
Equivariance versus Augmentation for Spherical Images

Jan E. Gerken^{1,2,3} Oscar Carlsson¹ Hampus Linander⁴ Fredrik Ohlsson⁵ Christoffer Petersson^{6,1}
Daniel Persson¹

Abstract

We analyze the role of rotational equivariance in convolutional neural networks (CNNs) applied to spherical images. We compare the performance of the group equivariant networks known as S2CNNs and standard non-equivariant CNNs trained with an increasing amount of data augmentation. The chosen architectures can be considered baseline references for the respective design paradigms. Our models are trained and evaluated on single or multiple items from the MNIST or FashionMNIST dataset projected onto the sphere. For the task of image classification, which is inherently rotationally *invariant*, we find that by considerably increasing the amount of data augmentation and the size of the networks, it is possible for the standard CNNs to reach at least the same performance as the equivariant network. In contrast, for the inherently *equivariant* task of semantic segmentation, the non-equivariant networks are consistently outperformed by the equivariant networks with significantly fewer parameters. We also analyze and compare the inference latency and training times of the different networks, enabling detailed tradeoff considerations between equivariant architectures and data augmentation for practical problems. The equivariant spherical networks used in the experiments are available at https://github.com/JanEGerken/sem_seg_s2cnn.

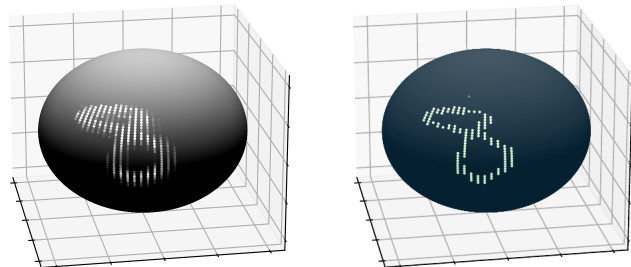


Figure 1: Sample from the spherical MNIST dataset used for semantic segmentation. Left: input data. Right: segmentation mask.

1. Introduction

In virtually all computer vision tasks, convolutional neural networks (CNNs) consistently outperform fully connected architectures. This performance gain can be attributed to the weight sharing and translational equivariance of the CNN layers: The network does not have to learn to identify translated versions of an image, since the inductive bias of equivariance already implies this identification. In contrast, fully connected layers require many more training samples to learn an effective form of equivariance. One way of ameliorating this problem is to supply the network with translated copies of the original training images, a form of data augmentation. However, training with this kind of data augmentation requires much longer training times and the performance of CNNs may not be reached in this way.

A crucial point is that in the equivariant setting, we are viewing the input image not only as a vector, but as a signal defined on a two-dimensional grid. The translational symmetry then has a geometric origin: it is a symmetry of the grid. From this perspective it is natural to envision generalizations of CNNs where not only translational symmetries are implemented but also more general transformations, such as rotations. Examples of networks that realize this property are group equivariant CNNs which are equivariant with respect to rotations on the plane (Cohen & Welling, 2016), or spherical CNNs which are equivariant with respect to rotations of the sphere (Cohen et al., 2018). Similarly to the case of translations discussed above, these symmetry properties of the data can be learned approximately by a non-equivariant model using data augmentation.

¹Department of Mathematical Sciences, Chalmers University of Technology, Gothenburg, Sweden ²Machine Learning Group at Berlin Institute of Technology, Berlin, Germany ³Berlin Institute for the Foundations of Learning and Data (BIFOLD), Berlin, Germany ⁴Department of Physics, University of Gothenburg, Gothenburg, Sweden ⁵Department of Mathematics and Mathematical Statistics, Umeå University, Umeå, Sweden ⁶Zenseact, Gothenburg, Sweden. Correspondence to: Daniel Persson <daniel.persson@chalmers.se>.

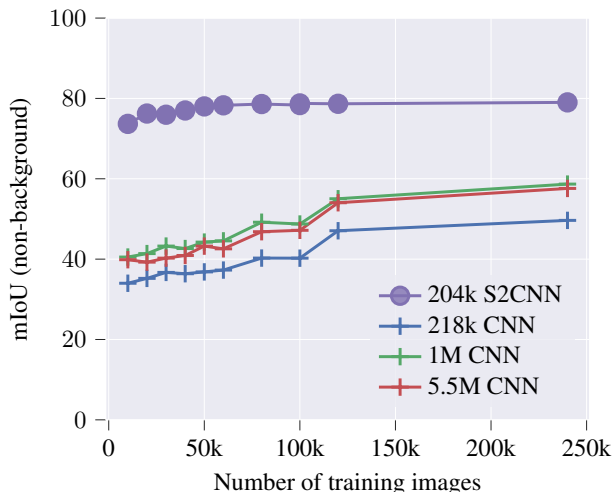


Figure 2: *Semantic Segmentation on Spherical MNIST*. The performance of equivariant (S2CNN) and non-equivariant (CNN) semantic segmentation models for various amounts of data augmented spherical MNIST single digit training images. Performance is measured in terms of mean intersection over union (mIoU) for the non-background classes. The numbers in the model names refer to the number of trainable parameters. The non-equivariant models are trained on randomly rotated samples, whereas the equivariant models are trained on unrotated samples.

In the present paper, we evaluate the performance of equivariant classification and semantic segmentation models (S2CNNs (Cohen et al., 2018)) on spherical image data subject to rotational symmetry (see Figure 1 for an example), and compare it to the performance obtained using rotational data augmentation in ordinary CNNs (see Figure 2). We conduct comparisons for several datasets to elucidate the influence of the type and complexity of the task on the performance. Our overall aim is to investigate whether equivariant architectures offer an inherent advantage over data augmentation in non-equivariant CNN models.

Our choice of using an ordinary CNN, without additional structure to make it more compatible with spherical data, is motivated by the desire to isolate the limits of geometric data augmentation.

While our present investigation is concerned with rotational equivariance in spherical data, group equivariant networks generalize to any homogeneous space with a global action of a symmetry transformation. In fact, the theoretical development in Section 2 applies to any homogeneous space, and the question of equivariance versus data augmentation is relevant also in the general setting.

1.1. Summary of contributions and outline

Here we list the main contributions of the paper, both theoretical and experimental.

- We define a new group equivariant CNN layer (Equation (2.5)), designed for the task of semantic segmentation for arbitrary Lie groups G . This generalizes previous results in (Linmans et al., 2018).
- We extend the S2CNN architecture by adding a layer which allows for equivariant outputs on the sphere, as required for semantic segmentation tasks, and present a detailed proof of equivariance (Appendix A).
- We demonstrate that non-equivariant classification models require considerable data augmentation to reach the performance of smaller equivariant networks (Section 3.2).
- We show that the performance of non-equivariant semantic segmentation models saturates well below that of equivariant models as the amount of data augmentation is increased (Section 3.3). We confirm that this result still holds when the complexity of the original segmentation task is increased (Section 3.4).
- We perform detailed inference time profiling of the GPU-based implementation of the equivariant models, and measure the throughput overhead compared to the non-equivariant models. Our results indicate that most inference time is spent in the last network layers processing the largest $SO(3)$ tensors, suggesting possible avenues towards optimizing the S2CNN architecture (Section 3.6).
- We show that the total training time for an equivariant model is shorter compared to a non-equivariant model at matched performance (Section 3.6).

Appendix A contains mathematical details about our new final layer used for semantic segmentation. Details of the model generation for the equivariant and non-equivariant networks can be found in Appendix B. Appendix C contains details of the datasets and augmentation. Finally, the latency profiling is summarized in Appendix D.

1.2. Related literature

The theory of group equivariant neural networks was first developed in (Kondor & Trivedi, 2018; Cohen et al., 2019; Esteves, 2020), generalizing the ordinary planar convolution of CNNs. An implementation for spherical data with a rotational $SO(3)$ symmetry was introduced in (Cohen et al., 2018; Kondor et al., 2018; Cobb et al., 2021).

There are two main approaches to equivariance in semantic segmentation tasks. In the first, one utilizes the methods available for ordinary flat CNNs by modifying the shape of convolutional kernels to compensate for spherical distortion (Tateno et al., 2018), by cutting the sphere in smaller pieces or otherwise preprocessing the data to be able to apply an ordinary CNN without much distortion (Zhang et al., 2019; Lee et al., 2019; Haim et al., 2019; Eder & Frahm, 2019; Du et al., 2021; Shakerinava & Ravanbakhsh, 2021). In the second approach, one avoids spherical distortions by Fourier decomposing signals and filters on the sphere (Esteves et al., 2020). Semantic segmentation for group equivariant CNNs is analyzed in (Linmans et al., 2018).

Data augmentation in the context of symmetric tasks was studied previously in (Gandikota et al., 2021), where a method to align input data is presented and compared to data augmentation. Closest to our work is a comparison between an equivariant model and a non-equivariant model for reduced training data sizes for an MRI application in (Müller et al., 2021). In contrast, we systematically compare data augmentation with increased training data sizes for different tasks, datasets and several non-equivariant models with an equivariant architecture.

2. Theory

2.1. Group equivariant networks

In this section we introduce the basic mathematical structure of GCNNs. Let G be a group and $H \subset G$ a closed subgroup¹. The input space of the network is the homogeneous space $\mathcal{M} = G/H$, meaning that a feature map in the first layer is a map

$$f : G/H \rightarrow \mathbb{R}^K, \quad (2.1)$$

where K denotes the number of channels. This feature map represents the input data, e.g. the pixel values of an image represented on the homogeneous space G/H . Features of subsequent layers are obtained using the convolution between the feature map f and a filter $\kappa : G/H \rightarrow \mathbb{R}^{K',K}$ defined by

$$(\kappa \star f)(g) = \int_{G/H} \kappa(g^{-1}x)f(x) dx, \quad (2.2)$$

where $g \in G$ and dx is the invariant measure on G/H induced from the Haar measure on G .

The action of G extends from $\mathcal{M} = G/H$ to G itself and the convolution (2.2) is equivariant with respect to this action, $g(\kappa \star f) = \kappa \star (gf)$, $g \in G$, meaning that the result obtained by transforming the convoluted feature map $\kappa \star f$ by an element $g \in G$ is identical to that obtained by con-

volving the transformed feature map gf . In other words, the transformation $g \in G$ commutes with the convolution.

The networks considered here all have scalar features throughout the architecture. In general, however, the group G can act through non-trivial representations on both the manifold G/H and the feature maps (see e.g. (Kondor & Trivedi, 2018; Cohen et al., 2019; Aronsson, 2021; Gerken et al., 2021)).

Note that the output of the convolution (2.2) is a function on G , rather than on the homogeneous space G/H . Limiting feature maps to convolutions on the original manifold G/H is in general far too restrictive, as was first noticed in the context of spherical signals by (Makadia et al., 2007). In order to keep the network as general as possible, and maximize the expressiveness of each individual layer, subsequent convolutions will all be taken on the group G ,

$$(\kappa \star f)(g) = \int_G \kappa(g'^{-1}g)f(g') dg', \quad (2.3)$$

$g \in G$, except for the last one which will be discussed in more detail below.

2.2. Semantic segmentation and equivariance

In the final layers we must take into account the desired task that the network is designed to perform. If we are interested in a classification task, we want the entire network to be invariant with respect to G . This can be achieved by integrating over G following the final convolution

$$f^{\text{final}} = \int_G f(g) dg. \quad (2.4)$$

For semantic segmentation, however, we are classifying individual pixels into a segmentation mask. The network output should therefore be equivariant with respect to transformations of the input. One way to achieve this aim is to define a final convolution

$$f^{\text{final}}(x) = \int_G \kappa(g'^{-1}g_x)f(g') dg', \quad (2.5)$$

where $g_x \in G$ is a representative in G corresponding to the point x in the coset space G/H and κ is required to be an H -invariant kernel on G . This generalizes the convolution defined in (Linmans et al., 2018) for finite roto-translations on the plane. The output (2.5) is then a signal on G/H , rather than G , and equivariance ensures that a transformation by G on the input image will result in the same transformation on the output.

Alternatively, similarly to the classification case, an output segmentation mask on G/H can be obtained by integrating along the orbits of the subgroup H

$$f^{\text{final}}(x) = \int_H f(g_x h) dh, \quad (2.6)$$

¹In this paper, we will always consider the case where G is a Lie group and H is a compact subgroup.

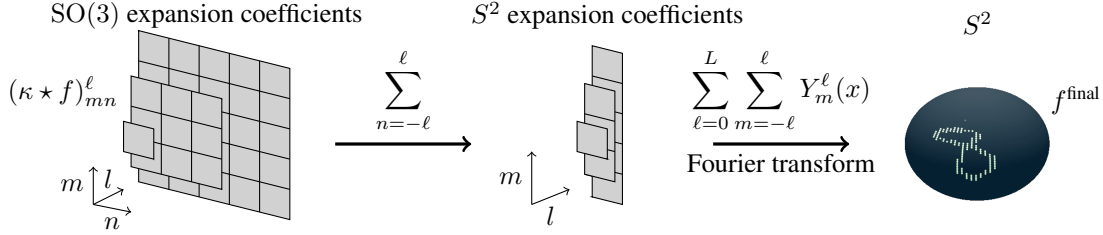


Figure 3: The final layer (3.3) of our fully equivariant architecture takes Fourier coefficients on $\text{SO}(3)$ and sums them over n to yield Fourier coefficients on the sphere, representing a segmentation mask.

with g_x defined as above. This segmentation mask is equivariant with respect to the action of G by construction.

3. Data augmentation for classification and semantic segmentation

In this section, we investigate several aspects of the performance difference between equivariant and non-equivariant models, applied to both classification and segmentation tasks on spherical images. Consequently, we consider input data in the form of grayscale images defined on the sphere $\mathcal{M} = S^2$ with continuous rotational symmetry $G = \text{SO}(3)$. Taking $H = \text{SO}(2)$ to be the isotropy subgroup of a point $x \in S^2$, the sphere can be expressed as a homogeneous space $S^2 = \text{SO}(3)/\text{SO}(2)$.

3.1. Equivariant model

As discussed in Section 2, the equivariant network architecture for spherical signals necessitates a redefinition of the convolution compared to ordinary CNNs. For our implementation we will use the S2CNN architecture of (Cohen et al., 2018), in the implementation available at (Köhler et al., 2021), adapted in the case of semantic segmentation.²

For S2CNNs the first layer takes an input feature map f defined on S^2 , convolves it with a kernel κ defined on $\text{SO}(3)$ and outputs a feature map defined on $\text{SO}(3)$ according to

$$(\kappa \star f)(R) = \int_{S^2} \kappa(R^{-1}x) f(x) dx, \quad (3.1)$$

where $R \in \text{SO}(3)$. In subsequent layers, the input feature map f is also defined on $\text{SO}(3)$ and the convolution takes the form

$$(\kappa \star f)(R) = \int_{\text{SO}(3)} \kappa(S^{-1}R) f(S) dS. \quad (3.2)$$

In the S2CNN architecture, these convolutions are computed in the respective Fourier domain of S^2 and $\text{SO}(3)$. For S^2 , this amounts to an expansion in spherical harmonics Y_m^ℓ ;

²Our adaptation is available at https://github.com/JanEGerken/sem_seg_s2cnn.

for $\text{SO}(3)$, to an expansion in Wigner matrices \mathcal{D}_{mn}^ℓ with $\ell = 0, \dots, L$ and $m, n = -\ell \dots \ell$ for some bandlimit L .³

The original S2CNN architecture introduced in (Cohen et al., 2018) was used for classification tasks and hence in the last convolutional layer the feature map was integrated over $\text{SO}(3)$ to render the output invariant under rotations of the input. We use the same setup for the classification task discussed below.

In contrast, for semantic segmentation, we need an equivariant network, as detailed in Section 2.2 above. To this end, instead of the Fourier-back-transform on $\text{SO}(3)$, we use for the last convolution

$$f^{\text{final}}(x) = \sum_{\ell=0}^L \sum_{m=-\ell}^{\ell} \sum_{n=-\ell}^{\ell} (\kappa \star f)_{mn}^\ell Y_m^\ell(x), \quad (3.3)$$

where $\kappa \star f$ is as in (3.2) on $\text{SO}(3)$. The sum over n corresponds roughly to the Fourier space version of the integral over H presented in Equation (2.6) and ensures that the output f^{final} lives on S^2 , as illustrated in Figure 3. We provide more mathematical details in Appendix A.

After a softmax nonlinearity, we interpret f^{final} as the predicted segmentation mask and use a point-wise cross entropy loss for training.

To facilitate the segmentation task, we use a sequence of downsampling $\text{SO}(3)$ convolutions followed by a sequence of upsampling convolutions. These networks do not contain any fully connected layers. For the classification task, we only use the downsampling layers, which are, after the integration over $\text{SO}(3)$, followed by three fully connected layers, alternated with batch-normalization layers.

The downsampling layers are realized in the S2CNN framework by choosing the bandlimit for the Fourier-back-transform to be lower than the bandlimit of the Fourier transform, i.e. we are dropping higher order modes in the result of the convolution. This is similar to strides used in

³Note that following the original S2CNN code (Cohen et al., 2018), we use Wigner matrices for the Fourier transform and complex conjugated Wigner matrices for the inverse Fourier transform, contrary to the usual convention.

downsampling layers of ordinary CNNs. Since the Fourier transforms of the feature maps are computed using the FFT algorithm, this also decreases the spatial resolution of the feature maps.

For the upsampling layers, we select the Fourier-back-transform to have higher bandlimit than the Fourier transform. The missing Fourier modes are here filled with zeros, yielding a feature map of higher spatial resolution (but of course with still lower resolution in the Fourier domain). This is similar to upsampling by interpolation.

To fix the precise architectures and hyperparameters for the experiments without biasing for or against the equivariant models, we generated architectures at random. For the equivariant models, we generated 20 models at random for each parameter range and selected the one performing best on a reference task, as detailed for the semantic segmentation models in Appendix B.2. This resulted in two equivariant models with 204k and 820k parameters, respectively for the semantic segmentation tasks and one model with 150k parameters for the classification task.

For the semantic segmentation tasks, we generated three non-equivariant models as a baseline with 218k, 1M and 5.5M parameters, respectively. Each of these performed best out of 20 randomly generated models with a fixed parameter budget. The non-equivariant models have (ordinary) convolutional layers for downsampling and transpose convolutions for upsampling layers which mirror the image dimensions and channels of the downsampling layers exactly. We add skip connections over each (transpose) convolution, resulting in a ResNet-like architecture. For details, cf. Appendix B.1. The precise architectures of the models used in our experiments are summarized in Tables 4 and 5 in the appendix.

Similarly, for the classification task, we generated three non-invariant models with 114k, 0.5M and 2.5M parameters, respectively. These models only have downsampling layers, followed by an average pooling layer and three fully connected layers alternated with batch-normalization layers. We did not use skip connections in this case.

3.2. Classification

The primary question we want to investigate in this work is whether data augmentation can make up for the benefits of equivariant network architectures. We first study this question in the context of classification by training equivariant and non-equivariant models on training data with different amounts of data augmentation, as depicted in Figure 4. The input data consists of single digits sampled with replacement from the MNIST (Lecun et al., 1998) dataset which are projected onto the sphere and labeled with the classes of the digits. The spherical pictures are rotated by a random rota-

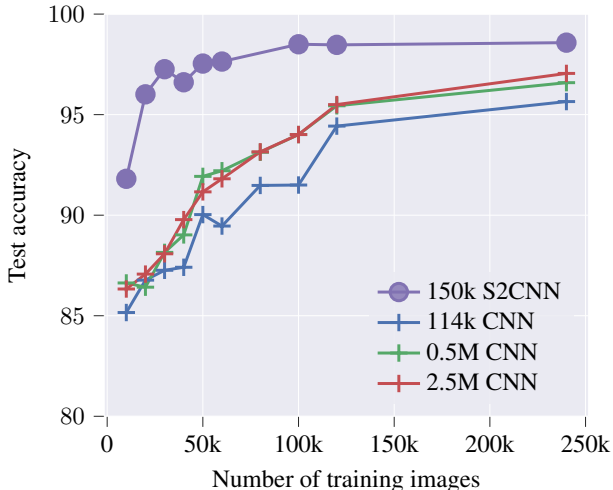


Figure 4: *Image Classification on Spherical MNIST*. Accuracy of invariant and non-invariant classification models for various amounts of data augmentation as in Figure 2. Training data for the equivariant model is sampled randomly with replacement from the available 60k training samples.

tion matrix in $SO(3)$. A sample is depicted in the left panel of Figure 1. For dataset larger than the 60k original MNIST dataset, digits are necessarily repeated but have different rotations, hence these correspond to data augmentation as compared to the original dataset. The equivariant networks were trained on unrotated spherical images with the digits being projected on the southern hemisphere, sampled randomly with replacement from the original 60k training samples of MNIST.⁴

Figure 4 shows that for the smaller data regimes the accuracy of the equivariant model dominates the non-equivariant models, even though it uses fewer parameters than the non-equivariant models. Whereas the non-equivariant models continue to benefit from the increased dataset size, the equivariant model does not improve beyond the original 60k training samples of MNIST as expected. Looking at the trend of the non-equivariant models in Figure 4 it is unclear if they would eventually match the equivariant model for large enough augmented datasets. It turns out that for the task of spherical MNIST classification, a large enough non-equivariant CNN trained on augmented data can achieve similar performance to an equivariant model, see Figure 10 left.

A priori, it is not clear whether increasing model size and data augmentation is always sufficient for non-equivariant models to match the performance of equivariant models. What is clear is that spherical MNIST classification leaves much to be desired in terms of task and dataset complexity.

⁴Note that we did not take special precautions to treat ambiguous cases such as “6” vs. “9”.

3.3. Semantic segmentation

Semantic segmentation is an interesting example of a task where the output is equivariant, in contrast to the invariance of classification output. This example lets us investigate if data augmentation on large enough non-equivariant models can match the equivariant models not only in classification tasks, but also in more difficult and truly equivariant tasks.

For our experiments, we replaced the classification labels from the spherical MNIST dataset described above by segmentation masks. A sample mask is depicted in the right panel of Figure 1, further details can be found in Appendix C.

In order to investigate whether data augmentation can push the performance of non-equivariant networks to match that of equivariant architectures, we trained the non-equivariant networks on rotated samples and compare to equivariant networks trained only on non-rotated samples. For evaluation, we use the mean intersection over union (mIoU) where we drop the background class. More technical details can be found in Appendix C.

The plot in Figure 2 shows the results of this experiment. Sample predictions for the same task with four digits are shown in Appendix C.4. As expected, more training data (and hence stronger data augmentation) increases the performance of the non-equivariant models. However, the equivariant models outperform the non-equivariant models even for copious amounts of data augmentation. As is also shown in Figure 2, larger models outperform smaller models, as expected. However, there seems to be a saturation of this effect as the 5.5M parameter model performs on par (within statistical fluctuations) with the 1M parameter model. Notably though, even the largest model trained on data augmented with a factor of 20 cannot outperform the equivariant models. We also trained the non-equivariant models on even larger datasets with up to 1.2M data points, but their performance saturates at a level comparable to what is reached at 240k train data points, see Figure 10 right.

We also trained larger spherical models to see if we could push performance even further. However, even the 820k parameter model specified in Table 5 in the appendix performed only at 76.71% non-background mIoU for 60k training data points. This is on par with the performance of 78.28% non-background mIoU that the 204k spherical model reached for this dataset and hence suggests that the smaller model already exhausts the chosen architecture for this problem. Since the larger model requires much more compute, we performed all the remaining experiments only with the 204k spherical model.

In order to verify that our non-equivariant models are in principle expressive enough to learn the given datasets, we also trained them on unrotated data, as we did for the spher-

ical models. From the plot in the left panel of Figure 5, it is clear that all the models perform well when evaluated on the same data on which they were trained. As shown in the right panel of Figure 5, performance deteriorates to almost random guessing (as expected) if the models are evaluated on rotated test data. The performance of the equivariant models is identical for both cases. For the task of training and evaluation on unrotated data, which contains no symmetries, the considerably larger non-equivariant models slightly outperform the equivariant model. The performance of the smallest non-equivariant model is on par with that of the equivariant model.

Note that the performance of the non-equivariant models furthermore crucially depends on where on the spherical grid the digits are projected. For the runs depicted in Figure 5, the digits were projected in the center of the Driscoll-Healy grid, so that the ordinary CNNs could benefit maximally from their translation equivariance and distortions were minimal. In Appendix C.3 we show that performance is greatly reduced if the digits are projected closer to the pole of the sphere, where distortions in the equirectangular projection of the Driscoll-Healy grid are maximal.

3.4. Dataset complexity

The experiments described in the previous section used a very simple dataset, and since only one class needed to be predicted, all non-zero pixels belonged to the same foreground class. We have therefore investigated if the observed performance gain of the equivariant models persists for more complex datasets.

The first modification we performed on the dataset depicted in Figure 1 is to project four MNIST digits onto the same sphere and construct a corresponding segmentation mask. A sample from this dataset is shown in Figure 6. The results of these experiments are summarized in Figure 7, sample predictions for the best equivariant and non-equivariant models are shown in Appendix C.4. Note that the non-monotonic increase in performance in Figure 7 is due to sampling effects during the data generation. We have explicitly verified that these features are within the range of the statistical fluctuations of this sampling and since this affects all models equally, these features are irrelevant for the model comparison. More details on this point are given in Appendix C.2.

Moreover, for an increased number of digits, we observe a clear benefit of the equivariant architectures. Again, increasing model sizes beyond a certain parameter count does not translate into an increase in performance. In comparison to Figure 2, we see that all models in Figure 7 benefited from seeing more samples during training, with a steeper increase in performance with number of training samples and earlier saturation.

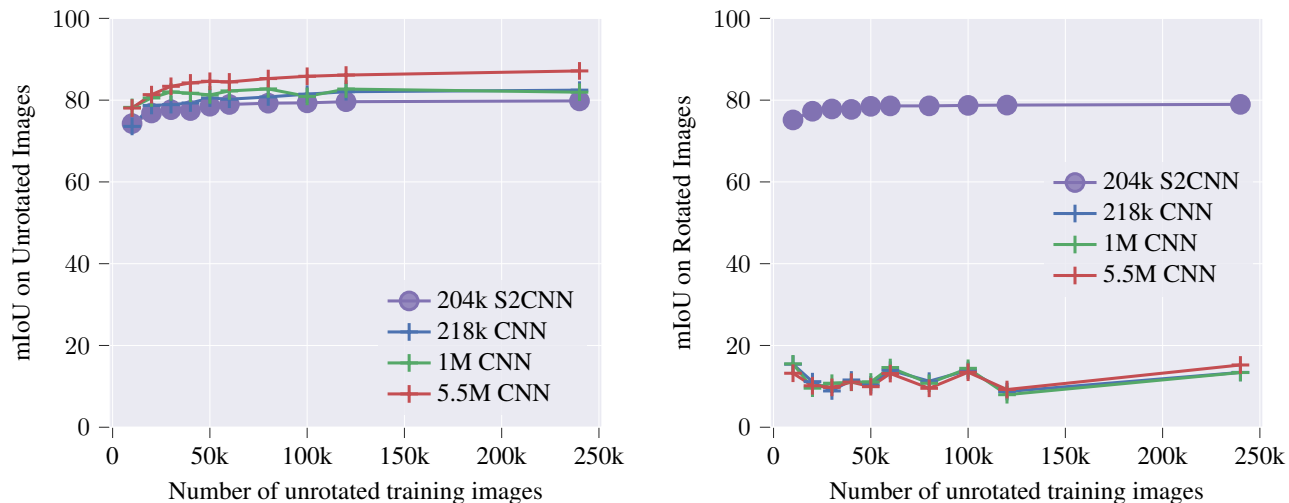


Figure 5: *Training on Unrotated Images*. Performance of equivariant and non-equivariant models in semantic segmentation for various amounts of data augmentation for models trained on unrotated data with one digit. Performance is measured in terms of mIoU for the non-background classes. Left: evaluated on unrotated test data. Right: evaluated on rotated test data.

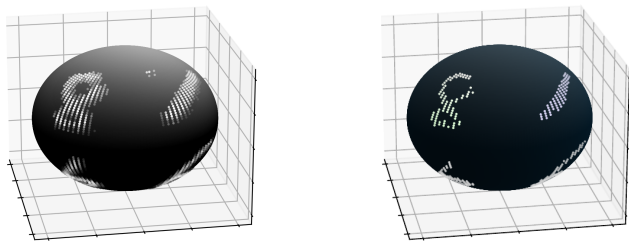


Figure 6: Sample from the spherical MNIST dataset with four digits projected onto the sphere. Left: input data. Right: segmentation mask.

In addition to increasing the number of MNIST digits in each datapoint, we have also made the object recognition aspect more difficult by swapping the MNIST digits for items of clothing from the FashionMNIST dataset (Xiao et al., 2017). A sample datapoint is depicted in Figure 8. The results shown in Figure 9 reflect the higher difficulty of this task: The performance of all models is lower compared to Figure 2. This is partly due to the difficulty of constructing segmentation masks for the items of clothing, as discussed in Appendix C. However, the equivariant models still outperform the non-equivariant models for all training data sizes that we tried by a large margin.

3.5. Non-equivariant performance saturation

To check whether the non-equivariant models could be pushed to the performance of the equivariant models by extending the training data even further, we trained our non-equivariant models on larger datasets.

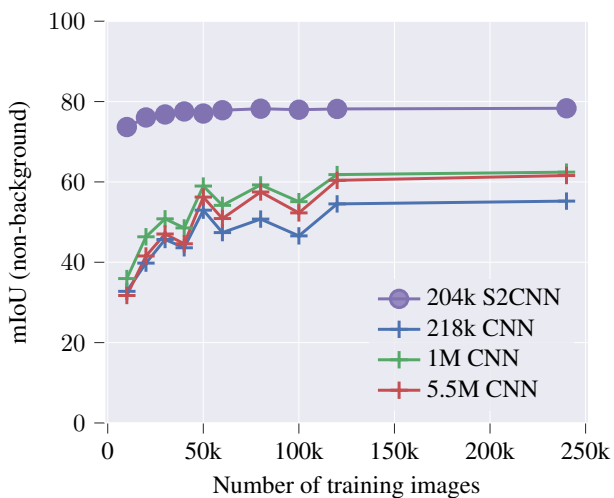


Figure 7: *Segmentation on four digit Spherical MNIST*. Performance of equivariant and non-equivariant models in semantic segmentation for various amounts of data augmentation as in Figure 2, but with four digits projected onto the sphere.

For spherical MNIST classification the non-equivariant models can match the accuracy of the smaller equivariant model with enough augmented data, cf. Figure 10 left. On the other hand, for semantic segmentation, as shown in Figure 10 right, even these much larger training datasets did not improve the test performance of the non-equivariant models.

These experiments support the intuition that for equivariant tasks, equivariant models perform so much better than non-equivariant models that even very large amounts of data

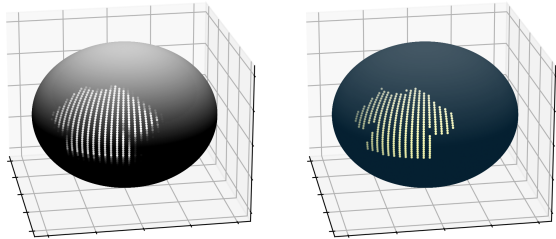


Figure 8: Sample from the spherical FashionMNIST dataset. Left: input data. Right: segmentation mask.

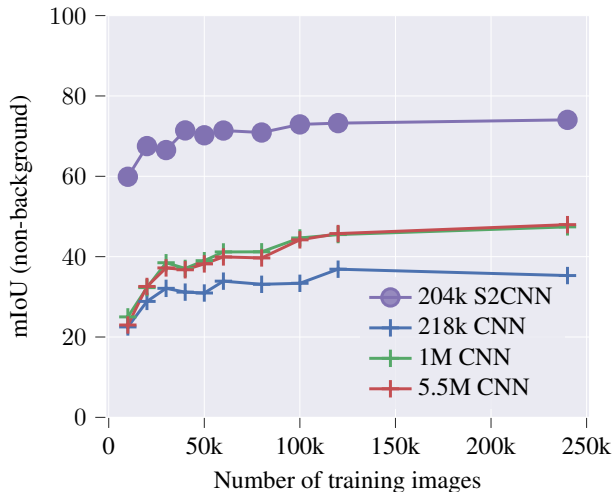


Figure 9: *Segmentation on Spherical FashionMNIST*. Performance of equivariant and non-equivariant models in semantic segmentation for various amounts of data augmentation as in Figure 2, but with items of clothing from the FashionMNIST dataset projected onto the sphere.

augmentation cannot compensate for this advantage. In contrast, although equivariant models (which were made invariant only in the last layer) still show higher performance for invariant tasks, non-equivariant models can ultimately reach the same performance if enough data augmentation is applied.

3.6. Inference latency and training times

At similar parameter counts, the GPU based implementation (Cohen et al., 2018) of the equivariant layers have an order of magnitude higher inference latency as can be seen in Tables 1 and 2.

A detailed profiling of the model and CUDA implementation shows that the bulk of the inference time is spent in the later upsampling layers which compute the largest $SO(3)$ tensors in the network. In terms of operations, almost half of the time is spent in the custom implementation of the complex matrix multiplication of $SO(3)$ tensors. There is also a

Table 1: Runtime latency and throughput for the equivariant semantic segmentation model (204k S2CNN in Appendix Table 5) on an Nvidia T4 16GB GPU. Latency measures the time of a forward pass through the model on the GPU, throughput is the corresponding number of samples per second given the batch size. The larger batch size is chosen to maximize the throughput for the T4.

Batch size	Latency (ms)	Throughput (N/s)
1	111 ± 0.6	9.0 ± 0.04
7	479 ± 2.2	14.6 ± 0.07

Table 2: Runtime latency and throughput for non-equivariant CNN model (200k CNN in Appendix Table 5) on an Nvidia T4 16GB GPU. Latency measures the time of a forward pass through the model on the GPU, throughput is the corresponding number of samples per second given the batch size. The larger batch size is chosen to maximize the throughput for the T4.

Batch size	Latency (ms)	Throughput (N/s)
1	5.93 ± 0.24	169 ± 5.8
60	87.98 ± 0.17	682 ± 1.3

significant overhead in the transformation back and forth to the Fourier domain. See Appendix D for the details on the profiling of layers and operations.

The backpropagation latency of the equivariant models mirrors the inference latency resulting in about an order of magnitude slower training as compared to the non-equivariant CNNs. Note that even though the training is slower, because of the increase in data augmentation needed for the non-equivariant models, at a fixed performance goal the equivariant model actually trains faster, cf. Table 3 for training times for the classification task at about 97.5% accuracy level. A similar comparison for semantic segmentation is less relevant as even large amount of data augmentation leaves a big performance gap compared to the equivariant model.

4. Conclusions

Our results indicate that equivariant models possess an inherent advantage over non-equivariant ones, which cannot be overcome by data augmentation when applied to tasks probing the full rotational equivariance of the spherical image data. In order to corroborate and generalize these findings, several extensions of the current study are natural to pursue. In particular, it would be interesting to consider non-equivariant models specifically adapted to the sphere.

Furthermore, even though our results indicate that the advantage of equivariance is not explained by low data com-

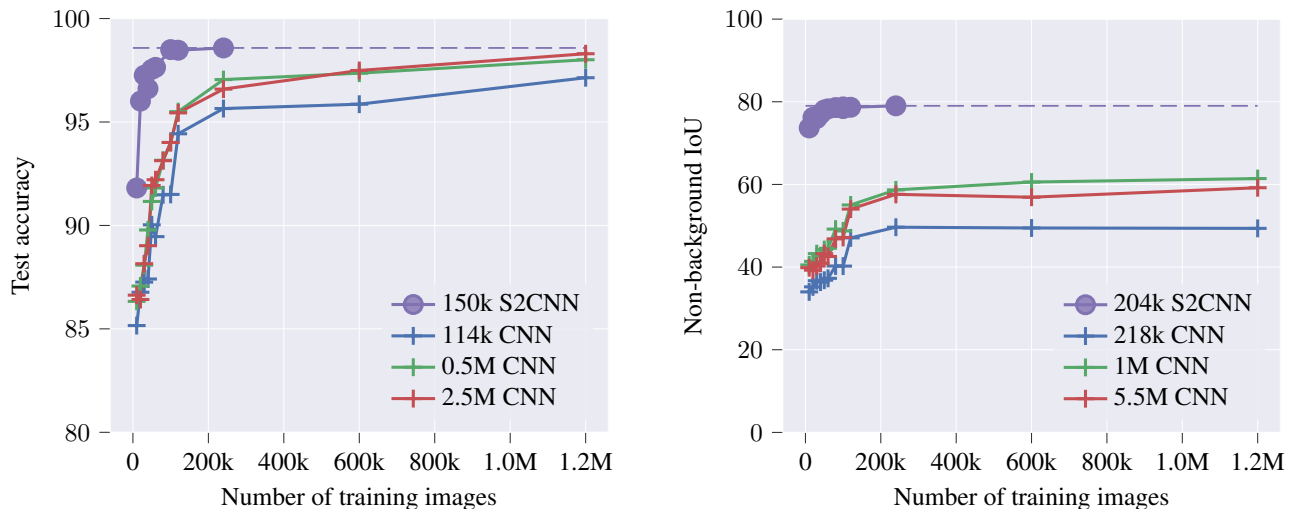


Figure 10: *Non-equivariant performance saturation for segmentation.* Left: For classification of spherical MNIST as in Figure 4, the non-equivariant models reach the test accuracy of the equivariant models for very large amounts of data augmentation. Right: For semantic segmentation of one-digit spherical MNIST as in Figure 2, the non-background IoU of the non-equivariant models saturates well below the performance of the equivariant model even for moderately high amounts of data augmentation.

Table 3: Training times for the S2CNN classification model and non-equivariant CNN model at matched accuracy on rotated spherical images. The S2CNN model is trained on non-rotated images whereas the CNN is trained on an augmented dataset with rotated images. A single Nvidia T4 16GB was used for training.

Model	Accuracy	Training time
150k S2CNN	97.64%	15h
5M CNN	97.49%	26h

plexity, it would be interesting to investigate data sets which are both richer in complexity and native to the sphere rather than projected onto it.

In terms of inference latency the CUDA implementation of the equivariant convolution in the Fourier domain, together with the large $SO(3)$ tensors, is significantly slower than a traditional spatial convolution and the profiling shows where future optimizations should be targeted. With wider adoption it is likely that this situation would improve on multiple fronts.

It is appealing to think of symmetries as a fundamental design principle for network architectures. In this paper we use the symmetry of the sphere as a guiding principle for the network architecture. From this perspective it is natural to consider the question of how to train neural networks in the case of other “non-flat” data manifolds, i.e. when the domain \mathcal{M} is a (possibly curved) manifold. This research field is referred to as *geometric deep learning*, an umbrella

term first coined in (Bronstein et al., 2017) (see (Bronstein et al., 2021) and (Gerken et al., 2021) for recent reviews). The results of the present paper may thus be viewed as probing a small corner of the vast field of geometric deep learning.

Extending the exploration of inherent advantages of equivariance to other tasks, data manifolds and, possibly local, symmetry groups offers exciting prospects for future research.

Acknowledgments

We are very grateful to Jimmy Aronsson for valuable discussions and helpful feedback on the text. We also thank the anonymous referees for their comments.

The work of D.P. and O.C. is supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. D.P. and J.G. are supported by the Swedish Research Council, J.G. is also supported by the Knut and Alice Wallenberg Foundation and by the German Ministry for Education and Research (BMBF) under Grants 01IS14013A-E, 01GQ1115, 1GQ0850, 01IS18025A and 01IS18037A.

The computations were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at C3SE partially funded by the Swedish Research Council through grant agreement no. 2018-05973.

References

- Aronsson, J. Homogeneous vector bundles and G -equivariant Convolutional Neural Networks. *arXiv:2105.05400, to appear in Sampling Theory, Signal Processing, and Data Analysis*, May 2021. [3](#)
- Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, July 2017. doi: 10.1109/MSP.2017.2693418. [9](#)
- Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. *arXiv:2104.13478 [cs, stat]*, April 2021. [9](#)
- Cobb, O. J., Wallis, C. G. R., Mavor-Parker, A. N., Marignier, A., Price, M., d’Avezac, M., and McEwen, J. D. Efficient generalized spherical cnns. In *International Conference on Learning Representations*, Feb 2021. [2](#)
- Cohen, T., Geiger, M., and Weiler, M. A General Theory of Equivariant CNNs on Homogeneous Spaces. In Wallach, H., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., December 2019. [2, 3](#)
- Cohen, T. S. and Welling, M. Group Equivariant Convolutional Networks. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR. [1](#)
- Cohen, T. S., Geiger, M., Köhler, J., and Welling, M. Spherical CNNs. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. [1, 2, 4, 8](#)
- Driscoll, J. and Healy, D. Computing Fourier Transforms and Convolutions on the 2-sphere. *Advances in Applied Mathematics*, 15(2):202–250, June 1994. ISSN 0196-8858. doi: <https://doi.org/10.1006/aama.1994.1008>. [14](#)
- Du, H., Cao, H., Cai, S., Yan, J., and Zhang, S. Spherical Transformer: Adapting Spherical Signal to CNNs. *arXiv:2101.03848 [cs]*, January 2021. [3](#)
- Eder, M. and Frahm, J.-M. Convolutions on Spherical Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–5, June 2019. [3](#)
- Esteves, C. Theoretical Aspects of Group Equivariant Neural Networks. *arXiv:2004.05154 [cs, stat]*, April 2020. [2](#)
- Esteves, C., Makadia, A., and Daniilidis, K. Spin-Weighted Spherical CNNs. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 8614–8625. Curran Associates, Inc., October 2020. [3](#)
- Gandikota, K. V., Geiping, J., Löhner, Z., Czaplinski, A., and Moeller, M. Training or Architecture? How to Incorporate Invariance in Neural Networks. *arXiv:2106.10044 [cs]*, June 2021. [3](#)
- Gerken, J. E., Aronsson, J., Carlsson, O., Linander, H., Ohlsson, F., Petersson, C., and Persson, D. Geometric Deep Learning and Equivariant Neural Networks. *arXiv:2105.13926 [hep-th, cs]*, May 2021. [3, 9, 11](#)
- Haim, N., Segol, N., Ben-Hamu, H., Maron, H., and Lipman, Y. Surface Networks via General Covers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 632–641, October 2019. [3](#)
- Kondor, R. and Trivedi, S. On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2747–2755. PMLR, 10–15 Jul 2018. [2, 3](#)
- Kondor, R., Lin, Z., and Trivedi, S. Clebsch-Gordan Nets: A Fully Fourier Space Spherical Convolutional Neural Network. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., November 2018. [2](#)
- Köhler, J., Cohen, T. S., Geiger, M., and Welling, M. Spherical CNNs, October 2021. URL <https://github.com/jonkhler/s2cnn>. [4](#)
- Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. ISSN 1558-2256. doi: 10.1109/5.726791. [5](#)
- Lee, Y., Jeong, J., Yun, J., Cho, W., and Yoon, K.-J. SpherePHD: Applying CNNs on a Spherical PolyHeDron Representation of 360 degree Images. *arXiv:1811.08196 [cs]*, April 2019. [3](#)
- Linmans, J., Winkens, J., Veeling, B. S., Cohen, T. S., and Welling, M. Sample Efficient Semantic Segmentation using Rotation Equivariant Convolutional Networks. *International Conference on Machine Learning Workshop “Towards learning with limited labels: Equivariance, Invariance, and Beyond”*, July 2018. [2, 3](#)
- Makadia, A., Geyer, C., and Daniilidis, K. Correspondence-free Structure from Motion. *International Journal of Computer Vision*, 75(3):311–327, 2007. ISSN 1573-1405. doi: 10.1007/s11263-007-0035-2. [3](#)
- Müller, P., Golkov, V., Tomassini, V., and Cremers, D. Rotation-Equivariant Deep Learning for Diffusion MRI. *arXiv:2102.06942 [cs]*, February 2021. [3](#)
- Shakerinava, M. and Ravanbakhsh, S. Equivariant Networks for Pixelized Spheres. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9477–9488. PMLR, 18–24 Jul 2021. [3](#)
- Tateno, K., Navab, N., and Tombari, F. Distortion-Aware Convolutional Filters for Dense Prediction in Panoramic Images. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y. (eds.), *Computer Vision – ECCV 2018*, volume 11220 of *Lecture Notes in Computer Science*, pp. 732–750. Springer International Publishing, 2018. ISBN 978-3-030-01269-4 978-3-030-01270-0. doi: 10.1007/978-3-030-01270-0_43. [3](#)

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747 [cs, stat]*, September 2017. 7

Zhang, C., Liwicki, S., Smith, W., and Cipolla, R. Orientation-Aware Semantic Segmentation on Icosahedron Spheres. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3532–3540. IEEE, 2019. ISBN 978-1-72814-803-8. doi: 10.1109/ICCV.2019.00363. 3

A. Mathematical properties of the final S2CNN layer

In this appendix, we show the equivariance of the final S2CNN layer used for semantic segmentation (3.3) and give an interpretation in terms of the projection (2.6) from G to G/H . For convenience, we reproduce (3.3) here,

$$f^{\text{final}}(x) = \sum_{\ell=0}^L \sum_{m=-\ell}^{\ell} \sum_{n=-\ell}^{\ell} (\kappa \star f)_{mn}^{\ell} Y_m^{\ell}(x). \quad (\text{A.1})$$

A.1. Proof of equivariance

Using basic properties of Wigner matrices and spherical harmonics, we find for $R \in \text{SO}(3)$

$$f^{\text{final}}(Rx) = \sum_{\ell=0}^L \sum_{m,n=-\ell}^{\ell} (\kappa \star f)_{mn}^{\ell} Y_m^{\ell}(Rx) \quad (\text{A.2})$$

$$= \sum_{\ell=0}^L \sum_{m,n,k=-\ell}^{\ell} (\kappa \star f)_{mn}^{\ell} \overline{\mathcal{D}_{mk}^{\ell}(R)} Y_k^{\ell}(x) \quad (\text{A.3})$$

$$= \sum_{\ell=0}^L \sum_{m,n,k=-\ell}^{\ell} (\kappa \star f)_{mn}^{\ell} \mathcal{D}_{km}^{\ell}(R^{-1}) Y_k^{\ell}(x), \quad (\text{A.4})$$

where in (A.3) we used the transformation property of the spherical harmonics and in (A.4) that the Wigner matrices form a unitary representation. For a summary of the necessary formulae and the conventions⁵, see, e.g., (Gerken et al., 2021).

Using the definition of the Fourier coefficients on $\text{SO}(3)$,

⁵We want to reiterate that we use the complex conjugate conventions for Wigner matrices in comparison to the reference.

we obtain

$$\begin{aligned} & \sum_{m=-\ell}^{\ell} (\kappa \star f)_{mn}^{\ell} \mathcal{D}_{km}^{\ell}(R^{-1}) \\ &= \sum_{m=-\ell}^{\ell} \frac{2\ell+1}{8\pi^2} \int_{\text{SO}(3)} (\kappa \star f)(S) \mathcal{D}_{mn}^{\ell}(S) \mathcal{D}_{km}^{\ell}(R^{-1}) dS \\ &= \frac{2\ell+1}{8\pi^2} \int_{\text{SO}(3)} (\kappa \star f)(S) \mathcal{D}_{kn}^{\ell}(R^{-1}S) dS \\ &= \frac{2\ell+1}{8\pi^2} \int_{\text{SO}(3)} (\kappa \star f)(RS') \mathcal{D}_{kn}^{\ell}(S') dS' \\ &= (\kappa \star L_R f)_{kn}^{\ell}, \end{aligned} \quad (\text{A.5})$$

where we have used the notation $(L_R(f))(Q) = f(RQ)$ and the equivariance of (3.2) in the last step.

Plugging (A.5) into (A.4) completes the prove of equivariance of the last layer

$$f^{\text{final}}(Rx) = \sum_{\ell=0}^L \sum_{m,n=-\ell}^{\ell} (\kappa \star L_R f)_{mn}^{\ell} Y_m^{\ell}(x). \quad (\text{A.6})$$

A.2. Projection from $\text{SO}(3)$ to S^2

To see the connection between (A.1) and the integration over H (in this case $H = \text{SO}(2)$) in (2.6), we rewrite (A.1) in position space

$$\begin{aligned} f^{\text{final}}(x) &= \sum_{\ell=0}^L \sum_{m=-\ell}^{\ell} \sum_{n=-\ell}^{\ell} (\kappa \star f)_{mn}^{\ell} Y_m^{\ell}(x) \quad (\text{A.7}) \\ &= \sum_{\ell=0}^L \sum_{m,n=-\ell}^{\ell} \int_{\text{SO}(3)} (\kappa \star f)(S) \mathcal{D}_{mn}^{\ell}(S) Y_m^{\ell}(x) dS \\ &= \sum_{\ell=0}^L \sum_{n=-\ell}^{\ell} \frac{2\ell+1}{8\pi^2} \int_{\text{SO}(3)} (\kappa \star f)(S) Y_n^{\ell}(S^{-1}x) dS. \end{aligned}$$

Now, we factorize S into an element $\alpha \in \text{SO}(2)$ which stabilizes x and an element $y \in S^2$. This can always be done uniquely. With this, we obtain

$$f^{\text{final}}(x) = \int_{S^2} \widetilde{(\kappa \star f)}(y) \sum_{\ell=0}^L \sum_{n=-\ell}^{\ell} \frac{2\ell+1}{8\pi^2} Y_n^{\ell}(y) dy \quad (\text{A.8})$$

where we have defined

$$\widetilde{(\kappa \star f)}(y) = \int_{\text{SO}(2)} (\kappa \star f)(y, \alpha) d\alpha. \quad (\text{A.9})$$

From (A.8), we can conclude that (A.1) can be understood as the projected version of $\kappa \star f$, (A.9), integrated against the function $\sum_{\ell=0}^L \sum_{n=-\ell}^{\ell} \frac{2\ell+1}{8\pi^2} Y_n^{\ell}$.

B. Random model generation

In this appendix, we give further details about the procedure used to randomly generate the model architectures we used in our experiments.

We generated 20 random models for each desired parameter range, trained it to convergence, evaluated it on a holdout dataset and picked the best model according to the non-background mIoU.

It should be noted that the goal of the procedure is not to come up with a selection of models that perform optimally on the given task for a certain parameter budget, but rather to select architectures to fairly compare equivariant and non-equivariant models.

B.1. Non-equivariant models

As mentioned in the main text, the non-equivariant models consist of skipped convolutional downsampling and skipped transposed convolutional upsampling layers. Specifically, the downsampling layers are defined by

$$\begin{aligned} \text{down}(n_i, n_o, k, s) &= \text{conv}(n_i, n_o, k, s) \\ &+ \text{conv}(n_i, n_o, 1, 1) \circ \text{maxPool}(k, s), \end{aligned} \quad (\text{B.1})$$

where $\text{conv}(n_i, n_o, k, s)$ is a 2d convolution with n_i input channels, n_o output channels, kernel size k and stride s , \circ denotes function composition and $\text{maxPool}(k, s)$ is a 2d max pooling operation with kernel size k and stride s . The upsampling layers are defined by

$$\begin{aligned} \text{up}(n_i, n_o, k, s) &= \text{tConv}(n_i, n_o, k, s) \\ &+ \text{upsample}(d) \circ \text{conv}(n_i, n_o, 1, 1), \end{aligned} \quad (\text{B.2})$$

where $\text{tConv}(n_i, n_o, k, s)$ is a 2d transpose convolution with n_i input channels, n_o output channels, kernel size k and stride s and $\text{upsample}(d)$ is a 2d upsampling layer to a feature tensor with spatial resolution $d \times d$ and nearest neighbor interpolation. In (B.2), d is chosen such that the spatial dimensions of both summands match. After each down- and upsampling layer, we apply a ReLU nonlinearity.

We randomly select the number of layers and their channels, the kernel sizes and the strides in the following way: First, we select the depth of the network between 1 and $\lfloor \hat{p}/(2 \cdot 10^4) \rfloor$, where \hat{p} is the upper limit of the parameter range. Then, in order to obtain an hourglass-shape for the network, we select the size of the image dimension at the bottleneck between 2 and 30 and linearly interpolate from that to the size of the input (and output) image. These interpolated image sizes are the target dimensions d_i^{target} which we then try to approximate by choosing the kernel sizes and strides of the convolutions appropriately. This is done only for the downsampling layers as the upsampling layers are set to exactly mirror the downsampling architecture.

The kernel sizes k_i are selected at random to be odd integers between 1 and 9. If the image in layer i has smaller size than 9×9 , we take instead the highest odd integer below the image dimension as the upper bound for the random choice. The strides s_i are then computed from the target dimension d_i^{target} , the input dimension d_i^{in} of the layer and the kernel size k_i according to

$$s_i = \text{round}\left(\frac{d_i^{\text{in}} - k_i}{d_i^{\text{target}} - 1}\right). \quad (\text{B.3})$$

If the number generated by (B.3) is 0, we set the stride to 1 instead. The actual output dimensions are then given by

$$d_{i+1}^{\text{in}} = \frac{d_i^{\text{in}} - k_i}{s_i} + 1. \quad (\text{B.4})$$

In order to keep the total number of features roughly constant across layers, we set the number of channels N_i^c to

$$N_i^c = \left\lceil \frac{f}{(d_{i+1}^{\text{in}})^2} \right\rceil, \quad (\text{B.5})$$

where $f = 11 \cdot (2 \cdot L)^2$ is the number of features in the final layer and L is the bandwidth of the input (and output) data.

During the model generation process, architectures are generated according to the procedure summarized above and the total number of parameters for each architecture is computed. If the parameter count lies in the desired range, the model is accepted, otherwise it is rejected and a new model is generated.

All models are trained with batch size 32 and learning rate 10^{-3} using Adam on a segmentation task with one MNIST digit on the sphere and 60k rotated training samples until convergence and then evaluated. In all experiments, we use early stopping on the non-background mIoU metric and a maximum of 100 epochs.

In this way, for each desired parameter range, 20 models were trained and evaluated, we then picked the best performing models (according to the non-background mIoU) and used them for our experiments. The resulting architectures of the three non-equivariant models are summarized in Table 4.

B.2. Equivariant models

The equivariant models consist of the three layers described in Section 3.1. We will denote the operation (3.1) which takes input features on S^2 and returns output features on $\text{SO}(3)$ by

$$\text{S2SO3conv}(n_i, n_o, b_i, b_o, \hat{\beta}), \quad (\text{B.6})$$

where n_i is the number of input channels, n_o is the number of output channels, b_i is the bandwidth of the input feature

Table 4: Non-equivariant model architectures used in our experiments. The up- and down-sampling layers are defined in (B.1) and (B.2), respectively. The columns labeled as “output” contain the channel- and spatial dimensions of the output feature maps.

Model	218k CNN		1M CNN		5.5M CNN	
	block	output	block	output	block	output
Down	input	1×100^2	input	1×100^2	input	1×100^2
	conv(1,13,5,1)	13×96^2	conv(1,12,3,1)	12×98^2	conv(1,12,3,1)	12×98^2
	down(13,15,3,1)	15×94^2	down(12,13,3,1)	13×96^2	down(12,15,5, ,1)	15×94^2
	down(15,22,9,1)	22×86^2	down(13,16,5,1)	16×92^2	down(15,16,3,1)	16×92^2
	down(22,31,7,1)	31×80^2	down(16,77,5,2)	77×92^2	down(16,85,7,2)	85×43^2
	down(31,141,3,2)	141×39^2	down(77,96,3,1)	96×42^2	down(85,191,3,1)	191×39^2
			down(96,163,5,1)	163×38^2	down(191,191,3,1)	191×37^2
					down(191,1100,3,2)	1100×18^2
Up	up(141,31,3,2)	31×80^2	up(163,96,5,1)	96×42^2	up(1100,191,3,2)	191×37^2
	up(31,22,7,1)	22×86^2	up(96,77,3,1)	77×44^2	up(191,141,3,1)	141×39^2
	up(22,15,9,1)	15×94^2	up(77,16,5,2)	16×92^2	up(141,85,5,1)	85×43^2
	up(15,13,3,1)	13×96^2	up(16,13,5,1)	13×96^2	up(85,16,7,2)	16×92^2
	tConv(13,11,5,1)	11×100^2	up(13,12,3,1)	12×98^2	up(16,15,3,1)	15×94^2
			tConv(12,11,3,1)	11×100^2	up(15,12,5,1)	12×98^2
				tConv(12,11,3,1)	11×100^2	
Params.	218,144		1,042,013		5,519,335	

Table 5: Equivariant model architectures used in our experiments. The notation for the layers is introduced in (B.6)–(B.8). Note that the product of the *input* bandwidth and β stays constant throughout the network. The columns labeled as “output” contain the channel- and spatial dimensions of the output feature maps.

Model	204k S2CNN		820k S2CNN	
	block	output	block	output
Down	input	1×100^2	input	1×100^2
	S2SO3conv(1, 11, 42, $0.1238 \cdot \pi$)	11×84^3	S2SO3conv(1, 13, 43, $0.1881 \cdot \pi$)	13×86^3
	SO3conv(11, 12, 35, $0.1474 \cdot \pi$)	12×70^3	SO3conv(13, 16, 36, $0.2187 \cdot \pi$)	16×72^3
	SO3conv(12, 13, 27, $0.1768 \cdot \pi$)	13×54^3	SO3conv(16, 19, 30, $0.2613 \cdot \pi$)	19×60^3
	SO3conv(13, 14, 20, $0.2292 \cdot \pi$)	14×40^3	SO3conv(19, 21, 23, $0.3135 \cdot \pi$)	21×46^3
			SO3conv(21, 24, 16, $0.4089 \cdot \pi$)	24×32^3
			SO3conv(24, 27, 10, $0.5878 \cdot \pi$)	27×20^3
Up	SO3conv(14, 13, 27, $0.3095 \cdot \pi$)	13×54^3	SO3conv(27, 24, 16, $0.9405 \cdot \pi$)	24×32^3
	SO3conv(13, 12, 35, $0.2292 \cdot \pi$)	12×70^3	SO3conv(24, 21, 23, $0.5878 \cdot \pi$)	21×46^3
	SO3conv(12, 11, 42, $0.1768 \cdot \pi$)	11×84^3	SO3conv(21, 19, 30, $0.4089 \cdot \pi$)	19×60^3
	SO3S2conv(11, 11, 50, $0.1474 \cdot \pi$)	11×100^2	SO3conv(19, 16, 36, $0.3135 \cdot \pi$)	16×72^3
			SO3conv(16, 13, 43, $0.2613 \cdot \pi$)	13×86^3
		SO3S2conv(13, 11, 50, $0.2187 \cdot \pi$)	11×100^2	
Params.	204,073		820,184	

map, b_o is the bandwidth of the output feature map and $\hat{\beta}$ is related to the kernel size as described further down. The operation (3.2) which takes and returns feature maps on $\text{SO}(3)$ is similarly denoted by

$$\text{SO3conv}(n_i, n_o, b_i, b_o, \hat{\beta}) \quad (\text{B.7})$$

and the operation (3.3) with input feature maps on $\text{SO}(3)$ and output feature maps on S^2 is denoted by

$$\text{SO3S2conv}(n_i, n_o, b_i, b_o, \hat{\beta}). \quad (\text{B.8})$$

After each layer, we apply a ReLU activation function.

We also experimented with adding the equivariant skip connections provided by the original S2CNN implementation to the layers (B.7) but found no performance gain, so we did not use them in the experiments.

For the spherical models, the procedure to generate architectures is similar to the non-equivariant case, but now the bandlimit replaces the role of the image dimension in the construction.

First, we select the depth and the bottleneck size in the same way as for the non-equivariant models. Then, we select the bandlimit at the bottleneck between 3 and 20 and compute a linear interpolation between the input bandlimit and the bandlimit at the bottleneck. Since the equivariant layers in the S2CNN architecture directly take the input and output bandlimits as parameters, we only have to round those interpolated bandlimits to integers.

On top of the bandlimits, the spherical convolutional layers also depend on the grid on which the kernel is sampled. For this grid, we take a Driscoll-Healy grid (Driscoll & Healy, 1994) around the unit element in $\text{SO}(3)$. The three Euler angles α , β and γ of the grid points lie between 0 and 2π for α and γ but β takes values between 0 and some upper bound $\hat{\beta}$. This upper bound is responsible for the locality of the kernel and is analogous to the kernel size in standard CNN layers. However, since the bandlimit L_i of the feature map determines how fine the grid for the feature map is, the effective kernel size is characterized by the product $\hat{\beta} \cdot L_i$. In order to keep this product fixed throughout the network, we set $\hat{\beta}_i = L/L_i \cdot \hat{\beta}^{\text{ref}}$ where L is the bandlimit of the input and $\hat{\beta}^{\text{ref}}$ is a reference value that we pick at random between 0.02 and 0.25. Note that this construction implies that the number of trainable parameters in the spatial dimensions of the kernel is fixed throughout the network.

For the equivariant models, we randomly select the channel numbers by first sampling a maximum channel number at the bottleneck between 11 and 30 and then linearly interpolating between 11 (the channel number at the output) and this maximum.

Again, as in the non-equivariant case, we generate architectures according to this procedure and reject the ones whose

parameter count lies outside the desired range. The models obtained in this way are trained with batch size 32 and learning rate 10^{-3} using Adam on a segmentation dataset of 60k unrotated training samples containing one MNIST digit on the sphere each until convergence and then evaluated. In all experiments, we use early stopping on the non-background mIoU metric and a maximum of 200 epochs.

As in the non-equivariant case, we generated, trained and evaluated 20 models in the desired parameter ranges and picked the best performing ones according to the non-background mIoU. These models were then used for the experiments. The resulting architectures of the two equivariant models are summarized in Table 5.

C. Details on experiments

In this appendix, we give more details about the experimental setup for the spherical runs described in Section 3.

C.1. Data generation

As discussed in the main text, the input data to our models consist of MNIST digits and FashionMNIST items of clothing which were projected onto the sphere, labeled by their corresponding segmentation masks. Here we give more details on the data generation process.

For an input image with n digits / items of clothing on the sphere, we first sampled n images from MNIST or FashionMNIST and pasted them at a random position on a 60×60 canvas. This canvas is then projected onto the sphere by projecting the spherical grid points onto the plane using a stereographic projection and performing a bilinear interpolation in the plane. To obtain a rotated input sample, we rotate the grid points with a random rotation matrix before projecting them.

We generate the segmentation mask for a single digit / item of clothing by considering all pixels with a grayscale value above a certain threshold as belonging to the target class and to the background class otherwise. These segmentation masks are then assembled into the 60×60 canvas and projected onto the sphere. Instead of a bilinear interpolation, we use nearest-neighbor interpolation for the segmentation masks.

For MNIST, a threshold value of 150 yielded good results. For FashionMNIST however, we use a value of 10 to capture finer details of the cloths, as illustrated in Figure 11. Even lower values lead to a blurring along the edges in the segmentation mask.

For validation, we generated datasets in the same way as for training, but sampling 10,000 data points from the test split of the MNIST dataset.



Figure 11: Examples of segmentation masks for Fashion-MNIST generated from the original (left) with threshold values of 150 (center) and 10 (right).

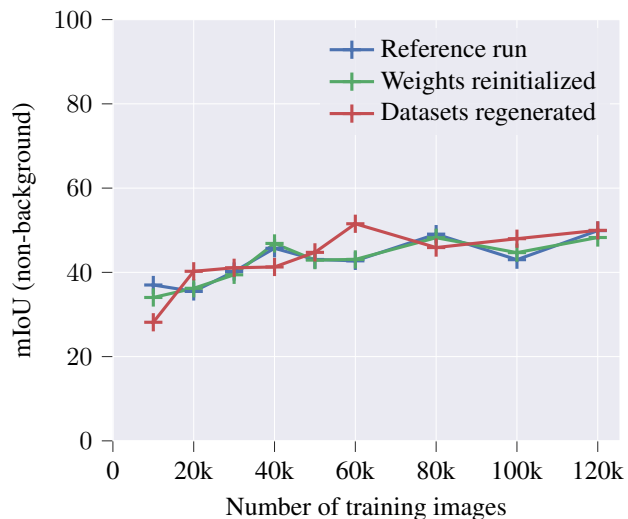


Figure 12: *Variation in Performance*. Performance for the non-equivariant 218k parameters model trained three times on rotated datasets of 2 MNIST digits on the sphere of various sizes.

C.2. Performance variation due to data sampling

As mentioned in the main text, increases or decreases in performance with varying dataset sizes that occur completely in parallel for all models are due to sampling effects during the data generation process. This point is illustrated by the plot depicted in Figure 12, where we trained the non-equivariant model with 218k parameters three times on rotated datasets of various sizes with 2 MNIST digits on the sphere. One run is for reference, one with randomly reinitialized weights before training and one was trained on a randomly regenerated dataset. We can see clearly that the weight reinitialization only had a minor influence on the model performance, but the peculiar irregularities of the performance increase with growing training dataset are within the variation of regenerating the dataset.

C.3. Influence of projection point for non-rotated data

The spherical data we use in our experiments is given in the Driscoll-Healy grid which is an equispaced grid in spherical

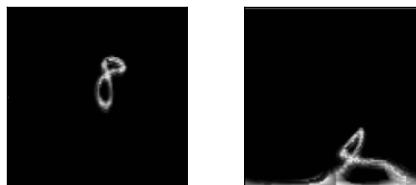


Figure 13: Example of the input data in the Driscoll-Healy grid for a digit projected onto the grid center (left) and onto the pole of the sphere (right).

coordinates, i.e. in the azimuthal angle ϕ and polar angle θ . Therefore, what the input data looks like on this grid depends crucially on the projection point relative to the grid. Figure 13 illustrates this with a digit projected into the center of the ϕ, θ grid (i.e. on the equator, close to the $\phi = \pi$ line) and a digit projected onto the pole.

For the experiments for non-rotated data depicted in Figure 5 of the main text, we projected the input images onto the grid center, to help the non-equivariant networks. For this projection point, slight variations of the digit positions correspond almost to translations in the Driscoll-Healy grid with respect to which the ordinary CNNs are equivariant. For comparison, Figure 14 shows the results of training on data projected onto the pole of the sphere. Note that the performance on unrotated data is reduced considerably as compared to Figure 5, but slightly improved on rotated data. The reason is that slight positional variations across the pole lead to very non-linear deformations of the data represented in the Driscoll-Healy grid and the translational equivariance of the CNNs cannot help the training process. On the other hand, since the polar projections are most challenging, overall performance on rotated test data is improved (although of course still very poor).

Since the S2CNN models are equivariant with respect to all rotations on the sphere, their performance does not depend on the projection point of the digit, cf. Figure 14 vs. Figure 5. Therefore, we trained the S2CNN models for all other experiments on images projected onto the pole.

C.4. Sample predictions

In Figures 15 and 16, we show ten random sample predictions from the best equivariant and non-equivariant models in the four-digit task.

D. Inference latency profiling

Table 6 shows the inference latency per layer for the equivariant semantic segmentation model. Here the bulk of the time is spent in the upsampling layers and in particular the last $SO(3)$ to S^2 convolution takes up almost half of the total inference time. Comparing with Table 5 this also

Equivariance versus Augmentation for Spherical Images

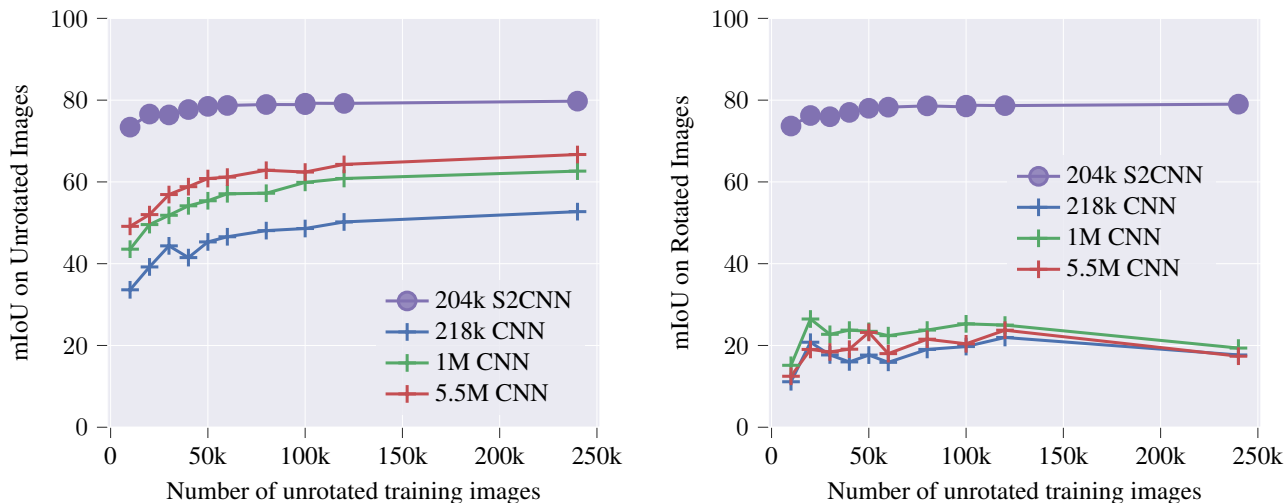


Figure 14: *Polar Training Images*. Performance of equivariant and non-equivariant models for various amounts of data augmentation for models trained on unrotated data, projected on the pole of the sphere. Performance is measured in terms of mIoU for the non-background classes. Left: Evaluated on unrotated test data. Right: evaluated on rotated test data.

Table 6: Latency per layer and fraction of total time for the equivariant semantic segmentation model (204k S2CNN in Appendix Table 5) on an Nvidia T4 16GB GPU as measured by NSight systems with NVTX for batch size 1. Latency measures the time of a forward pass through the layer on the GPU.

	Layer	Latency (ms)	Fraction (%)
Down	S2SO3conv	6.7	5.0
Down	SO3conv	16.1	11.8
Down	SO3conv	7.3	5.3
Down	SO3conv	3.1	2.3
Up	SO3conv	6.8	5.0
Up	SO3conv	15.3	11.2
Up	SO3conv	26.4	19.4
Up	SO3S2conv	54.5	40.0

coincides with the largest $SO(3)$ tensors being processed.

In Table 7 the fraction of inference time is instead shown per operation. Here it is clear that a large portion is spent in the custom CUDA implementation for the complex matrix multiplication of $SO3$ tensors, a possible avenue for optimization reducing overall inference latency. Most of the remaining time is spent in the Fast Fourier Transform ($SO3_fft_real$) and the inverse FFT ($SO3_ifft_real$).

Table 7: Fraction of time spent per operation ($> 1\%$) for the equivariant semantic segmentation model (204k S2CNN in Appendix Table 5) on an Nvidia T4 16GB GPU as measured by NSight systems for batch size 1.

Module	Op	Fraction of time (%)
s2cnn	<code>_cuda_SO3_mm</code>	45.2
s2cnn	<code>SO3_ifft_real</code>	15.6
s2cnn	<code>SO3_fft_real</code>	14.4
pytorch	<code>einsum</code>	12.2
pytorch	<code>contiguous</code>	8.4
		95.7

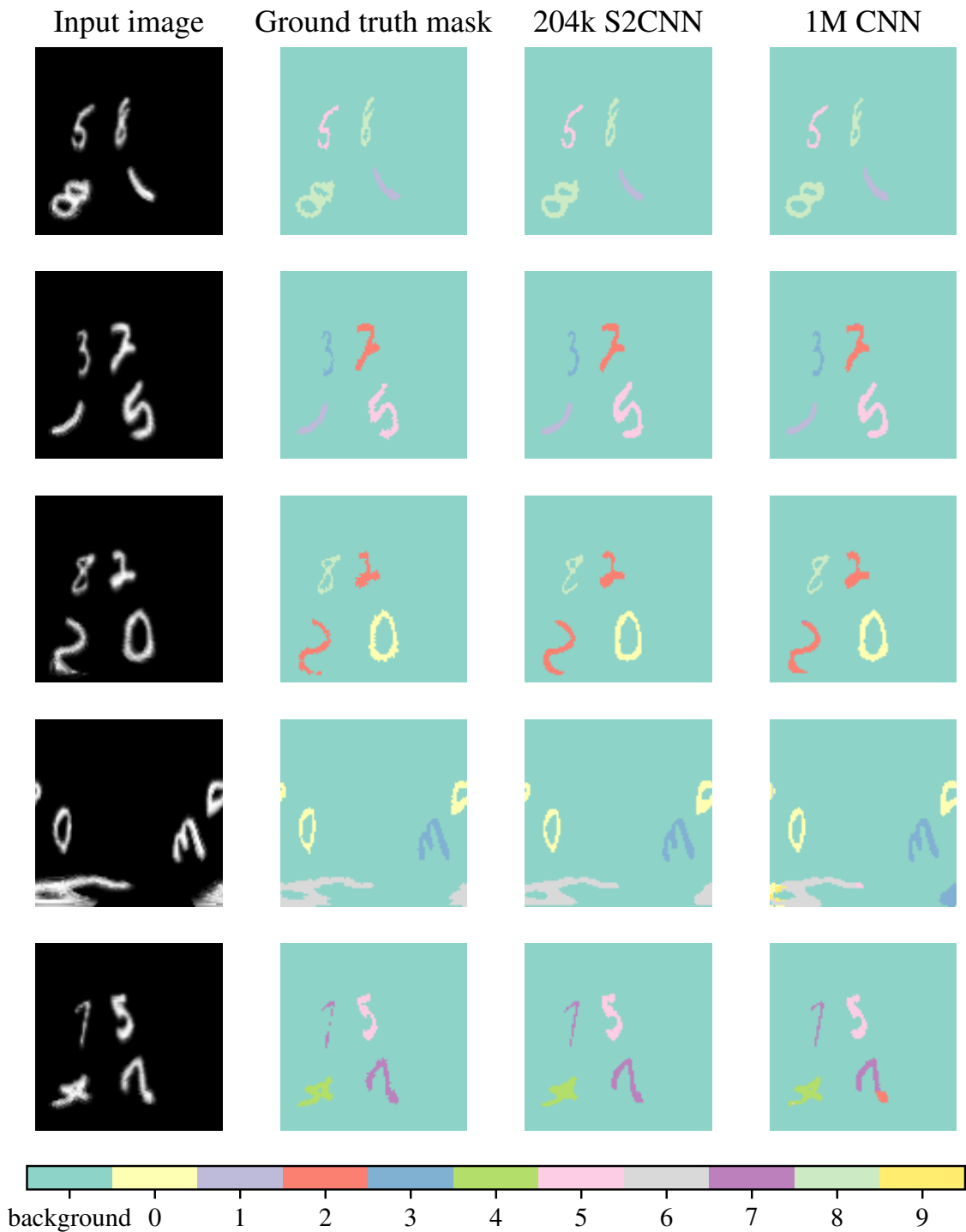


Figure 15: Sample predictions on the test dataset for the best equivariant model (240k S2CNN, trained on 240k samples) and non-equivariant model (1M CNN, trained on 600k samples) on the four-digits task depicted in Figure 6. The five samples were selected at random from the dataset and we depict here the raw data on the Driscoll–Healy grid. Five more samples are depicted in Figure 16.



Figure 16: Five more samples of model predictions, cf. Figure 15.