# Group Equivariance and Invariance in Neural Networks

*William J. Toner*

Master of Science
School of Informatics
University of Edinburgh
2019

# Abstract

Recent work has shown that despite impressive performance on certain standard datasets, modern image classification techniques still have a number of flaws. In particular, neural networks show a lack of robustness in the face of dataset biases and shifts in domain. Classifiers generally apply learning methods which concentrate on superficial class and dataset specific variations rather than core features of the object class as understood by humans. We look at a range of methods of building classifiers which are structurally invariant to simple group actions. Examples of such groups include reflection, rotation, changes in brightness and contrast and scale. We show that this can be done in a theoretically principled way which leads to improvements in robustness without loss in accuracy.

## Acknowledgements

This work was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

We would like to thank Prof. Amos Storkey for supervising this project. We wish also to acknowledge the support and advice given by members of his research group throughout the past months.

# Table of Contents

# Chapter 1

# Introduction

Over the last few decades there has been an explosion of interest and application of machine learning methods. Most popular among these methods is the neural network which has proved surprisingly successful on a range of tasks which had historically proved difficult for computers. Most notable among these is image classification (Huang et al. (2018) Krizhevsky et al. (2012)). These results are despite the lack of an accepted theory which explains this success. This has led to these methods being dubbed “black-box" methods. It has become customary to throw complex deep architectures at tasks without significant conscious applications of domain knowledge. This is further compounded by the wide-variety of different architectures and the lack of a comprehensive set of rules by which one might choose a network for a given task. This dissertation investigates the extent to which the use of group equivariances can solve these problems.

One of the main flaws of neural networks is their lack of robustness to perturbations and shifts in domain. fig. 1.1 depicts two pictures of panda bears. The panda on the left is correctly categorised with probability 57.7%. On the right is the same picture with the addition of a small amount of noise, now classified as a gibbon (Goodfellow et al. (2015)). This is an example of an adversarial example in this case constructed using a technique called the fast gradient sign method. Adversarial examples can be constructed for many varieties of network and across a wide variety of domain types from images to natural language. These examples, almost indistinguishable from their originals to the human eye, are quite shocking when first encountered. They indicate a profound lack of stability of networks to shift in domain (Tommasi et al. (2015) Ian Goodfellow (2019)Goodfellow et al. (2017)).

Aside from methods deliberately designed to attack networks, even minor reasonable transformations to an image such as rotation or inversion are able to totally confound a classifier (Engstrom et al. (2017)). Furthermore, perversely, these adversarial examples have exhibited a high degree of robustness. In particular, these examples may be distorted spatially and still confound visual systems. Examples of this (fig. 1.2) included a 3D-printed turtle (Athalye et al. (2018)) which is classified as a rifle by Google Goggles from all angles. Another example is a sticker which can be placed on objects resulting in them being classified as a toasters with high confidence (Brown et al. (2017)).

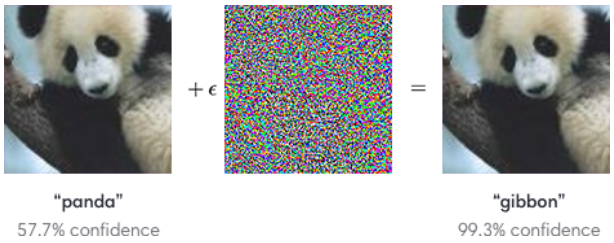


Figure 1.1: Adversarial Panda classified as a Gibbon Goodfellow et al. (2015)

There are many reasons why we are interested in improving the robustness of object recognition systems. In a world with increasingly large numbers of machine learning based systems, improving stability and reducing susceptibility to adversarial attack is imperative to safety. As a specific example, self-driving vehicles need to be able to recognise *Stop* signs both consistently and confidently (Ian Goodfellow (2019)). To compound this point, it has been observed that adversarial examples have a high degree of transferability between models. This means that even closed systems are susceptible to black-box adversarial attacks (Tramer et al. (2018)). One may find adversarial examples for an accessible model and, with high probability, they will also confound other systems. Beyond issues of safety, improving robustness of neural networks to dataset bias and adversarial attacks will also allow for increased usage of machine learning methods in industrial design (Ian Goodfellow (2019)). Specifically, increased robustness improves the reliability of a neural network to make accurate predictions at inputs outside of the training-data manifold.

A second significant issue with modern usage of neural networks is the quantity of data required to achieve high levels of accuracy. There are many task where high accuracies can be obtained with reasonably small datasets ($\sim$100s) Cho et al. (2015). Nevertheless, deep architectures are generally understood to require more data than clas-


Figure 1.2: 3D-printed Turtle categorised as rifle Athalye et al. (2018)

sical equivalents (Raudys (1991)). It is commonplace, particularly within computational linguistics to use datasets consisting of millions, billions or even trillions of examples (Davies (2011)). For complex problems with minimal or noisy data this constitutes a significant problem. One can partially circumvent these issues using methods like k-fold validation and data augmentation schema which increase the effective size of the dataset. Furthermore, $k-$shot meta-learning potentially offers some hope of dealing with the dependence on big data. However, for the moment at least, vast quantities of task data is the preferred way to improve model accuracy.

Another substantial problem with current machine learning approaches is that they are often highly computationally intensive. Beyond the substantial environmental harm caused by this (Strubell et al. (2019)), it also constitutes a barrier to entry for those who wish to use deep-learning methods to process data. Even with large GPU clusters, industrial models can commonly take weeks to train. Given the general correlation between number of parameters and generalisation accuracy, there is a tendency to construct and run the largest possible models. Recent work (Frankle and Carbin (2018)) gives indications that the true trained models which underlie the sophisticated architectures may often be quite simple. Consequently, it remains possible that there are neural methods capable of the same performance and without the heavy machinery.

We have looked at three of the large problems faced by modern neural architectures; robustness, computational expense and dependence upon big data. On top of this, we've briefly mentioned a couple of the methods which are commonly used to tackle these problems including data augmentation and adversarial training (Zuo (2018)). These methods are successful in improving robustness, performance and resilience to adversarial attack thus mitigating the need for elaborate networks or vast quantities of data

(Kuchnik and Smith (2019)). Nevertheless, these methods are a profoundly inelegant application of inductive biases. Data augmentation consists of taking a group such as the group of rotations, selecting random elements from this group and training on elements from both the dataset $\mathcal{D}$, and an augmented version $g \circ \mathcal{D}$. Consequently this approach scales poorly when the cardinality of the group is large (Kuchnik and Smith (2019)). Depending on precisely how augmentation is applied, it will learn slower than a system in which these invariances are structurally inbuilt. A final downside is that it also requires various augmentation functions which adds to computational demands.

We believe that one of the causes of the problems we have outlined, is a lack of effort to consciously build inductive biases into models. Instead there is a reliance that large amounts of data will solve everything. The goal of this dissertation is to look at simple methods to build structurally invariant classifiers and to investigate the effect of these methods on robustness and performance. The aforementioned success of augmentation in regards to these criteria inspires hope that this approach should perform well. This is on top of a broad existing body of literature and the long history use of symmetries in other fields, most notably Physics where it is central to special relativity, gauge theory, Noether's theorem and the standard model of particle physics.

In the case of most domains there are known invariances; families of transformations under which the classifier ought to be invariant. A canonical example of this in the case of images is translation: This is to say, the class of an object is independent of its position. This invariance has the potential to be a powerful inductive bias. If used correctly it gives our classifier the labels for a large locus of points in $\mathbb{R}^N$ from the observation of just a single training data point, namely all of its translates. In mathematical terms, the ability to use group invariances means that each labelled data point can be expanded into an entire labelled group orbit covering a large region of space. This offers the possibility of faster training as well as better generalisation and robustness; the precise magnitude dictated by the cardinality and salience of $G$.

# Chapter 2

# Methodology

**Definition 2.0.1. Group Action** A group $G$ is said to act on a set $X$ if there is a map $\phi : G \times X \to X$ called the **Group Action** such that for all $x \in X$, $\phi(e,x) = x$ and $\phi(g,\phi(h,x)) = \phi(gh,x)$ for all $g,h \in G$, $x \in X$. For compactness we follow the practice of denoting a group action by a '$\circ$' e.g. $g \circ x$.

## 2.1 Motivation

In mathematical terms, a neural network is a large parameterised family of functions between two sets defined by our task. One uses gradient descent and a set of labelled data to select a function from our family which is consistent with the training data and generalises well to unseen problems of the same type. Generally speaking, without any prior information about the task, this is an unsolvable problem since there are an uncountably infinite number of possible functions which are consistent with the pairs within the training set. Consequently, in order to select a function one must make a number of assumptions about the form that it should take. Often this is done implicitly, for example one may only look at functions which are continuous or differentiable. It is useful however to make the remark that no progress can be made on any regression task without the application of inductive biases.

In the Bayesian case, the inductive bias comes in the form of a prior over functions. One then makes updates to the posterior based upon the probability of various 'observations' within the training set. In principle, the Bayesian approach constitutes a more justified approach to regression but it is not without difficulties. First, one runs into the issue that this is generally a computational unfeasible approach. There is no way

in which one can store in memory a representation of all possible distributions over the function space. One can partially circumvent this by considering certain families of priors such as in Gaussian processes however this represents the importation of an inductive bias in itself. One has the additional problem of calculating the posterior since this is generally an intractable calculation unless one is dealing with well-behaved families of distributions.

**Definition 2.1.1** (**Invariance**)**.** Let $G$ be a group which acts on the sets $X$. We say that the function $f : X \to Y$ is $G-$*invariant* if for all $x \in X, g \in G$ $f(g \circ x) = g \circ (f(x))$

**Definition 2.1.2** (**Equivariance**)**.** Let $G$ be a group which acts on the sets $X$ and $Y$. We say that the function $f : X \to Y$ is $G-$*equivariant* if for all $x \in X, g \in G$ $f(g \circ x) = g \circ (f(x))$

The reason that we make these remarks is to remind ourselves that there truly is no such thing as a free lunch (Wolpert and Macready (1997)): The success of neural networks (as with any algorithm) is attributable to their common set of properties and the inductive biases that these represent. It is not always easy to unpick what these inductive biases are or why they work. However, it is clear that these properties implicitly take advantage of common structures within data across a variety of domains. Historically this has always been well understood and algorithms in the past were crafted using task knowledge. However, the success of big data and neural networks has become a license to apply these methods blindly. In essence the term 'black-box' is more of a reflection on how neural networks are treated than the methods themselves. In this work we aim to support this point by demonstrating how the performance and robustness can be boosted by consciously including invariances.

The goal that we would like to achieve is to associate with each point in the domain an orbit representative under the action by the invariance group $G$. At this point we would only need to define the neural network on the space of orbit representatives $X_{rep} = {}^{G}/_{\circ}$. However, it is a non-trivial exercise to find a well-behaved choice of orbit representatives. Throughout this work we seek to accomplish this implicitly using a weaker property called equivariance (definition 2.1.2). We construct a network $F$ which produces equivariant representations $F(g \circ f) = g \circ F(f)$. We then add a $G$ invariant final layer so that overall our neural network is an expressive $G$-invariant map defined

upon the set of orbits.

**Definition 2.1.3.** The convolution on of a function $f : \mathbb{R}^2 \to \mathbb{R}$ with a kernel function $k$ (denoted $f \star k$) is defined as the following integral

$$f \star k(x) := \int_{\mathbb{R}^2} f(y)k(x-y)dy \tag{2.1}$$

**Definition 2.1.4** (Discrete Convolution)**.** Let $f,k : \mathbb{Z}^2 \to \mathbb{R}$ be compactly supported functions on $\mathbb{Z}^2$. The discrete convolution of $f$ with $k$ denoted with a '$\star$' is defined as follows[1].

$$k \star f[x][y] = \sum_{i,j \in S \subset \mathbb{Z}^2} f[i][j]k[x-i][y-j] \tag{2.2}$$

In Chapter 7 we use an equivalent characterisation of the discrete convolution for a fixed kernel $k$ as a rank $(2,2)-$tensor $A_{ij}^{kl}$ such that $A_{ij}^{kl} = A_{i+a,j+a}^{k+a,l+a}$ for all $i,j,k,l,a$.

#### 2.1.0.1 Motivating Example: Translational Equivariance in CNNs

A explicit example of a structural prior already found in neural architectures is found in CNNs. Specifically, a CNN is a type of neural network, generally used for image processing which uses as its base operation, the discrete convolution operation. This is a linear map meaning that a CNN is actually a type of multi-layer perceptron (MLP) implementing a high degree of weight sharing. Unlike an arbitrary linear map, CNNs have the property of translation-equivariance. This means that a translation of the input will only have the effect of translating the output. We demonstrate this below. First, let $I_a(f)(x) := f(x+a)$ be the translation operator by the vector $a \in \mathbb{R}^2$. Then

$$\begin{aligned} I_a(f) \star k(x) &= \int_{\mathbb{R}^2} I_a(f)(y)k(x-y)dy && (2.3)\\ &= \int_{\mathbb{R}^2} f(y+a)k(x-y)dy && (2.4)\\ &= \int_{\mathbb{R}^2} f(y)k(x-y+a)dy = I_a(f \star k)(x) && (2.5) \end{aligned}$$

In fact it can easily be shown that the most general bounded linear operator exhibiting translation equivariance is the convolution operator. Thus, a CNN can be recovered from an MLP from the application of no more than a simple equivariance condition. The effect of this one bias is that CNNs train much faster and perform far better than simple MLPs which lack this domain information.

[1] Often $\star$ denotes the correlation and $*$ is used for convolution but we will us $\star$ throughout

## 2.2 Dataset and Tasks

This dissertation is organised in four content chapters which describe different approaches to exploring the same goal of building invariant architectures. In the first chapter we take at two simple approaches to making CNNs invariant to changes in contrast. The first of these is more local in nature and consists of making a minor change to the convolution operation so that it is invariant at each layer to action of the contrast group $\cong \mathbb{R}_{>0}$. The second is more in keeping with the other chapters and takes a more global approach. Here we make a small change so that each layer of the network is equivariant to changes in contrast. We test these results on CIFAR-10 and an augmented version where the testset has been "poisoned" by random variations in contrast. These results are compared with sensible baselines. We show that our network has the desired properties but at the expense of generalisation accuracy in the "unpoisoned" case.

In the next chapter we extend this work by looking at other groups which act on images, namely; brightness, rotation and reflection. Following the previous chapter we show mathematically how one can adapt a CNN so that it exhibits invariance for an arbitrary group which acts on the plane. As before, we construct these networks and compare their results with reasonable baseline architectures on augmented and unaugmented data. Once again we achieve the desired robustness. In the case of brightness we also observe an impressive boost in both learning speed and performance. In the cases of the other groups the robustness comes at the expense of average accuracy. With reference to lemma 5.3.1 for arbitrary groups we make the case that there is insufficient scope in an ordinary CNN to build in desired invariance without damaging accuracy.

In Chapter 3 we look to solve some of the issues discussed in the last chapter. We show mathematically that by changing the intermediate domains that one can achieve the desired robustness without compromising on accuracy for an arbitrary group. Much this is work follows similar lines to work done by Cohen and Welling (Cohen and Welling (2016)). Nevertheless the more rigorous mathematical treatment as discussed in notation and terminology (section 2.3) allows a greater generality of our work including rotation by angles other than $0^\circ, 90^\circ, 180^\circ, 270^\circ$. As before we compare our results to appropriate baselines on augmented and unaugmented versions of CIFAR-10. We show that we achieve robustness to augmentations but without the decline in accuracy on the unaugmented dataset as before. Our rotationally and reflectionally equivariant networks

outperform the baselines in all categories. We see drawbacks however, in the large increase in computation time.

The final chapter is a departure from the last three. Here we look at how one can theoretically use a form of $l2$-regularisation to impose approximate equivariance on a network. This is an extension of a paper by Yang et al. (2019) which uses regularisation to coerce *invariance* from a network. Unlike previous methods, adding more groups does not increase the computation cost of processing each epoch. Nevertheless, the memory requirement of this approach is enormous and acts as a significant barrier to implementation. We implement a cheaper, approximate version of the regulariser to a fully connected network for the translation group and show a corresponding boost in performance. Overall, this approximated regulariser does not scale when other groups are added. Overall we show that despite the elegant simplicity of the basic principle, the effort required to get results mean it is a poor method for achieving equivariance.

### 2.2.1 Dataset and Architectures

In all cases we will be using the CIFAR-10 dataset (Krizhevsky and Hinton (2009)). We conduct experiments in each chapter on this dataset and various augmented versions of it. These augmented version are created in situ by making random perturbations to elements of the test set. We call these datasets *poisoned*. For example, in chapter 5 we randomly rotate elements of the test set to investigate the robustness of classifiers to shifts in domain. This dataset has been chosen as it is a well-established baseline in image classification. In addition, the images of CIFAR10 ($32 \times 32$) are smaller in size when compared to datasets such as Mini-ImageNet. This means that one can run more experiments and try a larger array of ideas.

In almost all cases and unless otherwise specified we use the same architecture. This a simple structure consisting of four convolutional layers followed by three linear layers with leaky relu non-linearities. The kernels are of size ($3 \times 3$) and we use a stride of 2. The channel numbers are $64, 128, 256, 512$. The linear layers map from 512 to 256 to 128 to 10. Most hyperparameters remain constant throughout including the weight decay constant ($5e-4$), batch size (128) and multistep learning rate annealer milestones $(60, 120, 160)$. In most experiments, particularly Chapter 7 where there are multiple regularisers, we do brief hyperparameter and final performance is cited using

these values. We endeavour to repeat all experiments 5 times however this was not possible in all cases; in Chapter 6 experiments have been repeated 3 times each. Results are given as an average over runs together with a standard deviation. We avoid using augmentation schema unless explicitly stated. This is due to concerns that this might cause undesired interactions.

## 2.3 Notation and Terminology

For the purposes of processing, images are generally stored and manipulated using arrays or tensors. This is a consequence of the method of capture wherein digital cameras project light onto a rectangular array of pixels. Nevertheless, it is useful to remember that this data structure is only a discrete representation of an image. For the purposes of this dissertation, we model the space of one-channel images as the set of functions $f : \mathbb{R}^2 \to \mathbb{R}$ which are compactly supported and bounded above. Throughout the following we will use $H(\mathbb{R}^2)$ to denote this set. The function $f$ should be interpreted as the intensity of incident light at any point; a camera integrates the light from a region into a pixel value. As we will see, there are advantages to operating within this space rather than doing what is common within existing literature and treating an image as an array. In particular this interpretation means that our treatments are agnostic to resolution. In Chapter 6 we use the notation $H(S)$ to denote the real-valued, bounded, compactly supported functions on an arbitrary set $S$. In most cases during derivations we totally ignore the channel number $c$. This is purely for the sake of clarity and results have been checked to hold for $c > 1$.

Following on from this, it is useful also to take the continuous view of a neural network. Neural networks are generally viewed, both from the perspective of the computer and within academic literature as sequences of finite linear maps applied to arrays. However this is a simply a discrete approximation. In an idealised reality we are looking at sequences of operators between function spaces $F : H(\mathbb{R}^2)^{c_{in}} \to H(\mathbb{R}^2)^{c_{out}}$. The continuous analogue of a fully-connected map is the integral transform $\int_{\mathbb{R}^2} f(y)k(x,y)dy$. Furthermore, the convolution operator, so often used within neural architectures, is a discrete approximation of the continuous version $\star : H(\mathbb{R}^2) \to H(\mathbb{R}^2)$ defined $f \star k(x) := \int_{\mathbb{R}^n} f(y)k(x-y)$.

This discussion on the formalism in our representations may seem like needless pedantry. However it has at least two advantages of some significance: Firstly it allows one to keep track of the domains. This helps one reflect on the meaning of purpose of the various operations within a network. It also yields the following razor which can be helpful when designing new changes to neural architectures *Does what I'm doing have an analogue in continuous space?* If the answer to this is no then there is a good chance that the designed change constitutes more of an engineering hack than anything principled.

# Chapter 3

# Literature Review

The idea of using group invariances to construct neural networks is not novel and there is a sizeable body of literature on this topic. Much of this work already indicates that invariances can be exploited to improve performance, reduce dependence of big-data and improve robustness. As mentioned before, data augmentation is already a widespread and highly effective way in which known invariances are used to coax networks into becoming robust (Fawzi et al. (2016)). Dumont et al. showed that rotationally equivariant networks are less vunerable to all types of geometric adversarial attacks than non-equivariant equivalents Dumont et al. (2018).

In their 2015 paper *Understanding image representations by measuring their equivariance and equivalence* (Lenc and Vedaldi (2014)) Lenc and Vedaldi showed that AlexNetKrizhevsky et al. (2012) automatically learns equivariance to reflection in the $y-$axis. This suggests that equivariance is a desirable property for a network to have. Further evidence of this is the convolutional neural network. As previously mentioned, a CNN is the most general MLP which is translation equivariant. The addition of this one constraint means that CNNs learn faster, use fewer parameters and produce better results than the fully-connected equivalent. It is reasonable then to imagine that if one can take a similar approach for other known symmetries that we might observe commensurate improvements. If fact publications in recent years support this hypothesis. In 2015 Kaggle hosted a competition called the *Galaxy Challenge* to classify galaxies based upon their morphology. The winning model Dieleman et al. (2015) used the rotational symmetry of the objects being classified to achieve generalisation accuracy of 99%. More recently Winkels and Cohen used the three dimensional isometric symmetries of Pulmonary nodules to produce state-of-the art medical detection of lung nodulesWinkels

and Cohen (2018). This was a good exhibition of how the use of sensible inductive biases can compensate when operating in a data poor domain.

As humans we are able to decouple the pose of an object from its class. We can recognise a face even if it is far away, tilted and in poor lighting conditions. We understand that that the orientation, position and lighting are separate operators which have been applied to the fundamental object. This disentanglement is reflected in our language where we use nouns for objects and adjectives to denote their group-invariant operators. *The brown cat was smiling and wearing a hat.* In this sentence the cat is the object and the groups which act upon it are its colour, smilingness and hatness. Given an extensive use of biology to inspire learning models, the group equivariance of human language gives a good heuristic justification for building group equivariance into neural architectures. This idea of decomposing the presence and pose of an object using layerwise equivariance is used by Geoffrey Hinton in the capsule architecture is precisely this way (Hinton et al. (2011)).

There is extensive other work looking at equivariance in neural networks, most notably the work of Max Welling and Taco Cohen. In their paper *Group Equivariant Neural Networks* Cohen and Welling (2016) they produce a fairly general method for building equivariant neural networks. This is validated by empirical results for the group of rotations $\{0°, 90°, 180°, 270°\}$ and reflection in the $y-$axis. In two other papers from 2017/2019 respectively Cohen et al. produce a convolutional network for the sphere which accounts for the non-trivial holonomy and imposes rotational equivariance to produce exceptional results of climate pattern segmentation Cohen et al. (2019) Cohen et al. (2017). Scattering Networks Bruna and Mallat (2012) are another architecture, this time designed using the wavelet transform, which construct representations which are equivariant to translation and robust to other deformations.

Spatial transformer networks Jaderberg et al. (2015) are another impactful example of networks which use symmetries to boost performance. The network implements a learnable differentiable module to achieve invariance to various group operations including scale, translation and rotation. Other work in this area includes but is not limited to methods which achieve invariance by using reinforcement learning to build a bank of transformed filters Stollenga et al. (2014) and scale-invariant convolutional networksLiu et al. (2019). Additionally, Kondor et al. have produced a firm mathematical grounding

for this work, constructing general group-equivariant operators on arbitrary manifolds and graphs Maron et al. (2019).

## 3.1 Assessment

As we have seen there is already a fairly comprehensive body of work on the topic of group equivariant/invariant neural networks. Nevertheless there is still sufficient scope for expansion, generalisation and re-synthesis. With the notable exception of the work by Kondor et al. in which we see that there is still room to form a firmer mathematical grounding (Maron et al. (2019)). In almost all cases we see continued treatment of images as vectors in $\mathbb{R}^N$ rather than functions on the plane (Cohen and Welling (2016) Bruna and Mallat (2012) Bao and Song (2019) Jaderberg et al. (2015) Cohen et al. (2019) Cohen et al. (2017)) [1] It is our belief that this damages the generality of the work since many groups which act on the space of images can be observed when this perspective: A good example of this is the group of rotations by angles other than multiples of 90°. We also believe that this makes it harder to begin to construct a coherent theory. Following from this, there is also a small degree of overstatement about the generality of some of this work: In almost every case the authors have only considered groups which act on images via their action on the plane (Bao and Song (2019) Cohen and Welling (2016)). This is an important family of transformations which includes rescalings, rotations and reflections. However it doesn't include the vast majority of groups which act on images including brightness and contrast.

[1] Once again Maron et al. (2019) is an exception to this.

# Chapter 4

# A First Look at Invariance

This first chapter is a first simple look at building a group invariant network. We look two methods of making a CNN invariant to shifts in contrast. Here we identify contrast with the multiplicative group $(\mathbb{R}^\star, \times)$. Changes in contrast are then given by the action of this group on the space of images defined $(\lambda, f) \mapsto \lambda f$. Our desire to achieve contrast invariance in the case of object classification is based upon the belief that the class designation of an image should remain unaffected by a rescaling of the lighting conditions. As it stands CNNs do not, a priori, possess this invariance. The reason for this is that the discrete convolution operation, which forms the fundamental unit of a CNN, does not have this invariance. Consequently the first approach in this chapter makes a minor alteration to the convolution so that it is contrast invariant. In the second case we use the contrast *equivariance* of the convolution to impose contrast invariance in a less local manner.

## 4.1 Layerwise Invariance

**Note:** In the first set of experiments we redefined the convolution so that it was invariant to changes in scale. Unlike in all other sections we will use discrete notation for the purposes of clarity. We use $x$ to denote the input and $k$ to denote the kernel.

The convolution of an input with the kernel is formed of a series of dot products $k \cdot x_1$ computed between the kernel vector $k$ and some region $x_1$ of the input. We refined the convolution so that we divide the output of each dot product by the norm of the region upon which $k$ attends.

The effect of this is that the normalised convolution is now invariant to rescaling: $(\lambda x) \star_n k = x \star_n k$ for all $\lambda \neq 0$. Our motivation for this is as follows: Suppose that our kernel is of size 3 with entries $k = (1,2,1)$. Next suppose that we have two input vectors $x_1 = (1,2,1)$ and $x_2 = (4,2,4)$. The dot product of this kernel with these vectors $k \cdot x_1 = 6$ and $k \cdot x_2 = 12$. Notice that even though the vector $x_1$ is precisely equal to the kernel, the output of $k \cdot x_1 < k \cdot x_2$. This is despite the fact $x_2$ barely shares any resemblance to the kernel. The effect of our change is to bias convolutions more towards shape and away from intensity.

The first set of experiments we carried out was a comparison of the performance of the normed convolution with the regular convolution on the CIFAR10 dataset Krizhevsky and Hinton (2009). The architecture that we used is as outlined in the dataset and architecture section section 2.2.1. In the first set of experiments we used regular convolutional layers and in the second, we replaced each convolution with the normed version. We call the latter architecture NCNN to distinguish it from the standard CNN. In the first set of experiments we tested the CNN and NCNN on an unadulterated version of CIFAR10. However, in the second set we evaluated performance on a *poisoned* version of CIFAR10 where we randomly altered the contrast of elements in the testset. In each case we used identical learning rates and random seeds. The results are shown in table 4.1.

**Note:** The justification we gave for dividing by the norm after each dot-product was that this prevents the convolution from being biased by intensity. One might reasonably apply this reasoning to the kernel function and conclude that the convolution should be also invariant to rescaling of $k$; $x \star (\lambda k) = x \star k$. In this case, the schema being used is that of cosine distance (4.1), a well-known normalisation procedure that has been shown to outperform a wide variety of other batch and layer norms Luo et al. (2017). The argument usually presented in favour of normalising procedures such as batch norm or cosine similarity is that they nullify the internal covariate shift that occurs during training in networks Ioffe and Szegedy (2015)Luo et al. (2017). It is worthy of remark that one can derive these same procedures from the perspective of group invariance.

$$\frac{x \cdot y}{||x||_2 ||y||_2} \tag{4.1}$$

## 4.2 Invariance through Equivariance

In the previous section we made an alteration to the convolution so that the network was contrast invariant at each layer. Our next approach is to look at imposing contrast invariance across the length of a network. To do this we take a closer look at the convolution. We have already seen that the convolution is not contrast invariant. However, unlike in our first experiments where we tried to change this, we can instead take a different approach by using the contrast-*equivariance* of the convolution. That is, the property that $(\lambda x) \star k = \lambda(x \star k)\ \forall \lambda \neq 0$.

In the case of a network made up of a sequence of convolutions $k_1, k_2, \ldots k_m$, we have $k_n \star (k_{n-1} \star \ldots (k_1 \star \lambda x)) \ldots) = k_n \star (k_{n-1} \star \ldots \lambda(k_1 \star x)) \ldots) = \ldots = \lambda k_n \star (k_{n-1} \star \ldots (k_1 \star x)) \ldots)$. Consequently, in order to induce contrast invariance over the entire network, only the last layer needs to exhibit invariance. Generally the last layer will consist of a fully-connected map followed by a softmax. That is

$$y_i = \frac{\exp(-w_i \cdot \vec{x})}{\sum_i \exp(-w_i \cdot \vec{x})} \tag{4.2}$$

It is easy to check that this is not a contrast invariant function. Consequently we change this so that our final layer is instead

$$y_i = \frac{\exp(-w_i \cdot \frac{\vec{x}}{|x|_2})}{\sum_i \exp(-w_i \cdot \frac{\vec{x}}{|x|_2})} \tag{4.3}$$

In order for the network to exhibit equivariance we also require that the non-linearities are contrast equivariant; $\sigma(\lambda x) = \lambda \sigma(x)$. Whilst this does not hold for most sigmoid functions, we let $\sigma$ be a ReLU for which this holds. It is also worth remarking that for the entire network to be contrast invariant that there cannot be any bias terms on the convolutions. Consequently we alter the convolution layer so that it is of the form $k \star x + x$. Making these small alterations we construct a contrast invariant network which we label as CCNN. Following this we repeated the experiments described above in section 4.1 for this network on poisoned and unpoisoned versions of CIFAR10. These are shown in 4.1.

### 4.2.1 Experiments

Table 4.1 contains the comparisons of the three architectures. We see that the generalisation error is better for the regular CNN on the unpoisoned dataset than either of the

| | **Contrast** | | | |
|---|---|---|---|---|
| | CNN | NCNN | CCNN | CNN+BN |
| CIFAR10 | 76.7($\pm$0.31) | 74.6($\pm$0.32) | 72.4($\pm$0.24) | 80.4($\pm$0.18) |
| Poison CIFAR10 | 68.6($\pm$0.81) | 73.8($\pm$0.26) | 72.9($\pm$0.17) | 79.8($\pm$0.21) |

Table 4.1: Superior robustness for our architectures (CCNN and NCNN) but poor overall accuracy. All outperformed by a CNN using batch norm.

other methods. However, the CNN is highly fragile to augmentations in contrast and accuracy is badly damaged by the random contrast changes of the test set. In large part the decline in accuracy is explainable by the decision to use no bias term to preserve the equivariances and invariances of our networks.

We also looked at the effect of adding in a batch-norm after each convolution layer. This has been shown repeatedly to improve stability and generalisation of many architectures. It works by subtracting the batch mean from each datum and dividing by the batch standard deviation. The results of this can be seen in the last column of table 4.1. We can see that the impact of introducing batchnorm is to substantially improve the accuracy the CNN. In addition this batchnorm totally nullifies the effect of varying the contrast of the test set. In both categories the CNN with batch norm outperforms both of our models.

## 4.3 Conclusion

We implemented two methods to achieve contrast invariance in this section. In both cases our networks were invariant to contrast variations and performance on poisoned and unpoisoned dataset were almost identical for both models. Nevertheless overall performance was badly damaged in comparison to the CNN baseline. We saw additionally that batch normalisation, a simple and popular normalisation method, provided robustness against contrast variations and with a boost in generalisation accuracy. Our conclusions from this section are mixed. Whilst our models were uncompetitive in their performance, they demonstrate that only small, computationally inexpensive changes are needed to improve robustness. The success of batch norm also gives indication that as long as this is done in an appropriate way then the results can be impressive.

We also saw that the contrast equivariance (and hence robustness) of the networks depended upon our choice of non-linearity. In particular we saw that piecewise functions like *ReLU* (and its variants) would work, whilst other choices of sigmoid would not. Whilst this is outside of our scope, it is worth remarking whether the success of these ReLU-type non-linearities is related at all to their equivariance preserving properties (He et al. (2015)).

# Chapter 5

# Implementing Invariance in CNNs

We have seen in the previous chapter that one can make minor changes to existing architectures and succeed in imposing group equivariances which improve robustness. Additionally, this comes at little cost to the generalisation error on the original datasets. In this chapter we apply this idea to other groups, specifically, rotation, reflection and brightness. We see that this is highly effective method of improving robustness but comes at the expense of accuracy for more sophisticated groups. In the case of groups which act on the plane (eg rotation and reflection) we show that equivariance is obtainable by imposing a symmetry requirement on the kernel.

In the previous chapter we used discrete notation. That is to say, we represented images as vectors and we used the discrete version of the convolution. In this chapter we take the continuous view, modelling images $f$ (and kernels $k$) as functions as described in section 2.3. Correspondingly, we shift also from using the discrete to the continuous convolution definition 2.1.3.

## 5.1 Brightness

The group associated with brightness is $(\mathbb{R},+)$ and the action upon the group on images $f \in H(\mathbb{R}^2)$ is given by $\lambda \circ (f)(x) := f(x) + \lambda$ for all $\lambda \in \mathbb{R}$. Therefore, we can write

that a convolution is brightness equivariant if and only if

$$(\lambda \circ f) \star k(x) = \int_{\mathbb{R}^2} (f(y) + \lambda) k(x-y) dy \tag{5.1}$$

$$= \int_{\mathbb{R}^2} f(y)k(x-y) + \lambda k(x-y) dy \tag{5.2}$$

$$= f \star k(x) + \lambda \int_{\mathbb{R}^2} k(y) dy \tag{5.3}$$

$$= \lambda \circ (f \star k)(x) + \lambda \Big( \int_{\mathbb{R}^2} k(y) dy - 1 \Big) \tag{5.4}$$

Consequently, the convolution operation is brightness equivariant if and only if the kernel integrates to 1. It is trivial to see that this result holds true even when there is a bias term. However, whilst the convolution is equivariant under this condition we run into a probably vis-a-vis the non-linearity. In particular there are no non-trivial brightness equivariant non-linearities. We formalise this in the lemma below.

**Proposition 5.1.1.** *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be bijective continuous function such that $\sigma(x+\lambda) = \sigma(x) + \lambda$ for all $x, \lambda \in \mathbb{R}$. Then $\sigma(x) = x + c$ for some $c \in \mathbb{R}$*

*Proof.* Let $x = -\lambda$ then $\sigma(0) = \sigma(-\lambda) + \lambda \iff \sigma(x) = \sigma(0) + x \quad \forall x \in \mathbb{R}$. The result follows by setting $c := \sigma(0)$ □

**Remark:** It is worth remarking that in order for a function to exhibit brightness equivariance it is only necessary that the group $(\mathbb{R}, +)$ acts on both the domain and codomain and that the function is compatible with this action. Whilst the action of the group on the domain is given by definition (as $(\lambda, f) \to f + \lambda$), the precise details of this action on the codomain are not specified. Consequently the previous approach, where we have assumed that the group acts in the same way on the domain as the codomain, is not a priori justified. Now we show that this result holds with full generality.

**Lemma 5.1.1.** *Let $G = (\mathbb{R}, +)$ act on the domain via the action $(\lambda, f) \mapsto \lambda + f$. Let $G$ act on the codomain $H(\mathbb{R}^2)$ via some action $(\lambda, f) \mapsto F_\lambda(f)$. Let $\mathcal{F} : H(\mathbb{R}^2) \to H(\mathbb{R}^2)$ be a $G$-equivariance integral transform given by a non-zero convolution. Then it follows that $F_\lambda(f) = f + \lambda b$ for some $b \in \mathbb{R}$.*

*Proof.* By assumption, $\mathcal{F} : H(\mathbb{R}^2) \to H(\mathbb{R}^2)$ is of the form $\mathcal{F}(f) := f \star k$ for some non-zero kernel function in $H(\mathbb{R}^2)$. Hence $\mathcal{F}$ is $G$-equivariant if and only if $\mathcal{F}(f + \lambda) := (f \star k) + \lambda \star k = F_\lambda \circ (f \star k)$. If we let $h := f \star k$ then we see that this implies $h + \lambda b = F_\lambda \circ (h)$ for all $h \in H(\mathbb{R}^2)$ where $b := \int_{\mathbb{R}^2} k(y) dy$. Hence the action of $G$ on the codomain must be of the form $(\lambda, f) \mapsto f + \lambda b$. □

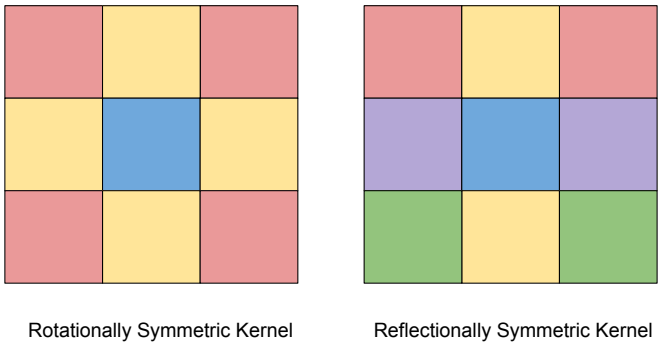


Figure 5.1: Rotationally and reflectionally symmetric $3 \times 3$ kernels

**Corollary 5.1.1.** *One cannot imbue a standard CNN with non-trivial brightness equivariance.*

*Proof.* We have seen in lemma 5.1.1 that a convolution is brightness equivariant if and only if the action of the group $G = (\mathbb{R}, +)$ is of the form $(\lambda, f) \mapsto f + \lambda b$. Consequently by proposition 5.1.1 we are done. □

#### 5.1.0.1 Brightness Invariance

We have seen in corollary 5.1.1 that under a general set of conditions that there are no non-trivial layerwise brightness equivariant CNNs. However we can produce a layerwise brightness *invariant* CNN. Setting $\int_{\mathbb{R}^2} K(x)dy = 0$ makes the convolution operation brightness invariant. We can then use a ReLU non-linearity as before. Additionally, unlike in the case of contrast, no changes need to be made to the final layer since the softmax is already brightness invariant. We construct a network by applying this idea to each of the convolution layers of a CNN; we call the resulting network a B-CNN. We then carry out a number of experiments comparing this to the usual CNN for an unaltered version of CIFAR10 and a version where the images in the test set are randomly varied in brightness. The results are shown in table 5.1. The precise architecture and dataset are as described in Dataset and Architecture section 2.2.1.

## 5.2 Reflection

The next group we look at is the group of order two formed from the identity and the horizontal flip operation. It is a unclear whether all image classifiers should have this as

an invariance. There is a good argument that human vision is not invariant to chirality: The letter *b* can be distinguished from the letter *d* and throughout the cultures of the world, movement from left to right is generally viewed as a sign of progression while right-left is viewed as retreat or decline (L Egizii et al. (2012)). However, the classes of CIFAR10 do exhibit this symmetry. Additionally, there is still potential advantage generally in producing a reflectionally *equivariant* network. Whether the network would also exhibit *invariance* would depend then only on the final layer. Let $g \in G$ represent a horizontal flip in the y-axis.

$$(g \circ f) \star k(x_1, x_2) = \int_{\mathbb{R}^2} f(-y_1, y_2) k((x_1, x_2) - (y_1, y_2)) dy \tag{5.5}$$

$$= \int_{\mathbb{R}^2} f(y_1, y_2) k((x_1, x_2) - (-y_1, y_2)) dy \tag{5.6}$$

$$g \circ (f \star k)(x) = \int_{\mathbb{R}^2} f(y_1, y_2) k((-x_1, x_2) - (y_1, y_2)) dy \tag{5.7}$$

Equating these two expressions we conclude that the convolution is reflectionally equivariant for all $f \in H(\mathbb{R}^2)$ precisely when the kernel used is reflectionally symmetric itself. Using this idea we construct a network (M-CNN) by imposing reflectional symmetry on the kernels of a CNN. As before we compare the M-CNN to a normal CNN on CIFAR10 and a version of CIFAR-10 (which we call *poison-CIFAR10*) where elements of the testset are randomly flipped horizontally. These results are then cross referenced with those obtained by training on augmented data obtained by flipping images in the $y-$axis. These results are shown in table 5.2.

## 5.3 Rotation

We now look at the case of the group of rotations $G \cong S^1$. Each element of the group can be identified with an angle $\theta \in [0, 2\pi)$ under the equivalence $a \sim b \iff a = b \mod (2\pi)$. The action here is defined $(g_\theta \circ f)(x) := f(Rot(\theta)\vec{x})$ for all $f \in H(\mathbb{R}^2)$. By following the lines of the reflectional case one can easily show that a convolution is

rotation equivariant if and only if the kernel is rotationally symmetric.

$$\text{Let } I_\theta(f)(x) := f(Rot(\theta)x) \tag{5.8}$$

$$\text{Then } I_\theta(f) \star k(x) = \int\int_{\mathbb{R}^2} f(Rot(\theta)y)k(x-y)dx \tag{5.9}$$

$$= \int\int_{\mathbb{R}^2} f(y)k(x - Rot(-\theta)y)dxdy \tag{5.10}$$

$$= \int\int_{\mathbb{R}^2} f(y)k(Rot(-\theta)(Rot(\theta)x - y))dxdy \tag{5.11}$$

$$\text{So then} \quad 0 = I_\theta(f) \star k(x) - I_\theta(f \star k)(x) \tag{5.12}$$

$$= \int\int_{\mathbb{R}^2} f(y)\Big(k\big(Rot(-\theta)(Rot(\theta)x - y)\big) - k(Rot(\theta)x - y)\Big) \tag{5.13}$$

$$\iff k\big(Rot(-\theta)(Rot(\theta)x - y)\big) = k\big(Rot(\theta)x - y\big) \quad \forall\theta \in \mathbb{R} \tag{5.14}$$

$$\iff k\big(Rot(\theta)x\big) = k\big(x\big) \quad \forall\theta \in \mathbb{R}, x \in \mathbb{R}^2 \tag{5.15}$$

As in the case of reflection we implement this by taking a CNN and altering it so that the kernels of each convolution layer were rotationally symmetric (by mutiples of 90°) fig. 5.1. We call the resultant network an R-CNN. We then compared the performance of this network with a normal CNN as in the case of reflection. We test the accuracy on an unaltered version of CIFAR10 and a version where elements of the testset have be randomly rotated by angles $\in \{0°, 90°, 180°, 270°\}$. As before this is cross-compared with results obtained by training on data augmented by random rotation. The results of these experiments are in table 5.3. The architecture is as described in section 2.2.1. The effect on the four convolution layers is reduce the dimensionality down from $32 \times 32$ to $1 \times 1$ before being passed into the fully connected layers. The effect of this is that our rotationally-equivariant network becomes rotationally invariant overall.

Having looked at three groups (brightness, reflection and rotation) we generalise this for a general group which acts on $H(\mathbb{R}^2)$ via its action on the plane. lemma 5.3.1 gives us the condition which must hold on the kernel function in order for the convolution operation to have $G-$equivariance.

**Lemma 5.3.1.** *Let G be some group which acts smoothly on $\mathbb{R}^2$. Let G act on $H(\mathbb{R}^2)$ via the action $(g \circ f)(x) := f(g \circ x)$. Then the convolution is $G-$equivariant if and only if the kernel has the following property:*

$$k(g \circ x)|det(Jac(g))| = k(x) \quad \forall x \in \mathbb{R}^2, g \in G \tag{5.16}$$

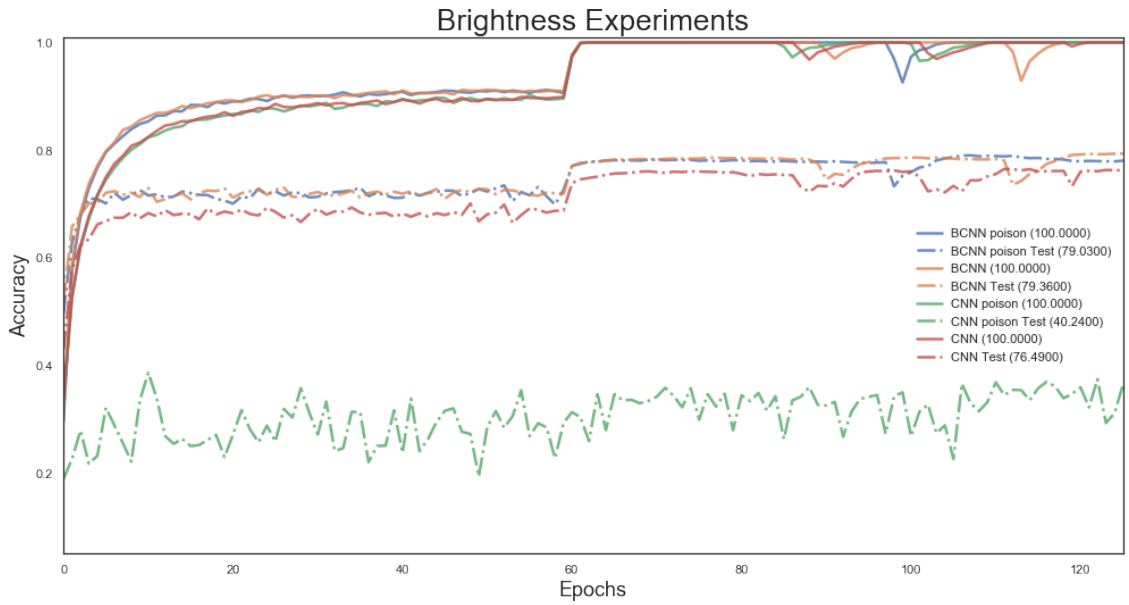


Figure 5.2: Faster training, greater robustness and better results for BCNN versus standard CNN

*$Jac(g)$ denotes the Jacobian one would obtain from change of variables $y' = g \circ y$.*

*Proof.* The proof of the above follows the exact lines of the rotational case. □

**Example 5.3.1** (General Linear Group)**.** Let $G = GL_2(\mathbb{R})$; the set of 2 by 2 matrices over $\mathbb{R}$ with non-zero determinant. In order for a convolution to exhibit this group equivariance we require that $k(Ax)det(A) = k(x)$ for all $A, x$. However, this can only hold where $k(x) = 0$ almost everywhere.

## 5.4 Experiments

The results of the brightness experiments table 5.1 show that the brightness equivariant network (B-CNN) outperforms the usual CNN both in terms of robustness and general accuracy. The B-CNN shows almost totally immunity to variations in brightness of the test set whereas this has an enormous impact of the efficacy of the standard CNN. In addition to this, the B-CNN also trained faster for the same choice of hyperparameters as the CNN fig. 5.2.

In contrast to the positive results for brightness, the results of the M-CNN (reflection) are fairly poor (table 5.2). The accuracy on both the poisoned and unpoisoned test sets are inferior to the CNN. What is notable is that the standard CNN is already fairly

| **Brightness** | | |
|---|---|---|
| | CNN | B-CNN |
| CIFAR10 | 76.7($\pm$0.31) | 79.4($\pm$0.25) |
| Poison CIFAR10 | 40.2($\pm$1.18) | 79.0($\pm$0.24) |

Table 5.1: Superior performance of Brightness invariant CNN for poisoned and unpoisoned datasets

| **Reflection** | | | | |
|---|---|---|---|---|
| | CNN | M-CNN | CNN+aug | M-CNN+aug |
| CIFAR10 | 76.7($\pm$0.31) | 71.8 | 81.7 | 73.2 |
| Poison CIFAR10 | 76.0($\pm$0.28) | 71.1 | 81.6 | 73.1 |

Table 5.2: Performance of CNN with reflectionally symmetric kernels (M-CNN) against a standard CNN. Performance compared on poisoned and unpoisoned test sets and with and without augmentation during training.

robust to horizontal flipping. This is consistent with existing literature we shows that convolutional architectures can learn reflectional equivariance automatically Lenc and Vedaldi (2014).

The results for the group of rotations (R-CNN) show complete robustness to rotations of the test set however this comes at the expense of a large decline in overall accuracy. With reference to table 5.3, we see that the results of the R-CNN are comfortably outperformed a scheme where one trains a CNN on rotationally augmented data. These poor results are consistent with other literature which shows that imposing symmetry requirements damages accuracy (Dudar and Semenov (2018)).

We briefly ran an additional experiment on the CNN which investigated whether this architecture had learned reflectional equivariance. To do this we measured the extent to which the kernels exhibited reflectional symmetry throughout the training process. Our metric is calculated by subtracting the kernel from a horizontally flipped version of itself and computing the $l2$-norm. We then divided this by the $l2$-norm of the kernel to make this metric agnostic to the magnitude of the values in the kernel. We recorded this value at the beginning of training and at the end. Overall we found a minor but statistically increase in reflectional symmetry.

| Rotation | | | | |
|---|---|---|---|---|
| | CNN | R-CNN | CNN+aug | R-CNN+aug |
| CIFAR10 | 77.0 | 55.0 | 74.6 | 57.1 |
| Poison CIFAR10 | 44.9 | 54.6 | 74.7 | 56.6 |

Table 5.3: CNN compared to CNN with rotationally symmetric kernels (R-CNN). Greater robustness of R-CNN to poisoning of test-set at the expense of accuracy. R-CNN outperformed by use of rotational augmentation.

## 5.5 Conclusion

The results of this section show that by making simple changes to a CNN that we can improve robustness to certain groups. In the case of brightness we showed that these alterations had the additional advantage of boosting accuracy generally. It is our belief that this boost in performance is in excess of that which might be explained by regularisation effects. Nevertheless, more experiments would be required to be certain. In the case of reflection our alterations provided no benefit in any category. Unlike Lenc and Vedaldi (2014) we were not able to show that the network had learned reflectional equivariance. In the case of rotation the changes we implemented heavily damaged accuracy but lead to an increase in robustness. Overall however the performance was sufficiently low that this was a poor trade-off.

In example 5.3.1 we saw that there are no non-trivial convolutions which have general-linear equivariance. This is despite the fact that the general linear group is a fairly basic example of group. Following from these observations, we conclude that this naive method of imposing coarse symmetry restrictions on the kernel is an inadequate method for achieving equivariance within neural networks. In the next chapter we look at a broader method for imbuing a CNN with group invariance.

# Chapter 6

# Structural Approaches to Equivariance

## 6.1 Solutions to the Equivariance Problem

In the previous chapter we concluded that in order to structurally inbuild group invariance and hence robustness, this must usually come at the expense of performance. This reasoning was built upon empirical evidence and lemma 5.3.1. However, all of these conclusions are predicated on a number of assumptions. The first of these assumptions is that the operators that form the network are linear. One solution to this problem is to look for non-linear operators with the desired properties to take the roll of the fundamental operation in neural networks (Masci et al., 2012). This is an extraordinarily difficult task when one remembers that we are limited also by computational feasibility. The natural generalisation is to look at bilinear maps (Soleymani et al., 2018). However, even bilinear functions $f : \mathbb{R}^n \to \mathbb{R}^m$ defined $x \mapsto x^T Ax + Bx + C$ (Where $A$ is rank 3 tensor) increases the number of parameters by a factor of order $n$ in the one dimensional case. This corresponds to a factor $O(wh)$ in the case of a *CNN* where $w, h$ are the width and heights of the feature maps at each layer.

A second assumption made in the previous chapter was that the domains represented by each layer of the network were $H(\mathbb{R}^2)^c$ (where $c$ is the number of channels). Consequently we only looked at operators $F : H(\mathbb{R}^2) \to H(\mathbb{R}^2)$. In this chapter we weaken this assumption allowing a broader family of linear operators. Our third and final assumption made implicitly in the last chapter was that a group must act identically upon the domain and co-domain in order to be equivariant. This overly literal interpretation of group equivariance is not uncommon in the literature (Zhang, 2019). With reference to definition 2.1.2, a function is group equivariant as long as the group

acts on the domain and co-domain and the function is compatible with these actions. In particular, the group of rotations $S^1$ doesn't have to act on the co-domain through rotation. Whilst the work in this chapter is not totally general, we endeavour to look at other forms of group actions.

**Example 6.1.1.** To demonstrate the points that we have just made, let us return to the case of brightness ($G \cong (\mathbb{R},+)$). We saw in the last chapter that a convolution $F : H(\mathbb{R}^2) \to H(\mathbb{R}^2)$ is brightness *invariant* if and only if the kernel integrates to 0. Now change the co-domain from $H(\mathbb{R}^2)$ to $H(\mathbb{R}^2) \times \mathbb{R}$ and consider the map $F : H(\mathbb{R}^2) \to H(\mathbb{R}^2) \times \mathbb{R}$ defined $F(f)(x,\lambda) = \left(\int_{\mathbb{R}^2} f(y)k(x-y)dy, \int_{\mathbb{R}^2} f(y)dy\right)$ where $\int_{\mathbb{R}^2} k(y)dy = 0$. Let $G$ act on the codomain through the action $(\lambda,(f,\mu))(x) \mapsto (f(x),\lambda+\mu)$. Then it trivial to check that the map $F$ is non-trivially $G$-equivariant even though the kernel integrates to 0.

**Note:** The purpose of the example above is simply to demonstrate that once we weaken the requirement that each of our domains is isomorhpic to $H(\mathbb{R})^c$, we are much less limited in how we can go about establishing equivariant networks.

Generally speaking, neural networks are composed of two building blocks; linear maps and non-linearities. In order for a network to exhibit $G$-equivariance overall, both of these must be $G$-equivariant. Before we proceed with the constructions in this chapter we introduce the following lemma which tell us that we do not have to worry about the non-linearities for a large family of groups $G$.

**Lemma 6.1.1** (Equivariance for Non-linearities)**.** *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be an arbitrary continuous bijective function. (Typically in neural networks $\sigma$ will be the ReLU function $ReLU(x) := max(0,x)$). Let $G$ be a group which acts smoothly on a set $S$. Furthermore, let $R_g : H(S) \to H(S)$ be a family of continuous operators parameterised by $G$ with the property that $R_g(f)(x) = f(gx)$ for all $g \in G$, $f \in H(S)$. Finally, suppose we have some continuous linear operator $F : H(S) \to H(S)$ where $F \circ R_g = R_g \circ F$, $\forall g \in G$. Then it follows that*

$$\sigma \circ F \circ R_g(f)(x) = R_g \circ \sigma \circ F(f)(x) \quad \forall g \in G, f \in H(S), x \in S \tag{6.1}$$

*Proof.* By the equivariance assumption on $F$ and associativity of composition,

$$\sigma \circ F \circ R_g(f)(x) = \sigma \circ (F \circ R_g)(f)(x) = \sigma \circ (R_g \circ F)(f)(x) \tag{6.2}$$

$$= \sigma \circ R_g \circ (F(f)(x)) = \sigma \circ (F(f)(gx)) = (\sigma \circ F(f)(gx)) \tag{6.3}$$

$$= R_g \circ (\sigma \circ F(f)(x)) = R \circ \sigma \circ F(f)(x) \tag{6.4}$$

□

In essence the above lemma tells us that if each of the linear maps of our network are $G-$equivariant and the action of $G$ on $H(S)$ is defined via the action of $G$ on $S$ then each of our non-linear layers are $G-$equivariant. This is very helpful lemma since it tell us that as long as the linear portion of each layer is $G$-equivariant then the whole network will be too for a large family of groups $G$. Examples of such groups include the group of rotations, translations and changes of scale and any group of the form $G =< \phi >$ where $\phi : S \to S$ is a homeomorphism on the set $S$.

### 6.1.1 Reflectionally Equivariant CNN

In this section we show how one can construct a reflectionally equivariant CNN. We then prove that it has the desired properties. For this small subsection we stick with the discrete case wherein an image is treated as a vector $\in \mathbb{R}^N$ rather than a function $\in H(\mathbb{R}^2)$. This is so the derivation gives a recipe for how this could be implemented in code.

Most of the structure of the reflectionally equivariant network (which we will refer to as MirrorNet fig. 6.1) is identical to that of a normal CNN with two changes: One change is the alteration that we make to the first operator in the network and the second is the alteration that we make to all other *intermediate* operators in the network. A simple depiction of this architecture in the case of $l = 2$ can be seen in fig. 6.1. In the first layer of the network we convolve the input with a kernel $k$ and then also with the horizontally flipped version of this kernel which we store in a separate dimension. The intermediate operators also has two parts. Firstly we convolve over these pairs tensors with a kernel $k1, k2$. After this we then swap round the kernels and flip them in the $y-$axis and convolve again. As can be seen in fig. 6.1, the effect of inputting a reflected image is that the hidden representations of the image are reflected and then permuted so

that the network as a whole is equivariant.

We can express this using matrix form. The first map in the network $A^{(1)} = (A_1^{(1)}, A_2^{(1)}) : \mathbb{R}^{n\times n} \to \mathbb{R}^{m\times m\times 2}$ is restricted so that $A_1^{(1)}$ is a convolution and $A_2^{(1)} := R_{ref} A_1^{(1)} R_{ref}$; where $R_{ref}$ is the reflection matrix. The latter operators are of the form $A^{(i)} = \begin{bmatrix} A_{11}^{(i)} & A_{12}^{(i)} \\ A_{21}^{(i)} & A_{22}^{(i)} \end{bmatrix} : \mathbb{R}^{m_1\times m_1\times 2} \to \mathbb{R}^{m_2\times m_2\times 2}$ where $A_{11}^{(i)}$ and $A_{12}^{(i)}$ are convolutions, $A_{11}^{(i)} = R_{ref} A_{22}^{(i)} R_{ref}$ and $A_{12}^{(i)} = R_{ref} A_{21}^{(i)} R_{ref}$.

**Theorem 6.1.1** (MirrorNet is reflectionally equivariant)**.** *The network defined above exhibits reflectional equivariance.*

*Proof.* We denote by $R_{ref}$ the reflection operator in a given domain and by $S$ the cyclic shift operator. That is $S : \mathbb{R}^{n^2\times 2} \to \mathbb{R}^{m^2\times 2}$ with $S(x,y) := (y,x) \quad \forall x,y \in \mathbb{R}^{n^2}$. Taking $i = 1$ we have $A_1^{(1)} R_{ref} = \begin{bmatrix} A_1^{(1)} \\ R_{ref} A_1^{(1)} R_{ref} \end{bmatrix} R_{ref} = \begin{bmatrix} A_1^{(1)} R_{ref} \\ R_{ref} A_1^{(1)} \end{bmatrix} = R_{ref} S A^{(1)}$. Next taking $1 < i < l$ we have $A^{(i)} R_{ref} S = \begin{bmatrix} A_{11}^{(i)} & A_{12}^{(i)} \\ A_{21}^{(i)} & A_{22}^{(i)} \end{bmatrix} R_{ref} S = \begin{bmatrix} A_{11}^{(i)} R_{ref} & A_{12}^{(i)} R_{ref} \\ A_{21}^{(i)} R_{ref} & A_{22}^{(i)} R_{ref} \end{bmatrix} S =$
$\begin{bmatrix} A_{21}^{(i)} R_{ref} & A_{22}^{(i)} R_{ref} \\ A_{11}^{(i)} R_{ref} & A_{12}^{(i)} R_{ref} \end{bmatrix} = \begin{bmatrix} R_{ref} A_{12}^{(i)} & R_{ref} A_{11}^{(i)} \\ A_{11}^{(i)} R_{ref} & A_{12}^{(i)} R_{ref} \end{bmatrix} = S \begin{bmatrix} R_{ref} A_{11}^{(i)} & R_{ref} A_{12}^{(i)} \\ A_{12}^{(i)} R_{ref} & A_{11}^{(i)} R_{ref} \end{bmatrix} =$
$R_{ref} S \begin{bmatrix} A_{11}^{(i)} & A_{12}^{(i)} \\ R_{ref} A_{12}^{(i)} R_{ref} & R_{ref} A_{11}^{(i)} R_{ref} \end{bmatrix} = R_{ref} S A^{(i)}$. Putting this together we have

$$A^{(l)} A^{(l-1)} \dots A^{(2)} A^{(1)} R_{refl} = A^{(l)} A^{(l-1)} \dots A^{(2)} (R_{ref} S A^{(1)}) = \dots = R_{ref} S (A^{(l)} A^{(l-1)} \dots A^{(1)}) \tag{6.5}$$

Hence by application of lemma 6.1.1 the entire network, including non-linearities, is reflectionally equivariant. □

#### 6.1.1.1 Generalising for all groups

We can use this approach to implement a broad range of group equivariances that we wish into a neural network. In this next section we show how to do this constructively for a finite group with a single generator. As in the reflectional case above we take the discrete view.

Let $R$ be a bijective linear map and let $G =< R >$ be the subgroup generated by $R$. That is $G = \{R^k | k \in \mathbb{Z}\}$. Suppose that $G$ is finite group so that $\exists g \in \mathbb{N}$ such that

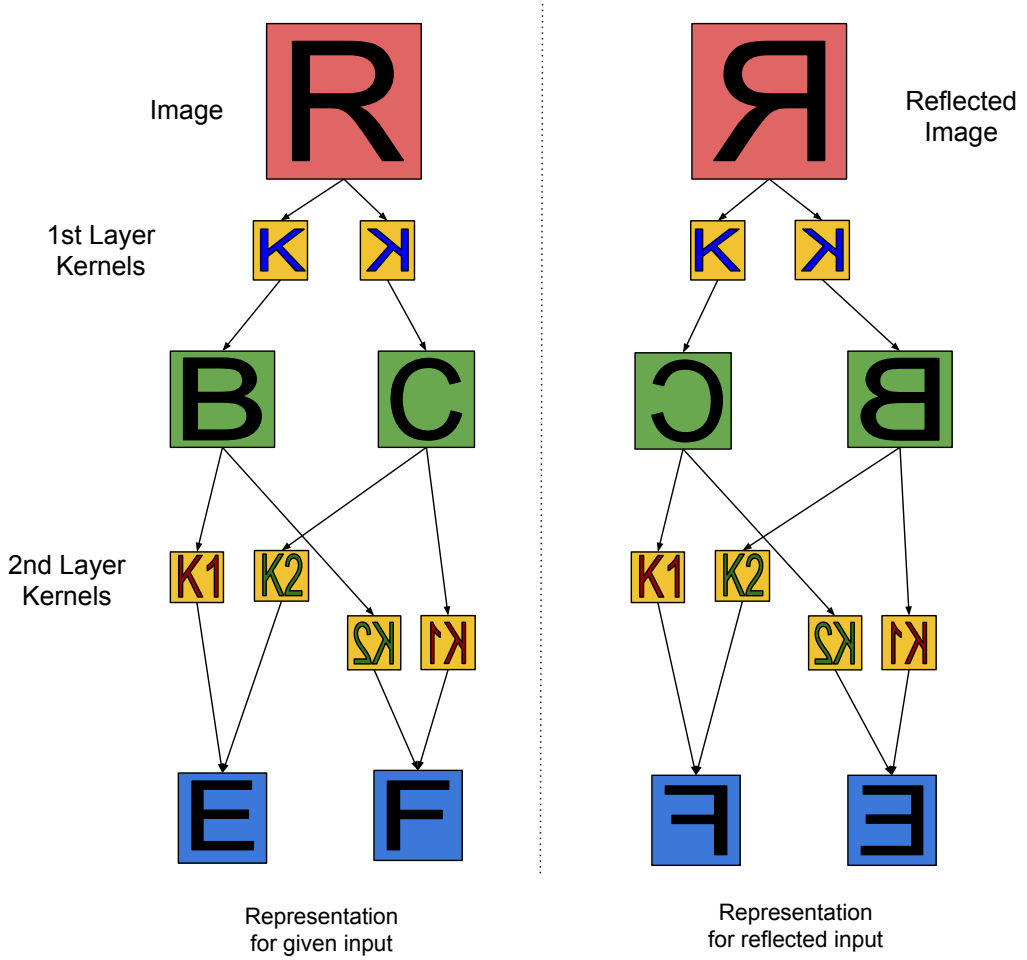


Figure 6.1: First two layers of MirrorNet network with one channel depicted. In the first layer we apply the kernel $K$ to the input $R$ to obtain $B$ and then flip $K$ and repeat to obtain $C$. The next layer convolves with the kernel $(K_1, K_2)$ to produce $E$. We then flip and swap the kernels and convolve again to produce $F$. The right portion of the image shows what the effect of flipping the input $R$ is. We see that reflection of the input permeates equivariantly through network.

$R^g = I$. Then we construct our neural architecture in the following way so that it exhibits $G-$equivariance alongside translational equivariance. Define the initial map $A^{(1)} = (A_1^{(1)}, A_2^{(1)}, \ldots, A_g^{(1)}) : \mathbb{R}^{n_1 \times n_1} \to \mathbb{R}^{n_2 \times n_2 \times g}$ so that $A_k^{(1)} = R^{g-k+1} A_1^{(1)} R^{k-1}$ where $A_1^{(1)}$ is a convolution. Then $A_1 R = (A_1^{(1)} R, R^{g-1} A_1^{(1)} R^2, \ldots\ RA_1^{(1)} R^g) = (A_1^{(1)} R, R^{g-1} A_1^{(1)} R^2, \ldots\ RA_1^{(1)}) = RSA_1$. For the intermediate layers in the network $1 < i < l$ we let

$$A^{(i)} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \ldots & A_{1g} \\ A_{21} & A_{22} & A_{23} & \ldots & A_{2g} \\ \ldots & & & & \\ A_{g1} & A_{g2} & A_{g3} & \ldots & A_{gg} \end{bmatrix} \tag{6.6}$$

We make the additional restriction that the submatrices $A_{ij}$ are convolutions and that $A_{ij} = R^{g-i+1} A_{kl} R^{i-1}$ whenever $(i+1) \mod g = k$ and $(j+1) \mod g = l$. So then

$$A_i RS = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \ldots & A_{1g} \\ R^{g-1} A_{1g} R & R^{g-1} A_{11} R & R^{g-1} A_{12} R & \ldots & R^{g-1} A_{1,g-1} R \\ \ldots & & & & \\ RA_{12} R^{g-1} & RA_{13} R^{g-1} & RA_{14} R^{g-1} & \ldots & RA_{11} R^{g-1} \end{bmatrix} RS = \tag{6.7}$$

$$\begin{bmatrix} RA_{12} & RA_{13} & RA_{14} & \ldots & RA_{11} \\ A_{11} R & A_{12} R & A_{13} R & \ldots & A_{1g} R \\ \ldots & & & & \\ R^2 A_{13} R^{g-1} & R^2 A_{14} R^{g-1} & R^2 A_{15} R^{g-1} & \ldots & R^2 A_{12} R^{g-1} \end{bmatrix} = RSA_i \tag{6.8}$$

Overall, even when we include a non-linearity between each layer, we have a neural network which has the desired equivariance built in on top of the translational equivariance of the convolution operation (lemma 6.1.1).

#### 6.1.1.2 Domains as products of Lie Groups

In the previous section we gave methods for how one can construct a $G-$equivariant CNN for a largely family of groups. In this next section we formalise what we have done in the last. Specifically we return to the continuous view of a neural network and give greater details about how theses recipes above have been derived. In the first part we just derive the condition which hold for each layer in order for the network to be $G$-equivariant.

**Proposition 6.1.1.** *Let $G$ be a Lie Group which acts on $H(\mathbb{R}^2)$. Suppose that we have a sequence of domains $D_1, D_2, \dots D_l$ where $D_1, D_l \cong H(\mathbb{R}^2)$ and $D_i \cong H(\mathbb{R}^2 \times G)$ for $1 < i < l$. Suppose that we construct an operator $\mathcal{F} := F_{l-1} \circ F_{l-2} \circ \dots F_1$ where $F_i : D_i \to D_{i+1}$ is given by an integral transform. Then $\mathcal{F}$ is $G-$equivariant and translation equivariant if for $1 < i < l$*

$$F_i(g \circ f) := \int_{\mathbb{R}^2 \times G} K(x-y,h,p) g \circ f(y,p) dy dp \tag{6.9}$$

$$= g \circ \left( \int_{\mathbb{R}^2 \times G} K(x-y,h,p) f(y,p) dy dp \right) =: g \circ (F_i(f)) \tag{6.10}$$

$$F_1(g \circ f) := \int_{\mathbb{R}^2} K(x-y,h) g \circ f(y) dy \tag{6.11}$$

$$= g \circ \left( \int_{\mathbb{R}^2} K(x-y,h) f(y) dy \right) =: g \circ (F_1(f)) \tag{6.12}$$

*Proof.* The proof of this follows from the definition of an integral transform the fact that a convolution is the most general bounded linear operator which commutes with translations. □

#### 6.1.1.3 Choice of action

In the proposition above we have left unspecified the details of the action of $G$ on $H(\mathbb{R}^2 \times G)$. The property of group-equivariance means that that our group $G$ acts on both the domain and codomain such that it commutes with the operator at each layer. However, these still leaves some choice in the exact form of the group action that we are interested. Whilst there are various other possible choices of action, we let $G$ act on the set $H(\mathbb{R}^2 \times G)$ via the action $(g \circ f)(x,h) := f(gx, gh)$ for this section.

**Note:** If $G$ is a group which acts on the plane $\mathbb{R}^2$, then $G$ acts on $H(\mathbb{R}^2)$ through the action defined $(g \circ f)(x) := f(g \circ x)$. This is also a group action since $(e \circ f)(x) = f(e \circ x) = f(x)$ and $g \circ (h \circ f)(x) = g \circ f(h \circ x) = f(g \circ (h \circ x)) = f(gh \circ x) = gh \circ f(x)$. For the following we will assume that when we talk about a group acting on $H(\mathbb{R}^2)$ that the action is of this type.

#### 6.1.1.4 The Initial Operator

Define the action of $G$ on $H(\mathbb{R}^2 \times G)$ as

$$(g \circ f)(x,h) := f(gx, gh) \tag{6.13}$$

So then, with reference to proposition 6.1.1 a $G-$*equivariant* operator from $H(\mathbb{R}^2)$ to $H(\mathbb{R}^2 \times G)$ is one where $\forall x \in \mathbb{R}^2, g,h \in G, f \in H(\mathbb{R}^2)$

$$(g \circ F(f))(x,h) = \int_{\mathbb{R}^2} K(gx-y,gh)f(y)dy \tag{6.14}$$

$$= \int_{\mathbb{R}^2} K(x-y,h)f(g \circ y)dy = F(g \circ f)(x,h) \tag{6.15}$$

If $G$ is a Lie Group and the action $\circ : G \times X \to X$ is a Lie Group action then $g : X \to X$ is a smooth bijective function for every $g \in G$. So then we can change coordinates $y' := g(y)$ so that the equality above can be re-written

$$\int_{\mathbb{R}^2} K(gx-y,gh)f(y)dy = \int_{\mathbb{R}^2} K(x-g^{-1}y,h)f(y)|det(J(g^{-1}))|dy \tag{6.16}$$

Where $det(J(g^{-1}))$ denotes the Jacobian matrix of the inverse of the group action. Since the Jacobian of the inverse of a transformation is the inverse of the Jacobian of a transformation and $det(A^{-1}) = det(A)^{-1}$ then we can rewrite this:

$$\int_{\mathbb{R}^2} f(y)\big(K(gx-y,gh)|det(J(g))| - K(x-g^{-1}y,h)\big)dy = 0 \tag{6.17}$$

Since this must hold for all functions $f \in H(\mathbb{R}^2)$ then this is equivalent to the statement that

$$K(gx-y,gh)|det(J(g))| = K(x-g^{-1}y,h) \quad \text{for all} \quad h \in G, x,y \in \mathbb{R}^2 \tag{6.18}$$

If we make the further assumption on the action that it is linear then we obtain the following equality:

$$K(gx,gh)|det(J(g))| = K(x,h) \quad \text{for all} \quad h \in G, x \in \mathbb{R}^2 \tag{6.19}$$

If we let $X_{rep} \subset \mathbb{R}^2 \times G$ be a set of orbit representatives of the action of $G$ on $\mathbb{R}^2 \times G$ defined $(g,(x,h)) = (g(x),gh)$ then $K$ is defined entirely by the values it takes on $X_{rep}$.

**Proposition 6.1.2.** *$X_{rep} := \mathbb{R}^2 \times \{e\}$ is a set of orbit representatives for this action.*

*Proof.* Let $(z,l)$ be an arbitrary point in $\mathbb{R}^2 \times G$. By setting $g = l$ we see that $(z,l)$ is in the same orbit as $(l^{-1}(z),e) \in X_{rep}$. Now we only need to show that none of the elements of $X_{rep}$ are in the same orbit. Let $(y,e),(x,e)$ be points in $X_{rep}$. Then $g \circ (x,e) = (gx,ge) = (y,e) \iff g = e \iff x = y$ and the result follows. □

Using proposition 6.1.2 we conclude that the inital operator of network is of the form.

$$F(f)(x,h) := \int_{\mathbb{R}^2} K(h^{-1}(x-y))|det(J(h))|^{-1} f(y)dy \tag{6.20}$$

Where $K : \mathbb{R}^2 \to \mathbb{R}^2$ is any choice of kernel function.

**Example 6.1.2** (Rotation)**.** Suppose that $G$ is the group of rotations of the plane. Using the derivation above and the fact that the determinant of the Jacobian in this case will be equal to one, we conclude the following form for the operator.

$$F(f)(x,\theta) := \int_{\mathbb{R}^2} K\big(Rot(-\theta)(x-y))\big) f(y) dy \tag{6.21}$$

Where $K : \mathbb{R}^2 \to \mathbb{R}^2$ is some kernel function. If we let $g_\theta$ be the set of functions parameterised by $\theta$ so that $g_\theta(x) := K(Rot(-\theta)x)$ then the operator $F$ can be expressed simply as $F(f)(x,\theta) = \int_{\mathbb{R}^2} K\big(Rot(-\theta)(x-y)) = \int_{\mathbb{R}^2} f(y) g_\theta(x-y) = f \star g_\theta(x)$. Consequently we can build our network in Pytorch using the inbuilt convolution operation on transformed kernels.

#### 6.1.1.5 Generalised Intermediate Operators

We have seen how one can define the first operator in the network so that it is expressive and exhibits $G$-equivariance eq. (6.20). With our choice for the action of $G$ on $H(\mathbb{R}^2 \times G)$ we generalise the above to the other intermediate operators in the networks.

Let the general form of the intermediate operator be as follows:

$$F(f)(x,h) := \int_{\mathbb{R}^2 \times G} K(x-y,h,l) f(y,l) dy dl \tag{6.22}$$

We make the additional requirement that this operator exhibits $G-equivariance$. As before we define the group action of $G$ on $H(G \times \mathbb{R}^2)$ by $(g \circ f)(x,h) := f(g(x), gh)$ This requirement is met if and only if

$$F(g \circ f)(x,h) = g \circ (F(f))(x,h) \tag{6.23}$$

$$\iff \int_{\mathbb{R}^2 \times} K(x-y,h,l) f(g(y), gl) dy dl \tag{6.24}$$

$$= \int_{\mathbb{R}^2 \times G} K(x-g^{-1}y, h, g^{-1}l) f(y,l) |det(J(g))|^{-1} dy dl \tag{6.25}$$

$$= \int_{\mathbb{R}^2 \times G} K(g(x)-y, gh, l) f(y,l) dy dl \quad \forall g,h \in G, x \in \mathbb{R}^2 \tag{6.26}$$

$$\iff |det(J(g))| K(g(x), gh, gl) = K(x,h,l) \tag{6.27}$$

We choose as a set of orbit representatives $X_{rep}$ of the action $G \times G \times G \times \mathbb{R}^2 \to G \times G \times \mathbb{R}^2$ as $X_{rep} = \mathbb{R}^2 \times \{id\} \times G$. To see that this is a set of orbit representatives, consider an arbitrary point $(x,g,b) \in \mathbb{R}^2 \times G \times G$, then this can be written as $g \circ (g^{-1}x, e, g^{-1}b)$. Now show that none of these points represent the same orbit: Suppose then that $(g(x), g, gb) = (y, e, a)$ then $g = e$ and thus $b = a$ and $x = y$. Then, the most general

operator from the specified domain to the specified codomain that exhibits the required equivariances is given below for some choice of function $K : \mathbb{R}^2 \times G \to \mathbb{R}$

$$G(f)(x,h) := \int_{\mathbb{R}^2 \times G} K(h^{-1}(x-y), h^{-1}l)|det(J(h))|f(y,l)dldy \tag{6.28}$$

**Example 6.1.3. The Intermediate Operator for Rotation** We have already seen the form that the first operator in the network will take in the case of the rotation group $G = S^1$. By applying eq. (6.28) using the fact that the jacobian equals one, then the intermediate operator for this group will be of the following form.

$$G(f)(x,\theta) := \int_{\mathbb{R}^2 \times [0,2\pi]} K\big(Rot(-\theta)(x-y), \phi - \theta\big) f(y,\phi) d\phi dy \tag{6.29}$$

We have shown through our derivations of eq. (6.20), eq. (6.28) how one can build a neural network which is equivariant to any $G$ which acts on the plane. Whilst earlier (theorem 6.1.1) we gave certain architecture and showed that it was equivariant, we have now provided the method by which they were constructed. For instance, one is able to derive the architecture of *RotateNet* by taking a discrete view of (example 6.1.2, example 6.1.3).

## 6.2 Experiments

In this section we implemented the ideas from the previous section above evaluate to create a couple of equivariant convolutional networks. In the first set of experiments we look at a network formed so that it has structural reflection equivariance. We call this *MirrorNet* (fig. 6.1). In the second case we look at two versions of a rotationally equivariant network; one for the group generated by rotation by $45^\circ$ and the other by $90^\circ$ called *RotateNet8* and *RotateNet4* repectively. These are constructed along the lines of section 6.1.1.1 where $R$ is the rotation matrix by angles $\theta = 45^\circ, 90^\circ$.

### 6.2.1 MirrorNet

The first set of experiments compared the MirrorNet with a CNN. As before the architecture and dataset are as described in section 2.2.1. We reduced the number of channels for MirrorNet so that overall it consists of a similar number of parameters as the CNN for fair comparison. We additionally cross-compared these results with the effects of combining this networks with augmentation. As in the previous chapter

| | MirrorNet | | | |
|---|---|---|---|---|
| | CNN | CNN+hflip | MirrorNet | MirrorNet+hflip |
| CIFAR10 | $76.5\pm(0.29)$ | $81.6\pm(0.14)$ | $78.2(\pm0.12)$ | $82.2(\pm0.05)$ |
| Poison CIFAR10 | $75.1\pm(0.10)$ | $81.6(\pm0.18)$ | $80.1(\pm0.14)$ | $82.0(\pm0.11)$ |

Table 6.1: Performance of MirrorNet compared against standard CNN. Performance compared on CIFAR10 and version where testset poisoned by reflections. Cross-comparison with architectures trained on data augmented by flipped data (hflip).

we looked at both generalisation accuracy on CIFAR10 and on a version where the test set received random reflections. The results of these experiments are below table 6.1.

### 6.2.2 Evaluation

With reference to table 6.1 we can see that MirrorNet outperforms a standard CNN across the board. First we see that MirrorNet outperforms the CNN in terms of generalisation accuracy on CIFAR10 achieving a few percent higher accuracy. Additionally by looking at the accuracy on the poisoned set we see improved robustness as well. In the last chapter we established that CNNs are already fairly robust to reflectional symmetry however, in excess of this, the accuracy of MirrorNet actually increases on the poisoned dataset . It is true that the MirrorNet is still marginally outperformed by CNN plus augmentation however MirrorNet compensates for this with faster training. Additionally, once augmentation is applied to MirrorNet it outperforms the CNN once again.

### 6.2.3 RotateNet

Using the construction given in section 6.1.1.1 as well as example 6.1.3 and example 6.1.2, we create two rotationally equivariant networks. The first is formed from the group of rotations of angles $\{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$ which we call Rotate4Net. The structure of this network is the same as the network called *p4* in Cohen and Welling (2016). The second network has higher rotational resolution and consists of angles which are multiples of $45^\circ$. We call this Rotate8Net. The precise architecure of the network is the rotationally extended version of that described in section 2.2.1. We

| | RotateNet | | | | | |
|---|---|---|---|---|---|---|
| | CNN | CNN+raug | Rot4 | Rot4+raug | Rot8 | Rot8+raug |
| CIFAR10 | 77.0 | 78.6 | **80.2** | 78.4 | 77.1 | 76.8 |
| Poison-CIFAR10 | 44.9 | 78.7 | 79.7 | 78.4 | **80.2** | 79.3 |

Table 6.2: Rotate4Net and Rotate8Net compared with standard CNN on CIFAR10 and a version poisoned by random rotations of the testset. Cross comparisons made when models are trained with rotated data (raug).

test these networks on CIFAR10 and a version poisoned by random rotations. The performance is compared to using rotational augmentation during training (raug).

### 6.2.4 Evaluation

Table 6.2 gives the performance of the Rotate4 and Rotate8 architectures as compared with a standard CNN with similar architecture and parameters. As we can see the performance of these networks is fairly impressive both in terms of overall accuracy and robustness on the poisoned dataset. Results for the Rotate4Net are superior even to the CNN with rotational augmentation. Additionally this occurred with far faster training. As we can see the Rotate8Net performed worse than the Rotate4Net on the unpoisoned dataset and better on the poisoned one. This is consistent with our expectation that there would be diminishing marginal returns in increasing the rotational resolution at the expense of number of channels beyond a certain point.

## 6.3 Conclusions

The models we have constructed in this chapter share similarity to those presented in Cohen and Welling (2016). Correspondingly our results are in general agreement with theirs. Specifically, that group equivariant networks of this type lead to small boosts in generalisation accuracy and more substantial gains on augmented datasets. The results in this section are in stark contrast to those in Chapter 5 (table 5.2 table 5.3). It is also worth highlighting some of the ways in which this work differed from and extended theirs. In the Cohen and Welling paper they did not construct purely reflectionally equivariant network but rather bundled the reflectional symmetry in with the rotational symmetry. Consequently it is noteworthy that this network alone was able to produce

a notable boost. Additionally we were able to produce rotationally equivariance for angles other than multiples of 90°.

Overall however, despite the good results there are also downsides to these methods. First, they are substantially slower than their CNN equivalents in terms of hours to train. The Rotate8Net was particularly slow taking over six times longer to train than the CNN equivalent. Part of this can be allayed by more efficient implementation: One of the major slowdowns is the greater efficiency currently of the two dimensional convolution as compared to the three dimension version in the Pytorch framework which we used extensively.

A second problem is that these architectures are fairly fiddly to create. Whilst this is not infeasible for small commonly occurring groups like those we have considered, this may prove not worthwhile for more sophisticated groups. The main takeaway from this chapter should be that it validates the central claim of this thesis. Specifically that that informed use of inductive biases offer the possibility to improve robustness without loss in accuracy. Nevertheless, it is our belief that more economic and simpler ways to implement these things need to be found.

# Chapter 7

# Equivariance through Regularisation

In this section we look at a non-structural method for achieving group equivariance. Our approach is to add a regulariser formed from the norm of the commutator $g \circ F - F \circ g$. This is with the idea that it should coerce the network into learning an equivariant representation without the hassle of building elaborate architectures. This chapter follows a similar principle to the recently published paper *Invariance-inducing regularisation using worst-case transformations* ...Yang et al. (2019). Whilst much of this work follows similar lines, it is important to note that they use regularisation to induce *invariance* rather than equivariance.

## 7.0.1 Inductive bias represented by L2 Regularisation

**Definition 7.0.1** (Operator Norm)**.** We define the operator norm $||\cdot||_{op}$ of an operator $A : X \to Y$ between two normed vector spaces as $sup_{x \in X} \frac{||Ax||}{||x||}$

### 7.0.1.1 Regularisor

Let $R_n : \mathbb{R}^{n \times n} \mapsto \mathbb{R}^{n \times n}$ be the matrix for a given transformation. Let $A \in \mathbb{R}^{n \times m}$ be the linear map for an arbitrary fully connected layer. The property of $R$-equivariance of $A$ can then be written as the requirement that $AR_n \approx R_m A$. We can then impose condition this by attempting to bound the operator norm of the commutator $||R_m A - AR_n||_{op}$. This can be achieved by using the fact that for a matrix, the operator norm can be bounded above by the Frobenius norm: $||A||_{op} \leq ||A||_2$. So then, we simply add a term to the loss function which is some multiple of the Frobenius norm of the commutator: $E_{equi.} = E + \lambda ||AR_n - R_m A||_2$. The precise value of lambda can be determined through

| l2 Equi-Regularisation | | | | |
|---|---|---|---|---|
| | FC Baseline | l2(x) | l2(x,y) | CNN Baseline |
| CIFAR10 | 59.3($\pm$0.10) | 59.4($\pm$0.11) | 59.9($\pm$0.06) | **63.0**($\pm$0.17) |

Table 7.1: Comparison of l2-equi-regularisation (l2(x), l2(x,y)) with fully connected (FC) and CNN baselines.

experimentation. In order to implement this for multiple groups and at every layer we simply sum over the groups and layers in the network $\sum_{j\in\mathcal{R},i\leq L}||A_iR_j - R_jA_i||_2$.

#### 7.0.1.2 First Experiments

Our first set of experiments tested this method for the case of translation. We took a fully-connected network consisting only of two fully-connected layers. The dimensions of these layers were 3072, 2250, 490, 10. We then constructed a regulariser as described above. We found that for all choices of $\lambda$ that this produced fairly poor results overall. The results of these experiments can been seen in table 7.1. The results of the regularised experiments are only marginally better than the fully-connected baseline and well below the CNN.

### 7.0.2 Explanation of Results

By looking at the values of the matrix $A$ as the experiments progressed we observed that the components were tending towards zero. In this section which give an explanation for this and use it to derive a superior regulariser.

We have seen previously that the convolution operation on functions is translational equivariant (eq. (2.3)). On the other hand, the discrete convolution operator as used in CNNs is only approximately translationally equivariant. We formalise this in the lemma below.

**Definition 7.0.2.** The positive $x$ translation operator in two dimensions is the linear map $R$ acting on the space of $m \times n$ matrices $x_{ij}$ with the property that $(Rx)_{i,j} = x_{i,j-1}$ for all $j \geq 2$ and zero otherwise.

**Note:** The definition above generalises intuitively to translation in the negative $x$

direction and to translation in the $y$-direction.

**Lemma 7.0.1** (Translational Equivariance in One Dimension)**.** *The only linear map $A : \mathbb{R}^n \to \mathbb{R}^m$ which commutes with both positive and negative translation operators is the rescaling operation $A(\vec{x}) = \lambda proj_m(\vec{x})$. Where $proj_m(x_1, x_2, \ldots x_n) := (x_1, x_2, \ldots x_m)$*

*Proof.* Proof in the Appendix section 9.0.1 □

**Lemma 7.0.2** (Translational Equivariance in Two Dimensions)**.** *The only linear map $A : \mathbb{R}^{n\times n} \to \mathbb{R}^{m\times m}$ which commutes with both positive and negative translation operators in the x and y directions is the rescaling operation $A(\vec{x}) = \lambda proj_{m\times m}(\vec{x})$.*

*Proof.* The proof of this is similar to the one dimensional case and is written in full in the Appendix chapter 9. □

The significance of lemma 7.0.2 is that it means that any attempt to impose pure translational equivariance using a $l2$-regulariser is bound to failure. For higher values of $\lambda$ we are in effect coercing the linear map at each layer to become a translation. We believe that this explains the poor results above. In order to proceed with experiments on equivariant regularisation we need to formalise the notion of approximate translational equivariance which we call *pseudo-equivariance* and adjust the regulariser accordingly.

**Definition 7.0.3** (Pseudo Translation Equivariance)**.** We say that a linear map $A^{kl}_{ij} : \mathbb{R}^{n\times n} \to \mathbb{R}^{m\times m}$ is *pseudo-translation equivariant* if $(AR)^{kl}_{ij} = (RA)^{kl}_{ij}$ for all $1 < i, j < m$ and $1 < k, l < n$. Where $R$ denotes any translation operator.

### 7.0.3 Pseudo-equivariance of the Convolution

We saw previously that the convolution operation does not fully satisfy the property of translational-equivariance. However, if we calculate $AR$ and $RA$ we see that for most cases $(AR)_{ij} = (RA)_{ij}$. We can be more concrete in our formulation of this statement.

**Lemma 7.0.3.** *Let $A : \mathbb{R}^{n\times n} \to \mathbb{R}^{m\times m}$ be a linear map. Then A is pseudo-translation equivariant if and only if it defines a convolution.*

*Proof.* Firstly suppose that $A$ is translation pseudo-equivariant. That is $(RA)^{kl}_{ij} = (AR)^{kl}_{ij}$ for all $1 < i, j < m$ and $1 < k, l < n$ for all translation operators $R$. Let $R$ be the positive $y$ translation operator. Then $(AR)^{kl}_{ij} = A^{k,l+1}_{ij}$ and $(RA)^{kl}_{ij} = A^{kl}_{i,j-1}$ so then the pseudo-equivariance condition implies that $A^{kl}_{ij} = A^{k,l+1}_{i,j+1}$ for all $i, k$ and $1 \leq j < m$, $1 < l \leq n$.

Looking similarly at the other three translation operators we derive the following equations:

$$A_{ij}^{kl} = A_{i,j+1}^{k,l+1} \quad \forall i,j,k,l \tag{7.1}$$

$$A_{ij}^{kl} = A_{i+1,j}^{k+1,l} \quad \forall i,j,k,l \tag{7.2}$$

Putting these together it follows that $A_{ij}^{kl} = A_{i+a,j+b}^{k+a,l+b}$ which is true precisely when $A$ is a convolution with stride equal to one. Conversely it is clear from the above characterisation of a convolution that the reverse implication also holds. □

Using this lemma we can make a change to our regulariser so that rather than the 2-norm of the commutator $||AR - RA||_2$ we take the sum of the squares of the elements $(AR-RA)_{ij}^{kl}$ where $1 < i,j < m$ and $1 < k,l < n$. Before we test the performance of this new regulariser we look at the regulariser one forms by considering scale equivariance.

#### 7.0.3.1 Small kernels as a consequence of scale equivariance

In CNNs it is customary to use kernels which are somewhere between $3 \times 3$ to $6 \times 6$ in size whereas the input is typical larger by approximately a factor of ten. One of the reasons for this is that fewer parameters means a lower likelihood of overfitting and reduced computational cost. Nevertheless, the efficacy of small kernels implies that there is a deeper inductive bias that this represents. Some literature exists which interprets in using ideas from signal analysis (Mirco Ravanelli (2018)). Below we show that small kernels can be understood as an approximate application of scale equivariance; one of the groups that have not looked at so far.

**Lemma 7.0.4** (Scale Equivariance)**.** *The convolution operation $f \star k$ is scale equivariant if the the kernel function $k : \mathbb{R}^2 \to \mathbb{R}^2$ has the property that is diminishes proportionally an inverse square in all directions.*

*Proof.* Let $f : \mathbb{R}^2 \to \mathbb{R}^2$. Let $I_\lambda$ denote the scale operation $I_\lambda(f)(x) = f(\lambda x)$ where $\lambda > 0$. Now consider the convolution

$$I_\lambda(f) \star k(x) = \int_{\mathbb{R}^2} f(\lambda y)k(x-y)dy = \int_{\mathbb{R}^2} \frac{1}{\lambda^2} f(y)k\left(\frac{1}{\lambda}(\lambda x - y)\right)dy \tag{7.3}$$

So then, the convolution is equivariant if and only if for every choice of $f : \mathbb{R}^2 \to \mathbb{R}^2$

$$I_\lambda(f \star k)(x) - I_\lambda(f) \star k(x) = 0 \tag{7.4}$$

$$\iff \int_{\mathbb{R}} f(y)\left(\lambda^2 k(\lambda x - y) - k\big(\frac{1}{\lambda}(\lambda x - y)\big)\right)dy = 0 \tag{7.5}$$

| l2 Pseudo Equi-Regularisation | | | | |
|---|---|---|---|---|
| | FC Baseline | $l2_p(x)$ | $l2_p(x,y)$ | CNN Baseline |
| CIFAR10 | 59.3($\pm$0.10) | 63.2($\pm$0.21) | **63.4**($\pm$0.11) | 63.0 ($\pm$0.20) |

Table 7.2: Comparison of fully-connected baseline (FC) with the new regulariser for $x$ translation ($l2_p(x)$) and $x,y$ translation $(l2_p(x,y))$. Results, cross-compared with CNN.

It follows then that $k(\lambda x) = \frac{1}{\lambda^2}k(x)$ for all $\lambda \neq 0$. Changing to polar coordinates this is equivalent to the statement that $k(r\cos(\theta), r\sin(\theta)) = \frac{k(\cos(\theta),\sin(\theta))}{r^2}$ for all $r \neq 0$. Thus the convolution is scale-equivariant if the kernel is of the form

$$k(x) = \frac{1}{||x||^2} k_0\left(\frac{x}{||x||}\right) \tag{7.6}$$

Where $k_0 : S^1 \to S^1$ is some function defined on the circle. □

In particular the kernel should have the property that $|K(x)| \to 0$ as $x \to \infty$.

#### 7.0.3.2 Scale Equivariant Regulariser

Using the results of lemma 7.0.3 and lemma 7.0.4 we can create a new regulariser which coerces greater scale equivariance as well as translational equivariance. In essence we are pushing our network towards having the form of a convolution with a small kernel. We form our regulariser from the sum of two regularisers. The first is formed from the $l2$-norm of the commutator as before. However, we take the sum of the squares of the elements $(AR - RA)^{kl}_{ij}$ where $1 < i, j < m$ and $1 < k, l < n$ in accordance with lemma 7.0.3. The second is the scale-regulariser defined in eq. (7.7). This is designed to encourage convolutions kernels to decrease as an inverse square.

$$\sum_{ijkl}\left(\left(\frac{i}{n} - \frac{k}{m}\right)^2 + \left(\frac{j}{n} - \frac{l}{m}\right)^2\right)A^{kl}_{ij} \tag{7.7}$$

#### 7.0.3.3 Repeated Experiments

We repeat the experiments from section 7.0.1.2 this time using the new regulariser which we label as $l2_p$. The results of these experiments can be seen in table 7.2. Once

again our models are compared with a fully-connected network and a CNN.

As we can see, the performance of these experiments is a huge improvement upon those in section 7.0.1.2. Both of the experiments outperform the fully-connected network. More surprisingly they also both perform as well as the CNN. It is possible that given the amount of hyperparameter searching this is due to overfitting to some degree.

## 7.1 Conclusion

Overall the effects on performance of using regularisers was good. We were able to show that regularisers formed by considering scale and translation equivariance are able to substantially boost the performance of a linear network. In the case of our experiments we had performance on par with a CNN.

Despite these good results there are a number of very important caveats. The first is that the regulariser is a bilinear form in *all* of the model parameters. Consequently the memory requirements to store this for large networks would rapidly become infeasible. Additionally, the results of table 7.2 are a consequence of a very large amount of hyperparameter tuning to make sure that the weightings were well balanced: If one was to consider even more groups then this would rapidly become an unreasonable task.

# Chapter 8

# Conclusion

## 8.1 Summary

We have tried a number of different approaches to building group equivariant/invariant networks. In chapter four we looked at a couple of simple ways that one can build contrast invariance into neural architectures: One method used layerwise contrast invariance and the second used equivariance. We found that our networks were substantially more robust to shifts in domain causes by changes in contrast. Nevertheless, this additional robustness came at the expense of mean accuracy. Additionally we found that batch normalisation produced better results than our method. Nevertheless, this work provides an interesting non-statistical explanation for the efficacy of batch-norm.

Chapter five extended on this work in a more structured manner for the groups defined by brightness, reflection and rotation. We were able to use minor alterations to make a CNN equivariant to these groups and showed that we had improved overall robustness to these transformations. Once again however we observed a general trade-off between robustness and accuracy. In the case of rotation we found that accuracy was hit particularly hard. It was our hypothesis that introducing these equivariances damages the overall expressivity of the network. This is supported by the mathematical work in this section which shows that even for unsophisticated groups such as $GL_2(\mathbb{R})$ that one cannot obtain non-trivial $G-$equivariant CNNs. We also observed the surprising efficacy of augmentation schema both on accuracy and robustness.

Following the empirical and theoretical results regarding expressivity in the last chapter we started to look at other methods for obtaining group equivariance. In chapter

six we showed that by altering our intermediate domains we could theoretically achieve robustness without loss of accuracy. We showed how one might construct such a network in all cases where $G$ was a Lie Group which acted on $H(\mathbb{R}^2)$ through its action on the plane. We then constructed these networks for the group of rotation and reflection in the $x-$axis. Results in the case of both rotation and reflection were impressive and we outperformed an equivalent CNN both in terms of robustness and accuracy. We also found that these networks made greater demands both in terms of memory and time to train. In general we believe that this would prove a poor approach for achieving group equivariance since the size of the network would have to increase in proportion to the cardinality of the group one was looking at.

In the final chapter we took a more radical approach to equivariance by looking at the effect of adding regularisers formed from the 2-norm of various group commutators. The belief was that this could potentially prevent the need for one to engineer complex architectures for every task. Instead one could use a simpler architecture with a regulariser automatically given by known domain symmetries. On top of the advantage of simplicity and generality this would come with the benefit that it would have no marginal computational cost for each group. Additionally, unlike structural methods of equivariance, a regulariser acts as an approximation to a prior distribution. This means that the network can opt for approximate equivariance in favour of improved accuracy. The results of these experiments were good but their impact is limited but the primitivity of the architecture. Additionally, hyperparameter tuning and memory requirements mean that this unlikely to scale to larger networks or larger groups.

## 8.2 Remarks

Overall there are a few major ways in which this work could have been improved. One of the main flaws was the choice of evaluation metrics. We used performance on poisoned and unpoisoned versions of CIFAR10 to compare our architectures with baseline models. Often we found that our models proved better on the poisoned sets but worse on the unaltered version. Our attitude that we should be able “win" in both cases led to a tendency to disregard these model. However, upon reflection, the performance of our models on the unpoisoned testset as compared to a standard CNN is, in essence, beside the point. CNNs, once thought to learn using “increasingly complex representations of

object shape" (Geirhos et al. (2019)) are now understood to be inherently biased towards superficial statistical properties of datasets (Geirhos et al. (2019)). Consequently we believe that a better metric for overall performance is to cross-verify on other dataset with the same categories. Beyond this project, it is our belief that this is a problem prevalent across machine learning within a culture that is more interested in state-of-the art results on particular datasets than the issues of robustness and generality.

A second more fundamental issue with this work and the body of existing literature is that it works on the assumption that the objects that we are interested in have the structure of a group. Whilst this lends itself to an elegant formalism it is not entirely clear that this is a reasonable assumption. A great many transformations that preserve the class are non-invertible meaning that they cannot be explained by this structure. Consequently one should be open to the notion that there is no neat way of wrapping invariances in the language of group theory.

## 8.3 Further Work

One of the major issues when working to build (or show that one cannot build) equivariant architectures is making sure that one has considered a comprehensive set of possible group actions. One always knows how the group acts on the domain of images since this is known by definition. However, there is freedom for many forms of action on the codomain. In lemma 5.1.1 we showed that the brightness action has to be of the form $(\lambda, f) \to \lambda + f$ at all layers of the network. In the proof we implicitly used the surjectivity of the convolution for certain family of functions. However this work needs to be made more rigorous. In particular we need to be more unambiguous in the definition of the space $H(\mathbb{R}^2)$ so that we can make more mathematically concrete claims. This would allow generalisation of lemma 5.1.1 to a broader range of groups. A sketch of how this might work is in the appendix in lemma 9.0.1. This would make the work in chapter 5 entirely general since it would mean that we had considered all possible group actions.

A interesting direction to take this work in would be to generalise spatial transformers using the architecture from chapter 6. We have provided a general framework for building a group-equivariant network however there is no need to explicitly specify

what this group is. Instead one can represent the generator of the group as an learnable map as in spatial transformers (Jaderberg et al. (2015)). That is we allow our network to learn the transformation which leads to highest accuracy on the training set; all one would need to specify is the order of the group. This would an interesting non-meta way of learning data invariances.

# Chapter 9

# Appendix

Proof of Translation Equivariance lemma .

*Proof.* To prove this we use the theorem for the one dimensional case above. Let $R^+_{n,n}$ be the operator which translates an image $x \in \mathbb{R}^{n\times n}$ one pixel to the right. As a matrix can be written as below:

$$R^+_{n,n} = \begin{bmatrix} R^+_n & 0 & 0 & \dots & 0 \\ 0 & R^+_n & 0 & \dots & 0 \\ 0 & 0 & R^+_n & \dots & 0 \\ \dots & & & & \\ 0 & 0 & 0 & \dots & R^+_n \end{bmatrix} \tag{9.1}$$

Where $R^+_n$ is the $n$ by $n$ one-dimensional translation operators.

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & A_{2,3} & \dots & A_{2,n} \\ \dots & & & & \\ A_{m,1} & A_{m,2} & A_{m,3} & \dots & A_{m,n} \end{bmatrix} \tag{9.2}$$

Where $A_{i,j}$ is a $m \times n$ submatrix.

$$AR^+_{n,n} = \begin{bmatrix} A_{1,1}R^+_n & A_{1,2}R^+_n & A_{1,3}R^+_n & \dots & A_{1,n}R^+_n \\ A_{2,1}R^+_n & A_{2,2}R^+_n & A_{2,3}R^+_n & \dots & A_{2,n}R^+_n \\ \dots & & & & \\ A_{m,1}R^+_n & A_{m,2}R^+_n & A_{m,3}R^+_n & \dots & A_{m,n}R^+_n \end{bmatrix} \text{ and } R^+_{m,m}A = \begin{bmatrix} A_{1,1}R^+_m & A_{1,2}R^+_m & A_{1,3}R^+_m & \dots \\ A_{2,1}R^+_m & A_{2,2}R^+_m & A_{2,3}R^+_m & \dots \\ \dots & & & \\ A_{m,1}R^+_m & A_{m,2}R^+_m & A_{m,3}R^+_m & \dots \end{bmatrix} \tag{9.3}$$

By equating submatrices and applying the previous theorem we see that each of the submatrices $A_{i,j}$ of $A$ are constant along the diagonals and zero everywhere else. We

now consider the subset of these maps which also commute with translation in the $y$-direction. Let $T^+_{n,n}$ denote translation by a single pixel in the positive $y$-direction. As a matrix it may be expressed as below.

$$T^+_{n,n} = \begin{bmatrix} 0 & I & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & I & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & I & \dots & 0 & 0 \\ \dots & & & & & & \\ 0 & 0 & 0 & 0 & \dots & I & 0 \end{bmatrix} \tag{9.4}$$

Where $I$ is the $n \times n$ identity matrix.

$$AT^+_{n,n} = \begin{bmatrix} 0 & A_{1,1} & A_{1,2} & \dots & A_{1,n-1} \\ 0 & A_{2,1} & A_{2,2} & \dots & A_{2,n-1} \\ \dots & & & & \\ 0 & A_{m,1} & A_{m,2} & \dots & A_{m,n-1} \end{bmatrix} \quad T^+_{m,m}A = \begin{bmatrix} A_{2,1} & A_{2,2} & A_{2,3} & \dots & A_{2,n} \\ A_{3,1} & A_{3,2} & A_{3,3} & \dots & A_{3,n} \\ \dots & & & & \\ A_{m-1,1} & A_{m-1,2} & A_{m-1,3} & \dots & A_{m-1,n} \end{bmatrix} \tag{9.5}$$

By comparing sub-matrices as before, we see that $A_{i,j} = \mathbf{0}$ for all $i < j$. Then by computing $T^-_{m,m}A$ and $AT^-_{n,n}$ and comparing sub-matrices then we obtain that $A_{i,j} = 0$ for all $i \neq j$.

Putting this all together we conclude that every $A$ is a diagonal matrix of the for $\lambda I_{m^2,n^2}$ □

**Lemma 9.0.1** (Generalised Action)**.** *Let $G = (\mathbb{R}, +)$ act on the domain via the action $(g, f) \mapsto g \circ f$ for some action $\circ$. Let G act on the codomain $H(\mathbb{R}^2)$ via some action $(g, f) \mapsto F_g(f)$. Let $\mathcal{F} : H(\mathbb{R}^2) \to H(\mathbb{R}^2)$ be a G-equivariance integral transform given by a non-zero convolution. Then it follows that $F_g(f) = g \circ f$ for all $g \in G$.*

*Proof.* By assumption, $\mathcal{F} : H(\mathbb{R}^2) \to H(\mathbb{R}^2)$ is of the form $\mathcal{F}(f) := f \star k$ for some non-zero kernel function in $H(\mathbb{R}^2)$. Hence $\mathcal{F}$ is $G$-equivariant if and only if $\mathcal{F}(g \circ f) = ((g \circ f) \star k) = F_g \circ (f \star k)$. Since the convolution operation defined $f \mapsto f \star k, k \neq 0$ is invertible on $H(\mathbb{R}^2)$ then this is equivalent to the expression that $(g \circ (f \star^{-1} k) \star k) = F_\lambda \circ (f)$ for all $f \in H(\mathbb{R}^2)$. Then using commutativity of the convolution and assoicativity, the action of $G$ on the codomain must be of the form $(g, f) \mapsto g \circ f$. □

### 9.0.1 Proof of lemma

*Proof.* Let $R^+_n, R^-_n : \mathbb{R}^n \to \mathbb{R}^n$ denote the positive and negative translations operators on $\mathbb{R}^n$. Let $A : \mathbb{R}^n \to \mathbb{R}^m$ be a linear map such that $AR^+_n = R^+_m A$ and $AR^-_n = R^-_m A$. As

matrices these translation operators can be written

$$R_n^+ = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \dots & & & & & \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix} \text{ and } R_n^- = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ \dots & & & & & \\ 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix} \tag{9.6}$$

Notice that $R^+$ is the transpose of $R^-$.

$$AR_n^+ = \begin{bmatrix} A_{1,2} & A_{1,3} & A_{1,4} & \dots & A_{1,n} & 0 \\ A_{2,2} & A_{2,3} & A_{2,4} & \dots & A_{2,n} & 0 \\ A_{3,2} & A_{3,3} & A_{3,4} & \dots & A_{3,n} & 0 \\ \dots & & & & & \\ A_{m,2} & A_{m,3} & A_{m,4} & \dots & A_{m,n} & 0 \end{bmatrix} \tag{9.7}$$

$$R_m^+ A = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ A_{1,1} & A_{1,2} & A_{1,3} & \dots & A_{1,n-1} & A_{1,n} \\ A_{2,1} & A_{2,2} & A_{2,3} & \dots & A_{2,n-1} & A_{2,n} \\ \dots & & & & & \\ A_{m-1,1} & A_{m-1,2} & A_{m-1,3} & \dots & A_{m-1,n-1} & A_{m-1,n} \end{bmatrix} \tag{9.8}$$

By comparing the components of these two matrices we conclude that $A_{i,j} = 0$ for all $j > i$ and that $A_{i,j} = A_{i+k,j+k}$ for all $k \in \mathbb{N}$.

However, repeating this for $R^-$ we conclude that $A_{i,j} = 0$ for all $j < i$. Consequently, $A$ is an $m \times n$ matrix where $A_{i,i} = \lambda$ and $A_{i,j}$ otherwise. □

# Bibliography


Athalye, A., L. Engstrom, A. Ilyas, and K. Kwok

2018. Synthesizing robust adversarial examples.

Bao, E. and L. Song

2019. Equivariant neural networks and equivarification. *CoRR*, abs/1906.07172.

Brown, T. B., D. Mané, A. Roy, M. Abadi, and J. Gilmer

2017. Adversarial patch. *CoRR*, abs/1712.09665.

Bruna, J. and S. Mallat

2012. Invariant scattering convolution networks. *CoRR*, abs/1203.1513.

Cho, J., K. Lee, E. Shin, G. Choy, and S. Do

2015. Medical image deep learning with hospital PACS dataset. *CoRR*, abs/1511.06348.

Cohen, T., M. Geiger, J. Köhler, and M. Welling

2017. Convolutional networks for spherical signals. *CoRR*, abs/1709.04893.

Cohen, T. S., M. Weiler, B. Kicanaoglu, and M. Welling

2019. Gauge equivariant convolutional networks and the icosahedral CNN. *CoRR*, abs/1902.04615.

Cohen, T. S. and M. Welling

2016. Group equivariant convolutional networks. *CoRR*, abs/1602.07576.

Davies, M.

2011. Google books corpus. based on: Quantitative analysis of culture using millions of digitized books. science 331 (2011) [published online ahead of print 12/16/2010].

Dieleman, S., K. W. Willett, and J. Dambre

2015. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, 450(2):1441–1459.

Dudar, V. and V. Semenov

2018. Use of symmetric kernels for convolutional neural networks. *CoRR*, abs/1805.09421.

Dumont, B., S. Maggio, and P. Montalvo

2018. Robustness of rotation-equivariant networks to adversarial perturbations. *CoRR*, abs/1802.06627.

Engstrom, L., D. Tsipras, L. Schmidt, and A. Madry

2017. A rotation and a translation suffice: Fooling cnns with simple transformations. *CoRR*, abs/1712.02779.

Fawzi, A., H. Samulowitz, D. Turaga, and P. Frossard

2016. Adaptive data augmentation for image classification. In *2016 IEEE International Conference on Image Processing (ICIP)*, Pp. 3688–3692.

Frankle, J. and M. Carbin

2018. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635.

Geirhos, R., P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel

2019. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*.

Goodfellow, I., N. Papernot, S. Huang, Y. Duan, P. Abbeel, and J. Clark

2017. Attacking machine learning with adversarial examples.

Goodfellow, I. J., J. Shlens, and C. Szegedy

2015. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572.

He, K., X. Zhang, S. Ren, and J. Sun

2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852.

Hinton, G. E., A. Krizhevsky, and S. D. Wang

2011. Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning – ICANN 2011*, T. Honkela, W. Duch, M. Girolami, and S. Kaski, eds., Pp. 44–51, Berlin, Heidelberg. Springer Berlin Heidelberg.

Huang, Y., Y. Cheng, D. Chen, H. Lee, J. Ngiam, Q. V. Le, and Z. Chen

2018. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *CoRR*, abs/1811.06965.

Ian Goodfellow, S.

2019. Lecture 16 | adversarial examples and adversarial training.

Ioffe, S. and C. Szegedy

2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

Jaderberg, M., K. Simonyan, A. Zisserman, and K. Kavukcuoglu

2015. Spatial transformer networks. *CoRR*, abs/1506.02025.

Krizhevsky, A. and G. Hinton

2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Krizhevsky, A., I. Sutskever, and G. E. Hinton

2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds., P. 1097–1105. Curran Associates, Inc.

Kuchnik, M. and V. Smith

2019. Efficient augmentation via data subsampling. In *International Conference on Learning Representations*.

L Egizii, M., J. Denny, K. Neuendorf, P. D Skalski, and R. Campbell

2012. Which way did he go? directionality of film character and camera movement and subsequent spectator interpretation.

Lenc, K. and A. Vedaldi

2014. Understanding image representations by measuring their equivariance and equivalence. *CoRR*, abs/1411.5908.

Liu, D., D. Du, L. Zhang, T. Luo, Y. Wu, F. Huang, and S. Lyu

2019. Scale invariant fully convolutional network: Detecting hands efficiently. *CoRR*, abs/1906.04634.

Luo, C., J. Zhan, L. Wang, and Q. Yang

2017. Cosine normalization: Using cosine similarity instead of dot product in neural networks. *CoRR*, abs/1702.05870.

Maron, H., H. Ben-Hamu, N. Shamir, and Y. Lipman

2019. Invariant and equivariant graph networks. In *International Conference on Learning Representations*.

Masci, J., J. Angulo, and J. Schmidhuber

2012. A learning framework for morphological operators using counter-harmonic mean. *CoRR*, abs/1212.2546.

Mirco Ravanelli, Y. B.

2018. Interpretable convolutional filters with sincnet. *In Proceedings of NIPS@IRASL 2018*.

Raudys, J.

1991. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. Pp. 252–263.

Soleymani, S., A. Torfi, J. M. Dawson, and N. M. Nasrabadi

2018. Generalized bilinear deep convolutional neural networks for multimodal biometric identification. *CoRR*, abs/1807.01298.

Stollenga, M. F., J. Masci, F. J. Gomez, and J. Schmidhuber

2014. Deep networks with internal selective attention through feedback connections. *CoRR*, abs/1407.3068.

Strubell, E., A. Ganesh, and A. McCallum

2019. Energy and policy considerations for deep learning in NLP. *CoRR*, abs/1906.02243.

Tommasi, T., N. Patricia, B. Caputo, and T. Tuytelaars

2015. A deeper look at dataset bias. *CoRR*, abs/1505.01257.

Tramer, F., A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel
2018. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*.

Winkels, M. and T. S. Cohen
2018. 3d g-cnns for pulmonary nodule detection. *CoRR*, abs/1804.04656.

Wolpert, D. H. and W. G. Macready
1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Yang, F., Z. Wang, and C. Heinze-Deml
2019. Invariance-inducing regularization using worst-case transformations suffices to boost accuracy and spatial robustness. *CoRR*, abs/1906.11235.

Zhang, R.
2019. Making convolutional networks shift-invariant again.

Zuo, C.
2018. Regularization effect of fast gradient sign method and its generalization.