On Invariant and Equivariant Neural Networks

Towards Geometric Machine Learning

Gabriel Ong

May 2022

MATH 2805: Mathematical Principles of Machine Learning

1 Invariance and Equivariance as a Desiderata

There are many transformations one can apply to training data of a neural network that ought not affect the output of the neural network's prediction. For example, why should an image of a cat be classified otherwise if it is fed into an algorithm upside down or after some planar translation? It would be desirable if our neural networks would not be affected by such transformations. Let us recall the following definitions (Sannai et al., 2019).

Definition 1.1 (Invariance)**.** Let G be a finite group and $f : \mathbb{R}^n \to \mathbb{R}^m$ where $G \circlearrowright \mathbb{R}^n$. Suppose that $\varphi : G \hookrightarrow S_n$ is given. f is G-invariant if and only if $f(\varphi(g) \cdot \vec{x}) = f(\vec{x})$ for any $g \in G$ and any $\vec{x} \in \mathbb{R}^n$.

Definition 1.2 (Equivariance)**.** Let G be a finite group and $f : \mathbb{R}^n \to \mathbb{R}^m$ where $G \circlearrowright \mathbb{R}^n$ and $G \circlearrowright \mathbb{R}^m$. Suppose that $\varphi : G \hookrightarrow S_n$ and $\psi : G \hookrightarrow S_m$ are given. f is G-equivariant if and only if $f(\varphi(g) \cdot \vec{x}) = \psi(g) \cdot f(\vec{x})$ for all $g \in G$ and any $\vec{x} \in \mathbb{R}^n$.

Thus invariance tells us that for a fixed input $\vec{x} \in \mathbb{R}^n$ the output of our function f is unchanged under any G action on the input. Equivariance tells us that if a group G acts on both the input and output space, then for some fixed output $\vec{x} \in \mathbb{R}^n$ taking the function of the g-action on the input is the same as taking the g-action on the output – in other words, that the group action commutes under function composition with the function f. We will make these notions of groups and actions more precise in the following section. Let us consider the following examples to get an intuitive understanding of invariance and equivariance.

Example 1. *Let $\mathcal{N} : \mathbb{R}^n \to [k]$ be a k-class image classifier and $\tau : \mathbb{R}^n \to \mathbb{R}^n$ a translation. We would like for $(\mathcal{N} \circ \tau)(\vec{x}) = \mathcal{N}(\vec{x})$ – that our neural network would classify both an image and its translate in the same way. Namely an idealized neural network $\mathcal{N}$ would be τ invariant.*

Example 2. *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a function that sharpens images and $\tau : \mathbb{R}^n \to \mathbb{R}^n$ be a $90°$ clockwise rotation. We would like for $(f \circ \tau)(\vec{x}) = (\tau \circ f)(\vec{x})$ – that first rotating then sharpening the image is the same as first sharpening then rotating the image. Namely, we would like our image sharpening function f to be equivariant under τ.*

The examples above are all but two examples that invariance and equivariance are desiderata for algorithms generally, and neural networks specifically. We note that the transformations we have seen so far are relatively simple. Ideally we seek layers of deep neural networks that are not only equivariant to mundane transformations like rotations and translations but also for more esoteric types of transformations such as reflections. In fact, in this paper, we will discuss why equivariance under the group wallpaper $p4m$ is a desideratum of image classification networks. Building to this goal requires a more mathematically mature understanding of transformations and symmetries – one achieved through the study of the theory of groups.

2 On the Theory of Groups and their Actions

Let us recall the following definition from our first course in abstract algebra.

Definition 2.1 (Group)**.** Let S be a set and $* : G \times G \to G$ a binary operation. $G = (G, *)$ is a group if and only if the following conditions hold:

- $g * h \in G$ for all $g, h \in G$.
- $(g * h) * k = g * (h * k)$ for all $g, h, k \in G$.
- There exists uniquely $e \in G$ such that $g * e = e * g = g$ for all $g \in G$.
- For all $g \in G$ there exists uniquely $g^{-1} \in G$ such that $g * g^{-1} = g^{-1} * g = e$.

However, in this case, we are concerned with how functions act on spaces such as the space of input data to a machine learning model. Thus we must understand how groups interact with certain spaces of interest.

Definition 2.2 (Group Action)**.** Let G be a group and X be a set. The action G on X is a map $G \times X \to X$ such that

- $e_G \cdot x = x$ for all $x \in X$
- $h \cdot (g \cdot x) = (g * h) \cdot x$ for all $g, h \in G$ and $x \in X$.

where we write $G \circlearrowright X$.

Let us now define the following.

Definition 2.3 (Orbit)**.** Let $G \circlearrowright X$. The orbit of $x \in X$ is the set

$$\mathrm{Orb}_X(x) = \{x' \in X | x \cdot g = x', \forall g \in G\}.$$

Definition 2.4 (Stabilizer)**.** Let $G \circlearrowright X$. The stabilizer of $x \in X$ is the set

$$\mathrm{Stab}_G(x) = \{g \in G | g \cdot x = x\}.$$

This gives us the following theorem.

Theorem 2.1 (Orbit-Stabilizer)**.** *Let G be a finite group and X a set where $G \circlearrowright X$. For all $x \in X$ the following identity holds:*

$$|G| = |\mathrm{Orb}_X(x)| \cdot |\mathrm{Stab}_G(x)|.$$

We omit the proof. We will eventually prove universal approximation for G equivariant neural networks when G is a finite group. We will thus need the following theorem of Cayely.

Theorem 2.2 (Cayley)**.** *If G is a finite group where $|G| = n$ then G is isomorphic to some subgroup of S_n.*

We conclude with two more definitions foundational to the understanding of group actions.

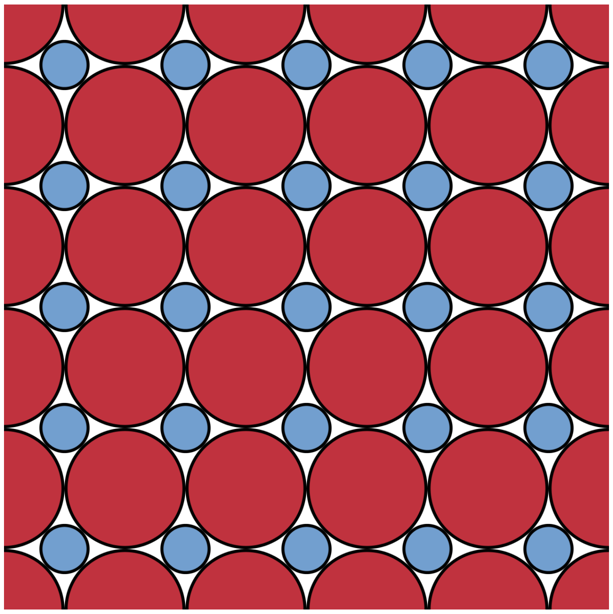
Definition 2.5 (Transitive Actions)**.** Let G be a group and X a set where $G \circlearrowright X$. $G \circlearrowright X$ is transitive if and only if for all $x, x' \in X$ there exists $g \in G$ such that $g \cdot x = x'$.

Definition 2.6 (Homogeneous Spaces)**.** Let G be a group and X a set. If $G \circlearrowright X$ is transitive then X is a homogeneous space of a group G

Definition 2.7 (G-Stability)**.** Let G be a group and X a set. If $\bigcup_{x \in X} \mathrm{Orb}_X(x) \subseteq X$ then X is stable under the G-action.

While these definitions and theorems may seem unmotivated and abstract, they actually play a foundational role in equivariance in machine learning. Let us now turn to an example pertinent to our task of image classification. In particular in our later experiment we will employ the symmetry group $p4m$.

Example 3 (The $p4m$ Group)**.** *Imagine a plane infinitely tiled as follows (Kennedy, 2004):*



The pattern-preserving symmetries here consist of rotations, translations, and reflections. To be more precise rotations by multiples of 90° *are pattern preserving, as are reflections along the vertical, horizontal, and both diagonal axes. Finally pattern preserving translations are also part of the pattern-preserving symmetries.*

3 Groups and Machine Learning

We have previously seen that convolutional neural networks are especially powerful algorithmic tools for feature detection. By design, convolutional neural networks are translation invariant – their output is unchanged when an object is shifted around in the grid so long as the grid still contains the entire object of interest. However, we seek to make our convolutional layers invariant under other "realistic" symmetries such as reflections. Cohen and Welling sought to answer the call by introducing group equivariant neural networks that increases the degree of weight sharing (Engelenburg, 2020). In particular, group equivariant neural networks are able to learn features under rotations and vertical, horizontal, and diagonal reflections (Engelenburg, 2020). This implies that the neural network with group equivariant layers is able to classify objects regardless of its classification.

We know from a celebrated theorem of Cybenko and others that continuous functions $f : K \to \mathbb{R}$ from some compact subset $K \subset \mathbb{R}^n$ we can find some neural network $\mathcal{N} : K \to \mathbb{R}$ such that $\|f(\vec{x}) - \mathcal{N}(\vec{x})\| < \epsilon$ for any choice of ϵ. Namely for any choice of continuous function

f we can find a neural network $\mathcal{N}$ that approximates f with arbitrary precision. Yet, it is entirely unclear if this theorem holds for these new classes of invariant and equivariant neural networks. In the 2017 work of Zaheer and coworkers, they showed the following.

Theorem 3.1 (Zaheer et al., 2017)**.** *Let G be a finite group which is a subgroup of S_n. Let $K \subset \mathbb{R}^n$ be compact which is stable for the for the corresponding G-action in $\mathbb{R}^n$. Then for any $f : K \to \mathbb{R}^m$ which is continuous and G-invariant and any $\epsilon > 0$ there exists some invariant neural network $\mathcal{N}_G^{\text{Inv}} : \mathbb{R}^n \to \mathbb{R}^m$ such that $\|f - \mathcal{N}_G^{\text{Inv}}\|_\infty < \epsilon$.*

Recall that we view our invariant neural network as

$$\mathcal{N}_G^{\text{Inv}} = \sum \circ L_H \circ \text{ReLU} \circ \cdots \circ L_1$$

where $L_i : \mathbb{R}^{n^{d_i} \times a_i} \to \mathbb{R}^{n^{d_i} \times a_{i+1}}$ are linear maps (defined by weight matrices) such that $L_i(g \cdot \vec{y}) = g \cdot L_i(\vec{y})$ for all $\vec{y} \in \mathbb{R}^{n^{d_i} \times a_i}$ and $g \in G$ where the group action is on each layer except the output layer. We also recall the following proposition from real analysis.

Proposition 3.2 (Kolmogorov-Arnold)**.** *Let $K \subset \mathbb{R}^n$ be a compact set and $f : K \to \mathbb{R}$ a continuous S_n invariant function. We can write*

$$f = \rho\left(\sum_{i=1}^{n} \phi(x_i)\right)$$

for some continuous function $\rho : \mathbb{R}^{n+1} \to \mathbb{R}$ and $\phi : \mathbb{R} \to \mathbb{R}^{n+1}$ defined as $x \mapsto (1, x, x^2, \ldots, x^n)^T$.

In the remainder of this section, we show the following theorem (Sannai et al., 2019).

Theorem 3.3 (Universal Approximation of G-Equivariant CNNs)**.** *Let G be a finite group which is a subgroup of S_n. Let $K \subset \mathbb{R}^n$ be compact and stable for the corresponding G-action in $\mathbb{R}^n$. Then for any $f : K \to \mathbb{R}^n$ by $\vec{x} \mapsto (f_1(\vec{x}), \ldots, f_n(\vec{x}))$ which is continuous and G-equivariant and any $\epsilon > 0$ there exists some equivariant neural network $\mathcal{N}_G^{\text{Equiv}} : K \to \mathbb{R}^n$ such that $\|f - \mathcal{N}_G^{\text{Equiv}}\|_\infty < \epsilon$.*

Our proof will proceed as follows. We will first show reduce an S_n equivariant map F to a Stab(1) invariant function where Stab(1) is the stabilizer of the number "1" in the natural action of S_n on $[n]$. We will then use the Kolmogorov-Arnold theorem to give us the existence of a Stab(1) invariant function f. By **Theorem 3.1**, we know that there is an invariant deep neural network approximating f. From here we can construct a neural network approximating F. We follow closely the proof of Sannai et al., 2019.

Proposition 3.4 (Equivariance and Invariance). *$F : \mathbb{R}^n \to \mathbb{R}^n$ is S_n equivariant if and only if there is a* Stab(1)*-invariant function $f : \mathbb{R}^n \to \mathbb{R}$ such that $F = (f, f \circ (12), \dots, f \circ (1n))^T$ where we denote $(1k) \in S_n$ the transposition of* 1 *and k.*

Proof. ($\Longrightarrow$) Suppose Stab(1) is a Stab(1) invariant function f, we show that the map $F = (f, f \circ (12), \dots, f \circ (1n))^T$ is S_n equivariant. We know from our first course in the theory of groups that the transpositions $(1i)$ generate S_n for $1 \leq i \leq n$. As such, it suffices to show that $F((1i) \cdot x) = (1i) \cdot F(x)$ for any $1 \leq i \leq n$. Denote $F(x)_j$ the jth entry of $F(x)$. If $i \neq j$ then $F((1i) \cdot x)_j = f((1j) \cdot ((1i) \cdot x))$ since f is Stab(1) invariant. By analagous arguments we have $F((1i) \cdot x)_i = f(x) = F(x)_1$ and $F((1i) \cdot x)_1 = f((1i) \cdot x) = F(x)_i$ demonstrating $F((1i) \cdot x) = (1i) \cdot F(x)$.

($\Longleftarrow$) Conversely suppose F is S_n equivariant so $F(\sigma x) = \sigma F(x)$ for all $x \in \mathbb{R}^n$ and $\sigma \in S_n$. We set $F = (f_1, \dots, f_n)^T$ where each $f_i : \mathbb{R}^n \to \mathbb{R}$ for all $1 \leq i \leq n$. But F is S_n equivariant by definition so $f_i(\sigma \cdot x) = f_{\sigma^{-1}(i)}(x)$ for any $x \in \mathbb{R}^n$ and $\sigma \in S_n$. This tells us that $f_1(\sigma \cdot x) = f_{\sigma^{-1}(1)}(x) = f_1(x)$ since $\sigma \in$ Stab(1) implies that $\sigma^{-1} \in$ Stab(1) as well. Thus f_1 is Stab(1) invariant. Now for $i = 1$ and the transposition $\sigma = (1j)$ where $j \neq 1$ we have $f_1((1j) \cdot x) = f_{(1j)}(x) = f_j(x)$ since $(1j) = (1j)^{-1}$. But this tells us that $F = (f_1, f_1 \circ (12), \dots, f_1 \circ (1n))^T$ which was precisely what we wanted. □

Note that Stab(1) $\cong S_{n-1}$ by the Orbit-Stabilizer theorem so we know that Stab(1) invariant functions $f : \mathbb{R}^n \to \mathbb{R}$ are S_{n-1} invariant functions allowing us to apply **Theorem 3.1**. We thus have the following, that is a corollary of **Theorem 3.1** and **Proposition 3.2**.

Proposition 3.5 (Representability of Stab(1) Invariant Functions). *Let $K \subset \mathbb{R}^n$ be compact and $f : K \to \mathbb{R}$ a continuous* Stab(1) *invariant function. We can write*

$$f = \rho\left(x_1, \sum_{i=2}^{n} \phi(x_i)\right)$$

for some continuous function $\rho : \mathbb{R}^{n+1} \to \mathbb{R}$ and $\phi : \mathbb{R} \to \mathbb{R}^{n+1}$ defined as $x \mapsto (1, x, x^2, \dots, x^n)^T$.

Proof. If f is Stab(1) invariant then it is S_{n-1} invariant. The theorem of Kolmogorov-Arnold we can induce the decomposition on the coordinates $x_2, \dots, x_n$ to give us the result. □

From this, we can see that the Stab(1) invariant function $f : \mathbb{R}^n \to \mathbb{R}$ can be written as

$f = \rho \circ L \circ \Phi$ where $\Phi : \mathbb{R}^n \to \mathbb{R} \times (\mathbb{R}^n)^{n-1}$ and $L : \mathbb{R} \times (\mathbb{R}^n)^{n-1} \to \mathbb{R} \times \mathbb{R}^n$ by

$$\Phi(x_1, \dots, x_n) = (x_1, \phi(x_1), \dots, \phi(x_n)) \quad L(x, (y_1, \dots, y_{n-1})) = \left(x, \sum_{i=1}^{n-1} y_i \right)$$

So the Stab(1) invariant function f is in fact a composition of a continuous functions ρ, L, Φ where ρ and Φ are Stab(1) invariant. Thus by **Theorem 3.1** we can replace both ρ and Φ by a invariant neural network approximations. Moreover with L continuous we can approximate L by a neural network up to arbitrary precision as well. Thus their composition approximates f up to arbitrary precision. Let us now apply **Proposition 3.4** which tells us for $F(x)$ an S_n equivariant function, we can write

$$F(x_1, \dots, x_n) = (f, f \circ (12), \dots, f \circ (1n))^T = (f, \dots, f)^T \circ (\varepsilon, (12), \dots, (1n))^T$$

where $\varepsilon \in S_n$ is the identity permutation and the composition of functions taken point-wise. The function $(f, \dots, f)^T$ can be approximated by some deep neural network since each function f can be approximated by a deep neural network. This function is also S_n equivariant since it is the same in every coordinate. It remains to show that the function $(\varepsilon, (12), \dots, (1n))^T$ is S_n equivariant as well.

We want to show that we can approximate $(\varepsilon, (12), \dots, (1n))^T$ with a neural network of arbitrary precision. Let us first state the following definition.

Definition 3.1 (The S_n Action on V^n)**.** Let V be a n-dimensional real vector space. Suppose Stab(1) acts on V by permutation denoted $\sigma \cdot \vec{x}$. We can thus define the action $* : S_n \times V^n \to V^n$ as follows:

$$\sigma * (\vec{x}_1, \dots, \vec{x}_n) = (\tilde{\sigma_1} \cdot \vec{x}_{\sigma^{-1}(1)}, \dots, \tilde{\sigma_n} \cdot \vec{x}_{\sigma^{-1}(n)}).$$

Where for any $1 \leq i \leq n$ we set $\tilde{\sigma_i} \in \text{Stab}(1)$ by $(1i)\sigma = \tilde{\sigma_i}(1\sigma^{-1}(i))$. We now show the following proposition.

Proposition 3.6 ($*$ is Well-Defined)**.** *The action $* : S_n \times V^n \to V^n$ is well-defined.*

Proof. Note that $S_n = \bigsqcup_{i=1}^n \text{Stab}(1)(1i)$ since if $(1\sigma^{-1}(1)) \in \text{Stab}(1)$ this $\sigma \in \text{Stab}(1)(1\sigma^{-1}(i))$ so for $(1i)\sigma$ then $(1i)\sigma$ is in the coset $\text{Stab}(1)(1\sigma^{-1}(i))$ since $\text{Stab}(1)(1((1i)\sigma)^{-1}(1)) = \text{Stab}(1)(1\sigma^{-1}(i))$. Thus we have a unique element $\tilde{\sigma_i} \in \text{Stab}(1)$ such that $(1i)\sigma = \tilde{\sigma_i}(1\sigma^{-1}(i))$. Suppose that $\sigma, \tau \in S_n$ and $X \in V^n$ so

$$\tau * (\sigma * X) = \tau * \begin{bmatrix} - & \tilde{\sigma_1} \cdot \vec{x}^T_{\sigma^{-1}(1)} & - \\ & \vdots & \\ - & \tilde{\sigma_n} \cdot \vec{x}^T_{\sigma^{-1}(n)} & - \end{bmatrix} = \begin{bmatrix} - & \tilde{\tau_1} \sigma_{\widetilde{\tau^{-1}}(1)} \cdot \vec{x}^T_{\sigma^{-1}(\tau^{-1}(1))} & - \\ & \vdots & \\ - & \tilde{\tau_1} \sigma_{\widetilde{\tau^{-1}}(n)} \cdot \vec{x}^T_{\sigma^{-1}(\tau^{-1}(n))} & - \end{bmatrix}$$

but from above we from $\tilde{\sigma_i} = (1i)\sigma(1\sigma^{-1}(i))$ and thus

$$\begin{aligned}\tilde{\tau_i}\sigma_{\tilde{\tau^{-1}(i)}} &= (1i)\tau(1\tau^{-1}(i))(1\tau^{-1}(i))\sigma(1\sigma^{-1}(\tau^{-1}(i)))\\ &= (1i)\tau\sigma(1(\tau\sigma)^{-1}(i))\\ &= \widetilde{(\tau\sigma)}_i\end{aligned}$$

so in fact

$$\tau * (\sigma * X) = \begin{bmatrix} - & \tilde{\tau_1}\sigma_{\tilde{\tau^{-1}(1)}} \cdot \vec{x}^T_{\sigma^{-1}(\tau^{-1}(1))} & - \\ & \vdots & \\ - & \tilde{\tau_1}\sigma_{\tilde{\tau^{-1}(n)}} \cdot \vec{x}^T_{\sigma^{-1}(\tau^{-1}(n))} & - \end{bmatrix} = \begin{bmatrix} - & \widetilde{(\sigma\tau)}_1 \cdot \vec{x}^T_{(\tau\sigma)^{-1}(1)} & - \\ & \vdots & \\ - & \widetilde{(\sigma\tau)}_n \cdot \vec{x}^T_{(\tau\sigma)^{-1}(n)} & - \end{bmatrix} = \tau\sigma * X$$

giving us well-definedness. □

The above proposition is key to showing that the layer $(\varepsilon, (12), \ldots, (1n))^T$ can be approximated with a neural network up to arbitrary precision. Note that $(f, \ldots, f)^T$ has input space $\mathbb{R}^{n\times n} = V^n$ (since each coordinate may take different inputs under the S_n action) so we want to show that there exists an equivariant input layer $g : \mathbb{R}^n \to V^n$ where $g = (g_1, \ldots, g_n)$ such that each $g_i = \text{ReLU} \circ l \circ (1i)$ for some $l : \mathbb{R}^n \to V$ Stab(1) invariant. We can now prove this proposition.

Proposition 3.7 (S_n Equivariance of G)**.** *$g : \mathbb{R}^n \to V^n$ is equivariant under the S_n action.*

Proof. For any i and $\sigma \in S_n$ and some $\vec{x} \in \mathbb{R}^n$ we have

$$(l \circ (1i) \circ I_n)(\sigma \cdot \vec{x}) = l(((1i)\sigma) \cdot \vec{x}).$$

Moreover for any i there exists $\tilde{\sigma_i} \in S_n$ uniquely such that $(1i)\sigma = \tilde{\sigma_i}(1\sigma^{-1}(i))$ per **Definition 3.1**. We thus have

$$\begin{aligned}l(((1i)\sigma) \cdot \vec{x}) &= l(\tilde{\sigma_i}(1\sigma^{-1}(i)) \cdot \vec{x})\\ &= \tilde{\sigma_i} \cdot l((1\sigma^{-1}(i)) \cdot \vec{x}) && \text{by Stab(1) equivariance of } l\end{aligned}$$

so we have that g_i as

$$\begin{aligned}(\sigma * g(\vec{x}))_i &= \tilde{\sigma_i} \cdot g(\vec{x})_{\sigma^{-1}(i)}\\ &= \tilde{\sigma_i} \cdot (\text{ReLU}(l((1\sigma^{-1}(i)) \cdot \vec{x})))\end{aligned}$$

but we know that $\tilde{\sigma_i} \circ \text{ReLU} = \text{ReLU} \circ \tilde{\sigma_i}$ so this gives us that g is S_n equivariant since it is equivariant in each coordinate. □

This allows us to conclude the proof of the theorem.

Proof of Theorem 3.2. Let F be an equivariant function. From above, we can write

$$F(x_1, \dots, x_n) = (f, f \circ (12), \dots, f \circ (1n))^T = (f, \dots, f)^T \circ (\varepsilon, (12), \dots, (1n))^T.$$

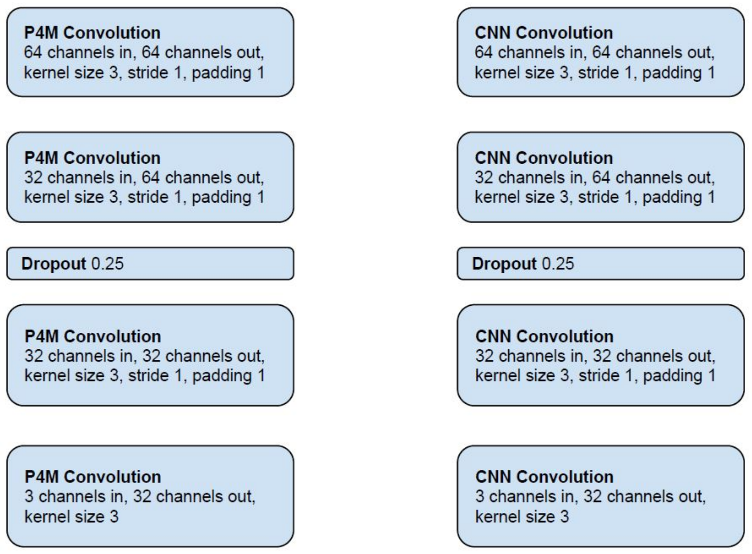
We know that $(f, \dots, f)^T$ can be approximated by some neural network $\mathcal{N}$ with arbitrary precision since each coordinate function f can be approximated with arbitrary precision. The input layer $(\varepsilon, (12), \dots, (1n))^T : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ can also be approximated by a neural network with arbitrary precision by g. Thus the composition $\mathcal{N} \circ g$ gives us an approximation of F by a neural network up to arbitrary precision as well, concluding our proof. $\square$

We remark, however, that this theorem only tells us about the approximation capability of equivariant layers but this is in fact sufficient since we know about the correspondence between convolutional and deep neural network layers.

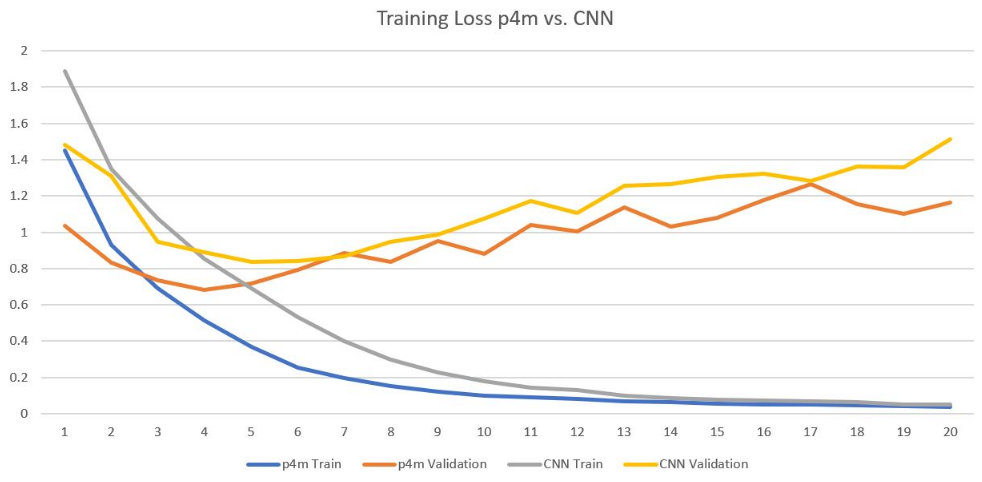
4 An Experiment

We conduct an experiment comparing the performance of a group equivariant neural network against a traditional convolutional neural network on the CIFAR10 image classification dataset. The CIFAR10 dataset consists of 60,000 32×32 colored images equally distributed between airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. To allow for a fair comparison, we allow our CNN model to view all D_4 symmetries of CIFAR10 images, namely the training set for our model has all 8 possible rotations and reflections of the square image.

We employ the Groupy package for group convolutional layer operations (Cohen and Welling, 2016) and structure our convolutional layers as follows:


P4M Convolution
64 channels in, 64 channels out, kernel size 3, stride 1, padding 1
P4M Convolution
32 channels in, 64 channels out, kernel size 3, stride 1, padding 1
Dropout 0.25
P4M Convolution
32 channels in, 32 channels out, kernel size 3, stride 1, padding 1
P4M Convolution
3 channels in, 32 channels out, kernel size 3
CNN Convolution
64 channels in, 64 channels out, kernel size 3, stride 1, padding 1
CNN Convolution
32 channels in, 64 channels out, kernel size 3, stride 1, padding 1
Dropout 0.25
CNN Convolution
32 channels in, 32 channels out, kernel size 3, stride 1, padding 1
CNN Convolution
3 channels in, 32 channels out, kernel size 3

where on the left we have our group equivariant CNN and on the right we have our baseline CNN. The outputs of these convolutional layers are then fed, separately, into a 3×3 pooling layer followed by dense layers of size 1600, 512, and 10. We also implement a dropout of 0.25 between the second-to-last and final dense layer. The CIFAR10 dataset of size 60,000 is already split into a training set of 50,000 images, and a test set of 10,000 images. From the set of 50,000 images, we remove 20% as a validation set before proceeding with training. We employ a batch size of 4 and use stochastic gradient descent with a learning rate of 0.001 with momentum 0.9. We train for 20 epochs.



We observe that both our models converge, but that the *p*4*m*-CNN converges faster than the traditional CNN since the entropy loss for the both the training and validation set decreases more quickly in the $p4m$-CNN model than it does for the CNN model. We also conjecture that the $p4m$-CNN model is less susceptible to overtraining since its validation loss increases more slowly than that of the CNN model. Following best-practices, we use the model with lowest validation error for training – this was in the 4th epoch for the $p4m$-CNN and the 5th epoch for the CNN. We then test the model on the 10,000 image CIFAR 10 test set and report both cross-entropy loss as well as classification accuracy below.

Metric	***p4m*-CNN**	**CNN**
Cross-Entropy Loss	0.679	0.832
Classification Accuracy	77%	70%

From these results, we can see that our $p4m$-CNN model outperforms the standard CNN model by a considerable margin. Given that our baseline CNN model was fed with all D_4 symmetries of the square image, the results imply that employing $p4m$ convolutions provides substantial insight into the underlying structure of the data over mere data augumentation. Examining the class-by-class classification breakdown we have the following:

Class	***p4m*-CNN**	**CNN**
Plane	75.8	70.9
Car	88.9	82.1
Bird	65.7	63.7
Cat	55.9	35.7
Deer	71.0	74.0
Dog	75.0	62.0
Frog	85.1	80.1
Horse	79.9	68.4
Ship	86.2	83.3
Truck	87.9	88.7

We observe that the $p4m$-CNN outperforms the traditional CNN in all classes except for "Deer" in which it only underperforms by 3%. These results mimic those found in the literature where $p4m$ convolutional neural networks are found to be much more powerful than traditional CNNs (Cohen and Welling, 2016). In particular, our results support the findings of Cohen and Welling where $p4m$-CNNs outperform standard CNNs on the rotated MINST dataset, consiting of rotated images of handwritten digits.

$p4m$-CNNs, however, are significantly more time consuming to train. Even with a training set 8 times the size, the CNN was able to finish training in about half the time it took to train the $p4m$-CNN. Using the `summary` syntax in `PyTorch`, we find that the $p4m$-CNN requires 12.42mb in each forward/backward pass while the standard CNN model only requires 1.66mb. This means that the $p4m$-CNN model requires almost 8 times the memory of the standard CNN. Using $p4m$-CNNs is thus a highly computationally intensive task.

We remark that these results could be improved upon drastically improved. Recently developed techniques such as residual networks (He et al., 2015) have allowed for classification of CIFAR10 on regular convolutional neural networks with accuracy approaching 95%. This could indeed narrow the performance gap between group convolutional and traditional convolutional neural networks.

5 Concluding Remarks

Group convolutional neural networks provide a way to exploit additional symmetric structure in data beyond translation symmetries. Moreover, we have shown them to be as powerful as traditional neural networks due to their universal approximation capabilities. Yet implementing group convolutional neural networks is a fraught matter. Group convolutional neural network packages have yet to be subsumed by the standard machine learning toolkits such as `PyTorch` and `TensorFlow`. Moreover their significantly higher memory requirements makes large-scale implementation of group convolutional neural networks extremely difficult especially as task sizes scale. Thus, for now, it seems that group convolutional neural networks are destined to remain on the theoretical sidelines.

References

Cohen T. S. and Welling M. (2016). “Group Equivariant Convolutional Networks”. DOI: 10.48550/ARXIV.1602.07576. URL: https://arxiv.org/abs/1602.07576.

Engelenburg C. van (2020). *Geometric deep learning: Group equivariant Convolutional Networks.* URL: https://medium.com/swlh/geometric-deep-learning-group-equivariant-convolutional-networks-ec687c7a7b41.

He K., Zhang X., Ren S., and Sun J. (2015). *Deep Residual Learning for Image Recognition.* DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

Kennedy T. (2004). “Compact packings of the plane with two sizes of discs”. DOI: 10.48550/ARXIV.MATH/0407145. URL: https://arxiv.org/abs/math/0407145.

Sannai A., Takai Y., and Cordonnier M. (2019). *Universal approximations of permutation invariant/equivariant functions by deep neural networks.* DOI: 10.48550/ARXIV.1903.01939. URL: https://arxiv.org/abs/1903.01939.

Zaheer M., Kottur S., Ravanbakhsh S., Poczos B., Salakhutdinov R., and Smola A. (2017). *Deep Sets.* DOI: 10.48550/ARXIV.1703.06114. URL: https://arxiv.org/abs/1703.06114.