Learning separation between equivariant and fully-connected neural networks

Bobak Kiani
MIT EECS
bkiani@mit.edu

Quynh Nguyen
MIT EECS
nguyentq@mit.edu

Abstract

Equivariant or group convolutional neural networks (G-CNN) generalize the translational symmetry properties of standard convolutional networks to symmetries over any group. The most evident applications of group convolutional neural networks are in learning classes of functions which are symmetric to group operations, where one would expect the equivariance properties of G-CNNs to excel. Compared to fully connected networks (FCN), which impart no group symmetries in their architecture, G-CNNs empirically require far fewer samples to learn such symmetric function classes. However, no formal proofs separating the learnability of FCNs and G-CNNs have been shown to date. Here, we bridge that gap and study a class of functions which provably require $O(1)$ samples to learn with a G-CNN as opposed to $\Omega(n)$ samples to learn with a FCN, where n is the dimension of the input. Experimental validation of our results show that the separation may be even larger than the proofs indicate, thus opening the possibility of proving larger learning separations in future work.

1 Introduction

Convolutional neural networks (CNNs) are a popular architecture of choice for many deep learning applications [6, 3], especially applicable in learning classes of functions which feature translational symmetries, as CNNs are inherently invariant to translations. Such symmetry patterns are not a feature of fully connected networks (FCNs) even though they are universal approximators. Recent work has rigorously shown that the architecture of a CNN provably learns certain function classes more optimally which have translational symmetry patterns. Most notably, [7] find a sample complexity separation between CNNs and FCNs in learning a simple task. Similarly, [9] show that CNNs can efficiently learn k-patterns (functions that only depend on k consecutive input bits); whereas, FCNs typically struggle as their gradients exponentially decay with dimension in this problem setting.

In this study, we expand these learning separation proofs to the setting of group convolutional neural networks (G-CNNs) [4, 1, 2]. G-CNNs generalize the translational symmetries of CNNs to symmetries over arbitrary groups. Due to their empirical success at learning various group theoretic tasks, G-CNNs have widespread application but a comprehensive theory to validate their success is mostly limited. To date, some work has theoretically analyzed the learning properties of G-CNNs [5, 4, 12, 8, 10], but to our knowledge, no work has yet proven a learning separation between G-CNNs and FCNs.

Here, we show that fully connected networks (FCNs) require $\Omega(n)$ more samples than G-CNNs in learning a function class which is symmetric to the group operations, where n is the dimension of the input. This target function class, called the coset equality function, is a binary classification task which assigns a positive classification whenever inputs are constant over left cosets of a group and negative classification otherwise. Though we provably show a $\Omega(n)$ sample complexity gap,

Preprint. Work in progress.

experiments show that the separation is actually $\Omega(n^2)$. In light of this, we hope that future work can further expand this sample complexity gap.

2 Notation and problem setting

Throughout this study, we denote vectors using bold lowercase script ($\boldsymbol{v}$), matrices using bold uppercase script ($\boldsymbol{M}$), and tensors using bold uppercase text ($\mathbf{T}$). We denote groups as G and group operations acting on a vector space as $\mathbb{T}_g$ where $g \in G$. In this study, the G-CNN is only considered in the case of finite groups G. We consider the task of binary classification of Boolean functions $f : \{-1, +1\}^{|G|} \to \{-1, +1\}$. For simplicity, we take a given ordering of group elements in G and consider inputs to the functions f as vectors $\boldsymbol{x} \in \{-1, +1\}^{|G|}$ with entries $\boldsymbol{x}_g$ for each group element $g \in G$. A group element $h \in G$ can also act on an input vector $\boldsymbol{x}_g$ by permuting its elements according to the left (or right) action of the group. *E.g.,* let $\mathbb{T}_h$ denote the left action of $h \in G$ on the vector $\boldsymbol{x}$, then $\mathbb{T}_h \boldsymbol{x}_g = \boldsymbol{x}_{hg}$. In employing this notation, we may abuse notation by dropping the $\mathbb{T}$ operator and write $h\boldsymbol{x}_g = \mathbb{T}_h \boldsymbol{x}_g$.

In this study, we employ two distinct notions of equivariances, one which applies to functions and another to algorithms. First, a function $f : \mathbb{R}^{|G|} \to \mathbb{R}^{|G|}$ is equivariant to the left or right action of a group G if applying a group operation before or after are equivalent.

Definition 2.1 (Functional equivariance)**.** Let $\mathbb{T}_g$ and $\mathbb{T}'_g$ be two operators which apply group transformations $g \in G$ on the vector space $\mathbb{R}^{|G|}$ as denoted earlier. A map $f : \mathbb{R}^{|G|} \to \mathbb{R}^{|G|}$ is equivariant to the actions of the group G if $\forall g \in G$:

$$f(\mathbb{T}_g \boldsymbol{x}) = \mathbb{T}'_g f(\boldsymbol{x}). \tag{1}$$

Second, an algorithm is equivariant to the actions of a (possibly separate) group if the algorithm outputs are equivalent after group operations applied to the data. More specifically, given a dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \in \mathcal{D}$ of size n and a hypothesis class $\mathcal{H}$, an algorithm $\mathcal{A} : \mathcal{D} \to \mathcal{H}$ is a mapping from datasets to functions in the hypothesis class.

Definition 2.2 (Algorithmic equivariance)**.** Let $\mathbb{T}_g \in \mathbb{R}^{n \times n}$ be a matrix drawn from a group G_A (*e.g.,* group of permutation matrices) which acts on the vector space $\mathbb{R}^n$ (we use notation G_A to note that this group is different than the group of the G-CNN). An algorithm $\mathcal{A} : \mathcal{D} \to \mathcal{H}$ is equivariant to the actions of the group G_A if $\forall\ \mathbb{T}_g \in G_A$:

$$\mathcal{A}\big(\{(\mathbb{T}_g \boldsymbol{x}_i, y_i)\}_{i=1}^n\big)(\mathbb{T}_g \boldsymbol{x}) = \mathcal{A}\big(\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n\big)(\boldsymbol{x}). \tag{2}$$

If the algorithm is stochastic, the equal sign is replaced by equal in distribution.

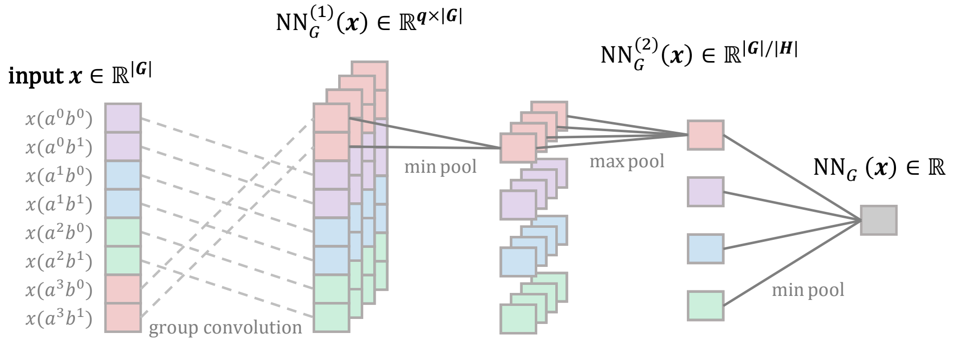
Target function Given a subgroup $H \subset G$, the target function f^* we would like to learn is the G/H-coset equality function which returns $+1$ when inputs are constant over all left cosets in G/H and -1 otherwise:

$$f^*(\boldsymbol{x}) = \begin{cases} +1 & \forall L \in G/H : \boldsymbol{x}_a = \boldsymbol{x}_b \ \ \forall a, b \in L \\ -1 & \text{otherwise} \end{cases} \tag{3}$$

Note, that the above target function is invariant to left group operations (*i.e.,* $f^*(g\boldsymbol{x}) = f^*(\boldsymbol{x})$ for all $g \in G$), since left group actions do not permute elements between left cosets.

Distribution We consider the following distribution over training and test data for our proofs. For each left coset $T \in G/H$, let the vector $\boldsymbol{x}_T \in \{-1, +1\}^{|G|}$ be such that $(\boldsymbol{x}_T)_g = 1$ if $g \in T$ and $(\boldsymbol{x}_T)_g = -1$ otherwise. For distinct $T, T' \in G/H$, let $\boldsymbol{x}_{T,T'}$ be such that $(\boldsymbol{x}_{T,T'})_g = 1$ if g is in T but except the last element (according to some fixed ordering of H), or if g is the last element in T', and $(\boldsymbol{x}_{T,T'})_g = -1$ otherwise. Let P be the distribution over $\{(\boldsymbol{x}_T, +1), (\boldsymbol{x}_{T,T'}, -1)\}_{g \in G, T, T' \in G/H}$ such that positive and negative examples are equally probable. Note that the number of $+1$ entries in each $\boldsymbol{x}$ is always equal to $|H|$.

We contrast the performance of a group convolutional neural network (G-CNN) $\mathrm{NN_G}(\cdot)$ to that of a fully connected network (FCN) $\mathrm{NN_{FC}}(\cdot)$.



$\mathrm{NN}_G^{(1)}(\boldsymbol{x}) \in \mathbb{R}^{q\times|G|}$
$\mathrm{NN}_G^{(2)}(\boldsymbol{x}) \in \mathbb{R}^{|G|/|H|}$
input $\boldsymbol{x} \in \mathbb{R}^{|G|}$
$x(a^0b^0)$
$x(a^0b^1)$
$x(a^1b^0)$
$x(a^1b^1)$
$x(a^2b^0)$
$x(a^2b^1)$
$x(a^3b^0)$
$x(a^3b^1)$
min pool
max pool
$\mathrm{NN}_G(\boldsymbol{x}) \in \mathbb{R}$
min pool
group convolution

Figure 1: Illustration of the structure of the G-CNN used to learn the G/H-coset equality function for the group D_8. Elements of a coset share the same color. The first layer is a q-channel group convolution layer which permutes cosets, but not elements between cosets. This is followed by a customized set of pooling layers that helps to learn the G/H-coset equality function.

G-CNN structure The G-CNN, whose architecture is illustrated in Figure 1, is cleverly constructed to efficiently learn the target function f^*. The first layer of the G-CNN performs group cross-correlation over the input with a filter $\boldsymbol{w}^{(\text{conv})}$ with q channels where each channel is supported on a single element of the group, *i.e.* $\boldsymbol{w}^{(\text{conv})} \in \mathbb{R}^q$, followed by ReLU activation. Note, that group-cross correlation of a filter $\boldsymbol{w}$ and input $\boldsymbol{x}$ in $\mathbb{R}^{|G|}$ is defined as

$$(\boldsymbol{w} \star \boldsymbol{x})_u = \sum_{v\in G} \boldsymbol{w}_v \boldsymbol{x}_{vu}. \tag{4}$$

Since for each channel, filters in our G-CNN are supported only on a single group element $v' \in G$, we can represent the weight for a given channel as a scalar w supported on group element v' and the above simplifies to $(\boldsymbol{w} \star \boldsymbol{x}) = w\mathbb{T}_{v'}\boldsymbol{x}$, where $\mathbb{T}_{v'}$ is the left action of v'. Explicitly, each channel k of the first layer of the G-CNN performs

$$\left[\mathrm{NN}_G^{(1)}(\boldsymbol{x})\right]_{k,\cdot} = \mathrm{ReLU}\left(\boldsymbol{w}_k^{(\text{conv})}\mathbb{T}_{v'}\boldsymbol{x}\right), \tag{5}$$

where $\left[\mathrm{NN}_G^{(1)}(\boldsymbol{x})\right]_{k,\cdot}$ denotes the k-th channel of the output of the first layer. The second layer of the G-CNN performs a minimum pooling over elements of the left cosets G/H followed by maximum pooling over the channel. The output is a vector with support over the left cosets $T \in G/H$:

$$\left[\mathrm{NN}_G^{(2)}(\boldsymbol{x})\right]_T = \max_{k\in[q]} \min_{u\in T} \left[\mathrm{NN}_G^{(1)}(\boldsymbol{x})\right]_{k,u}. \tag{6}$$

The final layer is a global pooling over all elements (including all channels) plus a multiplicative weight w_L and bias term b to properly separate the two output classifications,

$$\mathrm{NN}_G(\boldsymbol{x}) = w_L \left(\min_{T\in G/H} \left[\mathrm{NN}_G^{(2)}(\boldsymbol{x})\right]_T\right) + b. \tag{7}$$

FCN structure We consider a 3-layer FCN. The first and second hidden layers have $2 \cdot |G|$ and $|G|$ units, respectively, and each is followed by ReLU activations. The last layer (no activation) outputs a scalar whose sign indicates the prediction.

Initialization All parameters in both models are initialized from i.i.d standard Gaussian.

Loss function For training neural networks, we use the exponential binary classification loss defined as $\ell(y, y') = \exp(-yy')$.

3 Main results

Here, we present our main theoretical result which shows an $\Omega(|G|)$ sample complexity separation between the G-CNN and FCN. Put more simply, we prove here that a FCN would require at least on the order of samples of the size of the input to learn the target function that a G-CNN can learn with a constant number of samples. To prove this result, we consider groups where the size of the cosets remain constant, but the dimensions of the groups grow, *i.e.,* we fix $|H|$ to be constant and grow the size of the group (see example for the dihedral group in Section 4).

We note before proceeding that numerical experiments indicate that our sample complexity gap can be potentially made stronger and actually is at least $O(|G|^2)$ for the dihedral group studied in the experiments. This sample complexity gap is consistent with theoretical results from [7], showing a simple setting where standard CNNs can learn a given problem from $O(1)$ samples in comparison to $\Omega(n^2)$ samples (where n is the dimension of the input) for a FCN.

3.1 G-CNN learns target function with $O(1)$ samples

Our proof follows two steps. First, we show that the convolution and pooling structure of the G-CNN separates outputs in the final layer based on the coset properties of the input. Then, we show that the VC dimension of the class of G-CNNs is bounded by a constant which results in a proof of generalization.

First, we prove a helper lemma that shows that the first group convolution layer does not permute members between cosets.

Lemma 3.1. *Given a group G with subgroup H, left cosets $T \in G/H$, and the cross-correlation operation $\mathrm{NN}_G^{(1)}(\boldsymbol{x})$ defined in Equation 5, if input elements $\boldsymbol{x}_u = \boldsymbol{x}_v$ for $u, v \in T$ for all T, then $\left[\mathrm{NN}_G^{(1)}(\boldsymbol{x})\right]_{\cdot,u} = \left[\mathrm{NN}_G^{(1)}(\boldsymbol{x})\right]_{\cdot,v}$. Informally, if the input is constant over elements of all left cosets, then the output of the first layer is also constant over elements of individual left cosets.*

Proof. As a reminder, the first layer performs the following operation at a given channel q:

$$\left[\mathrm{NN}_G^{(1)}(\boldsymbol{x})\right]_{q,\cdot} = \mathrm{ReLU}\left(\boldsymbol{w}_q^{(\text{conv})}\mathbb{T}_{v'}\boldsymbol{x}\right). \tag{8}$$

Note, that $\mathbb{T}_{v'}\boldsymbol{x}_u = \boldsymbol{x}_{v'u}$. Since $\boldsymbol{x}_a = \boldsymbol{x}_b$ for all $a, b \in T$, then $\mathbb{T}_{v'}\boldsymbol{x}_a = \boldsymbol{x}_{v'a}$ and $\mathbb{T}_{v'}\boldsymbol{x}_b = \boldsymbol{x}_{v'b}$. These elements are clearly in the left coset $v'T = \{v'u : u \in T\}$. Furthermore, since $\mathrm{ReLU}(\cdot)$ acts element wise, then the output of $\left[\mathrm{NN}_G^{(1)}(\boldsymbol{x})\right]_{q,a} = \left[\mathrm{NN}_G^{(1)}(\boldsymbol{x})\right]_{q,b}$ for $a, b \in T$ since inputs are equal for each element in a coset. □

Next, we show that the output before the last layer of the G-CNN is zero if the input is not constant over cosets and greater than zero otherwise.

Lemma 3.2. *For a given set of convolution weights $\boldsymbol{w}^{(conv)} \in \mathbb{R}^q$ of NN_G, the output of the convolution and pooling operations $\min_{T\in G/H} \mathrm{NN}_G^{(2)}(\boldsymbol{x})_T$ takes one of three unique values, i.e.,*

$$\min_{T\in G/H} \mathrm{NN}_G^{(2)}(\boldsymbol{x})_T \in \left\{0, \max_{w\in\boldsymbol{w}^{(conv)}:w<0} |w|, \max_{w\in\boldsymbol{w}^{(conv)}:w>0} |w|\right\}. \tag{9}$$

Proof. As a reminder,

$$\left[\mathrm{NN}_G^{(1)}(\boldsymbol{x})\right]_{q,\cdot} = \mathrm{ReLU}\left(\boldsymbol{w}_q^{(\text{conv})}\mathbb{T}_{v'}\boldsymbol{x}\right), \tag{10}$$

$$\left[\mathrm{NN}_G^{(2)}(\boldsymbol{x})\right]_{T} = \max_{k\in[q]}\min_{u\in T} \left[\mathrm{NN}_G^{(1)}(\boldsymbol{x})\right]_{k,u}. \tag{11}$$

Note that $\min_{u\in T} \left[\mathrm{NN}_G^{(1)}(\boldsymbol{x})\right]_{k,u} > 0$ only if the function is constant over cosets. Otherwise, there will be one element of the coset for which $[\boldsymbol{w}_q^{(\text{conv})}\mathbb{T}_{v'}\boldsymbol{x}]_i$ is negative and the ReLU will set it to zero. Furthermore, note that if $\min_{u\in T} \left[\mathrm{NN}_G^{(1)}(\boldsymbol{x})\right]_{k,u} > 0$, then $\min_{u\in T} \left[\mathrm{NN}_G^{(1)}(\boldsymbol{x})\right]_{k,u} =$

$|(\mathbb{T}_{v'}^{-1}\boldsymbol{w}^{(\text{conv})})_k|$. This happens only when the sign of $\boldsymbol{x}$ in the coset is equal to $(\mathbb{T}_{v'}^{-1}\boldsymbol{w}^{(\text{conv})})_k$. Thus, there are two unique cases: one where $\boldsymbol{x}_T$ is positive and one where $\boldsymbol{x}_T$ is negative. The maximum pooling chooses the maximum of $|(\mathbb{T}_{v'}^{-1}\boldsymbol{w}^{(\text{conv})})_k|$ over $k \in [q]$ so only one unique value exists for the positive and negative case.

In summary, the output takes one of three unique values. It is set to zero when the input is not constant over cosets. And it takes the maximum absolute value of the weight of the channel which matches the sign of the input over cosets otherwise. This leads to three unique outcomes. □

Given the above property, the VC dimension of the hypothesis class is trivially equal to 3 (since the penultimate layer is grouped into three unique output values) [14, 13] and we obtain a proof of generalization as desired.

Theorem 3.3. *With $\delta > 0$, N training samples, weights drawn randomly from the standard normal distribution, and at least $q = O(\log_2(1/\delta))$ channels in the convolutional layer, the generalization error of the G-CNN scales as $O\left(\sqrt{\frac{3\ln(2N/3+1)-\ln(2/\delta)}{N}}\right)$ with probability at least $1-2\delta$.*

Proof. Before discussing generalization bounds, we must first ensure that the network can properly separate the three sets of inputs as discussed in Lemma 3.2. To achieve proper separation, one requires a channel in the convolutional filter with a positive and negative weight. For q channels and weights distributed according to the standard normal distribution, this occurs with probability $1 - \frac{1}{2^(q-1)}$. Hence to succeed with probability $1-\delta$, we require $O(\log_2(1/\delta))$ channels.

To prove our generalization bound, we first copy standard VC generalization bounds from [13]. Here, h is the VC dimension of the hypothesis class (equal to 3 for our G-CNNs) and $\varepsilon(h, N)$ is the generalization gap:

$$\varepsilon(h, N) = \sqrt{\frac{h\ln(2N/h+1) - \ln(2/\delta)}{N}}. \tag{12}$$

Plugging in $h = 3$ into the above, we obtain our desired result. □

Importantly, the above scaling of the generalization gap does not depend on the size of the group $|G|$ and we obtain our desired $O(1)$ sample complexity.

3.2 FCN requires $\Omega(|G|)$ samples to learn

Following proof methods of [7, 11], we use the algorithmic equivariance property of FCNs to show that at least $\Omega(|G|)$ samples are required to learn the coset-equality function. Specifically, we first formally restate the arguments in [7], showing that FCNs, when trained with i.i.d Gaussian initialization and popular optimizers (Adam, SGD, AdaGrad) are algorithmically equivariant to feature permutations.

Lemma 3.4. *Fully connected networks trained with (stochastic) gradient descent from i.i.d Gaussian initialization are algorithmically equivariant (see Definition 2.2) under the permutation group.*

Proof. Let $\Pi \in \mathbb{R}^{|G|\times|G|}$ be a permutation matrix. We consider the learning problem on datasets $S = \{(\boldsymbol{x}_i, y_i\}_{i=1}^m$ and $\widetilde{S} = \{(\Pi(\boldsymbol{x}_i), y_i\}_{i=1}^m$ using an L-layer FCN. Denote by $\mathbf{W}_1$ the weight matrix in the first layer, and by $\mathbf{W}_2$ all the remaining weights (including the offsets in the first layer). Using the notation $\mathbf{W} = (\mathbf{W}_1, \mathbf{W}_2)$, we define the transformation $O(\mathbf{W}) = (\Pi^{-1}\mathbf{W}_1, \mathbf{W}_2)$. Observe that $\mathrm{NN}_{\mathrm{FC}}(\boldsymbol{x}; \mathbf{W}) = \mathrm{NN}_{\mathrm{FC}}(\Pi\boldsymbol{x}; O(\mathbf{W}))$.

Here, we would like show that after t iterations of (S)GD, the predictions made by this FCN when trained on the two datasets S and $\widetilde{S}$ are equal in distribution

$$\mathrm{NN}_{\mathrm{FC}}(\boldsymbol{x}; \mathbf{W}^{(t)}) \overset{d}{=} \mathrm{NN}_{\mathrm{FC}}(\Pi\boldsymbol{x}; \widetilde{\mathbf{W}}^{(t)}). \tag{13}$$

It suffices to show $\widetilde{\mathbf{W}}^{(t)} \overset{d}{=} O(\mathbf{W}^{(t)})$ for any t. We proceed by an inductive proof.

1. At $t = 0$, we have $O(\mathbf{W}^{(0)}) \overset{d}{=} \widetilde{\mathbf{W}}^{(0)}$ due to the symmetry of i.i.d Gaussian initialization.

2. Assume $\widetilde{\mathbf{W}}^{(t)} \overset{d}{=} O(\mathbf{W}^{(t)})$ for $t \geq 0$. We consider the SGD update

$$\begin{aligned}\widetilde{\mathbf{W}}^{(t+1)} &= \widetilde{\mathbf{W}}^{(t)} - \eta \nabla_{\widetilde{\mathbf{W}}} \ell(\mathrm{NN}_{\mathrm{FC}}(\Pi \boldsymbol{x}; \widetilde{\mathbf{W}}^{(t)}), y). \\ &= \widetilde{\mathbf{W}}^{(t)} - \eta \nabla_{\widetilde{\mathbf{W}}} \ell(\mathrm{NN}_{\mathrm{FC}}(\boldsymbol{x}; O^{-1}(\widetilde{\mathbf{W}}^{(t)})), y).\end{aligned} \tag{14}$$

Replacing $\widetilde{\mathbf{W}}^{(t)} = O(\mathbf{W})$ (for some $\mathbf{W}$ sampled from the distribution of $\mathbf{W}^{(t)}$), using the chain rule and noting that Π is an orthogonal matrix ($\Pi^\top = \Pi^{-1}$) we obtain

$$\begin{aligned}\widetilde{\mathbf{W}}^{(t+1)} &= O(\mathbf{W}) - \eta \nabla_{\widetilde{\mathbf{W}}} \ell(\mathrm{NN}_{\mathrm{FC}}(\boldsymbol{x}; \mathbf{W}), y) \\ &= O(\mathbf{W}) - \eta O(\nabla_{\mathbf{W}} \ell(\mathrm{NN}_{\mathrm{FC}}(\boldsymbol{x}; \mathbf{W}), y)) \\ &= O(\mathbf{W} - \eta \nabla_{\mathbf{W}} \ell(\mathrm{NN}_{\mathrm{FC}}(\boldsymbol{x}; \mathbf{W}), y))\end{aligned} \tag{15}$$

Notice that the RHS is the update rule for $\mathbf{W}$ followed by the transformation O. Thus, the RHS is the random variable $O(\mathbf{W}^{(t+1)})$. Therefore, we have $\widetilde{\mathbf{W}}^{(t+1)} \overset{d}{=} O(\mathbf{W}^{(t+1)})$ as desired.

□

We note that the above proof actually holds more generally whenever Π is an orthogonal matrix, but for our purposes, we only require permutation equivariance.

Corollary 3.4.1. *FCNs trained with momentum or adaptive step size methods such as SGD momentum, Adagrad, Adam and any L_p regularization are permutation equivariant.*

Proof. It suffices to show that the inductive step in the proof of Lemma 3.4 still holds.

Indeed, for momentum methods (e.g. SGD momentum), we simply apply the tricks in Equation 14 and 15 to each individual past gradient in the momentum update rule. For adaptive step size methods (e.g. Adagrad), we observe that each step size η_w (for any $w \in \mathbf{W}$), which depends on the moving average of the gradient square $(\nabla_w \ell)^2$, is also permuted according to the transformation O. Finally, Adam combines these two methods and hence is also permutation equivariant. □

Theorem 3.5. *Any permutation equivariant algorithm $\mathcal{A}$ requires $\Omega(|G|)$ samples to learn the coset-equality problem described in Section 2.*

Proof. We first define $\Pi_{T,T'}(\boldsymbol{x})$ as the permutation which switches the positions of the last elements of the cosets. Given a training set $S_m = (\mathbf{X}_m, \mathbf{y}_n) = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^m \sim P^m$ where $m = \frac{|G/H|}{4}$, define $\varepsilon(\mathcal{A}(S_m)) = \mathbb{P}_{\boldsymbol{x},y\sim P}[\mathcal{A}(S_m)(\boldsymbol{x}) \neq y]$ as the test misclassification error. Observe that each $\boldsymbol{x}^{(i)}$ has at most two cosets that contain $+1$ entries, for any cosets T, T' other than these two, we have that $\Pi_{T,T'}(\boldsymbol{x}^{(i)}) = \boldsymbol{x}^{(i)}$. Thus, there are at least $\frac{|G/H|}{2}$ cosets such that for any pair (T, T') of them, applying $\Pi_{T,T'}$ leaves the training set $\mathbf{X}_m = \{\boldsymbol{x}^{(i)}\}_{i=1}^m$ unchanged. Define the event $B = [\Pi_{T,T'}(\mathbf{X}_m) = X_m]$, then we have $\mathbb{P}[B] \geq \frac{1}{4}$. Observe that

$$\begin{aligned}\varepsilon(\mathcal{A}(S_m)) &= \mathbb{P}_{\boldsymbol{x},y\sim P}[\mathcal{A}(\mathbf{X}_m, \mathbf{y}_n)(\boldsymbol{x}) \neq y] \geq \mathbb{P}_{\boldsymbol{x},y\sim P}[\mathcal{A}(\mathbf{X}_m, \mathbf{y}_n)(\boldsymbol{x}) \neq y | B] \cdot \mathbb{P}[B] \\ &\geq \frac{1}{8}\mathbb{P}_T[\mathcal{A}(\mathbf{X}_m, \mathbf{y}_n)(\boldsymbol{x}_T) \neq 1 | B] + \frac{1}{8}\mathbb{P}_{T,T'}[\mathcal{A}(\mathbf{X}_m, \mathbf{y}_n)(\boldsymbol{x}_{T,T'}) \neq -1 | B] \\ &\overset{(2.2)}{=} \frac{1}{8}\mathbb{P}_{T,T'}[\mathcal{A}(\Pi_{T,T'}(\mathbf{X}_m), \mathbf{y}_n)(\Pi_{T,T'}(\boldsymbol{x}_T)) \neq 1 | B] + \frac{1}{8}\mathbb{P}_{T,T'}[\mathcal{A}(\mathbf{X}_m, \mathbf{y}_n)(\boldsymbol{x}_{T,T'}) \neq -1 | B] \\ &= \frac{1}{8}\mathbb{P}_{T,T'}[\mathcal{A}(\mathbf{X}_m, \mathbf{y}_n)(\boldsymbol{x}_{T,T'}) \neq 1 | B] + \frac{1}{8}\mathbb{P}_{T,T'}[\mathcal{A}(\mathbf{X}_m, \mathbf{y}_n)(\boldsymbol{x}_{T,T'}) \neq -1 | B] \\ &= \frac{1}{8}.\end{aligned} \tag{16}$$

This directly implies that $\mathcal{A}$ requires more than $\frac{|G/H|}{4} = \Omega(|G|)$ samples to learn the coset-equality function (since $|H| = O(1)$). □

Given that FCNs, when trained with SGD or Adam are permutation equivariant, we obtain the desired $\Omega(|G|)$ sample complexity.

4 Numerical experiments

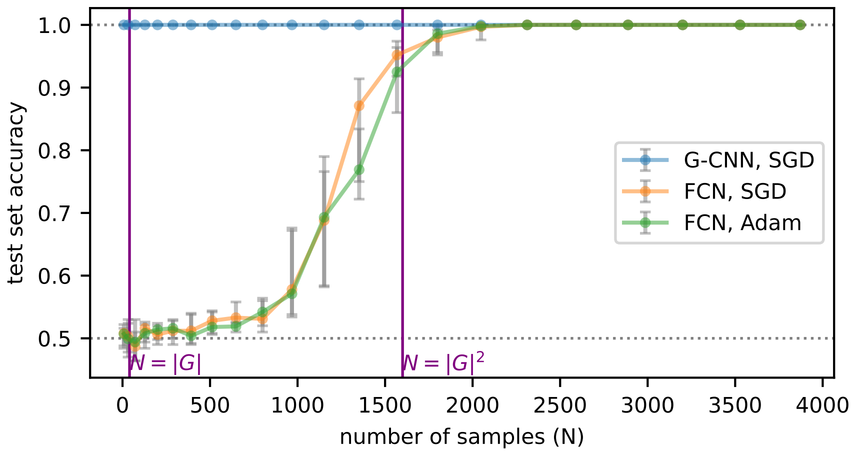


Figure 2: G-CNN learns the target function even with very few samples whereas FCN needs about $|G|^2$ samples to learn. Here the group $G = D_{40}$. Networks are trained over 15000 epochs using the learning algorithm indicated in the legend.

In this section, we perform experiments on the dihedral group D_{2n} of order $2n$ which is the symmetry group of the n-sided polygon. This group is the semidirect product of cyclic groups Z_n and Z_2. Any element $g \in D_{2n}$ of the dihedral group can be represented by the rotation (a) and flip (b) generators by $g = a^i b^j$ where $i \in [n]$ and $j \in \{0, 1\}$ indicate the number of rotations and flips respectively. Actions of the group have the property that $a^n = e$, $b^2 = e$, and $bab = a^{-1}$ where e is the identity operation of the group.

The target function we would like to learn is the G/H-coset equality function which is constant over left cosets over the subgroup $H = \{I, b\}$. There are n left cosets for this given subgroup. Training sets have equal number of inputs that are constant over cosets and not constant over cosets (*i.e.,* equal number of samples from each of the binary classes). To analyze our results in a more general setting and relax the assumptions of our proof, we draw samples from a more uniform distribution over training points. This is a more general distribution in comparison to that used in proving the sample complexity gap in Section 3.2. More specifically, positively classified samples are drawn uniformly from all $\boldsymbol{x}_i \in \{-1, +1\}^{|G|}$ which are constant over cosets. Negatively classified samples are drawn using a two step procedure. First, a positive sample is drawn uniformly from all constant-coset samples. Then, two bits are permuted between cosets such that the input is no longer constant over cosets.

For experiments over D_{40}, Figure 2 clearly shows that the G-CNN learns with significantly fewer samples than the FCN. Furthermore, comparisons between different group sizes shown in Figure 3 shows that FCNs need to learn on a number of samples that scales quadratically with the dimension of the group. This is in stark contrast to G-CNNs which learn with a constant number of samples regardless of the group size.

5 Discussion

Our results show that there exist provable sample complexity gaps between G-CNNs and FCNs. Though we proved sample complexity gap of $\Omega(|G|)$, experiments indicated that the sample complexity gap was higher in reality at $\Omega(|G|^2)$. Future work can analyze this larger separation and find functions which require sample complexity gaps that scale even more than $\Omega(|G|^2)$.

Since our analysis was focused on obtaining a provable sample complexity gap, we made a number of assumptions in our work to ensure the central concept was provable. Here, we list some of those assumptions and discuss future work that can help expand the ideas discussed in this study:

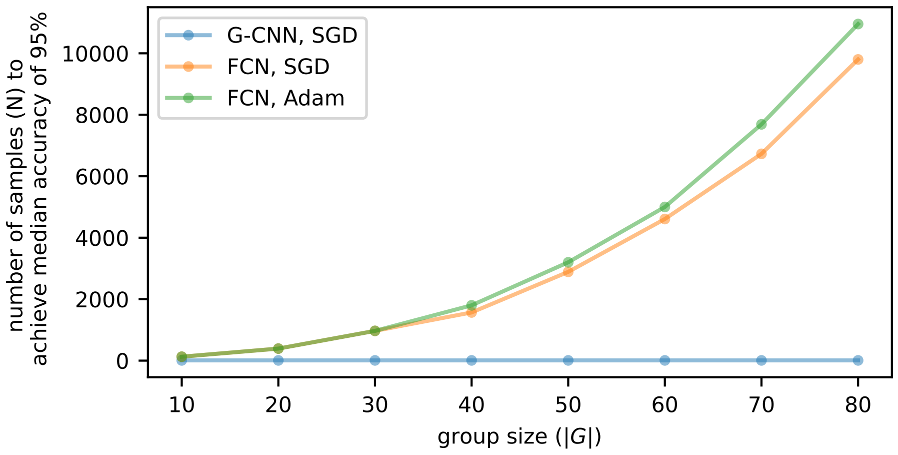


Figure 3: Number of samples needed to achieve 95 percent accuracy scales quadratically with the group size. Here, the median accuracy of 25 different runs for each group size is plotted.

- The pooling operations shown in Figure 1 were cleverly chosen to only allow the network to learn the target function. These pooling operations are rather restrictive and it is an open question whether they can be changed or relaxed to see if the results still hold.
- Sample complexity gaps are clearly dependent on the structure of the group or the target function. The sample complexity gap that we prove is consistent with those shown in prior work [11, 7]. It would be interesting to explore more complicated groups or target functions that could exhibit larger sample complexity gaps. Ref. [9] for example analyzes parity functions which can be harder for FCNs to learn.
- Dataset distributions in this analysis were rather contrived since the uniform distribution over binary data is heavily skewed towards inputs that are not constant over cosets. In our proofs, we explored a simple distribution that had support over $O(|G|^2)$ elements (to make proofs feasible) and later analyzed a distribution with much larger support over elements in the experiments. Future work can identify more “natural" problems which are not inherently skewed over the uniform distribution.

Looking forward, it is an interesting open question whether these results can be expanded by eliminating some of the assumptions made in this study and analyzing sample complexity gaps for more complex functions.

References

[1] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[2] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[4] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning*, pages 2747–2755. PMLR, 2018.

[5] Hannah Lawrence, Kristian Georgiev, Andrew Dienes, and Bobak T Kiani. Implicit bias of linear equivariant networks. *arXiv preprint arXiv:2110.06084*, 2021.

[6] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[7] Zhiyuan Li, Yi Zhang, and Sanjeev Arora. Why are convolutional nets more sample-efficient than fully-connected nets? *arXiv preprint arXiv:2010.08515*, 2020.

[8] Takanori Maehara and Hoang NT. A simple proof of the universality of invariant/equivariant graph neural networks. *arXiv preprint arXiv:1910.03802*, 2019.

[9] Eran Malach and Shai Shalev-Shwartz. Computational separation between convolutional and fully-connected networks. *arXiv preprint arXiv:2010.01369*, 2020.

[10] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International conference on machine learning*, pages 4363–4371. PMLR, 2019.

[11] Andrew Y Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78, 2004.

[12] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[13] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[14] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, 2015.

A Additional numerical experiment

Our proof of a sample complexity gap was performed over the data distribution described in Section 2. As a reminder, datasets for this distribution have support only over $O(|G|)$ elements. Figure 4 shows that the G-CNN also learns this target function with $O(1)$ samples whereas the FCN fails to learn the target function. As indicated in the main text, the FCN empirically requires $O(|G|^2)$ samples to learn the target function and our results here agree with this finding.

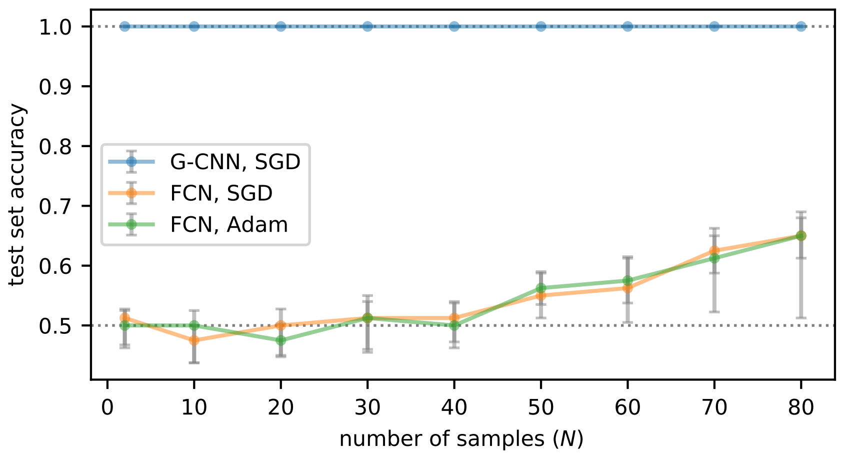


Figure 4: Numerical experiment with the data distribution in Section 2. G-CNN learns the target function even with very few samples whereas FCN fails to generalize. Here the group $G = D_{80}$. Networks are trained in 25 trials, each over 15000 epochs using the learning algorithm indicated in the legend .