put together, which can't be done if the image is flattened.

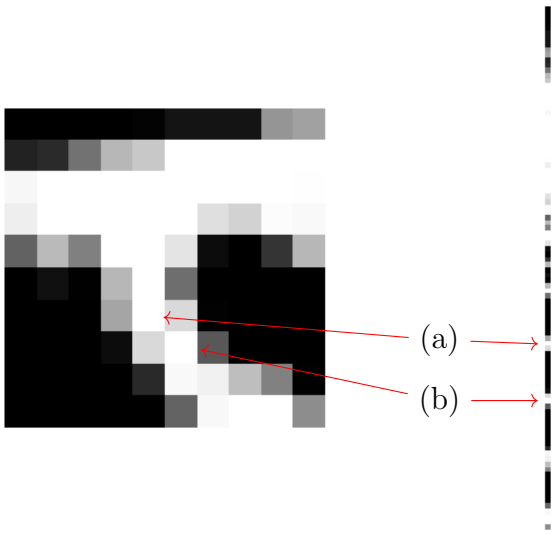


Figure 2.3: The left image is a subimage of an image from the MNIST dataset and is flattened to obtain the array on the right, and in doing this loses locality. For example, the white pixels at (a), and (b) were adjacent in the original image but not in the flattened version.

2.3.3 Equivariance of convolutions under rotations

Convolutional layers are equivariant under translation, which is a symmetry of the plane, but rotations are also in the symmetry group of the plane. Can we make normal convolution equivariant under rotations? The motivation for this was discussed in Section 1.

Let us, for the sake of using continuous rotations, go back to the continuous case where feature maps are functions from $\mathbb{R}^2$ to some V_i and the kernels are defined on $\mathbb{R}^2$. The more general case of an arbitrary homogeneous base space is discussed in [12]. To make clear what we are working with we refine the definition of feature map:

Definition 2.3.10 (Feature map of type ρ) A feature map of type ρ is a feature map which has the following transformation property under the action of an element g of some group G

$$f_i(x) \xrightarrow{g} l_g[f_i](x) = \rho_i(g) f_i(g^{-1}x). \tag{2.20}$$

Remark 2.3.11 The type of data determines which transformation is appropriate to choose for ρ. For example, if f would be a RGB image, then if we rotate the image we would not want the channels to mix and as a consequence we need to choose ρ as the direct sum of three one-dimensional trivial representations. That is, f consists of three separate scalar fields $f_{i=1,2,3}$ which each transform as

$$f_i(x) \xrightarrow{g} l_g[f_i](x) = f_i(g^{-1}x). \tag{2.21}$$

If the data was a vector field, e.g. wind data, then the components of the vector $f_i(x)$ would transform under g and the standard choice of ρ for G being the group of rotations in the plane, $SO(2)$, would be the canonical representation of $SO(2)$ acting on $f_i(x)$. If f_i would be a two-dimensional vector field this representation could be

$$R(\theta) = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}. \tag{2.22}$$

Remark 2.3.12 Feature maps transforming in this way, as well as compatible kernels, have been studied by Cohen et al. under the name of *steerable CNNs* [17].

Suppose we have representations ρ_i, and ρ_{i+1} so that the group of rotations acts through these on the feature vector spaces V_i and V_{i+1}. Equivariance with respect to rotations would then be

$$l_g[L_i(f)](x) = \rho_{i+1}(g) L_i(f_i)(g^{-1}x) = L_i(\rho_i(g) f_i)(g^{-1}x) = L_i(l_g[f_i])(x) \tag{2.23}$$

for some element g of the rotation group.

Lemma 2.3.13 (Equivariance of convolutions w.r.t. rotations) *The convolution*

$$f_{i+1}(x) = L_i(f_i)(x) = [k_i \star f_i](x) = \int_{y \in \mathbb{R}^2} k_i(y - x) f_i(y) dy \tag{2.24}$$

between a feature map of type ρ_i and one of type ρ_{i+1} is equivariant iff the kernel satisfies

$$k_i(z) = \rho_{i+1}(g) k_i(g^{-1}z) \rho_i(g^{-1}). \tag{2.25}$$

□

PROOF We start by transforming the input feature map and using the definition of the convolution

$$L_i(l_g[f_i])(x) = [k_i \star l_g[f_i]](x) = \int_{\mathbb{R}^2} k_i(z-x) l_g[f_i](z) dz \tag{2.26}$$

followed up by the definition of the group action on the feature map f

$$\int_{\mathbb{R}^2} k_i(z-x) l_g[f_i](z) dz = \int_{\mathbb{R}^2} k_i(z-x) \rho_i(g) f_i(g^{-1}z) dz. \tag{2.27}$$

Performing a change of coordinates $z' = g^{-1}z$ and using that g is a rotation of the plane and hence has determinant 1 we get

$$\int_{\mathbb{R}^2} k_i(z-x) \rho_i(g) f_i(g^{-1}z) dz = \int_{\mathbb{R}^2} k_i(gz-x) \rho_i(g) f_i(z) dz. \tag{2.28}$$

From the other side we first apply the definition of the group action

$$l_g[L_i(f_i)](x) = \rho_{i+1}(g) L_i(f_i)(g^{-1}x) = \rho_{i+1}(g)[k_i \star f_i](g^{-1}x) \tag{2.29}$$

followed with the definition of the convolution

$$\rho_{i+1}(g)[k_i \star f_i](g^{-1}x) = \rho_{i+1}(g) \int_{\mathbb{R}^2} k_i(z - g^{-1}x) f_i(z) dz. \tag{2.30}$$

We use that g is independent of the integration to move this inside the integral and thus obtaining

$$\rho_{i+1}(g) \int_{\mathbb{R}^2} k_i(z - g^{-1}x) f_i(z) dz = \int_{\mathbb{R}^2} \rho_{i+1}(g) k_i(z - g^{-1}x) f_i(z) dz. \tag{2.31}$$

The equality chain ending in (2.28) together with the chain ending in (2.31) we see that requiring equivariance gives the following criteria on the kernel

$$k_i(gz - x)\rho_i(g) = \rho_{i+1}(g) k_i(z - g^{-1}x). \tag{2.32}$$

If the kernel k_i satisfies this criteria then we can follow the steps above and see that equivariance is indeed fulfilled

$$\begin{aligned} L_i(l_g[f_i])(x) &= [k_i \star l_g[f_i]](x) = \\ &= \int_{\mathbb{R}^2} k_i(gz - x)\rho_i(g) f_i(z) dz \\ &= \int_{\mathbb{R}^2} \rho_{i+1}(g) k_i(z - g^{-1}x) f_i(z) dz \\ &= \rho_{i+1}(g) L_i(f_i)(g^{-1}x) \\ &= l_g[L_i(f_i)](x). \end{aligned} \tag{2.33}$$

Hence, requiring the convolution to be rotation equivariant and $z - g^{-1}x = g^{-1}(gz - x)$ together with (2.32) puts the following constraint on the kernel $k_i : \mathbb{R}^2 \to \mathbb{R}^{n_{i+1} \times n_i}$:

$$k_i(z) = \rho_{i+1}(g) k_i(g^{-1}z) \rho_i(g^{-1}), \tag{2.34}$$

and that property in turn gives equivariance. Which was what we wanted to show.■

Remark 2.3.14 In the case that the feature maps we transform are scalars, i.e. $\rho = 1$, the criteria on the kernel would be

$$k_i(z) = k_i(g^{-1}z) \tag{2.35}$$

and thus the kernel would have to be invariant under rotations.

3

Gauge equivariant networks

In this chapter we build up the theory needed to describe the gauge equivariant convolutions. We will start from the basic convolutional layer on flat data, equivariant only to translation and work up from there. We will assume a passing familiarity with topology, manifolds and fibre bundles; more specifically principal bundles, associated bundles, and frame bundles. If the reader feels unfamiliar with these subjects the relevant parts of the supporting theory can be found in Appendix A.

3.1 Geometric deep learning

Thus far in this thesis we have only discussed data on flat manifolds — e.g. a normal image, or a vector field on a flat space — but there are data we would like to analyse that do not live on an inherently flat space. An example would be the spherical images obtained from fisheye cameras used in autonomous cars.

So, what happens when one goes from a flat space to something curved? The short story it is that locally one can do the same things as in a flat space, but globally — on the entire space — one cannot. For example, it is impossible to create a global basis on the sphere, a fact sometimes known as the hairy ball theorem. Instead of viewing data as a function from the base space to some data space that is the same for every point in the base space one needs to view data as sections of fibre bundles.

Definition 3.1.1 (Bundle and fibre bundle) A bundle over a manifold M is a triple (E, π, M) consisting of a total space E, a projection $E \xrightarrow{\pi} M$, and the base space M. The so called fibre over a point p, denoted E_p, is the set of all elements in E that are projected to p by π:

$$E_p = \pi^{-1}(p) = \{e \in E \mid \pi(e) = p\}. \tag{3.1}$$

The bundle is a fibre bundle if all fibres are isomorphic to some "typical" fibre F, i.e.

$$E_p \simeq F \text{ for all } p \in M. \tag{3.2}$$

In the case where F is a vector space the construction is called a vector bundle. A fibre bundle with typical fibre F is denoted (E, π, M, F).

More definitions relating to fibre bundles can be found in appendix A.4.

Remark 3.1.2 Note that for a fibre bundle (E, π, M, F), even though $E_p \simeq F$ for every point in M, each E_p is still a different space, and a priori we cannot compare elements in fibres over different points.

The data, that previously was a function $f : M \to V$ for some feature space V, will now be a section of a fibre bundle.

Definition 3.1.3 (Section) A section σ of a bundle (E, π, M) is a map $\sigma : M \to E$ such that

$$\pi \circ \sigma = \mathrm{id}_M \,. \tag{3.3}$$

Remark 3.1.4 A section of a fibre bundle thus assigns to each point $p \in M$ an element in the fibre above p.

Example 3.1.5 Let f be a feature map on a flat space $f : M = \mathbb{R}^2 \to \mathbb{R}^k$ which assigns a k-dimensional feature vector to each point in M, then we could view this as a section of the vector bundle $(E = \mathbb{R}^k, \pi, \mathbb{R}^2, \mathbb{R}^k)$ where the total space consists of elements $e \in \mathbb{R}^k$ for each point in M.

Example 3.1.6 An important example of a vector bundle is the tangent bundle TM to M. At each point the vector space is the tangent space T_pM to M at p, and if M is a d-dimensional smooth manifold then $T_pM \simeq \mathbb{R}^d$.

On a flat space the concepts of function and section coincide since it is possible to join the fibres together in a canonical way, and it is therefore possible to move objects between the fibres without extra structure; the space is flat. For a general smooth curved manifold moving an object in a fibre around is no longer doable since the outcome will depend on the path chosen; this is the reason for why one needs to use fibre bundles and sections instead of functions. One can add additional

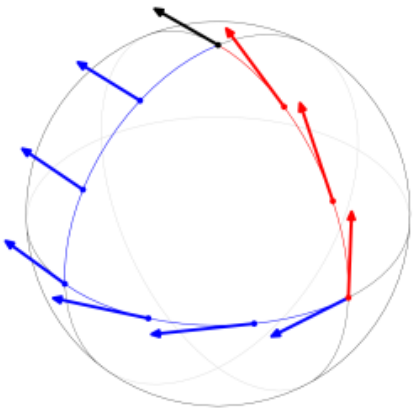


Figure 3.1: A tangent vector the sphere's north pole is moved along two different paths, and the result differs since the sphere is curved. Image source: [30], image in the public domain.

structure, a connection, that allows for transporting objects between fibres locally in a unique way by following specific set of paths called geodesics and this in turn allows for local comparison of objects in fibres.

All this means that everything that previously was done globally, e.g. convolution and coordinate changes (e.g. rotation), must now be done pointwise, and there need not be anything that links what is done in one point to the neighbouring points.

For more comments on networks on manifolds the reader is recommended to read [10].

3.2 Introductory discussion of gauge equivariance

To get a geometrical intuition of what we mean when we talk about gauge equivariance and changing gauge we will start by considering the following:

Let M be a d-dimensional smooth manifold and σ be a section of the tangent bundle to M; hence $\sigma(p)$ is a tangent vector to M at $p \in M$. A (local) change of gauge at p then amounts to changing the basis in the tangent space T_pM at p which can be realised as the action of $GL(d, R)$ on the tangent space. This does not change the tangent vector $\sigma(p)$ although the components of $\sigma(p)$ looks different when expressed in the bases corresponding to the different gauges. See Figure 3.2 and 1.3 for the combined intuition. Note that the change of coordinates for T_pM is the same for all of T_pM but might vary between different points $p, p' \in M$.

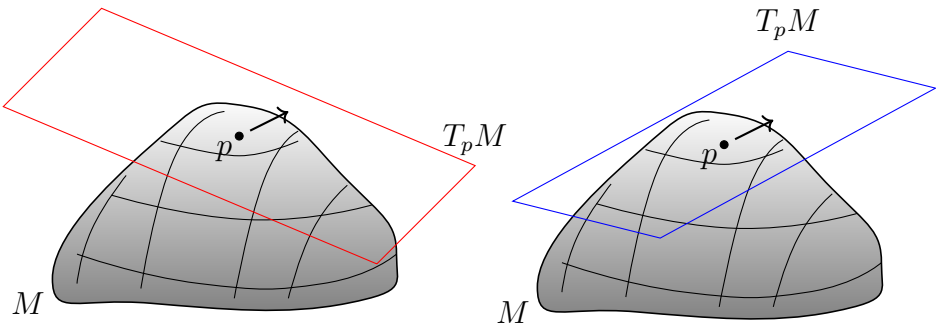


Figure 3.2: The manifold M has a different choice of gauge — local coordinates on M — at p in the left and the right figure leading to a different choice of basis in the corresponding tangent space T_pM.

Hence, for a convolution to be gauge equivariant we need that the action of the layer commutes with change of gauge, i.e. change of coordinates in the tangent space. To proceed we introduce the associated bundle.

3.3 The associated bundle

The associated bundle will contain the information of the structure group of M and will keep the invariance of the vectors by setting up equivalence classes.

Definition 3.3.1 (Associated bundle) Let

i. M be a d-dimensional smooth manifold,
ii. V be a vector space,
iii. LM be the (principal) frame bundle to M.

Since LM is a principal $G = GL(d, \mathbb{R})$-bundle we must define the action of G on V. We define this in the following way:

$$v \mapsto g \triangleright v = \rho(g)v. \tag{3.4}$$

Next we define an equivalence relation on $LM \times V$ as

$$[e, v] \sim_\rho [e', v'] \quad \text{iff} \quad \exists g \in G : \begin{cases} e' = e \triangleleft g \\ v' = g^{-1} \triangleright v = \rho(g^{-1})v. \end{cases} \tag{3.5}$$

The total space of the associated bundle will then be

$$LM_V = (LM \times V)/\sim_\rho \tag{3.6}$$

with the projection $\pi_V : LM_V \to M$ defined by

$$\pi_V([e, v]) = \pi(e) \tag{3.7}$$

where $\pi : LM \to M$ is the projection of the frame bundle.

The first thing to check is if the projection π_V is well defined, and it is:

$$\pi_V([e, v]) = \pi_V([e \triangleleft g, g^{-1} \triangleright v]) = \pi(e \triangleleft g) = \pi(e) \tag{3.8}$$

since the action of G doesn't change the fibres of the frame bundle: $e \triangleleft g \in L_pM$ iff $e \in L_pM$.

Lemma 3.3.2 *The above constructed associated bundle is a fibre bundle with typical fibre V.* □

Proof To prove that the associated bundle is a fibre bundle we must show that $\pi_F^{-1}(p) = V_p \simeq V$ for each $p \in M$.

First, define a function $\phi : LM_V \to LM$ which acts by

$$\phi([e, v]) = \{e' \mid e' = e \triangleleft g,\ g \in G\} = L_pM \subset LM \tag{3.9}$$

since G acts transitively on L_pM. This construction is necessary for ϕ to be well defined on equivalence classes of LM_V.

With this we get $\pi_V = \pi \circ \phi$ since $\pi_V([e, v]) = \pi(e)$ and $(\pi \circ \phi)([e, v])) = \pi(L_pM) = \pi(e)$ for $e \in L_pM$. Hence we have $\pi_V^{-1} = \phi^{-1} \circ \pi^{-1}$. By the definition of π we get that $\pi^{-1}(p) = L_pM$ and

$$\begin{aligned} \phi^{-1}(L_pM) &= \{[e, v] \mid \phi([e, v]) \in L_pM\} = \{[e, v] \mid e \in L_pM,\ v \in V\} \\ &= \{[e_0, v'] \mid v' \in V\} \simeq V \end{aligned} \tag{3.10}$$

where we used the transitive action of G on L_pM to choose a class representative that has a fixed frame e_0 as first component.

This shows that $\pi_V^{-1}(p) \simeq V$ for all $p \in M$ and hence is LM_V a fibre bundle with typical fibre V. ■

Remark 3.3.3 Letting V be the space of tangent vectors we retain the geometrical intuition presented in Figure 3.2.

A feature map at level i in a GCNN will be a section of an associated bundle with a specific representation ρ_i acting on a vector space V_i in which the feature vectors will live. The representation ρ_i will change depending on what type of data one uses; for example, if the input data to a GCNN, level $i = 0$, is an image then we would not want the intensity or the RGB channels to mix if we rotate the tangent plane, hence we would choose $\rho_0 = 1$ for such data, but if the input was something like a vector field, say wind directions on the Earth, a different ρ_0 would be chosen.

3.4 The invariant/equivariant layer

At the end of the last section we mentioned that the feature maps will be sections of a suitable associated bundle, depending on the level the feature map is in and what type of data it represents. This allows for the viewpoint that the layers are maps between associated bundles, which has some interesting consequences.

Proposition 3.4.1 *Let LM_{V_i} and $LM_{V_{i+1}}$ be two associated bundles under the same group G and let $\phi_i : LM_{V_i} \to LM_{V_{i+1}}$ be a map between those bundles with the action*

$$\phi_i([e, v]) = [e, L_i(v)]$$

where L_i is a map between V_i and V_{i+1} that is transforming the feature vector v.

Then, ϕ_i is well defined on the equivalence classes of the associated bundle LM_{V_i} iff L_i is equivariant under the action of G

$$L_i(g^{-1} \triangleright v) = g^{-1} \triangleright L_i(v) \tag{3.11}$$

□

Proof Requiring that ϕ_i be well defined on equivalence classes gives the following

$$\phi_i([e, v]) = \{ \text{ well defined } \} = \phi_i([e, v] \triangleleft g) = \phi_i([e \triangleleft g, g^{-1} \triangleright v]). \tag{3.12}$$

Employing the definition of the action of ϕ_i we get

$$\phi_i([e \triangleleft g, g^{-1} \triangleright v]) = [e \triangleleft g, L_i(g^{-1} \triangleright v)]. \tag{3.13}$$

Using the fact that $g^{-1} \triangleright g \triangleright u = u$ for $u \in V_i$ or V_{i+1}

$$[e \triangleleft g, L_i(g^{-1} \triangleright v)] = [e \triangleleft g, g^{-1} \triangleright g \triangleright L_i(g^{-1} \triangleright v)] \tag{3.14}$$

which, when using the definition of the action of G on the equivalence classes of the associated bundle, becomes

$$[e \triangleleft g, g^{-1} \triangleright g \triangleright L_i(g^{-1} \triangleright v)] = [e, g \triangleright L_i(g^{-1} \triangleright v)] \triangleleft g. \tag{3.15}$$

Since the equivalence classes are preserved under the action of G we arrive at

$$[e, g \triangleright L_i(g^{-1} \triangleright v)] \triangleleft g = [e, g \triangleright L_i(g^{-1} \triangleright v)]. \tag{3.16}$$

But since we require ϕ_i to be well defined on equivalence classes we also have

$$\phi_i([e, v]) = [e, L_i(v)]. \tag{3.17}$$

Putting (3.16) and (3.17) together one obtains

$$\phi_i([e, v]) = [e, L_i(v)] = [e, g \triangleright L_i(g^{-1} \triangleright v)] = \phi_i([e, v] \triangleleft g) \tag{3.18}$$

and hence

$$L_i(v) = g \triangleright L_i(g^{-1} \triangleright v) \quad \iff \quad g^{-1} \triangleright L_i(v) = L_i(g^{-1} \triangleright v). \tag{3.19}$$

That is, requiring ϕ_i to be well defined on equivalence classes of the associated bundle requires the map L_i, that transforms the feature vector, to be equivariant:

$$L_i(g^{-1} \triangleright v) = g^{-1} \triangleright L_i(v). \tag{3.20}$$

Checking that L_i is equivariant under G implies that ϕ_i is well defined is a short calculation, presented here with no further comment since every relevant step was mentioned above.

$$\begin{aligned} \phi_i([e, v]) &= [e, L_i(v)] \\ &= [e, L_i(v)] \triangleleft g \\ &= [e \triangleleft g, g^{-1} \triangleright L_i(v)] = \{\text{equivariance assumption}\} \\ &= [e \triangleleft g, L_i(g^{-1} \triangleright v)] \\ &= \phi_i([e \triangleleft g, g^{-1} \triangleright v]) \\ &= \phi_i([e, v] \triangleleft g). \end{aligned} \tag{3.21}$$

Therefore, the action of ϕ_i is well defined iff L_i is equivariant under the action of G; which was what we wanted to show. ∎

Remark 3.4.2 Recalling that $L_i(v) \in V_{i+1}$ for $v \in V_i$ and that different V_k:s transform under different ρ:s the equivariance becomes

$$L_i(\rho_i(g^{-1})v) = \rho_{i+1}(g^{-1})L_i(v) \tag{3.22}$$

which is exactly on the form presented in equation (2.15).

3.5 Gauge equivariant convolution

Now we got an idea of what the feature maps are and what gauge equivariance is, so now we discuss the aspects of gauge equivariant convolution.

One important reason for the adoption of convolutions for normal CNNs was weight sharing: the fact that moving one kernel over the image massively reduces the number of parameters one has to use. We would like to build something similar into the gauge equivariant convolution, but since the action of changing gauge around a point $p \in M$ is the same as changing basis in the tangent space T_pM we need to let

everything in the convolution be based on the point p or the tangent space T_pM in order to obtain the feature vector at p.

In the case when M is flat, e.g. $M = \mathbb{R}^d$, we have "normal convolution"

$$[k \star f](x) = \int_{\mathbb{R}^d} k(x, y) f(y) dy. \tag{3.23}$$

where one can see sections of a vector bundle f over M as functions $f : M \to V$ instead of mapping into the separate fibres V_p. An intuition for the kernel $k : M \times M \to \hom(V, V')$ is that $k(x, y)$ represents the kernel centred at x and evaluated in y so that we get the correct value of k to transform f at y.

Now we are transitioning to considering a general curved manifold, and before we can present a definition of a gauge equivariant convolution we need a couple of definitions.

Definition 3.5.1 (Feature map on manifold) A *feature map f_i on the manifold* M at level i of a network is a section of the vector bundle $E_i \xrightarrow{\pi} M$ with typical fibre V_i on which the action of $GL(d)$ is defined through a representation ρ_i:

$$f_i(p) \mapsto h \triangleright f_i(p) = \rho_i(h) f_i(p). \tag{3.24}$$

Sometimes $f_i(p)$ is also denoted $(f_i)_p$ to signify that this is a vector at the base point $p \in M$.

Definition 3.5.2 (Metric) A *metric* at a point p of a d-dimensional manifold M, denoted g_p, is a linear and symmetric map

$$g_p : T_pM \times T_pM \to \mathbb{R}. \tag{3.25}$$

The metric can be expressed in components

$$(g_p)_{ij} = g_p(e_i, e_j) \tag{3.26}$$

where $\{e_i\}_{i=1,\dots,d}$ is a basis for T_pM induced by a local chart. This way the metric have a matrix representation and the determinant of the metric is denoted

$$\det(g_p) = |g_p|. \tag{3.27}$$

Remark 3.5.3 In appendix A.5.4 there is a more explicit definition of metric.

Definition 3.5.4 (Kernel on manifold) The *kernel* between level i and $i + 1$ in a network on a manifold M is a map

$$k_i : M \times TM \to \hom(V_i, V_{i+1}) \tag{3.28}$$

where TM denotes the tangent bundle to M. And specifically at a point $p \in M$ we have

$$k_i(p) : T_pM \to \hom((V_i)_p, (V_{i+1})_p) \tag{3.29}$$

where $(V_i)_p$ is the fibre over p of the vector bundle E_i. Since E_i is a vector bundle $(V_i)_p \simeq V_i$.

Lastly we need a way to move vectors $f_i(p)$ between fibres. This can be done via parallel transport and the exponential map:

Definition 3.5.5 (Exponential map) Let (U_p, ψ) be a chart on the manifold M centred around p. Then the *exponential map* $\exp_p : T_pM \to M$ takes a tangent vector $v \in T_pM$ and parallel transports the point p along the unique geodesic $\gamma : I \to M$, such that $\gamma(0) = x$ and $(\psi \circ \gamma)'(0) = v$, to the point $\gamma(1)$.

Remark 3.5.6 This exponential map is used to transport vectors in the following way. Let $f(\exp_p v)$ be a vector at the point obtained by moving p along the curve γ with tangent vector $\gamma'(0) = v$ in some chart, then we write

$$f\Big|_{\exp_p v}(p) \tag{3.30}$$

to denote that the feature map f is evaluated in the point $f(\exp_x v)$ and then parallel transported back to p along the appropriate geodesic γ. See section A.6 for more details on geodesics and parallel transport.

With these definitions we can now state the gauge equivariant convolution as proposed by Cheng at. al. [11]:

Definition 3.5.7 (Equivariant convolution on manifold) Let (U_p, ψ) be a chart on M centred around $p \in M$ such that it induces a basis e on T_pM. Then the gauge equivariant convolution of a feature map f_i on M is

$$L_i(f_i)(p) = [k_i \star f_i](p) = \int_{B_{p,R}} k_i(p, v) f_i\Big|_{\exp_p v}(p)\sqrt{|g_p|}dv, \tag{3.31}$$

where

$$B_{p,R} = \{v \in T_pM \mid g_p(v, v) < R\} \tag{3.32}$$

is a ball in the tangent space T_pM.

Remark 3.5.8 Note that the chart (U_p, ψ) is necessary since to compute the determinant of the metric one needs it components, which only exist in relation to an induced basis. Also the parallel transport of the feature map needs the chart for numerical calculation.

Proposition 3.5.9 *Given a chart (U_p, ψ) on M centred around $p \in M$ such that it induces a basis e_p on T_pM, then the map $\phi_i : LM_{V_i} \to LM_{V_{i+1}}$ acting as*

$$\phi_i([e_p, f_i(p)]) = [e_p, L_i(f_i)(p)] \tag{3.33}$$

where L_i is the gauge equivariant convolution, is well defined iff the kernel k_i satisfies

$$k_i(p, v) = \rho_{i+1}(h)k_i(p, v)\rho_i(h^{-1}) \tag{3.34}$$

for $h \in GL(d)$. □

Remark 3.5.10 By proposition 3.4.1 that ϕ_i is well defined is equivalent with L_i being equivariant under the action of $G = GL(d)$ which is the group that acts on elements of the associated bundles LM_{V_i} and $LM_{V_{i+1}}$.

Lemma 3.5.11 (Invariant measure) *The construction $\sqrt{|g_p|}dv$ is invariant under the action of $h \in GL(d)$.* □

Proof Choose a chart (U_p, ψ), then the metric can be expressed in components by evaluating it on the basis vectors of a frame $e \in LM$ at p as

$$(g_p)_{ij} = g_p(e_i, e_j). \tag{3.35}$$

The basis vectors of the frame e change as

$$e_i \xrightarrow{h} \rho(h^{-1})^m{}_i e_m \tag{3.36}$$

by the action of $h \in GL(d)$ on the principal frame bundle LM. Hence we get that the components of the metric changes as

$$\begin{aligned} g_p(e_i, e_j) \xrightarrow{h} g_p(\rho(h^{-1})^m{}_i e_m, \rho(h^{-1})^k{}_j e_k) &= \rho(h^{-1})^m{}_i \rho(h^{-1})^k{}_j g_p(e_m, e_k) \\ &= \rho(h^{-1})^m{}_i \rho(h^{-1})^k{}_j (g_p)_{mk}. \end{aligned} \tag{3.37}$$

Writing this in terms of matrices we get the following transformation

$$g_p \xrightarrow{h} \rho((h^{-1})^T) g_p \rho(h^{-1}). \tag{3.38}$$

Hence $\sqrt{|g_p|}$ transforms as

$$\sqrt{|g_p|} \xrightarrow{h} \det(\rho(h^{-1}))\sqrt{|g_p|}. \tag{3.39}$$

The action of $GL(d)$, a change of gauge, is a general change of coordinates in the tangent space and since a tangent vector transforms under h as

$$v \xrightarrow{h} \rho(h)v \tag{3.40}$$

we get that the integration measure changes

$$dv \xrightarrow{h} \det(\rho(h))dv. \tag{3.41}$$

Note that h is the same and constant for the whole tangent space.

Taking this together with the square root of the metric we see that they indeed form an invariant integration measure under action of $GL(d)$:

$$\sqrt{|g_x|}dv \xrightarrow{h} \det(\rho(h^{-1}))\sqrt{|g_x|}\det(\rho(h))dv = \sqrt{|g_x|}dv. \tag{3.42}$$

Which was what we wanted. ■

Proof (Proof of proposition 3.5.9) Let (U_p, ψ) be a chart centred at $p \in M$. Letting l_h denote the action of $h \in GL(d)$, and f_i the feature map at level i in the network, the equivariance of L_i would be

$$L_i(l_h[f_i])(p) = [k_i \star l_h[f_i]](p) = l_h[k_i \star f_i](p) = l_h[L_i(f_i)](p). \tag{3.43}$$

To obtain the wanted result we work from both directions. From the left hand side

$$\begin{aligned}[k_i \star l_h[f_i]](p) &= \int_{B_{p,R}} k_i(p,v) l_h \left[f_i\Big|_{\exp_p v} \right](p)\sqrt{|g_p|}dv \\ &= \int_{B_{p,R}} k_i(p,v)\rho^i(h) f_i\Big|_{\exp_p v}(p)\sqrt{|g_p|}dv,\end{aligned} \tag{3.44}$$

and from the right hand side

$$\begin{aligned}l_h[k_i \star f_i](p) &= l_h\left[\int_{B_{p,R}} k_i(p,v) f_i\Big|_{\exp_p v}(p)\sqrt{|g_p|}dv\right] \\ &= \rho_{i+1}(h)\int_{B_{p,R}} k_i(p,v) f_i\Big|_{\exp_p v}(p)\sqrt{|g_p|}dv.\end{aligned} \tag{3.45}$$

since we know that the output is a vector in the fibre $(V_{i+1})_p$ at the point p and hence transforms according to (3.24). Taking (3.44) and (3.45) together we get:

$$\rho_{i+1}(h)k_i(p,v) = k_i(p,v)\rho_i(h) \quad \iff \quad k_i(p,v) = \rho_{i+1}(h)k_i(p,v)\rho_i(h^{-1}). \tag{3.46}$$

Hence the convolution is equivariant iff k_i satisfies this criteria, and therefore the map ϕ_i is well defined by proposition 3.4.1. Which is what we wanted. ∎

Remark 3.5.12 Note that the transformation of h happens in the tangent space at p, at other points p' one might use a different group element h'.

3.6 Discussion on the gauge equivariant convolution

In this section we present some discussion on the gauge equivariant convolution and how these relate to the flat case discussed in Chapter 2, as well as the case of integration of differential forms on manifolds.

3.6.1 Comparison to the flat case

The gauge equivariant convolution presented in the previous section was on the form

$$L_i(f_i)(p) = [k_i \star f_i](p) = \int_{B_{p,R}} k_i(p,v) f_i\Big|_{\exp_p v}(p)\sqrt{|g_p|}dv, \tag{3.47}$$

with the criteria on the kernel

$$k_i(p,v) = \rho_{i+1}(h)k_i(p,v)\rho_i(h^{-1}). \tag{3.48}$$

Comparing this with the criteria obtained for rotations on the plane seen in equation (2.25):

$$k_i(z) = \rho_{i+1}(g)k_i(g^{-1}z)\rho_i(g^{-1}). \tag{3.49}$$

covered in section 2.3.3, there are obvious similarities.

In the general case of the gauge equivariant convolution, local transformations at a point p does not move the point, and since z in (3.49) refers to the point the feature map is evaluated at, the local transformation would be $z = g^{-1}z$. Using this, the criteria would be

$$k_i(z) = \rho_{i+1}(g)k_i(z)\rho_i(g^{-1}). \tag{3.50}$$

which is truly on the same form as the kernel criteria for the gauge equivariant convolution. As such it is reasonable to say that the gauge equivariant convolution is a pointwise equivariant convolution.

3.6.2 Comparison to differential forms

In the area of differential geometry one has differential forms $\omega \in \Omega^k(M)$ defined on a d-dimensional manifold M such that $\omega : M \to \mathrm{Alt}^k(TM)$ and

$$\omega(p) \in \mathrm{Alt}^k(T_pM) \tag{3.51}$$

where $\mathrm{Alt}^k(T_pM) = \{\eta \mid \eta : T_pM \times \cdots \times T_pM \to \mathbb{R}\}$ is the vector space of k-alternating maps on T_pM. This allows for a unique definition of the integral of a differential form on a manifold

$$\int_M : \Omega^d(M) \to \mathbb{R}. \tag{3.52}$$

This theory is very well known and it would simplify future development being able to formulate the gauge equivariant convolutions in the language of differential forms.

When performing an integral of a differential form $\omega \in \Omega^d(M)$ on a d-dimensional manifold M one has to use a chart (U, ϕ) to relate the abstract integral to something we know how to compute, i.e. integrals on $\mathbb{R}^d$. The integral then takes the form

$$\int_U \omega = \int_{\phi(U)} (\phi^{-1})^* \omega \tag{3.53}$$

which is an integral on $\mathbb{R}^d$ since $\phi(U) \subset \mathbb{R}^d$, by the virtue of being a chart, and since $\phi^{-1} : \mathbb{R}^d \to M$, $(\phi^{-1})^*$ is a pullback and we have $(\phi^{-1})^*\omega \in \Omega^d(\mathbb{R}^d)$. This can then be translated into an ordinary integral on $\mathbb{R}^d$ by using the basis induced by the chart. A change of local chart then amounts to the ordinary change of coordinates in integrals on $\mathbb{R}^d$. That this is well defined hinges on the fact that the alternating maps are scalar valued and hence it is possible to uniquely transport the values at different points and add them; this is essentially what the integral of the differential form does: it puts a locally flat coordinate system on the manifold and uses this to add values at different points.

For a vector bundle valued differential form this is no longer unique and if one specifies a way to deal with this there is also the problem that the result of the integration will be dependent on which local trivialisation one chooses, e.g. how the geometrical vectors that live on the manifold are expressed in a basis for the vector space fibre. As the gauge equivariant convolution is formulated here this issue is not yet addressed and this will require further investigation. This is not touched upon in [11] where Cheng et al. introduces this convolution.

4

Applications to group equivariant CNNs

The second half of this thesis will contain details about ways to implement this framework for CNNs in the case where we have a global symmetry of the plane.

In this chapter we will start with a brief review of the theory of the group equivariant convolutional neural network and will follow up with some implementations. In the implementation part this chapter will cover the two types of group invariant networks that we have implemented — invariant since they deal with classification tasks. These are a network invariant under the group D_4 of 90° rotations of the plane and reflections [9], and a network invariant under the six-fold symmetry of a hexagonal grid [18]. In these networks the symmetry group acts on the plane by global change of basis. The code produced by the authors of the cited papers can be found in their respective `GitHub` [14, 19].

4.1 Group equivariant networks on homogeneous spaces

In this section we will present the closely related theory of the group equivariant convolutional neural networks also presented in [9, 12].

On a homogeneous space the manifold can be written as a quotient space $M = G/H$. Using this as our base space we can immediately construct an H-principal bundle by defining a projection $\pi : G \to G/H$ which maps an element $g \in G$ to a representative in its left coset $g \mapsto \pi(g) = [g] \in gH$. The typical fibre of the bundle is then H.

With this construction we see that functions on the base manifold instead can be seen as functions on the quotient space G/H.

In this section we will use the notation that Cohen et al.[9] used to denote the transformation of a scalar feature map via the group element g as

$$[l_g f](x) = [f \circ g^{-1}](x) = f(g^{-1}x). \tag{4.1}$$

Note that this is Definition 2.3.10 in the case when $\rho = 1$, which corresponds exactly to f being a scalar.

In the example case of g being an element of the group of translations in the plane (i.e. $SE(2)/SO(2)$) we have that $g^{-1}x = x - g$. As we will see the feature maps are easily thought of as functions on the group G after a group convolution and for this we can just replace x in f with an element of the group.

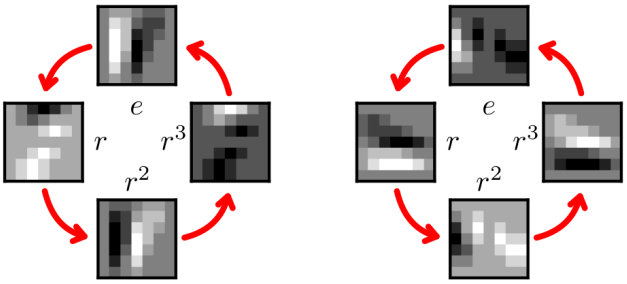


Figure 4.1: A visualisation of how a feature map defined on a group transforms. In practice the feature map has three arguments: the coordinates in $\mathbb{R}^2$ (the pixel numbers) and a group coordinate. Rotating the feature map both rotates the plane coordinates as well as changing the group coordinate. This figure is from [31]; used with permission.

Example 4.1.1 Let $G = SE(2)$ be the space of 2-dimensional translations and rotations in the plane. Let $H = SO(2) \subset G$ denote the rotations in the plane. Then we have that $\mathbb{R}^2 \simeq SE(2)/SO(2)$ is our base manifold, with the structure group $H = SO(2)$.

To visualise G in this case, one can imagine the plane with a circle attached to each point. And since $SO(2) \simeq S^1$ one can interpret functions on G as functions on $S^1 \rtimes \mathbb{R}^2$.

If we were to do a global change of basis, by rotation, on a function f defined locally on $G \simeq H \rtimes G/H$ we would both rotate the G/H argument, e.g. an ordinary rotation of a function defined on $\mathbb{R}^2$, as well as incrementing the H argument to keep track of this transformation. See Figure 4.1 — Figure 1 from [31] — for a visual description of this where the rotations are only the four 90° rotations, which is a symmetry of a square grid.

Cohen presents a generalised convolution of a feature map defined on $\mathbb{Z}^2$ by considering shifts by elements in the group G and not just translations in the plane:

$$[K \star f](g) = \sum_{y \in \mathbb{Z}^2} K(y^{-1}g) f(y) \tag{4.2}$$

where we interpret y as an element in the group of translations in the plane. Comparing this to the case of normal convolution on $\mathbb{Z}^2$:

$$[K \star f](x) = \sum_{y \in \mathbb{Z}^2} K(x - y) f(y), \tag{4.3}$$

the similarities are clear.

The generalised convolution turns the input feature map into a feature map on the group and therefore allows for the transformation behaviour discussed above. For all further convolutions to be defined we must also have the kernels defined as functions on the group which yields the following shape for the future convolutions

$$[K \star f](g) = \sum_{h \in G} K(g^{-1}h) f(h). \tag{4.4}$$

By the group action of the feature map presented in Equation (4.1) we can derive the equivariance of the convolution

$$\begin{aligned}
[K \star l_a[f]](g) &= \sum_{h \in G} K(h^{-1}g) f(a^{-1}h) = \{\ h' = a^{-1}h\} \\
&= \sum_{h' \in G} K(h'^{-1}a^{-1}g) f(h') \\
&= \sum_{h' \in G} K(h'^{-1}(a^{-1}g)) f(h') \\
&= [K \star f](a^{-1}g) \\
&= l_a[K \star f](g).
\end{aligned} \tag{4.5}$$

4.2 General implementation idea

The general implementation idea for the group equivariant networks comes from the discussion of functions on groups in Section 4.1. The plan goes roughly as follows, and the details depend on which framework the implementation is done in:

Given an input image that we would like to convolve over in an equivariant way, under some group H, we first transform each filter with each element of the group H and perform the convolution with respect to all the transformed filters. The transformed channels then form "group channels", i.e. the H coordinate in the discussion in Section 4.1, and a transformation of the input image under H will result in a permutation of the data in the group channels. Note that the set of group channels are equivariant under rotations if one includes the permutation of the channels in the action of rotating an image. In implementations these group channels are then carried to the end of the network and to receive an equivariant result without group channels one must pool over this group coordinate in a permutation invariant way. See Figure 4.2 which depicts this by using an image from MNIST before and after equivariant convolution. If one wants to achieve an invariant network one would then apply some H-invariant function on the equivariant feature map, e.g. taking the max or average of all pixels. This is often useful for classifying tasks, which are the ones these implementations deal with.

Since the number of feature maps increase by $|H|$ in the intermediate layers, due to the group channels, the number of parameters in the kernels will increase, and a way to counteract this is to reduce the number of kernels. This is done in the tests presented later.

Remark 4.2.1 Note that this way of transforming filters imposes shape restrictions on the filters since to achieve exact transformations without interpolations one must map each element of the filters to another place in the filter. For exact rotational equivariance of images on a square grid one must thus use the group of 90° rotations and reflections and only use square filters.

Another way to implement this is to transform the input feature map instead of the filters. This has the advantage that it only needs to be done once in the beginning but with the downside that your dataset becomes as many times larger as you have elements in H. For more details, see the discussion in [20].

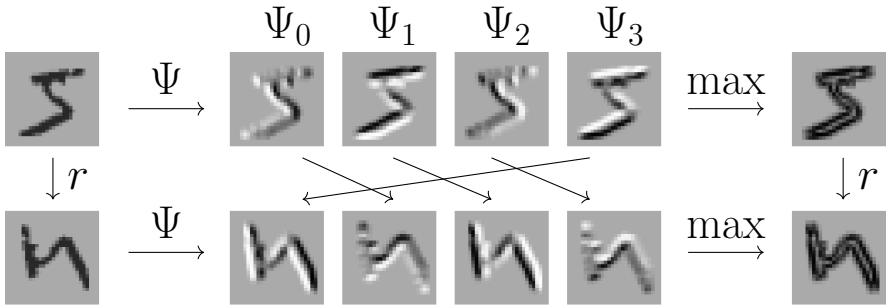


Figure 4.2: Extracted data from a network which uses $H = C_4 \leq D_4 = G$. The input images, related by a 90° rotation are convolved with a single filter transformed for each of the four elements of C_4, creating four "group channels". The result is the group feature map displayed in the middle. It's easy to see that the group channels for one image is related to those of the other by a rotation and a shift of the group channels. To obtain the equivariant feature map these are combined in a permutation invariant way, in this case by taking the max for each pixel. This is called pooling over the group. The colourmap assigns darker colour to larger output.

4.3 A D_4 invariant network by Cohen et.al.

As mentioned previously, the `python` package used for this network is primarily collected from Cohen's `GitHub` [14]. Code from Bas Veeling's `GitHub` was also used [15] which includes a `Keras` [21] implementation of the group convolutional layers for `Tensorflow` [22] that Cohen uploaded to their `GitHub`. The reader who would like to download the code files and run the experiments themselves should be aware that some tinkering is required to get things to work.

In Appendix B we present some examples of convolutional neural networks written in `Python` that uses all features that Cohen published on their `GitHub`; these are also the networks used in the tests presented below.

4.3.1 Testing and results

We test the performance of the example networks presented in Appendix B. The smaller network structure is

i. 2 convolutional layers with *relu* as activation function
ii. max pooling with pooling size 2×2
iii. batch normalisation
iv. 1 convolutional layer with *relu* activation
v. pooling over the network group (for the normal CNN this is an identity layer)
vi. global average pooling
vii. softmax

and the larger network has the following structure

i. 2 convolutional layers with *relu* as activation function
ii. max pooling with pooling size 2×2

iii. batch normalisation
iv. 5 convolutional layers with *relu* as activation function
v. pooling over the network group (for the normal CNN this is an identity layer)
vi. global average pooling
vii. softmax

Note that after the group pooling layer we have obtained an equivariant result, as in the right side of Figure 4.2, and it is the global average pooling that makes the network invariant under the group. Thus the invariance is obtained using the equivariant structure, hence we might be a bit sloppy and refer to these as equivariant network when they actually are invariant under the group.

Both of these network structures were tested by testing all possible combinations of the following:

i. Network invariance (i.e. which group the network is constructed to be invariant under among $\mathbb{Z}^2$, C_4, and D_4). For $\mathbb{Z}^2$ this is just an ordinary CNN where the group pooling layer in the minimal example has been replaced by an identity layer.
ii. Which group we transform data by, either C_4 or D_4. We need not transform the data by $\mathbb{Z}^2$ since convolutional networks already exhibit the wanted equivariance/invariance towards translation.
iii. Whether or not the training data is transformed by the chosen group.
iv. Whether or not the testing data is transformed by the chosen group.

For every combination the corresponding network was trained on the MNIST dataset [13] consisting of 60 000 images of handwritten digits over five epochs. Each trained model was then evaluated on 10 000 MNIST images it had not seen before. The accuracy of the combination is taken as the mean of ten runs.

The observed spread in the network accuracy has several sources. Firstly, all the layers are initiated with random weights, and secondly, the MNIST data is loaded from scratch for each iteration and if data is transformed each image is individually rotated and/or mirrored in a random manner.

The transformation of the images are done by the following functions where we with the word "exact" mean a map that maps pixels one-to-one.

i. For the 90° rotations we used `numpy.rot90` which does not use any interpolation in the rotation; i.e. the rotation is exact
ii. For the mirroring we used `numpy.flip` which also is an exact operation
iii. For the continuous rotations we used `scipy.ndimage.rotate` which relies on the standard representation of 2d rotations

$$R(\theta) = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \tag{4.6}$$

and hence will not map pixels one-to-one, as can be seen in Figure 4.3

Since the networks built to be invariant under $G = C_4$ and D_4 transform their filters during convolution they will essentially have $|G|$ times more channels in their feature maps. This means, since the pooling over the group channels does not happen until the end, they need more parameters in the intermediate layers and to keep the networks at the approximate same number of parameters we reduce the

(a) Normal MNIST image. **(b)** MNIST image rotated by $\theta = 42°$.

Figure 4.3: Example of the continuous rotation. It is clear that the rotation is not exact, i.e. this is not a bijective map between pixels.

number of filters used in the invariant networks roughly by $\sqrt{|G|}$. This results in the smaller network structure having the following number of trainable parameters

$$\begin{aligned} &i.\ \text{Normal CNN: 1940 parameters} \\ &ii.\ C_4\text{CNN: 2755 parameters} \\ &iii.\ D_4\text{CNN: 2841 parameters.} \end{aligned} \tag{4.7}$$

For the larger network we instead get

$$\begin{aligned} &i.\ \text{Normal CNN: 5580 parameters} \\ &ii.\ C_4\text{CNN: 6355 parameters} \\ &iii.\ D_4\text{CNN: 5433 parameters.} \end{aligned} \tag{4.8}$$

We also ran the tests on the network using "valid" padding — i.e. not adding zeros to the edge of the image and therefore the image spatial size diminishes for each convolution — and "same" padding which adds zeros to make sure that the spatial image size remains the same.

Table layout

The data presented in the tables are, as mentioned previously, the average accuracy over 10 runs of training the network for 5 epochs for each combination of structural parameters; i.e. which group the network is invariant under and how the data is transformed. Denoted in the subscript is the standard deviation for those 10 runs.

Each table corresponds to a specific group chosen to act, or not act, on the training and testing data, while each subtable shows the results for a network designed to be invariant under the action of some group: $\mathbb{Z}^2$, C_4, or D_4.

For each data class (training and testing) N indicates that that data was not transformed under the group while T indicates that is was. Since we are interested in the networks capabilities to use the invariant structure obtained from the equivariant layers to recognise transformed data, even if the training data wasn't transformed,

the most interesting elements in the tables below are: Train (N) and Test (T), denoted as NT.

Since these tests were run on the minimal example network they are not optimised for neither accuracy nor loss, but this is purely to demonstrate the effect of invariance, which is clearly seen.

In Table 4.1 we present the accuracy data for the smaller network with "valid" padding where the data transformation group is C_4, and in Table 4.2 one can see the same networks when the data transformation is D_4.

In Table 4.3 we compare the effect of the size of the network, and how the padding impacts the results.

For the sake of space we will only present a representative subset of the data which allows for relevant discussion in the main text. All obtained test data can be seen in Appendix C.

4.3.2 Discussion of testing results

Here we present discussion on some aspects seen in the testing data. In this section we will denote the group structure of the network as G and the group transforming the data as H. When referring to smaller networks in this section we refer to the networks with parameter numbers listed in 4.7, and with larger networks we refer to 4.8. The network structure for these are specified at the beginning of section 4.3.1.

Effect of parameter and filter number

The reduction of the number of filters to keep the number of parameters roughly the same put the C_4 network at 5 filters in the intermediary layers and the D_4 network at 3 filters. Even though the group invariant networks had more parameters they achieved a lower accuracy when testing on untransformed data, which is interesting. Our guess is that when dealing with this low number of filters, each filter is worth more than parameters since the filters determine the amounts of features the network can learn. This effect would be more pronounced when training such a small network since each filter is proportionally a large part of the networks learn ability.

The observed data, presented in the right hand columns in the subtables of Table 4.3, does not conclusively support or conflict with this hypothesis since the accuracy for the large D_4 network is lower than the observed accuracy for the large ordinary CNN: 0.884 against 0.961 with both a standard deviation between 0.03 to 0.04. The large D_4 network has both fewer parameters and fewer filters than the large ordinary CNN. Note that the large ordinary CNN (with same padding) also produced the result $0.855_{\sigma=0.258}$, which indicates that to get a good answer to this one would need to do more than ten runs for the average, and that the group invariant network is of the same approximate learn ability.

For the smaller networks the D_4 network has fewer filters but more parameters compared to the smaller ordinary CNN and the observed accuracy for the D_4 network is 0.578 compared to 0.809 for the ordinary CNN. This difference in accuracy is larger than the accuracy difference for the lager networks, which indicates that effect of a single filter and/or parameter diminishes as their number go up; which is reasonable. Hence it seems like adding the invariant structure to a CNN while

Table 4.1: The results for the three types of CNNs with the smaller network structure and with "valid" padding: ordinary, C_4 invariant, and D_4 invariant. The data, training and testing, is either transformed (T) or not (N) under C_4. Note that the difference between the accuracy for NN (Train: N, Test: N) and NT is basically none once the group the network is invariant under is a subgroup to the data transformation group.

(a) Ordinary CNN

		Test	
		N	T
Train	N	$0.877_{\sigma=0.014}$	$0.343_{\sigma=0.017}$
	T	$0.571_{\sigma=0.044}$	$0.571_{\sigma=0.020}$

(b) C_4 invariant CNN

		Test	
		N	T
Train	N	$0.721_{\sigma=0.044}$	$0.704_{\sigma=0.028}$
	T	$0.707_{\sigma=0.066}$	$0.704_{\sigma=0.045}$

(c) D_4 invariant CNN

		Test	
		N	T
Train	N	$0.685_{\sigma=0.055}$	$0.687_{\sigma=0.059}$
	T	$0.696_{\sigma=0.027}$	$0.679_{\sigma=0.052}$

Table 4.2: The results for the three types of CNNs with the smaller network structure and with "valid" padding: ordinary, C_4 invariant, and D_4 invariant. The data, training and testing, is either transformed (T) or not (N) under D_4. Note that the difference between the accuracy for NN (Train: N, Test: N) and NT decreases the closer the group the network was designed to be invariant under is to the data transformation group.

(a) Ordinary CNN

		Test	
		N	T
Train	N	$0.871_{\sigma=0.017}$	$0.279_{\sigma=0.013}$
	T	$0.530_{\sigma=0.051}$	$0.544_{\sigma=0.046}$

(b) C_4 invariant CNN

		Test	
		N	T
Train	N	$0.695_{\sigma=0.023}$	$0.582_{\sigma=0.029}$
	T	$0.636_{\sigma=0.028}$	$0.679_{\sigma=0.046}$

(c) D_4 invariant CNN

		Test	
		N	T
Train	N	$0.686_{\sigma=0.048}$	$0.680_{\sigma=0.039}$
	T	$0.654_{\sigma=0.028}$	$0.681_{\sigma=0.034}$

Table 4.3: In this table we show the effects of network size and padding for an ordinary CNN and a CNN invariant under D_4. For all accuracies in this table we use untransformed test data and untransformed training data. For more data see Appendix C.

(a) Ordinary CNN

		Padding	
		valid	same
Size	Small	$0.877_{\sigma=0.014}$	$0.809_{\sigma=0.073}$
	Large	$0.778_{\sigma=0.091}$	$0.961_{\sigma=0.030}$

(b) Network invariant under D_4.

		Padding	
		valid	same
Size	Small	$0.685_{\sigma=0.055}$	$0.578_{\sigma=0.059}$
	Large	$0.922_{\sigma=0.074}$	$0.884_{\sigma=0.037}$

keeping the number of parameters roughly the same is more efficient the larger the network is. Although this would need further investigation to produce conclusive results.

Effect of network size and padding

In Table 4.3 we present data to test the effects of changing the network size and whether padding is used. If padding is not used ("valid" padding) then the spatial size of the images will reduce for each convolutional layer, and for larger networks this risks losing data for the network to calculate gradients on. This effect is actually most noticeable in the standard deviation of the observed accuracies; e.g. for the large network (7 convolutional layers) and "valid" padding we observe a higher uncertainty in the obtained accuracy. This comes from that the network essentially only trains in the first epoch and thus becomes much more susceptible to the random initialisation of the filters.

More data can be seen in Appendix C.

Expected accuracy of ordinary CNN for rotated data

The fact that the ordinary CNN had an accuracy of $0.343_{\sigma=0.017}$ when testing on rotated images, see Table 4.1a, is not really surprising. Since the MNIST dataset contains handwritten digits: 0,1,2,3,4,5,6,7,8,9 and some of them look roughly the same a 180° rotation (0,1,8) we would expect to see a higher accuracy than 0.25. In fact, if we would assume that the network learnt to recognise each normally oriented digit with 100% accuracy we would expect the following accuracy for the rotated data

$$\underbrace{0.7}_{\text{for 2,3,4,5,6,7,9}} \cdot \underbrace{0.25}_{\text{one orientation}} + \underbrace{0.3}_{\text{for 0,1,8}} \cdot \underbrace{0.5}_{\text{two orientations}} = 0.325 \tag{4.9}$$

and we expect to see some variance on this. Hence the result of 0.343 is not very surprising.

Data augmentation or group invariance?

This discussion is somewhat covered in [20]. If wanting a classification network to be invariant under a group G then one could take the path of augmenting the training data set by transforming each image in the data set by every element of G, which

would make the augmented data set take $|G|$ times more memory space. To allow for the network to properly learn the transformed data one would probably need more filters as well, hence this approach is very memory intensive.

A more memory efficient approach would be to transform each image by every element in G just as it enters the network and then combine the results at the end of the network, or one could transform the filters by every element in G. Either way, this is more computationally heavy since every convolution will either operate on $|G|$ times more data or use $|G|$ times more filters for each input image.

The exact usage of the network will help dictate which is the more reasonable approach, but as mentioned before these transformations impose restrictions on either the image shape or the filter shape: if one transforms the images the image shape must be invariant under G, and the same goes if one wants to transform the filters.

Effect of group invariance and choice of group

The effect of the invariant structure is clearly seen in Tables 4.1 and 4.2. Note that the difference between training and testing on untransformed (NN) and training on untransformed data whilst testing on transformed (NT) is bigger the smaller the network invariance group G is. This can be seen by comparing NT for the three subtables in Table 4.2. This is also under the condition that G is a subgroup of H. As long as $G \geq H$ the size of G does not matter, which can be seen in Table 4.1 where both the C_4 invariant network and the D_4 invariant network are largely unaffected by transformations by C_4 since $C_4 < D_4$.

The intuition for this is that the invariant network sets up an equivalence class where two images are counted as identical if they differ by an element of the group that the network is constructed under. Hence, for a network constructed under a discrete rotation group, this effectively reduces to possible angles a picture can be rotated by to the angle difference of the elements in the rotation group. This is easily visible if for a trained model you rotate a single image by every angle since the resulting graph becomes periodic with a period of $360/4 = 90°$, see Figure 4.4. This is also a clear indication that increasing the degree of the discrete rotation invariance increases the general rotation invariance, and in the limit the network would be completely rotation invariant. Unfortunately the method with discrete rotation invariance is probably not very suitable for such generalisation and the easier direction would probably be to use the Gaussian filters and complex convolution presented by Worrall et al. [8].

Another example of invariance of these networks can be seen in Figure 4.5 in which the raw output of a C_4 invariant network for a single image, rotated all four ways, is displayed.

4.4 Discussion on discrete group in- and equivariance

The approach of achieving in- and equivariance by transforming the kernels exactly has some severe limitations because this requires the sampling of the image to also be exact under this group. In the case of a flat image a sampling amounts to tiling the plane and there are only three regular tilings of the plane: the square tiling (normal square image pixels), triangle tiling, and hexagonal tiling. If one wants to only use regular tilings to sample the image then the largest symmetry on can use is D_6.

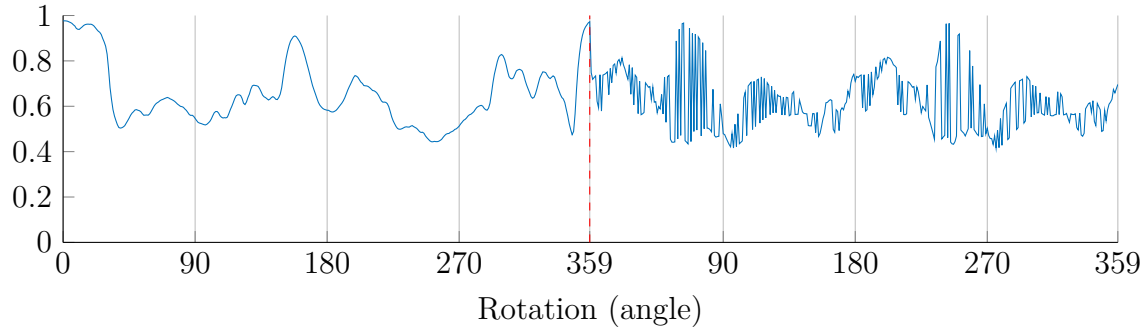
There are ways around this if one loosens on the criteria that the transformation of the kernels need to be exact (i.e. a bijective mapping between elements of the kernel) and instead uses Gaussian kernels. This allows for more general rotations with a high, but not exact, degree of in-/equivariance by simply rotating the angle component of the kernel, see [23, 24].

4.5 A C_6 equivariant network by Hoogeboom et. al.

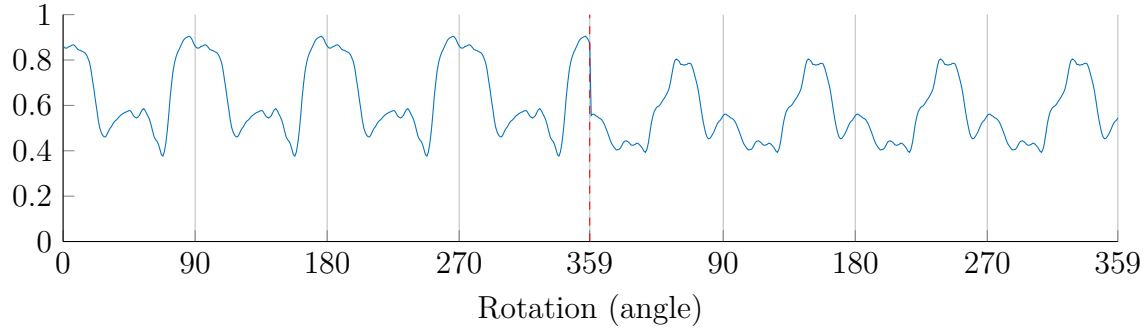
Since the network performs better on transformed data the larger the group it is constructed to be invariant under is it is useful to look at ways to use a larger group. This is what Hoogeboom et. al. did in their paper "HexaConv" [18] in which they sample images to a hexagonal grid in order to let the network to be invariant under D_6. The general idea of the implementation is still the same: transform each filter for each element in the group, perform the convolution with the transformed filters, and at the end pool over the group channels.

We will not go into further detail here about the implementation of the hexagonal grid but will refer to their paper for that. An example of the hexagonal sampling can be seen in Figure 4.6.

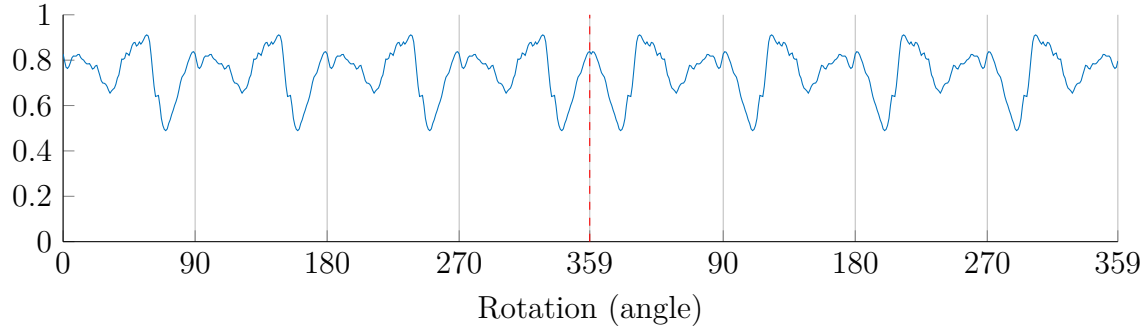
Cohen et al. used the hexagonal structure presented in [18] to create a CNN operating on spherical data by approximating the sphere with a triangulated icosahedron [25]. This approach with a local symmetric triangulation — i.e. that the triangulation locally maps to itself under e.g. rotations — seems to indicate a possible method to perform these convolutions on an arbitrary manifold, but even if a locally symmetrical triangulation is possible one would need to have some kind of transition criteria between different local triangulations so this approach might in reality be quite restricted. This will require further research.



(a) Network invariant under $\mathbb{Z}^2$.



(b) Network invariant under C_4.



(c) Network invariant under D_4.

Figure 4.4: Each figure corresponds to a trained model designed to be invariant under some group. What is plotted is the norm of the model output for a single input image. The input image is transformed, on the left side of each figure, by just rotating the image by the noted angle (x-axis), and on the right side the image is first mirrored horizontally or vertically (randomly chosen) and then rotated by the noted angle. The network designed to be invariant only under $\mathbb{Z}^2$ is clearly unpredictable when the input is rotated, as is expected. The networks designed to be invariant under C_4 and D_4 show the expected property that the invariance quotient away all rotation angles outside of 0-90°. On the side corresponding to mirrored the image both the C_4 and D_4 show the same periodic behaviour, which is expected, but the C_4 network show a different profile compared to just the rotated image, while the D_4 network keeps the same profile, albeit mirrored due to the mirrored image.

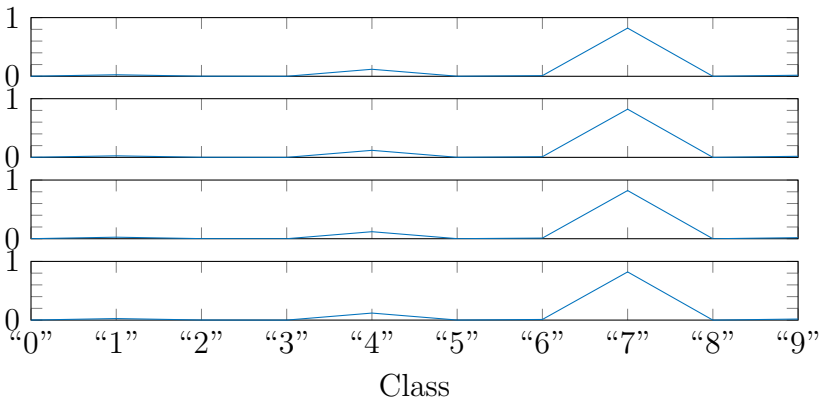


Figure 4.5: The output of a C_4 invariant network for a single image rotated through all 90° rotations. On the y-axis for each graph is displayed the probability the network assigns the digit to be a specific class.

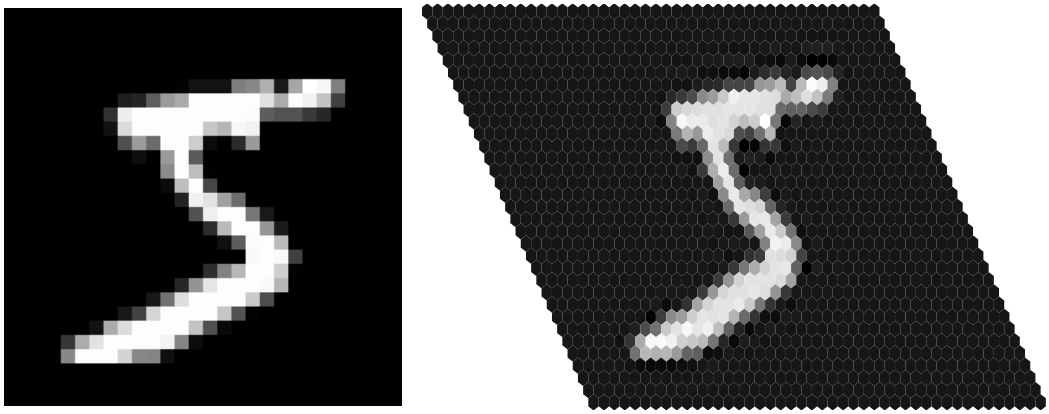


Figure 4.6: A sample of the MNIST data set displayed on the left in the ordinary square pixel sampling and on the right sampled in a hexagonal grid. The visualisation of the hexagonal sampling is done using a plotting function written by Hoogeboom [18], and even though the hexagonal sampling uses regular hexagons this plotting function displays them a bit elongated. Hence the number looks a bit stretched vertically.

5

Summary, conclusion and outlook

This thesis has presented existing theory of Cheng et al. [11] — in which they presented a gauge equivariant convolution — and made their implied details explicit, as well as added the new interpretation of layers in a neural network as parts of a map defined on equivalence classes of associated bundles; equivariance of the neural layer is then equivalent to the map between associated bundles being well defined.

We have also presented the theory for the group equivariant networks used in [9], written a small example of a convolutional neural network using this structure to classify MNIST digits invariant under 90° rotation, and tested this network to show the effect of adding the equivariant structure to the convolutional layers.

In testing this structure we observe that the group invariant networks are able to efficiently use the group invariance in order to generalise training done on untransformed data to transformed data. Compared to an ordinary CNN the group invariant networks, when modified so that their parameter numbers where roughly the same, had the same approximate accuracy when testing on both untransformed data and transformed data while the ordinary CNN performed much worse on transformed data. So in conclusion, the group invariant networks perform as well as their ordinary counterpart whilst also being able to recognise transformed data without needing to train on it.

This can be applied, and in some extent have been applied, to classification of objects in for example satellite images, medical images, and weather patterns. It would be interesting to see if something could be obtained by letting a network with a group structure act on the cosmic microwave background radiation (CMB) in order to detect patterns.

Going forward, an interesting application in physics could be to apply this approach to other gauge symmetries other than the local change of geometric coordinates. For example, in constructing the standard model one builds the most general theory that is invariant under the currently observed symmetry group $SU(3) \otimes SU(2) \otimes U(1)$ under some additional constraints, e.g. anomaly cancellation. The best possible scenario would then be if one could construct a network that could learn symmetries of a dataset. If so one could apply this to analyse data from CERN using this framework to look for new symmetries in the data which might help with building new models or finding new angles to attack the standard model.

It would also be interesting to see developments — interpretational, mathematical, and implementations — of geometry that changes between layers.

A

Theoretical background

In this appendix we present relevant theory that readily found in textbooks on each subject but consolidated here for convenience.

A.1 Topology

We start with introducing the most basic structure one can place on a set: a topology. Intuitively when we define a topology on a set we just say which subset that are open, and this will allow us to expand the definition of continuity to work on general spaces. Let's start by defining a topology.

Definition A.1.1 (Topology) Let X be a set and $\mathcal{P}(X)$ the powerset — the set of all subsets — of X. A *topology* on X, denoted $\mathcal{T}_X$, is a subset $\mathcal{T}_X \subseteq \mathcal{P}(X)$ which satisfies the following conditions:

i. $\emptyset \in \mathcal{T}_X$ and $X \in \mathcal{T}_X$;
ii. for I a finite index set and $U_i \in \mathcal{T}_X$ $i \in I$ then $\bigcap_{i\in I} U_i \in \mathcal{T}_X$;
iii. for I an arbitrary index set and $U_i \in \mathcal{T}_X$ $i \in I$ then $\bigcup_{i\in I} U_i \in \mathcal{T}_x$.

I.e. a topology is a subset of the powerset which is closed under finite intersections and arbitrary unions.

With topology defined we define a subset $U \subseteq X$ as open if $U \in \mathcal{T}_X$.

Definition A.1.2 (Topological space) A *topological space* is simply a set X along with its topology $\mathcal{T}_X$ and is formally denoted $(X, \mathcal{T}_X)$. For the sake of convenience we will often just say that X is a topological space and leave $\mathcal{T}_X$ implicit.

Before we can continue to the definition of a continuous function we need the concept of preimage, which is a generalisation of the inverse of a function.

Definition A.1.3 (Preimage) Let $f : X \to Y$ be a map between the sets X and Y. The *preimage* is then defined as

$$\text{preim}_f : Y \to X \tag{A.1.1}$$

$$y \mapsto \text{preim}_f(y) := \{x \in X \mid f(x) = y\}. \tag{A.1.2}$$

When each $y \in Y$ has a unique preimage then $\text{preim}_f = f^{-1}$. For convenience we will often just write f^{-1} and use this to refer to preim_f even when f might not have an inverse.

Now we have the tools to define continuity and the structure preserving maps for topological spaces: homeomorphisms.

Definition A.1.4 (Continuous function) A function $f : (X, \mathcal{T}_X) \to (Y, \mathcal{T}_Y)$ between two topological spaces is *continuous* if

$$\forall V \in \mathcal{T}_Y : \ \text{preim}_f(V) \in \mathcal{T}_X. \tag{A.1.3}$$

I.e. a function is continuous if the open sets in the target space are mapped to by open sets in the domain.

Definition A.1.5 (Homeomorphism) A homeomorphism is a continuous function with continuous inverse.

If you have previously only seen the definition of continuity from analysis using $\varepsilon - \delta$ this might seem weird since the continuity of a function now depends on the topology on the target and domain.

Example A.1.6 Let $X = \{1, 2, 3\}$ and $Y = \{1, 2, 3\}$ with the topologies:

$$\mathcal{T}_X = \{\emptyset, \{1, 2, 3\}, \{1, 2\}\} \text{ and } \mathcal{T}_Y = \mathcal{P}(Y). \tag{A.1.4}$$

Furthermore, let $f : X \to Y$ by the action $x \mapsto f(x) = x$, and $g = f^{-1} : Y \to X$. Then g is continuous since all sets in $\mathcal{T}_X$ exist in $\mathcal{T}_Y$. While g is continuous, f is not, since for example $\text{preim}_f(\{2\} \in \mathcal{T}_Y) = \{2\} \notin \mathcal{T}_X$.

Definition A.1.7 (Countable basis) A topological space X has a *countable basis* for its topology if there exists open sets

$$\{U_i \in \mathcal{T}_X | i \in \mathbb{N}\} \tag{A.1.5}$$

such that every open set $U \subseteq X$ can be written as

$$U = \bigcup_{i \in I} U_i. \tag{A.1.6}$$

Definition A.1.8 (Open cover) An *open cover* of a topological space X is the set $\{U_\alpha \in \mathcal{T}_X | \alpha \in I\}$ such that

$$\bigcup_{\alpha \in I} U_\alpha = X. \tag{A.1.7}$$

Definition A.1.9 (Quotient topology) Let X be a topological space and $\sim$ an equivalence relation. Then we denote an equivalence class as

$$[x]_\sim = \{y \in X | x \sim y\} \tag{A.1.8}$$

and the set of equivalence classes as

$$X/\sim = \{[x]_\sim | x \in X\}. \tag{A.1.9}$$

Let Π be the quotient map as $\Pi : X \to X/\sim$ by $x \mapsto [x]_\sim$. Then the quotient topology is

$$\{V \subseteq X/\sim \ | \Pi^{-1}(V) \text{ is open in } X\}. \tag{A.1.10}$$

A.2 Manifolds

The general spaces we will be working with are based on manifolds, or more specifically topological manifolds, but before we can define manifolds we need to define atlases and locally Euclidean.

Definition A.2.1 (Locally Euclidean) A topological space X is *locally Euclidean* of dimension d if every point $p \in X$ has a neighbourhood U such that there is a homeomorphism ϕ from U to an open set $\phi(U) \subseteq \mathbb{R}^d$. We say that such a space X is locally homeomorphic to $\mathbb{R}^d$.

Now we can present the definition of a manifold.

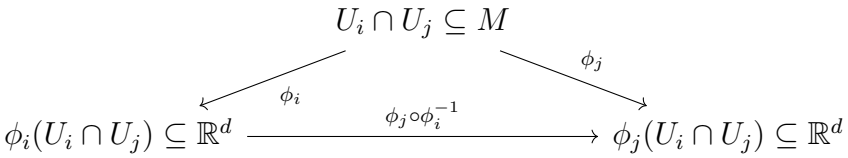
Definition A.2.2 (Topological manifold) A *topological manifold* M of dimension d is a topological Hausdorff space that has a countable basis for its topology and is locally homeomorphic to $\mathbb{R}^d$.

Definition A.2.3 (Chart and atlas) A *chart* on an d-dimensional manifold M is a pair (U, ϕ) where U is an open set (i.e. in the topology $\mathcal{T}_M$) and $\phi : U \to \phi(U) \subseteq \mathbb{R}^d$ is a homeomorphism. A collection of charts $\mathcal{A} = \{(U_\alpha, \phi_\alpha) | \alpha \in I\}$ is called an *atlas* if $\{U_\alpha\}$ covers M.

Now we want to add differentiable structure to the manifold. To do this we introduce chart transition maps.

Definition A.2.4 (Chart transition map) Let M be a topological manifold of dimension d with atlas $\mathcal{A} = \{(U_i, \phi_i)\}$ and let $p \in M$ be a point. Choose two charts (U_i, ϕ_i) and (U_j, ϕ_j) such that $p \in U_i \cap U_j$. Then $\phi_j \circ \phi_i^{-1} : \phi_i(U_i \cap U_j) \to \phi_j(U_i \cap U_j)$ is a chart transition map.

Since each ϕ_i maps locally from M to $\mathbb{R}^n$ chart transition maps are functions from open sets in $\mathbb{R}^n$ to open sets in $\mathbb{R}^n$ and as such they can be smooth. An atlas where all transition functions are smooth is called a smooth atlas.



Two smooth atlases $\mathcal{A}$ and $\mathcal{B}$ are said to be equivalent if $\mathcal{A} \cup \mathcal{B}$ also is a smooth atlas. They are also sometimes referred to as being smoothly compatible. This results in a natural definition of an equivalence class of smooth atlases and an equivalence class under this relation is called a smooth structure.

Definition A.2.5 (Smooth manifold) A *smooth manifold* is a topological manifold with a smooth structure.

Since we aren't able to differentiate on manifolds we will always use this structure of studying charts maps from the manifold to $\mathbb{R}^d$. To make this explicit we have the following definition.

Definition A.2.6 (Smooth map between manifolds) . Let M_1 and M_2 be smooth manifolds and let $f : M_1 \to M_2$ be a continuous map. f is called smooth at a point $p \in M_1$ if there exists charts $(U_1, \phi_1) \in \mathcal{A}_1$ and $(U_2, \phi_2) \in \mathcal{A}_2$ such that $p \in U_1 \cap f^{-1}(U_2)$ and

$$\phi_2 \circ f \circ \phi_1^{-1} : \phi_1(f^{-1}(U_2)) \to \phi_2(U_2). \tag{A.2.1}$$

If f is smooth for all $p \in M_1$ then f is a smooth map.

$$\begin{array}{ccc} U_1 \subseteq M_1 & \xrightarrow{\quad f \quad} & U_2 \subseteq M_2 \\ \downarrow \phi_1 & & \downarrow \phi_2 \\ \phi_1(U_1) \subseteq \mathbb{R}^d & \xrightarrow{\phi_2 \circ f \circ \phi_1^{-1}} & \phi_2(U_2) \subseteq \mathbb{R}^d \end{array}$$

The next definition for the part on manifolds is the structure preserving maps for manifolds: diffeomorphisms.

Definition A.2.7 (Diffeomorphism) Let M_1 and M_2 be smooth manifolds, then a map $f : M_1 \to M_2$ is a *diffeomorphism* if it is smooth and has a smooth inverse.

Next we define the maximal atlas:

Definition A.2.8 (Maximal atlas) A *maximal atlas* on a manifold M is the atlas given by

$$\mathcal{A}_M = \{(U, \phi : U \to U') | U \subseteq M \text{ is open}, U' \subseteq \mathbb{R}^n \text{ is open}, \phi \text{ is a diffeomorphism}\}.$$

A crucial point will be the tangent space to a manifold, and later the tangent bundle. To begin we define a smooth curve.

Definition A.2.9 (Smooth curve) A *curve* γ on a smooth manifold M is a map from an open subset $I \subseteq \mathbb{R}$ to M and is smooth at $p \in M$ if there exists a chart (U_i, ϕ_i) such that $p \in U_i$ and $\phi_i \circ \gamma : \mathbb{R} \to \mathbb{R}^d$ is smooth. A *smooth curve* is a curve that is smooth at each point.

$$\begin{array}{ccc} I \subseteq \mathbb{R} & \xrightarrow{\gamma} & M \\ & \searrow^{\phi \circ \gamma} & \downarrow \phi \\ & & \mathbb{R}^d \end{array}$$

Definition A.2.10 (Equivalence of curves) Let γ_1 and γ_2 be two curves such that $\gamma_1(0) = \gamma_2(0) = p \in U \subset M$ with (U, ϕ) being a chart on M, then $\gamma_1 \sim \gamma_2$ iff

$$\frac{d}{dt}(\phi \circ \gamma_1)(t)|_{t=0} = \frac{d}{dt}(\phi \circ \gamma_2)(t)|_{t=0}. \tag{A.2.2}$$

Definition A.2.11 (Tangent space) The *tangent space* of M at $p \in M$ is denoted T_pM and is the set of all equivalence classes of curves $\gamma : I \subseteq \mathbb{R} \to M$ such that $\gamma(0) = p$.

The chart (U, ϕ) induces a map $\Psi_\phi : T_pM \to \mathbb{R}^d$ by mapping

$$[\gamma] \mapsto \frac{d}{dt}(\phi \circ \gamma)(t)|_{t=0}. \tag{A.2.3}$$

This map is a linear isomorphism between T_pM and $\mathbb{R}^d$ and can be identified with the directional derivative $D_p\phi : T_pM \to \mathbb{R}^d$. This has an inverse map $D_{\phi(p)=x}\phi^{-1} : \mathbb{R}^d \to T_pM$ and act on basis vectors e_i of $\mathbb{R}^d$ through

$$(D_x f)(e_i) = \frac{d}{dt} f(x + te_i)|_{t=0} = \frac{\partial f(x)}{\partial x^i} \tag{A.2.4}$$

or in other words

$$e_i \mapsto \frac{\partial}{\partial x^i} \tag{A.2.5}$$

and since T_pM is isomorphic to $\mathbb{R}^d$ we have that $\partial/\partial x^i$ span the tangent space at p or with more precise notation

$$\left(\frac{\partial}{\partial x^1}\right)_p \cdots \left(\frac{\partial}{\partial x^n}\right)_p \tag{A.2.6}$$

span T_pM.

A.3 Group theory and representations

We will in this section present relevant background within group theory and representations.

A.3.1 Group theory

We start by stating the definition of a group, which can be found almost everywhere and also here for completeness.

Definition A.3.1 (Group) A *group* is a set G together with a binary operator $\varphi : G \times G \to G$ sometimes denoted as (G, φ) which satisfies a the following:

i. It exists an element $e \in G$ called the *identity* such that $\varphi(e, g) = \varphi(g, e) = g$ for all $g \in G$.

ii. For each element $g \in G$ there exists $g^{-1} \in G$ called the *inverse* of g such that $\varphi(g, g^{-1}) = \varphi(g^{-1}, g) = e$.

iii. The set G is closed under the action of φ, i.e. for all $g, h \in G : \ \varphi(g, h) \in G$.

iv. The operator φ is associative, i.e. $\varphi(\varphi(a, b), c) = \varphi(a, \varphi(b, c))$.

Normally one does not write the operator but rather $\varphi(g, h) = gh$.

Definition A.3.2 (Subgroup) Given a group G, a *subgroup* (H, φ) consists of a subset $H \subseteq G$ which is a group under φ. This is shorthanded as $H \leq G$.

Definition A.3.3 (Left/Right coset) Given a group G, an element $g \in G$, and a subgroup $H \leq G$, then we define the

- *left coset* as $gH = \{gh | h \in H\}$

- *right coset* as $Hg = \{hg | h \in H\}$.

Since H is a subgroup $e \in H$ and thus $g \in gH$ for all $g \in G$. We call g the representative of the coset gH. The same argument goes for right cosets. We can also define a so called section $s : G/H \to G$, more on this in the section on manifolds, see Section A.4, such that $s(H) = e$ and that $\pi \circ s = id_{G/H}$.

Definition A.3.4 (Normal subgroup) A subgroup $H \leq G$ is called *normal* if $gH = Hg$ for all $g \in G$.

Definition A.3.5 (Quotient group) The *quotient group* G/H is defined as the set of all cosets, i.e.

$$G/H = \{gH | g \in G\}. \tag{A.3.1}$$

The group G with a subgroup $H \leq G$ comes with a natural (surjective) projection map $\pi : G \to G/H$ which acts as $\pi(g) = gH$.

Theorem A.3.6 (Independence of representative) *An element gH of the coset group G/H is independent of its representative; i.e. for $p, q \in gH$ it holds that $gH = pH = qH$.* □

Proof Let $G/H \ni gH$ be a coset for some $g \in G$ and let $g' \in gH$. Then since $g' \in gH$ it exists an $h \in H$ such that $g' = gh$. Since H is a subgroup $h^{-1} \in H$ and thus we get $gH = (g'h^{-1})H = g'(h^{-1}H) = g'H$ where we used the associativity of the group operator in the second equality. ■

Theorem A.3.7 (Coset partition of G) *Let G be a group and $H \leq G$ a subgroup, then the elements of G/H are disjoint and*

$$\cup_{A \in G/H} A = G. \tag{A.3.2}$$

□

Proof Let G be a group, $H \leq G$ subgroup, and let $C_1 = g_1 H$ and $C_2 = g_2 H$ be different cosets; i.e. neither $C_1 \setminus C_2$ nor $C_2 \setminus C_1$ are the empty set. Then let $g \in C_1 \cap C_2$ and therefore there exists $h_1, h_2 \in H$ such that $g = g_1 h_1 = g_2 h_2$ since g is in both C_1 and C_2. Since H is a subgroup there exists inverses of h_1 and h_2 in H which gives $g_1 = g_2 h_2 h_1^{-1} = g_2 h'$ with $h' \in H$. Putting this together we get $C_1 = g_1 H = (g_2 h')H = g_2 H = C_2$. Thus if $C_1 \cap C_2 \neq \emptyset$ then $C_1 = C_2$. For the cover part we have that $g \in gH$ and $G/H = \{gH | g \in G\}$ and the result follows. ■

Definition A.3.8 (Action of a group on a set) Let G be a group and X be a set. We say that G acts on X if there exists a map $\Psi : G \times X \to X$.

Definition A.3.9 (Transitive group action) Given a group G, and a set X then G is said to act transitively on X if for each $x, y \in X$ there exists a $g \in G$ such that $\Psi(g, x) = y$.

A special case of this is when X is the quotient group G/H: G acts transitively on

G/H since if gH and pH are two different cosets then the group element pg^{-1} maps gH onto pH. In this case we write the transitivity condition as $\forall x, y \in X\ \exists g \in G : gx = y$ since the group action φ for G can be used as Ψ on elements of X as well.

We will now introduce a definition which can come in useful when we will talk about fibre bundles.

Definition A.3.10 (Twist map) Given a group G, a subgroup $H \leq G$, their quotient group G/H, and a section $s : G/H \to G$ we define $h : G/H \times G \to H$ as $h(x, g) = (s(gx))^{-1} g s(x)$, or we can also define $h(x, g)$ as the element of H which satisfies $gs(x) = s(gx)h(x, g)$.

The map h encapsulates the "twisty-ness" of the section, so that when we define a section we can determine the map h such that Definition A.3.10 is satisfied.

A further fundamental object that is required for the discussion of principal and associated bundles is Lie groups. There is a lot of theory developed for Lie groups and we will only present that which is absolutely necessary for the theory needed for this thesis.

Definition A.3.11 (Lie group) A *Lie group* is a group (G, ϕ) such that G is a finite-dimensional smooth manifold and the group multiplication $\phi : G \times G \to G$ as well as group inversion $\bullet^{-1} : G \to G$ (by $g \mapsto g^{-1}$) are smooth functions.

Definition A.3.12 (Left and right action) Let (G, ϕ) be a Lie group and M a smooth manifold. The *left action* of G on M is a map

$$\triangleright : G \times M \to M \tag{A.3.3}$$

that satisfies

i. for all $p \in M$ we have $e \triangleright p = p$, where e is the identity of G

ii. for all $p \in M$ and $g_1, g_2 \in G$ we have $g_1 \triangleright (g_2 \triangleright p) = (g_1 g_2) \triangleright p$.

The *right action* of a Lie group (H, ψ) on a smooth manifold M is similarly a map $\triangleleft : M \times H \to M$ that satisfies

i. $p \triangleleft e = p$ for all $p \in M$

ii. for all $h_1, h_2 \in H$ and $p \in M$ we have $(p \triangleleft h_1) \triangleleft h_2 = p \triangleleft (h_1 h_2)$

Definition A.3.13 (Equivariance of Lie groups) Let (G, ϕ) and (H, ψ) be two Lie groups, $\rho : G \to H$ a homomorphism between them, and M, N be two smooth manifolds with $f : M \to N$ a smooth map between them. Furthermore, let G, H act on M, N from the left

$$\begin{aligned} \triangleright &: G \times M \to M \\ \triangleright' &: H \times N \to N. \end{aligned} \tag{A.3.4}$$

Then the map f is called *ρ-equivariant* if

$$f(g \triangleright m) = \rho(g) \triangleright' f(m) \tag{A.3.5}$$

i.e. the diagram commutes

$$\begin{array}{ccc} G \times M & \xrightarrow{\rho \times f} & H \times N \\ \downarrow{\scriptstyle \triangleright} & & \downarrow{\scriptstyle \triangleright'} \\ M & \xrightarrow{\quad f \quad} & N \end{array}$$

Some final definitions regarding Lie groups.

Definition A.3.14 (Orbit and stabiliser) Let $\triangleright : G \times M \to M$ be a left action of a Lie group G on a smooth manifold M. Then we define the following.

i. The *orbit* of a point $p \in M$ is the set $\mathcal{O}_p = \{a \in M \mid \exists g \in G : g \triangleright p = a\}$

ii. Define the equivalence relation such that two points are equivalent $p \sim_{\mathcal{O}} q$ iff $p \in \mathcal{O}_q$, i.e. $\exists g \in G : p = g \triangleright q$

iii. The *stabiliser* for a point $p \in M$ is the set $\mathcal{S}_p = \{g \in G \mid g \triangleright p = p\}$

Proof (Equivalence relation) It is very straight forward to show that the proposed relation is indeed an equivalence relation:

i. *Reflexivity*: $e \in G$

ii. *Symmetric*: $p \sim q$ means $\exists g \in G : p = g \triangleright q$ but that gives $g^{-1} \triangleright p = q$

iii. *Transitive*: The group acts transitively on itself which gives the transitivity of $\sim_{\mathcal{O}}$. ∎

Definition A.3.15 (Orbit space) Let M, G, and $\triangleright$ be as in previous definition. Then the space $M/\sim_{\mathcal{O}} =: M/G$ is called the *orbit space* of M under G.

Definition A.3.16 (Stabiliser) Given M, G and $\triangleright$ as above, then the stabiliser of a point $p \in M$ is $\mathcal{S}_p = \{g \in G \mid g \triangleright p = p\}$.

Definition A.3.17 (Free action) A left action of G on M is *free* if $\mathcal{S}_p = \{e\}$ for all $p \in M$.

A.3.2 Representations

One can study aspects of a group by letting it act on other spaces.

Definition A.3.18 (Representation) Let G be a group and V a vector space (over $\mathbb{R}$) then $\rho : G \to GL(V)$ is a representation of G on V if for all $g, h \in G$ we have that $\rho(gh) = \rho(g)\rho(h)$. Here $GL(V)$ is the general linear group on V (i.e. has as elements all bijectively linear maps from V to V) . The kernel of a representation is its preimage of the identity, i.e. $\operatorname{Ker} \rho = \{g \in G | \rho(g) = \mathrm{id}_V\}$.

With this definition we can explicitly extend the definition of the action of a group G on a vector space where $\Psi : G \times X \to X$ now acts as $\Psi(g, x) = \rho(g)x$ for a representation $\rho : G \to GL(X)$.

In this thesis we will only work with vector spaces V of finite dimension n, and in that case $GL(V)$ is the set of $n \times n$ invertible matrices with respect to some chosen basis on V.

Definition A.3.19 (Faithful) A representation $\rho : G \to GL(V)$ is *faithful* if it is injective, i.e. every distinct element $g \in G$ is mapped to a distinct linear map $\rho(g)$.

A.4 Fibre bundles

If one tries to define global functions on a general curved manifold one will encounter problems. To generalise the function concept we need the formalism of fibre bundles. The theory presented here is heavily based upon that presented in [26, 27]

Definition A.4.1 (Bundle) A triple (E, π, M), sometimes denoted $E \xrightarrow{\pi} M$, where E is a topological space called the *total space*, M is a topological space called the *base space*, and $\pi : E \to M$ is a continuous and surjective projection map, is called a *(fibred) bundle*.

We will in general let the base space be a smooth manifold.

Definition A.4.2 (Fibre at a point) Let $E \xrightarrow{\pi} M$ be a bundle and a point $p \in M$, then $\text{preim}_\pi(\{p\})$ is the *fibre* at p.

Definition A.4.3 (Fibre bundle) The bundle (E, π, M) is called a *fibre bundle* if there exists a *canonical fibre* F such that the fibre F_p over each point $p \in M$ is homeomorphic to this fibre. As a tuple the bundle is denoted (E, π, M, F).

A fibre bundle can also be defined in terms of local trivialisations.

Definition A.4.4 (Local trivialisation) A *local trivialisation* is a pair (U_i, φ_i) where U_i is an open set in the base space M and φ_i is a map

$$\begin{aligned} \varphi_i : \text{preim}_\pi(U_i) &\to U_i \times F \\ x &\mapsto (\pi(x), f_i(x)) \end{aligned} \tag{A.4.1}$$

where $f_i : \text{preim}_\pi(p) \to F$ is a homeomorphism since $F_p = \text{preim}_\pi(p) \sim F$.

Definition A.4.5 (Fibre bundle by trivialisations) A bundle (E, π, M) is called a fibre bundle if there for each point $p \in M$ is an open neighbourhood U_p such that there exists a homeomorphism $\varphi_p : \pi^{-1}(U_p) \to U_p \times F$, i.e. (U_p, φ_p) is a local trivialisation.

The local trivialisations will, by the definition above, map the fibre F_p at a point $p \in B$ to $\{p\} \times F$.

Definition A.4.6 (Transition function) Given a fibre bundle (E, π, M) with fibre F and two local trivialisations (U_i, φ_i) and (U_j, φ_j) such that $p \in U_i \cap U_j$ we can define a *transition function* between these trivialisations through

$$\Psi_{ij} = \varphi_i \circ \varphi_j^{-1} : U_j \times F \to U_i \times F \tag{A.4.2}$$

$$(p, v) \mapsto (p, f_i(f_j^{-1}(v))). \tag{A.4.3}$$

Often when we use the word transition function we are referring to the function $f_i \circ f_j^{-1} : F \to F$ which determines how the fibre "changes" when we change local trivialisations. Note that since f_i is a homeomorphism then $f_i \circ f_j^{-1}$ is a homeomorphism as well.

Sometimes the bundle (E, π, M, F) is denoted diagrammatically as

$$\begin{array}{ccc} F & \longrightarrow & E \\ & & \downarrow{\scriptstyle \pi} \\ & & M \end{array}$$

An important concept will be bundle morphisms and isomorphisms.

Definition A.4.7 (Bundle morphism) Let $E \xrightarrow{\pi} M$ and $E' \xrightarrow{\pi'} M'$ be two bundles and $u : E \to E'$, $v : M \to M'$ be maps between their total spaces and base spaces respectively. Then the pair (u, v) is called a *bundle morphism* if

$$\pi' \circ u = v \circ \pi. \tag{A.4.4}$$

I.e. the diagram commutes.

$$\begin{array}{ccc} E & \xrightarrow{u} & E' \\ \downarrow{\scriptstyle \pi} & & \downarrow{\scriptstyle \pi'} \\ M & \xrightarrow{v} & M' \end{array}$$

Definition A.4.8 (Bundle isomorphism) Two bundles $E \xrightarrow{pi} M$, $E' \xrightarrow{\pi'} M'$ are isomorphic as bundles if there exists two bundle morphisms (u, v) and (f, g) such that the diagram commutes:

$$\begin{array}{ccc} E & \underset{f}{\overset{u}{\rightleftarrows}} & E' \\ \downarrow{\scriptstyle \pi} & & \downarrow{\scriptstyle \pi'} \\ M & \underset{g}{\overset{v}{\rightleftarrows}} & M' \end{array}$$

Since the commuting requires $f \circ u = \mathrm{id}_E$ and $g \circ v = \mathrm{id}_M$ one denotes $(f, g) = (u^{-1}, v^{-1})$.

A fibre bundle (E, π, M) with typical fibre F is said to be trivial if it is isomorphic as bundles to the product bundle $(M \times F, \pi', M)$ where $\pi'(p, f) = p$.

On general manifolds one cannot talk about functions any more but rather the generalised concept of sections.

Definition A.4.9 (Section) A *section* is a continuous map $s : M \to E$ that satisfies $\pi \circ s = \mathrm{id}_M$.

There can be different structures added on top of the bundle formalism to give different types of bundles with different properties as is discussed in the following sections.

A.4.1 Principal G-bundle

Definition A.4.10 (Principal G-bundle) A bundle (E, π, M) is a *principal G-bundle* if it satisfies the following conditions:

1. The total space E is equipped with a right action of G, sometimes denoted $\triangleleft G$

2. The action $\triangleleft G$ on E is free
3. (E, π, M) is isomorphic as a bundle to $(E, \pi', E/G)$ where $\pi'(x) = [x]$. I.e. each element of E is projected down to a representative of its orbit.

A principal bundle is sometimes graphically denoted as

$$E \xleftarrow[\triangleleft G]{} E \xrightarrow{\pi} M$$

The group G is sometimes referred to a the *structure group* of the bundle.

Remark A.4.11 This is a fibre bundle with typical fibre G. This can be seen by $\pi'^{-1}([x]) = \{y \in E \mid \pi(y) = [x]\} = \{y \in E \mid \exists g \in G : y = x \triangleleft g\} = \mathcal{O}_x$ and $\mathcal{O}_x \simeq G$ since the action of G was free. Hence this is a fibre bundle with typical fibre G. This also shows that the action of G preserves the fibres.

Definition A.4.12 (Principal bundle map) Let (E, π, M) and (E', π', M') be two principal bundles the groups G and G', with possibly different right actions $\triangleleft, \triangleleft'$. If we have a Lie group homomorphism $\rho : G \to G'$, then the pair of maps (u, v) is called a principal bundle map if the following criteria are satisfied

i. $v \circ \pi = \pi' \circ u$

ii. $u(p \triangleleft g) = u(p) \triangleleft' \rho(g)$ (this is the equivariance property from Definition A.3.13.)

Graphically, this is the same as saying that the following diagram commutes

$$\begin{array}{ccc} E & \xrightarrow{u} & E' \\ {\scriptstyle \triangleleft G}\uparrow & & \uparrow{\scriptstyle \triangleleft' G'} \\ E & \xrightarrow{u} & E' \\ \downarrow{\scriptstyle \pi} & & \downarrow{\scriptstyle \pi} \\ M & \xrightarrow{v} & M' \end{array}$$

Example A.4.13 (The frame bundle) Let M be a d-dimensional smooth manifold and denote the tangent space at $p \in M$ as T_pM. Then $T_pM \simeq \mathbb{R}^d$ and we call the set of all possible bases for T_pM as

$$L_pM = \{(e_1, \ldots, e_d) \mid (e_1, \ldots, e_d) \text{is a basis for} T_pM\} \simeq GL(d, \mathbb{R}), \tag{A.4.5}$$

which one can see as all possible bases for $\mathbb{R}^d$. We then define the *frame bundle* as

$$LM = \bigsqcup_{p \in M} L_pM. \tag{A.4.6}$$

To make this into a principal bundle we need to first equip LM with an atlas inherited from M such that $\pi : LM \to M$ is continuous, where π is defined through

$$LM \ni (e_1, \ldots, e_d) \mapsto \pi(e_1, \ldots, e_d) = p \tag{A.4.7}$$

since there is a unique $p \in M$ such that $(e_1, \ldots, e_d) \in L_pM$. This makes $LM \xrightarrow{\pi} M$ into a smooth bundle.

Next we need to equip LM with a right action of some Lie group G. The natural choice for this is $G = GL(d, \mathbb{R})$ since from the construction of LM we already have a copy of $GL(d, \mathbb{R})$ at each point in M. We define this action by

$$(e_1, \ldots, e_d) \triangleleft g := (g^{\mu}{}_1 e_\mu, \ldots, g^{\mu}{}_d e_\mu) \tag{A.4.8}$$

where we write the components of $g \in GL(d, \mathbb{R})$ as $g^m{}_n$.

Is this a free action? Yes. This is easily seen by considering that

$$(e_1, \ldots, e_d) \triangleleft g = (g^\mu{}_1 e_\mu, \ldots, g^\mu{}_d e_\mu) = (e_1, \ldots, e_d) \tag{A.4.9}$$

requires that $g^m{}_n = \delta^m_n$ which corresponds to the identity in $GL(d, \mathbb{R})$.

Left is now only to check whether (LM, π, M) is isomorphic as a bundle to the bundle with the quotient space as base space: $(LM, \pi', LM/GL(d, \mathbb{R}))$. To do this we start by picking a frame $e \in LM$. (I.e. $e \in L_pM$ for some $p \in M$.) Since the action of $GL(d, \mathbb{R})$ on LM is free we have that the orbit of the frame e is the whole of $GL(d, \mathbb{R})$. Hence the quotient space $LM/GL(d, \mathbb{R})$ consists of a single frame, e.g. equivalence class, for each point in M. Associating that equivalence class with that point we get that $M \simeq LM/GL(d, \mathbb{R})$. In the diagram below we now choose $u = id_{LM}$, and thus (LM, π, M) is isomorphic to $(LM, \pi', LM/GL(d, \mathbb{R}))$, the frame bundle is a principal bundle, and we're done.

$$\begin{array}{ccc} LM & \xrightarrow{u} & LM \\ \downarrow{\scriptstyle \pi} & & \downarrow{\scriptstyle \pi'} \\ M & \xrightarrow{v} & LM/GL(d, \mathbb{R}) \end{array}$$

Definition A.4.14 (Extension and restriction of principal bundles) Let $P \xrightarrow{\pi} M$ and $P' \xrightarrow{\pi'} M$ be principal bundles under G and a closed subgroup $H \leq G$. If there exists a bundle map (u, v) such that the following holds

(i) $v \circ \pi = \pi' \circ u$ (bundle map)

(ii) $u(p \triangleleft h) = u(p) \triangleleft' h$ for all $h \in H \leq G$ and $p \in P$.

Then $P \xrightarrow{\pi} M$ is called a *G-extension* of the H-principal bundle $P' \xrightarrow{\pi'} M$ and $P' \xrightarrow{\pi'} M$ is called the *H-restriction* of the G-principal bundle $P \xrightarrow{\pi} M$. The criteria above can be represented diagrammatically as the following diagram commuting when we only let G act using elements in H.

$$\begin{array}{ccc} P & \xrightarrow{u} & P' \\ {\scriptstyle \triangleleft G}\uparrow & & {\scriptstyle \triangleleft' H}\uparrow \\ P & \xrightarrow{u} & P' \\ \downarrow{\scriptstyle \pi} & & \downarrow{\scriptstyle \pi'} \\ M & \xrightarrow{v} & M \end{array}$$

Theorem A.4.15 (Extension and restriction) *Let $H \leq G$ be a closed subgroup of the Lie group G. Then the following holds:*

(i) Any H-principal bundle can be extended into a G-principal bundle

(ii) A G-principal bundle can be restricted to a H-principal bundle iff

- *the bundle $P/H \xrightarrow{\pi'} M$ has a global section.* □

A.4.2 Associated bundle

We now give the definition on how to construct a bundle associated to a principal G-bundle.

Definition A.4.16 (Associated bundle) Given a G-principal bundle

$$P \xleftarrow[\triangleleft G]{} P \xrightarrow{\pi} M$$

and a smooth manifold F on which a left G-action $\triangleright : G \times F \to F$ is defined we define the *associated bundle* $P_F \xrightarrow{\pi_F} M$ (associated to the principal bundle) by several steps:

i. First define an equivalence relation on the product space $P \times F$ by

$$(p, f) \sim_G (p', f') \quad \text{iff} \quad \exists g \in G : p' = p \triangleleft g \text{ and } f' = g^{-1} \triangleright f. \tag{A.4.10}$$

We then define the total space to be the quotient space $P_F = (P \times F)/\sim_G$.

ii. Define the projection π_F by $\pi_F([p, f]) = \pi(p)$.

Then $P_f \xrightarrow{\pi_F} M$ is a fibre bundle with principal fibre F associated to the principal G-bundle.

Remark A.4.17 It's easy to check that the definition of the projection π_F is well defined: $\pi_F([p, f]) = \pi_F([p' \triangleright g, g^{-1} \triangleleft f]) = \pi(p' \triangleright g) = \pi(p)$ since the right action of G preserves the fibres.

Proof (Associated bundle is a fibre bundle) To show that $P_F \xrightarrow{\pi_F}$ is a fibre bundle with typical fibre F we need to show

$$\pi_F^{-1}(p) = F_p \simeq F \tag{A.4.11}$$

for all points $p \in M$.

Since $P \xrightarrow{\pi} M$ is a (principal) fibre bundle, it has a typical fibre which we call K. We will, for clarity, denote a local element $e \in P$ as $e = (p, \tilde{e}) \in \{p\} \times K_p$, the p is normally left implicit. We can do this since every fibre bundle over a smooth manifold is locally trivial.

We begin by constructing an auxiliary function $\phi_{e_0} : P_F \to P$ such that $\pi_F = \pi \circ \phi_{e_0}$. ϕ_{e_0} is defined by that it maps all elements of an equivalence class $[e, f] = [(p, \tilde{e}), f] \in P_F$ to a specific element e_0 in the total space P such that $e_0 = (p, \tilde{e}_0)$ with $\tilde{e}_0 \in K_p$. That is $\phi_{e_0}([(p, \tilde{e}), f]) = (p, \tilde{e}_0) \in P$. This works because $\tilde{e} \in K_p$ and the action of G is free and preserves the fibre of the principal bundle $P \xrightarrow{\pi} M$, i.e. given $\tilde{e} \in K_p$ we have that $\tilde{e} \triangleright g \in K_p$ for all $g \in G$.

This ϕ_{e_0} satisfies $\pi_F = \pi \circ \phi_{e_0}$ since $\pi_F([(p, \tilde{e}), f]) = \pi((p, \tilde{e})) = p$ by definition and $\pi(\phi_{e_0}([(p, \tilde{e}), f])) = \pi((p, \tilde{e}_0)) = p$.

Hence we get that $\pi_F^{-1} = (\pi \circ \phi_{e_0})^{-1} = \phi_{e_0}^{-1} \circ \pi^{-1}$. Letting this operate on $p \in M$ we get that $\pi^{-1}(p) = \{p\} \times K_p$; now $\phi_{e_0}^{-1}(\{p\} \times K_p)$ is the set

$$\begin{aligned} \phi_{e_0}^{-1}(\{p\} \times K_p) = & \\ &= \{[(p, \tilde{e}), f] \in P_F \mid \phi_{e_0}([(p, \tilde{e}), f]) \in \{p\} \times K_p\} \\ &= \{[(p, \tilde{e}), f] \mid \tilde{e} \in K_p, f \in F\} \end{aligned} \tag{A.4.12}$$

since $\phi_{e_0}([e,f]) = \phi_{e_0}([(p,\tilde{e}),f]) = (p,\tilde{e}_0)$. Now, since the action of G on the fibre K_p is free and preserves K_p we can choose an $e' = (p,\tilde{e}') \in P$ and there will always exists a $g \in G$ such that $[e,f] = [(p,\tilde{e}),f] \sim_G [(p,\tilde{e}'),f'] = [e',f']$ for some $f' \in F$. Hence

$$\{[(p,\tilde{e}),f] \mid \tilde{e} \in K_p, f \in F\} \simeq \{[(p,\tilde{e}'),f'] \mid \tilde{e}' \in K_p \text{ is fixed and } f' \in F\} \simeq F. \tag{A.4.13}$$

■

Hence we have shown $\pi_F^{-1}(p) = F_p \simeq F$ for all $p \in M$ and therefore is $P_F \xrightarrow{\pi_F} M$ a fibre bundle.

Definition A.4.18 (Associated bundle map) Given two associated bundles, $P_F \xrightarrow{\pi_F} M$, $P'_F \xrightarrow{\pi'_F} M'$, that share the same fibre but are associated to different principal bundles, a principal bundle map is a pair $(\tilde{u},\tilde{v})$ such that they are constructed from a principal bundle maps (u,v) (see Definition A.4.12) between the two underlying principal bundles as

$$\begin{aligned} \tilde{u}([e,f])) &= [u(e),f] \\ \tilde{v}(m) &= v(m). \end{aligned} \tag{A.4.14}$$

I.e. the following diagrams commutes

$$\begin{array}{ccc} P_F & \xrightarrow{\tilde{u}} & P'_F \\ \downarrow{\scriptstyle \pi_F} & & \downarrow{\scriptstyle \pi'_F} \\ M & \xrightarrow{\tilde{v}} & M' \end{array} \quad \text{and} \quad \begin{array}{ccc} E & \xrightarrow{u} & E' \\ {\scriptstyle \triangleleft G}\uparrow & & \uparrow{\scriptstyle \triangleleft' G'} \\ E & \xrightarrow{u} & E' \\ \downarrow{\scriptstyle \pi} & & \downarrow{\scriptstyle \pi} \\ M & \xrightarrow{v} & M' \end{array}$$

A.4.3 Vector bundles and associated vector bundles

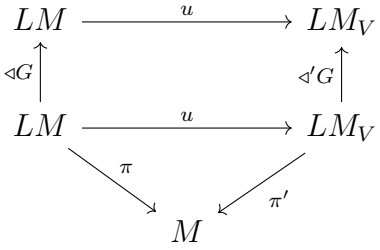
This is one of the most central bundle types to describe convolutional neural networks since the feature maps can be thought of as "vector fields" on the data manifold, usually $\mathbb{Z}^2$.

Definition A.4.19 (Vector bundle) A *vector bundle* is a fibre bundle (E,π,M) with canonical fibre V such that the fibre V_p at a point $p \in M$ is isomorphic to V, where V is a vector space. Usually this is $\mathbb{K}^m$, and, in our case, we will use $\mathbb{R}^m$.

Definition A.4.20 (Tangent bundle) For a smooth d-dimensional manifold M we define the tangent bundle as the fibre bundle which has at $p \in M$ the tangent space at p as fibre. Since, for a d-dimensional manifold, $T_pM \simeq \mathbb{R}^d$ we have that the tangent bundle is a vector bundle with $\mathbb{R}^d$ as typical fibre.

Since vector spaces can be made into smooth manifolds we will construct associated vector bundles, with a real n-dimensional vector space V as a fibre, to the (principal) frame bundle to a d-dimensional smooth manifold M. This will give our vectors their

geometric transformation behaviour. Diagrammatically we construct the commuting diagram



where $LM_V = (LM \times V)/\sim_G$ consisting of equivalence classes $[e, v]$, and $G = GL(d, \mathbb{R})$. When $V = \mathbb{R}^d \simeq T_pM$ this can intuitively be seen as a local frame e and a vector v expressed in that frame.

The equivalence relation is

$$[e, v] \sim_G [e, v] \triangleleft g = [e \triangleleft g, g^{-1} \triangleright v] \tag{A.4.15}$$

where we in the example of frame bundles defined the G action as

$$e \triangleleft g = (e_1, \ldots, e_d) \triangleleft g = (g^{\mu}{}_1 e_\mu, \ldots, g^{\mu}{}_d e_\mu). \tag{A.4.16}$$

We define the left action of G on V component wise as

$$(g \triangleright v)^\nu = \rho(g)^\nu{}_\mu v^\mu \tag{A.4.17}$$

where we have used the representation $\rho : G \to GL(n.\mathbb{R})$ that takes an element of $G = GL(d, \mathbb{R})$ and maps it to a $n \times n$ matrix. This representation must satisfy the equivariance condition from the commuting diagram

$$u(x \triangleleft g) = u(x) \triangleleft' \rho(g). \tag{A.4.18}$$

We can now see that this associated vector bundle $LM_V \xrightarrow{\pi_V} M$ with canonical fibre V is isomorphic to the vector bundle $E \xrightarrow{\pi} M$ with canonical fibre V. To do this we need an invertible map $u : LM_V \to E$ such that the following diagram commutes

$$\begin{array}{ccc} LM_V & \xrightarrow{u} & E \\ \downarrow{\scriptstyle \pi_V} & & \downarrow{\scriptstyle \pi} \\ M & \xrightarrow{id_V} & M \end{array}$$

This is satisfied with the map

$$\begin{aligned} u : LM_V &\to E \\ [e, v] &\mapsto v^\mu e'_\mu \in V. \end{aligned} \tag{A.4.19}$$

Here we have different cases: when $n = \dim V < d = \dim M$, when $n = d$, and when $n > d$.

$n = d$:

This is the simplest case where we can choose the basis vectors e'_μ of V as

$$e'_\mu = e_\mu \tag{A.4.20}$$

where e_μ is the frame of T_pM used in the equivalence class. In this we can also use the identity map on $GL(d, \mathbb{R})$ as the representation.

With this defined we see that expressing a vector at $p \in M$ in a local frame now is invariant under the G-action:

$$u([e, v]) = v^\mu e_\mu \xrightarrow{\triangleleft g} ((g^{-1})^\mu{}_\nu \tilde{v}^\nu) g^\alpha{}_\mu \tilde{e}_\alpha = \delta^\alpha_\nu v^\nu e_\alpha = \tilde{v}^\nu \tilde{e}_\nu = u([\tilde{e}, \tilde{v}]) \tag{A.4.21}$$

which is what we would expect of a vector.

$n < d$:

In this case the frame e of the tangent space has more basis vectors than V has dimensions. We thus construct the map

$$\begin{aligned} u : LM_V &\to E \\ [e, v] &\mapsto v^\mu e'_\mu \end{aligned} \tag{A.4.22}$$

where

$$e'_\mu = \alpha(e_\mu) \text{ for } \mu = 1, \ldots, n < d \text{ and } \alpha : \mathbb{R}^d \to \mathbb{R}^n \text{ by } \alpha(x^1, \ldots, x^d) = (x^1, \ldots, x^n). \tag{A.4.23}$$

I.e. we only keep the first n vectors of the frame e and project these onto $\mathbb{R}^n$.

Using a representation ρ of $G = GL(d, \mathbb{R})$ that extracts the first $n \times n$ block results in problems for general $g \in GL(d, \mathbb{R})$. For example, with $g, h \in GL(d, \mathbb{R})$ and $g', h' \in GL(n, \mathbb{R})$:

$$g = \left(\begin{array}{c|c} g' & g_a \\ \hline g_b & g'' \end{array}\right) \text{ and } h = \left(\begin{array}{c|c} h' & h_a \\ \hline h_b & h'' \end{array}\right)$$

we get

$$\rho(gh) = g'h' + g_a h_b \text{ but } \rho(g)\rho(h) = g'h'. \tag{A.4.24}$$

Hence this only works if we restrict the group of the principal bundle to those $g \in GL(d, \mathbb{R})$ on the form

$$g = \left(\begin{array}{c|c} g' & 0 \\ \hline 0 & 1 \end{array}\right).$$

Using this restriction we can check whether the map u is well defined.

$$\begin{aligned} u([e,v]) &= u([e \triangleleft g, g^{-1} \triangleright v]) \\ &= \sum_{\nu=1}^{n}\sum_{\sigma=1}^{n} \alpha(e \triangleleft g)_\nu \rho(g^{-1})^\nu{}_\sigma v^\sigma = \{\alpha(e \triangleleft g)_\nu = \rho(g)^\mu{}_\nu e'_\mu\} \\ &= \sum_{\mu=1}^{n}\sum_{\nu=1}^{n}\sum_{\sigma=1}^{n} \rho(g)^\mu{}_\nu e'_\mu \rho(g^{-1})^\nu{}_\sigma v^\sigma \\ &= \sum_{\mu=1}^{n}\sum_{\sigma=1}^{n} \delta^\mu_\sigma e'_\mu v^\sigma = v^\sigma e'_\sigma \\ &= u([e,v]). \end{aligned} \tag{A.4.25}$$

Thus, to make an associated vector bundle isomorphic to a vector bundle which typical fibre has lower dimensionality than the base manifold we can restrict the structure group of T_pM.

***n > d*:**

In this case the tangent vectors in the frame e of T_pM are to few to span V. Thus we define the map u as

$$\begin{aligned} u : LM_V &\to E \\ [e,v] &\mapsto v^\mu e'_\mu. \end{aligned} \tag{A.4.26}$$

The basis e'_μ of V is

$$e'_\mu = \begin{cases} \beta(e_\mu), & \text{for } \mu = 1,\dots,d \\ \xi_i, & \text{for } i = d+1,\dots,n \end{cases} \tag{A.4.27}$$

where $\beta : \mathbb{R}^d \to \mathbb{R}^n$ by $\beta(x^1,\dots,x^d) = (x^1,\dots,x^d,0,\dots,0)$ and ξ_i is the i:th standard basis vector of $\mathbb{R}^n$.

The representation of $GL(d,\mathbb{R})$ here is simply

$$\rho(g) = \left(\begin{array}{c|c} g & 0 \\ \hline 0 & 1 \end{array}\right).$$

Now we just have to check that the map u is well defined under the group action.

$$\begin{aligned} u([e,v]) &= u([e \triangleleft g, g^{-1} \triangleright v]) \\ &= \sum_{\sigma=1}^{n}\sum_{\nu=1}^{n} \beta(e \triangleleft g)_\sigma \rho(g^{-1})^\sigma{}_\nu v^\nu = \{\beta(e \triangleleft g)_\sigma = \rho(g)^\mu{}_\sigma e'_\mu\} \\ &= \sum_{\mu=1}^{n}\sum_{\sigma=1}^{n}\sum_{\nu=1}^{n} \rho(g)^\mu{}_\sigma e'_\mu \rho(g^{-1})^\sigma{}_\nu v^\nu \\ &= \sum_{\mu=1}^{n}\sum_{\nu=1}^{n} \delta^\mu_\nu e'_\mu v^\nu = v^\nu e'_\nu \\ &= u([e,v]). \end{aligned} \tag{A.4.28}$$

By these constructions we now see that the associated vector bundle $LM_V \xrightarrow{\pi_V} M$ can be made ismorphic to the vector bundle $E \xrightarrow{\pi} M$ that has the typical fibre V.

To summarise, the local vector v is invariant under g action but its components v^i transform equivariantly.

Since the frame bundle captures the local $GL(d, \mathbb{R})$ symmetry of the tangent space to M we associate bundles to the frame bundle to make sure that they contain the information of this local geometrical symmetry.

Example A.4.21 (Associated tangent bundle) One example of an associated vector bundle is the *associated tangent bundle* to the frame bundle where we let the smooth manifold $F = \mathbb{R}^d \simeq T_pM$ for a d-dimensional smooth manifold M.

The total space is then denoted $LM_{\mathbb{R}^d} = (LM \times \mathbb{R}^d)/GL(d, \mathbb{R})$, and it otherwise technically developed as stated in the text above.

This fibre bundle is isomorphic to the tangent bundle as commented on above for general n-dimensional vector spaces V.

Definition A.4.22 (Tensor bundle) The (p, q)-tensor bundle is a vector bundle where the canonical fibre is the vector space of multilinear maps

$$T : \underbrace{V \otimes \cdots \otimes V}_{p \text{ copies}} \otimes \underbrace{V^* \otimes \cdots \otimes V^*}_{q \text{ copies}} \to \mathbb{R} \tag{A.4.29}$$

where V is a vector space and V^* is its dual.

Normally the vector space V is taken to be the tangent space to the base manifold M, but one can generalise this to

$$T : V_1 \otimes \cdots \otimes V_p \otimes U_1^* \otimes \cdots \otimes U_q^* \to \mathbb{R} \tag{A.4.30}$$

where the V_i:s and U_i:s are potentially different, but this is unusual.

A.5 Integration on manifolds

In this section we introduce the structure necessary to perform integrals on manifolds. We start with introducing scalar valued differential forms on flat spaces, i.e. $\mathbb{R}^d$, and manifolds of dimension d, and their integration.

A.5.1 Differential forms

Definition A.5.1 (Alternating map) Let V be a vector space, then a multilinear map $\eta : \underbrace{V \otimes \cdots \otimes V}_{k \text{ copies}} \to \mathbb{R}$ is called *alternating* if

$$\eta(v_1, \cdots, v_i, \cdots, v_j, \cdots, v_k) = 0 \tag{A.5.1}$$

for $v_i = v_j$. The space of alternating maps $\eta : V^{\otimes_k} \to \mathbb{R}$ is denoted $\mathrm{Alt}^k(V)$.

We will usually work with the vector space $V \simeq \mathbb{R}^d$.

Definition A.5.2 (Smooth differential form) A *differential k-form* on a subset $U \subseteq \mathbb{R}^m$ is a smooth map

$$\omega : U \to \mathrm{Alt}^k(\mathbb{R}^d). \tag{A.5.2}$$

The space of differential k-forms on U is denoted $\Omega^k(U)$.

Note that intuitively a differential form ω is a parametrisation of alternating since for $x \in U$ we have $\omega(x) \in \mathrm{Alt}^k(\mathbb{R}^d)$. For ease of notation we will denote the differential form ω evaluated at the base point x as $\omega_x \in \mathrm{Alt}^k(\mathbb{R}^d)$.

Remark A.5.3 Note that $\mathrm{Alt}^0(\mathbb{R}^d) = \mathbb{R}$ and as such we get that $\Omega^0(U)$ is the space of smooth functions from U to $\mathbb{R}$.

Definition A.5.4 (Directional derivative) Let $\omega : U \to \mathrm{Alt}^k(\mathbb{R}^d)$ be a differential form and e_i the i:th basis vector of $\mathbb{R}^d$ then we define the directional derivative of a differential form $D\omega$ at the base point x and in the direction e_i as

$$(D\omega)_x(e_i) = D_x\omega(e_i) = \frac{d}{dt}\omega_{x+te_i}\bigg|_{t=0} = \frac{\partial \omega_x}{\partial x^i}. \tag{A.5.3}$$

Remark A.5.5 In that case that $\phi \in \Omega^0(U)$ we have that $\phi_x \in \mathbb{R}$ and that

$$(D\phi)_x(e_i) = \frac{\partial \phi_x}{\partial x^i} \tag{A.5.4}$$

is simply the partial derivative of ϕ in the direction of e_i. For a general vector $u = u^i e_i \in U$ we get

$$(D\phi)_x(u) = \frac{\partial \phi_x}{\partial x^i} u^i. \tag{A.5.5}$$

That was the case for $\phi : U \to \mathbb{R}$ and we can extend this to the case for $\phi : U \subseteq \mathbb{R}^m \to \mathbb{R}^p$ by letting

$$D_x\phi(e_i) = (D_x\phi^1(e_i), \ldots, D_x\phi^p(e_i)) \tag{A.5.6}$$

where ϕ^i is the i:th component of ϕ.

Definition A.5.6 (Exterior derivative) The exterior differential d is a linear operator

$$d : \Omega^k(U) \to \Omega^{k+1}(U) \tag{A.5.7}$$

defined at $x \in U$ by the action on $\omega \in \Omega^k(U)$

$$(d\omega)_x(\xi_1, \ldots, \xi_{k+1}) = \sum_{l=1}^{k+1} (-1)^{l-1} [D_x\omega(\xi_l)](\xi_1, \ldots, \hat{\xi}_l, \ldots, \xi_{k+1}) \tag{A.5.8}$$

where $\hat{\xi}_l$ means that ξ_l is removed.

Definition A.5.7 (Pullback) Let $\phi : U \subseteq \mathbb{R}^m \to V \subseteq \mathbb{R}^p$ be a smooth map. Then for $\Omega^k(U) = \{\omega \mid \omega : U \to \mathrm{Alt}^k(\mathbb{R}^m)\}$ and $\Omega^k(V) = \{\omega \mid \omega : V \to \mathrm{Alt}^k(\mathbb{R}^p)\}$ we define the pullback

$$\phi^* : \Omega^k(V) \to \Omega^k(U) \tag{A.5.9}$$

such that for $\omega \in \Omega^k(V)$, $x \in U$ and $\xi_1, \ldots, \xi_k \in \mathbb{R}^m$

$$(\phi^*\omega)_x(\xi_1, \ldots, \xi_k) = \omega_{\phi(x)}(D_x\phi(\xi_1), \ldots, D_x\phi(\xi_k)). \tag{A.5.10}$$

Remark A.5.8 Note that since $\phi : \mathbb{R}^m \to \mathbb{R}^p$ we have for $\xi_i \in \mathbb{R}^m$ that $D_x\phi(\xi_i) \in \mathbb{R}^p$.

Now, let M be a smooth manifold and (U, ϕ) a chart containing the point $p \in M$.

Definition A.5.9 (Differential form on manifold) A differential k-form ω on a smooth manifold M is a family $\omega = \{\omega_p\}_{p\in M}$ of alternating k-forms on the tangent space T_pM such that

$$\omega_p \in \mathrm{Alt}^k(T_pM). \tag{A.5.11}$$

Since (U, ϕ) is a chart we have $\phi : U \to V \subseteq \mathbb{R}^d$ as a homeomorphism. Therefore we can construct the pullback $(\phi^{-1})^* : \Omega^k(U) \to \Omega^k(V)$ which takes forms on the manifold and returns forms on $\mathbb{R}^d$. ϕ^{-1} is called a local parametrisation of the manifold around the point $p \in M$.

Note that for $x \in V = \phi(U) \subseteq \mathbb{R}^d$

$$D_x(\phi^{-1}) : \mathbb{R}^d \to T_{\phi^{-1}(x)}M \simeq \mathbb{R}^d \tag{A.5.12}$$

is an isomorphism for a manifold M of dimension d. This induces an isomorphism

$$\mathrm{Alt}^k(D_x(\phi^{-1})) : \mathrm{Alt}^k(T_{(\phi^{-1})(x)}M) \to \mathrm{Alt}^k(\mathbb{R}^d). \tag{A.5.13}$$

Let $V = \phi(U)$ be a subset of $\mathbb{R}^d$, we then define the map

$$(\phi^{-1})^*(\omega) : V \to \mathrm{Alt}^k(\mathbb{R}^d) \tag{A.5.14}$$

by specifying its value at $x \in V$ as

$$(\phi^{-1})^*(\omega)_x = \mathrm{Alt}^k(D_x(\phi^{-1}))(\omega_{\phi^{-1}(x)}) \tag{A.5.15}$$

where ω is a differential form on M. Since ω is a differential k-form on M we have for $p = \phi^{-1}(x)$ that $\omega_p = \omega_{\phi^{-1}(x)} \in \mathrm{Alt}^k(T_pM) = \mathrm{Alt}^k(T_{\phi^{-1}(x)}M)$. Therefore $\mathrm{Alt}^k(D_x(\phi^{-1}))(\omega_{\phi^{-1}(x)}) \in \mathrm{Alt}^k(\mathbb{R}^d)$. Hence we can view $(\phi^{-1})^*(\omega)$ as an element of $\Omega^k(V)$ since $(\phi^{-1})^*(\omega)_x \in \mathrm{Alt}^k(\mathbb{R}^d)$.

Definition A.5.10 (Smooth differential form on manifold) A k-form $\omega = \{\omega_p\}_{p\in M}$ is smooth if $(\phi^{-1})^*(\omega)$ is smooth for all local parametrisations ϕ^{-1}. The space of such forms is denoted $\Omega^k(M)$.

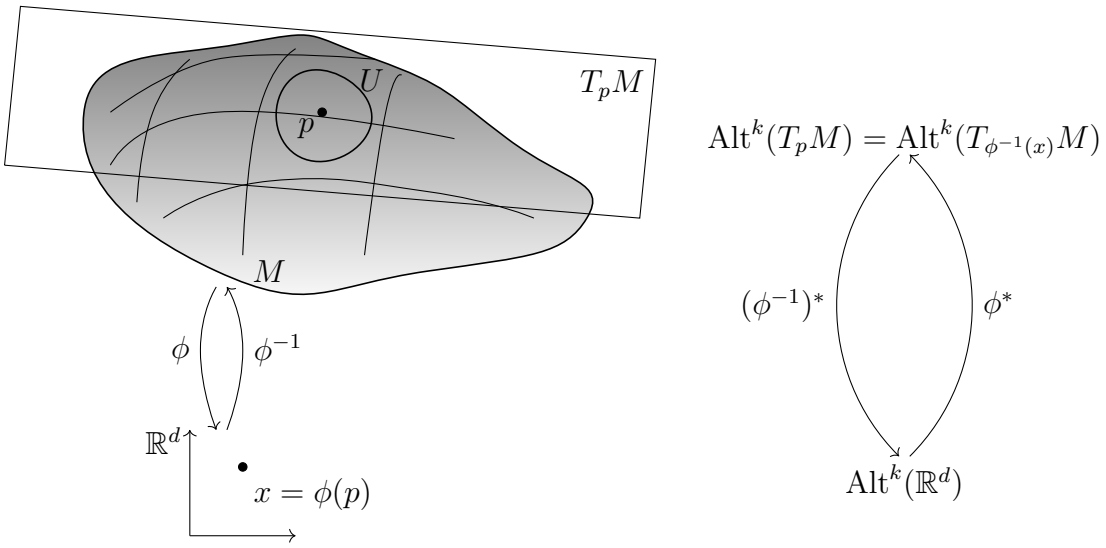
A.5.2 Bases

To be able to talk about bases we first need to introduce the exterior product, and for that we need one introductory concept.

Definition A.5.11 ((p, q)-shuffles) A (p, q)-shuffle is a permutation $\sigma \in S(p+q)$ such that

$$\begin{aligned} \sigma(1) < \sigma(2) < \cdots < \sigma(p) \\ \sigma(p+1) < \cdots < \sigma(p+q). \end{aligned} \tag{A.5.16}$$

These σ are called "ordered" permutations, and the set of all such σ is denoted $S(p, q)$.



Definition A.5.12 (Exterior product) Let $\omega_1 \in \mathrm{Alt}^p(V)$ and $\omega_2 \in \mathrm{Alt}^q(V)$ then for vectors $\xi_1, \ldots, \xi_{p+q} \in V$ we define

$$(\omega_1 \wedge \omega_2)(\xi_1, \ldots, \xi_{p+q}) = \sum_{\sigma \in S(p,q)} \mathrm{sign}(\sigma)\omega_1(\xi_{\sigma(1)}, \ldots, \xi_{\sigma(p)})\omega_2(\xi_{\sigma(p+1)}, \ldots, \xi_{\sigma(p+q)}) \tag{A.5.17}$$

where $\mathrm{sign}(\sigma)$ is the sign of the permutation $= (-1)^{\#\text{number of transpositions in } \sigma}$.

Now we present some useful lemmas regarding the exterior product.

Lemma A.5.13 *If $\omega_1 \in \mathrm{Alt}^k(V)$ and $\omega_2 \in \mathrm{Alt}^p(V)$ then $\omega_1 \wedge \omega_2 \in \mathrm{Alt}^{k+p}(V)$.* □

Lemma A.5.14 (Properties of the exterior product) *These are some properties of the exterior product.*

- $(\omega_1 + \omega_1') \wedge \omega_2 = \omega_1 \wedge \omega_2 + \omega_1' \wedge \omega_2$
- $(\lambda\omega_1) \wedge \omega_2 = \lambda(\omega_1 \wedge \omega_2) = \omega_1 \wedge (\lambda\omega_2)$
- $\omega_1 \wedge (\omega_2 + \omega_2') = \omega_1 \wedge \omega_2 + \omega_1 \wedge \omega_2'$
- $\omega_1 \wedge (\omega_2 \wedge \omega_3) = (\omega_1 \wedge \omega_2) \wedge \omega_3$ □

Lemma A.5.15 *For $\omega_1 \in \mathrm{Alt}^p(V)$ and $\omega_2 \in \mathrm{Alt}^q(V)$ we have $\omega_1 \wedge \omega_2 = (-1)^{pq}\omega_2 \wedge \omega_1$.* □

Now we can talk about bases. Let $\{e_i\}_{i \in I}$ be a basis for $\mathbb{R}^d$, and $\{\epsilon^j\}_{j \in J}$ be a basis for the dual space, $(\mathbb{R}^d)^* = \mathrm{Alt}^1(\mathbb{R}^d)$, such that

$$\epsilon^i(e_j) = \delta^i_j = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases}$$

Since $\epsilon^i \in \mathrm{Alt}^1(\mathbb{R}^d)$ we for $v^i e_i = v \in \mathbb{R}^d$ get that $\epsilon^i(v) = \epsilon^i(v^j e_j) = \{\epsilon^i \text{ is linear}\} = v^j \epsilon^i(e_j) = v^j \delta^i_j = v^i$, i.e. ϵ^i "picks out" the i:th coordinate of v.

Example A.5.16 Since $\Omega^0(U \subseteq \mathbb{R}^k) \simeq \mathbb{R}$ we can define $x^i : U \to \mathbb{R} \simeq \Omega^0(U)$ to be the i:th projection. Then $dx^i \in \Omega^1(U)$ is the constant map

$$dx^i : x \mapsto \epsilon^i \tag{A.5.18}$$

and we can consider $dx^i = \epsilon^i$. This is since for $v \in \mathbb{R}^k$

$$(dx^i)_x(v) = D_x x^i(v) = \frac{d}{dt} x^i(x + tv^j e_j)\Big|_{t=0} = v^j \frac{\partial x^i}{\partial x^j} = \epsilon^j(v)\delta^i_j = \epsilon^i(v). \tag{A.5.19}$$

The next element we introduce for ease of notation: is multiindeces.

Definition A.5.17 (Multiindex) An (ordered) multiindex I is a tuple $I = (i_1, \ldots, i_p)$ that satisfies

$$1 \leq i_1 < i_2 < \cdots < i_p \leq n. \tag{A.5.20}$$

We also define, for multiindeces, the notation

$$\epsilon^I = \epsilon^{i_1} \wedge \epsilon^{i_2} \wedge \cdots \wedge \epsilon^{i_p} \tag{A.5.21}$$

and

$$e_J = (e_{j_1}, e_{j_2}, \ldots, e_{j_p}). \tag{A.5.22}$$

By this notation we have

$$\epsilon^I(e_J) = (\epsilon^{i_1} \wedge \cdots \wedge \epsilon^{i_p})(e_{j_1}, \ldots, e_{j_p}) = \epsilon^{i_1}(e_{j_1})\epsilon^{i_2}(e_{j_2}) \cdots \epsilon^{i_p}(e_{j_p}) = \delta^{i_1}_{j_1} \cdots \delta^{i_p}_{j_p} = \delta^I_J \tag{A.5.23}$$

The arbitrary limit of n for the components of the multiindex comes from the fact that if $\dim(V) = n$ then any set of vectors $\{v_1, \ldots, v_k\}$ containing more than n vectors will be linearly dependent and hence $\epsilon^I(v_1, \ldots, v_k) = 0$.

Theorem A.5.18 (Basis for $\mathbf{Alt^k(V)}$) *The space* $\mathrm{Alt}^k(V)$ *has*

$$\epsilon^I, \quad I = (i_1, \ldots, i_k) \tag{A.5.24}$$

as a basis. □

Hence, since $\omega \in \Omega^k(V \subseteq \mathbb{R}^d)$ has $\omega_x \in \mathrm{Alt}^k(\mathbb{R}^d)$ we can write

$$\omega_x = f_I(x)\epsilon^I \quad \text{(using summation convention)} \tag{A.5.25}$$

where the components f_I are functions $f_I : V \to \mathbb{R}$.

We can now look again at Example A.5.16 and conclude that we can use dx^i:s to build a basis for $\mathrm{Alt}^k(V)$ and as such we write $\omega_x = f_I(x)dx^I$. This is the what we are going to use going forward. To perform integrals we need to transfer differential forms from the manifold to differential forms on $\mathbb{R}^d$ using the pullback $(\phi^{-1})^*$ above. To do this we have the following lemma.

Lemma A.5.19 (Properties of the pullback) *Let* ϕ^* *be a pullback as defined in definition A.5.7 and* $\omega \in \Omega^k(V)$ *then*

- $\phi^*(f\omega)_x = (f \circ \phi)(x)(\phi^*\omega)_x$

- *Let $\eta,\ \zeta$ be differential forms such that $\eta\wedge\zeta \in \Omega^k(V)$ then $\phi^*(\eta\wedge\zeta) = \phi^*\eta\wedge\phi^*\zeta$*
- $d(\phi^*\omega) = \phi^* d\omega$
- *Let $\eta,\ \zeta \in \Omega^k(V)$ then $\phi^*(\eta+\zeta) = \phi^*\eta + \phi^*\zeta$* □

Example A.5.20 Let ϵ^i be a basis vector of $\mathrm{Alt}^1(V)$ and $\phi : U \to V$ giving rise to a pullback $\phi^* : \Omega^1(V) \to \Omega^1(U)$. Then we for $u \in U$ get

$$\begin{aligned}(\phi^*\epsilon^i)_x(u) = \epsilon^i_{\phi(x)}(D_x\phi(u)) = \epsilon^i\left(\frac{\partial\phi^k(x)}{\partial x^j}u^j e_k\right) = \frac{\partial\phi^k(x)}{\partial x^j}u^j\epsilon^i(e_k) = \\ = \frac{\partial\phi^i(x)}{\partial x^j}u^j = \frac{\partial\phi^i(x)}{\partial x^j}(dx^j)_x(u) = (d\phi^i)_x(u)\end{aligned} \tag{A.5.26}$$

and hence $\phi^*\epsilon^i = d\phi^i$. With, from example A.5.16, that $dx^i = \epsilon^i$ we see $\phi^*\epsilon^i = \phi^* dx^i = d(\phi^* x^i) = d\phi^i$. And since $\phi^* x^i = (x\circ\phi)^i = \phi^i$ is the i:th component of ϕ this makes sense: the pullback of the coordinate x^i gives the coordinates in terms of the pullback ϕ^i.

Let now $\omega \in \Omega^k(M)$ be a differential form on the d-dimensional manifold M. Then at a point $p \in M$ we can write

$$\omega_p = f_I(p)\epsilon^I \tag{A.5.27}$$

where $f_I(p)$ is, for now, a section of the bundle (E,π,M) with typical fibre $\mathbb{R}$ and ϵ^I is a basis for $\mathrm{Alt}^k(T_pM)$. With (U,ϕ) a local chart containing $p \in M$ and $g = \phi^{-1} : \mathbb{R}^d \to U$ a local parametrisation such that $p = g(x)$, we can pull ω back to being a form on $\mathbb{R}^d$ by

$$(g^*\omega)_x = (f_I\circ g)(x)g^*\epsilon^I = f_I(g(x))\frac{\partial g^i(x)}{\partial x^j}dx^j \tag{A.5.28}$$

where we are using that we locally can see the target of g as a surface embedded in $\mathbb{R}^m$.

A.5.3 Integration of differential forms

Before we can define integrals of differential forms on manifolds we need to talk about orientations.

Definition A.5.21 (Orientation and orientation form) A smooth n-dimensional manifold is called oriented if there exists $\omega \in \Omega^n(M)$ such that $\omega_p \neq 0$ for all $p \in M$. Such an ω is called an orientation form. Two orientation forms η,ζ are said to be equivalent iff there exists a $f \in \Omega^0(M) = \mathcal{C}^\infty(M,\mathbb{R})$ such that $f(p) > 0$ and $\eta_p = f(p)\zeta_p$ for all $p \in M$.

Definition A.5.22 (Oriented basis) Given a orientation form ω on M a basis $a_1,\ldots,a_d$ of T_pM is positively (negatively) oriented if $\omega(a_1,\ldots,a_d)$ is positive (negative).

Definition A.5.23 (Orientation preserving map) Let $\phi : M_1 \to M_2$ be a diffeomorphism between two d-dimensional manifolds. Let $\omega_{1,2}$ be orientation forms on $M_{1,2}$. The pullback

$$\phi^*\omega_2 \in \Omega^d(M_1) \tag{A.5.29}$$

is then an orientation form on M_1 and ϕ is called orientation preserving if $\phi^*\omega_2$ determines the same orientation as ω_1.

We will define integration on manifolds by pulling back to $\mathbb{R}^d$ where we know how to integrate. Thus we need an orientation on $\mathbb{R}^d$.

Definition A.5.24 (Standard orientation of $\mathbb{R}^d$) The orientation form

$$dx^1 \wedge dx^2 \wedge \cdots \wedge dx^d \tag{A.5.30}$$

is called the standard orientation of $\mathbb{R}^d$ and is the one that is used if nothing else is specified.

Definition A.5.25 (Oriented charts and atlas) Around any point p in a smooth oriented manifold M we can find a chart (U, ϕ) such that ϕ is a orientation preserving diffeomorphism when U has the orientation of M and $\phi(U) \subseteq \mathbb{R}^n$ has the orientation of $\mathbb{R}^d$. This is called an oriented chart. An atlas consisting of only oriented charts is called an oriented atlas.

Definition A.5.26 (Integration of forms on $\mathbb{R}^n$) Let $V \subset \mathbb{R}$ be a bounded subset, $\omega \in \Omega^d(\mathbb{R}^d)$, and $\{dx^1, \ldots, dx^d\}$ be a basis for $\Omega^1(\mathbb{R}^d)$. Then ω can be written

$$\omega_x = f(x)dx^1 \wedge \cdots \wedge dx^d \tag{A.5.31}$$

for some $f \in \mathcal{C}^\infty(\mathbb{R}^d, \mathbb{R})$. Then we define the integral

$$\int_V \omega = \int_V f dx^1 \wedge \cdots \wedge dx^d := \int_V f(x^1, \ldots, x^d) dx^1 dx^2 \cdots dx^d \tag{A.5.32}$$

where the final integral is the well known Lebesgue integral. This definition relies on that ω is expressed in the orientation form of $\mathbb{R}^d$.

In this type of notation the usual change of variables in integrals looks like this: Let $U, V \subseteq \mathbb{R}^d$ be open sets and $\phi : U \to V$ be a diffeomorphism such that $y = \phi(x) \in V$. Then the change of variables looks like

$$\int_{V=\phi(U)} f(y)dy = \int_U f(y(x))|D_x\phi|dx \tag{A.5.33}$$

where $|D_x\phi|$ is the (Jacobi) determinant of the matrix with components $\frac{\partial \phi^i}{\partial x^j}$.

Definition A.5.27 (Support of differential form) The support of a differential k-form $\omega \in \Omega^k(M)$ is the set

$$\operatorname{supp}_M \omega = \{p \in M \mid \omega_p \neq 0\}. \tag{A.5.34}$$

If this set is compact then ω is said to have compact support. The space of differential k-forms with compact support is denoted $\Omega_c^k(M)$.

Lemma A.5.28 *Let $\phi : U \to V$ be a diffeomorphism between open subsets of $\mathbb{R}^d$. Assume that $|D_x\phi| = \delta$ is a constant for all $x \in U$. Then for $\omega \in \Omega_c^n(V)$ one has*

$$\int_{V=\phi(U)} = \delta \int_U \phi^*\omega. \tag{A.5.35}$$

□

The formalism thus far has revolved around compactly supported differential forms. To extend this to differential forms defined on all of M we need a partition of unity.

Definition A.5.29 (Partition of unity) Let $\{U_i\}_{i\in I}$ be an open cover of the manifold M. A partition of unity subordinate to this cover is a collection of smooth functions $\{\rho_i : M \to \mathbb{R}\}$ that satisfies

i. $0 \leq \rho_i(p) \leq 1$ for all $i \in I$ and $p \in M$
ii. $\text{supp}_M\, \rho_i \subset U_i$
iii. for each point $p \in M$ there exists a neighbourhood U where only finitely many ρ_i are non-zero
iv. $\sum_{i\in I} \rho_i(p) = 1$ for all $p \in M$.

Theorem A.5.30 (Existence of partition of unity) *Let M be a smooth manifold and $\{U_i\}_{i\in I}$ be any open cover. Then there exists a partition of unity subordinate to this cover.* □

Proposition A.5.31 *For any smooth oriented d-dimensional manifold M there exists a unique linear map*

$$\int_M : \Omega_c^d(M) \to \mathbb{R} \tag{A.5.36}$$

that has the property that if $\omega \in \Omega^d(M)$ and $\text{supp}_M\, \omega \subset U$ where (U, ϕ) is a smooth oriented chart, then

$$\int_M \omega = \int_U \omega = \int_{\phi(U)} (\phi^{-1})^* \omega \tag{A.5.37}$$

□

A.5.4 Riemannian manifolds

For the case when we move onto the gauge symmetries of the manifold it will be relevant to have some theory on Riemannian manifolds.

Definition A.5.32 (Riemannian structure (Metric)) A Riemannian structure (in physics often called a metric) on a smooth manifold M is a family of inner products

$$\langle , \rangle_p \quad \text{on } T_pM \text{ for all } p \in M \tag{A.5.38}$$

satisfying the condition that for any local parametrisation $f : V \to M$ where $V \subseteq \mathbb{R}^d$ and any pair of vectors $v_1, v_2 \in \mathbb{R}^d$ the map

$$x \;\mapsto\; g(x) = \langle D_x f(v_1), D_x f(v_2) \rangle_{f(x)} \tag{A.5.39}$$

is a smooth function of x. The pair (M, g) is called a Riemannian manifold.

Note that since $D_x f : \mathbb{R}^d \to T_pM$ the metric is a map

$$\langle , \rangle_p \;:\; T_pM \times T_pM \to \mathbb{R}. \tag{A.5.40}$$

It is also linear and symmetric.

Remark A.5.33 Due to the nature of the directional derivative $D_x f$ it is enough to ensure that the smoothness is satisfied for

$$g_{ij}(x) := \langle D_x f(e_i), D_x f(e_j) \rangle_{f(x)} \tag{A.5.41}$$

where $\{e_i\}$ is the standard basis for $\mathbb{R}^d$.

Definition A.5.34 (Isometry) Suppose (M, g_M), and (N, g_N) are two Riemannian manifolds. A smooth map $\phi : M \to N$ is called an isometry if it is a diffeomorphism such that

$$\phi^* g_N = g_M \tag{A.5.42}$$

where $g_{N,M}$ is the Riemannian structure on M, N. Manifolds which are related by an isometry are said to be isometric. Furthermore, ϕ is called a local isometry between M and N is every point $p \in M$ has a neighbourhood U such that $\phi\big|_U : U \to V \subset N$ is an isometry.

Definition A.5.35 (Orthonormal frame) Let (M, g) be a d-dimensional Riemannian manifold. A local frame $(e_1, \ldots, e_d)$ for M defined on an open subset $U \subset M$ is orthonormal if $(e_1|_p, \ldots, e_d|_p)$ is an orthonormal basis for T_pM, i.e. $\langle e_i, e_j \rangle = \delta_{ij}$

Proposition A.5.36 *For all points $p \in M$ there is a neighbourhood with a smooth orthonormal frame.* □

Proposition A.5.37 *Let (M, g) be an oriented Riemannian manifold. Then M has a uniquely determined orientation form vol_M such that*

$$\mathrm{vol}_M(e_1, \ldots, e_d) = 1 \tag{A.5.43}$$

for every positively oriented orthonormal basis $e_1, \ldots, e_d$ of T_pM. □

Proposition A.5.38 *In local chart coordinates $x^1, \ldots, x^d$ for an orientation preserving local parametrisation $f : V \to U \subset M$ we get that the pullback of the volume form vol_M is*

$$(f^*(\mathrm{vol}_M))_x = \sqrt{\det g_{ij}(x)} dx^1 \wedge \cdots \wedge dx^d \tag{A.5.44}$$

□

A.6 Connections on manifolds

In this section we introduce the concept of connections, geodesics, and parallel transport.

Definition A.6.1 (Connection on vector bundle) Let $E \xrightarrow{\pi} M$ be a smooth vector bundle over a smooth manifold M. Denoting the spaces of smooth sections of the vector bundle as $\Gamma(E)$. A *connection* on this bundle is an $\mathbb{R}$-linear map

$$\nabla : \Gamma(E) \to \Gamma(E \otimes T^*M) \tag{A.6.1}$$

where T^*M denotes the cotangent bundle. This map must satisfy the Leibniz rule

$$\nabla(\sigma f) = (\nabla\sigma)f + \sigma \otimes df \tag{A.6.2}$$

where $f \in \Omega^0(M)$ is a function on M and $\sigma \in \Gamma(E)$ is a smooth section.

Since the result is a section in $\Gamma(E \otimes T^*M)$ one can feed this a tangent vector (field) v in order to define a covariant derivative along this vector (field). This results in the map

$$\nabla_v : \Gamma(E) \to \Gamma(E) \tag{A.6.3}$$

by

$$\nabla_v\sigma = (\nabla\sigma)(v). \tag{A.6.4}$$

The covariant derivative then satisfies the following

i. $\nabla_v(\sigma_1 + \sigma_2) = \nabla_v\sigma_1 + \nabla_v\sigma_2$
ii. $\nabla_{v+u}\sigma = \nabla_v\sigma + \nabla_u\sigma$
iii. $\nabla_v(f\sigma) = f\nabla_v\sigma + \sigma \otimes df(v) = f\nabla_v\sigma + v(f)\sigma$
iv. $\nabla_{fv}\sigma = f\nabla_v\sigma$

for $f \in \Omega^0(M)$ and $\sigma \in \Gamma(E)$.

Example A.6.2 Let $v_p = v_p^\mu(\partial_\mu)_p$ be a tangent vector field at $p \in M$ and $\sigma \in \Gamma(E)$. Since $\sigma \in \Gamma(E)$, that means that at every point p on M we can decompose the vector $\sigma(p) = \sigma(p)^\nu e_\nu$ where e_ν is a basis for the typical fibre at p. We can then see σ as coordinate functions $\sigma^\nu \in \Omega^0(M)$ combined with the basis which is given by a section $e_\nu \in \Gamma(E)$. We can now talk about taking the covariant derivative of σ with respect to the tangent vector field v:

$$\nabla_v(\sigma) = \nabla_{v^\mu\partial_\mu}(\sigma^\nu e_\nu). \tag{A.6.5}$$

We start by using rule *iv* which gives

$$\nabla_{v^\mu\partial_\mu}(\sigma^\nu e_\nu) = v^\mu\nabla_{\partial_\mu}(\sigma^\nu e_\nu). \tag{A.6.6}$$

Then, since σ^ν are functions we use rule *iii* to obtain

$$v^\mu\nabla_{\partial_\mu}(\sigma^\nu e_\nu) = v^\mu(\sigma^\nu\nabla_{\partial_\mu}(e_\nu) + \partial_\mu(\sigma^\nu)e_\nu). \tag{A.6.7}$$

Since $\nabla_{\partial_\mu}(e_\nu) \in \Gamma(E)$ we can decompose this as we did with σ above. This yields $\nabla_{\partial_\mu}(e_\nu) = (\nabla_{\partial_\mu}(e_\nu))^\alpha e_\alpha$. Often $(\nabla_{\partial_\mu}(e_\nu))^\alpha$ is denoted by the symbol $\Gamma^\alpha_{\mu\nu}$ called the *Christoffel symbols*, which are not to be confused with the space of smooth sections. With this symbol we get

$$\nabla_v(\sigma) = \nabla_{v^\mu\partial_\mu}(\sigma^\nu e_\nu) = v^\mu(\sigma^\nu\Gamma^\alpha_{\mu\nu}e_\alpha + \partial_\mu(\sigma^\nu)e_\nu). \tag{A.6.8}$$

Relabelling a dummy index we obtain

$$\nabla_v(\sigma) = v^\mu(\sigma^\nu\Gamma^\alpha_{\mu\nu} + \partial_\mu(\sigma^\alpha))e_\alpha. \tag{A.6.9}$$

Choosing the tangent vector field just be the basis vector field ∂_μ we get the, in general relativity, well known expression

$$\nabla_\mu(\sigma) := \nabla_{\partial_\mu}(\sigma) = (\partial_\mu(\sigma^\alpha) + \sigma^\nu\Gamma^\alpha_{\mu\nu})e_\alpha. \tag{A.6.10}$$

Note that to compute this one needs expressions for

$$\Gamma^{\alpha}_{\mu\nu} = (\nabla_{\partial_\mu}(e_\nu))^\alpha; \tag{A.6.11}$$

roughly speaking, one needs to specify how the basis vectors change between points on the manifold.

Example A.6.3 In the case where the manifold is flat one can choose

$$\Gamma^{\alpha}_{\mu\nu} = (\nabla_{\partial_\mu}(e_\nu))^\alpha = 0, \tag{A.6.12}$$

i.e. one has constant basis vectors. Then we can compute the covariant derivative of a vector field σ with respect to another vector field v:

$$\nabla_v(\sigma) = \nabla_{v^\mu\partial_\mu}(\sigma^\nu e_\nu) = v^\mu(\sigma^\nu\nabla_{\partial_\mu}(e_\nu) + \partial_\mu(\sigma^\nu)e_\nu), \tag{A.6.13}$$

and since $\nabla_{\partial_\mu}(e_\nu) = 0$ we have

$$\nabla_v(\sigma) = v^\mu\partial_\mu(\sigma^\nu)e_\nu. \tag{A.6.14}$$

We can now express the components of the vector field as

$$v^\mu = dx^\mu(v) \tag{A.6.15}$$

and thus arrive at

$$\nabla_v(\sigma) = v^\mu\partial_\mu(\sigma^\nu)e_\nu = d\sigma(v). \tag{A.6.16}$$

Hence, for a flat space we have

$$\nabla_v(\sigma) = d\sigma(v) \tag{A.6.17}$$

and the covariant derivative is just the external derivative known from the study of differential forms, which just returns the derivative of a function along a vector; here v.

We can now turn to geodesics which roughly are the straightest possible curves one can move along on a manifold.

Definition A.6.4 (Geodesic) A *geodesic* through a point $p \in M$, where M is a smooth manifold with connection ∇ on the tangent bundle TM, is a curve $\gamma : I \to M$ such that

i) $\gamma(0) = p$
ii) $\nabla_{\dot\gamma(t)}\dot\gamma(t) = 0$ for all $t \in I$.

We can also talk about sections parallel to a curve.

Definition A.6.5 (Section parallel to curve) Let $E \xrightarrow{\pi} M$ be a smooth vector bundle over a smooth manifold M with a connection ∇, and $\gamma : I \to M$ a smooth curve in the manifold. Then a section $\sigma \in \Gamma(E)$ is parallel to the curve γ if

$$\nabla_{\dot\gamma(t)}\sigma(\gamma(t)) = 0 \tag{A.6.18}$$

for all $t \in I$.

This section is unique and it's possible to generate this parallel section σ given a curve γ such that $\gamma(0) = p$ and a starting vector $\sigma_p \in V_p$ by solving the ODE obtained from

$$\nabla_{\dot{\gamma}(t)}\sigma(\gamma(t)) = 0 \tag{A.6.19}$$

with the initial condition

$$\sigma(\gamma(0)) = \sigma(p) = \sigma_p. \tag{A.6.20}$$

With a local chart (U, ϕ) around $p = \gamma(0) \in M$ containing the endpoint $\gamma(1)$ such that $(\phi \circ \gamma)(t) = x(t)$ we have that the tangent vector takes the form

$$\dot{\gamma}(t) = \frac{dx^i}{dt}\left(\frac{\partial}{\partial x^i}\right)_{\gamma(t)}$$

and with the shortcut in notation

$$\nabla\left(\frac{\partial}{\partial x^i}\right)_{\gamma(t)} = \nabla_i \tag{A.6.21}$$

we can rewrite equation (A.6.18) as

$$\frac{dx^i}{dt}\nabla_i(\sigma(x(t))) = 0 \tag{A.6.22}$$

which gives a way to calculate the parallel sections in a given coordinate system x.

B

Code samples of group invariant networks

In this appendix we present the networks used in the tests presented in Chapter 4.3.1. The networks uses code published by Cohen et al. accompanying their article [9].

We have tested two different sizes of networks: one with three convolutional layers and one with seven convolutional layers, combined with invariance under different groups: $\mathbb{Z}^2$, C_4, and D_4 where the $\mathbb{Z}^2$ invariant network is just an ordinary CNN with a global average at the end.

```
import tensorflow as tf
from tensorflow import keras
from keras_gcnn.layers.convolutional import GConv2D  # Import the
    group convolution by Cohen et al.
from keras_gcnn.layers.normalization import GBatchNorm  # Import
    BatchNormalisation written to work with group
# equi-/invariant networks
from keras_gcnn.layers.pooling import GroupPool  # Import the
    pooling over the group channels
import numpy as np


def equivariantNetwork(group, padding='valid', large=False):
    # Define the layers that all the networks will use.
    softmax = tf.keras.layers.Activation('softmax')
    maxPooling = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))
    globalAvgPooling = tf.keras.layers.GlobalAveragePooling2D()

    if not large:  # The smaller network structure (3 conv layers)

        # Define the activation layers used after each convolution
        relu1 = tf.keras.layers.Activation('relu')
        relu2 = tf.keras.layers.Activation('relu')
        relu3 = tf.keras.layers.Activation('relu')

        if group == 'Z2':  # This is just an ordinary CNN

            # Define the convolution layers
            conv1 = tf.keras.layers.Conv2D(10, 3, padding=padding)
            conv2 = tf.keras.layers.Conv2D(10, 3, padding=padding)
            conv3 = tf.keras.layers.Conv2D(10, 3, padding=padding)

            # Define the normalisation and group pooling
            norm = tf.keras.layers.BatchNormalization()
```

```
            gPooling = tf.keras.layers.Lambda(lambda x: x)  # identity layer since no group action when only Z2

        elif group is 'C4':  # Network invariant under C4

            # Define the convolutional layers. Note that the number of filters has been reduced to keep the number of
            # parameters roughly the same as for the ordinary CNN
            conv1 = GConv2D(round(10 / 2), 3, 'Z2', group, input_shape=(28, 28, 1), padding=padding)
            conv2 = GConv2D(round(10 / 2), 3, group, group, padding=padding)
            conv3 = GConv2D(10, 3, group, group, padding=padding)

            # Define the normalisation and group pooling
            norm = GBatchNorm(group)
            gPooling = GroupPool(group)

        else:  # Network invariant under D4

            # Define the convolutional layers. Note that the number of filters has been reduced to keep the number of
            # parameters roughly the same as for the ordinary CNN
            conv1 = GConv2D(round(10 / 3), 3, 'Z2', group, input_shape=(28, 28, 1), padding=padding)
            conv2 = GConv2D(round(10 / 3), 3, group, group, padding=padding)
            conv3 = GConv2D(10, 3, group, group, padding=padding)

            # Define the normalisation and group pooling
            norm = GBatchNorm(group)
            gPooling = GroupPool(group)

        # Use Keras functional API to construct the network with the layers defined above
        inputs = keras.Input(shape=(28, 28, 1))

        x = conv1(inputs)
        x = relu1(x)
        x = conv2(x)
        x = relu2(x)
        x = maxPooling(x)
        x = norm(x, training=False)
        x = conv3(x)
        x = relu3(x)
        x = gPooling(x)
        x = globalAvgPooling(x)
        outputs = softmax(x)

    else:  # The larger network structure with 7 convolutional layers

        # Define the activation layers used after each convolution
        relu1 = tf.keras.layers.Activation('relu')
        relu2 = tf.keras.layers.Activation('relu')
        relu3 = tf.keras.layers.Activation('relu')
```

```
79          relu4 = tf.keras.layers.Activation('relu')
80          relu5 = tf.keras.layers.Activation('relu')
81          relu6 = tf.keras.layers.Activation('relu')
82          relu7 = tf.keras.layers.Activation('relu')
83
84          if group == 'Z2':  # This is just an ordinary CNN
85
86              # Define the convolution layers
87              conv1 = tf.keras.layers.Conv2D(10, 3, padding=padding)
88              conv2 = tf.keras.layers.Conv2D(10, 3, padding=padding)
89              conv3 = tf.keras.layers.Conv2D(10, 3, padding=padding)
90              conv4 = tf.keras.layers.Conv2D(10, 3, padding=padding)
91              conv5 = tf.keras.layers.Conv2D(10, 3, padding=padding)
92              conv6 = tf.keras.layers.Conv2D(10, 3, padding=padding)
93              conv7 = tf.keras.layers.Conv2D(10, 3, padding=padding)
94
95              # Define the normalisation and group pooling
96              norm = tf.keras.layers.BatchNormalization()
97              gPooling = tf.keras.layers.Lambda(lambda x: x)  #
    identity layer since no group action when only Z2
98
99          elif group is 'C4':  # Network invariant under C4
100
101             # Define the convolutional layers. Note that the number
     of filters has been reduced to keep the number of
102             # parameters roughly the same as for the ordinary CNN
103             conv1 = GConv2D(round(10 / 2), 3, 'Z2', group,
    input_shape=(28, 28, 1), padding=padding)
104             conv2 = GConv2D(round(10 / 2), 3, group, group, padding
    =padding)
105             conv3 = GConv2D(round(10 / 2), 3, group, group, padding
    =padding)
106             conv4 = GConv2D(round(10 / 2), 3, group, group, padding
    =padding)
107             conv5 = GConv2D(round(10 / 2), 3, group, group, padding
    =padding)
108             conv6 = GConv2D(round(10 / 2), 3, group, group, padding
    =padding)
109             conv7 = GConv2D(10, 3, group, group, padding=padding)
110
111             # Define the normalisation and group pooling
112             norm = GBatchNorm(group)
113             gPooling = GroupPool(group)
114
115         else:  # Network invariant under D4
116
117             # Define the convolutional layers. Note that the number
     of filters has been reduced to keep the number of
118             # parameters roughly the same as for the ordinary CNN
119             conv1 = GConv2D(int(np.floor(10 / np.sqrt(8))), 3, 'Z2'
    , group, input_shape=(28, 28, 1), padding=padding)
120             conv2 = GConv2D(int(np.floor(10 / np.sqrt(8))), 3,
    group, group, padding=padding)
121             conv3 = GConv2D(int(np.floor(10 / np.sqrt(8))), 3,
    group, group, padding=padding)
122             conv4 = GConv2D(int(np.floor(10 / np.sqrt(8))), 3,
```

```
    group, group, padding=padding)
123           conv5 = GConv2D(int(np.floor(10 / np.sqrt(8))), 3,
    group, group, padding=padding)
124           conv6 = GConv2D(int(np.floor(10 / np.sqrt(8))), 3,
    group, group, padding=padding)
125           conv7 = GConv2D(10, 3, group, group, padding=padding)
126
127           # Define the normalisation and group pooling
128           norm = GBatchNorm(group)
129           gPooling = GroupPool(group)
130
131       # Use Keras functional API to construct the network with
    the layers defined above
132       inputs = keras.Input(shape=(28, 28, 1))
133
134       x = conv1(inputs)
135       x = relu1(x)
136       x = conv2(x)
137       x = relu2(x)
138       x = maxPooling(x)
139       x = norm(x, training=False)
140       x = conv3(x)
141       x = relu3(x)
142       x = conv4(x)
143       x = relu4(x)
144       x = conv5(x)
145       x = relu5(x)
146       x = conv6(x)
147       x = relu6(x)
148       x = conv7(x)
149       x = relu7(x)
150       x = gPooling(x)
151       x = globalAvgPooling(x)
152       outputs = softmax(x)
153
154    # Finally actually create the model
155    model = keras.Model(inputs=inputs, outputs=outputs)
156
157    return model  # Return the constructed model
```

These models can then be tested on the MNIST dataset.

```
1  import tensorflow as tf
2  from tensorflow import keras
3  import numpy as np
4  import matplotlib.pyplot as plt
5
6  # Importing the MNIST data set
7  mnist = keras.datasets.mnist
8
9  (train_images, train_labels), (test_images, test_labels) = mnist.
       load_data()
10
11 train_labels = keras.utils.to_categorical(train_labels)
12 test_labels = keras.utils.to_categorical(test_labels)
13
14 train_images = train_images.reshape(len(train_images), 28, 28, 1)
```

```
test_images = test_images.reshape(len(test_images), 28, 28, 1)

train_images = train_images/255.0
test_images = test_images/255.0

# Define the network to be used
group = 'C4'
model = equivariantNetwork(group=group)
model.summary()

model.compile(optimizer='adam',loss='categorical_crossentropy',
    metrics=['accuracy'])  # Final set up of model

model.fit(train_images[:],train_labels[:],epochs=5,validation_data
    =(test_images,test_labels))  # Train the model

test_loss, test_acc = model.evaluate(test_images,test_labels)  #
    Evaluate after training
```

To finally be tested to check for invariance.

```
# Test of invariance (due to the rotation invariant global average
    pool) by manually rotating a single image

tmp = test_images[0:4,:,:,:]
tmp[1]=np.rot90(tmp[0],k=1)
tmp[2]=np.rot90(tmp[0],k=2)
tmp[3]=np.rot90(tmp[0],k=3)

images = np.array(model(tmp).numpy())  # Feed the rotated images
    through the model

# Plot the result from the model for the differently rotated images
    , if network is invariant these should be the same
input_shape = np.shape(images)
numChan = 1
numImgs = len(images)
fig, axes = plt.subplots(nrows=numImgs, ncols=numChan)

for ax, ind in zip(axes.flatten(), range(numChan * numImgs)):
    image = images[ind, :]
    ax.plot(image)
    ax.set_ylabel('Image nr. ' + str(ind))

plt.show(block=True)
```

C

Result tables

In this appendix we present more data obtained through the network tests. Each accuracy is determined as the mean of 10 runs and in each run the network is trained for 5 epochs. As a subscript to each accuracy is the standard deviation for that run of 10 iterations. No table is presented on this page due to space limitations. Also due to space limitations there will not be any further text in this appendix apart from table captions.

Table C.1: The results for the three types of CNNs with the **smaller** structure and **valid** padding: ordinary, C_4 invariant, and D_4 invariant. The data, training and testing, is either transformed (T) or not (N) under C_4.

(a) Ordinary CNN

		Test	
		N	T
Train	N	$0.877_{\sigma=0.014}$	$0.343_{\sigma=0.017}$
	T	$0.571_{\sigma=0.044}$	$0.571_{\sigma=0.020}$

(b) C_4 invariant CNN

		Test	
		N	T
Train	N	$0.721_{\sigma=0.044}$	$0.704_{\sigma=0.028}$
	T	$0.707_{\sigma=0.066}$	$0.704_{\sigma=0.045}$

(c) D_4 invariant CNN

		Test	
		N	T
Train	N	$0.685_{\sigma=0.055}$	$0.687_{\sigma=0.059}$
	T	$0.696_{\sigma=0.027}$	$0.679_{\sigma=0.052}$

Table C.2: The results for the three types of CNNs with the **smaller** structure and **valid** padding: ordinary, C_4 invariant, and D_4 invariant. The data, training and testing, is either transformed (T) or not (N) under D_4.

(a) Ordinary CNN

		Test	
		N	T
Train	N	$0.871_{\sigma=0.017}$	$0.279_{\sigma=0.013}$
	T	$0.530_{\sigma=0.051}$	$0.544_{\sigma=0.046}$

(b) C_4 invariant CNN

		Test	
		N	T
Train	N	$0.695_{\sigma=0.023}$	$0.582_{\sigma=0.029}$
	T	$0.636_{\sigma=0.028}$	$0.679_{\sigma=0.046}$

(c) D_4 invariant CNN

		Test	
		N	T
Train	N	$0.686_{\sigma=0.048}$	$0.680_{\sigma=0.039}$
	T	$0.654_{\sigma=0.028}$	$0.681_{\sigma=0.034}$

Table C.3: The results for the three types of CNNs with the **smaller** structure and **valid** padding: ordinary, C_4 invariant, and D_4 invariant. The data, training and testing, is either transformed (T) or not (N) under $SO(2)$.

(a) Ordinary CNN

		Test	
		N	T
Train	N	$0.855_{\sigma=0.041}$	$0.280_{\sigma=0.013}$
	T	$0.476_{\sigma=0.076}$	$0.485_{\sigma=0.023}$

(b) C_4 invariant CNN

		Test	
		N	T
Train	N	$0.716_{\sigma=0.018}$	$0.548_{\sigma=0.022}$
	T	$0.633_{\sigma=0.047}$	$0.633_{\sigma=0.029}$

(c) D_4 invariant CNN

		Test	
		N	T
Train	N	$0.695_{\sigma=0.028}$	$0.567_{\sigma=0.027}$
	T	$0.672_{\sigma=0.043}$	$0.610_{\sigma=0.039}$

Table C.4: The results for the three types of CNNs with the **smaller** structure and **valid** padding: ordinary, C_4 invariant, and D_4 invariant. The data, training and testing, is either transformed (T) or not (N) under $O(2)$.

(a) Ordinary CNN

		Test	
		N	T
Train	N	$0.869_{\sigma=0.031}$	$0.254_{\sigma=0.015}$
	T	$0.484_{\sigma=0.033}$	$0.486_{\sigma=0.018}$

(b) C_4 invariant CNN

		Test	
		N	T
Train	N	$0.705_{\sigma=0.058}$	$0.526_{\sigma=0.020}$
	T	$0.629_{\sigma=0.048}$	$0.585_{\sigma=0.028}$

(c) D_4 invariant CNN

		Test	
		N	T
Train	N	$0.672_{\sigma=0.042}$	$0.577_{\sigma=0.018}$
	T	$0.632_{\sigma=0.033}$	$0.625_{\sigma=0.032}$

Table C.5: The results for the three types of CNNs with the **smaller** structure and **same** padding: ordinary, C_4 invariant, and D_4 invariant. The data, training and testing, is either transformed (T) or not (N) under C_4.

(a) Ordinary CNN

		Test	
		N	T
Train	N	$0.81_{\sigma=0.073}$	$0.361_{\sigma=0.025}$
	T	$0.491_{\sigma=0.035}$	$0.491_{\sigma=0.023}$

(b) C_4 invariant CNN

		Test	
		N	T
Train	N	$0.592_{\sigma=0.033}$	$0.59_{\sigma=0.031}$
	T	$0.569_{\sigma=0.03}$	$0.542_{\sigma=0.033}$

(c) D_4 invariant CNN

		Test	
		N	T
Train	N	$0.578_{\sigma=0.059}$	$0.567_{\sigma=0.083}$
	T	$0.559_{\sigma=0.051}$	$0.534_{\sigma=0.036}$

Table C.6: The results for the three types of CNNs with the **smaller** structure and **same** padding: ordinary, C_4 invariant, and D_4 invariant. The data, training and testing, is either transformed (T) or not (N) under D_4.

(a) Ordinary CNN

		Test	
		N	T
Train	N	$0.797_{\sigma=0.029}$	$0.293_{\sigma=0.02}$
	T	$0.454_{\sigma=0.038}$	$0.461_{\sigma=0.035}$

(b) C_4 invariant CNN

		Test	
		N	T
Train	N	$0.587_{\sigma=0.038}$	$0.44_{\sigma=0.033}$
	T	$0.522_{\sigma=0.054}$	$0.553_{\sigma=0.036}$

(c) D_4 invariant CNN

		Test	
		N	T
Train	N	$0.535_{\sigma=0.05}$	$0.529_{\sigma=0.049}$
	T	$0.532_{\sigma=0.048}$	$0.549_{\sigma=0.053}$

Table C.7: The results for the three types of CNNs with the **larger** structure and **valid** padding: ordinary, C_4 invariant, and D_4 invariant. The data, training and testing, is either transformed (T) or not (N) under C_4.

(a) Ordinary CNN

		Test	
		N	T
Train	N	$0.778_{\sigma=0.091}$	$0.378_{\sigma=0.048}$
	T	$0.768_{\sigma=0.09}$	$0.745_{\sigma=0.138}$

(b) C_4 invariant CNN

		Test	
		N	T
Train	N	$0.842_{\sigma=0.089}$	$0.872_{\sigma=0.125}$
	T	$0.879_{\sigma=0.13}$	$0.958_{\sigma=0.036}$

(c) D_4 invariant CNN

		Test	
		N	T
Train	N	$0.922_{\sigma=0.075}$	$0.945_{\sigma=0.028}$
	T	$0.945_{\sigma=0.026}$	$0.938_{\sigma=0.036}$

Table C.8: The results for the three types of CNNs with the **larger** structure and **valid** padding: ordinary, C_4 invariant, and D_4 invariant. The data, training and testing, is either transformed (T) or not (N) under D_4.

(a) Ordinary CNN

		Test	
		N	T
Train	N	$0.748_{\sigma=0.135}$	$0.271_{\sigma=0.038}$
	T	$0.683_{\sigma=0.051}$	$0.698_{\sigma=0.143}$

(b) C_4 invariant CNN

		Test	
		N	T
Train	N	$0.91_{\sigma=0.094}$	$0.591_{\sigma=0.061}$
	T	$0.863_{\sigma=0.059}$	$0.902_{\sigma=0.043}$

(c) D_4 invariant CNN

		Test	
		N	T
Train	N	$0.95_{\sigma=0.029}$	$0.949_{\sigma=0.03}$
	T	$0.919_{\sigma=0.047}$	$0.939_{\sigma=0.037}$

Table C.9: The results for the three types of CNNs with the **larger** structure and **same** padding: ordinary, C_4 invariant, and D_4 invariant. The data, training and testing, is either transformed (T) or not (N) under C_4.

(a) Ordinary CNN

		Test	
		N	T
Train	N	$0.961_{\sigma=0.03}$	$0.414_{\sigma=0.013}$
	T	$0.847_{\sigma=0.025}$	$0.84_{\sigma=0.022}$

(b) C_4 invariant CNN

		Test	
		N	T
Train	N	$0.926_{\sigma=0.017}$	$0.937_{\sigma=0.011}$
	T	$0.944_{\sigma=0.008}$	$0.937_{\sigma=0.017}$

(c) D_4 invariant CNN

		Test	
		N	T
Train	N	$0.884_{\sigma=0.037}$	$0.883_{\sigma=0.03}$
	T	$0.887_{\sigma=0.026}$	$0.891_{\sigma=0.028}$

Table C.10: The results for the three types of CNNs with the **larger** structure and **same** padding: ordinary, C_4 invariant, and D_4 invariant. The data, training and testing, is either transformed (T) or not (N) under D_4.

(a) Ordinary CNN

		Test	
		N	T
Train	N	$0.855_{\sigma=0.258}$	$0.331_{\sigma=0.015}$
	T	$0.75_{\sigma=0.027}$	$0.779_{\sigma=0.024}$

(b) C_4 invariant CNN

		Test	
		N	T
Train	N	$0.932_{\sigma=0.022}$	$0.621_{\sigma=0.027}$
	T	$0.854_{\sigma=0.026}$	$0.894_{\sigma=0.016}$

(c) D_4 invariant CNN

		Test	
		N	T
Train	N	$0.89_{\sigma=0.03}$	$0.893_{\sigma=0.026}$
	T	$0.909_{\sigma=0.016}$	$0.811_{\sigma=0.241}$

D

List of definitions

Bibliography

[1] A. M. Turing, "Computing Machinery and Intelligence", Mind **LIX**, 433 (1950), https://academic.oup.com/mind/article/LIX/236/433/986238 (visited on 04/20/2020).

[2] *Machine, Learning, 1951*, https://www.the-scientist.com/foundations/machine--learning--1951-65792 (visited on 05/29/2020).

[3] K. Albertsson et al., "Machine Learning in High Energy Physics Community White Paper", (2019), arXiv:1807.02876.

[4] S. Seyedzadeh, F. P. Rahimian, I. Glesk, and M. Roper, "Machine learning for estimation of building energy consumption and performance: a review", Visualization in Engineering **6**, 5 (2018), https://doi.org/10.1186/s40327-018-0064-7 (visited on 05/26/2020).

[5] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search", Nature **529**, 484 (2016), https://www.nature.com/articles/nature16961 (visited on 05/26/2020).

[6] *Convolutional neural network*, in *Wikipedia*, Page Version ID: 961118375 (June 6, 2020), https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=961118375 (visited on 06/07/2020).

[7] *Feedforward neural network*, in *Wikipedia*, Page Version ID: 960861503 (June 5, 2020), https://en.wikipedia.org/w/index.php?title=Feedforward_neural_network&oldid=960861503 (visited on 06/07/2020).

[8] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow, "Harmonic Networks: Deep Translation and Rotation Equivariance", (2017), arXiv:1612.04642.

[9] T. S. Cohen and M. Welling, "Group Equivariant Convolutional Networks", (2016), arXiv:1602.07576.

[10] Y. LeCun, M. M. Bronstein, J. Bruna, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond Euclidean data", IEEE Signal Processing Magazine **34**, 18 (2017), arXiv:1611.08097.

[11] M. C. N. Cheng et al., "Covariance in Physics and Convolutional Neural Networks", (2019), arXiv:1906.02481.

[12] T. Cohen, M. Geiger, and M. Weiler, "A General Theory of Equivariant CNNs on Homogeneous Spaces", (2020), arXiv:1811.02017.

[13] K. Team, *Keras documentation: MNIST digits classification dataset*, https://keras.io/api/datasets/mnist/ (visited on 05/27/2020).

[14] T. S. Cohen, *GrouPy*, https://github.com/tscohen/GrouPy (visited on 12/11/2019).

[15] B. Veeling, *Keras GCNN*, Apr. 14, 2020, https://github.com/basveeling/keras-gcnn (visited on 05/03/2020).

[16] P. Mehta et al., "A high-bias, low-variance introduction to Machine Learning for physicists", Physics Reports **810**, 1 (2019), arXiv:1803.08823.

[17] T. S. Cohen and M. Welling, "Steerable CNNs", (2016), arXiv:1612.08498.

[18] E. Hoogeboom, J. W. T. Peters, T. S. Cohen, and M. Welling, "HexaConv", (2018), arXiv:1803.02108.

[19] ehoogeboom, *Ehoogeboom/hexaconv*, Oct. 22, 2019, https://github.com/ehoogeboom/hexaconv (visited on 11/08/2019).

[20] S. Dieleman, J. De Fauw, and K. Kavukcuoglu, "Exploiting Cyclic Symmetry in Convolutional Neural Networks", (2016), arXiv:1602.02660.

[21] *Keras: the Python deep learning API*, https://keras.io/ (visited on 05/26/2020).

[22] *TensorFlow*, https://www.tensorflow.org/ (visited on 05/26/2020).

[23] D. Worrall, *Harmonic Convolutions*, May 17, 2020, https://github.com/deworrall92/harmonicConvolutions (visited on 05/18/2020).

[24] M. Weiler and G. Cesa, "General E(2)-Equivariant Steerable CNNs", (2019), arXiv:1911.08251.

[25] T. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling, "Gauge Equivariant Convolutional Networks and the Icosahedral CNN", ArXiv **abs/1902.04615** (2019), arXiv:1902.04615.

[26] A. Marsh, "Gauge Theories and Fiber Bundles: Definitions, Pictures, and Results", (2019), arXiv:1607.03089.

[27] *Lecture series on "The Geometric anatomy of Theoretical Physics"*, in collab. with F. P. Schuller, Friedrich-Alexander-Universität Erlangen-Nürnberg, Institut für Theoretische Physik, 2013–2014, https://mathswithphysics.blogspot.com/2016/07/frederic-schullers-lectures-on-quantum.html.

Figure Sources

[28] Glosser, *Artificial neural network*, https://commons.wikimedia.org/w/index.php?curid=24913461 (visited on 05/29/2020).

[29] Aphex34, *Convolutional neural network*, https://commons.wikimedia.org/w/index.php?curid=45679374 (visited on 05/29/2020).

[30] Fjung, *Parallel transport on a sphere*, https://commons.wikimedia.org/wiki/File:Connection-on-sphere.png (visited on 05/29/2020).

[31] T. S. Cohen and M. Welling, *Figure 1 from "group equivariant convolutional networks"*, June 3, 2016, arXiv:1602.07576, http://arxiv.org/abs/1602.07576 (visited on 11/07/2019).