An Algebraic Approach To Group Equivariant Neural Networks

by

Max Hennick

Bachelor of Science, UNB, 2018

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

In the Graduate Academic Unit of Mathematics

Supervisor(s): Nicholas Touikan, PhD, Dept. of Mathematics & Statistics
Examining Board: Tariq Hasan, PhD, Dept. of Mathematics & Statistics, Chair
Moulay Akhloufi, PhD, Dept. of Mathematics & Statistics
Stijn de Baerdemacker, PhD, Dept. of Chemistry

This thesis is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

June 2021

Abstract

In recent years, two distinct paradigms have been emerging in machine learning. The first is the "data driven" paradigm which makes minimal assumptions about the underlying structure of the data, and instead relies on huge swathes of data and very general algorithms to achieve state-of-the-art performance. The other can be referred to as the "inductive prior" paradigm, which makes prior assumptions about the structure of data, and attempts to design algorithms which exploit this structure, i.e they have a "prior belief" about their input data.

Of interest in this work are deep learning systems belonging to the second paradigm which are built to exploit data with a group-like structure. In this work, we provide an algebraic foundation for such "group equivariant neural networks", and use this foundation to examine their properties. Furthermore, we provide a theorem which gives an explicit method for creating such networks from the underlying structure of the group.

Dedication

This thesis is dedicated to my Mom Laura, my Dad Mike, my Sister Nakaya, and my partner Scout, without whom nothing worthwhile I have done in my life would have been possible.

Acknowledgements

I would like to extend a special thanks to my supervisor Dr. Nicholas Touikan, who went above and beyond to help me achieve my goals, and always showed more belief in me than I probably deserved. I would also like to thank my parrot Captain, since not a word of this thesis was written without him by my side.

Contents

List of Figures

List of Symbols

G	locally compact group.
$\mathbb{F}$	a field, generally $\mathbb{R}$ or $\mathbb{C}$.
$C_c(X, \mathbb{F})$	the space of compactly supported continuous functions from X to $\mathbb{F}$, sometimes written as $C_c(X)$.
$C_0(X, \mathbb{F})$	Closure of $C_c(X)$.
$C_c^\infty(X, \mathbb{F})$	Compactly supported smooth functions on X.
$f * g$	Convolution of function f and g
λ_g	Left translation operator given via $\lambda_g f(x) = f(g^{-1}x)$
ρ_g	Right translation operator given via $\rho_g f(x) = f(xg^{-1})$
e	Group identity
δ_x	Dirac delta centred at x.
$M^\wedge$	Algebraic dual space of module M.

M^*	Topological dual of M.
$\mathrm{Hom}_R(M, N)$	Space of homomorphisms between R-modules M and N.
$M(X)$	Space of radon measures of space X.
D	A Schwartz distribution.
$\mathcal{D}(U)$	Space of Schwartz distributions on open set $U \subseteq X$ for locally compact Hausdorff space X.
$B_U(a, b)$	Compactly supported, smooth functions on open set $U \subset X$ bounded between a and b.

Chapter 1

Introduction

An Overview of Modern GCNN Literature

In recent years, there has been steadily growing interest in developing deep learning systems with a rational design based on the geometric structure of the data being worked with. This idea, called geometric deep learning, has many (deeply related) sub-branches. One such natural geometric prior to exploit is the natural symmetries of data. This is the principal idea driving the development of "Group equivariant convolutional neural networks" (GCNNs for short). The first such network to exploit such structures is really the original convolutional network introduced in [LBD+89] which is a $\mathbb{Z}^2$-equivariant (that is, equivariant to translation) neural network. In the field

of computer vision, equivariance to rotation and reflection was known to be an important prior on data, and had been addressed early on in classical vision by the theory of steerable filters [FA+91]. However, in the deep learning realm, such equivariance was "learned" using data augmentation, such as in [KSH12]. So the effect of roto-reflection groups had to be approximated using translations, requiring high sample complexity.

Unsurprisingly, much of the original (and current) work on GCNNs focuses on solving such problems in the vision domain. Two of the earliest examples to use more general group equivariance are [OM14] and [GD14], which use wavelet transforms and kernel-based interpolation to achieve equivariance respectively. Subsequent works built on the idea of equivariance, with [CW16a] presenting the first work which built GCNNs for discrete groups that were more "intuitively similar" to normal CNNs which could function as a drop-in replacement for normal convolutional layers. They also gave the idea that a more general theory for such networks exist. These types of networks became known as "regular GCNNs".

The next important development came with the works of [CW16b] and [WHS17], which developed GCNNs for vision using classical results in group representation theory in a way that was more parameter efficient, and generalized the idea of "capsules", which were a more "biologically plausible" vision system than normal convolutional networks [HKW11]. These types of networks, called "steerable GCNNs" were useful for making it possible to

work with larger symmetry groups due to how symmetries are represented in the networks, such as [WGTB16] which describes networks which are equivariant to $SE(2)$. Recently, a very general framework for working with networks equivariant to $E(2)$ was given in [WC19], which unifies and expands on previous works, and provides a powerful toolkit for using such networks in practice.

Such steerable networks, while useful for vision, have seen much of their research focused on more complex geometric domains. Much of the interest comes from applications to physics. [SSK$^+$18] developed a network architecture with equivariance properties in 3-dimensions for applications to the quantum many body problem. [TSK$^+$18] give a rigorous generalization of [SSK$^+$18], which provides an $SE(3)$ equivariant networks which functions on irregular point clouds. While less general than [TSK$^+$18], the work [WGW$^+$18] provides examines the instances where the point clouds have a regular, consistent structure which allows for a more efficient implementation. In November 2020, Deepmind released Alphafold2 [Dee21], a model for predicting how amino acids will fold into proteins. This model is considered by many to have effectively solved much of the protein folding problem, considered to be one of the most important problems in modern computational biology. The model was effectively a type of $SE(3)$-equivariant transformer network, similar to that of [FWFW20]. This demonstrates just how powerful such systems can be.

With a growing body of work, a more general theory to describe such networks was desired. The first was [KT18a] which described equivariant GCNNs on compact groups. This was soon followed by [CGW18a] which went further, and described GCNNs on arbitrary homogeneous spaces. This general theory showed that many previous works had, in fact, simply described the same thing unknowingly. It also gave theorems to aid in the development of GCNNs on other spaces. Building on this theory, [WC19] provided a concise system for working with GCNNs on symmetry groups of $\mathbb{R}^2$. More recently, [LW20] generalized the results for $E(2)$ to the action of compact groups on homogeneous spaces.

Remark 1.0.1. There exists a growing body of work applying GCNNs to specific problems and groups. The above summary is not meant to be comprehensive.

Summary of Results

The purpose of this thesis is to give an algebraic perspective on group equivariant convolutional neural networks consistent with the results of [CGW18a]. The usefulness of this is that while the more geometric interpretation of [CGW18a] is very much a natural way to think about group actions on feature spaces, algebra is the natural framework to study the maps between such feature spaces. We first prove an approximation result for equivariant

linear maps:

Theorem. *Let G be a locally compact Lie group, and let W and V be vector spaces over field $\mathbb{F}$. Let $L : C_c(G) \otimes_{\mathbb{F}} V \to C_c(G) \otimes_{\mathbb{F}} W$ be a G-equivariant linear map under the regular representation. L is then an element of the space $\prod_{i=0}^{n} Hom_{C_c(G,\mathbb{F})}(C_c(G,\mathbb{F}) \otimes_{\mathbb{F}} \mathbb{F}^m, C_c(G,\mathbb{F}))$. Furthermore, if f is a function bounded between $a < f(x) < b$ for all x in open set U, every map $L : C_c(G,\mathbb{F}) \to C_c(G,\mathbb{F})$ can always be given by the convolution of a measure ν against input feature map f:*

$$(\nu * f)(g) = \int_G f(gh) d\nu$$

and can be approximated (w.r.t the uniform metric) by a convolution of smooth functions integrated against the Haar measure.

Given the result of proposition 2.5.9 along with Schur's lemma for modules and our approximation result, we then give the following result about the structure of group equivariant convolutional neural networks under the induced representation:

Theorem. *Let G be a locally compact group, with H a compact subgroup of G. Consider now two feature spaces $C_c(G) \otimes_{C_c(H)} V, C_c(G) \otimes_{C_c(H)} W$ which are spaces of representations of G induced by H on the $\mathbb{F}$-vector spaces V, W. the space of G-equivariant $\mathbb{F}$-linear maps between these feature spaces is given*

as

$$Hom_{C_c(G)}(C_c(G) \otimes_{C_c(H)} V, C_c(G) \otimes_{C_c(H)} W)$$

Let L be such an equivariant map and let $\vec{f} \in C_c(G) \otimes_{C_c(H)} V$. L is approximated (w.r.t the uniform metric) by a linear combination

$$L(\vec{f})(g) = (\kappa_1(g)M_1 + \ldots \kappa_k(g)M_k)\vec{f}$$

Where each M_i are linearly independent, H-equivariant elements of $Hom(V, W)$, and each κ_i is a G-equivariant coefficient function $\kappa_i : G \to \mathbb{F}$.

Furthermore, if all the H-invariant subspaces of V are absolutely irreducible over the field $\mathbb{F}$, then the M_i are block matrices which contain scalar multiples of the identity.

This result is a purely algebraic solution (for the case where group representations are absolutely irreducible over the base field) of the kernel constraint in [CGW18a]. This result is also related to the more recent results of [LW20]. The recent work [LW20] in effect uses results from abstract harmonic analysis, drawing on results primarily used in physics, to solve for the kernel constraint of [CGW18a] in a general way, even when considering the representations which are not absolutely irreducible. The main separation between [LW20] and the work here is the approach. As mentioned, they utilize tools commonly used in physics, which is a common trend in GCNN

literature. We depart from this, and frame the problem algebraically.

We conclude this work by showing how one can apply the algebraic theory to develop deep learning models for predicting molecular properties from point clouds of atoms. This theory unifies much of the work done concerning point clouds, and can be seen as a generalization of [SSK+18].

Remark 1.0.2. It is assumed that the reader has some familiarity with the basics of deep learning, and knowledge of elementary group theory and functional analysis. We provide in the appendix an overview of the relevant mathematics and theorems used, along with introducing the basics of deep learning.

1.1 Understanding The Problem

In this work, we will examine how certain algebraic structures can be taken advantage of in order to aid in the solving of a **learning problem**. Thus, it is important to have a consistent definition of a learning problem, and to have a basic understanding of what we mean by "learning". Our introduction to these concepts will be somewhat informal, since we are mainly interested in using them to frame our discussion, not utilizing them for formal proofs.

Definition 1.1.1. Learning Problem: Let X and Y be arbitrary sets and suppose we have a function $f : X \to Y$ that we wish to learn. Given a finite

collection $S \subset X \times Y$ of samples of the form $(x, f(x)) \in X \times Y$ where x is an element of the **feature space** X, and $f(x)$ is in the **output space** Y. A learning problem can be viewed as finding a function $f_\epsilon : X \to Y$ given only the finite samples S.

Example 1.1.2 (Learning Even Numbers). Suppose we have a dataset containing labelled samples given by $(x, f(x))$ where $x \in \mathbb{N}$, and $f(x) = 1$ if x is even, and 0 otherwise. Now, let C_θ denote a classifier with **learnable parameters** θ. As a learning problem, we would like our classifier C_θ to learn a set of parameters θ^* such that

$$C_{\theta^*}(x) = \begin{cases} 1 & \text{if } \frac{x}{2} \in \mathbb{N} \\ 0 & \text{otherwise} \end{cases}$$

We thus refer to C_θ as a "**learnable function**".

While this definition is informal, it's sufficient for our purposes. For a more formal treatment of learning problems, [Vap95] serves as a good introduction.

Our main point of interest is examining structures associated with elements of the feature space X. To motivate our approach, we will start with a few informal examples and use them to frame our discussion.

Example 1.1.3 (The Titanic). Suppose we have a feature space X and a finite dataset $T \subset X$ consisting of everyone who was on the Titanic, and suppose for every person p we have 4 data points, say height, weight, age, and gender.

We then have for each person a feature vector $\vec{p} \in \mathbb{R}^4$. Notice however that our feature vector has a categorical variable in the gender datapoint. Furthermore, if we assume we are only considering age as a whole number, our actual feature space is given as $X = \mathbb{R}^2 \times \mathbb{N} \times \mathbb{N}$.

Example 1.1.4 (Dogs). Suppose we have a feature space X dataset and a $D \subset X$ of pictures of dogs. A standard RGB digital image I can be represented by an array of numbers of shape $(W, H, 3)$ where W is the width and H is the height, where the final dimension represents the "color channels" (the RGB values). Thus, given some pixel coordinate (w, h) we can define a color vector $\vec{c}_{w,h}$ at that point. Observe that such an image thus can be modelled as a compactly supported, bounded function $I : \mathbb{Z}^2 \to \mathbb{R}^3$ where for pixel at point (w, h) in the support of I has value given by $I((w, h))$.

Example 1.1.5 (Atomic Structures). Consider a dataset which consists of a collection of molecules, which we will denote $\mathcal{M}$, which is a finite subset of some feature space X. A molecule $M \in \mathcal{M}$ can be framed as a collection of atoms $(a_1, \ldots, a_n)$. Notice that the properties of each atom a_i can be described by its atomic number and the position of the atom within the molecule can be given as a position vector $\vec{r}_i \in \mathbb{R}^3$. So, each individual atom a_i is described by an element of the space $\mathbb{Z}^+ \times \mathbb{R}^3 = A$ (where $\mathbb{Z}^+$ are the non-negative integers). Since we are concerned with a feature space of molecules, which are collections of atoms, letting u denote some upper limit on the number of atoms in a molecule, our feature space X is described by the the u-fold product $\prod_{i=1}^{u} A$.

Thinking about these examples, notice that the features in the second and third both have inherent geometric structures. In the image example, notice that a structure such as a snout is defined by a collection of pixels within some given neighbourhood. Similarly, a molecule M is determined by the relative spatial locations of its atoms. In the first example, the feature vector $\vec{p}$ does not have such a structure. It should be no surprise then that if we are interested in determining if a dog is contained in an image, or estimating the free energy of a molecule, that having a learning system that takes advantage of the geometric information contained in a data point is beneficial. In what is to come, we will develop a framework for doing exactly this by exploiting symmetries in our data.

1.1.1 A Brief Introduction to Representation Learning

Central to the success of any machine learning system is how the data is represented when it is fed into the algorithm. One of the major reasons for the success of deep learning algorithms in particular is their ability to learn useful representations of data. Much of the current understanding of data representations in machine learning is ad-hoc and informal, but it is nonetheless an important tool.

To see why data representation is important, suppose someone asked you to compute $7 + 5$. Most people would not hesitate to answer 12. However,

suppose someone asked you to add the binary numbers $111 + 101$. Very few people would be able to answer 1100 quickly, if at all. Furthermore, most people who could figure out the answer would not do it by adding the binary numbers. They would convert 111 to the number 7 and 101 to 5, add them, then convert the resulting number 12 to binary. This is because for us binary is not an efficient representation of the data, so we convert its representation to something that is easier to work with.

Mathematically, the idea of representation learning can be thought of as having a system which learns a function R which takes in a datapoint x, and returns some representation of the data $R(x)$, which is then fed to some downstream system. $R(x)$ is sometimes referred to as a **feature representation** for x. How do we know though what makes a "good" representation for our data?

This is an open question in statistical learning theory, however, the most prominent hypothesis is that a good representation is one in which the features within the representation correspond to the underlying causes of the observed data where separate features (or directions in feature space) do not depend on one another, so that the representation disentangles causes of variation. In order to add a bit of formality at the cost of generality, we introduce a definition of this sort of "feature disentanglement" which is sufficient for our purposes.

Definition 1.1.6 (Representation Learner)**.** Suppose we have a feature space X, and a (finite) label space Y and some function we wish to learn $f : X \to Y$ which associates labels in Y to points in X. Let $C_{\theta_1} : V \to Y$ be a linear function with (learnable) parameters θ, where V is a vector space and Y is the label space, and let $R_{\theta_2} : X \to V$ be a (learnable) function from feature space X to vector space V. Consider then the function composition $C_{\theta_1} \circ R_{\theta_2} : X \to Y$. The goal of R_{θ_2} is to learn a representation of the feature space X on vector space V such that given the preimages $f^{-1}(y)$ of labels $y \in Y$, we have that for any pair $y_i, y_j \in Y$ the sets given by $R_{\theta_2}(f^{-1}(y_j))$ and $R_{\theta_2}(f^{-1}(y_i))$ can be separated from one another by hyperplanes in V.

This informal overview of representation learning is useful for contextualizing results which will appear later. Namely, knowing about feature representations allows one to understand results later which show that group representations can "disentangle" sources of variation within features.

Chapter 2

An Algebraic Perspective on Deep Learning

2.1 Convolutional Networks and Modules

Remark 2.1.1. This chapter assumes familiarity with the very basics of deep learning. For a very quick introduction, check appendix A.

2.1.1 Introducing Convolutional Networks

Most of the early work in exploiting geometric structures in data focused on data with Euclidean structure. For instance, in the example 1.1.4 each datapoint has a grid-like topological structure which is effectively a lattice in Euclidean space. Most advanced visual systems in the animal kingdom have methods for exploiting this structure of their visual input [GBC16]. While many of the low level aspects of animal visual systems (particularly mammalian) are generally well understood, many of the higher level aspects are not. However, one of the earliest works in exploiting geometry in learning problems (which is still one of the most prominent tools in deep learning) is very much inspired by some of the simple, low-level parts of these systems to utilize the grid structure of images as an (infinitely strong) prior on the weights of a neural network. This gives rise to the basic idea of the "convolutional layer". We first introduce such layers from the classical machine learning persepctive, starting with an example and definition, before giving the algebraic definition used throughout this work.

Example 2.1.2 (A Convolutional Network Example). Consider the feature space X of RGB images described in example 1.1.4. Now, suppose we wish to learn a function $f : X \to \{0, 1\}$ such that $f(x) = 1$ if the image x contains a dog, and 0 otherwise. Now, recall the idea of representation learning from definition 1.1.6, and suppose we then have a representation learner R_{θ_1} and a linear classifier C_{θ_2}. How might one go about constructing a function R_{θ_1}

for this problem?

Let's suppose for simplicity then that our input images $I \in X$ have a 3×3 resolution, so we can represent them as a multidimensional array:

$$I = \begin{bmatrix} \vec{x}_{1,1} & \vec{x}_{1,2} & \vec{x}_{1,3} \\ \vec{x}_{2,1} & \vec{x}_{2,2} & \vec{x}_{2,3} \\ \vec{x}_{3,1} & \vec{x}_{3,2} & \vec{x}_{3,3} \end{bmatrix}$$

where the $\vec{x}_{i,j}$ are the color vectors (usually referred to as channels). We can now describe what we will call a **collection of filters** $\Psi = (\psi_1, \dots \psi_n)$. The filters in Ψ have a hyperparameter called "kernel size", which is a 2-tuple of numbers which describes the size of the kernel. So, if we have a kernel size of $(2, 2)$, any given kernel $\psi_k \in \Psi$ can be given as the following multidimensional array:

$$\psi_k = \begin{bmatrix} \vec{\psi}_{1,1} & \vec{\psi}_{1,2} \\ \vec{\psi}_{2,1} & \vec{\psi}_{2,2} \end{bmatrix}$$

where $\dim(\vec{\psi}_{i,j}) = \dim(\vec{x}_{i,j})$.

Let's suppose now that our representation learner R_Ψ is parameterized by a set of filters. we would thus like to show how one can compute the output of $R_\Psi(I)$ for image I.

To start with, consider simply a single filter ψ_1 in our collection. Along with the kernel size, we also have another hyperparameter called "stride", which

like the kernel size a 2-tuple of numbers, which describe how we sample points from the input. To make all this clear, suppose we have a stride of $(1, 1)$. We then first compute the output of filter ψ_1 at the first spatial coordinate:

$$[I * \psi_1]((1,1)) = I((1,1)) \cdot \psi_1((1,1)) + I((1,2)) \cdot \psi_1((1,2))$$

$$+I((2,1)) \cdot \psi_1((2,1)) + I((2,2)) \cdot \psi_1((2,2))$$

we then take our stride, first by sliding the coordinates to the right:

$$[I * \psi_1]((2,1)) = I((2,1)) \cdot \psi_1((1,1)) + I((2,2)) \cdot \psi_1((1,2)) +$$

$$I((3,1)) \cdot \psi_1((2,1)) + I((3,2)) \cdot \psi_1((2,2))$$

Notice now that if we stride to the right again, we would end up with a portion of the filter outside the support of the image. We now have two simple options, which is to treat the coordinates outside of the support of I as zeroes (referred to as zero padding), or continue on to the next sample location contained within the support of the image. For simplicity, we will do the latter. Thus, since we can now longer move sideways, we move back to our starting coordinate, and stride down, giving us:

$$[I * \psi_1]((1,2)) = I((1,2)) \cdot \psi_1((1,1)) + I((2,2)) \cdot \psi_1((1,2))$$

$$+I((1,3)) \cdot \psi_1((2,1)) + I((2,3)) \cdot \psi_1((2,2))$$

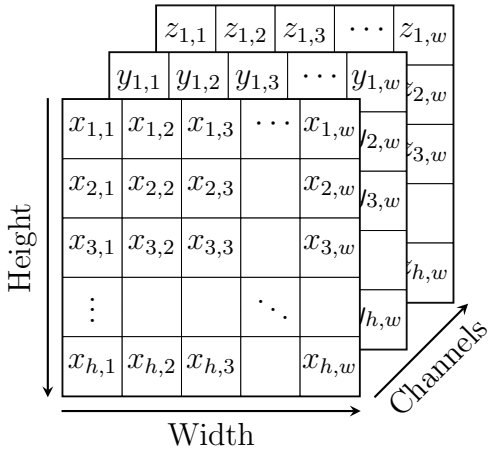

Figure 2.1: An RGB image as a stack of feature maps.

we then continue this way, striding over and down, and computing. This operation on the filter thus gives a matrix

$$[I * \psi_1] = \begin{bmatrix} I * \psi_1((1,1)) & I * \psi_1((2,1)) \\ I * \psi_1((1,2)) & I * \psi_1((2,2)) \end{bmatrix}$$

We then do this for every filter in our collection, giving us a collection of n matrices of shape 2×2 as the output of R_Ψ. This collection of matrices can then be flattened into a vector $\vec{v}$ of dimension $n \times 2 \times 2$, and fed into the classifier C_{θ_1}.

While the above example is illustrative of how convolutional layers are actually used and implemented in practice, the formal definition used to describe them tends to ignore things like stride for simplicity. Such a definition tends to be given as follows:

Definition 2.1.3 (Convolutional Layer)**.** Let l be the index of a layer in a network, and denote the number of channels at layer l by the superscript K^l. Suppose $f : \mathbb{Z}^n \to \mathbb{R}^{K^{l-1}}$ is the input feature maps at layer l, and x, y are two coordinates within the support of f. We then convolve this with K^l **filters** defined by $\psi_{i,j} : \mathbb{Z}^n \to \mathbb{R}$ where $i = 1, 2 \dots K^l$ and $j = 1, \dots, K^{l-1}$. So given the ith filter ψ_i and input f with K^{l-1} feature maps, we define

$$[\ f * \psi_i](x) \ = \sum_{y \in \mathbb{Z}^n} \sum_{j=1}^{K^{l-1}} f_j(y) \psi_{i,j}(y - x)$$

for each ψ_i.

Remark 2.1.4. In the above, we convolve against the filter on the right. We can however "convolve" against the filter on the left. Given

$$[\ \psi_i * f](x) \ = \sum_{y \in \mathbb{Z}^n} \sum_{j=1}^{K^{l-1}} \psi_{i,j}(y) f_j(y - x)$$

and taking the change-of-variables $y = x + y$ we get that

$$[\ \psi_i * f](x) \ = \sum_{y \in \mathbb{Z}^n} \sum_{j=1}^{K^{l-1}} \psi_{i,j}(x + y) f_j(y)$$

Remark 2.1.5. Note that if the input function f in the above definition has outputs in a vector space, in general we may replace the multiplication in the definition with an inner product on that space.

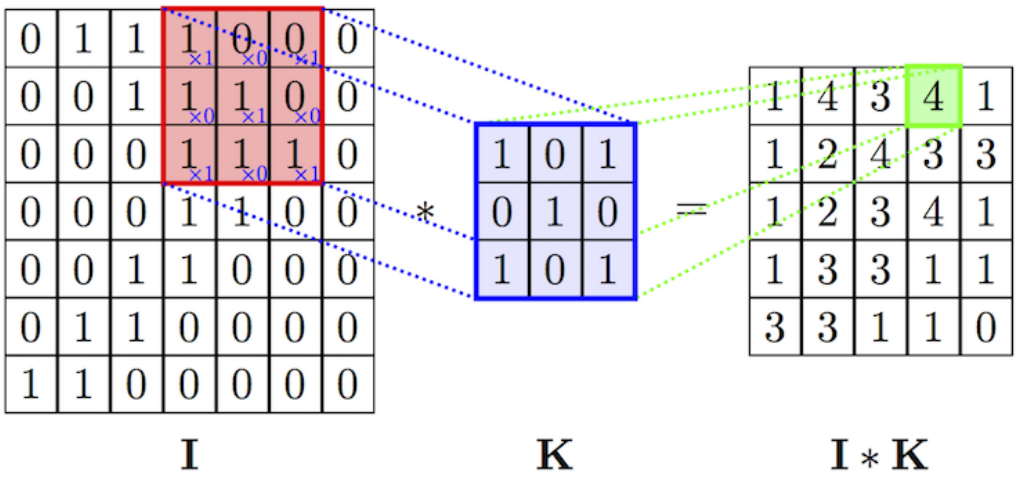

Figure 2.2: Convolution Between a Greyscale input and a Single Filter (from [Anh18])

Observe that we can feed the output of a convolutional layer as input to another convolutional layer, or flatten the output to a vector for input into a fully connected layer. A network with a convolutional layer is referred to as a **convolutional network**.

In the context of representation learning, we can think of each filter ψ_i as detecting some feature in the input f, so the output of $f * \psi_i$ returns a single channel which indicates the "likelihood" of the detection of the feature represented by ψ_i across the spatial locations in the input. In practice, early layers in a network learn simple features like lines and basic shapes, with the features becoming more abstract the deeper in the network one goes. An example of an early feature can be seen in figure 2.3, with a more abstract example being given in 2.4.

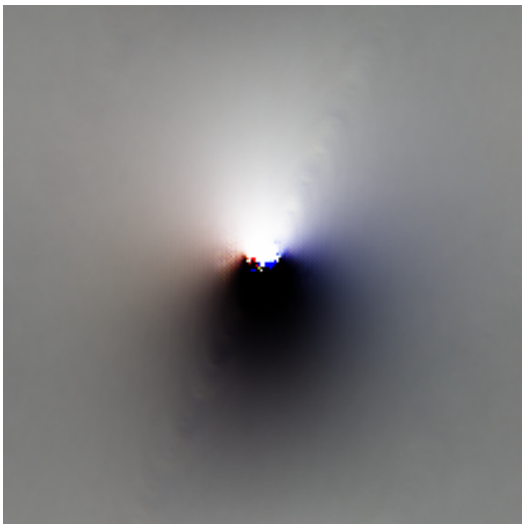

Figure 2.3: A visualization of a filter in the first layer of the network VGG19 trained on Imagenet (found using [Ope20]).

Figure 2.4: A more abstract filter in the last layer of VGG19 (found using [Ope20]).

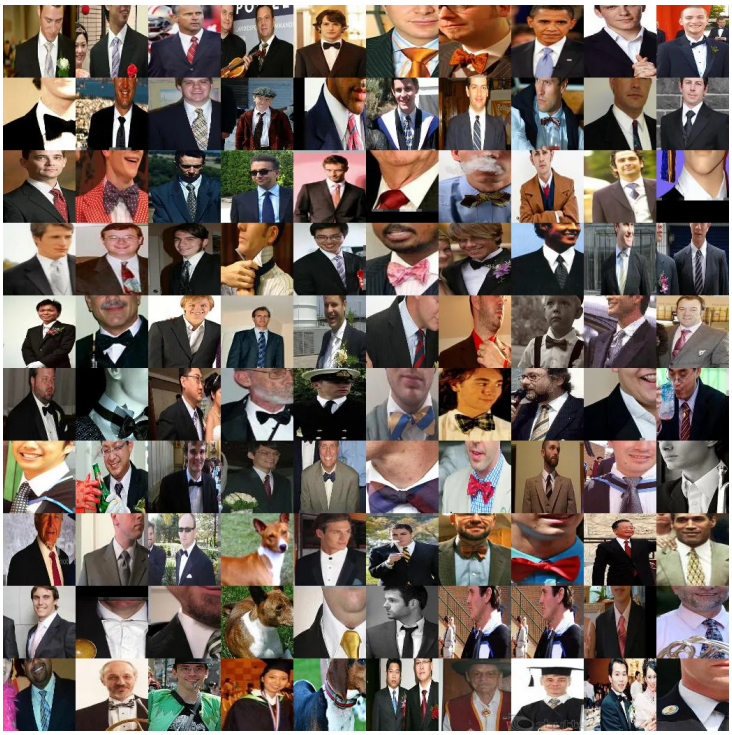

Figure 2.5: Dataset examples for which the filter in figure 2.4 activates strongly for (found using [Ope20]).

As noted above, this idea of convolutional layers originally introduced in [LBD+89] was inspired by neurological structures of the brain. However, we will show that the structure of convolutional layers is actually dictated by an algebraic structure associated with the feature space. Now, recall the definition of a **module**:

Definition 2.1.6 (Module)**.** Let R be a ring. A (left) R-module is then an abelian group M equipped with a scalar multiplicative operation $\cdot : R \times M \to R$ such that for all $r, s \in R$ and $x, y \in M$, the following conditions hold:

1. $r \cdot (x + y) = r \cdot x + r \cdot y$
2. $(r + s) \cdot x = r \cdot x + s \cdot x$
3. $r \cdot (s \cdot x) = rs \cdot x$
4. If R is unital, then $1_R \cdot x = x$ where 1_R is the identity in R

Considering a greyscale image f, we know that f is compactly supported and $\mathbb{Z}^2$ is discrete, so f can be written as a finite sum $\sum_i z_i \vec{r_i}$ where $z_i \in \mathbb{Z}^2$ and $\vec{r_i} \in \mathbb{R}$. This forms a ring, which we will denote $C_c(\mathbb{Z}^2, \mathbb{R})$. All rings form modules over themselves by defining the additive group of the ring R^+ as the abelian group M, with scalar multiplication given by elements of R in the same way as in the ring structure. It follows from this that we have $C_c(\mathbb{Z}^2, \mathbb{R})$ is a module with scalars in $C_c(\mathbb{Z}^2, \mathbb{R})$. Observe as well that within this ring, since features are defined as sums of the form $\sum_i x_i z_i$ (where x_i is

a coefficient, and z_i is a pair of coordinates) and the composition $x_i z_i \cdot x_j z_j$ is given by

$$x_i z_i \cdot x_j z_j = x_i x_j (z_i + z_j)$$

(that is, the x_i are multiplied while the z_i act on one another by translation) we have that left multiplication is given via a convolution.

2.1.2 Advantages of CNNs

CNNs have three main important properties [GBC16]. First, fully connected layers use a matrix multiplication by a matrix of weights where each weight describes the interaction between a specific input and specific output unit. Convolutional layers on the other hand allow for **sparse interactions**. We do this by making our filter smaller than the input. The reason this is effective is rather simple. Suppose we are using such a network on image data. An image has thousands (or even millions) of pixels. We can however detect small, meaningful features such as edges or simple shapes with kernels only tens or hundreds of pixels in size.

Convolutional layers also have a property called **parameter sharing**. In a fully connected network, each element of the weight matrix is used exactly once on a forward pass. However, the parameter sharing used in convolutional layers means that rather than having a separate set of a parameters for every

point in our input, we instead have one set of parameters that is shared across all locations of our input, so we can store our filters in memory as only a few tens of values.

The next property of convolutional networks is equivariance to translations. Before explaining this property, we give the definiton of equivariance.

Definition 2.1.7 (Group Action)**.** Let X be a set, and let G be a group. A (left) group action of G on X is a function $\alpha : G \times X \to X$ such that $\alpha(e, x) = x$ and that $\alpha(g, \lambda(h, x)) = \alpha(gh, x)$.

Definition 2.1.8 (Equivariance)**.** Let G be a group and let X and Y be sets which admit a G-action. Define $\lambda_{g^{-1}}$ to be the operator which performs left translate by g. A function $f : X \to Y$ is (left) equivariant with respect to G if

$$\lambda_{g^{-1}} f(x) = f(gx)$$

for all $g \in G$ and $x \in X$. Right equivariance is defined analogously:

$$\rho_{g^{-1}} f(x) = f(xg)$$

Convolutional networks have the property that they are equivariant to the action of $\mathbb{Z}^n$ (equivariant to translation), which we shall now prove.

Proposition 2.1.9. *Let t be an element of $\mathbb{Z}^n$, with $f : \mathbb{Z}^n \to \mathbb{F}^m$ a stack of feature maps, and $\psi : \mathbb{Z}^n \to \mathbb{F}^m$ belonging to a collection of filters. We then*

have that

$$[\ [\ \lambda_t f]\ *\psi]\ (x) = \lambda_t[\ f * \psi]\ (x)$$

Proof.

$$[\ [\ \lambda_t f]\ *\psi]\ (x) = \sum_y f(y-t)\psi(y-x)$$

by the substitution $y \to y + t$ we get

$$= \sum_y f(y)\psi(y+t-x)$$

$$= \sum_y f(y)\psi(y-(x-t))$$

$$= [\ f * \psi]\ (x-t)$$

$$\lambda_t[\ f * \psi]\ (x)$$

□

This property of translation equivariance says that if we translate a chunk of an image within its support which produces some activation in a filter, that activation has a corresponding translation in the output. As we will show, thinking of images as functions which form a module gives a natural algebraic description of convolutional layers as a type of linear map.

Definition 2.1.10 (Module Homomorphism)**.** Let $f : M \to N$ be a map from an R-module M to another R-module N and let $r, s \in R$ and $m, n \in M$.

Then f is a (right) module homomorphism if $f(r \cdot m + s \cdot n) = r \cdot f(m) + s \cdot f(n)$. If f is also bijective, it is called a module isomorphism.

A convolutional layer in practice is a map $L : C_c(\mathbb{Z}^k, \mathbb{R}^n) \to C_c(\mathbb{Z}^k, \mathbb{R}^m)$ where n and m are the number of "**channels**". Our results in the case of a single channel generalize readily to the multichannel case. This means that such a layer is equivariant to translations, which means it is linear with respect to $\mathbb{Z}^k$. Similarly, convolution has the property that given scalar a, $a(f * g) = af * g$. So, since by definition the layer L is $L(f) = f * \psi$ for filter function(s) ψ, it follows that $aL(f) = a(f * \psi) = af * \psi = L(af)$, so L is also $\mathbb{R}$-linear, and is thus a $C_c(\mathbb{Z}^k, \mathbb{R}^n)$-module homomorphism.

We can see from the above that convolutional layers effectively exploit properties of module homomorphisms on the feature space of images. In the coming sections, we generalize this to arbitrary locally compact groups.

2.2 Group Rings, Modules, and Function Spaces

2.2.1 Group Rings and Modules

We begin this section by giving a more general version of the space $C_c(\mathbb{Z}^2, \mathbb{R})$:

Definition 2.2.1 (Group Ring)**.** Let G be a locally compact group, and

consider the space of continuous compactly supported functions from G to underlying field $\mathbb{F}$, denoted $C_c(G, \mathbb{F})$. Now, we can turn this function space into a ring with multiplication defined by the convolution of functions, and addition given by normal function addition. For now we will refer to $C_c(G, \mathbb{F})$ as the group ring of G over $\mathbb{F}$. For simplicity, $C_c(G) = C_c(G, \mathbb{F})$ unless otherwise specified.

Remark 2.2.2. Note that we do not require rings to be unital. Furthermore, we would generally like for G to be Hausdorff. However, as explained in appendix C, this is not particularly restrictive and can be largely ignored for locally compact groups.

Similar to $C_c(\mathbb{Z}^2, \mathbb{R})$ case, these group rings can be equipped with a scalar multiplicative structure over themselves, which is given by

$$f_1 f_2 = (f_1 * f_2)$$

where $*$ is convolution. It is important to note that this convolution is commutative if and only if the group G is commutative. Furthermore, the additive structure of the group ring forms an abelian group, meaning the group ring $C_c(G, \mathbb{F})$ is thus a left module over itself with scalar multiplication given by convolution. We can also define a right multiplication by itself, meaning it can be viewed as a bimodule.

Example 2.2.3. Consider the group ring $C_c(C_4, \mathbb{R})$. Let f be an element of

this group ring. f then has the form

$$f = x_1e + x_2g_2 + x_3g_3 + x_4g_4$$

where e is the identity and the x_i are elements of $\mathbb{R}$. Now consider some other function

$$h = y_1e + y_2g_2 + y_3g_3 + y_4g_4$$

We can then consider the left multiplication of h by f which is given by the convolution $f * h$. This is computed by

$$f * h = (x_1e + x_2g_2 + x_3g_3 + x_4g_4)(y_1e + y_2g_2 + y_3g_3 + y_4g_4)$$

$$= x_1y_1e + x_1y_2g_2 + x_1y_3g_3 + x_1y_4g_4 +$$

$$x_2y_1g_2 + x_2y_2g_3 + x_2y_3g_4 + x_2y_4e +$$

$$x_3y_1g_3 + x_3y_2g_4 + x_3y_3e + x_3y_4g_2 +$$

$$x_4y_1g_4 + x_4y_2e + x_4y_3g_2 + x_4y_4g_3$$

collecting terms, this gives

$$= (x_1y_1 + x_2y_4 + x_3y_3 + x_4y_2)$$

$$+g_2(y_2x_1 + x_2y_1 + x_3y_4 + x_4y_3)$$

$$+g_3(x_1y_3 + x_2y_2 + x_3y_1 + x_4y_4)$$

$$+g_4(x_1y_4 + x_2y_3 + x_4y_1 + x_3y_2)$$

Following the linguistics of machine learning, we frequently refer to the modules formed by group rings $C_c(G)$ as feature spaces, and elements of these spaces as feature maps.

2.2.2 The Approximate Identity

Consider a group ring $C_c(G)$ where G is discrete as in example 2.2.3. Notice that in this case, one can define the identity element of the ring as $1e$ since given some element of the ring $\sum_i x_i g_i$ we have that $1e \sum_i x_i g_i = \sum_i x_i g_i$. Since multiplication is given by convolution, if G is not discrete, notice that the Dirac distribution δ_1 given by $(f * \delta_1)(g) = \int_G f(gh)\delta_1(h)d\mu = f(g)$ would behave as the identity. However, $\delta_1 \notin C_c(G, \mathbb{F})$, (since the delta function is not technically a function) and by the uniqueness of the identity, $C_c(G, \mathbb{F})$ does not have a proper identity. Because it lacks an identity, notice that we cannot have $G \subset C_c(G, \mathbb{F})$. In order to resolve this issue, we can use a **approximate identity**. If we equip $C_c(G, \mathbb{F})$ with some L^p norm for $1 \leq p < \infty$, the completion of this is then $L^p(G)$. This is important, as it gives us a route to our desired approximate identity. We first must provide a few results from [Fol13]:

Definition 2.2.4 (Left Uniform Continuity)**.** We call a function $f \in C_c(G)$ left (resp. right) uniformly continuous if for all $\epsilon > 0$ there is a neighbourhood U about the identity $e \in G$ such that $\|\lambda_g f - f\|_\infty < \epsilon$ for all $g \in U$.

Lemma 2.2.5 (([Fol13]))**.** *If $1 \leq p < \infty$ and $f \in L^p$ then $\|\lambda_g f - f\|_p$ and $\|\rho_g f - f\|_p$ tend to 0 as $g \to 1$ where λ_g is the left translate and ρ_g is the right translate.*

Lemma 2.2.6 (Minkowski's Inequality ([Fol13]))**.** *Let f, g be elements of $L^p(\mu)$ for $1 \leq p \leq \infty$. Then we have:*

$$\|f + g\|_p \leq \|f\|_p + \|g\|_p$$

Lemma 2.2.7 (Young's Inequality ([Fol13]))**.** *If $f \in L^1$ and $\psi \in L^p$ for any $1 \leq p \leq \infty$, then $f * \psi(x)$ exists for almost every x, and we have that*

$$\|f * \psi\|_p \leq \|f\|_1 \|\psi\|_p$$

Lemma 2.2.8 (([Fol13]))**.** *$f \in C_c(G)$ is both left and right uniformly continuous.*

Proposition 2.2.9 ([Fol95])**.** *Let $\mathcal{U}$ be a neighbourhood about the identity e in G. For each $U \subset \mathcal{U}$ let ψ_U be a function with compact support contained in U, $\psi_U \geq 0$, $\psi_U(g) = \psi_U(g^{-1})$ and $\int \psi_U d\mu = 1$ where μ is the Haar measure. Then $\|f * \psi_U - f\|_p \to 0$ as $U \to \{e\}$ (that is, as U shrinks to the identity) if $1 \leq p < \infty$ and $f \in L^p$ or if $p = \infty$ and f is right uniform continuous.*

Similarly, $\|\psi_U * f - f\|_p \to 0$ *as* $U \to \{e\}$ *if* $1 \leq p < \infty$ *and* $f \in L^p$ *or if* $p = \infty$ *and* f *is left uniform continuous.*

Proof. Since $\psi_U(g^{-1}) = \psi_U(g)$ and $\int \psi_U d\mu = 1$ we have

$$(f * \psi_U)(g) - f(g) = \int f(gh)\psi_U(h^{-1})d\mu - f(g)\int \psi_U(h)d\mu$$

$$= \int (\rho_{h^{-1}} f(g) - f(g))\psi_U(h)d\mu$$

and by Minkowski's inequality we have

$$\|f * \psi_U - f\|_p \leq \int \|\rho_{h^{-1}} f - f\|_p \psi_U(h)d\mu \leq \sup_{h\in U} \|\rho_{h^{-1}} f - f\|_p$$

and by our lemma 2.2.5, we have that $\|f * \psi_U - f\|_p$ goes to 0 and we are done. The second assertion is proven similarly. □

A family of functions $\{\psi_U\}$ form an **approximate identity**. Since each approximate identity is compactly supported and continuous, it belongs to $C_c(G)$.

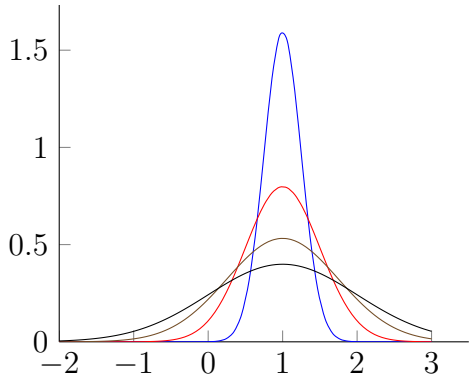

Figure 2.6: Approximations of the identity (dirac delta) δ

2.3 Equivariant Linear Maps

A layer between two feature spaces with channel depth 1 in a (regular) GCNN can be thought of as a function $L : C_c(G, \mathbb{F}) \to C_c(G, \mathbb{F})$ which is equivariant to the action of G and linear with respect to $\mathbb{F}$. Considering the equivariant layer L, we define the following map

$$\Lambda_e : C_c(G, \mathbb{F}) \to \mathbb{F}$$

as being given by

$$\Lambda_e(f) = [\ L(f)]\ (e)$$

First we show that Λ_e is a linear functional.

Proposition 2.3.1. *The map given as* $\Lambda_e(f) = [\ L(f)]\ (e)$ *is a* $C_c(G, \mathbb{F})$*-linear functional.*

Proof. To show that Λ_e is linear, we must show that it is homogeneous with respect to $\mathbb{F}$ and that it is additive. To see that it is additive, we observe that since L is linear, we have

$$\Lambda_e(f+\psi) = [\ L(f+\psi)]\ (e) = [\ L(f) + L(\psi)]\ (e)$$

$$= [\ L(f)]\ (e) + [\ L(\psi)]\ (e) = \Lambda_e(f) + \Lambda_e(\psi)$$

So Λ_e is indeed additive. Now let $r \in \mathbb{F}$ be an arbitrary scalar. To show that Λ_e is homogeneous with respect to $\mathbb{F}$, we once again use the linearity of L:

$$\Lambda_e(rf) = [\ L(rf)]\ (e)$$

$$= [\ rL(f)]\ (1) = r[\ L(f)]\ (e)$$

$$= r\Lambda_e(f)$$

so Λ_e is homogeneous, and is thus a linear functional. □

Before continuing, we will note a few things about this linear functional. By the Riesz representation theorem (for a review of the Riesz representation theorem and its variants, check D) we know that it can be given by an integral $\Lambda_e(f) = \int_G f(h)d\mu(h)$, and that $\Lambda_e(f)$ corresponds to the measure μ, which is the Dirac measure δ at the identity. We can thus think of $\Lambda_e(f)$ as describing the value of f integrated against μ. Thus, we can consider it

as the **evaluation functional** for μ. Defining now $\Lambda_x(f) = [\ L(f)]\ (x) = \int_G f(xh)d\mu(h)$ we give the following proposition to describe the group action on these functionals:

Proposition 2.3.2. *Let* $[\ L(f)]\ (x)$ *be a* $C_c(G, \mathbb{F})$*-linear functional. The right action* $\rho_{g_1^{-1}}$ *commutes with the left action* $\lambda_{g_2^{-1}}$ *on* $[\ L(f)]\ (x)$

Proof.

$$\lambda_{g_2^{-1}}\rho_{g_1^{-1}}[\ L(f)]\ (x) = [\ L(f)]\ (g_2 x g_1)$$

$$= \lambda_{g_2^{-1}}[\ L(f)]\ (xg_1)$$

$$= \rho_{g_1^{-1}}\lambda_{g_2^{-1}}[\ L(f)]\ (x)$$

□

Recall that a function $\rho : G \to H$ from group G to group H is a homomorphism if $\rho(g)\rho(h) = \rho(gh)$ for all $g, h \in G$. We now give the following two propositions:

Proposition 2.3.3. *Consider the linear functional* $\Lambda_x(f)$*, defined as* $[\ L(f)]\ (x)$*.* $\Lambda_x(f)$ *is right equivariant, and the action describes a group homomorphism to the base field* $\mathbb{F}$*.*

Proof. Observe that we have

$$\rho_{g^{-1}}\rho_{h^{-1}}[\ L(f)]\ (x) = [\ L(f)]\ (xhg)$$

and that

$$\rho_{g^{-1}h^{-1}}[\ L(f)]\ (x) = [\ L(f)]\ (xhg)$$

which implies that $\rho_{g^{-1}}\rho_{h^{-1}} = \rho_{g^{-1}h^{-1}}$, so ρ describes a homomorphism. $\square$

2.4 Kernel Functions and Equivariant Cross-Correlation

In this section we show how for almost all cases we are interested in, the G-equivariant linear maps we use for producing GCNNs can be given by a cross-correlation.

2.4.1 Informal Discussion of Previous Work

Theorem 6.1 in [CGW18a] states that an equivariant linear map is always given by a cross-correlation:

Theorem 2.4.1 (Theorem 6.1 in [CGW18a])**.** *An equivariant linear map can always be written as*

$$\int_G \kappa(g^{-1}h) f(h) dh$$

where $\kappa : G \to Hom(V_1, V_2)$ is a map from G to the linear operators between V_1 and V_2.

The purpose of κ outputting vector space homomorphisms is natural in the context in which the theorem is presented, as it was meant to give a cross-correlation with equivariant linear maps under the induced representation (which we will introduce later). There are however some small issues when considering said theorem in the context of regular representations, which we will now address.

In said proof, they use the fact that "All linear maps can be written as $\int f(x)\kappa(x,y)dx$" which is true, but only when κ is allowed to be a generalized function (a Schwartz Distribution). The κ function as defined in [CGW18a] is not a generalized function, as it takes in values of G and returns elements of a vector space. A distribution, in contrast, is given as a linear functional $D : C_c^\infty(U, \mathbb{F}) \to \mathbb{F}$, where U is an open set of G. In the proof it is also assumed that we can (for almost every instance that one would be interested in) always integrate with respect to the Haar measure, an assumption which we provide rigorous proof of.

2.4.2 Formalizing Equivariant Linear Maps as Convolutions

To understand equivariant layers, we need to understand the structure of the space of module homomorphisms.

Definition 2.4.2. Let M be a left (resp. right) module over ring R. The dual module $M^\wedge$ is given as

$$\mathrm{Hom}_R(M, R)$$

If R is non-commutative, then $M^\wedge$ is a right R-module. $M^\wedge$ is usually referred to as the **algebraic dual** of M. The subset of continuous maps of $M^\wedge$, denoted M^*, is referred to as the **topological dual** of M.

Under the view of $C_c(G, \mathbb{F})$ as a module over itself, the equivariant layer L can be viewed as belonging to the dual module of $C_c(G, \mathbb{F})$, $C_c(G, \mathbb{F})^\wedge$. Furthermore, we can identify this space with the space of module homomorphisms

$$\mathrm{Hom}_{C_c(G,\mathbb{F})}(C_c(G, \mathbb{F}), C_c(G, \mathbb{F}))$$

We would now like to relate this space with our linear functional Λ_e which can be thought of as an element of the dual module $\mathrm{Hom}_{\mathbb{F}}(C_c(G, \mathbb{F}), \mathbb{F})$. The result is related to a structure called the **contragradient representation** of a group. The contragradient representation is described by the action $\hat{\rho}_g \Lambda(f) = \Lambda(\rho_{g^{-1}} f)$ where Λ is a linear functional.

Recall that there is a version of the Riesz-Representation theorem (theorem D.0.5 appendix D) which says that there is an isometric isomorphism from the space of Radon measures on our space, which we will denote by $M(G)$, to the

(topological) dual space $C_0(G,\mathbb{F})^*$. Since for any positive linear functional Λ on $C_c(G)$, there exists a corresponding (unique) Radon measure μ (theorem D.0.2 appendix D), we get that all such positive linear functionals belong to $C_0(G,\mathbb{F})^*$. Furthermore, all bounded linear functionals can be expressed as a linear combination of positive linear functionals, so all bounded linear functionals are contained in $C_0(G,\mathbb{F})^*$.

Theorem 2.4.3 (Convolution of Measures ([Fol95]))**.** *Consider the space $M(G)$ and let $I : M(G) \to \mathbb{F}$ be the map defined by*

$$I(\psi) = \int\int \psi(xy)d\mu(x)d\nu(y)$$

where $\psi \in M(G)$. This map is a linear functional which satisfies the inequality

$$I(\psi) \leq \|\psi\|_\infty \|\mu\|\|\nu\|$$

and is thus given by a measure $(\mu * \nu) \in M(G)$ with $\|\mu * \nu\| \leq \|\mu\|\|\nu\|$. We then define the convolution of measures as

$$(\mu * \nu) = \int \psi d(\mu * \nu) = \int\int \psi(xy)d\mu(x)d\nu(y)$$

This lets us give $C_0(G,\mathbb{F})^* = M(G)$ the structure of an (associative) **algebra** (an algebra is an R-module M where multiplication is defined between elements of the underlying abelian group for M) called the **measure algebra**.

The space $M(G)$ is very large and complicated, but has some nice properties as well. It has a true multiplicative identity given as:

$$\int \psi d(\delta * \mu) = \int \int \psi(xy) d\delta(x) d\mu(y)$$

$$= \int \psi(y) d\mu(y)$$

Where δ is the Dirac measure at the identity. We can also define precisely what it means to convolve a measure and a function. We give this in [Fol95] as

$$(\nu * f)(g) = \int_G f(gh) d\nu$$

Theorem 2.4.4. *The space $Hom_{\mathbb{F}}(C_0(G, \mathbb{F}), \mathbb{F})$ has a natural identification with $Hom_{C_0(G,\mathbb{F})}(C_0(G, \mathbb{F}), C_0(G, \mathbb{F}))$.*

Proof. Consider again the linear functional $\Lambda_e : C_0(G, \mathbb{F}) \to \mathbb{F}$. Notice that by our definition of measure-function convolution, this functional corresponds to the value $\Lambda_e(f) = (\mu * f)(e)$. Consider now the contragradient left action given by $\lambda_g \Lambda_e(f) = \Lambda_e(\lambda_{g^{-1}} f)$. This corresponds to the convolution

$$(\mu * \lambda_{g^{-1}} f)(e) = \int f(gh) d\mu$$

Notice then that the contragradient left action is the same as the left action given by

$$\lambda_g(\mu * f)(e) = (\mu * f)(g)$$

This says that the function defined by $\mu * f$ in $C_0(G, \mathbb{F})$ evaluated at g is equivalent to the evaluation of f with the left contragradient translate of the linear functional for μ, which is an element of $\mathrm{Hom}_{C_0(G,\mathbb{F})}(C_0(G, \mathbb{F}), C_0(G, \mathbb{F}))$.

Now, in the other direction, consider the value

$$\lambda_g(\mu * f)(e) = (\mu * f)(g)$$

$$= \int f(gh) d\mu(h)$$

which is an element of $\mathrm{Hom}_{C_0(G,\mathbb{F})}(C_0(G, \mathbb{F}), C_0(G, \mathbb{F}))$. Notice that using the convolution of measures we can rewrite this as the integral

$$= \int f(h) d(\delta_g * \mu)$$

where δ_g is the Dirac measure centered at g. This integral is then a linear functional on $C_0(G, \mathbb{F})$ and is thus an element of the space $\mathrm{Hom}_{\mathbb{F}}(C_0(G, \mathbb{F}), \mathbb{F})$.

□

The above result shows that (regular) group equivariant convolutional layers are described by integral transforms that are given by a convolution of a measure and a function, so they can always be written as a type of convolution. This is the case since $\Lambda_g(f) = \int f(gh) d\mu = (\mu * f)(g)$. However, the measure μ is not necessarily a Haar measure, which means that the "volume"

described by the measure is not invariant, or that the measure μ is not absolutely continuous w.r.t the Haar measure. We focus now on rectifying this by showing that for reasonably restricted input functions, our maps can always be approximated (w.r.t the uniform metric) by a convolution of smooth functions integrated against the Haar measure.

2.4.3 Regular GCNNs are Approximated by Convolutions

Definition 2.4.5. Let X be a locally compact Hausdorff space, and let U be a compact subset of X. A **mollifier** is a compactly supported smooth $\psi_\epsilon : X \to \mathbb{F}$ belonging to a set of functions $\{\psi_\epsilon \forall \epsilon > 0\}$ such that $\int_U \psi_\epsilon dm = 1$, and $\lim_{\epsilon \to 0} \psi_\epsilon(x) = \delta(x)$ where δ is the dirac function.

Approximate identities (2.2.9) are mollifiers (this follows straightforwardly from the definition), and thus a mollifier exists for every ϵ. Now, define the function

$$f_\epsilon(x) = \int_U f(y^{-1}x)\psi_\epsilon(y)dm$$

We wish to show that this is a smooth function.

Proposition 2.4.6. *Let G be a finite dimensional Lie group, let f be a function in $C_c(G)$ and let ψ_ϵ be an approximation of the identity in $C_c(G)$. Then the function given by $f_\epsilon(x) = \int_U f(y^{-1}x)\psi_\epsilon(y)dm$ is smooth.*

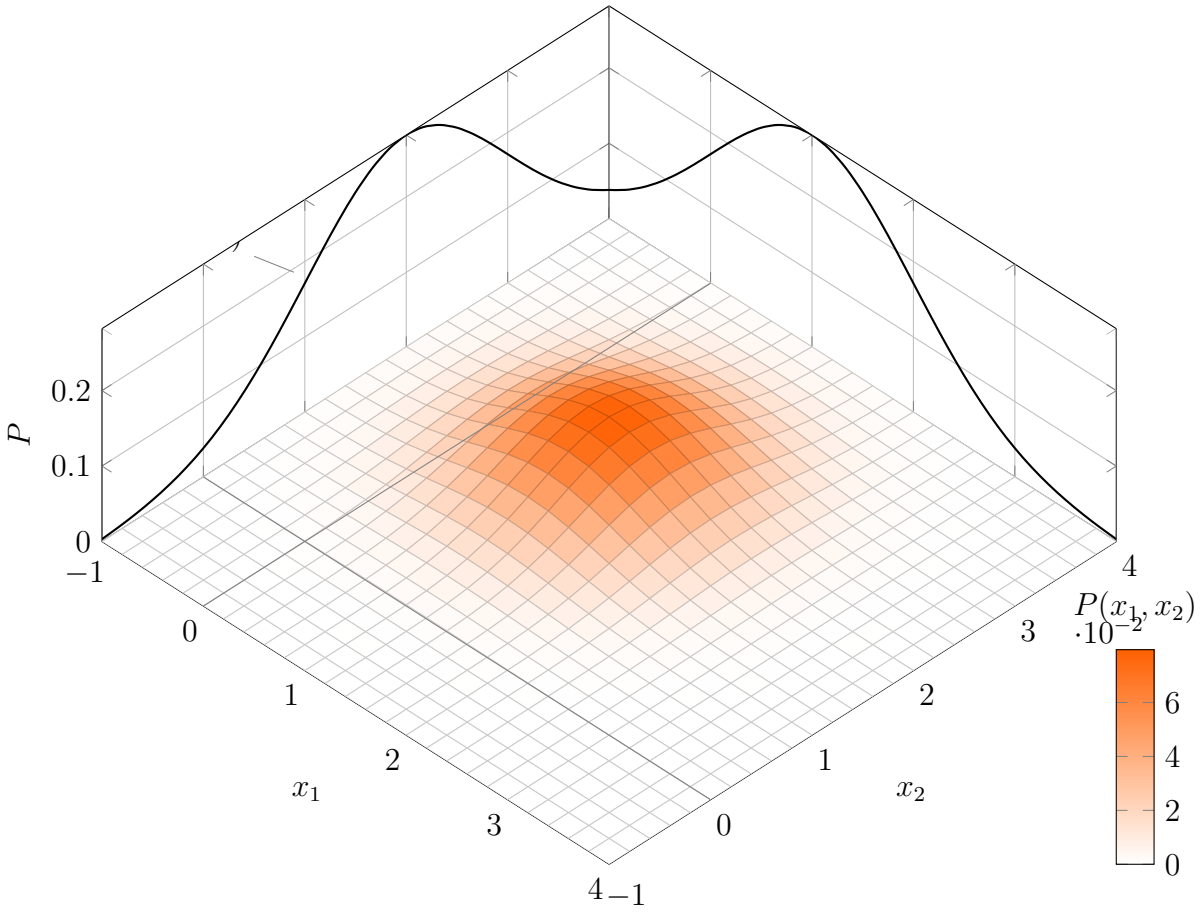

Figure 2.7: An Example of a Mollifier for functions on $\mathbb{R}^2$.

Proof. First, we suppose that ψ_ϵ is an approximation of the identity on both the left and right, so we get that $f * \psi_\epsilon = \psi_\epsilon * f$. Let ∂ be a differential operator on $C_c(G)$. We then have that

$$\partial f_\epsilon = \partial(f * \psi_\epsilon)$$

$$= \partial(\psi_\epsilon * f)$$

$$= \partial\psi_\epsilon * f$$

and since ψ_ϵ is smooth the result follows. □

We now have the following result from [Fol13]:

Theorem 2.4.7. *Let X be a locally compact Hausdorff space. f is bounded and uniformly continuous function on X, then there exists a sequence of functions $\{g_n\}$ such that for any $\epsilon > 0$ there exists an $N \in \mathbb{N}$ such that for all $n \geq N$ there exists a function g_n such that $\|g_n - f\|_\infty < \epsilon$.*

For the remainder of this section, we let f_ϵ denote the function mollified by ψ_ϵ such that $\|f_\epsilon - f\|_\infty < \epsilon$. Before moving on, since we are concerned with f being both bounded and compactly supported, it follows that $\int_U |f| d\mu < \infty$, so $f \in L^1$ and $f \in L^\infty$, so we are concerned with inputs $f \in L^1 \cap L^\infty$. We would also like f to be uniformly continuous as to be able to make f smooth. We first show that we can always give a uniformly continuous approximation

of f.

Theorem 2.4.8 (Stone-Weierstrass Theorem)**.** *Let X be a locally compact Hausdorff space. A subalgebra A of $C_0(X)$ is dense in $C_0(X)$ given the topology of uniform convergence if and only if it separates points and vanishes nowhere.*

It is straightforward from the above theorem that $C_c(G)$ is dense in $C_0(G)$. However, the restriction of f being bounded and compactly supported does not make f belong to $C_0(G)$ since it is need not be continuous. We thus give a justification for restricting f to being continuous as well. In practice we are generally considering discretized inputs, where functions are described on some $\mathbb{Z}^k$ grid. In the discrete topology these functions are continuous by definition. However, for full generality we appeal (somewhat informally) to Lusin's theorem:

Theorem 2.4.9 (Lusin's Theorem)**.** *Let f be a measurable function on locally compact Hausdorff space X that vanishes outside a set of finite measure. Then for any $\epsilon > 0$ there exists a $\psi \in C_c(X)$ such that $f = \psi$ except on a set of measure $< \epsilon$. If f is bounded, a ψ can always be found such that $\|\psi\|_\infty \leq \|f\|_\infty$.*

To see how this helps us, suppose κ is an L^1 function and let f be a bounded, compactly supported function. We then have by Lusin's theo-

rem that $\|\kappa\|_1\|\psi\|_\infty \leq \|\kappa\|_1\|f\|_\infty$. Then, by Young's Inequality, we have $\|\kappa * \psi\|_\infty \leq \|\kappa * f\|_\infty$.

Informally, Lusin's theorem tells us that for any input function f we are interested in, there is another function ψ which is continuous and equivalent to f on all but an arbitrarily small patch where ψ is "noisy". Convolution with this approximated function returns an output that is "smoother" (like smoothing in computer vision) around the noisy patch of the input. Thus, if we restrict f to be continuous, it does not greatly restrict our input space.

We are now prepared for our first approximation result.

Theorem 2.4.10. *Let G be a locally compact Hausdorff group and let $f \in C_0(G)$ be bounded. There then exists a smooth, compactly supported function $f_\epsilon \in C_c(G)$ such that $\|f_\epsilon - f\|_\infty < \epsilon$ for any $\epsilon > 0$*

Proof. Since $f \in C_0(G)$, by theorem 2.4.7 there exists an $f_{\epsilon_1} \in C_c(G)$ such that $\|f_{\epsilon_1} - f\|_\infty < \epsilon_1$ for any $\epsilon_1 > 0$. Furthermore, by theorem 2.2.9, for any $\epsilon_2 > 0$, there exists a smooth function f_{ϵ_2} given by $f_{\epsilon_1} * \psi_{\epsilon_2}$ such that $\|f_{\epsilon_2} - f_{\epsilon_1}\|_\infty < \epsilon_2$. It follows straightforwardly that

$$\|f_{\epsilon_2} - f_{\epsilon_1}\|_\infty + \|f_{\epsilon_1} - f\|_\infty < \epsilon_1 + \epsilon_2$$

Minkowski's inequality tells us that

$$\|f_{\epsilon_2} - f_{\epsilon_1} + f_{\epsilon_1} - f\|_\infty \leq \|f_{\epsilon_2} - f_{\epsilon_1}\|_\infty + \|f_{\epsilon_1} - f\|_\infty$$

so we have

$$\|f_{\epsilon_2} - f\|_\infty \leq \|f_{\epsilon_2} - f_{\epsilon_1}\|_\infty + \|f_{\epsilon_1} - f\|_\infty < \epsilon_1 + \epsilon_2$$

and since the values ϵ_1, ϵ_2 are arbitrary, we get our result. □

In light of the above, f_ϵ can be taken as a bounded, compactly supported (and smooth) function, so f_ϵ belongs to L^∞. So, we can define uniformly bounded sets of such functions by $B(a, b) = \{f : a < \|f\|_\infty < b\}$

Before continuing, we remind the reader of the idea of generalized functions (distributions):

Definition 2.4.11. Let U be an open set in a locally compact Hausdorff space X. A distribution is a linear functional $D : C_c^\infty(U) \to \mathbb{F}$.

A common notation when dealing with distributions is the inner product notation. That is, for a distribution D, $\langle D, f\rangle = D(f) = \int_U f d\mu$. There is also a common abuse of notation, which is $\langle \psi, f\rangle = \psi * f$. We now give a key result from [Tre67]:

Theorem 2.4.12 ([Tre67]). *Let U be an open subset of a locally compact Hausdorff space X. The space $C_c^\infty(U)$ is sequentially dense in the strong topology of the space of distributions on U, $\mathcal{D}(U)$, meaning that for distribution D, there is a sequence of smooth functions $\{\psi_i\}_{i=0}^\infty$ such that $\langle \psi_i, f\rangle \to D(f)$ as $i \to \infty$ for all $f \in C_c^\infty(U)$, where m is a canonical (Haar in our case) measure.*

The strong topology is sometimes referred to as the topology of uniform convergence on bounded subsets. We thus have the following:

Lemma 2.4.13. *Let U be an open subset of a locally compact Hausdorff space X and let $B_U(a,b)$ be a set of compactly supported, smooth functions functions on open set U bounded between a and b. Then for $D \in \mathcal{D}(U)$ there exists a sequence of smooth functions $\{\psi_n\} \in C_c(U)$ such that for any $\epsilon > 0$ there exists an $N \in \mathbb{N}$ such that for all $n \geq N$, we have $\|\langle \psi_n, \cdot\rangle - \langle D, \cdot\rangle\|_\infty < \epsilon$ on $B_U(a,b)$.*

The above is just a restatement of the proceeding result. In order to give our approximation, we must first understand how approximating f impacts the output. We give first the following basic fact about distributions:

Lemma 2.4.14 ([Tre67]). *Distributions are uniformly continuous. That is, given an open subset U of LCH space X, for any $\epsilon > 0$ there exists a $\delta > 0$ such that for distribution D on U, and functions f, g on U, $\|\langle D, f\rangle - \langle D, g\rangle\|_\infty < \epsilon$ whenever $\|g - f\|_\infty < \delta$.*

We can now give the main result of this section result:

Theorem 2.4.15 (Equivariant Convolutions)**.** *Let G be a LCH group and let $B_U(a,b)$ be a set of compactly supported functions on open set $U \subset G$ bounded between a and b, and let $L \in \mathcal{D}(U)$ be a G-equivariant linear map. Then for any $f \in C_0(G)$ and any $\epsilon > 0$, $\delta > 0$ there exists a smooth function $f_\delta \in C_c(G)$ such that $\|f_\delta - f\|_\infty < \delta$, and there exists a sequence of functions in $C_c^\infty(U)$, $\{\psi_j\}$ such that for all $n \geq N \in \mathbb{N}$ $\|\langle \psi_n, f_\delta\rangle - \langle L, f\rangle\|_\infty < \epsilon$. That is, $L(f)$ can always be approximated (w.r.t the uniform metric) by a cross-correlation of smooth functions.*

Proof. First for any $\delta > 0$, let f_δ be the (smooth) function such that $\|f - f_\delta\|_\infty < \delta$. Since L is a linear functional on the smooth functions on U, we can always find a $\delta > 0$ where $\|\langle L, f_\delta\rangle - \langle L, f\rangle\|_\infty < \epsilon_1$. Now by lemma 2.4.13, let $\{\psi_i\}_{i=1}$ be the sequence of smooth functions that approximates L. Since $\psi_i \in C_c^\infty(U)$ uniform continuity follows by lemma 2.2.8 so for any $\epsilon_2 > 0$ we can find a f_{δ_2} so the inequality $\|\langle \psi_j, f_{\delta_2}\rangle - \langle \psi_j, f\rangle\|_\infty < \epsilon_2$ holds. Now since $f \in B_U(a,b)$ and $\|\langle \psi_j, \cdot\rangle - \langle L, \cdot\rangle\|_\infty \to 0$ uniformly on $B_U(a,b)$, we can always find a ψ_j such that $\|\langle \psi_j, f\rangle - \langle L, f\rangle\|_\infty < \epsilon_3$ for any $\epsilon_3 > 0$ and any f in $B_U(a,b)$.

We can see that for any choice of ϵ_1 and ϵ_2 we can always pick a δ small enough so that both $\|\langle \psi_j, f_\delta\rangle - \langle \psi_j, f\rangle\|_\infty < \epsilon_2$ and $\|\langle L, f_\delta\rangle - \langle L, f\rangle\|_\infty < \epsilon_1$.

We then have the inequality

$$\|\langle \psi_j, f_\delta \rangle - \langle \psi_j, f \rangle\|_\infty + \|\langle \psi_j, f \rangle - \langle L, f \rangle\|_\infty < \epsilon_2 + \epsilon_3$$

and by Minkowski's inequality

$$\|\langle \psi_j, f_\delta \rangle - \langle \psi_j, f \rangle + \langle \psi_j, f \rangle - \langle L, f \rangle\|_\infty \leq \|\langle \psi_j, f_\delta \rangle - \langle \psi_j, f \rangle\|_\infty + \|\langle \psi_j, f \rangle - \langle L, f \rangle\|_\infty$$

so

$$\|\langle \psi_j, f_\delta \rangle - \langle L, f \rangle\|_\infty \leq \|\langle \psi_j, f_\delta \rangle - \langle \psi_j, f \rangle\|_\infty + \|\langle \psi_j, f \rangle - \langle L, f \rangle\|_\infty < \epsilon_2 + \epsilon_3$$

which tells us that $\|\langle \psi_j, f_\delta \rangle - \langle L, f \rangle\| < \epsilon_2 + \epsilon_3$ and since the ϵ values were chosen to be arbitrary, we can approximate $L(f)$ arbitrarily well. $\square$

This result can be seen as a more theoretical grounding for the use of convolutions to describe equivariant linear maps. It tells us that for almost all input functions we would care about (even in cases with analog inputs) all reasonable equivariant linear maps are given by a convolution.

2.5 Stacks of Feature Maps

In this section, we will show how our algebraic theory can be used to concisely describe equivariant convolutional networks mathematically, along with how such a framework can be used to produce parametrizations of such networks.

2.5.1 Stacks of Feature Maps Under the Regular Representation

So far we have primarily discussed results in terms of layers with a single input and output channel. In most cases however we would like to map a stack of feature maps to another stack of feature maps. That is, we would like a bounded equivariant map $L : C_c(G, \mathbb{F}^d) \to C_c(G, \mathbb{F}^k)$. We can rewrite $f \in C_c(G, \mathbb{F}^d)$ as a vector

$$\vec{f(g)} = \begin{bmatrix} f_1(g) \\ \vdots \\ f_n(g) \end{bmatrix}$$

where $f_i \in C_c(G, \mathbb{F})$. Furthermore, we can decompose this vector into an expression of the form

$$\vec{f} = \sum_{i=1}^{d} f_i \vec{e_i}$$

where the e_i form a basis of $\mathbb{F}^d$. We then consider the action of G on the right of f_i, given by $\rho_h f(g) = f(gh^{-1})$. The right action on the stack of feature maps is then given via:

$$\begin{bmatrix} f_1(gh^{-1}) \\ \vdots \\ f_n(gh^{-1}) \end{bmatrix} = \rho_h \begin{bmatrix} f_1(g) \\ \vdots \\ f_n(g) \end{bmatrix}$$

Mathematically, these "stacks of feature maps" are described as tensor products of modules:

Definition 2.5.1 (Tensor Product of Modules)**.** The tensor product of two R-modules $M \otimes_R N$ is an R-module that consists of formal R-linear combinations of tensors $m \otimes_R n$ for $m \in M$ and $n \in N$ with the following conditions:

i) $(m + n) \otimes_R p = m \otimes_R p + n \otimes_R p$

ii) $m \otimes_R (n + p) = m \otimes_R n + m \otimes_R p$

iii) $rm \otimes_R n = m \otimes_R rn$

This becomes an R-module with the multiplication rule $r(m \otimes_R n) = rm \otimes_R n$.

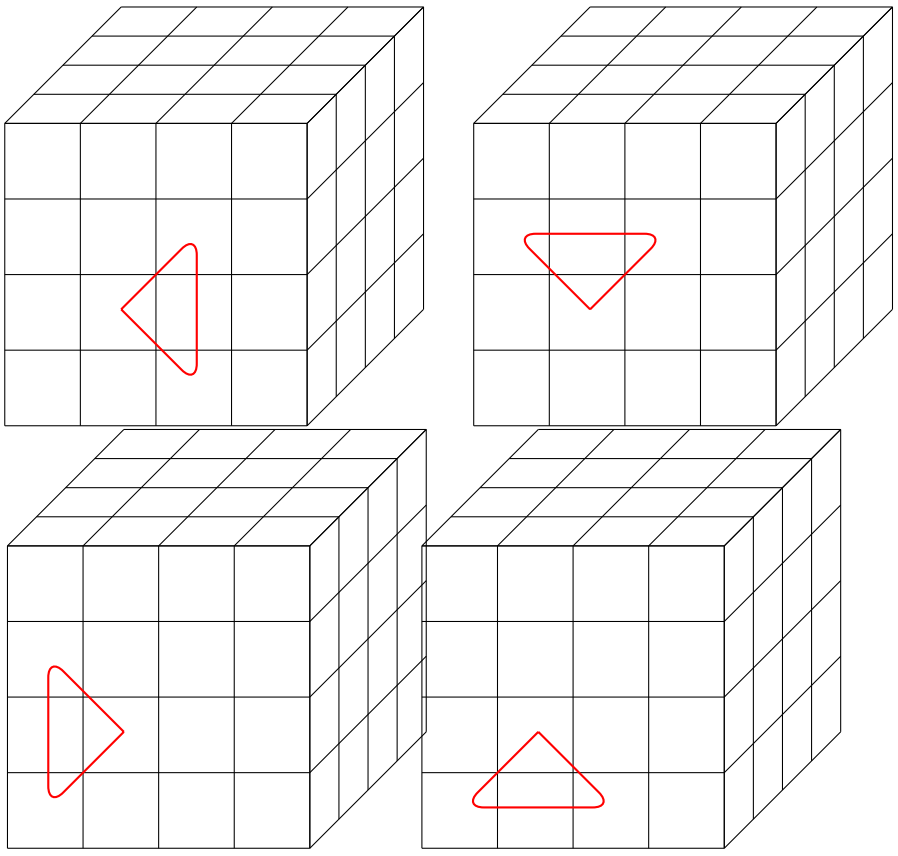

Figure 2.8: A feature map $\vec{f}$ from $C_c(\mathbb{Z}^2 \rtimes C_4) \otimes_{\mathbb{R}} \mathbb{R}^4$.

Notice that the figure above is effectively a 4-tensor. That is, it can be described as an operator $\psi_{i,j,k,l}$ where the first index i represents elements of the group C_4, the second index j is the channel depth, and k, l are the width and height. The action of C_4 on $\psi_{i,j,k,l}$ acts cyclically on the first index.

On a tensor product of modules, we can define an algebraic operation called an "extension of scalars":

Definition 2.5.2 (Extension of Scalars)**.** Let $f : R \to S$ be a homomorphism of rings R and S. Let M be an R-module and consider the tensor product

$M \otimes_R S$. Now, notice that we can give S the structure of a left R module via the (left) action $r\dot{s} = f(r)s$. Furthermore, S has a right action over itself, so S is an R, S-bimodule.

The tensor product considered above is related directly to what we call a **representation** of a group G. Representations are studied in many different contexts, and can be viewed in many different ways. For our purposes, we need only a few basic ideas from representation theory.

Definition 2.5.3 (Linear Representation)**.** Let V be a vector space and G a group. A linear representation of G on V is a homomorphism $\rho : G \to GL(V)$.

Let V be a vector space with an inner product that turns V into a Hilbert space on which we have a representation ρ for our group, a (nontrivial) subspace $W \subset V$ is called a **invariant subspace** if $\rho_g W \subseteq W$ for all $g \in G$. ρ then defines a representation on the subspace W, which is called a **subrepresentation** of ρ. If ρ admits such an invariant subspace, we say it is **reducible**, otherwise we call ρ **irreducible** (we usually refer to such a ρ as an irrep, short for irreducible representation).

Remark 2.5.4. It can be seen that every linear representation of a group G on a space X corresponds to a linear action of G on X, and vice-versa. For notational convenience, we denote the action "operator" as a subscript ρ_g. However, if we are discussing a specific realization of the action as a

representation, we will usually use the notation $\rho(h)$ to exemplify that $\rho(h)$ is a group homomorphism.

The construction $C_c(G, \mathbb{F}) \otimes_{\mathbb{F}} \mathbb{F}^k$ is the **regular representation** of our group G on the vector space $\mathbb{F}^k$. Notice that G acts on the left side of the tensor product, meaning that the representation on $\mathbb{F}^k$ is trivial. This implies that under the regular representation, group actions do not "mix" channels in a GCNN. In our tensor product description, notice that for every (closed) subgroup $H \subseteq G$ by using the extension of scalars we can take the tensor product $C_c(G, \mathbb{F}) \otimes_{C_c(H,\mathbb{F})} \mathbb{F}^k$. This is called the **induced representation** of G by H.

Remark 2.5.5. The subscript on the tensor product is omitted where in places where it should be clear from the context for notational convenience.

The induced representation is a generalization of the regular representation which is key for efficiently implementing GCNNs in practice for large groups. Before shifting our attention to the induced representation however, we will take a moment to examine how layers on "stacked" feature maps under the regular representation are described.

Consider again our G-equivariant and $\mathbb{F}$ linear map L. This extends naturally to

$$L : C_c(G, \mathbb{F}) \otimes_{\mathbb{F}} \mathbb{F}^k \to C_c(G, \mathbb{F}) \otimes_{\mathbb{F}} \mathbb{F}^d$$

which we can give by

$$L(\vec{f}) = L(\sum_{i=1}^{k} f_i \otimes \vec{e_i})$$

and evaluate as

$$[\ L(\vec{f})]\ (g)$$

We now describe the space of such maps, and show that their structure actually corresponds with the structure of normal convolutional layers that appear in deep learning.

Proposition 2.5.6. *$Hom_{C_c(G,\mathbb{F})}(C_c(G,\mathbb{F}) \otimes_{\mathbb{F}} \mathbb{F}^m, C_c(G,\mathbb{F}) \otimes_{\mathbb{F}} \mathbb{F}^n)$ is isomorphic to $\bigoplus_{i=0}^{n} Hom_{C_c(G,\mathbb{F})}(C_c(G,\mathbb{F}) \otimes_{\mathbb{F}} \mathbb{F}^m, C_c(G,\mathbb{F}))$ as right (resp. left) $C_c(G)$-modules.*

Proof. Since we have that $L \in \mathrm{Hom}(C_c(G,\mathbb{F}) \otimes_{\mathbb{F}} \mathbb{F}^m, C_c(G,\mathbb{F}) \otimes_{\mathbb{F}} \mathbb{F}^n)$ for a (regular) equivariant layer L, we can describe L as an $n \times m$ matrix of kernel functions:

$$L = \begin{bmatrix} \psi_{1,1} & \cdots & \psi_{1,m} \\ \vdots & \ddots & \vdots \\ \psi_{n,1} & \cdots & \psi_{n,m} \end{bmatrix}$$

For feature vector

$$\vec{f} = \begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix}$$

we can describe the map $L(\vec{f})$ as a type of matrix multiplication:

$$L\vec{f} = \begin{bmatrix} \psi_{1,1} & \cdots & \psi_{1,m} \\ \vdots & \ddots & \\ \psi_{n,1} & \cdots & \psi_{n,m} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix}$$

$$= \begin{bmatrix} \psi_{1,1} * f_1 + \cdots + \psi_{1,m} * f_m \\ \vdots \\ \psi_{n,1} * f_1 + \cdots + \psi_{n,m} * f_m \end{bmatrix}$$

Notice that this can always be given by an element of the space $\bigoplus_{i=0}^{n} \mathrm{Hom}_{C_c(G,\mathbb{F})}(C_c(G,\mathbb{F}) \otimes_{\mathbb{F}} \mathbb{F}^m, C_c(G,\mathbb{F}))$ which gives the desired result. □

This proposition tells us that the space of module homomorphisms between a space of m dimensional feature maps to the space of feature maps of dimension n is the same as the m-fold product of the dual module on the input space. This also lines up exactly with the behaviour of convolutional layers in practice, as every filter in a convolutional layer maps an m-dimensional input to a single output channel. Under this algebraic view, the filters are viewed as module homomorphisms which are effectively linear functionals.

If we think about regular GCNNs in the context of representation learning, it tells us something interesting. Filters in regular GCNN disentangle a feature from its orientations. In regular convolutional networks, equivariance is normally achieved by data augmentation during training. The idea is that

by training the network on transformed copies of the input data, it will learn filters that detect transformed versions of the same feature. So if filter ψ_1 detects some feature at orientation g, we have some other filter ψ_2 which detects the same feature at some other orientation h. In essence the network tries to learn a collection of filters which correspond roughly to the same object acted on by different transformations. However, in a GCNN the filters inherently contain information about transformations which impact orientation, and thus need not learn to disentangle such information. In [LBL$^+$18] they prove that one cannot learn general disentangled representations without strong inductive biases. While their work was mainly aimed at unsupervised learning, it explains the usefulness of GCNNs, as they effectively act as an inductive bias which allows some level of feature disentanglement.

2.5.2 Feature Maps and Layers Under the Induced Representation

For the remainder of this chapter we will be focusing on the induced representation. As we will show, there are two ways of thinking about the induced representation. First, one can view it as describing the action of G on functions on G by a representation created from a known representation of a subgroup H which is described the tensor product $C_c(G, \mathbb{F}) \otimes_{C_c(H,\mathbb{F})} \mathbb{F}^k$ as above. Or, as we will show, one can use it to describe the action of G on func-

tions on the coset space G/H, given by the tensor product $C_c(G/H, \mathbb{F}) \otimes_{\mathbb{F}} \mathbb{F}^k$. A treatment of this idea given in terms of local sections of vector bundles can be found in [CGW18b].

Given a group G and a subgroup H we have the canonical projection $P : G \to G/H$ and a choice of section $s : G/H \to G$. A section is a map with property that $P \circ s = e$ where e is the identity map for G/H. A section chooses a fixed representative for each coset, so for convenience we choose a section such that $s(H) = e$. Observe as well that for $u, g \in G$

$$u(s(gH))H = ugH = s(ugH)H$$

Remark 2.5.7. Given group G and subgroup H, we have that for all $gH \in G/H$, there exists a neighbourhood of U of gH and a local section $\sigma : U \to G$.

We can use this rule to understand the action of G on sections. Define the function $t : G/H \times G \to H$ via

$$us(gH) = s(ugH)t(gH, u)$$

where $u \in G$. We can see that $t(gH, u) = s(ugH)^{-1}us(gH)$.

We will use this to construct a sort of **twisted group ring** for the induced representation. Notice that we can lift functions on G/H to functions on G

via the composition $f(s(gH))$ and the left action $\lambda_{u^{-1}}$ gives us that

$$\lambda_{u^{-1}} f(s(gH)) = f(us(gH))$$

$$= f(s(ugH)t(gH,u))$$

$$= \rho_{t(gH,u)^{-1}} f(s(ugH))$$

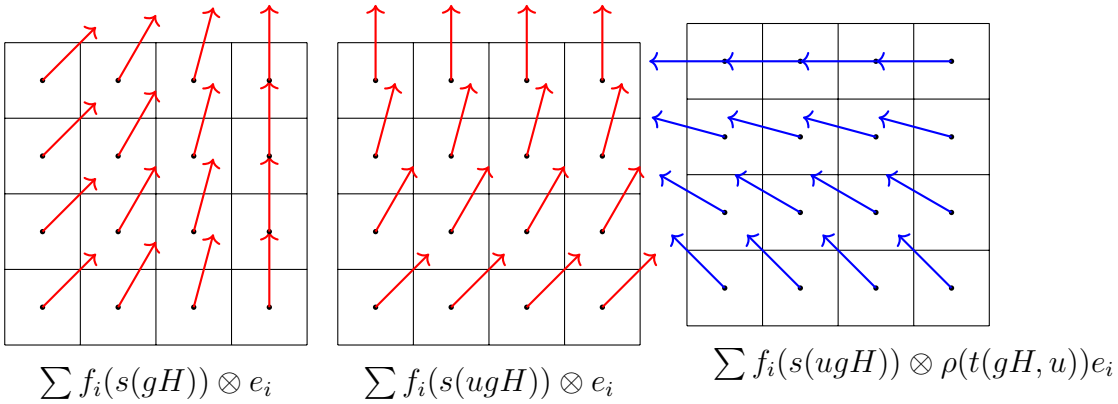

Figure 2.9: The action of G on $C_c(G/H) \otimes_{C_c(H)} V$.

Given that $s(H) = e$, we can see that

$$t(H,u) = s(uH)^{-1} us(H)$$

$$= s(uH)^{-1} u$$

this tells us then that for all $u \in G$, we can express it as

$$u = s(uH)t(H, u)$$

This gives us a way of relating our two descriptions of the induced representation. Consider the following element of $C_c(G, \mathbb{F}) \otimes_{C_c(H,\mathbb{F})} \mathbb{F}^k$

$$(\sum f_i \otimes e_i)(g) = (\sum f_i(g) \otimes e_i)$$

This can always be written as an element of $C_c(G/H, \mathbb{F}) \otimes_{\mathbb{F}} \mathbb{F}^k$

$$= (\sum f_i(s(gH)t(H, g)) \otimes e_i)$$

$$= (\sum \rho_{t(H,g)^{-1}} f_i(s(gH)) \otimes e_i)$$

and considering the operator $\rho_{t(H,g)^{-1}}$ acts via convolution with the point mass at $t(H, g)$ we get that

$$= (\sum \delta_{t(H,g)^{-1}} * f_i(s(gH)) \otimes e_i) \tag{2.1}$$

This tells us that our choice of section and the fact that the group H commutes across the tensor product allows us to map elements of $C_c(G/H, \mathbb{F}) \otimes_{\mathbb{F}} \mathbb{F}^k$ to $C_c(G, \mathbb{F}) \otimes_{C_c(H,\mathbb{F})} \mathbb{F}^k$ by a "twisting factor". Notice as well that $C_c(G/H)$ "inherits" the action of H from $\mathbb{F}^k$, since the representation ρ no longer commutes across the tensor product.

The consideration of functions on the coset space G/H is more geometrically meaningful, and generally computationally simpler to implement, however, it tends to add symbolic complications when working with it mathematically. Thus, we will generally stick to considering functions on G for simplicity.

Remark 2.5.8. Note that when working with these maps, our representation/action ρ is taken to be fixed.

The following proposition elucidates the structure of the G- equivariant maps on the induced representation:

Proposition 2.5.9. *Let G be a locally compact Lie group, and let $\mathbb{F}$ be either $\mathbb{R}$ or $\mathbb{C}$. Consider the $\mathbb{F}$-linear, G-equivariant map*

$$L : C_c(G, \mathbb{F}) \otimes_{C_c(H,\mathbb{F})} \mathbb{F}^k \to C_c(G, \mathbb{F}) \otimes_{C_c(H,\mathbb{F})} \mathbb{F}^d$$

Letting $\vec{f}$ be an element of $C_c(G, \mathbb{F}) \otimes_{C_c(H,\mathbb{F})} \mathbb{F}^k$, L is given by

$$[L(\vec{f})](g) = (\sum_i \kappa_i * f_i \otimes \psi_i e_i)(g)$$

$$= \sum_i (\kappa_i * f_i)(g) \otimes \psi_i e_i$$

where each ψ_i is H-equivariant on the vector space $\mathbb{F}^k$ and each κ_i is a G equivariant map $\kappa_i : G \to \mathbb{F}$.

Proof. First observe that for any linear map ψ on vector space $\mathbb{F}^k$ we have

that

$$\psi\vec{f} = \begin{bmatrix} \psi_{1,1} & \cdots & \psi_{1,m} \\ \vdots & \ddots & \\ \psi_{n,1} & \cdots & \psi_{n,m} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix}$$

and since each $\psi_{i,j}$ is a scalar, it commutes with the f_k, so we get

$$= \begin{bmatrix} f_1\psi_{1,1} + \cdots + f_m\psi_{1,m} \\ \vdots \\ f_1\psi_{n,1} + \cdots + f_m\psi_{n,m} \end{bmatrix}$$

Notice now that this can be written as

$$= \sum_i \begin{bmatrix} f_i\psi_{1,i} \\ \vdots \\ f_i\psi_{n,i} \end{bmatrix}$$

$$= \sum_i f_i \begin{bmatrix} \psi_{1,i} \\ \vdots \\ \psi_{n,i} \end{bmatrix}$$

and under the tensor product, we have

$$= \sum_i f_i \otimes \begin{bmatrix} \psi_{1,i} \\ \vdots \\ \psi_{n,i} \end{bmatrix}$$

$$= \sum_i f_i \otimes \psi e_i$$

Now, if ψ is H-equivariant, if $\rho_2(h)$ is a representation of H, since H commutes with the tensor product we have that

$$= \rho_2(h)(\sum_i f_i \otimes \psi e_i)$$

$$= \sum_i f_i \otimes \rho_2(h)\psi e_i$$

and since ψ is H-equivariant we have that

$$= \sum_i f_i \otimes \psi \rho_1(h) e_i$$

Notice that we can decompose ψ as a linear combination of basis matrices

$$\psi = \sum_{i,j} \psi_{i,j} M_{i,j} \tag{2.2}$$

and since the space

$$\mathrm{Hom}_{C_c(G,\mathbb{F})}(C_c(G,\mathbb{F}) \otimes_{C_c(H,\mathbb{F})} \mathbb{F}^k, C_c(G,\mathbb{F}) \otimes_{C_c(H,\mathbb{F})} \mathbb{F}^d)$$

is a right (resp. left) $C_c(G,\mathbb{F})$-module, given a collection of functions $\kappa_{i,j} \in$

$C_c(G, \mathbb{F})$, we can construct a new map

$$L = \sum_{i,j} \psi_{i,j} M_{i,j} \kappa_{i,j}$$

which can be rewritten as

$$L = \begin{bmatrix} \psi_{1,1}\kappa_{1,1} & \dots & \psi_{1,m}\kappa_{1,m} \\ \vdots & \ddots & \vdots \\ \psi_{n,1}\kappa_{n,1} & \dots & \psi_{n,m}\kappa_{n,m} \end{bmatrix}$$

We then have that, almost identical to the regular case

$$L\vec{f} = \begin{bmatrix} \psi_{1,1}(\kappa_{1,1} * f_1) + \dots + \psi_{1,m}(\kappa_{1,m} * f_m) \\ \vdots \\ \psi_{n,1}(\kappa_{n,1} * f_1) + \dots + \psi_{n,m}(\kappa_{n,m} * f_m) \end{bmatrix}$$

and factoring through our tensor product we get

$$= \sum_j (\sum_i \kappa_{i,j} * f_i \otimes \psi e_i)$$

We claim that if each κ is G equivariant, then L itself is G-equivariant. Now, let $\pi : G \to \mathbb{F}$ be a homomorphism of G into $\mathbb{F}$. If we consider the right G-action on an element of $C_c(G, \mathbb{F}) \otimes_{C_c(H,\mathbb{F})} \mathbb{F}^k$:

$$= \rho_{g^{-1}} \sum_i f_i \otimes \psi e_i$$

since any g can be written as $g = \zeta h$ where ζh is an element of a left H-coset determined by fixed section s we have

$$\rho_{g^{-1}} \sum_i f_i \otimes e_i = \rho_{(\zeta h)^{-1}} \sum_i f_i \otimes e_i$$

$$= \rho_{h^{-1}\zeta^{-1}} \sum_i f_i \otimes e_i$$

$$= \rho_{h^{-1}} \sum_i (f_i * \pi)(\zeta) \otimes e_i$$

$$= \sum_i (f_i * \pi)(\zeta) \otimes \rho_1(h) e_i$$

Considering the map L on the above, we see that

$$L(\sum_i (f_i * \pi)(\zeta) \otimes \rho_1(h) e_i) = \sum_j (\sum_i \kappa_{i,j} * (f_i * \pi)(\zeta) \otimes \psi \rho_1(h) e_i) \quad (2.3)$$

and since convolution is associative we have that $\kappa_{i,j} * (f_i * \pi)(\zeta) = (\kappa_{i,j} * f_i) * \pi(\zeta)$. This tells us that L is G-equivariant when each κ is G-equivariant and ψ is an H-equivariant element of $Hom(\mathbb{F}^k, \mathbb{F}^d)$.

To see that all such maps L must be of this form, notice that the space

$$\mathrm{Hom}_{C_c(G)}(C_c(G, \mathbb{F}) \otimes_{C_c(H,\mathbb{F})} \mathbb{F}^k, C_c(G, \mathbb{F}) \otimes_{C_c(H,\mathbb{F})} \mathbb{F}^d)$$

must admit a basis belonging to the space $\mathrm{Hom}_{\mathbb{F}}(\mathbb{F}^k, \mathbb{F}^d)$, which is the space of $k \times d$ matrices over $\mathbb{F}$. Furthermore, if M_i is an element of such a basis,

then M_i must be H-equivariant since given the action of G, as seen above we must have that

$$\rho_{g^{-1}} \sum_i f_i \otimes M_i e_i = \rho_{(\zeta h)^{-1}} \sum_i f_i \otimes M_i e_i$$

$$= \sum_i (f_i * \pi)(\zeta) \otimes \rho_2(h) M_i e_i$$

$$= \sum_i (f_i * \pi)(\zeta) \otimes M_i \rho_1(h) e_i$$

Finally we can see that since

$$\mathrm{Hom}_{C_c(G)}(C_c(G,\mathbb{F}) \otimes_{C_c(H,\mathbb{F})} \mathbb{F}^k, C_c(G,\mathbb{F}) \otimes_{C_c(H,\mathbb{F})} \mathbb{F}^d)$$

is a right (resp. left) $C_c(G)$-module with a basis of H-equivariant, $\mathbb{F}$-linear maps, it is a free module, where all elements can be written as a linear combination of the form

$$L = \sum M_i \kappa_i \tag{2.4}$$

where $\kappa_i \in C_c(G)$, as claimed.

□

2.6 Invariant Subspaces and Feature Representations

Here, we provide a more useful version of the parametrization theorem seen in the previous section. describing a more explicit method of such a parametrization of GCNN layers using the algebraic structure.

2.6.1 Algebraic Preliminaries

Under the view of an equivariant layer L as a module homomorphism, the kernel of L (in the algebraic sense) $Ker(L)$ must be an ideal of $C_c(G) \otimes \mathbb{F}^n$ (we omit the tensor subscript since this fact is unaffected by our choice of group representation). Earlier we defined invariant subspaces, along with reducible and irreducible representations. These objects are related to the kernel functions of our induced representation layer L, and provide us with a lot of useful information about induced GCNNs and how they can be constructed. To begin with, we recall some basic definitions from the theory of modules.

Definition 2.6.1. A module M is called a **simple** module if it contains no proper submodules.

Definition 2.6.2. A module M is called **semisimple** if M can be written

as the direct sum of simple modules.

When considering a feature space $C_c(G) \otimes_{C_c(H)} V$, the vector space V which we define the representation ρ of H on has a special (though obvious) structure.

Definition 2.6.3. A (left) G-module is an abelian group M (or module) with a map $\pi : G \times M \to M$ that describes a group action such that $g(a + b) = ga + gb$.

Now observe that a vector space is a module over a field, so the H action given by ρ turns V into an H-module. The decomposition of ρ into irreps gives a decomposition of V as invariant subspaces. If V is a finite dimensional complex Hilbert space, and H can be represented on the unitary operators of V (that is, H admits a unitary representation), such a decomposition always exists [Fol95]. By the definition of invariant subspaces, the H-module over V is then a direct sum of simple modules, and is thus semisimple. Each V_i in the decomposition of V is a cyclic H-module (generated by a single element) meaning that V is finitely generated as an H-module.

Suppose then that vector space V has the decomposition as invariant subspaces $\oplus_i V_i$ with the group action of H then given by the representation $\rho(h) = \oplus_i \rho_i(h)$. Letting $\vec{v} = \oplus_i \vec{v}_i$ we get

$$\rho(h)\vec{v} = \oplus_i \rho_i(h)\vec{v}_i$$

which tells us that the action $\rho(h)$ is determined on each $\vec{v}_i$ independently by the sub-representation $\rho_i(h)$. While this is relatively unremarkable from a mathematics point of view, it has interesting implications in a machine learning context.

Suppose we have a vector of features at some point in G/H which is given as $\vec{f}(gH) = \vec{v} = \oplus_i \vec{v}_i$. Each $\vec{v}_i$ can be thought of as describing a feature at the point. For each $\vec{v}_i$ we have a corresponding irreducible representation ρ_i of H which describes how $\vec{v}_i$ transforms under the action of H. Each $\vec{v}_i$ transforms according to its own independent representation. In the regular case, "features" were individual channels in a stack. In the induced case, our channels are invariant subspaces and the irreducible representation on them describes how the feature represented by that channel transforms under the group action.

Similarly to [CW16b], we will discuss how this is related to the idea of "**capsules**" from [HKW11]. Capsules can be thought of as similar to filters in convolutional layers, as they learn to recognize particular visual entities in their input. However, they learn to recognize them over a domain of possible transformations of the input, such as lighting, translation, etc. Such a capsule outputs the probability of detection of their specific visual entity, along with an **instantiation vector**. The instantiation vector encodes the information about the visual entity detected such as the position, deformations, and precise pose relative to some canonical version of the visual entity.

One particular type of these instantiation vectors that is studied more than others is a **pose vector** which encodes the position of the detected entity. The feature space defined by the output of a layer L of n capsules is then a tuple $(f_1, \ldots, f_n)$ where each f_i is a tuple $(p_j, \vec{w}_j)$ where p_i is the probability of detection of some visual entity in the pose described by $\vec{w}_j$.

Irreps can be seen to do something similar. For the feature vector $\vec{v}_i$ the irrep ρ_i describes how $\vec{v}_i$ transforms relative to the feature it detects. So if the feature corresponding to $\vec{v}_i$ is transformed by some $h \in H$, we have that the feature vector transforms as $\rho_i(h)\vec{v}_i$, so each invariant subspace encodes information about the pose (relative to H) and the detection of its associated feature.

The main advantage of using irreps over the classical capsules proposed in [HKW11] is that classical capsules must learn invariant/equivariant viewpoints to geometric transformations like rotations and reflections. Irreps however are by nature equivariant to the group they represent. Furthermore, capsules must be trained via a complicated and inefficient routing algorithm, where as GCNNs under the induced representations can be trained like any convolutional network. GCNNs can also be adapted to broader domains than vision, where capsule networks were designed based off of the human visual system, and thus are not well-suited to non-vision problems.

2.6.2 Refining the Structure of G-Equivariant Linear Maps on Induced Representations

Admitting a decomposition as the direct sum irreducible representations and the semisimplicity of such a representation can be used to help parametrize equivariant layers under the induced representation. Of use to us is the famous lemma of Schur:

Lemma 2.6.4 (Schur's Lemma for Modules)**.** *Let M, N be R-modules and let $f : M \to N$ be a homomorphism. The following are then true:*

1. *If M is simple, f is either trivial or injective.*
2. *If N is simple, f is either trivial or surjective.*
3. *If both are simple, f is either trivial or an isomorphism.*

Proof. For the first, since M is simple, the kernel of f is either trivial or M. Now, if N is simple, $f(M)$ must be either trivial or $f(M) = N$. Combining these two immediately gives the third. □

Next we give a basic result of linear algebra, extended naturally to the module case:

Lemma 2.6.5. *Let, M, N be left (resp. right) R modules and let L be a homomorphism. If I is a subspace of M, then $L(I)$ is a subspace of N.*

Thinking of invariant subspaces as H-modules, it follows that invariant subspaces are mapped to invariant subspaces. We will now use Schur's lemma to prove two results about the structure of our equivariant maps. We first consider the simple case where $\mathbb{F} = \mathbb{C}$, and use Schur's lemma on modules to give the classical version of Schur's lemma for group representations. We then discuss how Schur's lemma for modules allows one to generalize to cases where one makes minimal assumptions about the structure of $\mathbb{F}$.

Theorem 2.6.6. *Let $V_1 = V_2$ be complex vector spaces with irreducible H-representations $\rho_1 = \rho_2$. Then the only maps H-equivariant linear maps between V_1 and V_2 are scalar multiples of the identity, and the trivial map.*

Proof. Suppose that our representations are equivalent on the invariant subspaces $V_1 = V_2$. If we allow ourselves to work over the field $\mathbb{C}$ for any linear map ψ between $V_1 = V_2$ we can find an eigenvalue λ. Letting $\psi' = \psi - \lambda I$, the eigenvector $\vec{t}$ for eigenvalue λ has the property that

$$\psi'\vec{t} = \psi\vec{t} - \lambda I\vec{t} = 0$$

Now, by Schur's lemma, the kernel of ψ' is either trivial or it must be all of V_1. The kernel clearly contains the eigenvector $\vec{t}$, so it cannot be trivial and must be the entire space, meaning that $\psi' = 0$ is the trivial map. Thus we have that

$$\psi = \lambda I$$

□

As mentioned, we can actually generalize this to fields other than $\mathbb{C}$. We first define a simple module M over a ring R as being **absolutely simple** if it is still simple under the algebraic closure of R. Now let $\mathbb{F}$ be either $\mathbb{R}$ or a scalar extension of $\mathbb{R} \subset \mathbb{C}$. We can then say a representation ρ of a group G on vector space V with scalar field $\mathbb{F}$ is an **absolutely irreducible representation** of G (over $\mathbb{F}$) if ρ is irreducible over $\mathbb{C}$ (or, more generally, if it is irreducible over the algebraic closure of $\mathbb{F}$). Now notice that as a consequence of Schur's lemma on modules, the endomorphism ring of a simple module must be a division ring. Notice then that if we consider V an absolutely simple G-module with scalar ring $C_c(G, \mathbb{F})$, the endomorphism ring of V can be taken as being isomorphic to $C_c(G, \mathbb{F})$.

We can use these results to tell us how we can construct certain classes of equivariant kernels for absolutely irreducible representations. Considering a representation $\oplus_{i=1}^{n} \rho_i$ define for each ρ_i a **multiplicity** hyperparameter m_i which determines how many times the representation ρ_i appears in the output. Letting $I_{i,j}$ be jth copy of the identity matrix on the invariant subspace V_i of ρ_i we then give a collection of matrices $(\alpha_{i,1} I_{i,1}, \ldots, \alpha_{i,m_i} I_{i,m_i})$ where the $\alpha_{i,j}$ are scalar multiples which are our learnable parameters. We then vertically stack the matrices $(\alpha_{i,1} I_{i,1}, \ldots, \alpha_{i,m_i} I_{i,m_i})$, creating another matrix A_i. Doing this for each ρ_i, we then compute $\oplus_{i=1}^{n} A_n$, which is a block

diagonal matrix. This is our H-equivariant linear map. Notice that the individual identity matrices $I_{i,j}$ effectively form a basis for our H-equivariant linear maps. Furthermore, taking the scalar multiples $\alpha_{i,j}$ to instead be equivariant coefficient functions $\kappa_{i,j} : G \to \mathbb{F}$ we can construct fully G-equivariant kernels.

In the previous section, we gave a result about the structure of G-equivariant maps under the induced representation. We now give a simple extension of this result which describes how one can parametrize GCNNs in practice.

Theorem 2.6.7 (GCNN Parameterization Theorem)**.** *Let G be a locally compact Lie group, with H a compact subgroup of G. Consider now two feature spaces $C_c(G) \otimes_{C_c(H)} V, C_c(G) \otimes_{C_c(H)} W$ which are spaces of representations of G induced by H on the $\mathbb{F}$-vector spaces V, W. the space of G-equivariant $\mathbb{F}$-linear maps between these feature spaces is given as*

$$Hom_{C_c(G)}(C_c(G) \otimes_{C_c(H)} V, C_c(G) \otimes_{C_c(H)} W)$$

Let L be such an equivariant map and let $\vec{f} \in C_c(G) \otimes_{C_c(H)} V$. L is approximated (w.r.t the uniform metric) by a linear combination

$$L(\vec{f})(g) = (\kappa_1(g)M_1 + \dots \kappa_k(g)M_k)\vec{f}$$

Where each M_i are linearly independent, H-equivariant elements of $Hom(V, W)$, and each κ_i is a G-equivariant coefficient function $\kappa_i : G \to \mathbb{F}$.

Furthermore, if all the H-invariant subspaces of V are absolutely irreducible over the field $\mathbb{F}$, then the M_i are block matrices which contain scalar multiples of the identity.

Proof. Since we assume the invariant subspaces are absolutely irreducible over $\mathbb{F}$, by Schur's Lemma it follows that for a given invariant subspace V_i of V that an H-equivariant map M_i to invariant subspace W_i of W is either trivial or an isomorphism and by theorem 2.6.6, if this map is non-trivial it is a scalar multiple of the identity matrix. Considering proposition 2.5.9, if $\kappa_i : G \to \mathbb{F}$ is G-equivariant, then $\kappa_i M_i$ is also G-equivariant.

Given that all the (non-trivial) G-equivariant maps between invariant subspaces are of the given form, the proposed block diagonal structure follows simply from basic properties of matrix multiplication. Furthermore, by theorem 2.2.9, we can approximate arbitrary generalized functions (w.r.t the uniform metric) via convolution with compactly supported continuous functions on G, so the result follows. □

Remark 2.6.8. While we consider the result in terms of absolutely irreducible representations, even if representations are not absolutely irreducible, a version of the above theorem still holds, where the identity matrix blocks are replaced simply by matrices which define isomorphisms between invariant subspaces.

In relation to the result of [CGW18a] (theorem 2.4.1), the result here gives a method for generating GCNN layers (given you know the appropriate representations), but also shows that we can always approximate distributional kernels, meaning that theorem 2.4.1 holds, given the small amendment that "always can be written as" is changed to "always can be approximated by in the uniform metric". While this might not be of particular importance in practice, it makes working with GCNNs mathematically simpler, as working with spaces compactly supported continuous functions is in general easier than working with spaces of distributions.

Chapter 3

Applying the Algebraic Theory to SE(3)

3.1 SE(3)-Equivariant Networks for Quantum Chemistry

Computational problems in quantum chemistry (and physics in general) have a lot of potential for deep learning applications, due to the complex nature of physical systems. Indeed, most problems one wishes to solve with computational techniques in physics have NP computational complexity, so having neural networks that provide good approximations of these computational

techniques would be massively beneficial since a single forward pass (generally) has polynomial time evaluation. Furthermore, much of quantum physics essentially boils down to a study of symmetries of a system, so such problems are a natural place where one would benefit from a network which included such symmetries inherently.

Many of the models proposed to deal with problems arising in the quantum realm model the data as "point clouds". Point clouds can be thought of as functions which assign "feature vectors" to some finite set of points in space. There is already a large body of work studying the application of deep learning to such problems. Here, we leverage our algebraic theory to simply generalize Schnet [SSK$^+$18] and produce a network with similar properties to tensor field networks [TSK$^+$18], but which uses a method of "spatial normalization" to achieve translation invariance of the entire molecule, as opposed to the method of tensor field networks, which use Clebsch-Gordon coefficients to achieve translation equivariance. Thus our solution is less general than [TSK$^+$18] but is much simpler.

Remark 3.1.1. Note that we do not perform experiments for the network architecture we explore, as this meant to simply show how one can utilize the algebraic theory to construct networks for different problems.

3.1.1 Molecules and Group Actions

Here we will consider the problem of predicting properties of molecules. Molecules can be viewed (under mild assumptions) as a point cloud, which is to say that a molecule M is represented by a collection of points $(x_1, \ldots, x_n)$ in $\mathbb{R}^3$, each of which has an associated atom represented by feature vector $\vec{a}_i \in \mathbb{R}^d$ for some arbitrary dimension d. We will assume for now that the values of $\vec{a}_i$ are properties of the atom which do not depend on its location in space (this is generally the case in practice). This means $\vec{a}_i$ includes things like atomic weight, charge, atomic number, etc. We could also take an approach where the values of the atom vector $\vec{a}_i$ are given by some parametrized embedding function which takes in the atomic number/symbol and outputs a learned vector representation for the atom, as is done in [SSK$^+$18].

Example 3.1.2. Let's consider the a water molecule H_2O as a point cloud. For simplicity, we will work using spherical coordinates first, then convert to cartesian. Treating the oxygen molecule as the origin of $\mathbb{R}^3$, define the corresponding hydrogen atoms sit on the xy plane. Picking one to align with the x axis, the angle between the two hydrogen atoms is about $104.45°$. The distance between each hydrogen atom and the oxygen atom is about 95.84 pm (picometers). Using pm as our scale, We can define the positions of the three atoms in spherical coordinates as $(0, 0, 0)$ for the oxygen atom, $(95.84, 0, 0)$ for one of the hydrogen atoms, and $(95.84, 0, 104.45°)$.

Now let us assume our feature vector consists simply of the atomic weight and atomic number of the atom. The water molecule, converted to cartesian coordinates, can then defined by the point cloud as

$$(((0,0,0),(8,15.99)),((0,0,95.84),(1,1)),((92.80,0,-23.9),(1,1))) \quad (3.1)$$

To frame molecules as elements of a feature space in a computationally natural way, we can think of a molecule as being defined by simple functions on singleton sets in $\mathbb{R}^3$. So if a molecule M is defined by a collection of atoms $(\vec{a}_1,\ldots,\vec{a}_n)$ defined at points $(x_1,\ldots,x_n)$, we can represent it as a vector of simple functions $\vec{M} = \sum_i^n \vec{a}_i\chi_{x_i}$ where χ_{x_i} is the indicator function on the singleton for x_i. We would require these functions have the same sort of invariance as the atomic feature vectors in the point cloud.

While this view of molecules as simple functions of finite support is necessary for practical implementations, it has the problem that given measure μ, the integral $\int_U M(x)d\mu(x) = 0$. However, we can avoid this problem in the algebraic theory. To do this, let χ_{x_i} be an indicator function in the linear combination $\vec{M} = \sum_i^n \vec{a}_i\chi_{x_i}$. Now, let $B^r(x_i)$ be an open ball of radius r about x_i. We then construct a new continuous function χ'_{x_i} such that for all $x \in B^r(x_i)$, $1 \geq \chi'_{x_i}(x) > 0$, with $\chi'_{x_i}(x) = 1$ exactly when $x = x_i$, and $\chi'_{x_i}(x) = 0$ for all $x \notin B^r(x_i)$. By Urysohn's lemma, such a function always exists. While this induces some error, notice that by theorem 2.2.9, this

error can be made arbitrarily small, which justifies our choice of using simple functions of finite support for practical purposes.

So far, we have written molecules as elements of the space $C_c(\mathbb{R}^3) \otimes_{\mathbb{R}} \mathbb{R}^d$. The results of section 2.5.2 tell us that this space can be lifted to $C_c(SE(3)) \otimes_{C_c(SO(3))} \mathbb{R}^d$, and thus it has a $SE(3)$ group action. Letting $\vec{M} \in C_c(SE(3)) \otimes_{C_c(SO(3))} \mathbb{R}^d$, we have that $\vec{M}$ decomposes according to the invariant subspaces of $\mathbb{R}^d$ which correspond to our choice of irreducible representations of $SO(3)$ From a physics perspective, our choice of invariant subspaces encodes the natural symmetry transformations associated to different properties of the atom.

3.1.2 Practical Considerations

To build an equivariant convolutional network in $\mathbb{R}^3$, one might suspect that we could take a discretization approach by defining discrete "voxels" in $\mathbb{R}^3$, meaning our input features would belong to the space $C_c(\mathbb{Z}^3) \otimes_{\mathbb{R}} \mathbb{R}^d$, so they can be implemented on a digital computer. And indeed, this paradigm has been explored using both regular representations (in [WB18]) and induced representations (in [WGW$^+$18]). These approaches are well-suited for certain tasks, however, they are sub-optimal for predicting molecular properties from atomic structures. If we consider the regular representation approach of [WB18], we immediately see that our network cannot be equivariant to arbitrary rotations, since the space $\mathbb{Z}^3$ only admits rotational symmetry groups

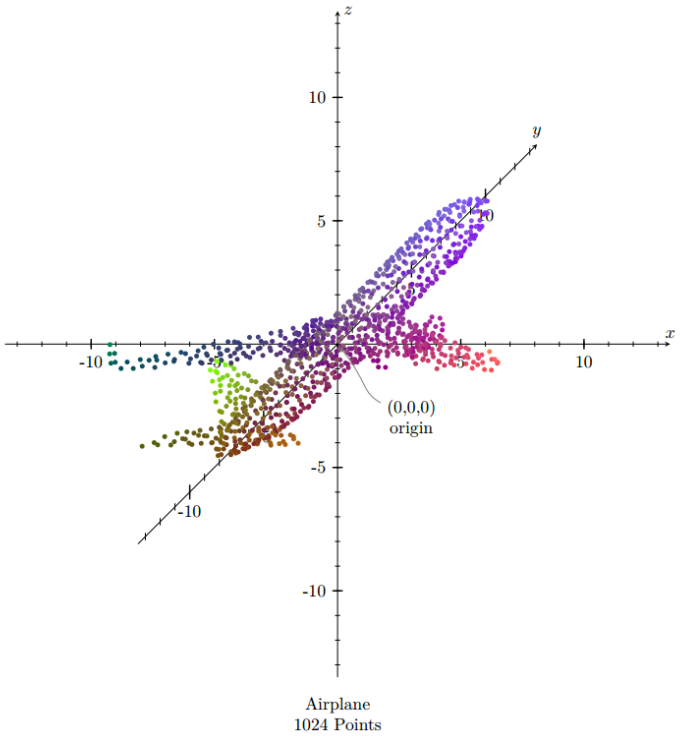

Figure 3.1: An example of a general point cloud.

with right-angle rotations. Under the steerable paradigm, this ceases to be a problem. This is done essentially by considering the input as a function on $C_c(\mathbb{R}^3) \otimes_{\mathbb{R}} \mathbb{R}^c$ which is sampled at points $(x_1, x_2, \ldots, x_n)$ where the x_i form a lattice, which is the approach of [WGW+18]. This is efficient for problems where uniform sampling of the input is an acceptable prior, which is usually the case when the point clouds are sufficiently "dense". This is the case for problems like 3D object recognition, however, when dealing with molecular structures, the point cloud is much more sparse, so uniform sampling will induce relatively significant interpolation errors.

The other approach one might take is to take the atomic data of a molecule $\vec{M} = (\vec{a}_1, \ldots, \vec{a}_n)$ and apply the each one some learnable weight matrix T, meaning we would compute $(T\vec{a}_1, \ldots T\vec{a}_n)$. Since this operation is performed independent of the position of the atoms, it is invariant. While some useful information can be extracted from knowing just the atomic composition, it is also important to understand the relative spatial location of the atoms in a molecule, and how the atoms might interact. Furthermore, invariance with respect to rotations and translations is sufficient to compute things like thermodynamic free energy, but predicting things like force fields produced by the molecule is not possible, since the field transforms equivariantly with respect to translations and rotations of molecules in space. Our goal is then to make a network that allows for equivariant transformations of features of a molecule.

Along with the problems with the discretization approaches mentioned above, there is also a problem with how to allow a network to learn equivariant properties of a molecule, when the input to the network consists of properties of atoms that do not transform equivariantly, but instead transform according to the trivial representation. In practice, this is almost always the case, as atoms are generally represented as vectors $\vec{a}$ taken according to some learned embedding function $E : \mathbb{N} \to \mathbb{R}^d$ which takes as input the atomic number outputs a vector which represents the atom. Since our input is a vector of functions $\vec{a}$ where the action of H on $\vec{a}$ is trivial, this tells us that the rep-

resentation ρ^0 of H decomposes as the direct sum of trivial representations $\oplus_{i=1}^{d} \rho_i^0$, which means the feature space of molecules decomposes into d one-dimensional subspaces which transform trivially. Furthermore, our algebraic theory says that in GCNN layers, invariant subspaces must map to isomorphic copies of themselves. So, we appear to be "stranded" with the trivial representation. As we will show however, we can escape this representation by equivariantly embedding our data and using a version of the continuous filter convolutions of [SSK$^+$18].

3.1.3 Equiverifying Atomic Data

Before we can find a method for "equiveriantly embedding" our data, we make note of a well-known property of the irreducible representations of $SO(3)$. $SO(3)$ has one irreducible representation in every odd dimension. Suppose we represent each atom a_i in the molecule by its atomic number. Let $d \geq 1$ and define the embedding map $E : \mathbb{Z} \to \mathbb{R}^{3d}$. We then have that $E(a_i) = \vec{a}_i \in \mathbb{R}^{3d}$ is a feature vector for an atom a_i at position $\vec{x}_i \in \mathbb{R}^3$.

Letting ρ_1 be a non-trivial representation of $SO(3)$ on $\mathbb{R}^3$, and let ρ_2 be a representation on $\mathbb{R}^{3d}$. By theorem 2.6.7 we can create an $SO(3)$-equivariant map $\psi : \mathbb{R}^3 \to \mathbb{R}^{3d}$ such that $\rho_2(h)\psi\vec{x} = \psi\rho_1(h)\vec{x}$ where $\vec{x} \in \mathbb{R}^3$. That is, if we think about $\mathbb{R}^3$ as an $SO(3)$-invariant subspace, ψ maps this space equivariantly into $\mathbb{R}^{3d}$. We can now describe the equivariant embedding:

Definition 3.1.3. Let $M = ((\vec{x}_1, \ldots, \vec{x}_n), (a_1, \ldots, a_n))$ be a molecule where the a_i are the atomic numbers of the element at position $\vec{x}_i \in \mathbb{R}^3$. Letting $\psi : \mathbb{R}^3 \to \mathbb{R}^{3d}$ (for some $d \geq 1$) be an SO(3)-equivariant map and $E : \mathbb{Z} \to V$ be an embedding into vector space V. For each a_i in M, we then compute the equivariant embedding as $\vec{a}_i = \|E(a_i)\|_2(\psi\vec{x}_i)$ for each a_i.

Letting $\vec{a}$ be an embedding, we consider now the value given by

$$(\Psi\vec{x}) \odot \vec{a}$$

where $\odot$ is the elementwise product. If we then consider Ψ as a two argument function via $(\Psi\vec{x} \odot \vec{a}) = \Psi(\vec{x}, \vec{a})$, we can define the action on Ψ as $\rho_h\Psi(\vec{x}, \vec{a}) = \Psi(\rho_1(h)\vec{x}, \rho_2(h)\vec{a})$.

This map gives us a way of creating features that transform equivariantly with respect to rotations, however, it is not necessarily equivariant to translations. Furthermore, it still only considers atoms within the molecule independently.

First we tackle the problem of translation equivariance. Inducing true translation equivariance would require solving for the coefficient functions κ which we presented in the previous chapter. Instead of solving this in the most general case, we instead create a method for inducing translation invariance (as invariance is a special case of equivariance) within the point cloud which

preserves the geometric relationships between atoms. This can be done by introducing a "normalizing function" which induces translation invariance into our network by a change of basis. Given the points $(x_1, \ldots, x_n)$, we can compute a point C_m which we will call the "center of mass". This can be done in a multitude of ways, by either computing the actual center of mass of the atom, finding a point minimizes the distance between itself and all atoms in the molecule, etc. Since this point is always defined by the spatial location of the molecule, we can take C_m to be the origin. We can then express each atomic position relative to this origin by applying some translation to each point. We will refer to this as **spatial normalization**, denoting the operation on molecule M as $\mathcal{S}(M)$.

We can now give a definition of an equivariant layer which is equivariant to rotations, invariant to translations, and can model atomic interactions.

Definition 3.1.4 (Hadamard Layer)**.** Let M be a molecule with atoms $(\vec{a}_1, \ldots, \vec{a}_n)$ at points $(x_1, \ldots, x_n)$, which we assume is spatially normalized. The Hadamard map is given by

$$\mathcal{N}(M) = \mathcal{N}((x_1, \ldots, x_n), (\vec{a}_1, \ldots, \vec{a}_n))$$

$$= (\vec{a}'_1, \ldots, \vec{a}'_n)$$

where $\vec{a}'_i$ is defined by

$$\vec{a}'_i = \sum_j^n \Psi(x_i - x_j, \vec{a}_j)$$

This is effectively an H-equivariant version of the continuous filter convolution of [SSK+18], and is strictly more general than their invariant continuous filters. One issue with these Hadamard layers is that they cannot recombine feature maps. That is, the dimension of the atomic feature vectors does not change. However, we can give an equivariant version of the atomwise layers of [SSK+18] as a straightforward consequence of the algebraic theory:

Definition 3.1.5 (Equivariant Atomwise Layers)**.** Let $\psi : \mathbb{R}^m \to \mathbb{R}^k$ be an $SO(3)$ equivariant linear map. We can then compute an atomwise layer via $\psi((\vec{a}_1, \ldots, \vec{a}_n)) = (\psi\vec{a}_1, \ldots, \psi\vec{a}_n)$.

Theorem 2.6.7 tells us how to construct these equivariant atomwise layers. For each invariant subspace V_i of $\mathbb{R}^d$, let k be the desired multiplicity. For each $j = 1, \ldots, k$ we construct a block matrix $N_{i,j}$ which contains a single block containing an $n \times n$ identity matrix where $n = \mathrm{Dim}(V_i)$. We would then normally construct an $SE(3)$-equivariant coefficient function $\kappa_{i,j} : G \to \mathbb{R}$. However, in our considerations, since our inputs to an atomwise layer are always spatially normalized, notice that we can simply take $\kappa_{i,j}$ to be a constant, so $\kappa_{i,j}N_{i,j}$ is equivariant to $SO(3)$. The weight matrix ψ in an atomwise layer is then the linear combination of matrices of the form $\kappa_{i,j}N_{i,j}$.

Example 3.1.6. Let $M = (\vec{a}_1, \ldots, \vec{a}_p)$ be the output of a Hadamard layer so that M is spatially normalized. Suppose each $\vec{a}_i$ in M decomposes into two 3-dimensional invariant subspaces V_1, V_2. Suppose ψ is an equivariant atomwise layer with multiplicity 2 for V_1 and multiplicity 1 for V_2. We then

get the following three block basis matrices:

$$N_{1,1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$N_{1,2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$N_{2,1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

We then have that

$$\psi = \kappa_{1,1} N_{1,1} + \kappa_{1,2} N_{1,2} + \kappa_{2,2} N_{2,2}$$

$$= \begin{bmatrix} \kappa_{1,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \kappa_{1,1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \kappa_{1,1} & 0 & 0 & 0 \\ \kappa_{1,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \kappa_{1,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \kappa_{1,2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \kappa_{2,1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \kappa_{2,1} & 0 \\ 0 & 0 & 0 & 0 & 0 & \kappa_{2,1} \end{bmatrix}$$

We now show an example for a Hadamard layer.

Example 3.1.7. Suppose for simplicity that we have a molecule M which consists of two atoms with $\vec{a}_1$ and $\vec{a}_2$ in $\mathbb{R}^6$ at spatial locations $\vec{e}_1$ and $\vec{e}_2$, which are the canonical basis vectors in $\mathbb{R}^3$. Now, suppose the $\vec{a}_i$ transform trivially. We would like to construct a Hadamard layer which outputs feature vectors which transform equivariantly according to $SO(3)$.

First, we must define a map $\Psi : \mathbb{R}^3 \to \mathbb{R}^6$ which transforms equivariantly. As in the previous example, theorem 2.6.7 tells us how to do this. Letting $\kappa_{i,j}$ be coefficient functions as before, we can take

$$\Psi = \begin{bmatrix} \kappa_{1,1} & 0 & 0 \\ 0 & \kappa_{1,1} & 0 \\ 0 & 0 & \kappa_{1,1} \\ \kappa_{1,2} & 0 & 0 \\ 0 & \kappa_{1,2} & 0 \\ 0 & 0 & \kappa_{1,2} \end{bmatrix}$$

We then have that

$$\vec{a}_1' = \Psi(\vec{e}_1 - \vec{e_2}) \odot \vec{a}_2$$

and

$$\vec{a}_2' = \Psi(\vec{e}_2 - \vec{e_1}) \odot \vec{a}_1 \tag{3.2}$$

which gives us that M' is described by $((\vec{e}_1, \vec{a}_1'), (\vec{e}_2, \vec{a}_2'))$

3.1.4 Discussing Induced SE(3)-networks

Notice that this equivariant atomwise layer is exactly how we described GCNN layers under the induced representation. Furthermore, notice that for both the atomwise and Hadamard layers, our results about the structure of equivariant maps under the induced representation give us a very straightforward way of creating such layers. As we mentioned, we are essentially considering the features of the atoms as transforming according to the three dimensional irreducible representation of $SO(3)$. In every Hadamard layer, the map Ψ has a straightforward construction as a block diagonal of d scalar multiples of the 3×3 identity matrix. These scalar multiples are exactly the learnable parameters of the Hadamard layer. The atomwise layers have a similar structure, with a slight modification.

Earlier we artificially restricted ourselves to features of atoms/molecular systems which are invariant to rotations and translations, since this is usually how such systems are represented. However, this need not be the case. The input features may transform according to any representation of $SO(3)$. The Hadamard layer requires minimal modification for this to function. Assuming one knows the invariant subspaces of the non-trivial representations of the input, one computes the Hadamard product by computing the elementwise multiplication on only the invariant subspace, thus preserving the transformation properties of the other features. The atomwise layer generalizes

straightforwardly.

We can use these layers to build up a core of an $SE(3)$ equivariant network. These networks are similar to the tensor field networks presented in [TSK+18], however, they utilize Clebsch-Gordon coefficients to produce a network which is equivariant to rotations and translations, while we use a simpler method to make our system invariant to translations, but which is still equivariant to rotations, and can produce equivariant output fields, predict atomic positions, etc, and provides much of the same utility as tensor field networks without as much explicit computation.

3.2 Relationship to Other Work, Possibilities for Future Works, and conclusion

Our results are also very similar to those of [CGW18a] and are thus a strict generalization of [KT18a]. One of their key results is that feature spaces are sections of the vector bundle associated to the principal H-bundle for H a subgroup of G. In other terms, they show that a feature map is a vector field $\mathcal{V}$ on the coset space G/H. They describe the action of G via the induced representation as first acting by translation on the coordinates of G/H, and then acting on the vectors $\vec{v}$ of the vector field. Symbolically, the transformation of $\mathcal{V}(x)$ by $g \in G$ is given by $\rho_g \mathcal{V}(g^{-1}x)$. This transformation is

effectively the same as the composition rule in our twisted group ring. The relationship between the two works is in a similar spirit to the "Serre-Swan Theorem" which which relates vector bundles on compact spaces to (projective) R-modules where R is a commutative ring. Our main contribution using our algebraic theory is providing an algorithm to compute the basis for the space of H-equivariant linear maps (intertwiners) between induced representations, which allows for much easier development of GCNNs. The work [LW20] can be seen as providing a method which computes the "coefficient functions" in our algebraic theory (which also implicitly computes the basis maps provided by our algorithm). This solution is more general than the one provided here, however the algebraic framework provides additional insight into the structure equivariant layers, and could potentially make implementations easier.

The authors of [CGW18a] (among others) produced a subsequent paper to make networks equivariant to gauge transformations [CWKW19]. Without diving deep into any formalism, we will brief informal description of how this fits in an algebraic theory. If M is a smooth manifold, and E is a $\mathbb{F}^k$ vector bundle on M, the space of smooth sections of E denoted by $\Gamma(E)$ is a $C^\infty(M)$-module. Swan's theorem states that this module is finitely generated and projective on $C^\infty(M)$. We then describe a representation of the "structure group" G for the manifold turning $\Gamma(E)$ into a G-module. Gauge equivariant layers are then described by maps between such modules.

The recent work [LW20] is also very closely related to work done here as mentioned previously. Their work uses tools from harmonic analysis to provide a general way of explicitly solving the kernel constraint from [CGW18a]. In contrast, we simply provide a way of computing a basis for H-equivariant linear maps, and supply a straightforward constraint to to make G-equivariant maps based on a choice of scalar functions.

The algebraic view also opens up possibilities for examining the structure of feature spaces in novel ways. In particular, feature spaces define sheaves, which gives a route for the application of algebraic geometry for deep learning. Previously it has been observed that there is a strong relationship between algebraic geometry and classical statistical learning theory [Wat09]. However, the work on the applications towards deep learning has been sparse, and the relationship between feature spaces and sheaves (while obvious in both the geometric and algebraic framework) has not been previously explored. One other route for generalization is to examine more general algebras, and define neural networks on them. This allows us to think about data that lives on more exotic domains which might not have the same symmetry structure as the modules examined here.

In conclusion we have presented here an algebraic framework for equivariant convolutional neural networks, and examined the structure of feature spaces and layers as modules of a group ring. It is hoped that these observations can be used to better understand the nature of feature representations in deep

learning, and potentially provide a route for developing less data intensive algorithms.

Bibliography

[AHK19] Brandon Anderson, Truong-Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks, 2019.

[AML18] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *CoRR*, abs/1803.10091, 2018.

[Anh18] Anh Vo. Deep learning – computer vision and convolutional neural networks, 2018. [Online; accessed May 10, 2020].

[AP77] S. K. Arora and I. B. S. Passi. Annihilator ideals in twisted group rings. *Journal of the London Mathematical Society*, s2-15(2):217–220, 1977.

[BCV12] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[BL18] Richard Brown and Gerton Lunter. An equivariant bayesian convolutional network predicts recombination hotspots and accurately resolves binding motifs. *Bioinformatics (Oxford, England)*, 35, 11 2018.

[BM17] Alberto Bietti and Julien Mairal. Group invariance, stability to deformations, and complexity of deep convolutional representations. *J. Mach. Learn. Res.*, 20:25:1–25:49, 2017.

[Bol13] Arkady Bolotin. Computational complexity and the interpretation of a quantum state vector. *arXiv: Quantum Physics*, 2013.

[BRT18] Benjamin Bloem-Reddy and Y. Teh. Neural network models of exchangeable sequences. 2018.

[CCC+19] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, M. Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91:045002, 2019.

[CGKW18] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *CoRR*, abs/1801.10130, 2018.

[CGW18a] Taco Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. *CoRR*, abs/1811.02017, 2018.

[CGW18b] Taco S. Cohen, Mario Geiger, and Maurice Weiler. Intertwiners between induced representations (with applications to the theory of equivariant neural networks). *CoRR*, abs/1803.10743, 2018.

[Chr17] Chris Olah and Alexander Mordvintsev and Ludwig Schubert. Feature visualization, 2017. [Online; accessed May 11, 2020].

[CW16a] Taco S. Cohen and Max Welling. Group equivariant convolutional networks. *CoRR*, abs/1602.07576, 2016.

[CW16b] Taco S. Cohen and Max Welling. Steerable cnns. *CoRR*, abs/1612.08498, 2016.

[CWKW19] Taco S. Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral CNN. *CoRR*, abs/1902.04615, 2019.

[Dee21] Deepmind. Deepmind alphafold2, 2021.

[DFK16] Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. *CoRR*, abs/1602.02660, 2016.

[Eat07] Morris L. Eaton. *Chapter 6: Topological Groups and Invariant Measures*, volume Volume 53 of *Lecture Notes–Monograph Series*, pages 184–232. Institute of Mathematical Statistics, Beachwood, Ohio, USA, 2007.

[ESF+18] Alexander S. Ecker, Fabian H. Sinz, Emmanouil Froudarakis, Paul G. Fahey, Santiago A. Cadena, Edgar Y. Walker, Erick Cobos, Jacob Reimer, Andreas S. Tolias, and Matthias Bethge. A rotation-equivariant convolutional neural network model of primary visual cortex, 2018.

[FA+91] William T Freeman, Edward H Adelson, et al. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991.

[Fol95] G. Folland. A course in abstract harmonic analysis. 1995.

[Fol13] G.B. Folland. *Real Analysis: Modern Techniques and Their Applications*. Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts. Wiley, 2013.

[Fre00] D.H Fremlin. *Measure Theory*. University of Essex, 2000.

[FWFW20] Fabian B. Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks, 2020.

[GBC16] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016.

[GD14] Robert Gens and Pedro M Domingos. Deep symmetry networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and

K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2537–2545. Curran Associates, Inc., 2014.

[HKW11] Geoffrey Hinton, Alex Krizhevsky, and Sida Wang. Transforming auto-encoders. volume 6791, pages 44–51, 06 2011.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[KLT18] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch-gordan nets: a fully fourier space spherical convolutional neural network, 2018.

[Kon18] Risi Kondor. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *CoRR*, abs/1803.01588, 2018.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[KT18a] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups, 2018.

[KT18b] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. *ArXiv*, abs/1802.03690, 2018.

[Lav19] Lavanya Shukla. Designing your neural networks, 2019. [Online; accessed May 10, 2020].

[LBD+89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[LBL+18] Francesco Locatello, Stefan Bauer, Mario Lucic, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. *CoRR*, abs/1811.12359, 2018.

[Lou18] Louis de Vitry. Word embeddings, 2018. [Online; accessed May 9, 2020].

[LW20] Leon Lang and Maurice Weiler. A wigner-eckart theorem for group equivariant convolution kernels, 2020.

[MVKT16] Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. *CoRR*, abs/1612.09346, 2016.

[na20] nLab authors. Serre-swan theorem, August 2020.

[OM14] Edouard Oyallon and Stéphane Mallat. Deep roto-translation scattering for object classification. *CoRR*, abs/1412.8659, 2014.

[Ope20] OpenAI. Openai microscope, 2020. [Online; accessed May 11, 2020].

[RSP17] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Equivariance through parameter-sharing, 2017.

[Rud87] Walter Rudin. *Real and Complex Analysis, 3rd Ed.* McGraw-Hill, Inc., USA, 1987.

[SLM+15] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.

[SSK+18] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller. Schnet – a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

[TQD+19] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *CoRR*, abs/1904.08889, 2019.

[Tre67] F. Treves. *Topological Vector Spaces, Distributions and Kernels*. ISSN. Elsevier Science, 1967.

[TSK+18] Nathaniel Thomas, Tess Smidt, Steven M. Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *CoRR*, abs/1802.08219, 2018.

[Vap95] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.

[Wat09] Sumio Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge University Press, USA, 2009.

[WB18] Daniel E. Worrall and Gabriel J. Brostow. Cubenet: Equivariance to 3d rotation and translation. *CoRR*, abs/1804.04458, 2018.

[WC19] Maurice Weiler and Gabriele Cesa. General e(2)-equivariant steerable cnns. *CoRR*, abs/1911.08251, 2019.

[WGTB16] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic networks: Deep translation and rotation equivariance. *CoRR*, abs/1612.04642, 2016.

[WGW+18] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *CoRR*, abs/1807.02547, 2018.

[WHS17] Maurice Weiler, Fred A. Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. *CoRR*, abs/1711.07289, 2017.

[ZYQJ17] Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Oriented response networks. *CoRR*, abs/1701.01833, 2017.

Appendix A

Basic Deep Learning

One of the key components in this work is the idea of deep learning. This section provides a brief overview of the very basics to give context to the coming sections. For a thorough treatment of the basics of modern deep learning, [GBC16] serves as the standard introduction.

A.1 Neural Networks

To build up the idea of deep learning, suppose we have an n dimensional vector $\vec{x}$ from some feature space X which we would like to classify into one of d classes. We can then define a $d \times n$ weight matrix W and compute the

value:

$$W\vec{x} = \vec{o}$$

where $\vec{o}$ is the output vector. Now, most have encountered the sigmoid function before:

$$S(x) = \frac{1}{1 - e^{-x}}$$

We apply this function to our output vector

$$S(\vec{o}) = S\left(\begin{bmatrix} o_1 \\ \vdots \\ o_d \end{bmatrix} \right)$$

$$= \begin{bmatrix} S(o_1) \\ \vdots \\ S(o_d) \end{bmatrix}$$

which returns a vector such that every entry is bounded above by 1 and below by 0. We then classify sample $\vec{x}$ by $\operatorname{argmax} S(\vec{o})$ (here the argmax returns the index of the maximum value in a vector). Our algorithm can "learn" to classify samples by finding values of our weight matrix that increase the probability that the output $\operatorname{argmax} S(\vec{o})$ corresponds to the labels y_i of the samples $\vec{x}_i$ in our training data.

Deep learning can be viewed as a sort of extension of this idea. The example above has a direct analogue which is the fundamental building block of deep

learning, the **fully connected** (or **dense**) layer.

Definition A.1.1 (Fully Connected Layer)**.** A fully connected layer in a neural network is a parametrized function described by an affine transformation. That is, a fully connected layers consists of a weight matrix W and a bias term b. Let l be an index and have N_l denote our layer. Given layer input $\vec{x}$, we have

$$N_l(x) = W_l\vec{x} + b_l$$

Usually network layers are also equipped with a non-linear **activation** function σ, which means that

$$\sigma \circ N_l(x) = \sigma(N_l(\vec{x})) = \sigma(W\vec{x} + b)$$

There are many different types of "layers" besides the basic fully connected ones, many of which are introduced in any elementary text on deep learning. Generally however, layers are effectively linear maps between vector spaces. We use fully connected layers here pedagogically so we can give a definition of deep networks.

Definition A.1.2 (Deep Network)**.** A deep neural network is a composition of fully connected layers and activation functions. That is, if we denote our m layer network as N, then for input $\vec{x}$ we have

$$N(\vec{x}) = \sigma_m \circ N_m \circ \sigma_{m-1} \circ N_{m-1} \circ \ldots \circ \sigma_0 \circ N_0(\vec{x})$$

for affine linear maps N_i and activation functions σ_i.

Remark A.1.3. This definition generalizes by simply replacing the affine linear transformations with other linear transformations.

Deep networks have proven that they are incredibly powerful, especially in domains with plentiful data. This success, as mentioned earlier is due to the ability of deep networks to learn useful representations of our data. If we split a network into two separate parts, denoting all but the final layer in the network as R, and denoting the final layer as a linear classifier C, if R can learn to represent the data in a way that linearly separates causes of variation between classes, C will have a much easier time learning the classification task. This idea of deep networks learning feature representations will come up frequently in our discussions and interpretations of GCNNs.

A.1.1 Training Neural Networks

To train such networks, we define a **loss function** which, given a set of labels Y is usually of the form $L : Y \times Y \to \mathbb{R}$. Letting N_θ denote our network with θ describing our free network parameters, we train our network by searching for parameters θ that minimizes the function $L(N_\theta(\vec{x}), y)$ where y is the true label of sample $\vec{x}$. The idea is that by minimizing our loss on a large enough training dataset, it will be approximately minimal on the entire space X. That is, by minimizing the loss our system should learn the proper labelling

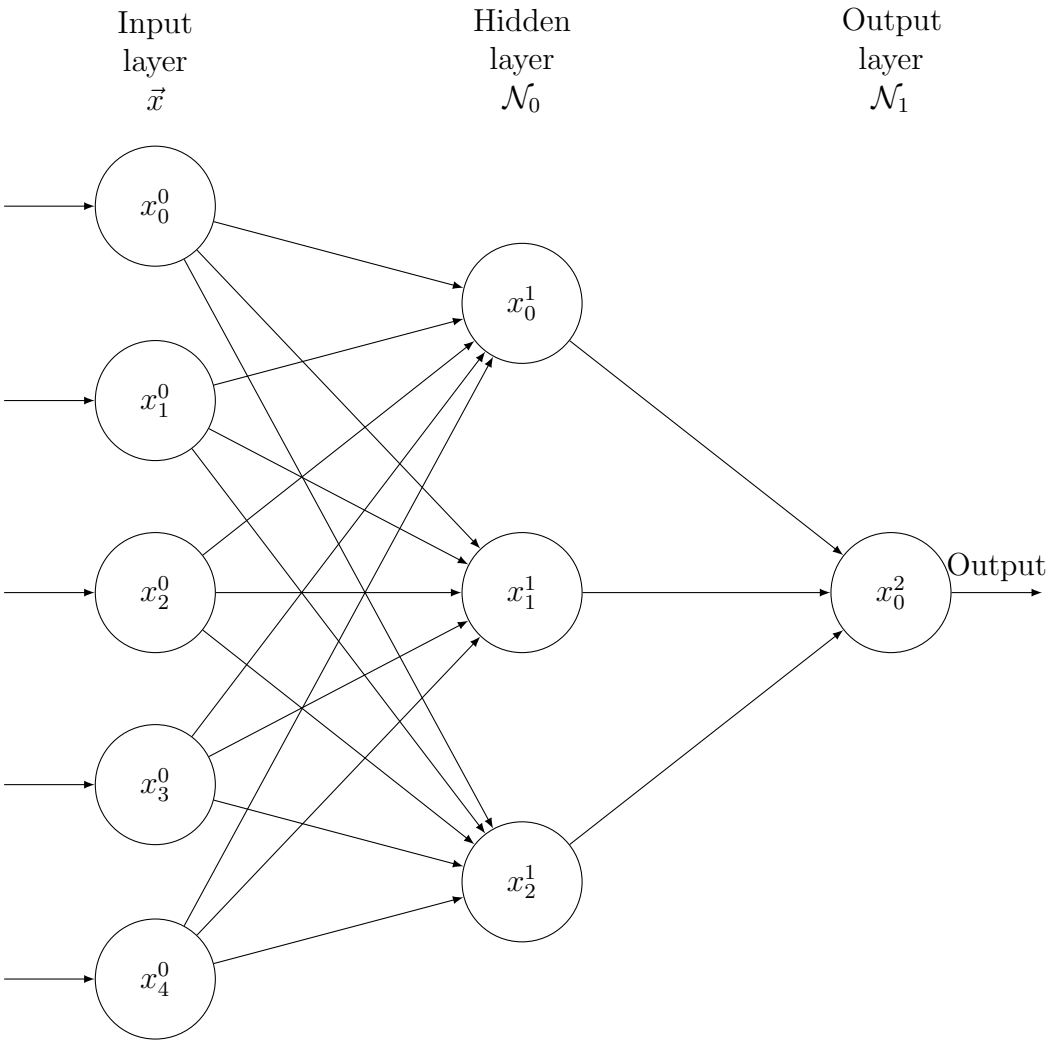

Figure A.1: A network with a single hidden layer.

of unseen samples from the space X.

This can be made more rigorous by the idea of **empirical risk minimization**, which is one of the key ideas of statistical learning theory. First, observe that there is some joint probability distribution $P(x, y)$ between feature space X and output space Y, and that our training set consists of some number n of samples $(x_1, y_1), \ldots, (x_n, y_n)$. Modelling as a probability distribution allows us to model uncertainty of classification of certain features, where the true class might be fundamentally unpredictable (this phenomena is called Bayes Error).

The risk R associated with N_θ is then

$$R(N_\theta) = \mathbb{E}[\ L(N_\theta(x), y)]$$

$$= \int L(N_\theta(x), y) dP(x, y)$$

so the ultimate goal of learning is to find a set of parameters θ^* such that $\theta^* = \text{argmin}_\theta\, R(N_\theta)$. We generally don't know the distribution $P(x, y)$ so we cannot directly compute $R(N_\theta)$, though we can compute an approximation called the empirical risk $R_{emp}(N_\theta)$ on our training set of size n given by

$$R_{emp}(N_\theta) = \frac{1}{n} \sum_{i=0}^{n} L(N_\theta(x_i), y_i)$$

We then attempt to find a set of parameters $\hat{\theta}$ such that

$$\hat{\theta} = \operatorname*{argmin}_{\theta} R_{emp}(N_\theta)$$

Examining this idea, we can see why it makes sense. Recall the general idea of the "law of large numbers" (of which there are many such laws). The key idea is that as you sample larger and larger sets, your expected values on the sets will converge towards the true value. This tells us that we can approximate the true risk arbitrarily well given access to the appropriate amount of data.

While we have explained what we are trying to optimize, we haven't really explained how we plan on finding the appropriate parameters. This is done in neural networks primarily using **gradient descent** methods (there are other methods, though they are uncommon in practice). Suppose we have a batch $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ of our dataset of size m. Let $N_\theta(x)$ be our network on input x when our network has parameters θ, and L our loss function. The (stochastic) gradient update is then given by

$$\theta_{t+1} = \theta_t - \lambda \frac{1}{m} \nabla_{\theta_t} \sum_{i=1}^{m} L(N_{\theta_t}(x_i), y_i)$$

where ∇_{θ_t} is the gradient with respect to the parameters and λ is the learning rate. The idea is that our loss function defines some surface in high

dimensions, and by following the negative of the gradient, we can find a set of parameters that minimizes our loss. The algorithm for actually computing the gradient is called backpropagation (simply referred to as backprop). We refer the interested reader to [GBC16] for an in-depth explanation of the algorithm.

Appendix B

Groups

Central to the the work presented here is the concept of a group.

Definition B.0.1 (Group)**.** A group G is defined as a set equipped with a binary operation $\cdot$ such that

- For all $a, b \in G$ we have that $a \cdot b \in G$
- For all $a, b, c \in G$ we have that $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- There exists a unique element $e \in G$ such that for all $a \in G$ we have $a \cdot e = a$.
- For all $a \in G$ there exists an element $a^{-1} \in G$ such that $a \cdot a^{-1} = e$.

We also utilize the idea of group homomorphisms:

Definition B.0.2 (Group Homomorphism)**.** Let G, H be groups and let $\psi : G \to H$ be a function. We call ψ a group homomorphism if for all a, b in G, we have $\psi(a \cdot b) = \psi(a) \cdot \psi(b)$. If such a map is bijective, it is a then a group isomorphism.

When discussing how to construct GCNNs, we also make heavy use of the idea of cosets:

Definition B.0.3. The (right) cosets of a subgroup H of G (a subgroup is a subset of G which is itself a group under the binary operation of G) are given by the sets

$$gH = \{gh : h \in H\}$$

for all $g \in G$.

In the examples used in the main work, we make use of a few particular groups, which we will define here.

Definition B.0.4 (Orthogonal Group)**.** The **orthogonal group** $O(n)$ is the group of distance preserving transformations of n-dimensional Euclidean space which preserve a fixed point. The **special orthogonal group** $SO(n)$ is a subgroup of $O(n)$ which preserves the orientation of the space.

Definition B.0.5 (Euclidean Group)**.** The **Euclidean group** $E(n)$ is the group of transformation which preserves distances between every pair of

points. The **special Euclidean group** $SE(n)$ is a subgroup of $E(n)$ which also preserves the orientation of the space.

We also consider the **cyclic groups**, which we can think of simply as the finite subgroups of $SO(n)$.

Appendix C

Basic Measure Theory

Many of the definitions used in this appendix are adapted from the text ([Fol13]), which is recommended for the interested reader.

Definition C.0.1 (σ-algebra)**.** A **σ-algebra** over a space X is a collection $\mathcal{M}$ of subsets of X that is closed under countable unions and finite compliments which contains both $\emptyset$ and X. The σ-algebra generated by the open sets of X is called the **Borel σ-algebra**.

Definition C.0.2 (Measures)**.** A **measure** is a function $\mu : \mathcal{M} \to [\,0, +\infty]$ such that $\mu(\cup_{i=1}^{\infty} E_i) = \sum_{i=0}^{\infty} \mu(E_i)$ and $\mu(\emptyset) = 0$. If $\mu(E) < \infty$ for all $E \in \mathcal{M}$, then μ is called **finite**. If $E = \cup_{i=0}^{\infty} E_i$ and $\mu(E_j) < \infty$ for each E_j, then μ is called **σ-finite**. If for each E such that $\mu(E) = \infty$ there exists an $F \in \mathcal{M}$ such that $F \subset E$ and $0 < \mu(F) < \infty$ then μ is called **semifinite**. A measure

is called **signed** if it is allowed to take on negative values.

Definition C.0.3 (Inner and Outer Regularity)**.** Let μ be a Borel measure on X. We call μ **outer regular** on E if

$$\mu(E) = \inf\{\mu(U) : E \subset U, U\, open\}$$

and **inner regular** if

$$\mu(E) = \sup\{\mu(K) : K \subset E, K\, compact\}$$

If μ is both inner and outer regular on all Borel sets, we call μ a **regular Borel Measure**.

Definition C.0.4 (Radon Measure)**.** A **Radon measure** on X is a Borel measure that is finite on all compact sets, outer regular on all Borel sets, and inner regular on all open sets.

Definition C.0.5 (Total Variation)**.** Let μ be a complex-valued measure. Given a μ-measurable set E, let π be the set of all partitions of E into a countable number of disjoint measurable subsets. The total variation of μ is then given via

$$|\mu|(E) = \sup_{\pi} \sum_{A \in \pi} |\mu(A)|$$

Our work here is focused on locally compact Hausdorff groups, which admits a specific type of radon measure called the **Haar measure** (among others).

Definition C.0.6. A left Haar measure on G is a Radon measure μ such that $\mu(gE) = \mu(E)$ where E is a Borel set and $g \in G$. The right Haar measure is then a measure μ where $\mu(Eg) = \mu(E)$

A frequent idea used in this thesis is that of L^p functions. We now remind the reader of the basic definitions of such functions and the spaces associated with them:

Definition C.0.7. For $1 \leq p < \infty$ the $L^p(\mu)$ norm of a function f is given by

$$\|f\|_p = \Big(\int_X |f|^p d\mu \Big)^{1/p}$$

If $p = \infty$ we define the norm as

$$\|f\|_\infty = \inf\{C \geq 0 : |f(x)| < C \text{for almost all } x\}$$

If $\|f\|_p < \infty$, we then say f is an L^p function.

We now address a point that might seem restrictive, and show that it in fact causes little problem. We consistently assume that whatever locally compact group we are working with is also Hausdorff. Recall that a space X is called T_1 if for any two points x, y, each belongs to a neighbourhood that does not contain the other. A space is Hausdorff if for all x, y in X there exists open sets U, V such that $U \cap V = \emptyset$ and $x \in U, y \in V$. We now give the following theorem from [Fol13]:

Theorem C.0.8. *Let G be a topological group. If G is T_1, it is Hausdorff. If it is not T_1, let H be the closure of $\{e\}$ in G. Then G/H is a Hausdorff topological group under the quotient topology.*

To see how this solves our problem, recall a function $f : X \to Y$ is called a **measurable function** if for all measurable sets E of Y, the preimage $f^{-1}(E)$ is a measurable set in X. Now, the definition of the quotient topology is that under a projection $P : G \to G/H$, a set $S \in G/H$ is open if and only if its preimage in G is open. Considering the closed subgroup about the identity H, notice that for the identity $e' \in G/H$, the inverse image of $\{e'\}$ is closed, since it is the subgroup H. Then, since the topology on a group is translation invariant, all singleton sets are closed in G/H. Since we are considering radon measures, all closed sets on G are measurable sets. Letting f be a Borel measurable function, consider the preimage of some point z so we can have $x \in f^{-1}(z)$. x belongs to some coset of H, so $x \in gH$ for some g, so we must have $xH = gH$ and so $f^{-1}(x) = xH$, so f is constant on cosets. This means that the function f can be taken on G/H instead of G, so we can always work with a Hausdorff space.

In our analysis we will mainly utilize the space of compactly supported continuous functions on locally compact Hausdorf groups $C_c(G)$, along with its dual space $C_0(G)$ which is the space of **continuous functions that vanish at infinity**. Our reason for choosing these spaces is that $C_c(G)$ is dense in

the space $L^p(\mu)$ for (any) radon measure μ and any $1 \leq p < \infty$, which we will prove. Recall that a **simple function** on a measurable space is a function $f = \sum_i a_i \chi_{E_i}$ where χ_{E_i} is the characteristic function of the measurable set E_i. In order to prove our desired result, we first state the following two theorems from [Fol13]:

Theorem C.0.9 (([Fol13])). *The simple functions are dense in L^p for $1 \leq p \leq \infty$.*

Theorem C.0.10 (([Fol13])). *A radon measure μ is inner regular on all of its σ-finite sets.*

We also require the following lemma:

Theorem C.0.11 (Urysohn's Lemma for LCH Spaces ([Fol13])). *Let X be LCH space. If $K \subset U \subset X$ where K is compact and U is open, there exists an $f \in C(X, [\ 0, 1]\ )$ such that $f = 1$ on K and $f = 0$ outside a compact subset of U.*

Proposition C.0.12 (([Fol13])). *If μ is a radon measure on LCH space X, $C_c(X)$ is dense in $L^p(\mu)$ for $1 \leq p < \infty$.*

Proof. Since the L^p simple functions are dense in L^p it suffices to show that for any Borel set E with $\mu(E) < \infty$, the characteristic function χ_E can be approximated in the L^p norm by functions of $C_c(X)$.

Given $\epsilon > 0$ we can choose a compact set $K \subset E$ and an open set U containing E such that $\mu(U \backslash K) < \epsilon$ and by Urysohn's lemma we can choose $f \in C_c(X)$ such that $\chi_K \leq f \leq \chi_U$. This says that $\|\chi_E - f\|_p < \epsilon^{\frac{1}{p}}$ so we are done. $\square$

Appendix D

Linear Functionals

In linear algebra, there exists the notion of the (topological) **dual space** of a vector space V, usually denoted V^* which is the space of linear maps $M : V \to \mathbb{F}$ where $\mathbb{F}$ is the underlying field of space V. The elements of this dual space are called **linear functionals**. A linear functional f is called a **positive linear functional** if for all positive $\vec{v}$ (meaning positive in all the arguments), $F(\vec{v}) \geq 0$.

For the space $C_c(G, \mathbb{F})$, we have two ways of constructing this dual module. Considering it first as a module over itself, we have the dual module $\mathrm{Hom}_{C_c(G,\mathbb{F})}(C_c(G, \mathbb{F}), C_c(G, \mathbb{F}))$. Similarly, considering $C_c(G, \mathbb{F})$ as a module with scalars $\mathbb{F}$ we have the dual module $\mathrm{Hom}_{\mathbb{F}}(C_c(G, \mathbb{F}), \mathbb{F})$.

Recall that a linear map $T : X \to Y$ between two normed vector spaces is called **bounded** if there exists a scalar C such that $\|Tx\| \leq C\|x\|$ for all $x \in X$. Furthermore, such a function is continuous. We now prove the following well-known result:

Proposition D.0.1. *A bounded linear functional B can be expressed as the difference between two positive linear functionals.*

Proof. First, define $B^+(f) = \sup\{B(g) : g \in L^p, 0 \leq g \leq f\}$ for $f \geq 0$. We would like to show that this is a bounded linear functional. First, notice that for scalar c,

$$\begin{aligned} cB^+(f) &= c\sup\{B(g) : g \in L^p, 0 \leq g \leq f\} \\ &= \sup\{cB(g) : g \in L^p, 0 \leq g \leq f\} \\ &= \sup\{B(cg) : g \in L^p, 0 \leq g \leq f\} \end{aligned}$$

If $0 \leq g_i \leq f_i$, then we have that $B(g_1) + B(g_2) = B(g_1 + g_2)$ and by the definition of the supremum we get $B(g_1 + g_2) \leq B^+(f_1 + f_2)$. Taking the supremum over all g_i we then get that $B^+(f_1) + B^+(f_2) \leq B^+(f_1 + f_2)$.

Now, suppose $0 \leq h \leq f_1 + f_2$ and set $g_1 = \sup(h - f_2, 0)$ and $g_2 = \inf(h, f_2)$. It follows then that $g_1 + g_2 = h$ and $0 \leq g_i \leq f_i$ and so $B(h) = B(g_1) + B(g_2) \leq B^+(f_1) + B^+(f_2)$. This is true for all $h \in L^p$, it follows that $B^+(f_1 + f_2) = B^+(f_1) + B^+(f_2)$. □

Measures on a space are related to (positive) linear functionals on the space of measurable functions. This relation is given by the Riesz Representation theorem. We will state the Riesz Representation theorem (in more than one form). For a formal proof, the reader is referred to [Fol13].

Theorem D.0.2 (Riesz Representation Theorem ([Fol13]))**.** *Let $C_c(X)$ be the space continuous compactly supported complex-valued functions on a locally compact Hausdorff space X. For any positive linear functional κ, there exists a unique Radon measure μ on X such that for every $f \in C_c(X)$ we have*

$$\kappa(f) = \int_X f(x) d\mu(x)$$

We also have another result which relates the set $C_0(X)$ of continuous functions on LCH space X which vanish at infinity to its dual. First however, we wish to show that $C_0(X)$ is the closure of $C_c(X)$.

Proposition D.0.3 (([Fol13]))**.** *If X is an LCH space, $C_0(X)$ is the closure of $C_c(X)$ in the metric induced by the uniform norm.*

Proof. If $\{f_n\}$ is a sequence of functions in $C_c(x)$ that converge (uniformly) to a function $f \in C(X)$, we have that for every $\epsilon > 0$ there exists an $n \in \mathbb{N}$ such that $\|f_n - f\| < \epsilon$. Then $|f(x)| < \epsilon$ if x is not in the support of f_n, so $f \in C_0(X)$.

Conversely, let $f \in C_0(X)$ and for $n \in \mathbb{N}$ let $K_n = \{x : |f(x)| \geq n^{-1}\}$.

K_n is a compact set and by Urysohn's lemma there must exist a function $g_n \in C_c(X)$ where $0 \leq g_n \leq 1$ and $g_n = 1$ on K_n. Define $f_n = g_n f$. This says that $f_n \in C_c(x)$ and $\|f_n - f\| \leq n^{-1}$, so f_n converges to f uniformaly. $\square$

We now have extensions of the Riesz Representation theorem:

Theorem D.0.4 (([Fol13])). *Let X be LCH. For any continuous linear functional ψ on $C_0(X)$ there exists a unique regular countably additive complex Borel measure μ on X such that for all $f \in C_0(X)$ we have that $\psi(f) = \int_X f(x)d\mu(x)$.*

Theorem D.0.5 (([Fol13])). *Let X be a LCH space and let $M(X)$ be the measures on X. If $\mu \in M(X)$ and $f \in C_0(X)$, let $I_\mu(f) = \int f d\mu$. The map $\mu \to I_\mu$ is then an isometric isomorphism from $C_0(X)^* \to M(X)$.*

Curriculum Vitae

Max Hennick:

St. Thomas University, Bachelor of Arts, 2013-2014

University of New Brunswick, Bachelor of Science, 2014-2018

University of New Brunswick, Master of Science, 2019-present

Publications: None

Conference Presentations: None