CENTER FOR Brains Minds+ Machines

CBMM Memo No. 074 December 31, 2017

# Exact Equivariance, Disentanglement and Invariance of Transformations

by

Qianli Liao and Tomaso Poggio
Center for Brains, Minds, and Machines, McGovern Institute for Brain Research,
Massachusetts Institute of Technology, Cambridge, MA, 02139.

**Abstract:** Invariance, equivariance and disentanglement of transformations are important topics in the field of representation learning. Previous models like Variational Autoencoder [1] and Generative Adversarial Networks [2] attempted to learn disentangled representations from data with different levels of successes. Convolutional Neural Networks are *approximately* equivariant and invariant (if pooling is performed) to input translations. In this report, we argue that the recently proposed Object-Oriented Learning framework [3] offers a new solution to the problem of Equivariance, Invariance and Disentanglement: it systematically factors out common transformations like translation and rotation in inputs and achieves **"exact equivariance"** to these transformations — that is, when the input is translated and/or rotated by some amount, the output and all intermediate representations of the network are also translated and rotated by **exactly the same amount**. The transformations are **"exactly disentangled"** in the sense that the translations and rotations can be read out directly from a few known variables of the system without any approximation. Invariance can be achieved by reading other variables that are known not to be affected by the transformations. No learning is needed to achieve these properties. Exact equivariance and disentanglement are useful properties that augment the expressive power of neural networks. We believe it will enable new applications including but not limited to precise visual localization of objects and measuring of motion and angles.

**This work was supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF - 1231216.**

## Contents

# 1 Introduction

We first discuss our interpretation of the following three concepts: Invariance, Equivariance and Disentanglement. **Invariance** is a property that the representation does not change under some transformations of the input. It has been studied extensively by I-theory and related work [4, 5, 6].

**Equivariance** is a property that the representation changes when input transforms (see [7] for related discussion). Note that equivariance under this general definition is easy to achieve — even random projections of the input or the input itself are equivariant. When we use the term equivariance, it stresses the fact that information of the transformations is retained/encoded by the network, instead of being discarded, contrasting the case of invariant representations.

**Disentanglement**, on the other hand, is highly related to equivariance but being more specific. It is a property that the information of some transformation of the input is not only encoded in the network/system, but also in only a few variables instead of being spreaded out in a distributed fashion. In other words, transforming the input should change the representation extracted by the network, but only affecting a few (preferably known) dimensions, leaving the other dimensions completely unchanged.

Disentanglement is a perfect balance of invariance and equivariance: for any transformation, the representation provides both invariant and equivariant code, but in separate (known) channels/dimensions. This allows for clear, versatile and interpretable manipulation of information throughout the system.

Notice that existing invariance scheme such as in convolutional networks or the more general ones described in i-theory can be made equivariant, invariant and disentagled by performing a read-out pooling operation at the very last layer.

## 1.1 Common Approaches to Invariance, Equivariance and Disentanglement

Translation is the most common transformation in a wide variaty of modalities. The most popular solution to invariance and equivariance of translation is the well-known convolutional neural networks (ConvNets) [8, 9, 10]. If spatial pooling is performed, the representation becomes invariant to local or global translation of the input.

Although ConvNets can be equivariant to translation, there is significant amount of quantization error as the signals travel from bottom to top of the network (even without local pooling). In higher layers, the spatial locations of inputs are not encoded in a precise fashion — the spatial uncertainty accumulates as the number of layers increases. Local and global poolings, if added to ConvNet, discard translation information in a unrecoverable fashion, making this problem even worse.

There has been a large body of literature (to be discussed more later) focusing on learning disentangled representations from data using Variational Autoencoder [1] and Generative Adversarial Networks [2]. They tend to show that some dimensions in high layers of the network encodes common transformations like translation, rotation and some more general transformations in a mixed fashion on digits (MNIST) [1] and recently on domain-specific datasets [11, 12]. As most of these models adopt ConvNets, they suffer from the same issue of not being precise (e.g., loss of translation information in higher layers). Hinton's Capsule model [13] also learns some level of disentangled representations from data. The limitation of above approaches are that they are very data dependent and have not been generalized well to broad categories of real-world objects (e.g., ImageNet). And it is not clear how to systematically deal with several transformations without interference.

## 1.2 Object-Oriented Learning: A New Solution to Invariance, Equivariance and Disentanglement

Object-Oriented Learning [3] provides a new approach to the problem of invariance, equivariance and disentanglement. Instead of encoding transformations into distributed code as that in the traditional "feature-oriented" learning, we adopt a basic representational atom “object/symbol” that packages transformation parameters as fields/properties (conceptually analogous to a class in object-oriented programming).

In our current system, each object has several fields/properties: its location represented by x and y coordinates (and z if extended into 3D), its pose of rotation represented by an angle (or quaternion in the case of 3D), its scale represented by a scaling factor and finally its signature, which is a vector that encodes information that can identify the object but invariant to its transformations (e.g., translation, rotation and scaling).

Each field of an object represents a disentangled transformation parameter. When an object is predicted, its transformation parameters are also predicted. The transformation parameters and signature of objects can be trained end-to-end on any task.

## 1.3 Equivariance and Disentanglement with Arbitrarily Fine Precision

The main advantage of our treatment of equivariance and disentanglement is that in our framework they can be made **“arbitrarily precise”**. Unlike ConvNets, which are restricted by its grid representation and inability to represent rotations and scalings, our framework encodes transformation parameters as fields of the objects, with arbitrary precision (e.g., floating point numbers in our experiments).

Our framework offers a new level of expressive power: whenever the input is transformed (e.g., translated, rotated or scaled [1]) by some amount $T_g$, the representation in every intermediate layer of our network reflects **exactly the same amount** of transformation $T_g$ in the objects' fields.

**Property 1: Exact Equivariance:** Let we use $x$ to denote the input to the system (or to any intermediate layers of it). Let $F()$ be the function of arbitrary number of “object-oriented layers”, $T_g()$ be a transformation function parameterized by transformation parameter $g$. We have:

$$T_g(F(x)) \equiv F(T_g(x)) \tag{1}$$

Essentially, the transformation commutes with arbitrary number of layers of processing of our object-oriented layers. In our current implementation, T() currently supports a combination of translations and rotations (in both 2D and 3D).

**Property 2: Exact Disentanglement:** there exist a read-out function R() that can always extract transformation parameters by comparing the **high level representations** before and after a transformation of the **input**:

$$R(F(T_g(x)), F(x)) \equiv g \tag{2}$$

Note that R() should be as simple as possible (e.g., reading only several known dimensions of the high-level representation) and ideally be non-parametric since if substantial learning is required to read transformation parameters, it can hardly be called a “disentangled representation”.

In above definitions, our models guarantees $\equiv$, that is why we name them “exact” equivariance and disentanglement. In previous models (like ConvNets), $\equiv$ may become $\approx$, thus becoming not exact.

# 2 Model

We describe the mechanisms that are needed to achieve Property 1 and 2.

[1]Scaling is currently treated in the system as several discrete scales. Continuous scaling is being implemented.

### 2.1 Voting Layer

The voting layer is described very briefly in [3] and in a more general and principled way in [14]. The main conclusion from [14] is the following claim:

**Proposition 1: Commutative Property of Voting Layer**

Assume the input $x$ are a list of objects (instead of pixels), the voting layer $V()$ commutes with the transformation $T()$. That is:

$$T(V(x)) = V(T(x)) \tag{3}$$

Note that T(x) means that the same transformation is applied to every element of the list of objects $x$.

### 2.2 Binding Layer

The binding layer is different from that in [3]. The main idea is to select a subset of objects as representatives of local clusters of using a suppression mechanism, and these representatives will bind neighbor objects (and itself) to form a new object. It has two meta parameters: binding radius $r_b$ and suppression radius $r_s$. There are different binding procedures within this class that can satisfy the exact equivariance property, but the the particular binding procedure we implemented is: 1. every object counts the number of neighbors within radius $r_b$ in euclidean space (2D or 3D) and compute the sum of distances from neighbors to itself. 2. Then every object looks at neighbors within radius $r_s$ to see if it is the object with most neighbors (if there is a tie, check if it has the lowest distances). If so, it becomes a representative. 3. Each representative averages all objects within radius $r_s$. By averaging, we refer to both averaging of signatures and properties To be precise, in our implementation, we performed sum of signatures and average of properties. Average of signatures should behave qualitatively similar, but we did not try it.

Finally, with this binding procedure, we have the additional claim:

**Proposition 2: Commutative Property of Binding Layer** Assume the input $x$ are a list of objects (instead of pixels), the voting layer $B()$ commutes with the transformation $T()$. That is:

$$B(V(x)) = B(T(x)) \tag{4}$$

Note that T(x) means that the same transformation is applied to every element of the list of objects $x$.

Both binding and voting layers commute with the transformations (translation and rotations).

## 3 Experiments

### 3.1 Examples of Exact Equivariance

We show several examples of exact equivariance on some simple data in Figure 1, 2 and 3.

### 3.2 Exact Equivariance in 3D

See [14] for extension of this model in 3D We have also checked

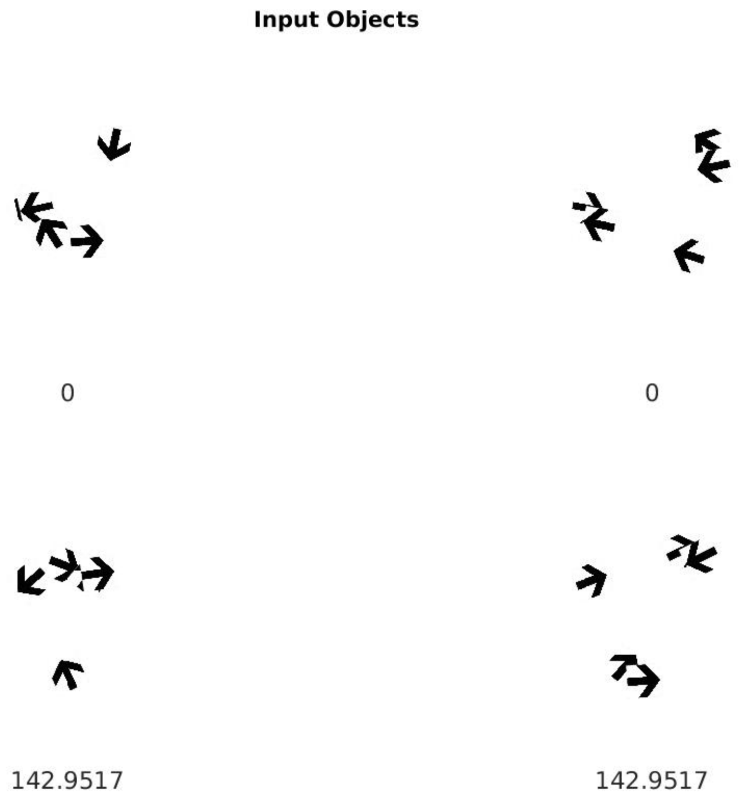

Figure 1: **Exact Equivariance Visualization:** Input objects to our network. First row shows two patterns (each pattern is a collection of objects, illustrated by arrows). Second row shows the two patterns rotated by some angles (indicated below the image). The rotated images look slightly different because the objects are drawn in some random orders — the overlay orders are different.

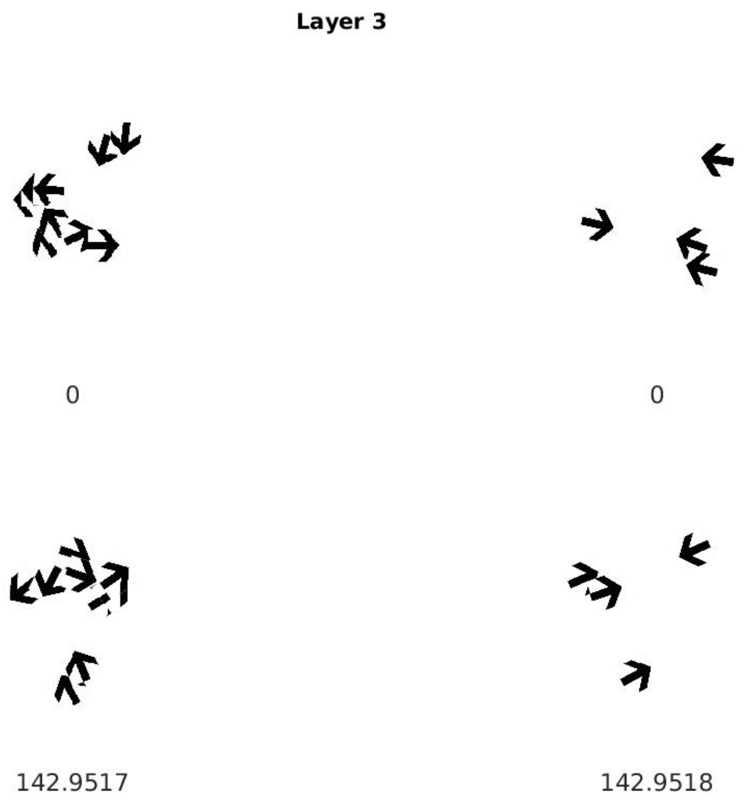

Figure 2: **Exact Equivariance Visualization:** Activation of the 3rd layer of our network. Recall that all the intermediate activations of our network are also objects. Each sub-figure corresponds to the 3rd layer activations induced by corresponding inputs in Figure 1. The numbers below the sub figures are the average of rotation angles of the objects (using first row objects as frame of reference — that is why first row values are 0s). We can see that the 3rd layer activations are exactly equivariant to rotations of the input in Figure 1. The rotated images might look slightly different because the objects are drawn in some random orders — the overlay orders are different.

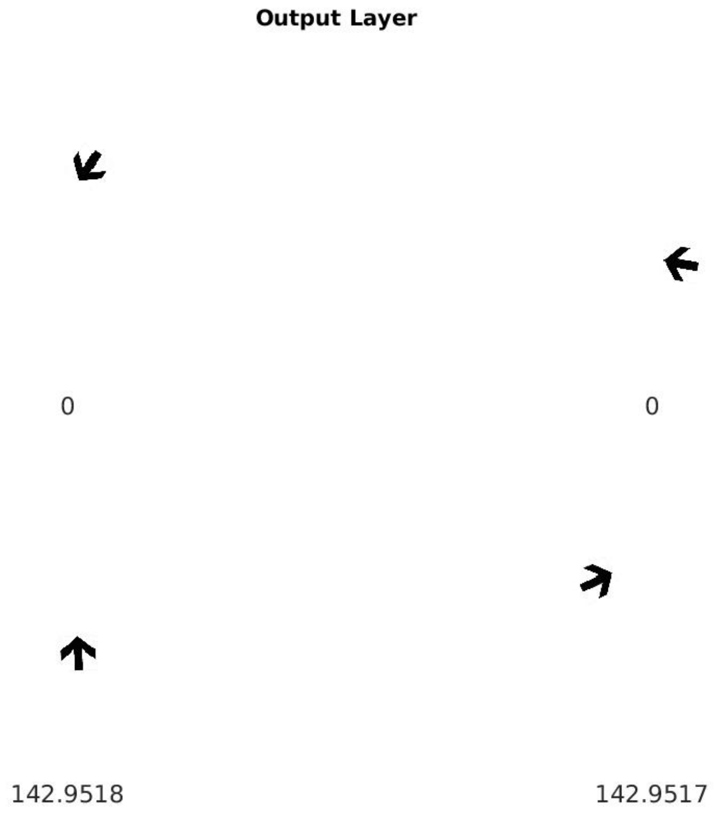

Figure 3: **Exact Equivariance Visualization:** Activation of the 6th layer of our network. Recall that all the intermediate activations of our network are also objects. Each sub-figure corresponds to the 6th layer activations induced by corresponding inputs in Figure 1. The numbers below the sub figures are the average of rotation angles of the objects (using first row objects as frame of reference — that is why first row values are 0s). We can see that the 6th layer activations are exactly equivariant to rotations of the input in Figure 1.

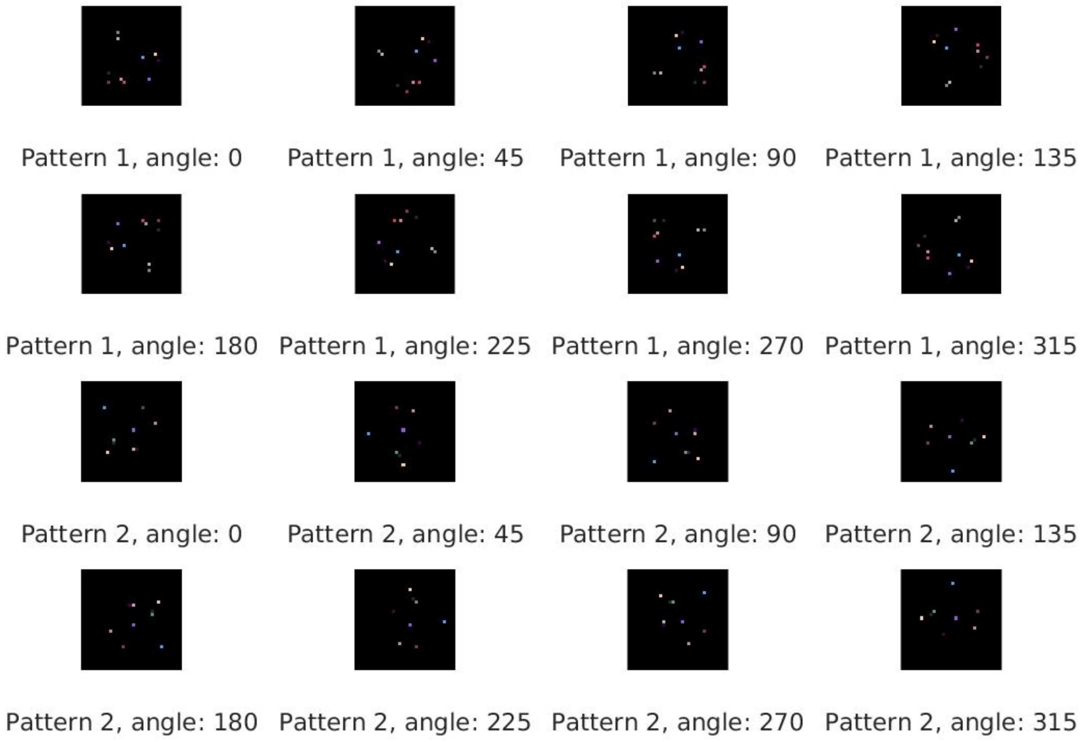

Figure 4: Visualization of 2D Patterns used in our experiments

### 3.3 Invariant Recognition of 2D Patterns

We train and test our model and ConvNet (the same one used in [3]) on recognizing 2D patterns. There are 10 "objects" (i.e., 10 dots in each pattern) scattered in 10 different ways (i.e., 10-way classification with chance performance being 90% error). See Figure 4 for visualization. The inputs to our model are objects. Each object has a signature (3-dimensional, RGB value), a position (x and y) and a pose (scalar angle). To compare with ConvNet, we embed the objects' signature and pose into corresponding positions of images. Since positions in images can only take integer values, we add a uniform noise in [-1,1] to the positions of the objects (different for every minibatch) and rounded them to integers — this input preprocessing procedure is done for both our model and ConvNet so that both models take in the same inputs.

The results are shown in Figure 5 and 6. Our model generalizes perfectly to novel rotations that the model has not seen before.

## References

[1] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[3] Q. Liao and T. Poggio, "Object-oriented deep learning," tech. rep., Center for Brains, Minds and Machines (CBMM), https://dspace.mit.edu/handle/1721.1/112103, October, 2017.

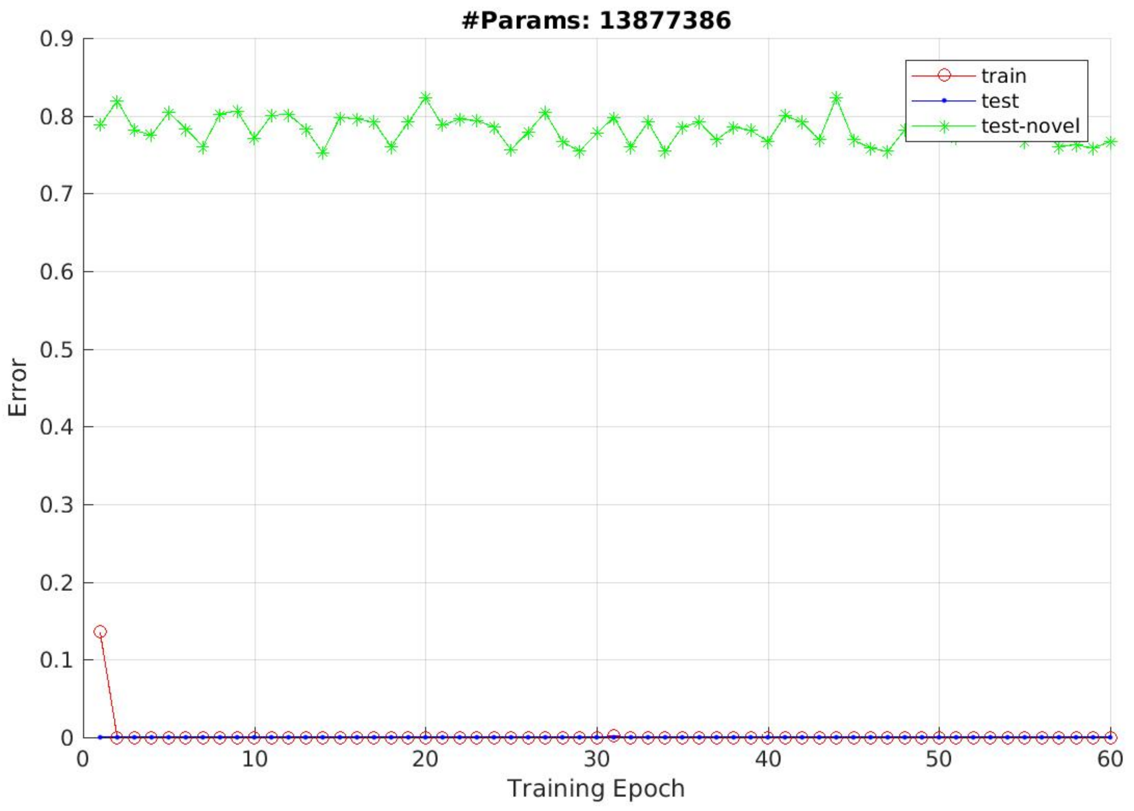

Figure 5: Performance of ConvNet (ResNet). It converges to 0 error quickly on training, but does not generalize at all to novel rotations.

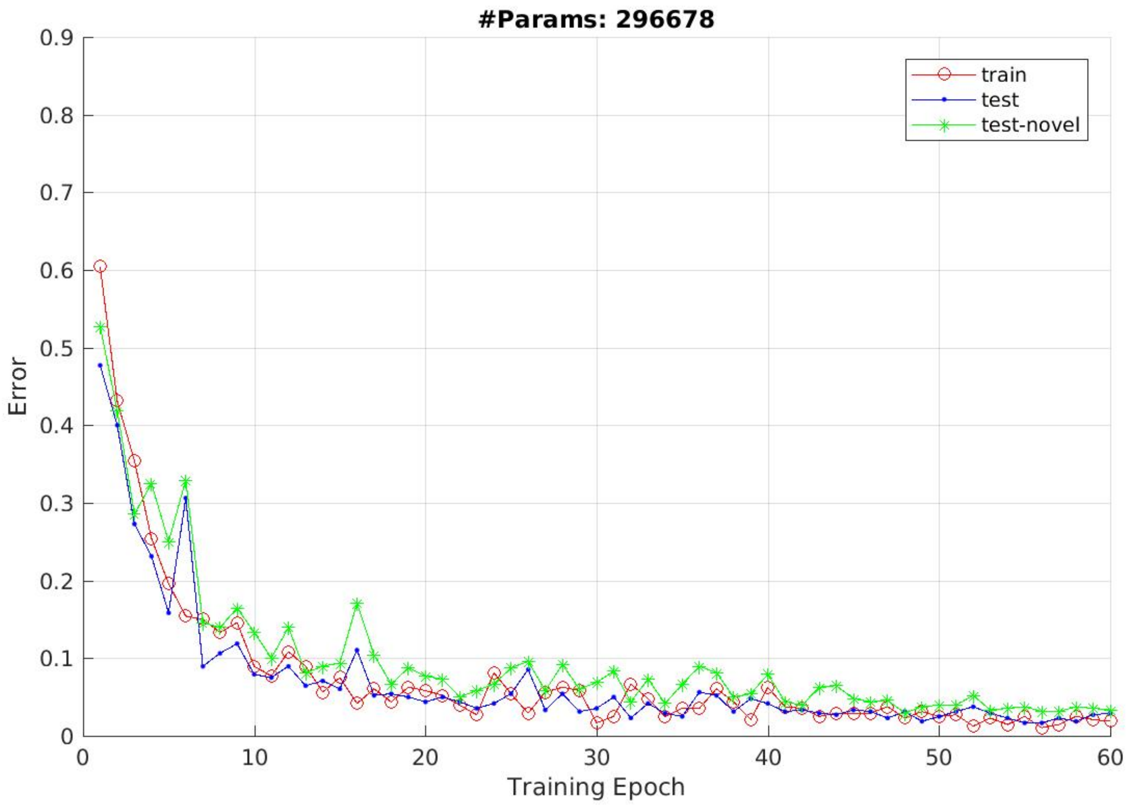

Figure 6: Performance of our model. Thanks to the exact equivariance and disentanglement property, there is no difference between the performance of training (rotation=0) test (rotation=0) and test-novel (360 degrees of rotations).

[4] F. Anselmi, J. Z. Leibo, L. Rosasco, J. Mutch, A. Tacchetti, and T. Poggio, "Unsupervised learning of invariant representations in hierarchical architectures," *arXiv preprint arXiv:1311.4158*, 2013.

[5] Q. Liao, J. Z. Leibo, and T. Poggio, "Learning invariant representations and applications to face verification," in *Advances in Neural Information Processing Systems*, pp. 3057–3065, 2013.

[6] J. Z. Leibo, Q. Liao, F. Anselmi, and T. Poggio, "The invariance hypothesis implies domain-specific regions in visual cortex," *bioRxiv, 10.1101/004473*, 2014.

[7] K. Lenc and A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 991–999, 2015.

[8] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, Apr. 1980.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, November 1998.

[10] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nat. Neurosci.*, vol. 2, no. 11, pp. 1019–1025, 1999.

[11] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," in *Advances in Neural Information Processing Systems*, pp. 2539–2547, 2015.

[12] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[13] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *International Conference on Artificial Neural Networks*, pp. 44–51, Springer, 2011.

[14] Q. Liao and T. Poggio, "3d object-oriented learning: An end-to-end transformation-disentangled 3d representation," tech. rep., Center for Brains, Minds and Machines (CBMM), 2017.