Equivariant Transformer Networks

Kai Sheng Tai 1  Peter Bailis 1  Gregory Valiant 1

Abstract
How can prior knowledge on the transformation invariances of a domain be incorporated into the architecture of a neural network? We propose Equivariant Transformers (ETs), a family of differentiable image-to-image mappings that improve the robustness of models towards pre-defined continuous transformation groups. Through the use of specially-derived canonical coordinate systems, ETs incorporate functions that are equivariant by construction with respect to these transformations. We show empirically that ETs can be flexibly composed to improve model robustness towards more complicated transformation groups in several parameters. On a real-world image classification task, ETs improve the sample efficiency of ResNet classifiers, achieving relative improvements in error rate of up to 15% in the limited data regime while increasing model parameter count by less than 1%.

1. Introduction
In computer vision, we are often equipped with prior knowledge on the transformation invariances of a domain. Consider, for example, the problem of classifying street signs in real-world images. In this domain, we know that the appearance of a sign in an image is subject to various deformations: the sign may be rotated, its scale will depend on its distance, and it may appear distorted due to perspective in 3D space. Regardless, the identity of the street sign should remain invariant to these transformations.

With the exception of translation invariance, convolutional neural network (CNN) architectures typically do not take advantage of such prior knowledge on the transformation invariances of the domain. Instead, current standard practice heuristically incorporates these priors during training via data augmentation (e.g., by applying a random rotation or scaling to each training image). While data augmentation typically helps reduce the test error of CNN-based models, there is no guarantee that transformation invariance will be enforced for data not seen during training.

In contrast to training time approaches like data augmentation, recent work on group equivariant CNNs (Cohen & Welling, 2016; Dieleman et al., 2016; Marcos et al., 2017; Worrall et al., 2017; Henriques & Vedaldi, 2017; Cohen et al., 2018) has explored new CNN architectures that are guaranteed to respond predictably to particular transformations of the input. For example, the CNN model family may be constrained such that a rotation of the input results in a corresponding rotation of its subsequent representation, a property known as equivariance. However, these techniques—most commonly designed for rotations and translations of the input (e.g., Dieleman et al. (2016); Marcos et al. (2017); Worrall et al. (2017))—fail to generalize to deeper compositions of continuous transformations. This limits the applicability of these techniques in more complicated real-world scenarios involving continuous transformations in several dimensions, such as the above example of street sign classification.

To address these shortcomings of group equivariant CNNs, we propose Equivariant Transformer (ET) layers, a flexible class of functions that improves robustness towards arbitrary pre-defined groups of continuous transformations. An ET layer for a transformation group $G$ is an image-to-image mapping that satisfies the following local invariance property: for any input image $\phi$ and transformation $T \in G$, the images $\phi$ and $T\phi$ are both mapped to the same output image. ET layers are differentiable with respect to both their parameters and input, and thus can be easily incorporated into existing CNN architectures. Additionally, ET layers can be flexibly combined to achieve improved invariance towards more complicated compositions of transformations (e.g., simultaneous rotation, scale, shear, and perspective transformations).

Importantly, the invariance property of ETs holds by construction, without any dependence on additional heuristics during training. We achieve this by using the method of canonical coordinates for Lie groups (Rubinstein et al., 1991). The key property of canonical coordinates that we utilize is their ability to reduce arbitrary continuous transfor-
Equivariant Transformer Networks

In this section, we begin by reviewing influential prior work on image canonicalization with Spatial Transformers (Jaderberg et al., 2015). We then argue that the lack of self-consistency in pose prediction is a key weakness with the standard ST that results in poor sample efficiency.

3.1. Image Canonicalization with Spatial Transformers

Suppose that we observed a collection of images \( \phi(x) \), each of which is a mapping from image coordinates \( x \in \mathbb{R}^2 \) to pixel intensities in each channel. Each image is a transformed version of some latent canonical image \( \phi_* : \phi = T_0 \phi_* : \phi_* = \phi_*(T_0 x) \), where the transformation \( T_0 : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \) is modulated by pose parameters \( \theta \in \mathbb{R}^k \).
Sample complexity for predicting rotations.

We refer to this desired property of the pose prediction CNN as self-consistency. In general, we say a function \( f : \Phi \rightarrow \mathbb{R}^k \) is self-consistent with respect to a transformation group \( G \) parameterized by \( \theta \in \mathbb{R}^k \) if \( f(T_\theta \phi) = f(\phi) + \theta \), for any image \( \phi \) and transformation \( T_\theta \in G \). We note that self-consistency is a special case of group equivariance.\(^2\)

However, there is no guarantee that self-consistency should hold when pose prediction is performed using a standard CNN or fully-connected network: while standard CNNs are equivariant with respect to translation, they are not equivariant with respect to other transformation groups (Cohen & Welling, 2016). In Figure 1, we illustrate a simple example of this limitation of standard CNNs. Using MNIST digits rotated by angles uniformly sampled in \( \theta \in [0, 2\pi) \), we train a CNN classifier with a ST layer that predicts the rotation angle of the input image. During training, the model receives a rotated input as input along with the class label \( y \in \{0, \ldots, 9\} \); the true rotation angle \( \theta \) is unobserved. In this example task, we find that the poses predicted by the CNN are only approximately self-consistent within a small range of angles, even when the network is trained with 50,000 examples. In contrast, a rotation-equivariant CNN can achieve approximate self-consistency given only 10,000 training examples.

\(^2\)A function \( f \) is equivariant with respect to the group \( G \) if there exist transformations \( T_g \) and \( T'_g \) such that \( f(T_g \phi) = T'_g f(\phi) \) for all \( g \in G \) and \( \phi \in \Phi \).
4. Equivariant Transformers

Due to this weakness of standard CNN pose predictors, we will instead use functions that are guaranteed by construction to satisfy self-consistency. We achieve this by leveraging the translation equivariance of standard CNN architectures in combination with specialized canonical coordinate systems designed for the particular transformation groups of interest. Canonical coordinates allow us to reduce the problem of self-consistent prediction with respect to an arbitrary continuous transformation group to that of self-consistent prediction with respect to the translation group.

We begin with preliminaries on canonical coordinates systems (§4.1). We then describe our proposed Equivariant Transformer architecture (§4.2). Next, we describe how canonical coordinates can be derived for a given transformation (§4.3). Finally, we describe how ET layers can be applied sequentially to handle compositions of several transformations (§4.4) and cover implementation details (§4.5).

4.1. Canonical Coordinate Systems for Lie Groups

The method of canonical coordinates was first described by Rubinstein et al. (1991) and later developed in more generality by Segman et al. (1992) for the purpose of computing image descriptors that are invariant under the action of continuous transformation groups.

A Lie group with parameters \( \theta \in \mathbb{R}^k \) is a group of transformations of the form \( T_\theta : \mathbb{R}^d \to \mathbb{R}^d \) that are differentiable with respect to \( \theta \). We let the parameter \( \theta = 0 \) correspond to the identity element, \( T_0 = 1 \). A canonical coordinate system for \( G \) is defined by an injective map \( \rho \) from Cartesian coordinates to the new coordinate system that satisfies

\[
\rho(T_\theta x) = \rho(x) + \sum_{i=1}^k \theta_i e_i,
\]

for all \( T_\theta \in G \), where \( e_i \) denotes the \( i \)-th standard basis vector. Thus, a transformation by \( T_\theta \) appears as a translation by \( \theta \) under the canonical coordinate system. To help build intuition, we give two examples of canonical coordinates:

**Example 1 (Rotation).** For \( T_\theta x = (x_1 \cos \theta - x_2 \sin \theta, x_1 \sin \theta + x_2 \cos \theta) \), a canonical coordinate system is the pole coordinate system, \( \rho(x) = (\tan^{-1}(x_2/x_1), \sqrt{x_1^2 + x_2^2}) \).

**Example 2 (Horizontal Dilation).** For \( T_\theta x = (x_1 e^\theta, x_2) \), a canonical coordinate system is \( \rho(x) = (\log x_1, x_2) \).

Reduction to Translation. The key property of canonical coordinates is their ability to adapt translation self-consistency to other transformation groups. Formally, this is captured in the following result (we defer the straightforward proof to the Appendix):
Local Invariance. Unlike ST layers, ET layers are endowed with a form of local transformation invariance: for any input image \( \phi \), we have that \( L_{G, \rho}(\phi) = L_{G, \rho}(T_\theta \phi) \) for all \( T_\theta \in G \). In other words, an ET layer collapses the orbit generated by the group action on an image to a single, “canonical” point. This property follows directly from the self-consistency of the pose predictor with respect to the group \( G \). Importantly, local invariance holds for any setting of the parameters of the ET layer; thus, ETs are equipped with a strong inductive bias towards invariance with respect to the transformation group \( G \).

Implementing Self-Consistency. We implement translation self-consistency in \( f \) by first predicting a spatial distribution by passing a 2D CNN feature map through a softmax function, and then outputting the coordinates of the centroid of this distribution. By the translation equivariance of CNNs, a shift in the CNN input results in a corresponding shift in the predicted spatial distribution, and hence the location of the centroid. We rescale the centroid coordinates to match the scale of the input coordinate grid.

4.3. Constructing Canonical Coordinates (Algorithm 1)

In order to construct an ET layer, we derive a canonical coordinate system for the target transformation. Canonical coordinate systems exist for all one-parameter Lie groups (Segman et al., 1992; Theorem 1). For Lie groups with more than one parameter, canonical coordinates exist for Abelian groups of dimension \( k \leq d \); that is, groups whose transformations are commutative.

Here, we summarize the procedure described in Segman et al. (1992). For clarity of exposition, we will focus on Lie groups representing transformations on \( \mathbb{R}^2 \) with one parameter \( \theta \in \mathbb{R} \). This corresponds to the practically useful case of one-parameter deformations of 2D images. In this setting, condition (1) reduces to:

\[
\rho(T_\theta \mathbf{x}) = \rho(\mathbf{x}) + \theta \mathbf{e}_1.
\]

Taking the derivative with respect to \( \theta \), we can see that it suffices for \( \rho \) to satisfy the following first-order PDEs:

\[
\begin{align*}
\left( \frac{\partial}{\partial \theta} \right)_{\theta=0} \frac{\partial}{\partial x_1} + \frac{\partial}{\partial \theta} \frac{\partial}{\partial x_2} \right) \rho_1(\mathbf{x}) &= 1, \\
\left( \frac{\partial}{\partial \theta} \right)_{\theta=0} \frac{\partial}{\partial x_1} + \frac{\partial}{\partial \theta} \frac{\partial}{\partial x_2} \right) \rho_2(\mathbf{x}) &= 0.
\end{align*}
\]

We can solve these first-order PDEs using the method of characteristics (e.g., Strauss, 2007). Observe that the homogeneous equation (4) admits an infinite set of solutions \( \rho_2 \); each solution is a different coordinate function that is invariant to the transformation \( T_\theta \). Thus, there exists a degree of freedom in choosing invariant coordinate functions; due to the finite resolution of images in practice, we recommend choosing coordinates that minimally distort the input image to mitigate the introduction of resampling artifacts.

Example 3 (Hyperbolic Rotation). As a concrete example, we will derive a set of canonical coordinates for hyperbolic rotation, \( T_\theta \mathbf{x} = (x_1 e^\theta, x_2 e^{-\theta}) \). This is a “squeeze” distortion that dilates an image along one axis and compresses it along the other. We obtain the following PDEs:

\[
\begin{align*}
(x_1 \partial / \partial x_1 - x_2 \partial / \partial x_2) \rho_1(\mathbf{x}) &= 1, \\
(x_1 \partial / \partial x_1 - x_2 \partial / \partial x_2) \rho_2(\mathbf{x}) &= 0.
\end{align*}
\]

In the first quadrant, the solution to the inhomogeneous equation is \( \rho_1(\mathbf{x}) = \log \sqrt{x_1^2 + x_2} + c_1 \), where \( c_1 \) is an arbitrary constant, and the solution to the homogeneous equation is \( \rho_2(\mathbf{x}) = h(x_1 x_2) \), where \( h \) is an arbitrary differentiable function in one variable (the choice \( h(z) = \sqrt{z} \) is known as the hyperbolic coordinate system). These coordinates can be defined analogously for the remaining quadrants to yield a representation of the entire image plane, excluding the lines \( x_1 = 0 \) and \( x_2 = 0 \).

4.4. Compositions of Transformations

A single transformation group with one parameter is typically insufficient to capture the full range of variation in object pose in natural images. For example, an important transformation group in practice is the 8-parameter projective linear group \( \text{PGL}(3, \mathbb{R}) \) that represents perspective transformations in 3D space.

In the special case of two-parameter Abelian Lie groups, we can construct canonical coordinates that yield self-consistency simultaneously for both parameters (Segman et al., 1992; Theorem 1). For example, log-polar coordinates are canonical for both rotation and dilation. However, for transformations on \( \mathbb{R}^d \), a canonical coordinate system can only satisfy condition (1) for up to \( d \) parameters. Thus, a single canonical coordinate system is insufficient for higher-dimensional transformation groups on \( \mathbb{R}^2 \) such as \( \text{PGL}(3, \mathbb{R}) \).

Stacked ETs. Since we cannot always achieve simultaneous self-consistency with respect to all the parameters,
of the transformation group, we instead adopt the heuristic approach of using a sequence of ET layers, each of which implements self-consistency with respect to a subgroup of the full transformation group. Intuitively, each ET layer aims to remove the effect of its corresponding subgroup.

Specifically, let $T_b$ be a $k$-parameter transformation that admits a decomposition into one-parameter transformations:

$$T_b = T_{\theta_1}^{(1)} \circ T_{\theta_2}^{(2)} \circ \cdots \circ T_{\theta_k}^{(k)},$$

where $\theta_i \in \mathbb{R}$. For example, in the case of PGL(3, $\mathbb{R}$), we can decompose an arbitrary transformation into a composition of one-parameter translation, dilation, rotation, shear, and perspective transformations. We then apply a sequence of ET layers in the reverse order of the transformations:

$$L(\phi) = L_{G^{(k)}} \circ L_{G^{(k-1)}} \circ \cdots \circ L_{G^{(1)}}(\phi),$$

where $\rho^{(i)}$ are canonical coordinates for each one-parameter subgroup $G^{(i)}$.

While we can no longer guarantee self-consistency for a composition of ET layers, we show empirically (§5) that this stacking heuristic works well in practice for transformation groups in several parameters.

4.5. Implementation

Here we highlight particularly salient details of our implementation of ETs. Our PyTorch implementation is available at github.com/stanford-futuredata/equivariant-transformers.

Change of Coordinates. We implement coordinate transformations by resampling the input image over a rectangular grid in the new coordinate system. This grid consists of rows and columns that are equally spaced in the intervals $[u_{\min}^{1}, u_{\max}^{1}]$ and $[u_{\min}^{2}, u_{\max}^{2}]$, where the limits of these intervals are chosen to achieve good coverage of the input image. These points $u$ in the canonical coordinate system define a set of sampling points $\rho^{-1}(u)$ in Cartesian coordinates. We use bilinear interpolation for points that do not coincide with pixel locations in the original image, as is typical with ST layers (Jaderberg et al., 2015).

Avoiding Resampling. When using multiple ET layers, iterated resampling of the input image will degrade image quality and amplify the effect of interpolation artifacts. In our implementation, we circumvent this issue by resampling the image lazily. More specifically, let $\phi^{(i)}$ denote the image obtained after $i$ transformations, where $\phi^{(0)}$ is the original input image. At each iteration $i$, we represent $\phi^{(i)}$ implicitly using the sampling grid $\mathcal{G}_i := \left(T_{\theta_i}^{(1)} \circ \cdots \circ T_{\theta_i}^{(i)}\right) \mathcal{G}_0$, where $\mathcal{G}_0$ represents the Cartesian grid over the original input. We materialize $\phi^{(i)}$ (under the appropriate canonical coordinates) in order to predict $\hat{\theta}_{i+1}$. By appending the next predicted transformation $T_{\delta_{i+1}}^{(i+1)}$ to the transformation stack, we thus obtain the subsequent sampling grid, $\mathcal{G}_{i+1}$.

5. Experiments

We evaluate ETs on two image classification datasets: an MNIST variant where the digits are distorted under random projective transformations (§5.1), and the real-world Street View House Numbers (SVHN) dataset (§5.2). Using projectively-transformed MNIST data, we evaluate the performance of ETs relative to STS in a setting where images are deformed by a known transformation group in several parameters. The SVHN task evaluates the utility of ET layers when used in combination with modern CNN architectures in a realistic image classification task. In both cases, we validate the sample efficiency benefits conferred by ETs relative to standard STs and baseline CNN architectures.

5.1. Projective MNIST

We introduce the Projective MNIST dataset, a variant of the MNIST dataset where the digits are distorted using randomly sampled projective transformations: namely rotation, shear, $x$- and $y$-dilation, and $x$- and $y$-perspective transformations (i.e., 6 pose parameters in total). The Projective MNIST training set contains 10,000 base images sampled without replacement from the MNIST training set. Each image is resized to 64 × 64 and transformed using an independently-sampled set of pose parameters.

We also generated three larger versions of the dataset for the purpose of controlled evaluation of the effect of (idealized) data augmentation: these additional datasets respectively contain 2, 4, and 8 copies of the base MNIST images, each transformed under different sets of parameters.

Unlike other MNIST variants such as Rotated MNIST (Larochelle et al., 2007), MNIST-RTS (Jaderberg et al., 2015), and SIM2MNIST (Esteves et al., 2018), our...
Table 1. Classification error rates on Projective MNIST (§5.1).
All methods use the same CNN architecture for classification and differ in the transformations applied to the input images. We train on up to 8 sampled transformations for each base MNIST image. LP: log-polar coordinates; sh \(_x\): x-shear; hr: hyperbolic rotation; p\(_x\): x-perspective; p\(_y\): y-perspective.

| Method   | Transformations | # sampled transformations |
|----------|----------------|--------------------------|
| Cartesian | -              | 11.91 9.67 7.64 6.93      |
| Log-polar | -              | 6.55 5.05 4.48 3.83       |
| ST-LP    | sh\(_x\)       | 5.77 4.27 3.97 3.47       |
| ST-LP    | sh\(_x\) \rightarrow hr | 4.92 3.87 3.22 3.03     |
| ST-LP*   | sh\(_x\) \rightarrow hr \rightarrow p\(_x\) \rightarrow p\(_y\) | - - - -                |
| ET-LP    | sh\(_x\)       | 5.48 4.67 3.63 3.21       |
| ET-LP    | sh\(_x\) \rightarrow hr | 4.18 3.17 2.96 2.62     |
| ET-LP    | sh\(_x\) \rightarrow hr \rightarrow p\(_x\) \rightarrow p\(_y\) | 3.76 3.11 2.80 2.60     |

*We omit this configuration due to training instability.

Projective MNIST dataset incorporates higher-dimensional combinations of transformations, including projective transformations not considered in prior work (e.g., perspective transforms). We provide further details on the construction of the dataset in the Appendix.

Network Architectures. We used a CNN architecture based on the Z2CNN from Cohen & Welling (2016), with 7 layers of \(3 \times 3\) convolutions with 32 channels, batch normalization after convolutional layers, and dropout after the 3rd and 6th layers. In addition to this baseline “Cartesian” CNN, we also evaluated a more rotation- and dilation-robust network where the inputs are first transformed to log-polar coordinates (Henriques & Vedaldi, 2017; Esteves et al., 2018).

We introduce a sequence of transformer layers before the log-polar coordinate transformation to handle the remaining geometric transformations applied to the input. For both the baseline STs and ETs, we apply a sequence of transformer layers, with each layer predicting a single pose parameter. The pose predictor networks in both cases are 3-layer CNNs with 32 channels in each layer. We selected the transformation order, dropout rate, and learning rate schedule based on validation accuracy (see the Appendix for details).

Classification Accuracy (Table 1). We find that the ET layers consistently improve on test error rate over both the log-polar and ST baselines. By accounting for additional transformations, the ET improves on the error rate of the baseline log-polar CNN by 2.79%—a relative improvement of 43%—when trained on a single pose per prototype. Note that we omit the ST baseline with the full transformation sequence due to training instability, despite more extensive hyperparameter tuning than the ET. We find that all methods improve from augmentation with additional poses, with the

ET retaining its advantage but at a reduced margin.

Hyperparameter Sensitivity (Figure 5). We compared the sensitivity of ET and ST networks to the initial learning rate by comparing validation error when training with learning rate values ranging from \(1 \times 10^{-4}\) to \(4 \times 10^{-3}\). For each setting, we trained 10 networks with independent random initializations on Projective MNIST with 10,000 examples, computing the validation error after each epoch and recording the minimum observed error in each run. We find that STs were significantly more sensitive to learning rate than ET, with far higher variance in error rate between runs. This suggests that the self-consistency constraint imposed on ETs helps improve the training-time stability of networks augmented with transformer layers.

5.2. Street View House Numbers (SVHN)

The goal of the single-digit classification task of the SVHN dataset (Netzer et al., 2011) is to classify the digit in the center of \(32 \times 32\) RGB images of house numbers. SVHN is well-suited to evaluating the effect of transformer layers since there is a natural range of geometric variation in the data due to differences in camera position—unlike Projective MNIST, we do not artificially apply further transformations to the data. The training set consists of 73,257 examples; we use a randomly-chosen subset of 5,000 examples for validation and use the remaining 68,257 examples for training. In order to evaluate the data efficiency of each method, we also trained models using smaller subsets of 10,000 and 20,000 examples. The dataset also includes 531,131 additional images that can be used as extra training data; we thus additionally evaluate our methods on the concatenation of this set and the training set.

Network Architectures. We use 10-, 18-, and 34-layer ResNet architectures (He et al., 2016) as baseline networks. Each transformer layer uses a 3-layer CNN with 32 chan-
Table 2. Classification error rates on SVHN (§5.2). For both STs and ETs, we used the following transformations: $x$- and $y$-translation, rotation, and $x$-scaling. Error rates are each averaged over 3 runs. ETs achieve the largest accuracy gains relative to STs and the baseline CNNs in the limited data regime.

| Network   | Transformer   | # training examples |
|-----------|---------------|---------------------|
|           |               | 10k                 |
|           |               | 20k                 |
|           |               | 68k                 |
|           |               | 600k                |
| ResNet-10 | None          | 9.83                |
|           | Spatial       | 7.90                |
|           | Equivariant   | 5.35                |
|           |               | 2.96                |
| ResNet-18 | None          | 9.23                |
|           | Spatial       | 7.31                |
|           | Equivariant   | 4.81                |
|           |               | 2.76                |
| ResNet-34 | None          | 8.73                |
|           | Spatial       | 7.05                |
|           | Equivariant   | 4.67                |
|           |               | 2.53                |

ETs can fail due to boundary effects that occur when image content is cropped after a transformation. This issue can be mitigated by padding the input such that the transformed image does not fall “out of frame”. Even without a strict self-consistency guarantee, we still observe gains when ET layers are used in practice (e.g., in our SVHN experiments).

6. Discussion and Conclusion

Limitations of ETs. The self-consistency guarantee of ETs can fail due to boundary effects that occur when image content is cropped after a transformation. This issue can be mitigated by padding the input such that the transformed image does not fall “out of frame”. Even without a strict self-consistency guarantee, we still observe gains when ET layers are used in practice (e.g., in our SVHN experiments).

Results (Table 2). We find that ETs improve on the error rate achieved by both STs and the baseline ResNets, with the largest gains seen in the limited data regime: with 10,000 examples, ETs improve on the error rates of the baseline CNNs and ST-augmented CNNs by 0.9–1.6%, or a relative improvement of 10–16%. We see smaller gains when more training data is available: the relative improvement between ETs and the baseline CNNs is 11–13% with 20,000 examples, and 6.4–9.5% with 68,257 examples.

When data is limited, we find that a simpler classifier where prior knowledge on geometric invariances has been encoded using ETs can outperform more complex classifiers that are not equipped with this additional structure. In particular, when trained on 10,000 examples, a ResNet-10 classifier with ET layers achieves lower error than the baseline ResNet-34 classifier. The baseline ResNet-34 has over 5.3M parameters; in contrast, the ResNet-10 has 1.2M parameters, with the ET layers adding only 31k parameters in total. The ET-augmented ResNet-10 therefore achieves improved error rate with an architecture that incurs less memory and computational cost than a ResNet-34.

Conclusion. We proposed a neural network layer that builds in prior knowledge on the continuous transformation invariances of its input domain. By encapsulating equivariant functions within an image-to-image mapping, ETs expose a convenient interface for flexible composition of layers tailored to different transformation groups. Empirically, we demonstrated that ETs improve the sample efficiency of CNNs on image classification tasks with latent transformation parameters. Using libraries of ET layers, practitioners are able to quickly experiment with multiple combinations of transformations to realize gains in predictive accuracy, particularly in domains where labeled data is scarce.

As discussed in §4.4, the method of stacking ET layers is ultimately a heuristic approach as it does not guarantee self-consistency with respect to the full transformation group. Moreover, higher-dimensional groups require the use of long sequences of ET layers, resulting in high computational cost. In such cases, we could employ a hybrid approach where “difficult” subgroups are handled by ET layers, while the remaining degrees of freedom are handled by a standard ST layer. In general, enforcing equivariance guarantees for higher-dimensional transformation groups in a computationally scalable fashion remains an open problem.

In contrast to the use of prior knowledge on transformation invariances in this work, there is a separate line of research that concerns learning various classes of transformations from data (Hashimoto et al., 2017; Thomas et al., 2018). Extending ETs to these more flexible notions of invariance may prove to be an interesting direction for future work.
Acknowledgements

We thank Pratiksha Thaker, Kexin Rong, and our anonymous reviewers for their valuable feedback on earlier versions of this manuscript. This research was supported in part by affiliate members and other supporters of the Stanford DAWN project—Ant Financial, Facebook, Google, Intel, Microsoft, NEC, SAP, Teradata, and VMware—as well as Toyota Research Institute, Keysight Technologies, Northrop Grumman, Hitachi, NSF awards AF-1813049 and CCF-1704417, an ONR Young Investigator Award N00014-18-1-2295, and Department of Energy award de-sc0019205.

References

Amari, S. Feature Spaces which Admit and Detect Invariant Signal Transformations. In International Joint Conference on Pattern Recognition, 1978.

Amit, Y., Grenander, U., and Piccioni, M. Structural image restoration through deformable templates. Journal of the American Statistical Association, 1991.

Cohen, T. S. and Welling, M. Group Equivariant Convolutional Networks. In International Conference on Machine Learning, 2016.

Cohen, T. S., Geiger, M., Köhler, J., and Welling, M. Spherical CNNs. In International Conference on Learning Representations, 2018.

De Castro, E. and Morandi, C. Registration of translated and rotated images using finite Fourier transforms. IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 700–703, 1987.

Dieleman, S., De Fauw, J., and Kavukcuoglu, K. Exploiting cyclic symmetry in convolutional neural networks. International Conference on Machine Learning, 2016.

Esteves, C., Allen-Blanchette, C., Zhou, X., and Daniilidis, K. Polar Transformer Networks. In International Conference on Learning Representations, 2018.

Freeman, W. T. and Adelson, E. H. The design and use of steerable filters. IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 891–906, 1991.

Gens, R. and Domingos, P. M. Deep Symmetry Networks. In Advances in Neural Information Processing Systems, 2014.

Hashimoto, T. B., Liang, P. S., and Duchi, J. C. Unsupervised transformation learning via convex relaxations. In Advances in Neural Information Processing Systems, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2016.

Henriques, J. F. and Vedaldi, A. Warped Convolutions: Efficient Invariance to Spatial Transformations. In International Conference on Machine Learning, 2017.

Jacobsen, J.-H., De Brabandere, B., and Smeulders, A. W. Dynamic Steerable Blocks in Deep Residual Networks. arXiv preprint arXiv:1706.00598, 2017.

Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. Spatial Transformer Networks. In Advances in Neural Information Processing Systems, 2015.

Laptev, D., Savinov, N., Buhmann, J. M., and Pollefeys, M. TI-POOLING: transformation-invariant pooling for feature learning in convolutional neural networks. In IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2016.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. An empirical evaluation of deep architectures on problems with many factors of variation. In International Conference on Machine Learning, 2007.

Lin, C.-H. and Lucey, S. Inverse Compositional Spatial Transformer Networks. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2017.

Lucas, B. D. and Kanade, T. An iterative image registration technique with an application to stereo vision. In International Joint Conference on Artificial intelligence, 1981.

Marcos, D., Volpi, M., Komodakis, N., and Tuia, D. Rotation Equivariant Vector Field Networks. In International Conference on Computer Vision, 2017.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In NIPS workshop on deep learning and unsupervised feature learning, 2011.

Rawlinson, D., Ahmed, A., and Kowadlo, G. Sparse unsupervised capsules generalize better. arXiv preprint arXiv:1804.06094, 2018.

Reddi, S. J., Kale, S., and Kumar, S. On the Convergence of Adam and Beyond. In International Conference on Learning Representations, 2018.

Reddy, B. S. and Chatterji, B. N. An FFT-based technique for translation, rotation, and scale-invariant image registration. IEEE Transactions on Image Processing, 5(8): 1266–1271, 1996.
Rubinstein, J., Segman, J., and Zeevi, Y. Recognition of distorted patterns by invariance kernels. *Pattern Recognition*, 24(10):959–967, 1991.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, 2017.

Segman, J., Rubinstein, J., and Zeevi, Y. Y. The canonical coordinates method for pattern deformation: Theoretical and computational considerations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1171–1183, 1992.

Shu, Z., Sahasrabudhe, M., Alp Guler, R., Samaras, D., Paragios, N., and Kokkinos, I. Deforming Autoencoders: Unsupervised Disentangling of Shape and Appearance. In *European Conference on Computer Vision (ECCV)*, 2018.

Strauss, W. A. *Partial Differential Equations: An Introduction*. Wiley, 2007.

Teo, P. C. *Theory and Applications of Steerable Functions*. PhD thesis, Stanford University, 1998.

Thomas, A., Gu, A., Dao, T., Rudra, A., and Ré, C. Learning compressed transforms with low displacement rank. In *Advances in Neural Information Processing Systems*, 2018.

Weiler, M., Hamprecht, F. A., and Storath, M. Learning Steerable Filters for Rotation Equivariant CNNs. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Harmonic Networks: Deep Translation and Rotation Equivariance. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Yuille, A. L. Deformable templates for face recognition. *Journal of Cognitive Neuroscience*, 1991.

Zhou, Y., Ye, Q., Qiu, Q., and Jiao, J. Oriented response networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.
A. Proof of Proposition 1

Proposition 2. Let \( f : \Phi \rightarrow \mathbb{R}^k \) be self-consistent with respect to translation and let \( \rho \) be a canonical coordinate system with respect to a transformation group \( G \) parameterized by \( \theta \in \mathbb{R}^k \). Then \( f_\rho(\phi) := f(\phi \circ \rho^{-1}) \) is self-consistent with respect to \( G \).

Proof. By the definition of \( \rho \),

\[
(r \circ T_\theta \circ r^{-1})(u) = r(r^{-1}(u)) + \sum_{i=1}^{k} \theta_i e_i
\]

\[
= u + \sum_{i=1}^{k} \theta_i e_i
\]

and therefore \( (T_\theta \circ r^{-1})(u) = r^{-1} \left( u + \sum_{i=1}^{k} \theta_i e_i \right) \). By this identity and translation self-consistency of \( f \),

\[
f_\rho(T_\theta \phi) = f((T_\theta \phi \circ r^{-1})(u))
\]

\[
= f((\phi \circ T_\theta \circ r^{-1})(u))
\]

\[
= f(\phi \circ r^{-1}) \left( u + \sum_{i=1}^{k} \theta_i e_i \right)
\]

\[
= f(\phi \circ r^{-1})(u) + \theta
\]

where in the second line we used the definition of \( T_\theta \phi \), in the third line we used the identity for \( (T_\theta \circ r^{-1})(u) \), and in the fourth line we used the translation self-consistency of \( f \). This establishes self-consistency with respect to \( G \).

B. Canonical Coordinate Systems

In Table 3, we list the set of canonical coordinates that we derived for our experiments along with their corresponding transformation groups. As explained in the main text, these coordinates are not unique for one-parameter transformation groups: in this case, there exists a degree of freedom in specifying the complementary set of coordinates.

In Figure 7, we plot some examples of canonical coordinate grids used in our experiments.

C. Experimental Details

C.1. Projective MNIST

Dataset. To construct the Projective MNIST dataset, we sampled 10,000 images without replacement from the MNIST training set (consisting of 60,000 examples). Each \( 28 \times 28 \) base image is extended to \( 64 \times 64 \) by symmetric zero padding. The images are then distorted using transformations sampled independently from the projective group. The 6 pose parameters were sampled uniformly from the ranges listed in Table 4. We excluded translation from this combination of transformations in order to avoid cropping issues due to the distorted digit exceeding the boundaries of the image. We selected these pose parameter ranges in order to evaluate performance on a more challenging set of transformations than those evaluated in Jaderberg et al. (2015), using a smaller training set in line with the popular Rotated MNIST dataset (Larochelle et al., 2007).

For each base image, we independently sampled 8 sets of pose parameters, thus yielding 8 transformed versions of each base image. We created 4 training sets with 10,000, 20,000, 40,000 and 80,000 examples respectively, each containing 1, 2, 4, or 8 versions of each base image.

In addition to the training sets, we generated a validation set of size 5000 using examples from the MNIST training set that were not used in the Projective MNIST training set. Each of the images in the validation set was transformed using an independently sampled set of pose parameters sampled from the same range as the training set.
Equivariant Transformer Networks

Figure 7. **Examples of canonical coordinate systems.** The corresponding images of \((x, y)\) for each transformation are (1a) rotation, (1b) dilation, (2) x-shear, (3) hyperbolic rotation, and (4) x-perspective.

We generated a test set using all 10,000 images in the MNIST test set. For each base image, we sampled 8 sets of pose parameters, thus yielding a test set of size 80,000 that contains 8 transformed versions of each base test.

**Preprocessing.** We preprocessed the data by subtracting the mean pixel value and dividing by the standard deviation. The mean and standard deviation were computed as scalar values, ignoring pixel locations.

**Network Architectures.** We used a baseline CNN architecture similar to the Z2CNN used in the experimental evaluation of Cohen & Welling (2016). In all our experiments, each convolutional layer is followed by a spatial batch normalization layer. This network has the following architecture, with output shapes listed in (channel \(\times\) height \(\times\) width format):

| Layer   | Output Shape |
|---------|--------------|
| input   | 1 \(\times\) 64 \(\times\) 64 |
| conv1   | 32 \(\times\) 62 \(\times\) 62 |
| avgpool | 32 \(\times\) 31 \(\times\) 31 |
| conv2   | 32 \(\times\) 29 \(\times\) 29 |
| avgpool | 32 \(\times\) 14 \(\times\) 14 |
| conv3   | 32 \(\times\) 12 \(\times\) 12 |
| dropout | 32 \(\times\) 12 \(\times\) 12 |
| conv4   | 32 \(\times\) 10 \(\times\) 10 |
| conv5   | 32 \(\times\) 8 \(\times\) 8 |
| conv6   | 32 \(\times\) 6 \(\times\) 6 |
| dropout | 32 \(\times\) 6 \(\times\) 6 |
| conv7   | 10 \(\times\) 4 \(\times\) 4 |
| maxpool | 10 |

For self-consistent pose prediction within ET layers, we used the following architecture:
Equivariant Transformer Networks

| Layer       | Output Shape |
|-------------|--------------|
| input       | $1 \times 64 \times 64$ |
| canon. coords. | $1 \times 64 \times 64$ |
| conv1       | $32 \times 32 \times 32$ |
| conv2       | $32 \times 32 \times 32$ |
| maxpool     | $32 \times 32$ |
| conv3       | $1 \times 32$ |
| softmax     | $1 \times 32$ |
| centroid    | $1$ |

In this network, the max-pooling layer eliminates the extraneous dimension in the feature map. For example, when predicting rotation angle using polar coordinates, this operator pools over the radial dimension; the remaining feature map is then indexed by the angular coordinate. We then pass this feature map through a softmax operator to obtain a spatial distribution, and finally compute the centroid of this distribution to obtain the predicted pose. When a pair of pose predictions are needed (e.g., when predicting rotation and dilation parameters simultaneously), we use two output branches that each pools over a different dimension.

For baseline, non-equivariant pose prediction within ST layers, we used the following architecture:

| Layer       | Output Shape |
|-------------|--------------|
| input       | $1 \times 64 \times 64$ |
| conv1       | $32 \times 32 \times 32$ |
| conv2       | $32 \times 32 \times 32$ |
| maxpool     | $32 \times 3 \times 3$ |
| fc          | $1$ |

Unlike the self-consistent pose predictor, this network does not represent the input using a canonical coordinate system. The pose is predicted using a fully-connected layer.

Hyperparameters. We tuned the set of ET layers, their order, the dropout probability, the initial learning rate and the learning rate decay factor on the validation set. ET layers were selected from subgroups of the projective group: rotation, dilation, hyperbolic rotation, $x$-shear, $x$-perspective and $y$-perspective. The dropout probability was selected from the set $\{0.25, 0.30, 0.35, 0.40\}$. We computed validation accuracy after each epoch of training and computed test accuracy using the network that achieved the best validation accuracy. For optimization, we used the AMSGrad algorithm (Reddi et al., 2018) with a minibatch size of 128. For the baseline and ET networks, we used an initial learning rate of $2 \times 10^{-3}$. For the ST networks, we used an initial learning rate of $2 \times 10^{-4}$; higher learning rates resulted in unstable training and hence reduced accuracy. We multiplicatively decayed the learning rate by 1% after each epoch. We trained all our networks for 300 epochs.

C.2. Street View House Numbers

Preprocessing. For each channel, we preprocessed the data by subtracting the mean pixel value and dividing by the standard deviation. The mean and standard deviation were computed as scalar values, ignoring pixel locations.

Network Architectures. We used standard ResNet architectures as baseline CNNs (He et al., 2016). For pose prediction, we used the same CNN architectures as in the Projective MNIST task, but with 3 input channels.

Hyperparameters. As with the Projective MNIST experiments, we tuned the set of ET layers, their order, the dropout probability, the initial learning rate and the learning rate decay factor on the validation set (a 5000-example subset of the SVHN training set). We tuned the transformation and dropout hyperparameters over the same set of possible values as for Projective MNIST. Again, we used the AMSGrad algorithm for optimization with a minibatch size of 128. Due to the large size ($\approx 600k$ examples) of the training set with the addition of the extra training images, we only trained our networks for 150 epochs in this setting. For the remaining runs, we trained for 300 epochs.
Equi\textbf{v}arient Transformer Networks

Table 3. Canonical coordinate systems implemented for our experiments with their corresponding transformation groups.

| Transformation    | $T_{\theta}x$                       | $\rho_1(x)$ | $\rho_2(x)$ |
|-------------------|-------------------------------------|-------------|-------------|
| $x$-translation    | $(x_1 + \theta, x_2)$               | $x_1$       | $x_2$       |
| $y$-translation    | $(x_1, x_2 + \theta)$               |             |             |
| Rotation Dilation  | $(x_1 \cos \theta - x_2 \sin \theta, x_1 \sin \theta + x_2 \cos \theta)$ | $\tan^{-1}(x_2/x_1)$ | $\log \sqrt{x_1^2 + x_2^2}$ |
| Dilation          | $(x_1 e^\theta, x_2 e^\theta)$     | $\log x_1$  | $\log x_2$  |
| Hyperbolic Rotation| $(x_1 e^\theta, x_2 e^{-\theta})$  | $\log x_1/x_2$ | $x_1x_2$     |
| $x$-shear          | $(x_1 - \theta x_2, x_2)$          | $-x_1/x_2$  | $x_2$       |
| $y$-shear          | $(x_1, x_2 + \theta x_1)$          | $x_2/x_1$   | $x_1$       |
| $x$-perspective    | $(x_1/(\theta x_1 + 1), x_2/(\theta x_1 + 1))$ | $1/x_1$     | $\tan^{-1}(x_2/x_1)$ |
| $y$-perspective    | $(x_1/(\theta x_2 + 1), x_2/(\theta x_2 + 1))$ | $1/x_2$     | $\tan^{-1}(x_1/x_2)$ |

Table 4. Ranges of sampled transformations for Projective MNIST.

| Transformation    | $T_{\theta}x$                       | Range       |
|-------------------|-------------------------------------|-------------|
| Rotation Dilation  | $(x_1 \cos \theta - x_2 \sin \theta, x_1 \sin \theta + x_2 \cos \theta)$ | $[-\pi, \pi]$     |
| Hyperbolic Rotation| $(x_1 e^\theta, x_2 e^\theta)$     | $[0, \log 2]$ |
| $x$-shear          | $(x_1 - \theta x_2, x_2)$          | $[-\log 1.5, \log 1.5]$ |
| $x$-perspective    | $(x_1/(\theta x_1 + 1), x_2/(\theta x_1 + 1))$ | $[-1.5, 1.5]$ |
| $y$-perspective    | $(x_1/(\theta x_2 + 1), x_2/(\theta x_2 + 1))$ | $\{(\theta_x, \theta_y) : |\theta_x| + |\theta_y| \leq 0.8\}$ |

D. Additional Experiments

D.1. Robustness to Unseen Transformations

In this experiment, we evaluate the robustness of CNNs with ET layers to transformations not seen at training time. Following the procedure of Sabour et al. (2017), we train on a variant of the MNIST training set where each digit is randomly placed on a 40 × 40 black background. This network is then tested on the affNIST test set, a variant of the MNIST test set where each digit is transformed with a small affine transformation. We perform model selection against a validation set of 5000 held-out MNIST training images, each randomly placed on the 40 × 40 background but subject to no further transformations.

Our CNN baseline uses three convolutional layers with 256, 256 and 128 channels, each with 5 × 5 kernels and a stride of 1. Each convolutional layer is followed by a batch normalization layer and a ReLU nonlinearity. The output of the final convolutional layer is average pooled to obtain a 128-dimensional embedding which is mapped to 10 classes by a fully-connected layer. We use dropout before the final classification layer.

We evaluate this CNN architecture with three ET configurations: (1) $x$- and $y$-translation followed by a transformation to log-polar coordinates, (2) $x$- and $y$-translation, followed by $x$-shear, followed by a transformation to log-polar coordinates, and finally (3) $x$- and $y$-translation followed by rotation/scale, without a further log-polar transformation.

Table 5 summarizes our results. We find that our baseline CNN already outperforms the Capsule Network architecture from Sabour et al. (2017), while the baseline CNN over log-polar coordinates (without any ET layers) performs poorly due to the

---

Footnotes:

1 The affNIST dataset can be found at https://www.cs.toronto.edu/~tijmen/affNIST/. The transformation parameters are sampled uniformly within the following ranges: rotation in $[-20, 20]$ degrees, shear in $[-0.2, 0.2]$, vertical and horizontal scaling in $[0.8, 1.2]$. The transformed digit is placed uniformly at random on a 40 × 40 black background, subject to the constraint that no nonzero part of the digit image is cropped.

2 This first ET configuration is equivalent to the Polar Transformer architecture introduced by Esteves et al. (2018).
Table 5. Test accuracies on affNIST. All models except (*) were trained only on randomly-translated MNIST training data and tested on affNIST images, which are MNIST test images distorted by random affine transformations. The classification layer for (*) was trained on affNIST data using a feature extractor that was trained on MNIST.

| Method                                      | affNIST test accuracy (%) |
|---------------------------------------------|---------------------------|
| Standard CNN (Sabour et al., 2017)          | 66                        |
| Standard CNN (ours)                         | 88.3                      |
| Capsule Network (Sabour et al., 2017)       | 79                        |
| Sparse Unsupervised Capsule features + SVM (Rawlinson et al., 2018) | 90.1*                    |
| Log-polar                                   | 76.6                      |
| ET-LP (translation)                         | 98.1                      |
| ET-LP (translation + x-shear)               | 98.3                      |
| ET-Cartesian (translation + rotation/scale)  | 98.2                      |

loss of translation equivariance. The ET-augmented CNNs improve on these results, with both networks demonstrating comparable robustness to affine transformations not seen at training time. The higher test accuracies achieved by the ET networks relative to the Capsule Network baselines reflect the stronger priors that have been built into the ET architecture—in contrast to the ET networks, the Capsule Network baselines have to learn invariances from the training data.