AdaTransform: Adaptive Data Transformation

Zhiqiang Tang
Rutgers University
zhiqiang.tang@rutgers.edu

Xi Peng
University of Delaware
xipeng@udel.edu

Tingfeng Li
Rutgers University
tingfeng.li1@rutgers.edu

Yizhe Zhu
Rutgers University
yizhe.zhu@rutgers.edu

Dimitris Metaxas
Rutgers University
dnm@cs.rutgers.edu

Abstract

Data augmentation is widely used to increase data variance in training deep neural networks. However, previous methods require either comprehensive domain knowledge or high computational cost. Can we learn data transformation automatically and efficiently with limited domain knowledge? Furthermore, can we leverage data transformation to improve not only network training but also network testing? In this work, we propose adaptive data transformation to achieve the two goals. The AdaTransform can increase data variance in training and decrease data variance in testing. Experiments on different tasks prove that it can improve generalization performance.

1. Introduction

The remarkable success of deep learning, from the data perspective, benefits from the capability of optimizing millions of free parameters [10, 11] to capture extensive data variance. Yet, sufficient data varieties are not always available in practice due to data scarcity and annotation cost [33].

The technique of perturbing data without changing class labels, also known as *data augmentation*, is widely used to address this issue. Generally speaking, data augmentation can be either sampled from predefined distributions or generated by learnable agents. The former, known as *random augmentation* [5, 8], usually relies on hand-craft rules without optimization, yielding insufficient training. The latter, known as *auto or adversarial augmentation* [19, 27, 16], also suffers from various limitations.

Auto augmentation [19] explores a huge solution space to achieve an optimal solution on the validation set, which is extremely time-consuming. The network training has to be repeated $15,000$ times to get the final policy. *Adversarial augmentation*, on the other hand, follows a greedy design to speed up learning. However, the current designs [27, 16] rely on comprehensive domain knowledge to specify the transformation types and boundaries. This inevitably results in restricted transformation space. Moreover, previous methods mainly focus on network training, neglecting the potential to apply data transformation in testing.

This raises research questions: 1) Can we learn data transformation more efficiently? 2) Can we explore the transformation space (types and boundaries) without comprehensive domain knowledge? 3) Can data transformation also help improve network deploying?

In this paper, we answer the questions by proposing AdaTransform: adaptive data transformation. We leverage reinforcement learning in conjunction with adversarial training to compose meta-transformations (discrete transformation operations). This enables us to efficiently explore a large transformation space with limited domain knowledge.

Specifically, we learn data transformation in bi-direction: At the training stage, AdaTransform performs *a competitive task to increase data variance*, reducing over-fitting; at the testing stage, AdaTransform performs *a cooperative task to decrease data variance*, yielding improved deploying. The two tasks are learned through optimizing a triplet: a transformer, a discriminator, and a target network, as illustrated in Figure 1. To summarize, our key contributions are:

- To the best of our knowledge, we are the first to investigate adaptive data transformation in order to improve both network training and testing.
- We propose to learn a competitive task (for training) and a cooperative task (for testing) simultaneously by jointly optimizing a triplet online.
- AdaTransform can automatically and efficiently explore the data transformation space, yielding a highly flexible and versatile solution for broad applications.
- Extensive experiments on *image classification*, *human pose estimation*, and *face alignment* prove the favorable performance of AdaTransform especially when testing perturbations exist.

2380-7504/19/$31.00 ©2019 IEEE
DOI 10.1109/ICCV.2019.00309

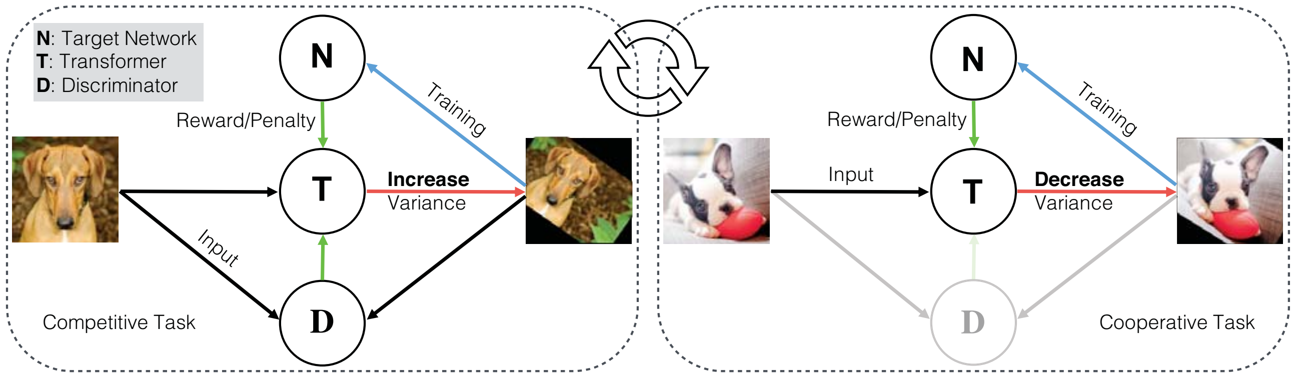


Figure 1: Overview of the adaptive data transformation. It consists of two tasks: competitive training and cooperative testing, and three components: a transformer T, a discriminator D, and a target network N. T increases the training data variance by competing with both D and N. It also cooperates with N in testing to reduce the data variance.

2. Related Work

We provide a brief overview of related works in the categories of data transformation, adversarial learning, reinforcement learning, hard example mining, human pose estimation, and face alignment.

Data transformation. Data transformations are commonly used to augment the training data [10, 8]. Recently, the adversarial data augmentations [27, 16] are proposed. But they heavily rely on human knowledge and can only handle limited transformations. Some works [4, 19] try to learn the data augmentation policy automatically. However, either they suffer from severe efficiency issue [4] or the policy learning is isolated with the target network training [19]. The high computational cost is due to the optimization of validation accuracy. The lack of joint optimization with the target network prevents it from dynamically increasing the data variance based on the individual images and target network state. Others [21, 2] learn to transfer the data transformations from large datasets to augment few-shot examples. The above methods are only to augment training data but cannot reduce the testing data variance. The Spatial Transformer Network (STN) [12] is designed to reduce the spatial variance of data. However, it can only handle differentiable spatial transformations, largely restricting its applications. Besides, it is only for the variance reduction but cannot increase the training data variance.

Adversarial learning. Generative Adversarial Networks (GANs) [9] includes two networks: generator and discriminator which compete against each other to improve generation performance. GANs are widely used in the image generations [9, 37] and translations [36]. Here we use the transformer to transform input images. It competes with the discriminator to make the transformed images still realistic but different from the original ones.

Reinforcement learning. In reinforcement learning (RL), an agent takes actions and then receives feedback from the environment, which may reward or penalize it. The agent learns to maximize its reward by taking appropriate actions. Reinforcement learning has been used with deep learning to play the Go game [23], search the neural network architecture [17], etc. In this paper, we use it to learn the transformer to handle data transformations.

Hard example mining. Hard example mining usually alternates between optimizing models and updating training data. Once a model is optimized on the current training set, it is used to collect more hard data for further training. This method was used in training SVM models for object detection [26]. Recently, Shrivastava *et al.* [22] adapted it into the neural network based object detector. The hard example mining focuses on selecting hard examples from existing data, the adaptive data transformation actively transforms the data to either increase or reduce their variance.

Human pose estimation. With recent advances in Deep Neural Networks (DNNs), image-based human pose estimation has achieved significant progress in the past few years [25, 24, 3]. DeepPose [25] is one of the first attempts of using DNNs for human pose estimation. Recently, multi-stage human pose prediction methods such as Convolutional Pose Machine [28] and stacked hourglasses [15] have become popular. The prediction results could be refined stage-by-stage. Instead of designing a new pose estimator, we improve pose estimation performance by increasing the training data variance and reducing the testing data variance.

Face alignment. Similarly, DNNs have largely reshaped the field of face alignment. Traditional methods like [29] could be easily outperformed by the DNNs based [34, 14]. In the recent Menpo Facial Landmark Localization Challenge [31], stacked hourglasses [15] achieves state-of-the-art performance. Given an off-the-shelf face alignment DNN, the adaptive data transformation can be used to improve its performance.

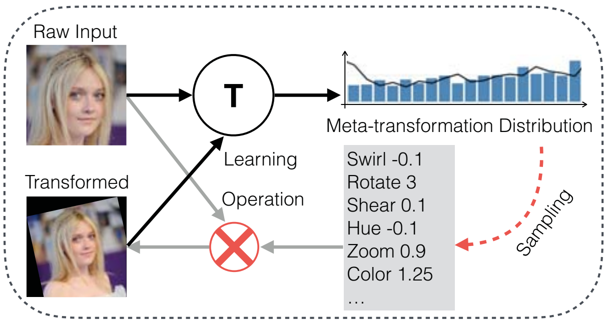


Figure 2: Incremental transformation. The transformer, conditioning on the input, outputs the distribution over the meta-transformations. A meta-transformation is sampled and transforms the input. Then the transformed data become the input and continue to be transformed.

3. Problem Definition and Task Modeling

Given a target network, *e.g.* an image classifier [10] or a human pose estimator [24], etc, the adaptive data transformation, named as **AdaTransform**, aims to improve both training and testing of the target network. More specifically, the agent performs two different tasks: (1) At the training stage, it performs a **competitive task** to increase data variance, improving the training of the target network. (2) At the testing stage, it performs a **cooperative task** to reduce data variance, boosting its testing performance. The two tasks are learned simultaneously by jointly optimizing a triplet: a transformer T, a discriminator D, and a target network N. An illustration is given in Figure 1.

3.1. Transformer T

The transformer T is designed to increase the data variance in the competitive task, while it learns to decrease the data variance in the cooperative task.

Transformation Definition. Transformation is domain-specific. It relies on both the data type and the target problem. Data of different modalities have dissimilar transformations. For example, images can utilize scale and rotation, while word replacement and switch may happen in text data.

Further, a transformation must preserve the data property of interest in the target problem. For instance, the shear operation can be applied in image classification since it does not change the image class labels. However, it is not a good choice for face recognition as it may alter the identity. The AdaTransform only needs limited domain knowledge to specify some meta-transformations. Then T learns to compose them for both competitive and cooperative tasks.

Competitive task. T learns to enlarge the data variance in training through increasing the loss of target network N. At the same time, it tries to fool the discriminator D by making the transformed data realistic. Thus, T must learn to satisfy the constraints from both N and D:

$$\max_{\theta_T} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim\Omega} \mathop{\mathbb{E}}_{\tau\sim T(\mathbf{x},0)} [\mathcal{L}(N(\tau(\mathbf{x})),\mathbf{y}) + \lambda \log(D(\tau(\mathbf{x})))], \tag{1}$$

where Ω is the training data, and τ is the transformation operation sampled from $T(x,0)$ in the competitive mode. $\mathcal{L}(\cdot,\cdot)$ is a predefined target loss function. λ balances the weight of two losses. T competes with both N and D in the competitive task. The competitive T is trained and applied to the training data.

Cooperative task. T also learns to reduce the data variance by lowering the loss of target network N:

$$\min_{\theta_T} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim\Omega} \mathop{\mathbb{E}}_{\tau\sim T(\mathbf{x},1)} [\mathcal{L}(N(\tau(\mathbf{x})),\mathbf{y})]. \tag{2}$$

where 1 indicates the cooperative mode of T. The discriminator D is not used in the cooperative task. Because the transformed data of reduced variance can hardly fall out of the real data distribution. T cooperates with N in the cooperative task. The cooperative T is trained on the training data and generalized to the testing data.

3.2. Discriminator D

The discriminator D aims to control the variance of transformed data. It learns to assign low scores to out-of-distribution transformed data and high scores to in-distribution data. To this end, D learns from both the original and transformed data as follows:

$$\max_{\theta_D} \mathop{\mathbb{E}}_{\mathbf{x}\sim\Omega} \mathop{\mathbb{E}}_{\tau\sim T(\mathbf{x})} [\log(1-D(\tau(\mathbf{x})))] + \mathop{\mathbb{E}}_{\mathbf{x}'\sim\Omega} [\log(D(x'))]. \tag{3}$$

D competes with the transformer T in the competitive task. It is a critical design to automate competitive training. Human users can be saved from the heavy burden of specifying the transformation boundaries, especially when multi-types of transformations are available. Without D, T would probably produce out-of-distribution transformations.

3.3. Target Network N

The goal of target network N is to generalize well on the testing data. The training data usually have some distribution shift from the testing data. The current neural networks are so powerful that they can easily overfit the training data. The transformer T can reduce the overfitting by adaptively increasing the training data variance. N learns from both the original and the transformed training data as follows:

$$\min_{\theta_N} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim\Omega} \mathop{\mathbb{E}}_{\tau\sim T(\mathbf{x})} [\mathcal{L}(N(\tau(\mathbf{x})),\mathbf{y}) + \mathcal{L}(N(\mathbf{x}),\mathbf{y})], \tag{4}$$

The target network N competes with the transformer T through learning from its transformed data.

Algorithm 1: Mini batch training of transformer T

```
Input: Mini-batch $B$, triplet $T$, $D$, and $N$.
Output: Transformer $T$
1 Replicate $B$ $s$ times to get $\mathbf{X}$ of size $M$;
2 Apply $T$ on $\mathbf{X}$ to get $\hat{X}$ and polices $\{\pi_t^i\} \in \mathbb{R}^{M\times K}$;
3 Compute rewards $\{r_t^i\} \in \mathbb{R}^{M\times K}$ of $\hat{X}$ by Eq. 5 and 6;
4 Get accumulated rewards $\{R_t^i\} \in \mathbb{R}^{M\times K}$ by Eq. 8;
5 Normalize $\{R_t^i\}$ to $\{\bar{R}_t^i\}$ by Eq. 9 and Eq. 10;
6 Call the gradient ascent on the sum of $\{\bar{R}_t^i \log \pi_t^i\}$;
```

4. Learning Strategy

The triplet T, D, and N are jointly learned in the adaptive data transformation. The main challenge comes from learning T since many transformation operations are not differentiable. The gradients cannot flow to T directly from D and N. To deal with this issue, we use reinforcement learning with meta-transformations to train T.

4.1. Meta-transformation

The meta-transformations define the small transformation operations [19]. Table 1 lists examples of meta-transformations in natural images. A large transformation can be decomposed as a combination of multiple meta-transformations. Despite some precision loss, it barely affects the target network training. Specifying the meta-transformations requires much less domain knowledge than tuning the boundaries of multi-type transformations and choosing their combinations [27, 16].

The meta-transformation offers flexibility and scalability to achieve complex transformations. We can efficiently explore an ample transformation space by traversing the combinations of meta-transforms. More importantly, the meta-transformations make it possible to train T in a tractable manner via reinforcement learning.

4.2. Reinforcement Learning Formulation

The transformer T incrementally transforms the data using meta-transformations. An illustration is shown in Figure 2. Let x and $\hat{x}$ denote the original and transformed data points. At step t, T conditioning on $\hat{x}_{t-1}$ outputs the distribution $T(\hat{x}_{t-1})$ over all the meta-transformations. Then the meta-transformation τ_t is sampled from it. The loss of transformed data $\hat{x}_t = \tau_t(\hat{x}_{t-1})$ is computed as:

$$\ell(\hat{x}_t) = \begin{cases} \mathcal{L}(N(\hat{x}_t), y) + \lambda \log(D(\hat{x}_t)), & \textit{competitive case}. \\ -\mathcal{L}(N(\hat{x}_t), y), & \textit{cooperative case}. \end{cases} \tag{5}$$

where $\mathcal{L}$ denotes the loss function for the target task and λ is the weight of the discriminator loss. In the *competitive* mode, the transformer learns to expand the data variance by increasing the target network loss. On the other hand, it also tries to keep high probabilities of transformed data being realistic. In the *cooperative* mode, the transformer learns to reduce the data variance by increasing the negative target loss, i.e., decreasing the target loss.

Algorithm 2: Joint training scheme of T, D, and N

```
Input: Training data $X$, triplet $T$, $D$, and $N$.
Output: Triplet $T$, $D$, and $N$.
1  while not end do
2      for mini batch B in X do
3          Apply $T$ on $B$ with probability $p$ to get $\hat{B}$;
4          Train $N$ with the mixed data $\hat{B}$;
5      end
6      for mini batch B in X do
7          Train competitive $T$ with $D$, $N$ by Alg. 1;
8          Train cooperative $T$ with $N$ by Alg. 1;
9      end
10 end
```

The reward r_t for meta-transformation τ_t is the incremental loss:

$$r_t = \ell(\hat{x}_t) - \ell(\hat{x}_{t-1}). \tag{6}$$

Suppose the transformer T is applied K steps, a reward sequence $\{r_1, r_2, \cdots, r_K\}$ is produced, where reward r_1 is $r_1 = \ell(\hat{x}_1) - \ell(x)$. Summing up these rewards results in

$$\sum_{t=1}^{K} r_t = \ell(\hat{x}_K) - \ell(x). \tag{7}$$

The discriminator and target network are fixed when training the transformer. Given an original data point x, $\ell(x)$ is a constant, which can be ignored. $\ell(\hat{x}_K)$ is the objective in either Equations 1 or 2 since $\hat{x}_K$ is the final transformed data point. Therefore, optimizing the objectives in Equations 1 or 2 can be converted into maximizing the sum of rewards.

We apply the policy gradients to maximize the sum of rewards. Two common techniques are used to reduce the variance in estimating the rewards. First, we compute the reward for transformation $\tau_{t'}$ as the accumulated future reward $\sum_{t>=t'}^{K} r_t$ rather than $r_{t'}$ only. A discounting factor γ is used to model the delaying effects of future rewards. Therefore, the accumulated discounted reward $R_{t'}$ is:

$$R_{t'} = \sum_{t>=t'}^{K} \gamma^{t-t'} r_t, \tag{8}$$

where we set $\gamma = 0.5$ in the experiments.

Also, the raw value of reward $R_{t'}$ may not be meaningful. Positive values do not necessarily mean rewards. We only push up the probability of a meta-transformation, if its reward is higher than the expectation. Here we use the mean

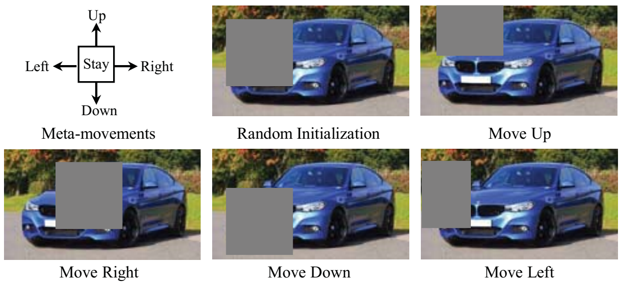


Figure 3: Meta-movements. AdaCutout/AdaErasing first samples a random mask, then moves it up, right, down, left.

of reward $R_{t'}$ within each mini-batch of h training samples as the reference. For each original data point, we sample s different rewards $R_{t'}$. Thus, the mean of $R_{t'}$ is:

$$b_{t'} = \frac{1}{h \times s} \sum_{i=1}^{h \times s} R_{t'}^{i} \tag{9}$$

Note that it is important to compute the reward mean online within the mini-batches instead of using the moving averages of all history rewards. Because the discriminator and target network become more and more powerful in training. The history rewards cannot reflect their current states well.

At step t', each reward $R_{t'}$ is normalized by subtracting its mean $b_{t'}$. A positive normalized value means reward, whereas a negative normalized one means penalty. According to the policy gradients formula, we compute the gradient of transformer T at step t':

$$\nabla_{\theta_T} T(\hat{x}_{t'}) = \sum_{i=1}^{h \times s} (R_{t'}^{i} - b_{t'}) \nabla_{\theta_T} \log \pi_{\theta_T}(\tau_{t'} | \hat{x}_{t'}), \tag{10}$$

where $\pi_{\theta_T}(\tau_{t'}|\hat{x}_{t'})$ is the policy, i.e., the probability of taking meta-transformation $\tau_{t'}$ given the input $\hat{x}_{t'}$. Updating transformer T with the gradient ascent can push up or pull down the probabilities if the corresponding meta-transformations yield rewards or penalties at step t'.

Finally, we sum up the gradients of T from all K steps:

$$\nabla_{\theta_T} T(\cdot) = \sum_{t'=1}^{K} \nabla_{\theta_T} T(\hat{x}_{t'}). \tag{11}$$

Basically, the transformer T is updated each time using the accumulated gradients from K steps and $h \times s$ samples. Algorithm 1 summarizes the training scheme of T.

4.3. Joint learning of T, D, and N

The transformer T is jointly optimized with the discriminator D and target network N during the training. The training procedure is described in Algorithm 2. More specifically, we train N for several epochs and then update T and D once. N needs to learn from both the transformed and original data. To this end, we apply T with some probability $p(0 < p < 1)$ on the training data of N.

T and D are updated alternately inside each iteration. Given a mini-batch of data, D is updated on both the original (real) and transformed (fake) data of T. Then we update T separately in the *competitive* and *cooperative* modes. T receives the feedback from N in the *cooperative* case while it requires the additional feedback from D in the *competitive* case. We add a zero or one map to the input of T as the condition of *competitive* or *cooperative* modes.

Table 1: Examples of meta-transformations in natural images. A meta-transformation defines a small operation. A combination of multiple meta-transformations can approximate a large transformation space.

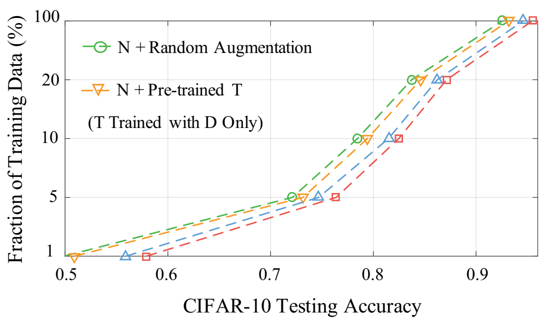
Type	Meta-values
Rotation	$2.5°, -2.5°, 5°, -5°$
Zoom	0.9x, 1.1x, , 0.75x, 1.25x
Shear/Swirl	$0.1°, -0.1°, 0.25°, -0.25°$
Hue Shift	0.1, -0.1, 0.25, -0.25
Brightness/Color	0.75, 1.25, 0.5, 1.5
Sharpness/Contrast	0.75, 1.25, 0.5, 1.5
Horizontal Flip	-

5. Applications of AdaTransform

AdaTransform provides a versatile solution for general data analytic tasks with proper domain knowledge. In this paper, we focus on its application to visual tasks.

AdaImgTransform. For natural images, there are many available transformation types such as scale, rotation, translation, flipping, swirl, shear, contrast enhancement, color enhancement, brightness enhancement, sharpness enhancement, and hue shift. Table 1 lists the corresponding meta-transformations. We can adjust the meta-transformation pool according to domain knowledge of a specific task. We apply adaptive data transformation to learning to combine the proper meta-transformations conditioning on the input image, target network state, and the transformer mode. They can be used to either increase or reduce data variance.

AdaCutout/AdaErasing. Occlusions are quite common in natural images where the object of interest is partially occluded. The cutout [6] and random erasing [35] are recently proposed to simulate the occlusions on the images. To be specific, a fixed-size square mask (cutout) or a flexible one (erasing) are used to occlude the image region centered at a randomly chosen position. We apply the adaptive transformation to control cutout or erasing. More specifically, we use random cutout or erasing for the initialization. Then the transformer learns to move the cutout mask progressively. Each step it can be moved up, right, down, left, or stay still. Figure 3 illustrates the five meta-movements.



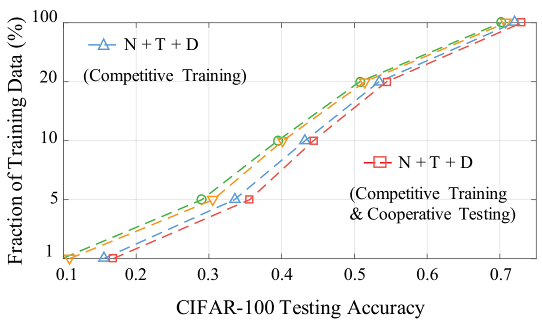


Figure 4: Validation of competitive and cooperative tasks. We show the testing accuracy with respect to the fraction of training data. The joint learning of competitive training and cooperative testing achieves the best performance (**lowest**). Its superior performance is more significant when less data is used in training (**right to left**).

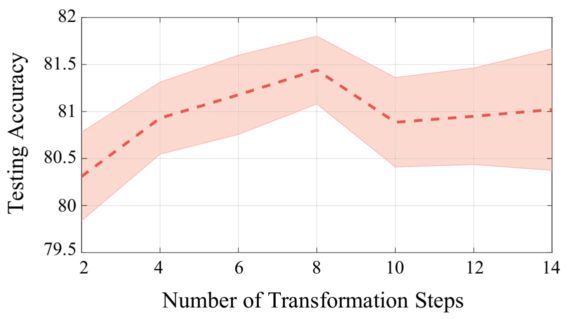


Figure 5: Effect of transformation steps. The dotted line and shaded areas represent *mean* and *std*, respectively. More steps would increase *std* (high model variance). The best trade-off between testing accuracy and model variance is obtained at 8-step.

6. Experiment

The experiments include three parts: ablation study, robustness test, and comparison with state-of-the-art methods. We evaluate AdaTransform on three different tasks: image classification, human pose estimation, and face alignment. We apply meta-transformations given in Table 1 for image classification. For the other two tasks, we remove shear and swirl due to the shifting of ground truth.

6.1. Experimental Settings

Transformer T and discriminator D. The transformer and discriminator use the common networks. More specifically, the transformer has the architecture of ResNet-18 [10]. Besides, we add the dropout layers after each 3×3 convolution layer and before the fully connected layer. The discriminator is the same as the one in DCGAN [18].

Target network N. Different tasks have their target networks. In **image classificatin**, we use the 32-layer ResNet (ResNet32) [10] in the ablation study. The comparisons with state-of-the-art data augmentation method AutoAugment [4] are based on more complex models: Wide-ResNet-28-10 [32], Shake-Shake [7] and ShakeDrop [30]. For **human pose estimation** and **face alignment**, we use the two stacked hourglasses [15] in all the experiments.

Table 2: Evaluation of AdaCutout and AdaErasing using 10% training data of CIFAR-10 and CIFAR-100.

Method	CIFAR-10	CIFAR-100
Cutout [6]	77.21	40.41
AdaCutout	**78.02**	**41.02**
Erasing [35]	77.25	40.53
AdaErasing	**78.12**	**41.21**

Hyperparameters. We use two transformers for adaptive cutout (AdaCuout) and adaptive image transformation (AdaImgTransform). The AdaCutout transformer is trained with learning rate 3e-5 and weight decay 1e-5 whereas the AdaImgTransform transformer has learning rate 1e-4 and weight decay 1e-4. AdaCutout moves the occlusion mask 2 pixels each step. We set step number $K = 3$ for AdaCutout and $K = 8$ for the AdaImgTransform. Besides, AdaCutout is applied with probability 0.3 on each mini-batch when training the target network. On the other hand, we use AdaImgTransform on all the training data but stop it for the last ten epochs.

Datasets. We use the benchmark datasets: CIFAR-10 and CIFAR-100 for image classification; MPII Human Pose [1] and Leeds Sports Pose (LSP) [13] for human pose estimation; 300-W challenge [20] for face alignment. The 300-W test set consists of easy and challenging subsets. We use the classification accuracy/error, Percentage of Correct Key points (PCK), and normalized mean error (NME) as the measurements of image classification, human pose estimation, and face alignment. In particular, MPII and LSP use PCKh@0.5 and PCK@0.2.

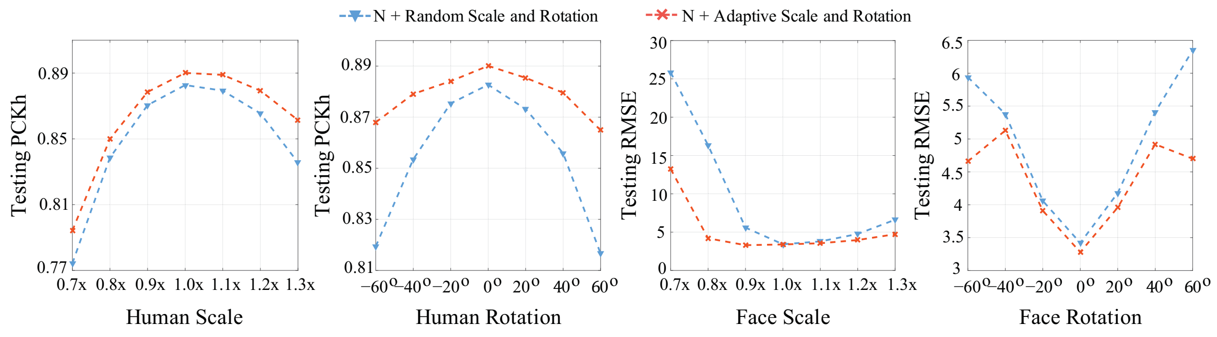


Figure 6: Robustness against rotations and scale perturbations. We investigate human pose estimation (**left two, the higher the better**) and face alignment (**right two, the lower the better**). Network (N) trained using adaptive data transformation outperform random ones with substantial margins. The performance improvements are more significant when increasing perturbations, indicating the effectiveness in learning more robust models.

Table 3: Effect of different types of adaptive transformation in human pose estimation. We report per-joint PCKh (%). A single kind of adaptive transformation would improve the performance compared with randomly performed. Jointly applying all transformations has the best performance.

Method	Head	Sho.	Elb.	Wri.	Hip	Knee	Ank.	Mean
RandomAugment.	95.7	95.0	89.1	83.4	88.2	84.0	80.2	88.1
AdaImgTexture	95.3	95.3	89.7	84.8	89.0	84.9	80.9	88.7
AdaCutout	95.5	95.2	89.7	84.6	88.5	84.7	80.9	88.6
AdaScaleRotation	95.5	95.6	89.8	85.0	89.4	84.7	80.8	88.9
AdaAll	**95.8**	**96.0**	**90.1**	**85.4**	**89.8**	**85.7**	**81.3**	**89.3**

6.2. Ablation Study

Effect of transformation steps. The transformer incrementally transforms an image for several steps. It is interesting to observe how test accuracy changes with the step number. We train 6 models for each step number using 10% CIFAR-10 training data. Figure 5 shows the *mean* and *std* of the testing accuracy. A modest increase of step number can produce more complex transformations, increasing testing accuracy. However, more transformation steps are difficult to learn and result in high model variance.

Validation of competitive and cooperative tasks. We incrementally add each component and observe the changes in test accuracy. Figure 4 gives the comparison of four variants. They all use eight transformation steps and the same meta-transformation pool in Table 1. The competitive training and cooperative testing can both increase test accuracy with different percentages of training data. In the case of only 1% training data, the competitive training can improve $\sim$5% accuracy on both CIFAR-10 and CIFAR-100 over the pre-trained transformer, indicating the importance of joint training with the target network. The cooperative testing, on the other hand, further brings $\sim$2% gain for the two datasets. Even with 100% training data, they can separately get $\sim$1% improvements on both datasets.

Table 4: Robustness against texture (color, brightness, contrast, sharpness, and hue) perturbations. We investigate standard (**top two rows**) and perturbed (**bottom two rows**) testing. In particular, AdaImgTexture is more robust against texture perturbations.

Method	Head	Sho.	Elb.	Wri.	Hip	Knee	Ank.	Mean
RandomAugment.	**95.7**	95.0	89.1	83.4	88.2	84.0	80.2	88.1
AdaImgTexture	95.3	**95.3**	**89.7**	**84.8**	**89.0**	**84.9**	**80.9**	**88.7**
RandomAugment.	94.4	93.9	86.9	81.5	86.7	82.0	77.0	86.3
AdaImgTexture	**94.9**	**94.9**	**88.5**	**83.2**	**88.2**	**83.6**	**79.7**	**87.8**

Evaluation of AdaCutout and AdaErasing. Apart from the above AdaImgTransform, we also evaluate AdaCutout and AdaErasing. The results are given in Table 2. Cutout and random erasing obtain similar accuracy. AdaCutout and AdaErasing can both improve the baselines.

Effect of different types of adaptive transformation. We categorize the transformations into three groups: spatial variations (scale and rotation), occlusion (Cutout [6]), and texture changes (image color, brightness, contrast, sharpness, and hue). It may be interesting to study their separate contributions. AdaTransform can utilize them both independently and jointly. Table 3 gives the results on human pose estimation. The spatial transformations bring more improvement (0.8%) than the other two (0.5% and 0.6%), indicating its importance in human pose estimation.

6.3. Robustness Test

In the traditional test, testing images are usually static with no perturbations. However, in practice, an image may be affected by many factors, such as scale and rotation. A robust model should handle well not only the original image but also its variants under reasonable perturbations. In this experiment, we test models under the condition of different scales, rotations, and texture variations of testing data. To

Table 5: Comparison with AutoAugment [4] in terms of image classification errors. AdaTransform has comparable performance with all the three classifiers. However, it is much more efficient than AutoAugment.

Model	CIFAR-10		CIFAR-100	
	AutoAug.	Ours	AutoAug.	Ours
Wide-ResNet [32]	**2.68**	2.95	**17.09**	17.42
Shake-Shake[7]	**1.99**	2.11	**14.28**	15.01
ShakeDrop[30]	**1.48**	1.72	**10.67**	11.21

Table 6: Comparison with adversarial data augmentation [16] in human pose estimation. We use two stacked hourglasses and report PCKh@0.5 on MPII validation set (**top**) and PCK@0.2 on LSP test set (**bottom**).

Method	Head	Sho.	Elb.	Wri.	Hip	Knee	Ank.	Mean
AdvAug. [16]	**96.5**	95.5	89.8	84.5	89.4	85.0	80.7	88.9
AdaTransform	95.8	**96.0**	**90.1**	**85.4**	**89.8**	**85.7**	**81.3**	**89.3**
AdvAug. [16]	96.8	93.7	90.9	**88.0**	92.0	93.7	92.4	92.5
AdaTransform	**96.9**	**94.1**	**91.0**	87.8	**93.0**	**94.5**	**93.3**	**92.9**

Table 7: Comparison with adversarial data augmentation [16] in face alignment (NME) on 300-W dataset.

Method	Easy Subset	Hard Subset	Full Set
AdvAug. [16]	2.87	4.98	3.28
AdaTransform	**2.82**	**4.96**	**3.24**

evaluate the robustness of AdaTransform, we compare the models trained with it and random augmentation.

Robustness against scale and rotation perturbations. Figure 6 shows the robustness comparisons in two tasks. AdaTransform can consistently improve testing performance over a range of scales and rotations, especially at the ends. In human pose estimation, we observe $\sim$3% accuracy increase for scales 0.7/1.3 and $\sim$5% increase for rotations $-60°/60°$. For face alignment, the large error drops $\sim$12% and $\sim$2% happen at scale 0.7 and rotations $-60°/60°$.

Robustness against texture perturbations. To get reasonable texture perturbations, we train transformer with only discriminator using CIFAR-10. During testing, we use 15 trained transformer models to perturb the testing images. Table 4 gives the robustness comparisons with random augmentation. AdaTransform can get higher PCKh on both the standard test and test with texture perturbations. Moreover, the PCKh gap 1.5% in the perturbed test is much larger than the 0.6% in the standard test.

6.4. Comparison with State-of-the-art Methods

Image classification. We first compare AdaTransform (AdaImgTransform + AdaCutout) with state-of-the-art AutoAugment [4]. Table 5 shows the comparisons on both

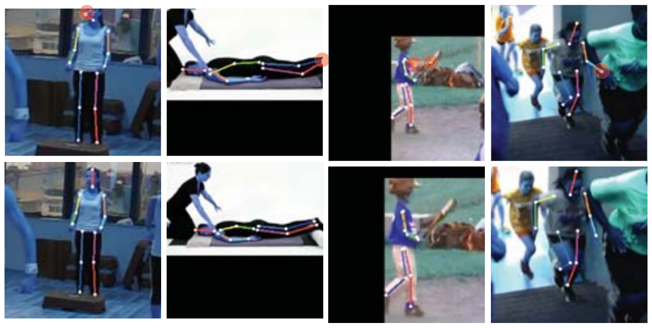


Figure 7: Cooperative zoom-out (**left**) and zoom-in (**right**) in human pose estimation. False positives, marked by red circles, are detected on the original scales (**top**). Zoom-out (**bottom**) can help detect the joints, such as head, wrist, and ankle, falling out of scope in the original scale. Zoom-in (**bottom**), on the other hand, can reduce the ambiguity sometimes caused by the background noise.

CIFAR-10 and CIFAR-100. AdaTransform obtains comparable performance as the AutoAugment. However, it needs to train only three models. In contrast, the AutoAugment requires to train fifteen thousand models to search the final augmentation policy. Although each model in AdaTransform may take longer to train, it is still much more efficient.

Please note that the AutoAugment cannot work if only training several models. It is a purely reinforcement-based method, optimizing the validation error. The trained model number represents its search space. AdaTransform, on the other hand, integrates the adversarial training with reinforcement learning, optimizing the training loss.

Human pose estimation. We also compare AdaTransform with state-of-the-art adversarial data augmentation [16] on human pose estimation. Table 6 gives the comparisons based on two stacked hourglasses [15]. AdaTransform obtains %0.4 mean improvements on both datasets. AdaTrasnform can search a larger transformation space by composing multi-type meta-transformations.

Face alignment. AdaTransform and state-of-the-art adversarial data augmentation [16] can both apply to face alignment. We use two stacked hourglasses as the target network. The results are shown in Table 7. AdaTransform obtains %0.05 and %0.02 lower errors on the easy and challenging subsets respectively.

7. Conclusion

We have proposed AdaTransform to manipulate data variance in bi-direction at the training and testing stages. It can be learned efficiently by jointly optimizing a triplet online. Experimental results on three different tasks: image classification, human pose estimation, and face alignment demonstrate its superior performance in network training and testing especially when perturbations exist.

References

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2D human pose estimation: New benchmark and state of the art analysis. In *CVPR*, 2014.

[2] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv*, 2017.

[3] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *CVPR*, 2016.

[4] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv*, 2018.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[6] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv*, 2017.

[7] Xavier Gastaldi. Shake-shake regularization. *arXiv*, 2017.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[11] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *CVPR*, 2017.

[12] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NIPS*, 2015.

[13] Sam Johnson and Mark Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *BMVC*, 2010.

[14] Jiangjing Lv, Xiaohu Shao, Junliang Xing, Cheng Cheng, and Xi Zhou. A deep regression architecture with two-stage re-initialization for high performance facial landmark detection. In *CVPR*, 2017.

[15] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[16] Xi Peng, Zhiqiang Tang, Fei Yang, Rogerio S Feris, and Dimitris Metaxas. Jointly optimize data augmentation and network training: Adversarial data augmentation in human pose estimation. In *CVPR*, 2018.

[17] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv.*, 2018.

[18] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv*, 2015.

[19] Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. In *NIPS*, 2017.

[20] Christos Sagonas, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *ICCVW*, 2013.

[21] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Rogerio Feris, Abhishek Kumar, Raja Giryes, and Alex M Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In *NIPS*, 2018.

[22] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016.

[23] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 2017.

[24] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*, 2014.

[25] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014.

[26] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *IJCV*, 2013.

[27] Xiaolong Wang, Abhinav Shrivastava, and Abhinav Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. *arXiv.*, 2017.

[28] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016.

[29] Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *CVPR*, 2013.

[30] Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise. Shakedrop regularization for deep residual learning. *arXiv*, 2018.

[31] Stefanos Zafeiriou, George Trigeorgis, Grigorios Chrysos, Jiankang Deng, and Jie Shen. The menpo facial landmark localisation challenge: A step towards the solution. In *CVPRW*, 2017.

[32] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv*, 2016.

[33] Yong Zhang, Baoyuan Wu, Weiming Dong, Zhifeng Li, Wei Liu, Bao-Gang Hu, and Qiang Ji. Joint representation and estimator learning for facial action unit intensity estimation. In *CVPR*, 2019.

[34] Zhanpeng Zhang, Ping Luo, Chen C. Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *ECCV*, 2014.

[35] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *arXiv*, 2017.

[36] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *ICCV*, 2017.

[37] Yizhe Zhu, Mohamed Elhoseiny, Bingchen Liu, Xi Peng, and Ahmed Elgammal. A generative adversarial approach for zero-shot learning from noisy texts. In *CVPR*, 2018.