IMPLICIT EQUIVARIANCE IN CONVOLUTIONAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional Neural Networks (CNN) are inherently equivariant under translations, however, they do not have an equivalent embedded mechanism to handle other transformations such as rotations and change in scale. Several approaches exist that make CNNs equivariant under other transformation groups by design. Among these, steerable CNNs have been especially effective. However, these approaches require redesigning standard networks with filters mapped from combinations of predefined basis involving complex analytical functions. We experimentally demonstrate that these restrictions in the choice of basis can lead to model weights that are sub-optimal for the primary deep learning task (*e.g.* classification). Moreover, such hard-baked explicit formulations make it difficult to design composite networks comprising heterogeneous feature groups. To circumvent such issues, we propose Implicitly Equivariant Networks (IEN) which induce equivariance in the different layers of a standard CNN model by optimizing a multi-objective loss function that combines the primary loss with an equivariance loss term. Through experiments with VGG and ResNet models on Rot-MNIST , Rot-TinyImageNet, Scale-MNIST and STL-10 datasets, we show that IEN, even with its simple formulation, performs better than steerable networks. Also, IEN facilitates construction of heterogeneous filter groups allowing reduction in number of channels in CNNs by a factor of over 30% while maintaining performance on par with baselines. The efficacy of IEN is further validated on the hard problem of visual object tracking. We show that IEN outperforms the state-of-the-art rotation equivariant tracking method while providing faster inference speed.

1 INTRODUCTION

Over the last decade, state-of-the-art deep learning algorithms have continued to push the accuracy on computer vision tasks such as classification on challenging datasets (Deng et al., 2009). The translation equivariance property inherent to CNNs has been instrumental in enabling this performance. However, the vulnerability of CNNs to transformations such as rotation and scaling remains a challenge (Cohen & Welling, 2016; Weiler et al., 2018b; Sosnovik et al., 2020b), since standard CNNs do not learn features that are equivariant with respect to these transformations.

One way to circumvent the above issue is to design networks invariant to such transformations. However, for deep learning problems, intermediate layers should not be invariant so that the relative pose of local features is preserved for the later layers (Cohen & Welling, 2016; Hinton et al., 2011). Therefore, several approaches have been proposed to design networks that are *equivariant* to certain transformation groups. A network is defined to be equivariant if the output that it produces transforms in a predictable way on transformations of the input. Among the various methods that exist for constructing equivariant representations, steerable CNNs are particularly effective since variants of this approach have produced SOTA results for transformation groups of rotation, reflection and scale change on deep learning tasks of classification (Cohen & Welling, 2016; 2017), segmentation (Weiler et al., 2018b), object tracking (Sosnovik et al., 2020a; Gupta et al., 2021), among others.

Steerable CNNs estimate filters for different orientations of a transformation group without the need for tensor resizing. The resultant architecture shares weights over the filter orientations, thereby improving generalization and reducing sample complexity. These methods essentially rely on the construction of filters from linear combinations of certain sets of basis functions. The choice of

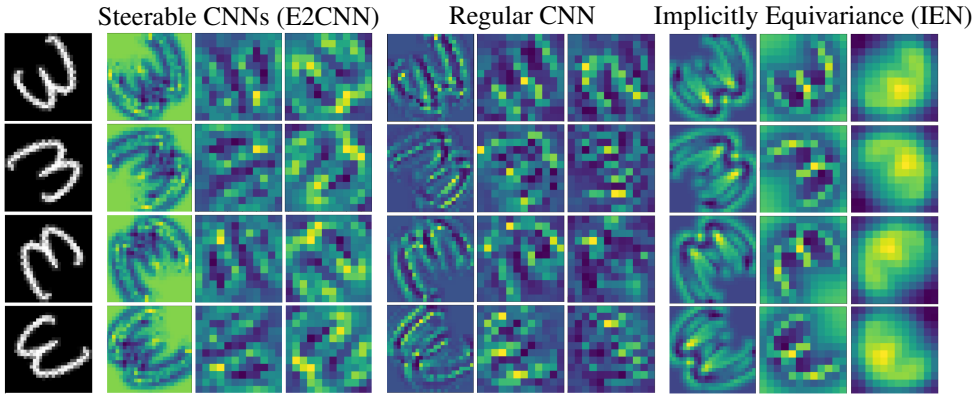


Figure 1: Feature maps obtained after first, third and fourth convolutions for 4 rotated versions of an MNIST digit with steerable networks (E2CNN), CNN and Implicitly Equivariant Network (IEN).

the basis can vary depending on the transformation group for which equivariance is to be induced. Examples include the use of circular harmonic functions (Worrall et al., 2017; Weiler et al., 2018b) and Hermite polynomials (Sosnovik et al., 2020b) for rotation- and scale-equivariance, respectively.

Interestingly, while the basis can change across different constructions of steerable CNNs, the trained models are mostly similar to regular CNNs at inference time except that the filter weights differ. Based on this observation, we hypothesize that it should be theoretically possible to achieve absolute equivariance with regular CNN models as well - we need to find a model training procedure that biases the weights to converge towards portions of the optimization landscape that induce equivariance. However, as no simple mechanism exists that can force regular CNNs to learn equivariant representations, steerable CNNs, given their SOTA performance for rotation- and scale-equivariance on popular deep learning tasks (Weiler & Cesa, 2019; Sosnovik et al., 2020b; Gupta et al., 2021), have thus far been preferred.

It is important to note that while inducing equivariance in the network is a desirable property for certain deep learning problems, the primary goal is still to maximize its performance on tasks such as classification, regression and segmentation. Given this goal, we posit that by restricting the weight parameterization space so that it can only be represented by combinations of certain basis sets, we are overly constraining the model. While such a strategy makes the CNN model equivariant to the desired transformation group, the performance on the actual learning task may be sub-optimal. Later in this paper, we validate this argument through numerical experiments. We show that for certain scenarios, regular CNN models outperform steerable CNNs given the same number of channels. Equivariance is only meant to help the model generalize better on a task in a data efficient manner, and we propose that the extent of equivariance required in a model could alternatively be discovered through the training process. This further implies that equivariance to a certain transformation group should be treated as an additional objective for which the CNN model should be trained for.

Another limitation of steerable CNNs is that not only do they require redesigning standard networks by incorporating sets of complex basis functions, but also, these basis functions tend to vary for different transformation groups (Weiler et al., 2018b; Weiler & Cesa, 2019; Sosnovik et al., 2020b). Significant effort would therefore be required to convert existing deep learning architectures into their group-equivariant counterparts. Furthermore, with such formulations, it is hard to design composite networks that are equivariant to multiple transformation groups *e.g.*, rotations, reflections and change in scale. These networks also mostly work with homogeneous filter groups, where each filter group has the same group size. We hypothesize and later validate that not all filters require higher order discretizations. Therefore, it should be possible to achieve desired performance on given tasks with heterogeneous filter groups. An example would be to design convolutional layers with composite rotation groups of size 1, 2, 4 and 8 combined together. With reduced number of total channels, the inference speed of the model can be boosted.

In this paper, we propose to achieve equivariance under a certain transformation group by adding constraints to the model training procedure. A better intuition of the idea can be obtained from

Figure 1. For 4 different orientations of an MNIST digit, we show example feature maps for the first, third and fourth convolutional layers of E2CNN (Weiler et al., 2018b; Weiler & Cesa, 2019), a regular CNN model and our Implicitly Equivariant Network (IEN). While the regular CNN fails to preserve equivariance, we see that E2CNN as well as IEN maintain equivariance under rotations. In terms of mean-squared error, the extent of equivariance exhibited by E2CNN and IEN is on the order of 10^{-12} and 10^{-6}, respectively. We observe that while IEN is less equivariant to rotations, it performs approximately at par with E2CNN in terms of classification accuracy on the Rot-MNIST dataset. We show later that for more complex datasets, IEN surpasses performance of E2CNN even with a lower level of equivariance. Based on the above observations, we formulate *the implicit equivariance hypothesis*.

The Implicit Equivariance Hypothesis. *For a group-equivariant network where equivariance to a certain transformation group is hard-wired in the architecture by restricting the choice of filters to a certain subset (e.g., circular harmonics for rotation group), the performance of the trained model is not always optimal and standard CNN architectures when trained with additional constraints may be able to achieve better accuracy at same or lower inference cost than the former.*

Based on the hypothesis above, we transform the regular CNN model into a multi-objective formulation where an additional equivariance loss term is optimized together with the primary loss component of the model. We refer to this approach as Implicitly Equivariant Network (IEN) in the rest of the paper and a more formal problem description is provided in Section 4. IEN makes existing CNN models robust to desired transformations with minimal modifications to the network.

Contributions.

- We demonstrate that by implicitly constraining regular CNNs to learn equivariance during training, it is possible to achieve better performance on deep learning tasks compared to steerable CNNs.
- IEN architectures perform at par with steerable CNNs while reducing the number of channels per layer by more than 30%. This reduction happens because IEN facilitates heterogeneous combinations of transformation groups of different orders.
- In contrast to the difficulty of designing steerable CNNs that are equivariant under multiple transformation groups, we show that it is relatively simple to design IEN architectures that are equivariant under multiple transformations groups.
- We show that IEN achieves performance at par with steerable networks while providing better inference speed over the latter by applying it on the problem of visual object tracking.
- Our Implicitly Equivariant Network (IEN) formulation treats equivariance to a transformation group as an additional objective to be optimized. This ensures that absolute equivariance is not explicitly hardwired in the model. Rather, it is induced only to the extent that model performance on the primary task is maximized.

Implications. With the concept of implicit equivariance outlined above, we hope to have paved the groundwork for easy and efficient implementation of equivariance in existing CNN architectures. We also hope that this research promotes further development on the following aspects.

Learning equivariance for black-box transformations. With the simplicity of IEN, future CNN architectures can be made robust against transformations for which the choice of analytical basis functions is not straightforward (*e.g.*, occlusion).

Designing highly robust models for open-world scenarios. Since IEN can combine equivariance to multiple transformation groups easily, advanced CNN architectures can be designed that are equivariant to a large set of transformations in a unified manner, thus making them more robust with respect to open-world problems.

Easy integration with existing models. Since IEN requires minimal modifications to existing CNN architectures it can be easily integrated with existing CNN models.

Improved inference speed. Due to weight sharing, existing steerable CNNs use relatively larger number of channels per layer. This limits their use in problems where inference speed is crucial. Our heterogeneous IEN formulations alleviate this issue, thereby increasing their scope of applicability.

2 RELATED WORK

A commonly utilized technique to tackle variations of orientations and scale in computer vision algorithms is data augmentation. This involves supplying transformed copies of the input data to the model which primarily helps in learning invariance. However, this strategy can learn equivariance as well to a limited extent (Krizhevsky et al., 2017; Lenc & Vedaldi, 2015; Laptev et al., 2016). Methods such as equivariant Boltzmann machines (Sohn & Lee, 2012) and equivariant descriptors (Schmidt & Roth, 2012) have been proposed to construct learning algorithms that give rise to equivariant representations. A different set of algorithms focus on inducing equivariance in the model by design. Collectively referred as group-equivariant networks, these involve reparameterization and obtain the filter weights by employing a set of atomic bases or by projecting to a different space.

A large body of work exists on designing group-equivariant convolutions. These include discrete roto-translations in 2D (Cohen & Welling, 2016; Weiler et al., 2018b; Bekkers et al., 2018a) and (Worrall & Brostow, 2018), continuous rot-translations in 2D (Worrall et al., 2017) and 3D (Weiler et al., 2018a; Bekkers et al., 2018b), in-plane reflections (Weiler et al., 2018a; Weiler & Cesa, 2019), continuous rotations on the sphere (Esteves et al., 2018), and equivariance to discrete scales (Sosnovik et al., 2020b), among others. The GCNN approach (Cohen & Welling, 2016) effectively learns equivariant representations but scales linearly in the number of orientations which prohibits its applicability on large-scale problems. This issue is overcome by steerable CNNs that generate equivariant representation using composition of elementary features or atomic filters (Cohen & Welling, 2017). Subsequent improvements on these architectures have also been made recently (Weiler et al., 2018b; Weiler & Cesa, 2019). Among these, steerable CNNs have been most effective (Weiler & Cesa, 2019). The idea of using steerable bases is not new and was proposed in earlier research (Freeman & Adelson, 1991; Teo & Hel-Or, 1997). Our idea of IEN is inspired by the design of steerable CNNs presented in Weiler et al. (2018b).

Recently, an approach to learn the rotational versions of the filter through an additional equivariance loss has been presented (Diaconu & Worrall, 2019). However, it focused on learning to transform the filters without compromising the equivariance property of the model. This helps to tackle the adverse effects of *post hoc* discretization suffered by steerable CNNs (Weiler et al., 2018b). In contrast to this, our work focuses on learning the right balance between making the model equivariant and optimizing its performance on the primary deep learning task. A few approaches exist that use regularization to maintain consistency at the final output (Sajjadi et al., 2016; Bethelot et al., 2019), however, unlike IEN, these are limited to achieving invariance. The approach of Benton et al. (2020) is closest to IEN since this strategy enforces equivariance at the end of the network through data augmentation. However, this approach focuses on learning the distribution of input and does not include an additional loss term to induce equivariance in the network, rather it relies on a combination of forward and inverse transforms of the input data and the output to achieve equivariance.

3 EQUIVARIANCE IN CNNS

Equivariance refers to the property of a function to commute with the actions of any transformation group G acting on its input as well as output. Mathematically, any function $f : X \rightarrow Y$ is equivariant to the transformation group G, if

$$f(\varphi_g^X(x)) = \varphi_g^Y(f(x)) \qquad \forall g \in G, x \in X, \tag{1}$$

where φ_g^X and φ_g^Y denote group actions in the respective spaces. Note that for φ_g^Y =id, equivariance simplifies to the special case of invariance.

CNN layers, involving transformation of feature maps h through convolution with filters Ψ are by architecture design equivariant under translations, *i.e.*, $(\mathcal{T}_d h) * \Psi = \mathcal{T}_d(h * \Psi)$ where $\mathcal{T}_d$ is an action of the translation group $T \in (\mathbb{R}, +)$ that shifts the input by $d \in \mathbb{R}^2$. Apart from translation, there are often other transformations that occur in images, such as rotations, reflections and dilations. However, CNNs are not equivariant to such transformations by design. This limits the generalization of the model as patterns learnt for one orientation are not applicable to other orientations. To make deep learning models robust with respect to these transformations, equivariant models are used.

Steerable CNNs. Steerable CNNs extend the notion of weight sharing to transformation groups beyond translations, thus making CNNs jointly equivariant to translations and other transformations.

In the context of rotations, this implies performing convolutions with rotated versions of each filter. This weight sharing over rotations leads to rotational equivariance in a manner analogous to how translational weight sharing in standard CNNs gives rise to translational equivariance.

Steerable filters can be expressed as a linear combination of a fixed set of atomic basis functions $\{\psi_q\}_{q=1}^{Q}$ such that the filter can be steered directly based on transformation of the input. Since we present our discussions in this paper primarily in the context of rotations, we briefly discuss the formulation of steerable filters for rotations. For this case, Gaussian radial bases can be used to create rotationally steerable filters for an arbitrary angle θ. Such a rotationally steerable filter $\Psi : \mathbb{R}^2 \rightarrow \mathbb{R}$ satisfies the following property for all angles $\theta \in (-\pi, \pi]$ and for angular expansion coefficient functions κ_q,

$$\rho_\theta \Psi(x) = \sum_{q=1}^{Q} \kappa_q(\theta)\psi_q(x) \tag{2}$$

where ρ_θ is a rotation operator. With the above steerable filter formulation, the response of different orientations can then be conveniently expressed in terms of the atomic basis functions,

$$(f * \rho_\theta \Psi)(x) = \sum_{q=1}^{Q} \kappa_q(\theta)(f * \psi_q)(x). \tag{3}$$

4 Implicit Equivariance

General description. *Implicit equivariance* refers to the process of inducing equivariance in the model during training by optimizing it with respect to an additional loss term. We add an equivariance loss term to the objective function that promotes exploiting parts of the weight space for which the network has relatively improved equivariance under desired transformations. Unlike other equivariance approaches (Weiler et al., 2018b; Weiler & Cesa, 2019; Sosnovik et al., 2020b), the model is not forced to learn absolute equivariance. Rather, the implicit equivariance formulation achieves equivariance only to the extent that model performance on the primary task, *e.g.* classification, is maximized.

Figure 2 shows a schematic representation of the implicit equivariance concept. The network takes an input image and its transformed versions and computes the primary loss $\mathcal{L}_\tau$, *e.g.* loss corresponding to classification error, for each of these inputs. Additional loss terms $\mathcal{L}_G$ are computed from different parts of the network which are combined together to form the equivariance loss. We present the discussion primarily for rotation equivariance, however, IEN can be formulated for other transformations as well. We show this through using IEN for scale equivariance as well.

To simplify the implementation of the equivariance loss, we introduce the concept of a feature group. Each feature group corresponds to a set of feature maps that undergo a pooling operation, referred to as *group-pooling* or $\mathcal{G}(\cdot)$, that takes pixel-wise maximum values across all the feature maps within the feature group to compute a pooled response. The equivariance loss computed at the i^{th} layer corresponds to the pixel-wise error between the aligned versions of feature groups of the input and feature groups of the transformed versions. Therefore, the network utilizes the presence of multiple feature maps within each group to learn the desired equivariance. The max-pooling operation can introduce a small amount of invariance, but we use it as computing the equivariance loss over pooled feature maps is easier to implement than using the original feature maps directly.

Mathematical formulation. We first provide a general mathematical formulation for the training of a standard CNN model for a certain deep learning task τ. We define the CNN mapping as $\mathcal{F}$ with weights $\mathbf{W}$ and a set of hidden channels $\mathbf{H}$. Further let $\mathbf{h}_i \in \mathbf{H}$ denote the set of channels at the i^{th} layer of the network. Let $\mathcal{L}_\tau$ denote the loss function used to compute the performance for task τ during training. Based on this description, the mathematical statement describing the optimization of the CNN model is

$$\min_{\mathbf{W}} \mathcal{L}_\tau(\mathcal{F}(\mathbf{H}(\mathbf{W}); \mathbf{x}), \mathbf{y}), \tag{4}$$

where $\{\mathbf{x}, \mathbf{y}\} \in \mathcal{D}$ are data samples used to train the network.

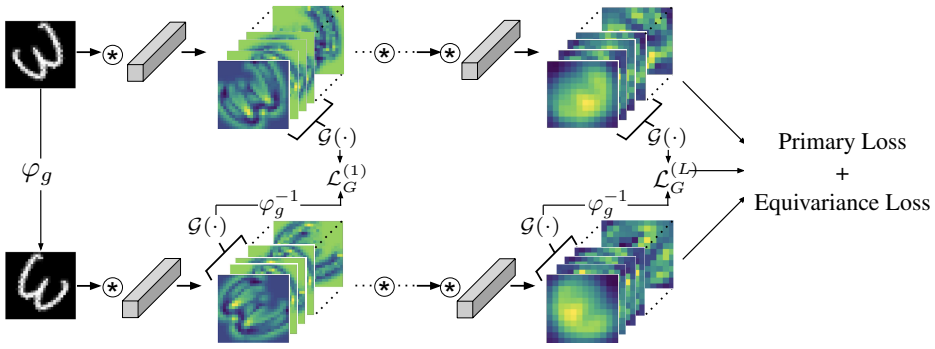


Figure 2: Schematic representation showing the working of Implicitly Equivariant Networks. Here φ_g, $\varphi_{g^{-1}}$ and $\mathcal{G}(\cdot)$ denote rotation, rotation in opposite direction and group-pooling, respectively.

Typically, group-equivariant models (Weiler et al., 2018b; Sosnovik et al., 2020b) induce equivariance by computing the feature maps for every hidden layer of the network for Λ different orientations. Intuitively, the Λ orientations attempt to approximate the continuous transformation field, and the extent of equivariance improves with larger values of Λ. Inspired by their idea of filter groups, we split the feature maps $\mathbf{h}_i$ of the i^{th} layer of $\mathcal{F}$ into groups of Λ. Based on this notion, we have $n(\mathbf{h}_i)/\Lambda$ feature groups for the i^{th} layer of the network. Note that unlike the group-equivariant networks which share weights across sets of Λ filters, no weight sharing is involved in IEN.

Intermediate group-pooling. In IEN, feature maps within every feature group are trained to collectively learn information for Λ orientations. To extract the final outcome from the convolutional layers, a group-pooling layer is employed that computes pixel-wise maximum value across all the orientations to construct the output feature map. In addition to using group-pooling after the last convolutional layer of $\mathcal{F}$, IEN employs additional group-pooling at intermediate layers of the network where equivariance loss is computed. These operations are performed on the hidden feature groups, $\mathbf{h}_{ij}$, where i and j are used as indices for the hidden layer of the network and the feature group, respectively. We add a group pooling operation $\mathcal{G}(\cdot)$ on every feature group, such that any resultant feature map $\hat{h}_i \in \hat{\mathbf{h}}_i$ is obtained by pooling over the Λ orientations $\mathbf{h}_{ij}$, denoted as $\hat{\mathbf{h}}_i = \mathcal{G}(\mathbf{h}_{ij})$.

Figure 2 shows how feature maps pooled over the orientations are used to compute equivariance loss from different parts of the network. Note that while the equivariance loss in the intermediate layers is computed using $\hat{\mathbf{h}}_i$, the input to the next layer $\mathbf{h}_{ij}$ contains all the original feature maps. This means that the next layers are not impacted by this max-pooling operation and our intermediate group-pooling operations *do not detrimentally affect* the equivariance in later parts of the network.

Equivariance loss. This loss term provides a measure of non-equivariance that exists in a model for any group action φ_g from the transformations G. We define our equivariance loss measure across the network based on the absolute equivariance condition stated in Eq. 1 as (see Appendix A)

$$\mathcal{L}_G(\mathbf{x}) = \sum_{i=1}^{L} \sum_{g \in G} \|\hat{\mathbf{h}}_i(x_p) - \varphi_g^Y(\hat{\mathbf{h}}_i(x_q))\|^2 \cdot \beta_i \quad \forall \ \{x_p, x_q\} \in \mathbf{x} \ \text{ s.t. } \ x_p = \varphi_g^X(x_q). \tag{5}$$

The weighting term β_i provides control over the amount of equivariance imposed in different parts of the network. For layers where equivariance is to be implicitly enforced, $\beta_i > 0$, else $\beta_i = 0$.

Multi-objective optimization. Our IEN architectures are trained for multi-objective loss that combines primary loss $\mathcal{L}_\tau$ with the equivariance loss term. The IEN optimization problem is then stated as

$$\min_{\mathbf{W}} \mathcal{L}_\tau(\mathcal{F}(\mathbf{H}(\mathbf{W}); \mathbf{x}), \mathbf{y}) + \sum_{G \in \mathcal{S}} \alpha_G \mathcal{L}_G(\mathbf{x}), \tag{6}$$

where $\mathcal{S}$ denotes the set of transformations for which equivariance is to be learnt and $\alpha_{\mathcal{S}}$ is the corresponding weighting term.

Heterogeneous IEN. An advantage of IEN is that filter groups of different sizes can be combined together. For N filter groups in steerable CNNs (Weiler et al., 2018b; Weiler & Cesa, 2019) as well

as the N feature groups in the IEN implementation above, there would be $N \cdot \Lambda$ feature maps in the respective layer of the network. This number scales linearly in N and Λ. Het-IEN relies on the assumption that not all features need cyclic groups of size Λ. For example, for $\Lambda = 8$, simpler features could be fully represented with lower order cyclic groups as well, for example 4, 2, or even 1. We refer to such combinations as heterogeneous feature groups. For example, by expressing N feature groups of $\Lambda = 8$ with $0.5N$ groups of 8 orientations, $0.25N$ groups of 4 orientations, $0.125N$ groups of 2 orientations and $0.125N$ groups of 1 orientation, the number of feature maps in the respective layer can be reduced by 33%, which results in significant boost in inference speed.

5 EXPERIMENTS

To demonstrate the efficacy of IEN, we conduct three sets of experiments and discuss related insights. Note that the goal of the experiments is not to improve over the state-of-the-art, rather to demonstrate that simple implicit formulations can also learn equivariance to the extent required for performing at least at par with conventional equivariant methods. First we evaluate the effectiveness of IEN for the task of classification. We focus on the transformation groups of rotations, reflections and scale, however, IEN can be applied to other transformations as well with minimal modifications. We further study the performance of Heterogeneous IEN and investigate if it can reduce inference cost without compromising on model performance. Finally, to evaluate IEN on more complex problems, we analyze its performance on the hard problem of visual object tracking in terms of inference speed and accuracy. Details related to all experiments including description of data and models, training procedure and associated hyperparameters is provided in the supplementary material.

Baselines. Since we aim to identify the optimal amount of equivariance required to maximize the performance of the model on the primary task, we compare different models in terms of performance on the classification task as well as in terms of success rate and precision for tracking. We compare our IEN models with steerable CNNs which induce equivariance in the network by design. We employ the best performing variant of steerable CNNs based on the survey presented in (Weiler & Cesa, 2019), and refer to this method itself as E2CNN. We perform all comparisons at the same inference budgets. As a secondary baseline, we also study the performance of regular CNN models of similar inference budgets trained with data augmentation. For scale variation, We compare the results with three baseline models, the SS-CNN (Ghosh & Gupta, 2019) and the two recent implementations of steerable networks, namely Deep Scale Space (DSS) (Worrall & Welling, 2019) and scale equivariant steerable networks (SESN) (Sosnovik et al., 2020b). For the tracking problem, our baseline is RE-SiamFCv2 (Gupta et al., 2021) (referred as RE-SiamFC), a rotation-equivariant variant of SiamFC tracker (Bertinetto et al., 2016).

Learning equivariance under rotations and reflections. We conduct experiments on Rot-MNIST (Worrall et al., 2017), Rot-TIM (Rotation versions of TinyImageNet (TIM) dataset) and R2-TIM datasets (Rotation+Reflection versions of TinyImageNet (TIM) dataset).

Rot-MNIST classification. This dataset comprises 12000 and 50000 MNIST digits in the train and validation sets, respectively, and each digit is rotated by a random angle between 0 and 360 degrees. Additional details are provided in the supplementary material. For baseline, we implemented E2CNN with rotation groups comprising 4 equidistant orientations, thus $\Lambda = 4$. Similarly, for IEN, we choose feature groups of size 4 in each layer of the network. Qualitative results related to this experiment are shown in Figure 1 where we demonstrated that IEN learns the desired equivariance.

We implemented IEN with the same number of channels per layer as the E2CNN architecture. While our E2CNN implementation obtained an accuracy of 98.8, IEN achieved 98.6% which is almost at par. Interestingly, the regular CNN model with channels equivalent to E2CNN and IEN, performed quite well on this dataset, with only a marginal drop in accuracy (98.4%). This implies that even though CNN4-aug does not exhibit equivariance (as shown in Figure 1), it is still sufficiently good for Rot-MNIST when trained with data augmentation. Nevertheless, Rot-MNIST is a relatively easy dataset, and we believe that scores on this dataset should not be used as a measure to rank the models. We also briefly experimented with implementation of heterogeneous IEN models for this problem. We replaced 25% of the feature groups of size 4 with 2 and 25% with 1, thereby reducing the number of feature maps in every layer by 31%. Even with this significance reduction in model size, no drop in model performance is observed. Details are provided in the supplementary material.

Table 1: Performance scores for ResNet18 and its equivariant versions on Rotation (Rot-TIM) and Rotation+Reflection (R2-TIM) versions of TinyImageNet (TIM) dataset. Here, CNN4-aug and CNN8-aug denote regular CNN models similar in architecture to those of R4 and R8, respectively, at inference phase.

Data-type	Model	Eq-type	Acc. %
Rot-TIM	CNN	-	42.5
	CNN4-aug	-	56.7
	E2CNN	R4	53.5
	IEN	R4	**56.9**
	CNN8-aug	-	58.5
	E2CNN	R8	56.5
	IEN	R8	**59.7**
R2-TIM	CNN	-	43.2
	CNN-aug	-	**56.3**
	E2CNN	R4R	55.9
	IEN	R4R	56.1

Table 2: Performance scores for VGG and its equivariant versions on Rotation (Rot-TIM) and Rotation+Reflection (R2-TIM) versions of TinyImageNet (TIM) dataset. Here, CNN4-aug and CNN8-aug denote regular CNN models similar in architecture to those of R4 and R8, respectively, at inference phase.

Data-type	Model	Eq-type	Acc. %
Rot-TIM	CNN	-	32.1
	CNN4-aug	-	45.3
	E2CNN	R4	46.7
	IEN	R4	**47.4**
	CNN8-aug	-	51.5
	E2CNN	R8	50.2
	IEN	R8	**51.6**
R2-TIM	CNN	-	32.5
	CNN-aug	-	50.0
	E2CNN	R4R	**51.1**
	IEN	R4R	49.9

Rot-TIM classification. Rot-TIM is the rotation variant of TinyImagenet (TIM) comprising 100 classes. Since Imagenet is considered a challenging dataset, we believe that Rot-TIM is a suitable choice for analyzing equivariance of models under rotations. Tables 1 and 2 present results obtained with variants of ResNet18 and VGG models, respectively. To design the various models, we first create E2CNN variants with the same number of parameters as standard VGG and ResNet18 architectures and then build regular CNN and IEN models that match the number of channels per layer of E2CNN. For VGG, equivariance loss is computed after every layer. For ResNet18 variants, we compute equivariance loss after every block. Within the block, $\beta_i = 0$. We experiment with values of 0.01, 0.1 and 1.0 for β_i in the IEN objective function and report the scores for best configurations here. Details related to the datasets and architectures follow in the supplementary material.

For almost all cases, IEN outperforms E2CNN for the same inference cost. For the ResNet18 experiment with R8 elements, IEN improves over E2CNN by a margin of more than 3% in accuracy. We also report scores with CNN models (same in size as E2CNN inference model) with and without data augmentation. Interestingly with the scaled up size, the regular CNN models perform fairly well. With data augmentation, the regular CNN models often outperform E2CNN. This typically happens when the equivariance error in the last layer of such networks is similar to that of IEN. The level of equivariance at the last convolutional layer is the determining factor for model performance, and regular CNN models are often able to steer it accordingly (see the supplementary material for details). This implies that for certain cases, even without imposing the equivariance constraint, performance on the primary task can be maximized. However, since this cannot be known beforehand, implicit equivariance formulation should be a preferred choice.

R2-TIM classification. This dataset is an extended version of Rot-TIM with random flips optionally applied on every sample of the dataset. IEN and CNN with augmentations outperform E2CNN on ResNet, see Table 1. The score of CNN is slightly higher than IEN, which implies that imposing equivariance does not help in this case. An exception to the performance of IEN is the VGG experiment with R2-TIM dataset where E2CNN achieves the highest score (Table 2). With improved training procedure, we hope that IEN should match E2CNN scores for this experiment as well.

Equivariance with heterogeneous filter groups. We explore here the potential of IEN in the context of improving inference speed. We employ heterogeneous groups in the model (denoted as Het-IEN) and evaluate the performance (results shown in Table 4). Our results show that Het-IEN implementations mostly outperform the baseline E2CNN scores while reducing the number of channels per layer of the network by more than 30%. This translates into reduction of the filter blocks between every two layers by approximately a factor of 2, thereby significantly improving the inference speed. Detailed results are presented in Table 4 of Appendix C.1.

Learning equivariance under change of scale. For the generalizablity of IEN across different transformations, we also investigated its application for handling scale variations. We conducted experiments on Scale-MNIST and STL-10 datasets, and the results are described in Table 10 of Section C.4. On Scale-MNIST with images of size 28×28 pixels, SS-CNN and SESN obtained error scores of 1.95 and 1.76, respectively. In the similar setting, IEN obtained an average error score of 1.78 over three runs, performing at par with IEN. For images of size 56×56 , IEN achieves an error score of 1.33 compared to 1.42 and 1.57 obtained by SESN and DSS, respectively. This shows that IEN achieves performance similar to or better than the implicit formulation SESN, which, in this case is the equivalent explicit formulation of steerable networks for scale equivariance. We implement IEN similar to the vector formulation of SESN described in Sosnovik et al. (2020b). For STL-10 dataset, IEN obtains a mean classification error of 10.11% over three runs compared to 10.83% and 11.28% obtained for SESN and DSS, respectively. These results clearly demonstrate that the implicit learning of equivariance in IEN leads to improved classification performance even under variations of scale.

Rotation Equivariance in Object Tracking. Table 3 presents the results obtained with various models on Rot-OTB dataset (Gupta et al., 2021). All tracking algorithms presented here are variants of SiamFC (Bertinetto et al., 2016), a popular algorithm for object tracking problems. As evaluation metrics, we use the commonly employed area under the curve scores for success and precision values (Wu et al., 2013). As shown already in (Gupta et al., 2021), making SiamFC equivariant to rotations makes the tracker more robust to this transformation, see scores for RE-SiamFC. Details related to setup of tracking experiments are in the supplementary material.

Table 3: Results for Object Tracking. Scores are presented for IEN implementations and two baselines: SiamFC (Bertinetto et al., 2016) and RE-SiamFC (Gupta et al., 2021)

Model	Channels%	Succ.	Pr.
SiamFC	-	0.29	0.47
CNN4-SiamFC	100	0.33	0.56
RE-SiamFC	100	0.35	0.62
IEN-SiamFC	100	0.37	0.63
Het-SiamFC	69	0.37	0.62

Table 3 shows the results obtained with the IEN implementation of SiamFC (referred as IEN-SiamFC). We show that IEN-SiamFC slightly outperforms RE-SiamFC for this task as well. Interestingly, IEN required very minimal modifications over the original SiamFC implementation, while RE-SiamFC required significant modifications to the network design. We further present tracking scores with a Het-IEN implementation that reduces number of channels per layer by 31%. The results show that even with this significant drop in the number of channels, our Het-SiamFC model performs well for the object tracking task. This result clearly shows that with IEN implementations, it is possible to design trackers robust to rotations, while also providing good inference speed.

Limitations. IEN requires training of additional parameters. While our current choice of weighting terms used for equivariance loss works well for the chosen problems, it might not be the best choice for other problems. A rigorous study on this aspect is needed to understand the full potential of IEN. We choose mean-squared error as the measure of equivariance loss, however, limitations of this choice still need to be investigated. Training IEN becomes more complex when the weighting term for equivariance loss is chosen differently for each layer. We have only briefly investigated varying the weight of equivariance loss across different layers of the network, and the model performance is sensitive to it. Thus, for other problems, it might be of interest to vary this weighting term.

6 Conclusions

In this paper, we have demonstrated that while restricting the basis choice for network weights to certain set of analytical functions can help the model to achieve perfect equivariance under certain transformations, it can be sub-optimal in performance on the primary deep learning task. To circumvent this issue, we have presented an implicit equivariance formulation, referred to as Implicit Equivariance Networks (IEN), and demonstrated through several numerical experiments that IEN can induce the required amount of equivariance needed in the network to maximize the performance on the primary task. Our IEN strategy requires minimal modifications of the existing CNN models and can significantly reduce inference cost compared to SOTA methods for equivariant transformations. Further implications of our research results are presented in Section 1 of this paper.

REFERENCES

Erik J. Bekkers, Maxime W. Lafarge, Mitko Veta, Koen A. J. Eppenhof, Josien P. W. Pluim, and Remco Duits. Roto-translation covariant convolutional networks for medical image analysis. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger (eds.), *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, pp. 440–448, 2018a.

Erik J. Bekkers, Maxime W. Lafarge, Mitko Veta, Koen A. J. Eppenhof, Josien P. W. Pluim, and Remco Duits. Roto-translation covariant convolutional networks for medical image analysis. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger (eds.), *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, pp. 440–448, 2018b.

G. Benton, M. Finzi, P. Izmailov, and A. Wilson. Learning invariances in neural networks from training data. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *Computer Vision – ECCV 2016 Workshops*, pp. 850–865, 2016.

D. Bethelot, N. Carlini, I. Goodfellow, A. Oliver, N. Papernot, and C. Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999, 2016.

Taco S Cohen and Max Welling. Steerable CNNs. *International Conference on Learning Representations (ICLR)*, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Nichita Diaconu and Daniel Worrall. Learning to convolve: A generalized weight-tying approach. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1586–1595, 2019.

Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so(3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

Rohan Ghosh and Anupam K. Gupta. Scale steerable filters for locally scale-invariant convolutional neural networks. *CoRR*, abs/1906.03861, 2019.

Deepak K. Gupta, Devanshu Arya, and Efstratios Gavves. Rotation equivariant siamese networks for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming auto-encoders. In Timo Honkela, Włodzisław Duch, Mark Girolami, and Samuel Kaski (eds.), *Artificial Neural Networks and Machine Learning – ICANN 2011*, pp. 44–51, 2011.

Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1562–1577, 2021. doi: 10.1109/TPAMI.2019.2957464.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Dmitry Laptev, Nikolay Savinov, Joachim M. Buhmann, and Marc Pollefeys. Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks, 2016.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence, 2015.

M. Sajjadi, M. Javanmardi, and T. Tasdizen. Regularization with stochastic transformations and-perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Uwe Schmidt and Stefan Roth. Learning rotation-aware features: From invariant priors to equivariant descriptors. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2050–2057, 2012. doi: 10.1109/CVPR.2012.6247909.

Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *CoRR*, abs/1803.09820, 2018. URL `http://arxiv.org/abs/1803.09820`.

Kihyuk Sohn and Honglak Lee. Learning invariant representations with local transformations, 2012.

Ivan Sosnovik, Artem Moskalev, and Arnold Smeulders. Scale equivariance improves siamese tracking. *arXiv preprint arXiv:2007.09115*, 2020a.

Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. Scale-equivariant steerable networks. In *International Conference on Learning Representations*, 2020b.

P.C. Teo and Y. Hel-Or. A computational approach to steerable functions. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 313–318, 1997.

Maurice Weiler and Gabriele Cesa. General E(2)-equivariant steerable CNNs. In *Advances in Neural Information Processing Systems*, pp. 14334–14345, 2019.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems*, 2018a.

Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 849–858, 2018b.

Daniel Worrall and Gabriel Brostow. Cubenet: Equivariance to 3d rotation and translation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (eds.), *Computer Vision – ECCV 2018*, pp. 585–602, 2018.

Daniel Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5028–5037, 2017.

Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2411–2418, 2013.

APPENDIX

A EQUIVARIANCE LOSS: ADDITIONAL MATHEMATICAL DETAILS

We define our equivariance loss measure based on the absolute equivariance condition stated in Eq. 1 and express it in terms of mean-squared error as

$$\mathcal{L}_G(x) = \sum_{g \in G} \|f(\varphi_g^X(x)) - \varphi_g^Y(f(x))\|_2^2 \qquad \forall x \in X. \tag{7}$$

Here, $\mathcal{L}_G$ denotes equivariance loss computed over all group actions φ_g from G. The term f represents a subset of the IEN model $\mathcal{F}$, and typically comprises one or more convolutional layers, each followed by a nonlinear activation and a batch-normalization module. Based on the notion of feature groups defined earlier, equivariance loss for transformations G at the i^{th} layer can be stated as

$$\mathcal{L}_G^{(i)}(x_p, x_q) = \sum_{g \in G} \|\hat{\mathbf{h}}_i(x_p) - \varphi_g^Y(\hat{\mathbf{h}}_i(x_q))\|^2 \quad \text{s.t. } x_p = \varphi_g^X(x_q). \tag{8}$$

Here, x_p and x_q are two different inputs from the input set $\mathbf{x}$, and as stated in the equality constraint, one is the transformed version of other. Based on Eq. 8, the equivariance error across all L layers of the network for the input set $\mathbf{x}$ is

$$\mathcal{L}_G(\mathbf{x}) = \sum_{i=1}^{L} \sum_{g \in G} \|\hat{\mathbf{h}}_i(x_p) - \varphi_g^Y(\hat{\mathbf{h}}_i(x_q))\|^2 \cdot \beta_i \quad \forall \ \{x_p, x_q\} \in \mathbf{x} \text{ s.t. } x_p = \varphi_g^X(x_q). \tag{9}$$

The weighting term β_i provides control over the amount of equivariance imposed in different parts of the network. For layers where equivariance is to be implicitly enforced, $\beta_i > 0$, else $\beta_i = 0$.

B EXPERIMENTS: ADDITIONAL DETAILS

B.1 DATASETS

Rot-MNIST. Rot-MNIST Dataset is a variation of the popular MNIST dataset containing handwritten digits. In Rot-MNIST, the digits are rotated by an angle generated uniformly between 0 and 2π radians. Therefore, in addition to the variations induced by the different handwriting styles for the digits 0 to 9, Rot-MNIST contains an additional source of variations, the rotation angle. This dataset has been used as a benchmark in several previous papers (Worrall et al., 2017; Weiler et al., 2018b; Weiler & Cesa, 2019). It contains 12,000 images in the training dataset and 50,000 images in the validation dataset, where the size of each image is 28x28x1.

Rot-TIM and R2-TIM. TIM stands for TinyImageNet Dataset which is a miniature version of the ImageNet Dataset (Deng et al., 2009). TIM contains 200 classes. There are 500 images available for each class in the training set. On the other hand, the validation set contains 50 images for each class. All images have size [64x64x3]. We created 2 dataset variations of TIM, referred to as Rot-TIM and R2-TIM, each containing 100 classes that were randomly selected from the 200 classes in the TinyImageNet Dataset.
Rot-TIM is the Rotated variation of TIM where each image was rotated by an angle generated uniformly between 0 and 2π radians. Therefore, Rot-TIM contains a total of 50,000 images for training across 100 classes where each class has 500 training images. The validation set for Rot-TIM, which contains 10,000 images, was created by sampling 2 rotated variants of each image from the original validation set of TIM. The rotation angles were once again sampled from a uniform distribution between 0 and 2π radians.
R2-TIM is the Rotated & Reflected variation of TIM. This dataset is the same as Rot-TIM apart from the fact that some images were flipped at random along the horizontal or vertical axis.

Rot-OTB. Rot-OTB is a rotational variant of OTB100 dataset (Wu et al., 2013) and was originally presented in (Gupta et al., 2021) in the context of rotation equivariant object tracking. It contains 100 videos of the original OTB100 dataset, where the frames are rotated at a rate of 0.5 frames per second. In this paper, we use this dataset for the purpose of validation of the object tracking task.

GOT-10k. GOT-10k (Huang et al., 2021) is a popular dataset for training object tracking models. We use this training set in order to train various object tracking models in this paper. The training set comprises 9340 videos containing 840 classes of objects commonly found in daily life. Due to the diversity of objects available as well as the variations across the different challenges of object recognition, this dataset is popularly used in tracking.

B.2 Evaluation Metrics

Validation Accuracy. It denotes the percentage of samples that are correctly classified by the model from the total samples available in the validation set. The validation accuracy is therefore a measure of the ability of the learning algorithm to generalize to data not seen during training.

Precision and Success. Precision refers to the center location error and denotes the average Euclidean distance between the center location of the tracked targets and the manually labelled ground-truths (Wu et al., 2013). To understand the success score, we first need to understand overlap score. Overlap score denotes the ratio of intersection over union of the area of the predicted bounding box from a tracker and the manually labelled ground-truth. Success score denotes the average overlap score computed at various thresholds between 0 and 1. For more details, see (Wu et al., 2013).

B.3 Training Hyperparameters

In this section we provide details related to the hyperparameters used during training. We also discussed sensitivity to any new hyperparameter introduced in this paper.

Rot-MNIST classification. The E2CNN result reported in Table 5 on the Rot-MNIST classification dataset was achieved using the same conditions as described in (Weiler & Cesa, 2019). On the other hand, all results of standard CNN and IEN reported in Table 5 are achieved using Adam optimizer. One Cycle LR Policy (Smith, 2018) was used to tune the Learning Rate. A cycle of 70 epochs was employed in which initially the Learning Rate was increased from 1e-5 to 5e-3 in the first one-fourth cycle and then decreased again to 1e-5 in the remaining part of the cycle. After cycle completion, we continued training for 20 further epochs at a constant LR of 1e-5. Apart from the above mentioned set of hyperparameters, we tried several combinations of LR decay (exponential decay, one cycle policy with cycle of 40, 60 and 70 epochs), LR range (5e-3 to 1e-5 and 1e-2 to 1e-4) and weight decay (1e-5 and 1e-7) and reported the best results across these hyperparameter variations for each method in the table. Batch size of 64 was used for all experiments and β_i = 0.01, 0.1, 0.5, 1.0, 10, 100, 1000 were experimented. Among these, β_i = 1 exhibited highest performance which we have reported in the main paper.

Rot-TIM and R2-TIM classifications. For both IEN and E2CNN, the same set of hyperparameters were used as the ones used in Rot-MNIST classification experiments with just the batch size reduced to 32. We tried β_i = 0.1, 0.01 and 1 for all experiments of IEN. The best results are mentioned in the main paper while results for all values of β_i are mentioned in Table 11. In addition for both E2CNN and IEN ResNet18 architectures we interpolated the size of the image from 64 to 65 using bilinear interpolation so that after each convolution the spatial size comes out to be a perfect integer therefore allowing the equivariance to propagate to later layers.

Object tracking on Rot-OTB. All training and inference conditions are chosen to be the same as those described in (Gupta et al., 2021). We experimented with $\beta_i = 0.01, 0.1$ and 1.0. Among these, we found that the performance was best with $\beta_i = 0.1$. We reported this performance in the main paper. We believe that the performance on tracking should be more stable with respect to the choice of β_i and we are exploring this as a part of our ongoing follow up research on the subject.

B.4 Compute Resources

All Rot-MNIST experiments have been completed using Google Colab Pro. For experiments on TIM dataset, we used a machine with 1 Nvidia GeoForce Titan X card. For tracking experiments, we use 1 RTX GPU with 32Gb memory.

C ADDITIONAL RESULTS

C.1 RESULTS FOR EQUIVARIANCE UNDER HETEROGENEOUS FILTER GROUPS

Table 4: Performance scores for Heterogeneous IEN implementations on Rot-TIM dataset, created through reducing the size of fraction of feature groups per layer of the network. For example, R4-2-1 implies that R4 is modified such that 50% feature groups per layer are R4, 25% are R2, and rest are R1, thus reducing channels per layer to 69%.

Model	Eq-type	Channels%	Acc.%
	ResNet18 variants		
E2CNN	R4	100	53.5
IEN	R4	100	56.9
Het-IEN	R4-2-1	69	55.4
E2CNN	R8	100	56.5
IEN	R8	100	59.7
Het-IEN	R8-4-2-1	67	57.7
	VGG variants		
E2CNN	R4	100	46.7
IEN	R4	100	47.4
Het-IEN	R4-2-1	69	46.5
E2CNN	R8	100	50.2
IEN	R8	100	51.6
Het-IEN	R8-4-2	69	51.3
Het-IEN	R8-4-2-1	67	50.8

C.2 CLASSIFICATION OF MNIST DIGITS

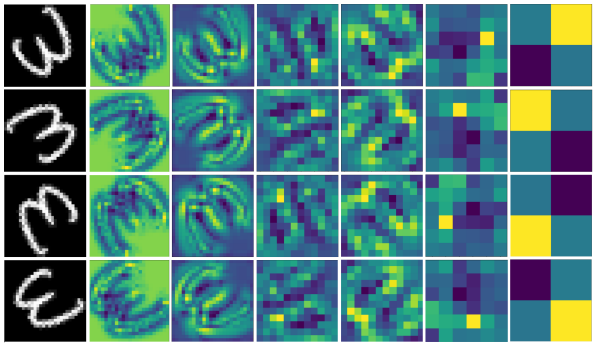
The architecture for E2CNN was adopted from (Weiler & Cesa, 2019). Equivalent standard CNN and IEN architectures were implemented such that the number of inference parameters used by these models was equal to E2CNN. The models contain 6 convolutional layers. Full results for the experiments on Rot-MNIST dataset for E2CNN, IEN, and standard CNN as well as the extent of equivariance achieved in each layer measured by the amount of equivariance loss in each layer is reported in Table 5. Also, the extent of equivariance observed in these models is shown in Figure 3 for all 6 `conv` layers. From the figure, it can be seen that equivariance is achieved in each layer of IEN.

C.3 RESULTS WITH ROT-TIM AND R2-TIM

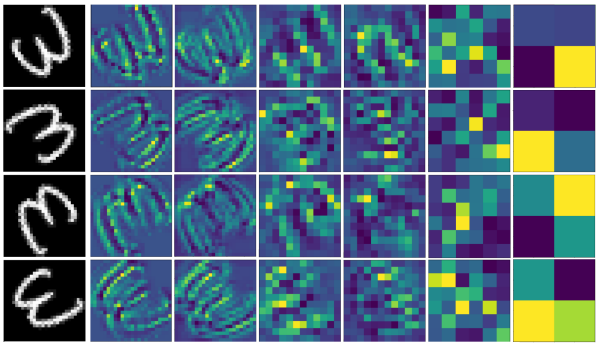
An E2CNN model equivalent to ResNet18 and VGG was developed with the same layers and equal number of total trainable parameters. To maintain equal parameters count, a ratio was found in

Table 5: Performance scores of E2CNN(Weiler & Cesa, 2019), standard CNN and IEN on Rot-MNIST validation dataset along with extent of equivariance achieved in each `conv` layer as measured by the equivariance loss for that layer.

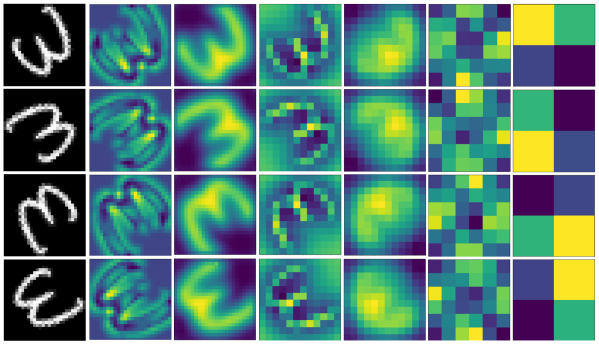
Model	Acc.%	Equivariance $\mathcal{L}_G$(lower values are better)					
		`conv1`	`conv2`	`conv3`	`conv4`	`conv5`	`conv6`
CNN (No aug)	96.3	10^{0}	10^{-1}	10^{0}	10^{0}	10^{0}	10^{-1}
CNN (with aug)	98.4	10^{0}	10^{-1}	10^{0}	10^{0}	10^{0}	10^{-1}
E2CNN	98.8	10^{-11}	10^{-10}	10^{-8}	10^{-9}	10^{-8}	10^{-8}
IEN (β_i = 1)	98.6	10^{-7}	10^{-6}	10^{-5}	10^{-5}	10^{-5}	10^{-4}
IEN (β_i = 0.1)	98.5	10^{-5}	10^{-5}	10^{-4}	10^{-4}	10^{-3}	10^{-2}
IEN (β_i = 0.01)	98.6	10^{-3}	10^{-3}	10^{-2}	10^{-2}	10^{-2}	10^{-1}
Het-IEN (β_i = 1)	98.6	10^{-6}	10^{-6}	10^{-5}	10^{-5}	10^{-5}	10^{-4}



(a) Steerable CNN (E2CNN Weiler & Cesa (2019))



(b) Regular CNN with augmentation (equivalent to IEN with $\beta_i = 0$).



(c) Implicit Equivariant Network (IEN); $\beta_i = 1.0$.

Figure 3: Extended results for the MNIST feature map representations shown in the main paper. We show here the feature maps for all 6 layers of the network for the 4 orientations of the input image.

Table 6: Details on the composition of E2CNN architectures based on ResNet18 model experimented in this paper. Here R4 and R8 denote equivariance to 4 and 8 equidistant orientations, respectively, and R4R denotes equivariance to 4 equidistant rotations and reflections. Numbers for `conv` layers denote channels per orientation. Channels per layer in all variant are chosen such that the total parameters in the model are same as the base ResNet18 model

Layer	kernel size	Resnet-18	E2CNN R4	E2CNN R8	E2CNN R4R
`conv-1`	3×3	64	34	24	24
`block-1`	3×3	64	34	24	24
	3×3	64	34	24	24
`block-2`	3×3	128	68	48	48
	3×3	128	68	48	48
`block-3`	3×3	256	136	96	96
	3×3	256	136	96	96
`block-4`	3×3	512	272	192	192
	3×3	512	272	192	192
`avg-pool`	-	512	272	192	192
`fc-layer`	-	100	100	100	100

Table 7: Details on the composition of E2CNN architectures based on VGG model experimented in this paper. Here R4 and R8 denote equivariance to 4 and 8 equidistant orientations, respectively, and R4R denotes equivariance to 4 equidistant rotations and reflections. Numbers for `conv` layers denote channels per orientation. Channels per layer in all variant are chosen such that the total parameters in the model are same as the base VGG model.

Layer	kernel size	VGG	E2CNN R4	E2CNN R8	E2CNN R4R
`conv-1`	3×3	64	36	25	25
`max-pool`	2×2	64	36	25	25
`conv-2`	3×3	128	72	50	50
`max-pool`	2×2	128	72	50	50
`conv-3`	3×3	256	144	100	100
`conv-4`	3×3	256	144	100	100
`max-pool`	2×2	256	144	100	100
`conv-5`	3×3	512	288	200	200
`conv-6`	3×3	512	288	200	200
`max-pool`	2×2	512	288	200	200
`conv-7`	3×3	512	288	200	200
`conv-8`	3×3	512	288	200	200
`avg-pool`	-	512	288	200	200
`fc-layer`	-	1024	1024	1024	1024
`fc-layer`	-	100	100	100	100

channels of the standard architecture and feature fields of E2CNN for different equivariance type. Full details about number of channels per layer in the base model and in E2CNN corresponding to equal trainable parameter count is provided in Table 8 and 9.

In addition, we developed the IEN architecture such that it exactly matches the architecture of E2CNN at inference time. Full results for the experiments on Rot-TIM and RR-TIM dataset for E2CNN, standard CNN and IEN experiments with different values of β_i are shown in Table 11. The extent of equivariance achieved in all `conv` blocks in case of ResNet18 and in 4 alternate conv

Table 8: Details on the composition of E2CNN architectures based on ResNet18 model experimented in this paper. Here R4 and R8 denote equivariance to 4 and 8 equidistant orientations, respectively, and R4R denotes equivariance to 4 equidistant rotations and reflections. Numbers for `conv` layers denote channels per orientation. Channels per layer in all variant are chosen such that the total parameters in the model are same as the base ResNet18 model

Layer	kernel size	Resnet-18	E2CNN R4	E2CNN R8	E2CNN R4R
`conv-1`	3×3	64	34	24	24
`block-1`	3×3	64	34	24	24
	3×3	64	34	24	24
`block-2`	3×3	128	68	48	48
	3×3	128	68	48	48
`block-3`	3×3	256	136	96	96
	3×3	256	136	96	96
`block-4`	3×3	512	272	192	192
	3×3	512	272	192	192
`avg-pool`	-	512	272	192	192
`fc-layer`	-	100	100	100	100

layers in case of VGG (`conv1`, `conv3`, `conv5`, `conv7`) are also reported in Table 11. Since the extent of equivariance is computed using the equivariance loss, lower values are better. The extent of equivariance observed in case of R8 in ResNet18 based E2CNN and standard CNN models is shown in Figure 4 and Figure 5, respectively. The corresponding results for IEN (β_i = 0.01), IEN (β_i = 0.1) are provided in Figure 6, Figure 7 and Figure 8, respectively.

An E2CNN model equivalent to ResNet18 and VGG was developed with the same layers and equal number of total trainable parameters. To maintain equal parameters count, a ratio was found in channels of the standard architecture and feature fields of E2CNN for different equivariance type. Full details about number of channels per layer in the base model and in E2CNN corresponding to equal trainable parameter count is provided in Table 8 and 9.

In addition, we developed the IEN architecture such that it exactly matches the architecture of E2CNN at inference time. Full results for the experiments on Rot-TIM and RR-TIM dataset for E2CNN, standard CNN and IEN experiments with different values of β_i are shown in Table 11. The extent of equivariance achieved in all `conv` blocks in case of ResNet18 and in 4 alternate conv layers in case of VGG (`conv1`, `conv3`, `conv5`, `conv7`) are also reported in Table 11. Since the extent of equivariance is computed using the equivariance loss, lower values are better. The extent of equivariance observed in case of R8 in ResNet18 based E2CNN and standard CNN models is shown in Figure 4 and Figure 5, respectively. The corresponding results for IEN (β_i = 0.01), IEN (β_i = 0.1) are provided in Figure 6, Figure 7 and Figure 8, respectively.

C.4 Results of IEN under scale variations

Results related to performance of IEN under scale variations are shown in Table 10.

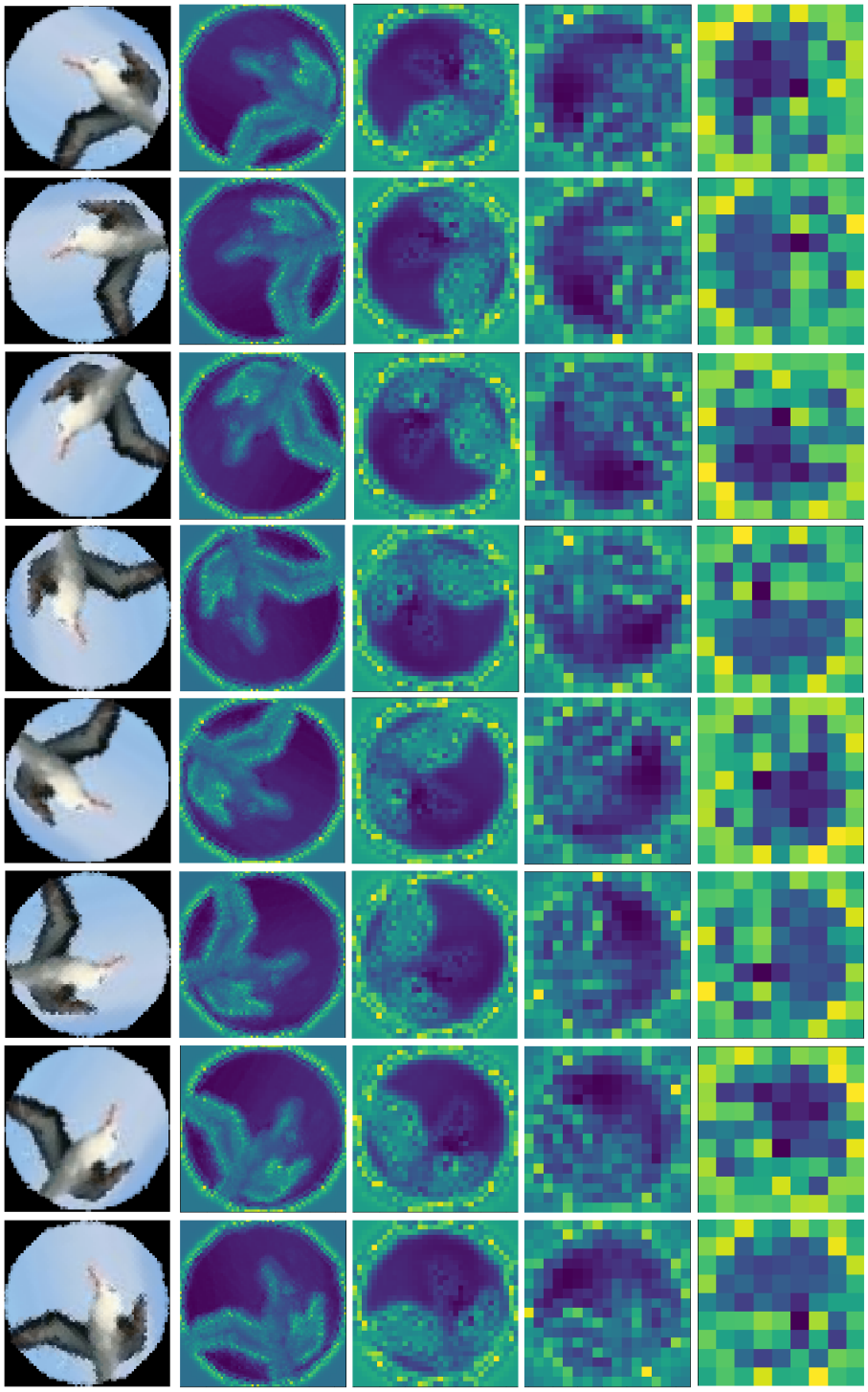


Figure 4: Feature maps outputs from 4 blocks of E2CNN variant of ResNet18 shown for 8 orientations of an input.

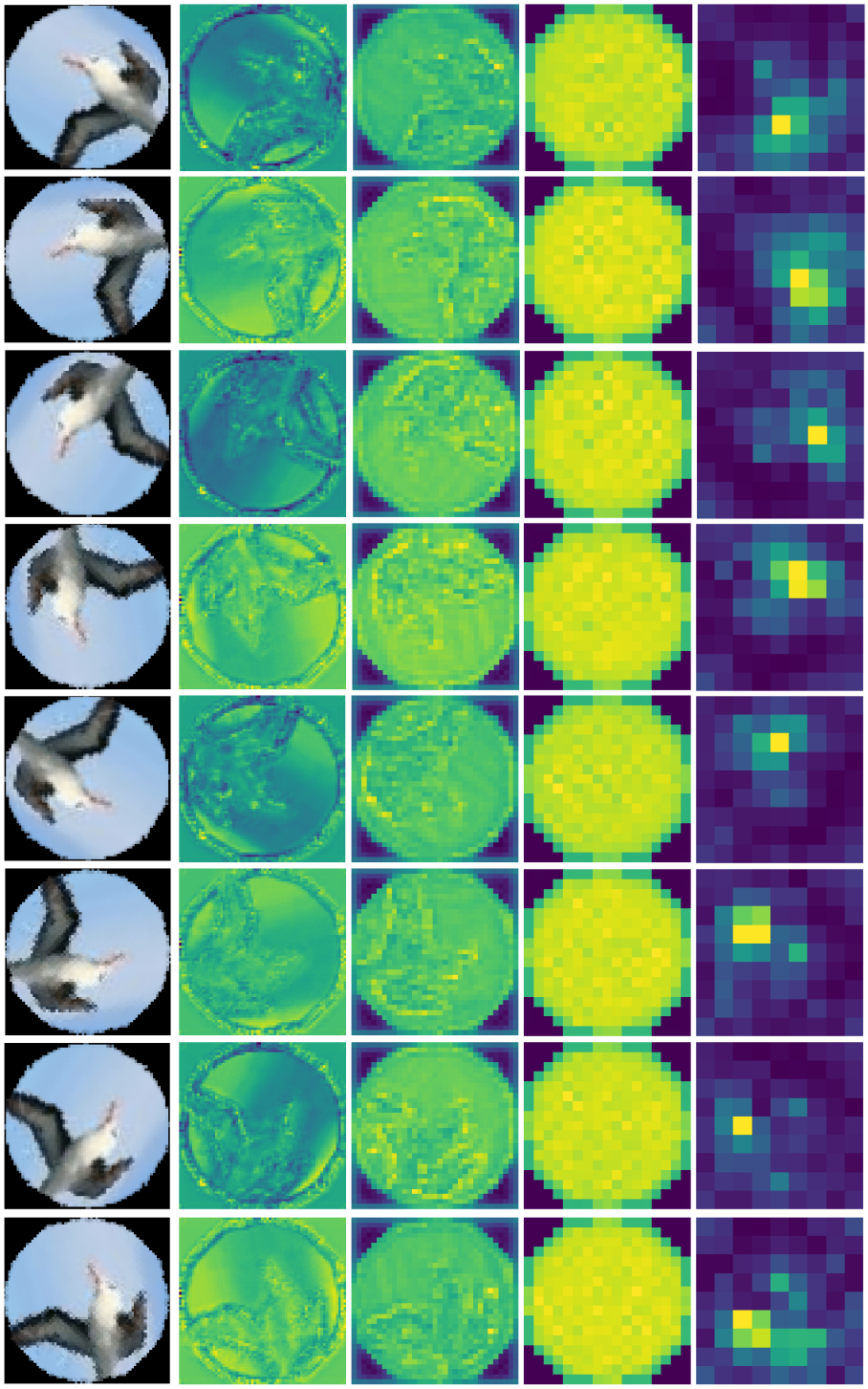


Figure 5: Feature maps outputs from 4 blocks of regular CNN-8 data augmentation version of ResNet18 shown for 8 orientations of an input.

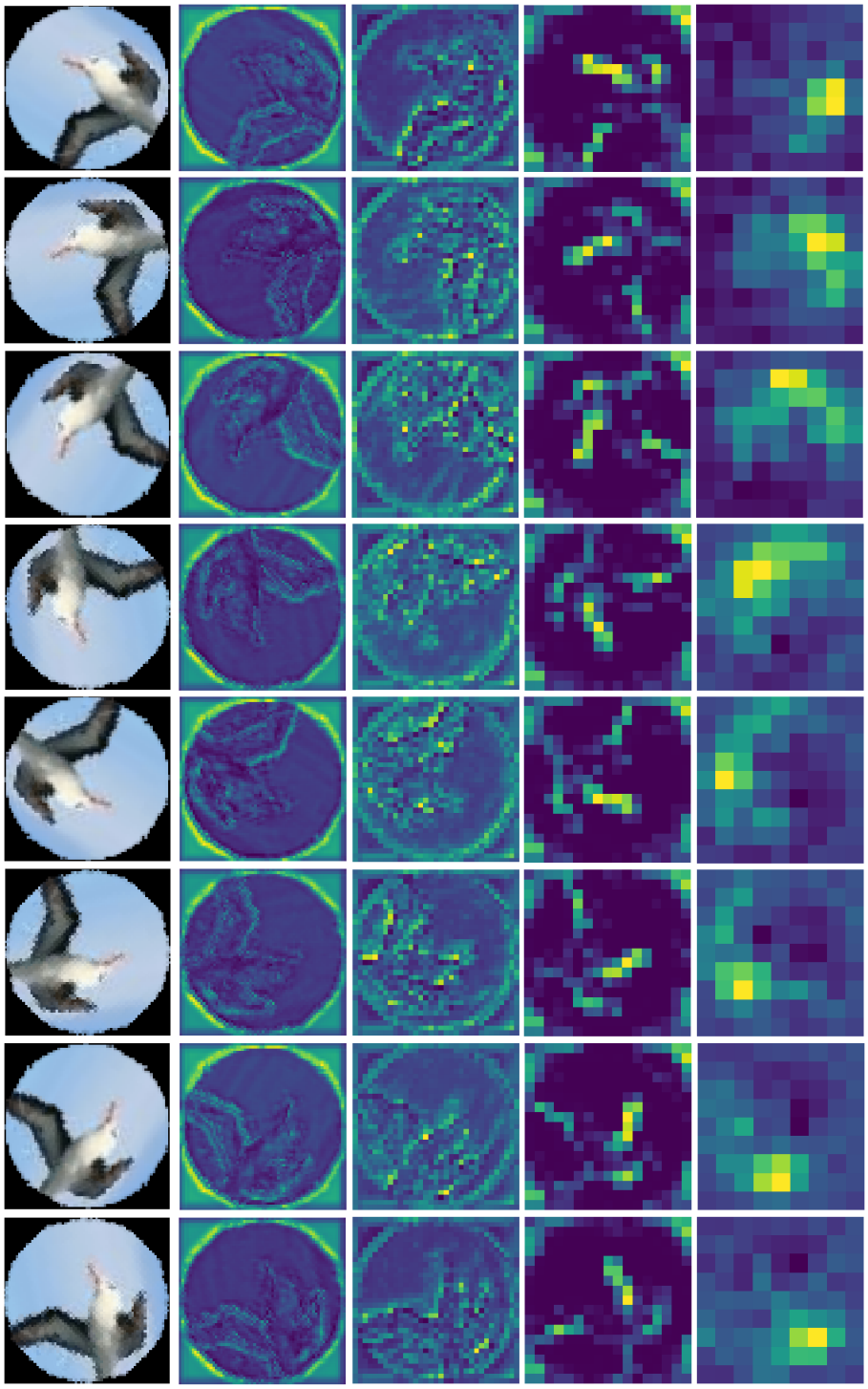


Figure 6: Feature maps outputs from 4 blocks of implicitly equivariant (IEN-R8) version of ResNet18 shown for 8 orientations of an input (all $\beta_i = 0.01$.)

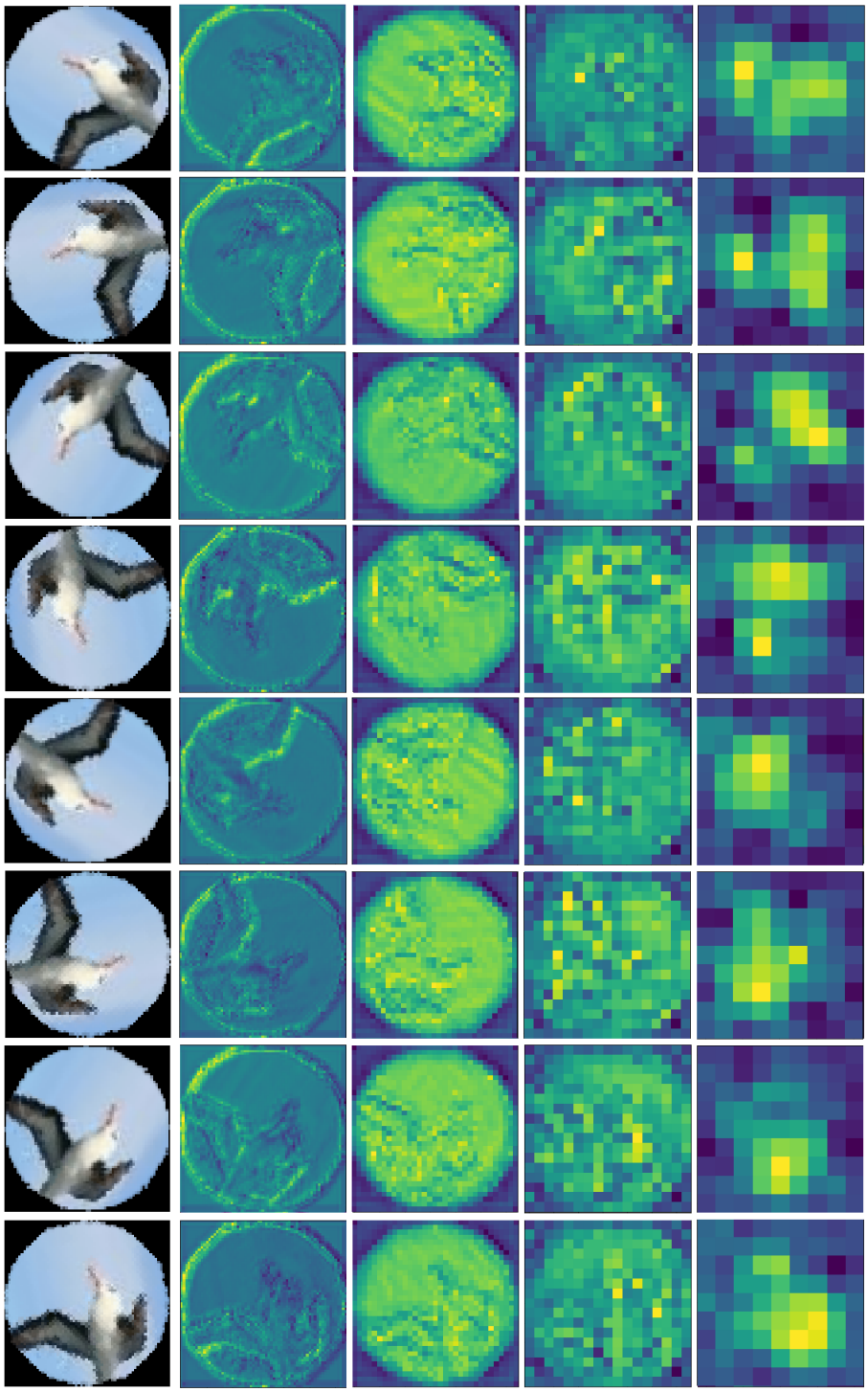


Figure 7: Feature maps outputs from 4 blocks of implicitly equivariant (IEN-R8) version of ResNet18 shown for 8 orientations of an input (all $\beta_i = 0.1$.)

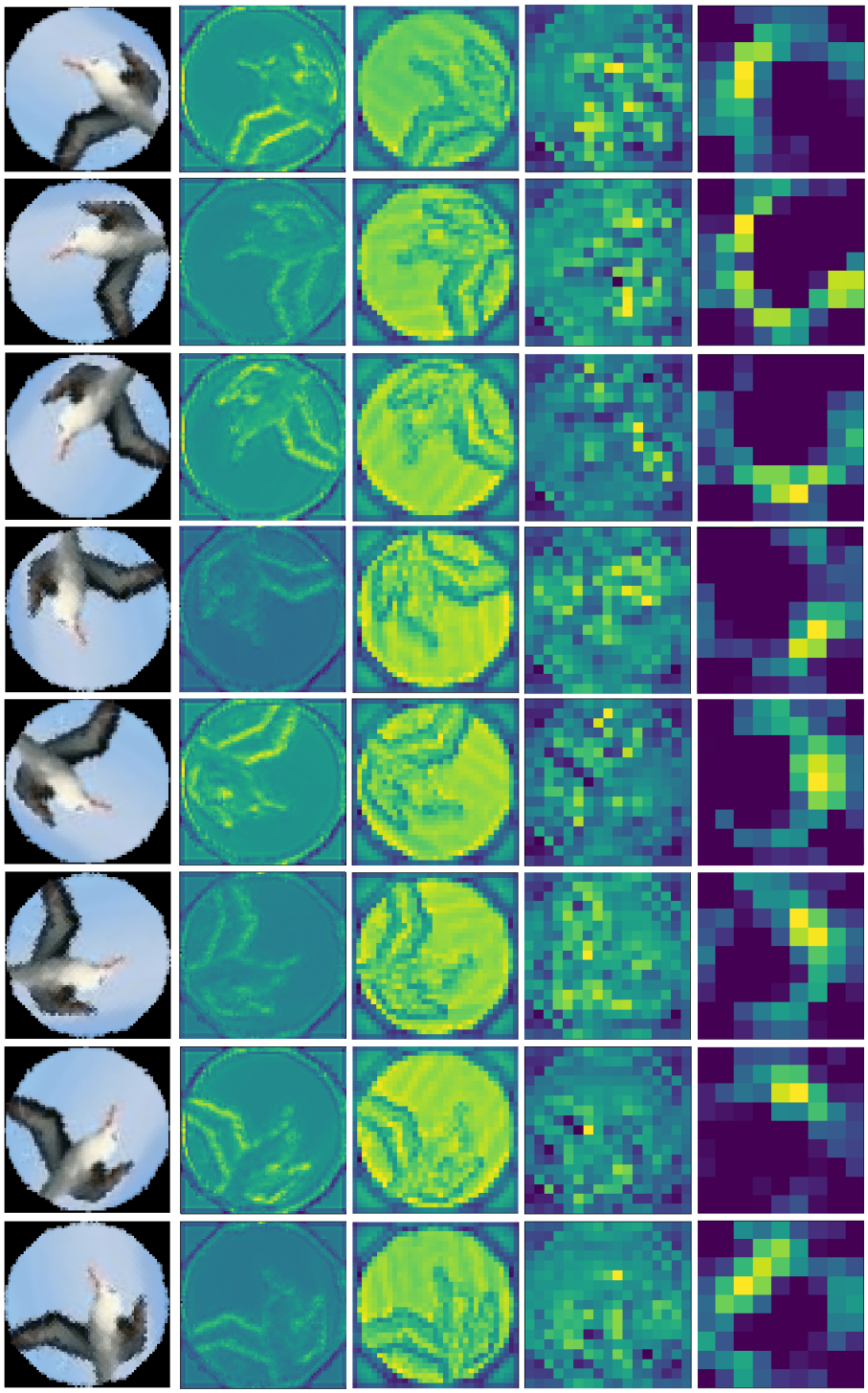


Figure 8: Feature maps outputs from 4 blocks of implicitly equivariant (IEN-R8) version of ResNet18 shown for 8 orientations of an input (all $\beta_i = 1.0$.)

Table 9: Details on the composition of E2CNN architectures based on VGG model experimented in this paper. Here R4 and R8 denote equivariance to 4 and 8 equidistant orientations, respectively, and R4R denotes equivariance to 4 equidistant rotations and reflections. Numbers for `conv` layers denote channels per orientation. Channels per layer in all variant are chosen such that the total parameters in the model are same as the base VGG model.

Layer	kernel size	VGG	E2CNN R4	E2CNN R8	E2CNN R4R
`conv-1`	3×3	64	36	25	25
`max-pool`	2×2	64	36	25	25
`conv-2`	3×3	128	72	50	50
`max-pool`	2×2	128	72	50	50
`conv-3`	3×3	256	144	100	100
`conv-4`	3×3	256	144	100	100
`max-pool`	2×2	256	144	100	100
`conv-5`	3×3	512	288	200	200
`conv-6`	3×3	512	288	200	200
`max-pool`	2×2	512	288	200	200
`conv-7`	3×3	512	288	200	200
`conv-8`	3×3	512	288	200	200
`avg-pool`	-	512	288	200	200
`fc-layer`	-	1024	1024	1024	1024
`fc-layer`	-	100	100	100	100

Table 10: Classification error on Scale-MNIST and STL-10 datasets for our IEN and three baseline methods, namely SS-CNN (Ghosh & Gupta, 2019), DSS (Worrall & Welling, 2019) and SESN (Sosnovik et al., 2020b). For Scale-MNIST, we use two variants: image sizes of 28×28 and 56×56. All baseline implementations are based on the description provided in Sosnovik et al. (2020b).

Dataset	Model	Succ.
Scale-MNIST (28×28)	SS-CNN	2.10
	DSS	1.95
	SESN	1.76
	IEN	1.78
Scale-MNIST (56×56)	SS-CNN	1.76
	DSS	1.57
	SESN	1.42
	IEN	1.33
STL-10	SS-CNN	25.47
	DSS	11.28
	SESN	10.83
	IEN	10.11

Table 11: Performance scores and equivariance loss achieved for ResNet18 and its equivariant versions as well as VGG and its equivariant versions on Rotation (Rot-TIM) and Rotation+Reflection (R2-TIM) are shown. Here R4 and R8 denote equivariance to 4 and 8 equidistant rotations and R4R denotes equivariance to 4 equidistant rotations and reflections. The extent of equivariance achieved in the 4 conv blocks of ResNet18 and in 4 alternate conv layers in case of VGG are reported for each model, lower values are better.

Eq-type	Model	ResNet18		VGG	
		Acc %	Equi.($\mathcal{L}_G$)	Acc %	Equi.
R4	CNN (No Aug)	42.5	10^{3},10^{3},10^{3},10^{0}	32.1	10^{1},10^{2},10^{2},10^{1}
	CNN (with aug)	56.7	10^{3},10^{3},10^{3},10^{0}	45.3	10^{1},10^{2},10^{2},10^{1}
	E2CNN	53.5	10^{-11},10^{-10},10^{-9},10^{-8}	46.7	10^{-11},10^{-9},10^{-9},10^{-8}
	IEN (β_i = 0.01)	56.9	10^{-3},10^{-4},10^{-3},10^{-1}	45.8	10^{-2},10^{-2},10^{-2},10^{-1}
	IEN (β_i = 0.1)	55.8	10^{-4},10^{-5},10^{-5},10^{-2}	47.4	10^{-4},10^{-3},10^{-3},10^{-2}
	IEN (β_i = 1)	54.7	10^{-5},10^{-6},10^{-5},10^{-3}	46.1	10^{-5},10^{-4},10^{-4},10^{-3}
R8	CNN (No Aug)	42.6	10^{3},10^{3},10^{3},10^{0}	32.1	10^{1},10^{2},10^{2},10^{1}
	CNN (with aug)	58.5	10^{3},10^{3},10^{3},10^{0}	51.5	10^{1},10^{2},10^{2},10^{1}
	E2CNN	56.5	10^{-10},10^{-10},10^{-9},10^{-9}	50.2	10^{-11},10^{-9},10^{-9},10^{-8}
	IEN (β_i = 0.01)	58.6	10^{-3},10^{-4},10^{-3},10^{-1}	51.6	10^{-2},10^{-2},10^{-1},10^{-1}
	IEN (β_i = 0.1)	59.7	10^{-5},10^{-5},10^{-4},10^{-2}	51.1	10^{-3},10^{-3},10^{-2},10^{-2}
	IEN (β_i = 1)	58.6	10^{-5},10^{-6},10^{-5},10^{-2}	48.9	10^{-4},10^{-4},10^{-3},10^{-3}
R4R	CNN (No Aug)	43.2	10^{3},10^{3},10^{3},10^{0}	32.5	10^{1},10^{2},10^{2},10^{1}
	CNN (with aug)	56.3	10^{3},10^{3},10^{3},10^{0}	50.0	10^{1},10^{2},10^{2},10^{1}
	E2CNN	55.9	10^{-11},10^{-10},10^{-9},10^{-9}	51.1	10^{-11},10^{-9},10^{-9},10^{-9}
	IEN (β_i = 0.01)	56.1	10^{-3},10^{-4},10^{-3},10^{-1}	49.9	10^{-2},10^{-2},10^{-1},10^{-1}
	IEN (β_i = 0.1)	55.7	10^{-4},10^{-5},10^{-4},10^{-2}	49.7	10^{-3},10^{-3},10^{-2},10^{-1}
	IEN (β_i = 1)	54.0	10^{-5},10^{-6},10^{-5},10^{-3}	48.3	10^{-5},10^{-4},10^{-3},10^{-3}