Towards Resisting Large Data Variations via Introspective Learning

Yunhan Zhao  Ye Tian  Wei Shen  Alan Yuille
The Computational Cognition, Vision, and Learning Lab, Johns Hopkins University
{yzhao83, yitian27, wshen10, ayuillel}@jhu.edu

Abstract

Learning deep networks which can resist large variations between training and testing data are essential to build accurate and robust image classifiers. Towards this end, a typical strategy is to apply data augmentation to enlarge the training set. However, standard data augmentation is essentially a brute-force method which is inefficient, as it performs all the pre-defined transformations to every training sample. In this paper, we propose a principled approach to train networks with significantly improved resistance to large variations between training and testing data. This is achieved by embedding a learnable transformation module into the introspective network [14, 18, 20], which is a convolutional neural network (CNN) classifier empowered with generative capabilities. Our approach alternatively synthesizes pseudo-negative samples with learned transformations and enhances the classifier by retraining it with synthesized samples. Experimental results verify that our approach significantly improves the ability of deep networks to resist large variations between training and testing data and achieves classification accuracy improvements on several benchmark datasets, including MNIST, affNIST, SVHN, CIFAR-10 and miniImageNet.

1. Introduction

There has been rapidly progress on vision classification problem due to advances in convolutional neural networks (CNNs) [19, 17, 31, 32, 8, 9]. CNNs are able to produce promising performance given sufficient training data. However, when the training data is limited and unable to cover all the data variations in the testing data (e.g., the training set is MNIST, but the testing set is affNIST), the trained networks generalize poorly on the testing data. Consequently, how to learn deep networks which can resist large variations between training and testing data is a significant challenge for building accurate and robust image classifiers.

To address this issue, a typical strategy is to apply data augmentation to enlarging the training set, i.e., applying various transformations, including random translations, rotations and flips as well as Gaussian noise injection, to the existing training data. This strategy is very effective in improving the performance, but it is essentially a brute-force method which is inefficient, as it exhaustively performs all these transformations to every training samples. Neither is it theoretically well formulated.

It is clear that we can synthesize extra training samples using generative models. But, how can we generate synthetic samples which are able to improve the robustness of CNNs to large variations between training and testing data? In this paper, we achieve this by embedding a learnable transformation module into introspective networks (INs) [14, 18], a CNN classifier which is also generative. We name our approach introspective transformation network (ITN) and we train by the reclassification-by-synthesis algorithm. This alternatively synthesizes samples with learned transformations and enhances the classifier by retraining it with synthesized samples. The intuition of our approach is illustrated in Figure 1. Similar to the adversarial training [22], we use a min-max formulation to learn our ITN, where the transformation module transforms the synthesized pseudo-negative samples to maximize their variations to the original training samples and the CNN classifier is updated by minimizing the classification loss of the transformed synthesized pseudo-negative samples. The transformation modules are learned jointly with the CNN classifier, which augments training data in an intelligent manner by narrowing down the search space for the variations.

Our proposed framework is general, which can also work with other generative models, such as generative adversarial networks (GANs) [6, 23, 1, 7, 26, 30, 11, 37]. But we
choose introspective networks in our approach rather than GANs, due to the following reasons. First, INs can quantitatively prevent generating bad samples as they have a quality control mechanism in the generation process. Instead of directly decoding the latent vectors in GANs, INs iteratively refine the generated samples until they satisfy the criteria. Second, INs avoid adversarial training, which allows them to generate samples more stably than GANs. To verify our choice w.r.t. generative models in our framework, i.e., INs rather than GANs, we also implement auxiliary classifier GANs (AC-GANs) [25] under our framework, named auxiliary classifier generative adversarial transformation networks (AC-GATNs). AC-GAN is adopted since our framework requires the generative models have the ability to generate class dependent samples to enhance the classifier. We quantitatively and qualitatively compare ITN and AC-GATN in Section 4.1 to evidence the benefits of choosing introspective networks for building robust image classifiers.

The main contribution of the paper is that we propose a principled approach that endows classifiers with the ability to resist larger variations between training and testing data in an intelligent and efficient manner. ITN has the ability to efficiently generalize beyond the training set. Experimental results show that our approach achieves better performance than standard data augmentation on both classification and cross-dataset generalization. Furthermore, we also show that our approach has great abilities to resist different types of variations between training and testing data.

2. Related Work

In recent years, a significant number of works have emerged focus on resisting large variations between training and testing data. The most widely adopted approach is data augmentation that applies pre-defined transformations to the training data. Nevertheless, this method lacks efficiency and stability since the users have to predict the types of transformations and manually applies them to the training set. Better methods have been proposed by building connections between generative models and discriminative classifiers [5, 21, 34, 13, 36]. This type of methods capture the underlying generation process of the entire dataset. The discrepancy between training and test data is reduced by generating more samples from the data distribution.

GANs [6] have led a huge wave in exploring the generative adversarial structures. Combining this structure with deep CNNs can produce models that have stronger generative abilities. In GANs, generators and discriminators are trained simultaneously. Generators try to generate fake images that fool the discriminators, while discriminators try to distinguish the real and fake images. Many variations of GANs have emerged in the past three years, like DCGAN [26], WGAN [1] and WGAN-GP [7]. These GANs variations show stronger learning ability that enables generating complex images. Techniques have been proposed to improve adversarial learning for image generation [30, 7, 2] as well as for training better image generative models [26, 11].

Introspective networks [33, 18, 14, 20] provide an alternative approach to generate samples. Introspective networks are closely related to GANs since they both have generative and discriminative abilities but different in various ways. Introspective networks maintain a single model that is both discriminative and generative at the same time while GANs have distinct generators and discriminators. Introspective networks focus on introspective learning that synthesizes samples from its own classifier. On the contrary, GANs emphasize adversarial learning that guides generators with separate discriminators. The generators in GANs are mappings from the features to the images. However, Introspective networks directly models the underlying statistics of an image with an efficient sampling/inference process.

Our approach ITN is built upon introspective learning, but our focus is to address the large variations between training and testing data. ITN works in a similar way as data augmentation, however, data augmentation is an exhaustively searching method. ITN forms a natural min-max problem that searches samples more efficiently and effectively than standard data augmentation.

3. Method

We now describe the details of our approach in this section. We first briefly review the introspective learning framework proposed by [33]. This is followed by a detailed mathematical explanation of our approach. In particular, we focus on explaining how our model generates unseen examples that complement the training dataset.

3.1. Background: Introspective Learning

We only briefly review introspective learning for binary-class problems, since the same idea can be easily extended to multi-class problems. Let us denote $x \in \mathbb{R}^d$ as a data sample and $y \in +1, -1$ as the corresponding label of $x$. The goal of introspective learning is to model positive samples by learning the generative model $p(x|y = +1)$. Using Bayes rule, we have

$$p(x|y = +1) = \frac{p(y = +1|x)p(y = -1)}{p(y = -1|x)p(y = +1)} p(x|y = -1),$$

(1)

where $p(y|x)$ is a discriminative model. For pedagogical simplicity, we assume $p(y = 1) = p(y = -1)$, hence this equation can be further simplified as:

$$p(x|y = +1) = \frac{p(y = +1|x)}{p(y = -1|x)} p(x|y = -1).$$

(2)
The above equation suggests that a generative model for the positives \( p(x|y = +1) \) can be obtained from the discriminative model \( p(y|x) \) and a generative model \( p(x|y = -1) \) for the negatives. However, to faithfully learn \( p(x|y = +1) \), we need to have a representative \( p(x|y = -1) \), which is very difficult to obtain. A solution was provided in [33] which learns \( p(x|y = -1) \) by using an iterative process starting from an initial reference distribution of the negatives \( p_0(x|y = -1) \), e.g., \( p_0(x|y = -1) = U(x) \), a Gaussian distribution on the entire space \( \mathbb{R}^d \). This is updated by

\[
p_{t+1}(x|y = -1) = \frac{1}{Z_t} q_t(y = +1|x) p_t(x|y = -1),
\]

where \( q_t(y|x) \) is a discriminative model learned on a given set of positives and a limited number of pseudo-negatives sampled from \( p_t(x|y = -1) \) and \( Z_t = \int q_t(y = +1|x) p_t(x|y = -1) \) is the normalizing factor. It has been proven that \( KL(p(x|y = +1)||p_{t+1}(x|y = -1)) \leq KL(p(x|y = +1)||p_t(x|y = -1)) \) (as long as each \( q_t(y|x) \) makes a better-than-random prediction, the inequality holds) in [33], where \( KL(\cdot||\cdot) \) denotes the Kullback-Leibler divergences, which implies \( p_t(x|y = -1) \) by following this iterative process of Eqn.\( (3) \), the samples drawn from \( x \sim p_t(x|y = -1) \) become indistinguishable from the given training samples.

### 3.2. Large Variations Resistance via Introspective Learning

Introspective Convolutional Networks (ICN) [14] and Wasserstein Introspective Neural Networks (WINN) [20] adopt the introspective learning framework and strengthen the classifiers by a reclassification-by-synthesis algorithm. However, both of them fail to capture large data variations between the training and testing data, since most of the generated pseudo-negatives are very similar to the original samples. But in practice, it is very common that the test data contain unseen variations that are not in training data, such as the same objects viewed from different angles and suffered from shape deformation.

To address this issue, we present our approach building upon the introspective learning framework to resist large data variations between training and testing data. Arguably, even large training sets cannot fully contain all the possible variations. Our goal is to quickly generate extra training samples with beneficial unseen variations that is not covered by the training data to help classifiers become robust. We assume that we can generate such training samples by applying a transformation function \( \mathcal{T}(:; \sigma) \) parametrized by learnable parameters \( \sigma \) to the original training samples. Let us denote \( g(\cdot; \psi) \) as the function that maps the samples \( x \) to the transformation parameters \( \sigma \), where \( \psi \) is the model parameter of the function \( g \). The generated samples still belong to the same category of the original samples, since the transformation function \( \mathcal{T} \) only changes the high-level geometric properties of the samples. The outline of training procedures of ITN is presented in Algorithm 1. We denote \( S^+ = \{ (x_i^+, +1), i = 1...|S^+| \} \) as the positive sample set, \( \mathcal{I}(S^+; \sigma_t) = \{ (x_i^+, +1), i = 1...|S^+|, x_i^T = \mathcal{T}(x_i^+; \sigma_t) \} \) as the transformed positive sample set at the \( t^{th} \) iteration with transformation parameter \( \sigma_t \) and \( S_i^- = \{ (x_i^-, -1), i = 1...|S^-| \} \) as the set of pseudo-negatives drawn from \( p_t(x|y = -1) \). We then will describe the detail of the training procedure.

**Discriminative model** We first demonstrate the approach of building robust classifiers with given \( \sigma_t \). For a binary classification problem, at the \( t^{th} \) iteration, the discriminative model is represented as

\[
q_t(y|x; \theta_t) = \frac{1}{1 + \exp(-y f_t(x; \theta_t))}
\]

where \( \theta_t \) represents the model parameters at iteration \( t \), and \( f_t(x; \theta_t) \) represents the model output at the \( t^{th} \) iteration. Note that, \( q_t(y|x; \theta_t) \) is trained on \( S^+, \mathcal{I}(S^+; \sigma_t) \) and pseudo-negatives drawn from \( p_t(x|y = -1) \). In order to achieve stronger ability for resisting unseen variations, we want the distribution of \( \mathcal{I}(S^+; \sigma_t) \) to be approximated by the distribution of pseudo negatives \( p_t(x|y = -1) \), which can be achieved by minimizing the following Wasserstein distance [7]:

\[
D(\theta_t, \omega_t) = \mathbb{E}_{x^T \sim \mathcal{T}(S^+; \sigma_t)}[W_i(f_t(x^T; \theta_t); \omega_t)] - \mathbb{E}_{x^- \sim S_i^-}[W_i(f_t(x^-; \theta_t); \omega_t)] + \lambda \mathbb{E}_{x \sim \tilde{X}_t} ||\nabla_x W_i(f_t(x; \theta_t); \omega_t) - 1 ||_2^2.
\]

where \( W_i(\cdot; \omega_t) \) represents the function that computes the Wasserstein distance and \( \omega_t \) is the function parameter at the \( t^{th} \) iteration. Note that the input of the function \( W_i(\cdot; \omega_t) \) is the \( f_t(\cdot; \theta_t) \). Each \( \tilde{x} \) in the set \( \tilde{X}_t \) is computed with the formula \( \tilde{x} = cx^T + (1-\epsilon)x^- \), where \( \epsilon \) samples from uniform distribution \( U(0, 1) \), \( x^T \in \mathcal{T}(S^+; \sigma_t) \) and \( x^- \in S_i^- \). The term \( \lambda \mathbb{E}_{x \sim \tilde{X}_t} ||\nabla_x W_i(f_t(x; \theta_t); \omega_t) - 1 ||_2^2 \) is the gradient penalty that stabilizes the training procedure of the Wasserstein loss function.

The goal of the discriminative model is to correctly classify any given \( x^+, x^T \) and \( x^- \). Thus, the objective function of learning the discriminative model at iteration \( t \) is

\[
\min_{\theta_t, \omega_t} J(\theta_t) + D(\theta_t, \omega_t),
\]

where \( J(\theta_t) = \mathbb{E}_{(x,y) \sim S^+ \cup S_i^- \cup \mathcal{T}(S^+; \sigma_t)}[-\log q_t(y|x; \theta_t)] \). The classifiers obtain the strong ability in resisting unseen variations by training on the extra samples while preserving the ability to correctly classify the original samples.
Exploring variations. The previous section describes how to learn the robust classifiers when the \( \sigma_t \) is given. However, \( \sigma_t \) is unknown and there are huge number of possibilities for selecting \( \sigma_t \). Now, the problem becomes how do we learn the \( \sigma_t \) in a principled manner and apply it towards building robust classifiers? We solve this issue by formulating a min-max problem based upon Eqn.(6):

\[
\min_{\theta, \omega} \max_{\sigma} J(\theta, \sigma) + D(\theta, \omega, \sigma), \tag{7}
\]

Here, we rewrite \( J(\theta) \) and \( D(\theta, \omega) \) in Eqn.(5) and Eqn.(6) as \( J(\theta, \sigma) \) and \( D(\theta, \omega, \sigma) \), since \( \sigma \) is now an unknown variable. We also subsequently drop the subscript \( t \) for notational simplicity. The inner maximization part aims to find the transformation parameter \( \sigma \) that achieves the high loss values. On the other hand, the goal of the outer minimization is expected to find the the model parameters \( \theta \) that enables discriminators to correctly classify \( x^T \) and \( \omega \) allows the negative distribution to well approximate the distribution of \( T(S^+; \sigma) \). However, directly solving Eqn.(7) is difficult. Thus, we break up this learning process and first find a \( \sigma^* \) that satisfies

\[
\max_{\sigma} E_{(x^T, y) \sim T(S^+; \sigma)} \left[ -\log (q(y | x^T)) \right] + E_{x \sim T(S^+; \sigma)} [W(f(x^T; \theta); \omega)] + \lambda E_{\tilde{x} \sim \tilde{X}} [\| \nabla_{\tilde{x}} W(f(\tilde{x}; \theta); \omega) \|_2^2] \tag{8}
\]

where \( \theta \) and \( \omega \) are fixed. Then, \( \theta \) and \( \omega \) are learned by Eqn.(6) by fixing \( \sigma = \sigma^* \). Empirically, the first term in Eqn.(8) dominates over other terms, therefore we can drop the second and third terms to focus on learning more robust classifiers. The purpose of empirical approximation is to find the \( \sigma^* \) that make \( x^T \) hard to classify correctly. Instead of enumerating all possible examples in the data augmentation, Eqn.(8) efficiently and precisely finds a proper \( \sigma \) that increase the robustness of the current classifiers.

The parameter \( \sigma \) is learned with function \( q(\cdot; \psi) \). We have \( \sigma = g(x; \psi) + \zeta \), where \( \zeta \) is random noise follows the standard normal distribution. The function parameter \( \psi \) is learned by optimizing Eqn.(8). Notably, following the standard backpropagation procedure, we need to compute the derivative of the transformation function \( T \) in each step. In other words, the transformation function \( T(\cdot; \sigma) \) need to be differentiable with respect to the parameter \( \psi \) to allow the gradients to flow through the transformation function \( T \) when learning by backpropagation.

**Generative model** Now we describe the detailed method for generating \( S^- \). In the discriminative models, the updated discriminative model \( p(y|x) \) is learned by Eqn.(6). The updated discriminative model is then used to compute the generative model by Eqn.(3) in section 3.1. The generative model is learned by maximizing the likelihood function \( p(x) \). Note that learning the generative models is redundant since the update at each iteration only involves pseudo-negatives samples instead of the full generative model.

Let us denote the initial reference distribution by \( p_0^- (x) \) and \( p_n(x|y = -1) \) as \( p_n^- (x) \) for simplicity. Following standard introspective learning, we approximate samples drawn from latest negative distribution by first sampling from \( p_0^- (x) \) and iteratively update them to approach desired samples. With \( p_0^- \) and Eqn.(3), we have

\[
p_n^- (x) = \left( \prod_{t=1}^{n-1} \frac{q_t(y = +1|x)}{q_t(y = -1|x)} \right) p_0^- (x), \tag{9}
\]

where \( Z_t \) indicates the normalizing factor at the \( t \)-th iteration. The random samples \( x \) are updated by increasing maximizing the log likelihood of \( p_n^- (x) \). Note that maximizing log \( p_n^- (x) \) can be simplified as maximizing

\[
\prod_{t=1}^{n-1} \frac{q_t(y = +1|x)}{q_t(y = -1|x)} \tag{10}
\]

Since \( Z_t \) and \( p_0^- \) are fixed in Eqn.(9). From this observation, we directly learn a model \( h_t(x) \) such that

\[
h_t(x) = \frac{q_t(y = +1|x)}{q_t(y = -1|x)} = \exp (f_t(x; \theta_t)) \tag{10}
\]

Taking natural logarithm on both side of the equation above, we get \( \ln h_t(x) = f_t(x; \theta_t) \). Therefore, log \( p_n^- (x) \) can be rewritten as

\[
\log p_n^- (x) = \log \left( \prod_{t=1}^{n-1} \frac{q_t(y = +1|x)}{q_t(y = -1|x)} \right) p_0^- (x) \tag{11}
\]

where \( \theta_t \) and \( \omega_t \) are fixed. Then, \( \theta_t \) and \( \omega_t \) are learned by Eqn.(6) by fixing \( \sigma = \sigma^* \). Empirically, the first term in Eqn.(8) dominates over other terms, therefore we can drop the second and third terms to focus on learning more robust classifiers. The purpose of empirical approximation is to find the \( \sigma^* \) that make \( x^T \) hard to classify correctly. Instead of enumerating all possible examples in the data augmentation, Eqn.(8) efficiently and precisely finds a proper \( \sigma \) that increase the robustness of the current classifiers.

The parameter \( \sigma \) is learned with function \( g(\cdot; \psi) \). We have \( \sigma = g(x; \psi) + \zeta \), where \( \zeta \) is random noise follows the standard normal distribution. The function parameter \( \psi \) is learned by optimizing Eqn.(8). Notably, following the standard backpropagation procedure, we need to compute the derivative of the transformation function \( T \) in each step. In other words, the transformation function \( T(\cdot; \sigma) \) need to be differentiable with respect to the parameter \( \psi \) to allow the gradients to flow through the transformation function \( T \) when learning by backpropagation.

**Generative model** Now we describe the detailed method for generating \( S^- \). In the discriminative models, the updated discriminative model \( p(y|x) \) is learned by Eqn.(6). The updated discriminative model is then used to compute the generative model by Eqn.(3) in section 3.1. The generative model is learned by maximizing the likelihood function \( p(x) \). Note that learning the generative models is redundant since the update at each iteration only involves pseudo-negatives samples instead of the full generative model.

Let us denote the initial reference distribution by \( p_0^- (x) \) and \( p_n(x|y = -1) \) as \( p_n^- (x) \) for simplicity. Following standard introspective learning, we approximate samples drawn from latest negative distribution by first sampling from \( p_0^- (x) \) and iteratively update them to approach desired samples. With \( p_0^- \) and Eqn.(3), we have

\[
p_n^- (x) = \left( \prod_{t=1}^{n-1} \frac{q_t(y = +1|x)}{q_t(y = -1|x)} \right) p_0^- (x), \tag{9}
\]

where \( Z_t \) indicates the normalizing factor at the \( t \)-th iteration. The random samples \( x \) are updated by increasing maximizing the log likelihood of \( p_n^- (x) \). Note that maximizing log \( p_n^- (x) \) can be simplified as maximizing

\[
\prod_{t=1}^{n-1} \frac{q_t(y = +1|x)}{q_t(y = -1|x)} \tag{10}
\]

Since \( Z_t \) and \( p_0^- \) are fixed in Eqn.(9). From this observation, we directly learn a model \( h_t(x) \) such that

\[
h_t(x) = \frac{q_t(y = +1|x)}{q_t(y = -1|x)} = \exp (f_t(x; \theta_t)) \tag{10}
\]

Taking natural logarithm on both side of the equation above, we get \( \ln h_t(x) = f_t(x; \theta_t) \). Therefore, log \( p_n^- (x) \) can be rewritten as

\[
\log p_n^- (x) = \log \left( \prod_{t=1}^{n-1} \frac{q_t(y = +1|x)}{q_t(y = -1|x)} \right) p_0^- (x) \tag{11}
\]
where $C$ is a constant computed from the normalizing factors $Z_t$. This conversion allows us to maximize $\log p_n(x)$ by maximizing $\sum_{t=1}^{n-1} f_t(x; \theta_t)$. By taking the derivative of $\log p_n(x)$, the update step $\nabla x$ is:

$$\nabla x = \lambda \nabla (\sum_{t=1}^{n-1} f_t(x; \theta_t)) + \eta,$$

(12)

where $\eta \sim N(0, 1)$ is the random Gaussian noise and $\lambda$ is the step size that is annealed in the sampling process. In practice, we update from the samples generated from previous iterations to reduce time and memory complexity. An update threshold $T_u$ is introduced to guarantee that the generated negative images are close to the normal distribution. The reason for this threshold is to make sure the generated negative images are close to the majority of transformed positive images in the feature space.

### 3.3. Multi-class Classification

When dealing with multi-class classification problems, it is necessary to adapt the above reclassification-by-synthesis scheme to the multi-class case. We can directly follow the strategies proposed in [14] and extend ITN to deal with multi-class problems by learning a single CNN classifier with the softmax function. Therefore, under multi-class settings the discriminative model is represented as:

$$q_t(y = k|x; \theta_t) = \frac{\exp(f_t(x; \theta^k_t))}{\sum_{i=1}^{n} \exp(f_t(x; \theta^i_t))},$$

(13)

where $n$ is the number of total classes, $f_t(x; \theta^k_t)$ represents the model output of the $k^{th}$ class at the $t^{th}$ iteration. The objective function of the multi-class discriminative model is same as Eqn. (6) and the $J(\theta_t)$ is rewritten as:

$$J(\theta_t) = \mathbb{E}_{(x,y) \sim S} - \sum_{k=1}^{n} \log q_t(y|x; \theta_t) + \mathbb{E}_{(x,y) \sim S}\left[-\log(1 + \exp(1 + f_t(x; \theta^k_t)))\right].$$

(14)

### 4. Experiments

In this section, we demonstrate the ability of our algorithm to resist large variations between training and testing data through a series of experiments. First, we show the strong classification performance of ITN on MNIST and affNIST datasets. Following that we present a series of analyses of resisting ability of ITN, where experiments and analyses of our choice w.r.t generative models are also included. Then, we verify the performance of ITN on SVHN, CIFAR-10 and a more challenging dataset, miniImageNet. Finally, we illustrate the flexibility of our architecture by addressing different types of unseen variations.

#### Experiment Setup

Following the setup used in WINN [20], all experiments are conducted with a simple CNN architecture [20] unless otherwise specified. We name this simple CNN architecture, B-CNN for notational simplicity. B-CNN contains 4 convolutional layers, each having a $5 \times 5$ filter size with 64 channels and stride 2 in all layers. Each convolutional layer is followed by a batch normalization layer [10] and a swish activation function [27]. The last convolutional layer is followed by two fully connected layers to compute logits and Wasserstein distances. The optimizer used is the Adam optimizer [15] with parameters $\beta_1 = 0$ and $\beta_2 = 0.9$. Our method relies on the transformation function $T(\cdot)$ to convert the original samples to the unseen variations. In the following experiments, we demonstrate the ability of ITN to resist large variations with spatial transformers (STs) [12] as our transformation function unless specified. Theoretically, STs can represent all affine transformations, which gives more flexible ability for resisting unseen variations. More importantly, STs are fully differentiable, which allows learning by standard backpropagation.

**Baselines**

B-CNN mentioned above is our baseline model in this section. Following [14] and [20], we compare our method against B-CNN, DCGAN [26], WGAN-GP [7], ICN [14] and WINN [20]. Since DCGAN and WGAN-GP are generative models, to apply them for classification, we adopt the same strategy used in [14]. The strategy makes the training phase become a two-step implementation. We first generate negative samples with the original implementation. Then, the generated negative images are used to augment the original training set. We train the B-CNN on the augmented training set. We denote these two GAN based classification method DCGAN+B-CNN and WGAN-GP+B-CNN. Like our method IN, they are built upon the baseline B-CNN. We denote them ICN (B-CNN), WINN (B-CNN) and ITN (B-CNN). All results reported in this section are the average of multiple repetitions.

#### 4.1. MNIST & affNIST

MNIST is a benchmark dataset that includes 55000, 5000 and 10000 handwritten digits in the training, validation and testing set, respectively. The affNIST dataset is a variant from the MNIST dataset and it is built by applying various affine transformations to the samples in MNIST dataset. To be consistent with the MNIST dataset and for the purpose of the following experiments, we reduce the size of training, validation and testing set of the affNIST dataset.
Comparing against standard data augmentation We compare ITN with other baselines + standard data augmentation. The data augmentation we applied in the following experiments is the standard data augmentation that includes affine transformations, such as rotation, translation, scaling and shear. The range of the data augmentation parameters are equivalent to the learned \( \sigma \) in Eqn.(8), which ensures the fair comparison between ITN and data augmentation. As shown in Table 1, ITN outperforms all other methods with data augmentation.

Combining with standard data augmentation We investigate the compatibility of our framework with standard data augmentations. As shown in the Table 1, combining data augmentation and ITN leads to performance improvements. This result shows the practicability of our framework as it can be jointly applied with standard data augmentation to enhance discriminators without contradictions.

Choice w.r.t. generative models We implement our framework by using AC-GAN as the generative model and name it AC-GATN. AC-GAN has the ability to generate class dependent samples, which is required in our framework. The loss function of AC-GAN is replaced with the loss function from WGAN-GP to directly compare it with ITN. All experimental settings are same for the purpose of a fair comparison. The results shown in Figure 3 clearly illustrate that under our framework, using INs as the generative model achieve better performance than using GANs. By visualizing the generated samples from AC-GATN and ITN (shown in Figure 4), both AC-GATN and ITN generate clear and sharp images. However, samples generated by AC-GATN clearly have lower quality on average in terms of the human justification as some of them are close to other objects in the dataset, i.e. the number 3 is close to number 6 in epoch 100. These lower quality samples can mislead the classifier and lead to performance decrease. Consequently, we choose INs rather than GANs in our approach. We will provide more comparisons of AC-GATN and ITN on other datasets in the supplementary material.

Effects of the update threshold \( T_u \) The update threshold \( T_u \) introduced in INs quantitatively controls the quality of samples in the generation process. In Table 2, we present

---

**Table 1.** Testing errors on MNIST and affNIST datasets, where /w DA represents the method is trained with standard data augmentation.

| Method                | MNIST | AffNIST |
|-----------------------|-------|---------|
| B-CNN (w DA)          | 0.57% | 1.65%   |
| DCGAN+B-CNN (w DA)    | 0.57% | 1.63%   |
| WGAN-GP+B-CNN (w DA)  | 0.56% | 1.56%   |
| ICN (B-CNN) (w DA)    | 0.56% | 1.54%   |
| WINN (B-CNN) (w DA)   | 0.52% | 1.48%   |
| ITN (B-CNN)           | 0.49% | 1.42%   |
| ITN (B-CNN) (w DA)    | 0.47% | 1.09%   |

---

Figure 2. Images generated by ITN. Each row from top to bottom represents the images generated on MNIST, affNIST, SVHN and CIFAR-10 dataset.

Figure 3. Testing errors of AC-GATN (B-CNN) and ITN (B-CNN) on the MNIST dataset.

Figure 4. Samples generated by AC-GATN (B-CNN) and ITN (B-CNN) on the MNIST dataset.
the results of ITN on MNIST dataset with different thresholds to explore the relationship between samples quality and threshold. Not surprisingly, we observe that the performance of ITN drops when increasing the threshold. By visualizing the samples generated by different thresholds, it is clear that the performance drops due to the decrease in the quality of generated samples (Fig. 5). Although our performance drops with the increase of the threshold, in a certain range ($<e^{-3}$), our result is still better than others shown in Table 1, which shows that our approach tolerates samples of low qualities in some extend.

| $T_u$ | 1e-3 (default) | 5e-3 | 1e-2 | 5e-2 | 1e-1 |
|-------|----------------|------|------|------|------|
| ITN error | 0.49% | 0.51% | 0.67% | 0.78% | 0.92% |

Table 2. Testing errors of ITN (B-CNN) with various thresholds on MNIST dataset.

Figure 5. Samples generated by ITN with different thresholds $T_u$. The number below each sample represents the threshold.

| Method | Error |
|-------|-------|
| B-CNN (w/ DA) | 40.74% |
| DCGAN+B-CNN (w/ DA) | 38.51% |
| WGAN-GP+B-CNN (w/ DA) | 36.29% |
| ICN (B-CNN) (w/ DA) | 35.79% |
| WINN (B-CNN) (w/ DA) | 33.53% |
| ITN (B-CNN) | 31.67% |
| ITN (B-CNN) (w/ DA) | 21.31% |

Table 3. Testing errors of the cross dataset generalization task.

Cross dataset generalization We further explore the ability of ITN to resist large variations. We design a challenging cross dataset classification task between two significantly different datasets (i.e., cross dataset generalization). The training set in this experiment is the MNIST training data while the testing set is the affNIST testing data. Even though the training and testing samples are from “different datasets”, we believe it is reasonable to consider the training and testing samples are still from the same underlying distribution. From this perspective, the difficulty of this task is how to overcome such huge data discrepancy between training and testing set since the testing set includes much more variations. As shown in the Table 3, ITN outperforms other methods with standard data augmentation.

Limited training data Another way to evaluate the ability of resisting variations is to reduce the amount of training samples. Intuitively, the data variations between the training and testing sets increase when the number of testing data remains the same while the number of samples in the training set shrinks. Thus, we implicitly increase the variations between the training and testing data by reducing the number of samples in the training data. The purpose of this experiment is to demonstrate the potential of ITN to resist unseen variations from a different perspective.

We design a new experiment where the training set is the MNIST dataset with only 0.1%, 1%, 10% and 25% of the whole training set while the testing set is the entire MNIST testing set. The reduced training set is built by randomly sampling data from the MNIST training data while keeping the number of data per class same. As shown in Table 4, our method has better results on all tasks, which are consistent with the previous results. The constantly superior performance of ITN over data augmentation indicates its effectiveness.

| Method | 0.1% | 1% | 10% | 25% |
|--------|------|----|-----|-----|
| B-CNN (w/ DA) | 18.97% | 4.48% | 1.24% | 0.83% |
| DCGAN+B-CNN (w/ DA) | 16.17% | 4.13% | 1.21% | 0.81% |
| WGAN-GP+B-CNN (w/ DA) | 15.35% | 3.98% | 1.18% | 0.79% |
| ICN (B-CNN) (w/ DA) | 15.12% | 3.74% | 1.09% | 0.80% |
| WINN (B-CNN) (w/ DA) | 14.64% | 3.66% | 1.00% | 0.77% |
| ITN (B-CNN) | 12.85% | 3.18% | 0.93% | 0.73% |
| ITN (B-CNN) (w/ DA) | 11.97% | 2.78% | 0.89% | 0.65% |

Table 4. Testing errors of the classification results with limited training data, where 0.1% means the training data is randomly selected 0.1% of the MNIST training data while the testing data is the entire MNIST testing data.

4.2. SVHN, CIFAR-10 and miniImageNet

SVHN [24] is a dataset that contains house numbers images from Google Street View. There are 73257 digits for training, 26032 digits for testing in SVHN dataset. The CIFAR-10 dataset [16] consists of 60000 color images of size $32 \times 32$. This set of 60000 images is split into two sets, 50000 images for training and 10000 images for testing. In this section, we also use ResNet-32 [8] as a baseline backbone to validate the performance of our framework with deeper network architectures, following the setting in [20]. ITN outperforms other methods on SVHN and CIFAR-10 datasets as shown in Table 5. Some samples generated by ITN are shown in Figure 2.

| Method | SVHN | CIFAR-10 |
|--------|------|----------|
| B-CNN (w/DA) | 7.01% | 24.35% |
| ResNet-32 (w/DA) | 4.03% | 7.51% |
| DCGAN + ResNet-32 (w/DA) | 3.87% | 7.17% |
| WGAN-GP + ResNet-32 (w/DA) | 3.81% | 7.05% |
| ICN (ResNet-32) (w/DA) | 3.76% | 6.70% |
| WINN (ResNet-32) (w/DA) | 3.68% | 6.43% |
| ITN (ResNet-32) | 3.47% | 6.08% |
| ITN (ResNet-32) (w/DA) | 3.32% | 5.82% |

Table 5. Testing errors on SVHN and CIFAR-10 datasets.

Extending to more challenging dataset We further verify the scalability of ITN by evaluating our proposed method on a new dataset named miniImageNet [35,
MiniImageNet dataset is a modified version of the ILSVRC-12 dataset [29], in which 600 images for each of 100 classes were randomly chosen to be part of the dataset. All images in this dataset are of size $84 \times 84$ pixels. The results are shown in Table 6 and ITN shows consistent better performance than all other comparisons.

| Method                  | Error          |
|-------------------------|----------------|
| ResNet-32 (w/DA)        | 35.25%         |
| DCGAN + ResNet-32 (w/DA)| 33.06%         |
| WGAN-GP + ResNet-32 (w/DA)| 33.42%         |
| ICN (ResNet-32) (w/DA)  | 32.87%         |
| WINN (ResNet-32) (w/DA) | 32.18%         |
| ITN (ResNet-32)         | 31.56%         |
| ITN (ResNet-32) (w/DA)  | 29.65%         |

Table 6. Testing errors on the miniImageNet dataset.

### 4.3. Beyond Spatial Transformer

Even though we utilize STs to demonstrate our ability to resist data variations, our method also has the ability to generalize to other types of transformations. Our algorithm can take other types of differentiable transformation functions and strengthen the discriminators in a similar manner. Moreover, our algorithm can utilize multiple types of transformation functions at the same time and provide even stronger ability to resist mixed variations simultaneously. To verify this, we introduce another recently proposed work, Deep Diffeomorphic Transformer (DDT) Networks [3, 4]. DDTs are similar to STs in a way that both of them can be optimized through standard back-propagation.

We replace the ST modules with the DDT modules and check whether our algorithm can resist such type of transformation. Then, we include both STs and DDTs in our model and verify the performance again. Let MNIST dataset be the training set of the experiments while the testing sets are the MNIST dataset with different types of transformation applied. We introduce two types of testing sets in this section. The first one is the normal testing set with random DDT transformations and strengthen the discriminators in a similar manner. DDTs are similar to STs in a way that both of them can be optimized through standard back-propagation.

We replace the ST modules with the DDT modules and check whether our algorithm can resist such type of transformation. Then, we include both STs and DDTs in our model and verify the performance again. Let MNIST dataset be the training set of the experiments while the testing sets are the MNIST dataset with different types of transformation applied. We introduce two types of testing sets in this section. The first one is the normal testing set with random DDT transformations. The second one is similar to the first one but includes both random DDT and affine transformations. The DDT transformation parameters are drawn from $N(0, 0.7 \times I_d)$ as suggested in [3], where $I_d$ represents the $d$ dimensional identity matrix. Then the transformed images are randomly placed in a $42 \times 42$ images. We replicate the same experiment on the CIFAR-10 dataset.

**Agnostic to different transformation functions**

We can observe from Table 7 that ITN-V1 (B-CNN) improves the discriminator performance by 4.35% from WINN (B-CNN) on MNIST with random DDT transformations and by 21.81% on CIFAR-10 dataset. In other words, ITN successfully resists DDT type of variations by integrating with DDT transformation function. Together with results from Table 1, we see that ITN has the ability to combine with different types of transformation function and resists the corresponding type of variations.

**Integrating multiple transformation functions**

Another important observation from Table 7 is that ITN-V2 (B-CNN) can utilize multiple transformations at the same time to resist a mixture of corresponding variations. Comparing against ITN-V1 (B-CNN), ITN-V2 (B-CNN) reduces the testing errors by 6.23% on MNIST dataset with random DDT + ST type of variations. Additionally, it reduces the testing errors by 6.61% on CIFAR-10 dataset.

More importantly, the performance of ITN does not degrade when the model has transformation functions that does not match the type of variations in the testing data, e.g. ITN-V2 (B-CNN) on testing data with DDT only. From this observation, we conclude that applying extra transformation functions in the ITN will not degrade the performance even though the testing data does not have such transformations. The reason to this observation is that ITN generates different transformations in every iterations, which helps it avoid over-reliance on a particular transformation. After correctly classifying original samples and transformed samples, ITN can model more complicated data distributions than other methods.

Ideally, we always want to apply the class of transformation functions that covers all possible data variations in the testing data. But we don’t know the types of transformation exist in the testing data in practice. Therefore a combination of common transformation functions is a general approach to resist the unknown variations in the testing data.

### 5. Conclusion

We proposed a principled and efficient approach that endows the classifiers with the ability to resist larger variations between training and testing data. Our method, ITN, strengthens the classifiers by generating unseen variations with various learned transformations. Experimental results show consistent performance improvements not only on the classification tasks but also on the other challenging classification tasks, such as cross dataset generalization. Moreover, ITN demonstrates its advantages in both effectiveness and efficiency over data augmentation. Our future work includes applying our approach to large scale datasets and extending it to generate samples with more types of variations.
References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. arXiv preprint arXiv:1701.07875, 2017. 1, 2
[2] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In Advances in neural information processing systems, pages 1486–1494, 2015. 2
[3] N. S. Dutta, O. Freifeld, and S. Haurberg. Deep diffeomorphic transformer networks. Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 8
[4] O. Freifeld, S. Haurberg, K. Batmanghelich, and J. W. Fisher. Transformations based on continuous piecewise-affine velocity fields. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017. 8
[5] J. Friedman, T. Hastie, and R. Tibshirani. The elements of statistical learning, volume 1. Springer series in statistics New York, NY, USA:, 2001. 2
[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014. 1, 2
[7] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In Advances in Neural Information Processing Systems, pages 5769–5779, 2017. 1, 2, 3, 5
[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016. 1, 7
[9] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016. 1, 7
[10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015. 5
[11] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. arXiv preprint, 2017. 1, 2
[12] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In Advances in neural information processing systems, pages 2017–2025, 2015. 5
[13] T. Jehbara. Machine learning: discriminative and generative, volume 755. Springer Science & Business Media, 2012. 2
[14] L. Jin, J. Lazarow, and Z. Tu. Introspective classification with convolutional nets. In Advances in Neural Information Processing Systems, pages 823–833, 2017. 1, 2, 3, 5
[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014. 5
[16] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. 7
[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012. 1
[18] J. Lazarow, L. Jin, and Z. Tu. Introspective neural networks for generative modeling. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2774–2783, 2017. 1, 2
[19] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4):541–551, 1989. 1
[20] K. Lee, W. Xu, F. Fan, and Z. Tu. Wasserstein introspective neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. 1, 2, 3, 5, 7
[21] P. Liang and M. I. Jordan. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In Proceedings of the 25th international conference on Machine learning, pages 584–591. ACM, 2008. 2
[22] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083, 2017. 1
[23] M. Mirza and S. Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784, 2014. 1
[24] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In NIPS workshop on deep learning and unsupervised feature learning, volume 2011, page 5, 2011. 7
[25] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 2642–2651, JMLR. org, 2017. 2
[26] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015. 1, 2, 5
[27] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. 2018. 5
[28] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. arXiv preprint arXiv:1803.00676, 2018. 8
[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3):211–252, 2015. 8
[30] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In Advances in Neural Information Processing Systems, pages 2234–2242, 2016. 1, 2
[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. 1
[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. CVPR, 2015. 1
[33] Z. Tu. Learning generative models via discriminative approaches. In Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on, pages 1–8. IEEE, 2007. 2, 3
[34] Z. Tu, K. L. Narr, P. Dollár, I. Dinov, P. M. Thompson, and A. W. Toga. Brain anatomical structure segmentation by hybrid discriminative/generative models. IEEE transactions on medical imaging, 27(4):495–508, 2008.

[35] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In Advances in Neural Information Processing Systems, pages 3630–3638, 2016.

[36] M. Welling, R. S. Zemel, and G. E. Hinton. Self supervised boosting. In Advances in neural information processing systems, pages 681–688, 2003.

[37] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. arXiv preprint, 2017.