Learning to Transform for Generalizable Instance-wise Invariance

Utkarsh Singhal[1] Carlos Esteves[3] Ameesh Makadia[3] Stella X. Yu[1,2]
[1] UC Berkeley [2] University of Michigan [3] Google Research

Abstract

Computer vision research has long aimed to build systems that are robust to spatial transformations found in natural data. Traditionally, this is done using data augmentation or hard-coding invariances into the architecture. However, too much or too little invariance can hurt, and the correct amount is unknown a priori and dependent on the instance. Ideally, the appropriate invariance would be learned from data and inferred at test-time.

We treat invariance as a prediction problem. Given any image, we use a normalizing flow to predict a distribution over transformations and average the predictions over them. Since this distribution only depends on the instance, we can align instances before classifying them and generalize invariance across classes. The same distribution can also be used to adapt to out-of-distribution poses. This normalizing flow is trained end-to-end and can learn a much larger range of transformations than Augerino and InstaAug. When used as data augmentation, our method shows accuracy and robustness gains on CIFAR 10, CIFAR10-LT, and TinyImageNet.

1. Introduction

One of the most impressive abilities of the human visual system is its robustness to geometric transformations. Objects in the visual world often undergo rotation, translation, etc., producing many variations in the observed image. Nonetheless, we classify them reliably and efficiently.

Any robust classifier must encode information about the expected geometric transformations, either explicitly (e.g., through architecture) or implicitly (e.g., invariant features). What would this knowledge look like for humans?

Scientists have extensively investigated this question [1]. We know that it generalizes to novel (but similar) categories, e.g., we can instantly recognize a new symbol from many poses after seeing it just once [2]. For unfamiliar categories or poses, we can learn the invariance over time [3]. Finally, while we quickly recognize objects in typical poses, we can also adapt to "out-of-distribution" poses with processes like mental rotation [4]. These properties help us robustly handle novel categories and novel poses (Figure 1).

In contrast, modern classifiers based on deep learning are brittle [5]. While these methods have achieved super-human accuracy on curated datasets like ImageNet [6], they are unreliable in the real world [7], showing poor generalization and even causing fatal outcomes in systems relying on computer vision [8]. Thus, robust classification has long been an aim of computer vision research [5, 9]. This paper asks:

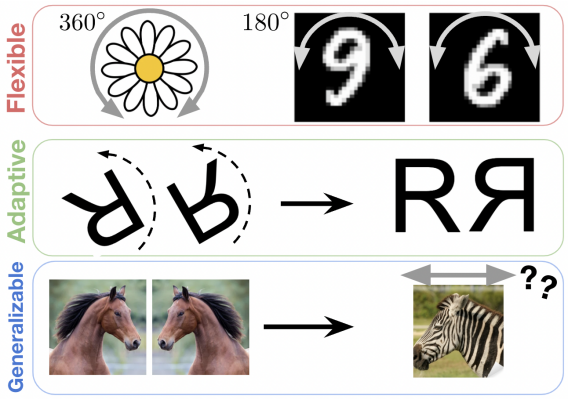

Figure 1: Our goal is to build flexible, adaptive, and generalizable invariances. **Flexible:** The ideal invariance is flexible and instance-dependent. Different objects in different poses require different degrees of invariance. Too much hurts accuracy, and too little hurts robustness. **Adaptive**: The model should adapt to unexpected (out-of-distribution) poses. The figure above shows mental rotation, a process by which humans align unfamiliar objects in unexpected poses to classify them. **Generalizable**: Knowledge of invariances should generalize from previous experience, e.g., learning bilateral symmetry for horses and transferring it to zebras.

Can we replicate this flexible, generalizable, and adaptive invariance in artificial neural networks?

For some transformations (e.g., translation), the invariance can be hard-coded into the architecture. This insight has led to important approaches like Convolutional Neural Networks [10, 11]. However, this approach imposes severe architecture restrictions and thus has limited applicability.

An alternative approach to robustness is data augmentation [12]. Input data is transformed through a predefined set of transformations, and the neural network learns to perform the task reliably despite these transformations. Its success

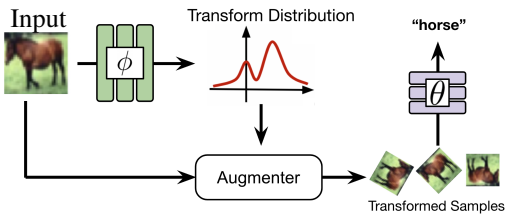

Figure 2: Our image classification pipeline. The normalizing flow model predicts a distribution over image transformations. Samples from this distribution are passed to a differentiable augmented, which transforms the input image into a set of augmented images. The images are passed to a classifier, and predictions are averaged. Crucially, the transform distribution g_ϕ can generalize across classes and datasets.

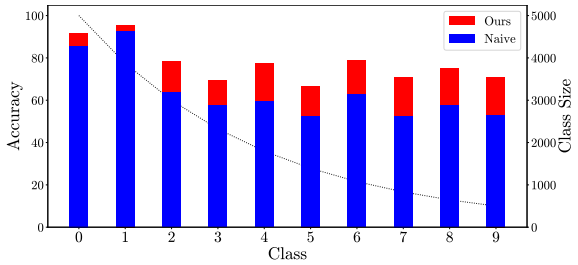

Figure 3: Our method delivers strong gains for imbalanced classification. On CIFAR10-LT with 5000 to 500 instances per class from head to tail (black curve), our class-agnostic instance-wise transform distribution helps boost the classification accuracy by large margins (red bars) over the standard softmax baseline (blue bars) especially for the tail classes.

and wide applicability have made it ubiquitous in deep learning. However, data augmentation is unreliable since the learned invariance breaks under distribution shifts and fails to transfer from head classes to tail classes in imbalanced classification settings [13].

Both these approaches *prescribe* the invariances while assuming a known set of transformations. However, the correct set of invariances is often unknown *a priori*, and a mismatch can be harmful [14, 15, 12]. For instance, in fine-grained visual recognition, rotation invariance can help with flower categories but hurt animal recognition [16].

A recent line of methods [14, 17, 15] aims to *learn* the useful invariances. Augerino [14] learns a range of transformations shared across the entire dataset, producing better generalizing models. However, these methods use a fixed range of transformations for all inputs, thus failing to be flexible. InstaAug [15] learns an instance-specific augmentation range for each transformation, achieving higher accuracy on datasets such as TinyImageNet due to its flexibility. However, since InstaAug learns a range for each parameter separately, it cannot represent multi-modal or joint distributions (e.g., it cannot discover rotations from the set of all affine matrices). Additionally, these approaches don't explore generalization across classes and adaptation to unexpected poses (Figure 1).

We take inspiration from Learned-Miller *et al.* [2] and model the relationship between the observed image and its class as a graphical model (Figures 2 and 5). We also represent the instance-wise distribution of transformations using a normalizing flow and apply it to robust classification. Our experiments show that the properties like adaptability and generalizability emerge naturally in this framework.

Contributions: **(1)** We propose a normalizing flow model to learn the image-conditional transformation distribution. **(2)** Our model can represent multi-modal and joint distributions over transformations, being able to model more complex invariances, and **(3)** helps achieve higher test accuracy on datasets such as CIFAR10, CIFAR10-LongTail (Figure 3), and TinyImageNet. Finally, **(4)** combined with our graphical model, this model forms a flexible, generalizable, and adaptive form of invariance. It can be used to **(a)** align the dataset and discover prototypes like congealing [2], **(b)** adapt to unexpected poses like mental rotation [3], and **(c)** transfer invariance across classes like GAN-based methods [13].

2. Related Work

Mental rotation in humans: Shepard and Metzler [4] were among the first to measure the amount of time taken by humans to recognize a rotated object. They found that the response time increased linearly with rotation, suggesting a dynamic process like mental rotation for recognizing objects in unfamiliar poses. Tarr and Pinker [1] further study mental rotation as a theory of invariant object recognition, contrasting it against invariant features and a multiple-view theory. Cooper and Shepard [18] found that revealing identity and orientation information beforehand helped the subjects make constant-time predictions. Hock and Tromley [19] found that the recognition time is nearly constant for characters perceived as "upright" over a large range of rotations. However, outside that range (and for characters with narrow "upright" ranges), the recognition time follows the same linear relationship, indicating mental rotation is needed when the object is detected as "not upright." Koriat and Norman [3] investigated mental rotation as a function of familiarity, finding that humans adapt to unfamiliar objects with practice, gaining robustness to small rotations around the upright pose. The response curve thus becomes flatter around the upright pose. These works suggest a flexible, adaptive, and general form of robustness in the human vision.

Invariance in Neural Networks: Neural networks invariant to natural transformations have long been a central goal in deep learning research [9]. Bouchacourt *et al.* [12] and Madan *et al.* [5] studied the invariances present in modern models. One of the earliest successes includes architectures like Convolutional Neural Networks [10, 11], and more recently, applications such as medical image analysis [20, 21, 22], cosmology [23, 24], and

physics/chemistry [25, 26, 27]. Kondor and Trivedi [28] and Cohen *et al*. [29] established a general theory of equivariant neural networks based on representation theory. Finzi *et al*. [30] combined equivariant and non-equivariant blocks through a residual connection.

The dominant way to add invariance into neural networks is data augmentation. Dao *et al*. [31] shows that to a first-order approximation, data augmentation is equivalent to averaging features over transformations. Bouchacourt *et al*. [12] found data augmentation to be crucial for invariance in many modern computer vision architectures. Zhou *et al*. [13] demonstrated a key failing of data augmentation in imbalanced classification and used a GAN to generate a broad set of variants for every instance. Our method is complementary to theirs and can be combined in future work. We also note that the experiments in this paper only use affine image transformations and yet achieve comparable accuracy to theirs on CIFAR10LT. Congealing [2] aligns all the images in a class, simultaneously producing a prototype and inferring the relative pose of each example. The aligned dataset can be used for robust recognition, and the learned pose distribution can be used for new classes. However, this method assumes the transformation distribution is class-wise, whereas we model it for every instance. Learned canonicalization [32] learns an energy function that is minimized at test time to align the input to a canonical orientation. Spatial Transformer Networks [33] predict a transformation from the input image in an attempt to rectify it and improve classification accuracy. However, STNs cannot represent a distribution of transformations. Probabilistic Spatial Transformer Networks [34] model the conditional distribution using a Gaussian distribution with mean and variance predicted by a neural network. In contrast, we use a normalizing flow model. We also study the generalizability as well as adaptation to unexpected poses.

Augerino: [14] aims to learn the ideal range of invariances for any given dataset. It uses the reparametrization trick and learns the range of uniform distribution over each transformation parameter separately (e.g., range of translations, rotations, etc.). This ability allows Augerino to learn the useful range of augmentations (and thus invariances) directly and produce more robust models with higher generalization. However, Augerino is sensitive to the regularization amount and the parametrization of the augmentation range (Table 3). LILA [17] tackles this problem using marginal likelihood methods. However, for both Augerino and LILA, the resulting invariance is shared among all classes, even though different classes (such as 0 and 6 in a digit classification setting) may have entirely different ideal augmentation distributions. Figure 4 illustrates how these limitations lead Augerino to learn an overly restricted augmentation range.

InstaAug: [15] fixes the inflexibility of Augerino by predicting the augmentation ranges for every instance and

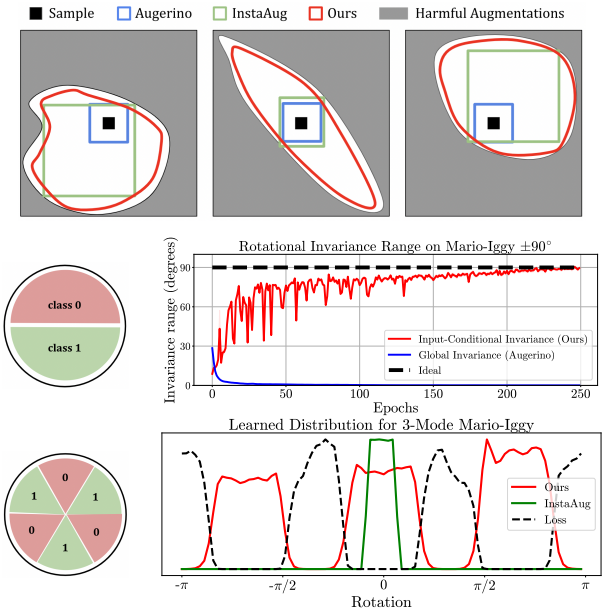

Figure 4: Our normalizing flow model can represent input-dependent, multi-modal, and joint distributions over augmentation parameters. **(top)** We illustrate three samples, each with a different set of correct augmentations. Augerino learns a range shared between all samples, so the learned range is too restrictive. InstaAug learns an instance-wise range but cannot handle a non-axis-aligned augmentation set (middle). In contrast, our model can adapt to the loss landscape and produce the largest possible set. **(middle)** Augerino [14] fails to learn augmentations in challenging settings. Learned rotation range for a version of Mario-Iggy with $\pm 90^\circ$ rotation range. The class boundaries touch each other, so some instances lie close to the boundary, and thus, global augmentation schemes like [14, 17] are forced to learn a range of 0. Our method learns the correct range. **(bottom)** InstaAug fails to capture the distribution for a multi-modal version of the Mario-Iggy dataset.

provides a theoretical argument connecting it to generalization error. In our knowledge, it is the first work to do so. This allows for larger effective ranges and, thus, impressive generalization gains in image classification and contrastive learning settings. However, while InstaAug is instance-wise, it models the range of each parameter separately (the *mean-field* assumption). Thus, it cannot represent multi-modal or joint distributions (Figure 4). Like Augerino, the representational limitations greatly limit the set of learnable transformations, especially for complex augmentation classes like image cropping [15], necessitating tricks like selecting among a pre-defined set of crops. Furthermore, InstaAug is sensitive to parametrization (see Figure 7 and Table 3).

3. Methods

We begin by describing our probabilistic model. We derive its inference equation and training loss and compare it to existing methods. We then construct a normalizing flow model to represent the conditional transform distribution. We also derive an analytical expression for the model's approximate invariance. Finally, we describe the mean-shift algorithm for adapting to out-of-distribution poses.

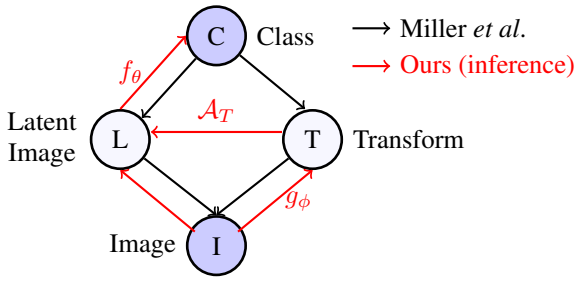

Figure 5: Our graphical model inspired by Miller *et al.* [2]. Shaded nodes represent variables observed in data (C, I). In contrast to Miller *et al.*, we only model the inference process and assume that T is instance-wise, not classwise. Our flow model g_ϕ predicts image-conditional transform, and the classifier f_θ classifies the resulting image L.

Graphical model: We follow the model described in Figure 5. Here, C refers to the class, I refers to the observed image, L refers to the latent image (equivalent to the prototype in [2]), and T refers to the unobserved transformation parameters connecting the latent image and the observed image. The latent image is produced by passing the pair (I, T) through a differentiable augmenter $\mathcal{A}$, which applies the transform to the observed image, i.e., $L = \mathcal{A}_T(I)$.

One notable difference to Miller *et al.* [2] is that our distribution is instance-wise (similar to [15]), not class-wise. This allows for a more general conditional distribution model.

Given the values C, L, T, I, the model defines a joint probability distribution $P(C, L, T, I)$:

$$P(C, L, T, I) = P(C|L)P(L|T, I)P(T|I)P(I) \quad (1)$$

and the conditional class probability $P(C|I)$ as:

$$P(C|I) = \int_{L,T} P(T|I)P(L|T, I)P(C|L)dLdT \quad (2)$$

Since $L = \mathcal{A}_T(I)$, this can be further simplified to:

$$P(C|I) = \int_T P(T|I)P(C|L = \mathcal{A}_T(I))dT \quad (3)$$

$$= \mathbb{E}_{T\sim P(T|I)}\big[P(C|L = \mathcal{A}_T(I))\big] \quad (4)$$

Thus, the predicted class probability is averaged over transformations sampled from the conditional transform distribution $P(T|I)$. This is analogous to the idea of "test-time augmentations" used in image classification literature. Augerino assumes that the transformation T is independent of I. InstaAug models T as a uniform distribution conditioned on I. PSTN [34] arrives at the same expression and uses a Gaussian distribution. All these frameworks can be viewed as different approximations in this formulation. However, we also analyze the invariance properties of this formulation and applications of $P(T|I)$.

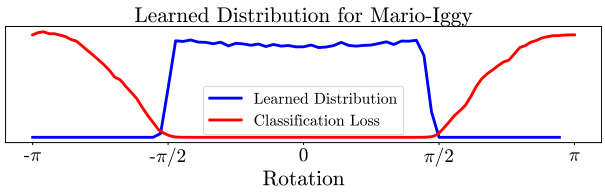

Figure 6: The ideal learned distribution maximizes the range while minimizing the overall classification loss

Neural network approximation: We approximate each of the key distributions $P(C|L)$ and $P(T|I)$ with neural networks. Our $f_\theta(C; L)$ is a simple classifier, and $g_\phi(T; I)$ is a normalizing flow model [35] which takes in the image I:

$$f_\theta(C; L) \approx P(C|L), \quad g_\phi(T; I) \approx P(T|I) \quad (5)$$

Since $L = \mathcal{A}_T(I)$, we use $f_\theta(C; L)$, $f_\theta(C; T, I)$ and $f_\theta(C; \mathcal{A}_T(I))$ interchangeably.

Inference: The expression for $P(C|I)$ then becomes:

$$p_{\theta,\phi}(C|I) = \int_T g_\phi(T; I)f_\theta(C; \mathcal{A}_T(I))dT \quad (6)$$

$$= \mathbb{E}_{T\sim g_\phi(T;I)}\big[f_\theta(C; \mathcal{A}_T(I))\big] \quad (7)$$

This equation describes the act of sampling transformations from the normalizing flow model and averaging the classifier predictions over the sampled transformations.

Classifier loss: During training, we observe (I, C) pairs. We train the classifier f_θ by maximizing a lower bound to the average $\log p_{\theta,\phi}(C|I)$. It is common to use Jensen's inequality to make this tractable:

$$\log p_{\theta,\phi}(C|I) \geq \mathbb{E}_{T\sim g_\phi(T;I)}\big[\log f_\theta(C; \mathcal{A}_T(I))\big] \quad (8)$$

and maximize the resulting lower bound instead. This further reduces to the loss function $\mathcal{L}_{\text{classifier}}$:

$$\mathcal{L}_{\text{classifier}} = \mathbb{E}_{T\sim g_\phi(T;I)}\big[-\log f_\theta(C; \mathcal{A}_T(I))\big] \quad (9)$$

which is simply the cross-entropy loss averaged over sampled augmentations.

Augmenter loss: Intuitively, we would like the transform distribution g_ϕ to have a large diversity of augmentations and minimal classification loss (see Figure 6). However, in practice, minimizing the classification loss leads to g_ϕ collapsing to a single peak (0-variance distribution) as the model overfits to the training data (as observed in Augerino [14] without regularization).

Since our normalizing flow model already produces log probability for each generated sample, *entropy regularization*

is a natural match for our method. We penalize the average $\log g_\phi$ for sampled transformations:

$$\mathcal{L}_{\text{augmenter}} = \mathcal{L}_{\text{classifier}} + \alpha \mathbb{E}_{T \sim g_\phi} \left[\log g_\phi(T; I) \right] \quad (10)$$

This regularization is a generalization of the one used by Augerino, since for uniform distributions, $\log p \propto -\log(\text{width})$. InstaAug derives a similar expression as a Lagrange relaxation of entropy constraints and applies it to simple distributions like uniform and categorical.

We apply it to normalizing flows, which can model more general distributions, and our graphical model helps us understand this loss and connect it to variational inference.

Understanding entropy regularization: Here, we analyze the form of the distribution learned through entropy regularization. Consider the following loss:

$$\mathcal{L}_{\text{augmenter}}[g_\phi] = \mathcal{L}_{\text{classifier}}[g_\phi] - \alpha \mathbb{H}[g_\phi] \quad (11)$$

where $\alpha \in \mathbb{R}^+$ is a regularization constant and $\mathbb{H}[g_\phi]$ is the entropy of the distribution g_ϕ. This expression reduces to:

$$= \mathbb{E}_{T \sim g_\phi(T;I)} \left[\alpha \log g_\phi(T; I) - \log f_\theta(C; \mathcal{A}_T(I)) \right]$$

We rescale this loss by $\lambda = \frac{1}{\alpha}$ to simplify:

$$\equiv \mathbb{E}_{T \sim g_\phi(T;I)} \left[\log g_\phi(T; I) - \lambda \log f_\theta(C; \mathcal{A}_T(I)) \right]$$

Note that this loss is equivalent to a KL-divergence between g_ϕ and a special target distribution $\tilde{p}^\lambda_\theta(T|C, I)$:

$$\mathcal{L}_{\text{augmenter}}[g_\phi] = \text{KL} \left[g_\phi(T; I) || \, \tilde{p}^\lambda_\theta(T|C, I) \right]$$

where the target distribution $\tilde{p}^\lambda_\theta(T|C, I)$ is defined as:

$$\tilde{p}^\lambda_\theta(T|C, I) = \frac{1}{Z(\lambda)} f_\theta(C; T, I)^\lambda$$

where $Z \in \mathbb{R}^+$ is a normalization constant and $\lambda \in \mathbb{R}^+$ is a temperature constant. This distribution is formed by computing $p_{\theta,\phi}(C|T, I)^\lambda$ over transforms T and normalizing them. Thus, it assigns a higher probability to the transformations with lower classification loss. λ here is analogous to the temperature parameter in softmax, and large values of λ make the distribution highly peaked. In contrast, small values suppress peaks and make the distribution less ill-behaved as a target. $\lambda \to 0$ corresponds to a uniform distribution, whereas $\lambda \to \infty$ collapses the distribution to the single transformation that minimizes the classification loss.

We also note that when $\lambda = 1$, the target distribution $\frac{1}{Z} p_\theta(C|T, I)$ is exactly the posterior $p_\theta(T|C, I)$, assuming a uniform prior for the unknown $p_\theta(T|I)$. Different choices of this prior lead to other loss functions, like a Gaussian prior penalizing the transformation norm. However, we stick to the uniform prior for simplicity.

Representing the conditional distribution: Our approach uses parametrized differentiable augmentations similar to Augerino. However, instead of learning the global range of transformations, we predict a distribution over the transformations conditioned on the input image. We use an input-conditional normalizing flow model [35].

A normalizing flow model starts with a simple pre-defined probability distribution p_0, e.g., Normal distribution. For a sample $z_0 \sim p_0$, it successively applies transformations $f_1, f_2, \ldots, f_K$, producing a more complicated distribution by the end. The log probability density of the final sample is given by $\log p(z_k) = \log p_0(z_0) - \log |\det \frac{dz_k}{dz_0}|$, and the architecture is designed to allow efficient sampling and computation of $\log p$. We use the samples to augment the input (Figure 2) and $\log p$ term in the loss. Our model is based on RealNVP [36], using a mixture of Gaussians as the base p_0.

Given any input image I, we use a convolutional feature extractor to extract an embedding vector e. This embedding vector is then projected down to a scale and bias used by each layer of the normalizing flow and the base distribution. This normalizing flow model outputs samples s from the augmentation distribution and their corresponding log-probabilities $\log p(s)$. These samples are passed to the differentiable augmentation, which transforms the input image to be processed by the model (Figure 2) using PyTorch's *grid_sample*. While we use affine image transformations for our experiments, our method generalizes to any differentiable transformation.

Approximate invariance: Here, we formalize the notion of approximate invariance and connect it to our classifier and flow model. Intuitively, the approximate invariance in our method comes from both the augmenter and the classifier. Their contributions can be divided into (1) the classifier's inherent insensitivity to transformations, (2) the width of the transform distribution being used for averaging, and (3) the canonicalization effect of the transform distribution. Each of these properties corresponds to a different theory of object recognition explained by Tarr and Pinker [1] and connected to deep neural networks by Kaba *et al*. [32]. We formalize this intuitive argument as follows: Given an input image I, our model's output is the classifier prediction averaged over $g_\phi(T; I)$, i.e. $p_{\theta,\phi}(C|I) = \mathbb{E}_{T \sim g_\phi(T;I)} \left[f_\theta(C; \mathcal{A}_T(I)) \right]$ (see equation 9). Let a new image I' be formed by transforming the original image by a transformation ΔT, i.e. $I' = \mathcal{A}_{\Delta T}(I)$. Then:

$$p_{\theta,\phi}(C|I') = \int_T g_\phi(T; \mathcal{A}_{\Delta T}(I)) f_\theta(C; \mathcal{A}_{T+\Delta T}(I)) dT$$

$$= \int_T g_\phi(T - \Delta T; I') f_\theta(C; \mathcal{A}_T(I)) dT$$

Where the last step substitutes T for $T + \Delta T$. Then, the change in prediction, denoted as $\text{err}(C; I, I')$, is:

$$\text{err}(C; I, I') = |p_{\theta,\phi}(C|I) - p_{\theta,\phi}(C|I')|$$

$$= \left| \int_T \left[g_\phi(T - \Delta T; I') - g_\phi(T; I) \right] f_\theta(C; \mathcal{A}_T(I)) dT \right|$$

Next, we derive bounds on this quantity based on g_ϕ and f_θ. Let $S = \text{supp}(g_\phi(.; I)) \cup \text{supp}(g_\phi(.; I'))$ is the support set of the transform distributions, i.e. all the samples for I and I' are inside S. We can thus limit the integration to S:

$$= \Big| \int_{T \in S} \big[g_\phi(T - \Delta T; I') - g_\phi(T; I) \big] f_\theta(C; \mathcal{A}_T(I)) dT \Big|$$

Let's now quantify the behavior of f_θ on S. Let M be the maximum and m be the minimum of f_θ on this set, i.e.

$$M = \max_{t \in S} f_\theta(C; \mathcal{A}_T(I)), \quad m = \min_{t \in S} f_\theta(C; \mathcal{A}_T(I)),$$

Note that the first term $g_\phi(T - \Delta T; I') - g_\phi(T; I)$ is the difference of two probability density functions and so integrates to 0. Thus, if we add a constant value to f_θ, it doesn't change the whole integral. Subtracting m, we get:

$$\Big| \int_{T \in S} \big[g_\phi(T - \Delta T; I') - g_\phi(T; I) \big] (f_\theta(C; T, I) - m) dT \Big|$$

Using $|\int f(x)dx| \leq \int |f(x)|dx$ and $|xy| = |x||y|$ we have:

$$\leq \int_{T \in S} \Big| g_\phi(T - \Delta T; I') - g_\phi(T; I) \Big| \Big| f_\theta(C; T, I) - m \Big| dT$$

$$\leq (M - m) \int_{T \in S} \Big| g_\phi(T - \Delta T; I') - g_\phi(T; I) \Big| dT$$

$$= 2(M - m)\, \text{TV}[g_\phi(T - \Delta T; I') \| \, g_\phi(T; I)]$$

where TV refers to the Total Variation Distance defined as $\text{TV}[\text{p} \| \text{q}] = \frac{1}{2} \int |p(x) - q(x)| dx$. In summary:

$$\text{err}(C; I, I') \leq 2(M - m)\, \text{TV}[g_\phi(T - \Delta T; I') \| g_\phi(T; I)]$$

Thus, the prediction change ($\text{err}(C; I, I')$) is upper bounded by two factors: **(1)** $M - m$, which measures how much the classifier predictions change over the relevant range, and **(2)** the total variation distance between the original transform distribution $g_\phi(T; I)$ and the new version $g_\phi(T - \Delta T; I')$. This result explains how the method achieves approximate invariance. If the classifier features are invariant to the input transformations, we get $M - m \approx 0$, and thus $error \approx 0$. The same is true if the transform distribution is approximately equivariant, i.e. $g_\phi(T - \Delta T; I') \approx g_\phi(T; I)$.

Mean-shift for handling out-of-distribution poses: While the conditional transformation distribution $g_\phi(T; I)$ can adjust to in-distribution pose variation, this approach does not work for out-of-distribution poses (see Figure 10). We use a modified version of the well-known *mean-shift algorithm*. Instead of sampling points from a dataset and weighting them with a kernel, we directly use g_ϕ samples.

The core idea is to push the image closer to a local mode where our models may work better. We start with image I_0 and the transform parameter $T_0 = 0$. Then, at every step:

$$T_k := T_{k-1} + \gamma \mathbb{E}_{T \sim g_\phi(T; I_{k-1})}[T], \quad I_k := \mathcal{A}_{T_k}(I_0)$$

where $\gamma \in \mathbb{R}^+$ is the step size. In summary, the algorithm repeatedly samples from the conditional distribution, computes the mean, and accumulates the result into T.

Since our method learns an input-conditional probability distribution, the mean of the augmentation transformation $\mathbb{E}_{T \sim g_\phi(T; I)}[T]$ for any given image is an estimate of the difference between the local mode and the current transform T. Thus, each step moves the image closer to the local mode, which is the fixed point for this process.

4. Experiments

We benchmark accuracy on datasets such as CIFAR10 and TinyImageNet, and plot the learned transformation distribution for toy examples on Mario-Iggy [14] and MNIST. Finally, we test applications of the learned distribution. The code and scripts to reproduce all the results can be found at https://github.com/sutkarsh/flow_inv/

	CIFAR10	**FMNIST**	**MNIST**	**CIFAR10-LT**
Baseline	74.1 ± 0.5	89.6 ± 0.2	99.1 ± 0.02	70.8 ± 0.8
Augerino	79.0 ± 1	90.1 ± 0.1	98.3 ± 0.1	63.6 ± 1.3
LILA	84.2 ± 0.8	91.9 ± 0.2	**99.4** ± 0.02	76.4 ± 0.9
Ours	**86.8** ± **0.4**	**92.3** ± **1.4**	99.2 ± 0.1	**78.1** ± **1**
Gain	(+2.6)	(+0.4)	(-0.2)	(+1.7)

Table 1: Classification accuracy on the modified ResNet used by LILA [17]. Numbers for baselines reproduced from [17]. Our method helps the classifier achieve the highest test accuracy on CIFAR10 and CIFAR10-LT(rho=10). Imbalanced classification is particularly challenging since invariances learned through augmentations do not transfer from head classes to tail classes [13]. We note that our method is complementary to LILA and can be combined in future work.

CIFAR10: We benchmark our method against Augerino and LILA [17] on learning affine image transformations for CIFAR10 classification. We use the models and libraries provided by [17]. We use a RealNVP flow [36] with permutation mixing, 12 affine coupling layers, and a 2-layer MLP of width 64 for each layer. We turn the input into an embedding using a 5-layer CNN and append this embedding to each layer's MLP input as well as project it to the parameters of the base distribution, which is a mixture of Gaussians. We also add a $\tanh$ at the end of the flow to ensure the produced distribution stays within bounds. Please see the supplementary material for more details. Using a modified ResNet18 [37], and train our model for 200 epochs. We report the accuracy in Section 4. Our method is able to

achieve a 7.8% test accuracy gain compared to Augerino and 2.6% against LILA. We note that our method is still based on maximum likelihood; thus, LILA's marginal likelihood method is complementary to ours. These methods may be combined for even higher accuracy in future work. We also report the accuracies for MNIST and FashionMNIST.

Imbalanced CIFAR-10 Classification: Imbalanced classification is a challenging setting for invariance learning. As shown by [13], invariances learned through data augmentation do not transfer from head classes to tail classes. This is especially harmful since the tail classes, due to a small number of examples, benefit the most from the invariance. CIFAR10-LT is an imbalanced version of CIFAR10 where the smallest class is 10x smaller than the largest. Here, we outperform Augerino by 14.5% and LILA by 1.7%.

Augerino 13-layer CIFAR10: We also evaluate our method on Augerino's 13-layer network, re-using the same hyperparameters as the LILA experiments Section 4. Our method achieves 94.3% test accuracy (0.5% gain).

	No Aug.	Fast AA	Augerino	Ours
Acc	90.6	92.65	93.8	**94.3** $\pm$ 0.08

Table 2: Test accuracies for Augerino's 13-layer model. Baseline numbers quoted from [14].

TinyImageNet Classification: We evaluate our method against InstaAug on the TinyImageNet dataset. This 64x64 dataset contains 200 classes. The goal of this task is to learn cropping augmentations. A crop can be parametrized with four parameters: $(\text{center}_x, \text{center}_y, \text{width}, \text{height})$, so we represent it with a 4-dimensional distribution. Please see the supplementary material for more details.

Cropping is a challenging augmentation to learn since the crop location and size are correlated. InstaAug's mean-field representation cannot represent this, so achieves low accuracy without the location-related parameterization (LRP). LRP consists of 321 pre-defined crops and predicts the probability of each crop. This approach does not scale to high dimensional distributions (e.g. specifying more transformations). In contrast, our method can achieve high accuracy without LRP, beating InstaAug by nearly 11% (Table 3).

Learned invariance visualization Mario-Iggy is a toy dataset from [14] consisting of rotated versions of two images. Upright and upside-down images are classified as different classes, and each sample lies within $\pm 45°$ of its class prototype. As the total range of rotations can be easily varied, this dataset is useful for studying learned invariance. We consider two variations: $\pm 90°$ **rotation range**, and **Multi-modal dataset with** 3 **modes**.

The ideal augmentation distribution for Mario-Iggy dataset is $\pm 90°$ around the class prototype. As the input image rotates, the augmentation distribution shifts such that the resulting augmented image distribution is constant. Our model trained on Mario-Iggy can reliably learn an invariant augmentation distribution (Figure 4). In the challenging multimodal distribution setting, our model can represent the three modes, whereas InstaAug fails.

Method	**Acc (%)**	**+LRP(%)**
Baseline	55.1	—
Random Crop	64.5	—
Augerino	55.0	—
InstaAug	54.4	66.0
Ours	**65.4**	66.0

Table 3: TinyIN classification accuracy on PreActResNet used by InstaAug, with and without location-relation parameterization. InstaAug is limited by its mean-field representation, performing poorly without LRP. In contrast, our method performs well regardless of parametrization.

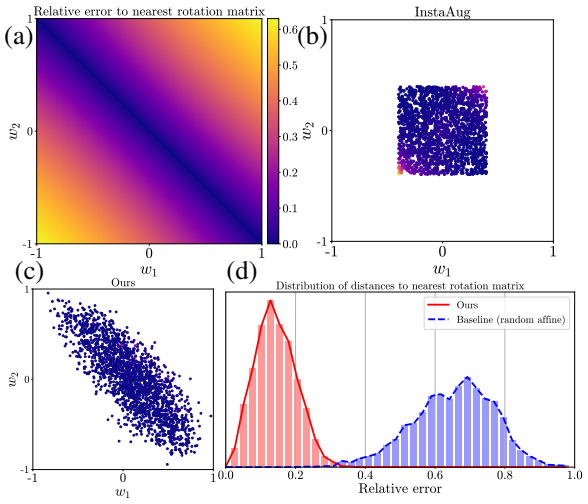

Figure 7: Our model learns the rotation constraint from data, while InstaAug fails to represent non-axis-aligned distributions. The goal of the "rotation discovery" task is to learn the joint distribution of affine matrix parameters such that the result is rotation. (w_1, w_2) pairs on diagonal (i.e., $w_1 = -w_2$) correspond to exact rotations and thus incur a small classification loss. **(a)** Relative error to the nearest rotation matrix. The ideal distribution of augmentations is in the form of a diagonal strip. **(b)** InstaAug produces a small square as its mean-field parametrization is unable to represent correlations between two parameters. **(c)** Our model learns to produce samples on the diagonal and learns a much larger range than InstaAug. **(d)** We plot the histogram of relative errors of the produced samples to the nearest rotation matrix. It is much smaller than the random affine baseline. Our model learns the joint distribution and discovers rotations from the full set of affine parameters, while InstaAug fails.

Representing joint distributions: We test the ability of our normalizing flow to represent joint distributions by

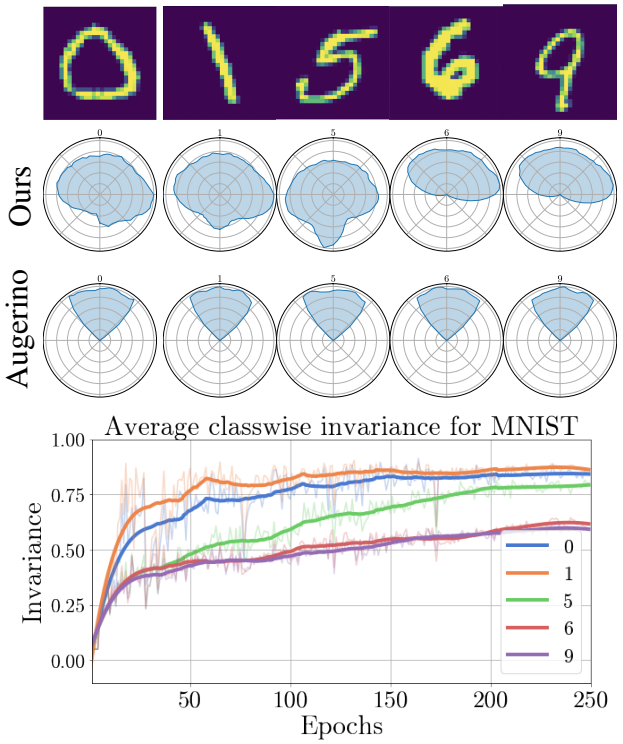

Figure 8: Our method learns flexible instance-wise augmentation distributions. We illustrate learned invariance for a subset of MNIST digits (0,1,5,6,9). The classes 0,1,5 can be learned with full invariance, whereas 6 and 9 require partial invariance ($\pm 90°$). Our model (top) can learn the correct instance-dependent range, whereas Augerino (middle) instead learns a much narrower shared invariance for all classes. (bottom) A plot of the classwise learned rotational invariance for our model over time. Classes 0, 1, and 5 achieve close to full rotational invariance, whereas 6 and 9 achieve close to $\pm 90°$ rotational invariance.

intentionally sampling from a larger set of transformations and letting the model learn the useful subset. Specifically, we start from the Lie algebra parametrization of affine transforms (used by Augerino). For rotation by r radians, the transformation matrix is:

$$T_{\text{Augerino}}(r) = \exp\left(\begin{bmatrix} 0 & r & 0 \\ -r & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) \tag{12}$$

For this experiment, we generalize this formulation as:

$$T_{\text{Decoupled}}(a,b,c,d,e,f) = \exp\left(\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}\right) \tag{13}$$

This matrix represents a rotation if $b = -d$. Since the Mario-Iggy dataset only contains rotations, the goal is to produce samples such that $b = -d$. Samples that do not follow this constraint will be out-of-distribution. Figure 7 shows that, unlike our model, InstaAug [15] fails to learn rotation transforms for Mario-Iggy, even though skewed samples incur

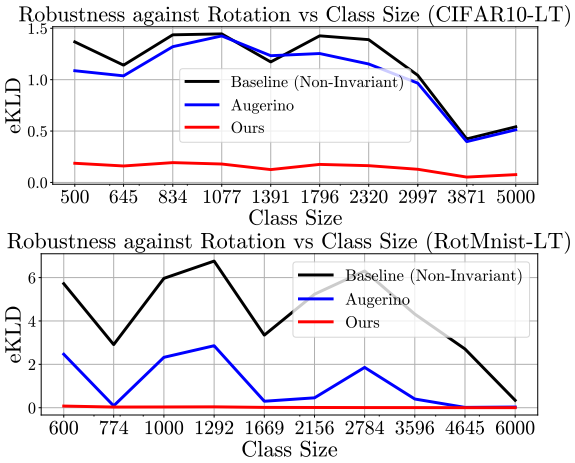

Figure 9: Invariance transfer from head classes to tail classes in imbalanced classification. We follow Zhou *et al.* [13] (Fig 3) and plot the expected KL-divergence under image rotations for RotMNIST-LT and CIFAR10-LT (lower is better). RotMNIST-LT is a long-tail version of the MNIST dataset where each image has been randomly rotated. As Zhou *et al.* [13] shows, neural networks learn rotational invariance for head classes (indicated by low eKLD) but fail to transfer this invariance to tail classes. This problem persists for Augerino to a lesser extent. In contrast, our method successfully transfers invariance across classes. This effect is even more pronounced for CIFAR10-LT ($\pm 10°$ rotations)

a higher loss. This is due to InstaAug's mean-field model, which predicts the range for each parameter separately, thus preventing it from following the $b = -d$ constraint. In contrast, our model learns to represent this joint distribution. We further test our model's ability to learn the rotation constraint on all 6 affine parameters. Figure 7 also shows the deviation of sampled transformations from a true rotation matrix. Our learned distribution is concentrated close to the rotation transformations, showing that our method can start from a large group of transformations and learn to constrain it to only what is useful for the dataset and task.

Learning selective invariance for MNIST: We test our model's selective invariance ability on the MNIST dataset (specifically 0,1,5,6,9) and visualize the augmentation range for a few examples as well as class averages (see Figure 8). For digits 0, 1, and 5, which can be recognized from any rotation, the learned rotation range corresponds to the entire $360°$, whereas for 6 and 9, which may be confused with each other, the range is only $180°$. In contrast, augerino learns a constant range. We find the same trend at the class level.

Generalizing invariance across classes: Zhou *et al.* [13] shows that invariances learned from head classes fail to transfer to tail classes. This is a major drawback of traditional data augmentation. We test generalization across classes by plotting the same metric as [13] (expected KL divergence) across a range of rotations for CIFAR10-LT and RotMNIST-

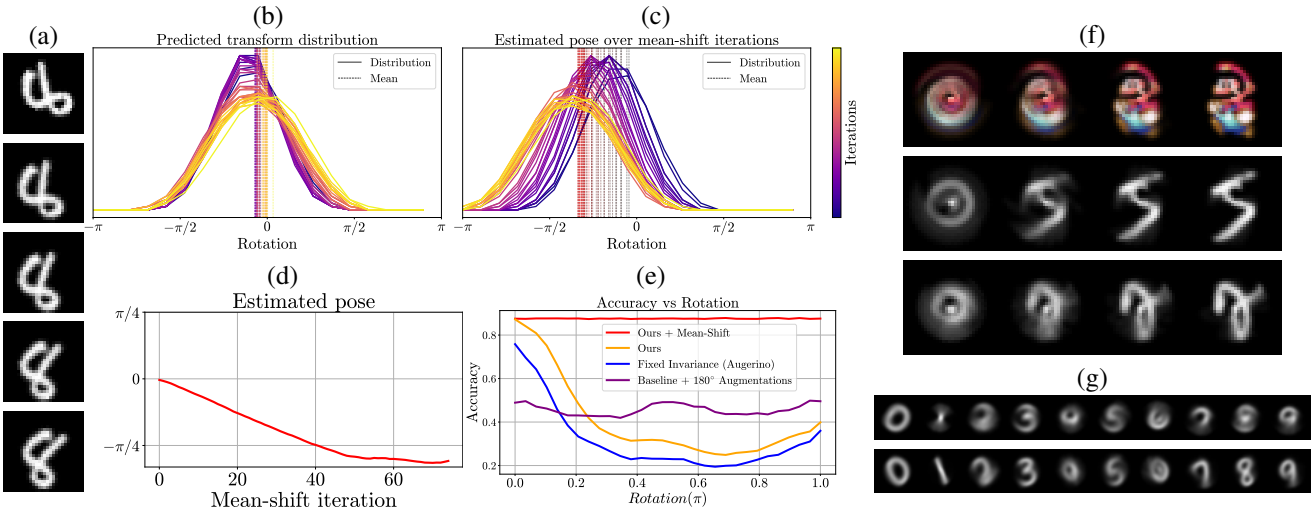

Figure 10: The conditional augmentation distribution can be used to align an image dataset, discover prototypes similar to congealing [2], and adapt to out-of-distribution poses. **(a)** Example of the mean shift algorithm aligning a digit belonging to an unseen class. **(b-d)** Figures showing the modified mean-shift algorithm. For a given input, we repeatedly compute the mean of the conditional transform distribution and perturb the input in that direction, pushing the input close to a local mode. As a result, the mean of the transformation distribution slowly shifts to 0 while the estimated pose gets closer to the true pose. **(e)** Modified mean-shift algorithm can add robustness against unexpected poses without reducing accuracy. We plot each CIFAR10 model's accuracy as images rotate at test time. Augerino is susceptible to large rotations since they are out-of-distribution for CIFAR10. The baseline trained with augmentations is robust but inaccurate. Our method with mean-shift achieves high accuracy for both in-distribution and out-of-distribution rotations. **(f)** Demonstration of an augmentation distribution aligning rotated ($\pm 90°$) versions of a single image. We separately apply mean-shift to each rotated image and observe that they converge to the same mode. Unlike [2], there is no joint optimization, and each image is "aligned" separately. This alignment also works for MNIST images even though the model has only trained on Mario-Iggy. **(g)** We apply the model trained on Mario-Iggy to align each class in the MNIST test set, and we make the task more challenging by adding $\pm 45°$ rotations to each image. The top row shows the average class image before alignment, and the bottom row shows images after alignment. We successfully discover prototypes for $0, 1, 3, 8, 9$, whereas for classes like $4, 6$, the model fails due to multiple possible modes.

LT classifiers. Since RotMNIST-LT is a rotationally invariant dataset, we rotate all the images randomly in the $\pm 180°$ range, whereas for CIFAR10-LT we use a $\pm 10°$ range. Our model achieves significantly lower eKLD, especially for tail classes (Figure 9), indicating higher robustness.

Aligning image datasets like in Congealing [2]: We apply the mean-shift algorithm using the augmentation distribution trained on the Mario-Iggy ($45°$) dataset. The Mario-Iggy dataset contains rotated versions of the Mario image with one unknown prototype, making it ideal for this test.

For each image, we apply the mean-shift algorithm. Each step moves the image closer to the local mode. We apply this procedure for 50 iterations for every image separately. This process results in all the images in a small neighborhood agglomerating to the local prototype (Figure 10).

We also tested this approach on MNIST, an out-of-distribution dataset for the mario-iggy model, and added $\pm 45°$ rotations for an additional challenge. Surprisingly, the method still aligns images and discovers prototypes (Figure 10) despite not being trained on any MNIST images.

Robustness to out-of-distribution poses: We benchmark our model's ability to handle out-of-distribution poses on CIFAR10 and measure how the mean-shift method helps the model adapt to unexpected poses. We plot the classification accuracy curves in Figure 10 as the inputs rotate. For the modified mean-shift method, we sample 100 transform samples, $\gamma = 0.1$, and 10 iterations. The fully-invariant baseline is robust but inaccurate. Augerino, which induces invariance to a small range of rotations, fails for large rotations. Our model without mean-shift also fails under large rotations. However, with mean-shift, it is accurate and robust.

Summary: We propose normalizing flows to learn the instance-wise distribution of image transformations. It helps us make robust and better generalizing classifiers, perform test-time alignment, discover prototypes, transfer invariance, and achieve higher test accuracy. These results highlight the potential of flexible, adaptive, and general invariance in computer vision.

Acknowledgements: This work was supported, in part, by the BAIR/Google fund.

References

[1] Michael J Tarr and Steven Pinker. Mental rotation and orientation-dependence in shape recognition. *Cognitive psychology*, 21(2):233–282, 1989. 1, 2, 5

[2] E.G. Miller, N.E. Matsakis, and P.A. Viola. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, volume 1, pages 464–471 vol.1, 2000. 1, 2, 3, 4, 9

[3] Asher Koriat and Joel Norman. Mental rotation and visual familiarity. *Perception & Psychophysics*, 37(5):429–439, 1985. 1, 2

[4] Roger N Shepard and Jacqueline Metzler. Mental rotation of three-dimensional objects. *Science*, 171(3972):701–703, 1971. 1, 2

[5] Spandan Madan, Tomotake Sasaki, Tzu-Mao Li, Xavier Boix, and Hanspeter Pfister. Small in-distribution changes in 3d perspective and lighting fool both cnns and transformers, 2021. 1, 2

[6] 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*, pages 248–255. Ieee, 2009. 1

[7] Vaishaal Shankar, Achal Dave, Rebecca Roelofs, Deva Ramanan, Benjamin Recht, and Ludwig Schmidt. Do image classifiers generalize across time? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9661–9669, 2021. 1

[8] NTS Board. Collision between vehicle controlled by developmental automated driving system and pedestrian. *Transportation Safety Board, Washington, DC, USA*, HAR19-03, 2019. 1

[9] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021. 1, 2

[10] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999. 1, 2

[11] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988. 1, 2

[12] Diane Bouchacourt, Mark Ibrahim, and Ari Morcos. Grounding inductive biases in natural images: invariance stems from variations in data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 19566–19579. Curran Associates, Inc., 2021. 1, 2, 3

[13] Allan Zhou, Fahim Tajwar, Alexander Robey, Tom Knowles, George J. Pappas, Hamed Hassani, and Chelsea Finn. Do deep networks transfer invariances across classes? 2022. 2, 3, 6, 7, 8, 13

[14] Gregory Benton, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. Learning invariances in neural networks, 2020. 2, 3, 4, 6, 7, 13

[15] Ning Miao, Tom Rainforth, Emile Mathieu, Yann Dubois, Yee Whye Teh, Adam Foster, and Hyunjik Kim. Instance-specific augmentation: Capturing local invariances, 2022. 2, 3, 4, 8, 13

[16] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. In *International Conference on Learning Representations*, 2021. 2

[17] Alexander Immer, Tycho F. A. van der Ouderaa, Gunnar Rätsch, Vincent Fortuin, and Mark van der Wilk. Invariance learning in deep neural networks with differentiable laplace approximations, 2022. 2, 3, 6, 14

[18] Lynn A Cooper and Roger N Shepard. Chronometric studies of the rotation of mental images. In *Visual information processing*, pages 75–176. Elsevier, 1973. 2

[19] Howard S Hock and Cheryl L Tromley. Mental rotation and perceptual uprightness. *Perception & Psychophysics*, 24(6):529–533, 1978. 2

[20] Marysia Winkels and Taco S. Cohen. Pulmonary nodule detection in CT scans with equivariant cnns. *Medical Image Anal.*, 55:15–26, 2019. 2

[21] Maxime W Lafarge, Erik J Bekkers, Josien PW Pluim, Remco Duits, and Mitko Veta. Roto-translation equivariant convolutional networks: Application to histopathology image analysis. *Medical Image Analysis*, 68:101849, 2021. 2

[22] Simon Graham, David B. A. Epstein, and Nasir M. Rajpoot. Dense steerable filter cnns for exploiting rotational symmetry in histology images. *IEEE Trans. Medical Imaging*, 39(12):4124–4136, 2020. 2

[23] Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1889–1898, 2016. 2

[24] Nathanaël Perraudin, Michaël Defferrard, Tomasz Kacprzak, and Raphaël Sgier. Deepsphere: Efficient spherical convolutional neural network with healpix sampling for cosmological applications. *Astron. Comput.*, 27:130–146, 2019. 2

[25] Brandon M. Anderson, Truong-Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14510–14519, 2019. 3

[26] Kristof Schütt, Oliver T. Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 9377–9388, 2021. 3

[27] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al.

Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. 3

[28] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning, ICML*, 2018. 3

[29] Taco S Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. In *Advances in Neural Information Processing Systems*, pages 9142–9153, 2019. 3

[30] Marc Finzi, Gregory Benton, and Andrew G Wilson. Residual pathway priors for soft equivariance constraints. In *Advances in Neural Information Processing Systems*, volume 34, 2021. 3

[31] Tri Dao, Albert Gu, Alexander J. Ratner, Virginia Smith, Christopher De Sa, and Christopher Ré. A kernel theory of modern data augmentation. *ICML*, 97:1528–1537, 2019. 3

[32] Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, Yoshua Bengio, and Siamak Ravanbakhsh. Equivariance with learned canonicalization functions. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*, 2022. 3, 5

[33] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015. 3

[34] Pola Schwöbel, Frederik Rahbæk Warburg, Martin Jørgensen, Kristoffer Hougaard Madsen, and Søren Hauberg. Probabilistic spatial transformer networks. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022. 3, 4

[35] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. 4, 5

[36] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017. 5, 6, 12

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016. 6

[38] Vincent Stimper, David Liu, Andrew Campbell, Vincent Berenz, Lukas Ryll, Bernhard Schölkopf, and José Miguel Hernández-Lobato. normflows: A PyTorch Package for Normalizing Flows. *arXiv preprint arXiv:2302.12014*, 2023. 12

[39] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. 12

[40] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2017. 13

A. Supplementary Material

We **(a)** present a class-wise accuracy analysis for CIFAR10-LT results, **(b)** present further analysis of the alignment experiment, discussing failure modes, and **(c)** share experimental details.

A.1. CIFAR-10LT class-wise accuracy

In Figure 11, we take the models from Figure 9 for CIFAR10-LT and plot the class-wise accuracy for each. We find that our model achieves higher accuracy for each class, and the margin is larger for the tail classes.

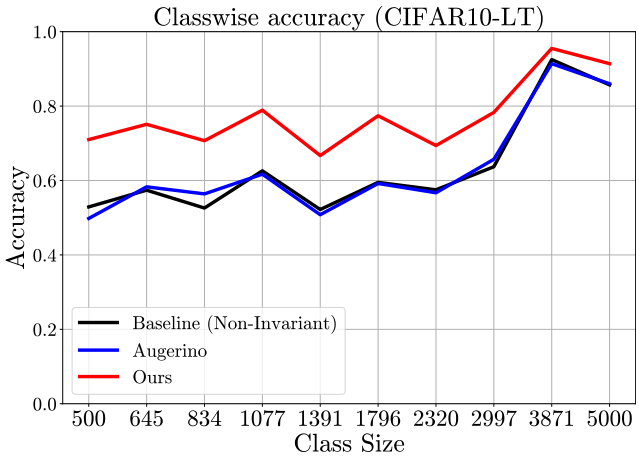

Figure 11: Our model achieves higher accuracy than Augerino and a naive model on every class on CIFAR10-LT, and the margin is larger for tail classes.

A.2. Mean-shift alignment failure due to multi-modality

We explore one of the reasons why our mean-shift method fails to align some digits in Figure 10 when the augmenter is trained on Mario/Iggy and tested on MNIST. We show that this algorithm is susceptible to multiple modes as it has no information about the true pose distribution of the MNIST digits. In Figure 12, we show three examples of MNIST digits. We rotate each digit by $\pm 180°$, run the alignment, and show the rotated and aligned versions superimposed. We find that for digits such as 4, 6, and 7 there are multiple modes/orientations onto which the algorithm can converge. This problem is more prevalent when the variation in poses in a class is large, leading to a similar kind of blurring of the post-alignment images as observed in Figure 10.

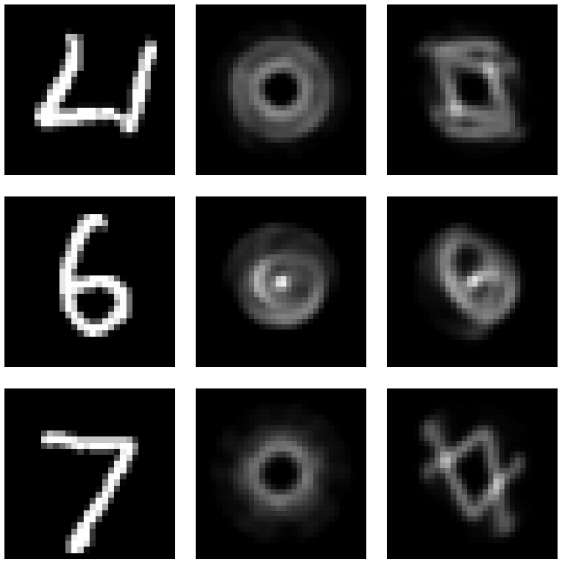

Figure 12: Mean-shift alignment can fail on an unseen dataset due to multi-modality. We demonstrate three examples of MNIST digits for an augmenter trained on the Mario/Iggy dataset. The digits are rotated by $\pm 180°$ and processed through the mean-shift alignment algorithm. We find that for digits like $4, 6, 7$, the algorithm can converge to distinct modes, leading to the post-alignment images looking blurry.

A.3. Experimental Details

A.3.1 Normalizing flow model

We use RealNVP [36] to model the distributions over transformation parameters with a $\tanh$ layer after every mixing step and at the end to ensure the model produces samples within $\big[-1, 1\big]$. The MLP at each layer of the model has 2 layers of width 32 each. Since the goal is an input-conditional distribution, we use a pose-embedding CNN to create an embedding vector e with 32 elements. This vector is used in two places: **(1)** it is projected through a learned linear layer to the base distribution parameters (location, scale, and the weight of each mode for the conditional mixture of Gaussians), and **(2)** it is concatenated to the MLP used by RealNVP at every layer. We scale MLP and projection weights by 0.01 to initialize them to small values, allowing the flow model to start as identity. We developed our implementation on top of normflows [38].

A.3.2 Base Distribution

We use a mixture of Gaussians or uniform random variables as the base distribution. We further use the Gumbel-softmax trick [39] to estimate gradients with respect to the weight of each mode. We found this to be important for learning equally weighted modes in the multi-modality experiments.

A.3.3 Stabilizing training with PID

A central issue in augmentation learning literature is how to use the optimal range of augmentations. If the range is too small, the model overfits, whereas if the range is too wide, the model underfits. This tradeoff is decided using a regularization constant that controls the augmentation distribution's width. Augerino fixes a regularization constant, but the resulting distribution width is difficult to control as it depends on the ratio of the classification loss and the regularization constant. This is further complicated by the fact that the classification loss changes over the course of training [15].

InstaAug constrains the distribution *entropy* to be in some pre-defined range $[H_{min}, H_{max}]$ exponentially increasing the regularization factor with each step until the entropy returns to the range. In practice, we found this method to be unstable in practice and difficult to tune. We replace this implementation with a PID controller, which adjusts the regularization constant to keep the entropy close to a set value. PID control is well understood in control systems literature and significantly easier to tune in practice. The regularization term increases linearly with the distance to the target and over time (as opposed to the exponential increase used by InstaAug), resulting in more stable dynamics.

A.3.4 Pose-embedding CNN

We use a 5-layer CNN architecture similar to the one used by Augerino [14]. It contains 4 convolutional layers (widths 32, 64, 128, 256 and kernel size 3) and ReLU non-linearity after each. This is followed by max-pooling and flattening to a 256-D vector which goes through a learned linear layer to produce the 32-D embedding.

A.3.5 Figure 2: Mario/Iggy experiments

We use a normalizing flow model with 4 layers, batch size 1024, and learning rate 10^{-3} with the AdamW [40] optimizer (betas: $0.9, 0.999$). We train the classifier without any augmentations for the first 5 epochs and for 100 epochs in total. We use the target entropy of 2 nits, with the PID control constants $(0.01, 0.01, 0)$ (corresponding to PI control). We further smooth the inputs and outputs of the PID controller with an exponential moving average with the smoothing constant 0.9. The base distribution is a conditional uniform distribution with 1 mode.

A.3.6 MNIST (Classes $0, 1, 5, 6, 9$)

We use the same model as the previous experiment, but with 12 layers for the normalizing flow and 3 modes in the mixture. We optimize it for 250 epochs.

A.3.7 Multi-modal experiments

We use a similar model as the MNIST classes experiment. However, we use a batch size 256, and train without augmentations for 3 epochs. We train for a total of 150 epochs and use a mixture of Gaussians with 120 modes. The Gumbel-softmax has a temperature of 0.05. We also set the regularization factor to 0.5 (no PID).

A.3.8 Rotation discovery experiments in Figure 4

We use the same model as the multi-modal experiments. However, we use a batch size of 512, 32 layers, LR 3×10^{-3}, and regularization factor to 0.002. Finally, we use a frozen MLP trained on the Mario/Iggy dataset and the augmentation model described in Equation 23.

A.3.9 Mean-shift alignment experiments

We use the model trained on the Mario/Iggy $45°$, and run the mean-shift algorithm for 20 iterations with $\alpha = 0.1$ and 100 transformation samples per iteration. For the CIFAR10 mean-shift experiments in Figure 10, we use 10 iterations, 50 samples, $\alpha = 0.1$.

A.3.10 eKLD plots

CIFAR10-LT: We use the trained model from CIFAR10-LT experiments, and rotate each image uniformly between $\pm 10°$. We follow the method used in [13]. **RotMNIST-LT**: We construct a fully rotated and long-tail version of the MNIST dataset with $\rho = 10$ and train the model with batch size 128 and 10 modes.

A.3.11 InstaAug experiments

We build upon the InstaAug [15] codebase. Our normalizing flow model has 12 layers. Since InstaAug already uses a CNN to convert images into a 321-dimensional embedding, we no longer use a CNN. We project the embedding through a learned linear layer to the base distribution parameters and add the 321-d embedding to each of the RealNVP's MLP layers. **Without LRP**: We follow the same training schedule as InstaAug, but replace their regularization scheduler with our PID ($k_p = 0.1, k_i = 0.5$) and their crop sampler with ours. We also set the target entropy to 2.75. We produce samples as 4-D vectors with entries in the $[-1, 1]$ range. We then convert this vector to a crop size and crop location. To speed up the training to match InstaAug's schedule, we limit the model predictions to valid crops throughout the training. To predict valid crop sizes in the range $[\text{llim}, \text{ulim}]$, we translate and scale the entries accordingly, and to predict the valid crop locations, we scale the crop center prediction by $(1 - size)$. The correlations between valid crop sizes and centers caused by this scaling are learned by our flow model.

Initially the limits are $[0.7, 1]$, followed by the less restrictive limits $[0.35, 1]$ after 20 epochs. **With LRP**: InstaAug's location-related parametrization (LRP) is a categorical distribution over crop locations and sizes. It already produces a good distribution of crops for TinyImageNet, so we build on it. We use InstaAug's entropy scheduler as well as crop sampler. We combine the two models by adding samples from our flow to the InstaAug output to increase crop diversity while maintaining the advantages of LRP.

A.3.12 CIFAR10

We build upon the codebase used by LILA [17] and use their ResNet $8 - 16$ model. We use a regularization factor 0.1 and initial LR 0.1 with the augmenter LR 10^{-4}. We train the model for 200 epochs and reduce the LR by a factor of 10 after every 80 epochs. **FMNIST**: We use regularization factor 0.03 and LR 0.03 and augmenter LR 10^{-3}. We train the model for 250 epochs, reducing the LR by a factor of 20 every 80 epochs. For MNIST, we reduce the LR to $1e - 3$ and decay it by 10 every 15 epochs. **CIFAR10-LT**: We use the same hyperparameters as the CIFAR10 experiment but instead reduce the LR by a factor of 20 instead. **Rejection sampling for test-time augmentation**: When the augmentation budget for TTA is small (e.g. ≤ 30 augmentations), we use rejection sampling to select the most useful augmentations. Specifically, we choose samples w such that $\|w\|_2 < 1$. This step significantly reduces the variance of the output (and thus the augmentations needed), while $\|w\|_2 < 1$ still covers the full range for any individual transformation type (i.e. $[-1, 1]$ along the axis).