OnlineAugment: Online Data Augmentation with Less Domain Knowledge

Zhiqiang Tang[1]*, Yunhe Gao[1], Leonid Karlinsky[2], Prasanna Sattigeri[2], Rogerio Feris[2], and Dimitris Metaxas[1]

[1] Rutgers University, {zhiqiang.tang, yunhe.gao, dnm}@rutgers.edu
[2] IBM Research AI, {LEONIDKA@il, psattig@us, rsferis@us}.ibm.com

Abstract. Data augmentation is one of the most important tools in training modern deep neural networks. Recently, great advances have been made in searching for optimal augmentation policies in the image classification domain. However, two key points related to data augmentation remain uncovered by the current methods. First is that most if not all modern augmentation search methods are *offline* and learning policies are isolated from their usage. The learned policies are mostly constant throughout the training process and are *not adapted* to the current training model state. Second, the policies rely on class-preserving image processing functions. Hence applying current offline methods to new tasks may require domain knowledge to specify such kind of operations. In this work, we offer an orthogonal *online* data augmentation scheme together with three new augmentation networks, co-trained with the target learning task. It is both more efficient, in the sense that it does not require expensive offline training when entering a new domain, and more adaptive as it adapts to the learner state. Our augmentation networks require less domain knowledge and are easily applicable to new tasks. Extensive experiments demonstrate that the proposed scheme alone performs on par with the state-of-the-art offline data augmentation methods, as well as improving upon the state-of-the-art in combination with those methods. Code is available at https://github.com/zhiqiangdon/online-augment.

1 Introduction

Data augmentation is widely used in training deep neural networks. It is an essential ingredient of many state-of-the-art deep learning systems on image classification [33,21,9,42], object detection [14,8], segmentation [12,32,39], as well as text classification [40]. Current deep neural networks may have billions of parameters, tending to overfit the limited training data. Data augmentation aims to increase both the quantity and diversity of training data, thus alleviates overfitting and improves generalization.

Traditionally, data augmentation relies on hand-crafted policies. Designing the polices is usually inspired by domain knowledge and further verified by testing performance [34,17]. For example, the typical routine in training CIFAR

* Corresponding author: Zhiqiang Tang

classifiers uses random cropping and horizontal flip to conduct data augmentation. Intuitively, these operations do not change the image labels, and they can also improve testing performance in practice.

Recently, AutoML techniques [45,2] are used to automate the process of discovering augmentation polices. The resulted approaches, such as AutoAugment [6] and its variants [21,44,24,18] are quite successful and achieve state-of-the-art results. We name them *offline data augmentation* since the policy learning and usage are isolated. Moreover, these approaches use pre-specified image processing functions as augmentation operations. Defining the basic operations requires domain knowledge, which may impede their applications to more tasks.

In this paper, we propose *OnlineAugment*, which jointly optimizes data augmentation and target network training in an online manner. The merits of OnlineAugment lie in three-fold. First, it is orthogonal to the offline methods. Their complementary nature makes it possible to apply them together. Second, through the online learning, the augmentation network can adapt to the target network through training from the start to the end, saving it from the inconveniences of pre-training [23] or early stopping [29]. Third, it is easy to implement and train OnlineAugment. In contrast, learning offline policies usually rely on distributed training, as there are many parallel optimization processes.

Furthermore, we propose more general data augmentation operations with less domain knowledge. Instead of using pre-defined image processing functions, such as rotation and color, we design neural networks to perform data augmentation. Specifically, we devise three learnable models: augmentation STN (A-STN), deformation VAE (D-VAE), and Perturbation VAE (P-VAE). It is nontrivial to craft STN [19] and VAE [20] to conduct data augmentation. We also propose new losses to regularize them in training. Besides, OnlineAugment integrates both adversarial training and meta-learning in updating the augmentation networks. Adversarial training is to prevent overfitting, whereas meta-learning encourages generalization.

In summary, our key contributions are:

– We propose a new online data augmentation scheme based on meta-learned augmentation networks co-trained with the target task. Our framework is complementary to the state-of-the-art offline methods such as AutoAugment. Experiments on CIFAR, SVHN, and ImageNet show that on its own, OnlineAugment achieves comparable performances to AutoAugment. More excitingly, OnlineAugment can further boost state-of-the-art performances if used jointly with AutoAgument policies.
– We propose three complementary augmentation models responsible for different types of augmentations. They replace the image processing functions commonly used in contemporary approaches and make our method both more adaptive and less dependent on domain knowledge.
– We show that the proposed OnlineAugment can generalize to tasks different from object classification by applying it to a liver&tumor segmentation task, demonstrating improved performance compared with the state-of-the-art RandAugment on this task.

2 Related Work

Data augmentation has been shown to improve the generalization of machine learning models and is especially effective in training deep neural networks. It is essential in the situation where only limited data is available for the target task, but is also crucial for generalization performance in case the data is abundant.

Known class-preserving transformation has been routinely used to expand labeled datasets for training image classifiers. These include operations such as cropping, horizontal and vertical flips, and rotation [5,33,21]. Recently, reinforcement learning has been used to learn the optimal sequence of such transformations for a given dataset that leads to improved generalization [29]. AutoAugment [6] falls under this category of methods and actively explores policies and validates their effectiveness by repeatedly training models from scratch. Due to the large search space, the searching process, based on reinforcement learning, severely suffers from high computational demand. Subsequent works [7,18] in this direction have been aimed at reducing the computational complexity of the search strategies. However, they all follow the formulation of AutoAugment that first searches policies using a sampled small dataset, and then applies them to the final large dataset. Thus the policy learning and usage are isolated. Adversarial AutoAugment [43] jontly optimizes the polices and target learner. However, the learned policies are still based on domain-specific image processing functions.

More general transformations, such as Gaussian noise and dropout, are also effective in expanding the training set [35,36,10]. Spatial Transformer Network (STN) can perform more adaptive transformations than image processing functions such as rotation. However, it was designed for localization, not for data augmentation. In this paper, we craft it to conduct data augmentation. Generative models are also helpful for data augmentation. DAGAN [1] employs a Generative Adversarial Network (GAN) [15] to learn data augmentation for few-shot classification. In this work, we devise two augmentation models based on Variational Auto-encoder (VAE) [20], as another popular generative model.

Adversarial training [16,22,27] can serve as a general data augmentation framework. It aims to generate adversarial perturbations on input data. The adversarial examples are further used in training models to improve their robustness. It has been shown that adversarial training can hurt model generalization although it can boost robustness [27,38]. Concurrent work AdvProp [37] successfully adapts adversarial training to advance model generalization. It uses the common adversarial attack methods, such as PGD and I-FGSM, to generate additive noises. In contrast, our models can learn to generate more diverse augmentations: spatial transformation, deformation, and additive noises. We also use adversarial training together with meta-learning.

Learning data augmentation is to train the target learner better, i.e., learning to learn better. Validation data have been used in meta-learning literatures for few-shot learning [30,31,26], where very limited training data are available. Here we follow the MAML [11] algorithm to set a meta-objective for the augmentation network. That is, augmentations conducted on a training mini-batch is evaluated on another validation one.

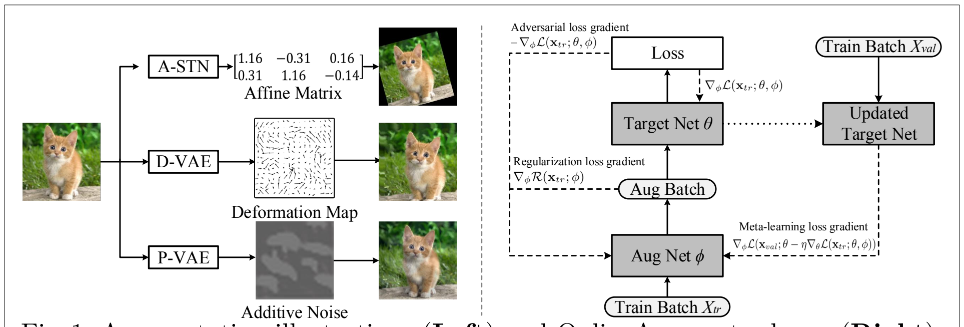
A-STN
1.16 −0.31 0.16
0.31 1.16 −0.14
Affine Matrix
D-VAE
Deformation Map
P-VAE
Additive Noise
Adversarial loss gradient
$-\nabla_\phi\mathcal{L}(\mathbf{x}_{tr};\theta,\phi)$
Loss
$\nabla_\phi\mathcal{L}(\mathbf{x}_{tr};\theta,\phi)$
Train Batch X_{val}
Target Net θ
Updated Target Net
Regularization loss gradient
$\nabla_\phi\mathcal{R}(\mathbf{x}_{tr};\phi)$
Aug Batch
Meta-learning loss gradient
$\nabla_\phi\mathcal{L}(\mathbf{x}_{val};\theta-\eta\nabla_\theta\mathcal{L}(\mathbf{x}_{tr};\theta,\phi))$
Aug Net ϕ
Train Batch X_{tr}

Fig. 1: Augmentation illustrations (**Left**) and OnlineAugment scheme (**Right**). We propose three models to conduct spatial transformation, deformation, and noise perturbation. OnlineAugment can jointly optimize each plug-in augmentation network with the target network. Updating the augmentation network incorporates adversarial training, meta-learning, and some novel regularizations.

3 The Online Data Augmentation Formulation

In this section, we introduce our online data augmentation paradigm: updating target model θ and augmentation model ϕ alternately. In this way, data augmentation and target model are learned jointly. Benefiting from the joint learning, the augmentation model ϕ can adapt to the target model θ in training.

For simplicity, let $\mathbf{x}$ be the annotated data, including both input and target. Note that $\mathbf{x}$ can come from any supervised task such as image classification or semantic segmentation. Let θ and ϕ denote the target and augmentation models. During training, the target model θ learns from the augmented data $\phi(\mathbf{x})$ instead of the original x. Note that ϕ will also transform the ground truth annotation if necessary. For example, in semantic segmentation, ϕ applies the same spatial transformations to both an image and its segmentation mask. Without loss of generality, we assume θ and ϕ are parameterized by deep neural networks, which are mostly optimized by SGD and its variants. Given a training mini-batch $\mathbf{x}_{tr}$ sampled from training set $\mathcal{D}_{tr}$, θ is updated by stochastic gradient:

$$\nabla_\theta\mathcal{L}(\mathbf{x}_{tr};\theta,\phi), \tag{1}$$

where the choice of $\mathcal{L}$ depends on the task. In the case of object classification, $\mathcal{L}$ is a cross entropy function.

The goal of data augmentation is to improve the generalization of the target model. To this end, we draw on inspirations from adversarial training and meta-learning. Adversarial training aims to increase the training loss of the target model by generating hard augmentations. It can effectively address the overfitting issue of the target model. Meta-learning, on the other hand, can measure the impact of augmented data on the performance of validation data. If a validation set $\mathcal{D}_{val}$ is possible, we can sample from it a validation mini-batch $\mathbf{x}_{val}$. Otherwise, we can simulate the meta-tasks by sampling two separate mini-batches from train set $\mathcal{D}_{tr}$ as $\mathbf{x}_{tr}$ and $\mathbf{x}_{val}$. Mathematically, the stochastic gradient of

```
Algorithm 1: OnlineAugment: Online Data Augmentation
   Input: Initial target model $\theta$, initial augmentation model $\phi$, training set $\mathcal{D}_{tr}$,
          and validation set $\mathcal{D}_{val}$
1  while not converged do
2      Sample mini-batches $\mathbf{x}_{tr}$ and $\mathbf{x}_{val}$ from $\mathcal{D}_{tr}$ and $\mathcal{D}_{val}$ respectively
3      Update augmentation model $\phi$ by stochastic gradient:
         $\nabla_\phi \mathcal{L}(\mathbf{x}_{val}; \theta - \eta \nabla_\theta \mathcal{L}(\mathbf{x}_{tr}; \theta, \phi)) + \lambda \nabla_\phi \mathcal{R}(\mathbf{x}_{tr}; \phi) - \beta \nabla_\phi \mathcal{L}(\mathbf{x}_{tr}; \theta, \phi)$
4      Update target model $\theta$ by stochastic gradient: $\nabla_\theta \mathcal{L}(\mathbf{x}_{tr}; \theta, \phi)$
5  end
   Output: Optimized target model $\theta^*$
```

augmentation model ϕ is computed as:

$$\nabla_\phi \mathcal{L}(\mathbf{x}_{val}; \theta - \eta \nabla_\theta \mathcal{L}(\mathbf{x}_{tr}; \theta, \phi)) + \lambda \nabla_\phi \mathcal{R}(\mathbf{x}_{tr}; \phi) - \beta \nabla_\phi \mathcal{L}(\mathbf{x}_{tr}; \theta, \phi), \tag{2}$$

where $\mathcal{L}(\mathbf{x}_{val}; \theta - \eta \nabla_\theta \mathcal{L}(\mathbf{x}_{tr}; \theta, \phi))$, $\mathcal{R}(\mathbf{x}_{tr}; \phi)$, and $-\mathcal{L}(\mathbf{x}_{tr}; \theta, \phi)$ are the generalization, regularization, and adversarial losses. λ and β are the balancing weights. $\theta - \eta \nabla_\theta \mathcal{L}(\mathbf{x}_{tr}; \theta, \phi)$ represents the the updated target network by augmented data $\phi(\mathbf{x}_{tr})$. For simplicity, here we use a vanilla gradient descent with learning rate η. Other more complex optimizers are also applicable. For efficient training, we use the second-order approximation [25] to compute the meta-gradient.

$\mathcal{R}(\mathbf{x}; \phi)$ measures the distance between the original and augmented data. Adding this regularization term is to constrain the augmented data within reasonable distributions. Otherwise, adversarial training may cause meaningless augmentations that hurt training. Theoretically, the generalization term can also help regularize the augmentations implicitly. In practice, we find that the explicit regularization term is critical for practical adversarial training. Besides, adversarial training is performed by minimizing the negative training loss $-\mathcal{L}(\mathbf{x}_{tr}; \theta, \phi)$. In this way, the augmentation model ϕ learn to generate hard augmentations. The training scheme is presented in Algorithm 1 and the right figure in Fig. 1.

Relation to offline augmentation methods. The formulation of our online augmentation differs from the previous offline ones [6,7,18] mainly in three aspects. First, OnlineAugment alternates updating the target model θ and augmentation model ϕ. The offline methods usually perform a full optimization for θ in each step of updating ϕ. Second, to get the optimized target model, the offline methods usually require a two-stage training: learning policies and applying them. However, OnlineAugment can optimize the target model in one training process. Third, we use adversarial training in learning the augmentation model. The offline methods only have one generalization objective, maximizing performance on validation data.

Relation to adversarial training methods. Adversarial training [16,22,27] is mainly used to improve the robustness of the target model. Some works [27,38] have shown that robust models usually come at the cost of degraded generalization to clean data. The goal of OnlineAugment is generalization rather than robustness. To pilot adversarial training for generalization, we design new regularization terms and add meta-learning in OnlineAugment.

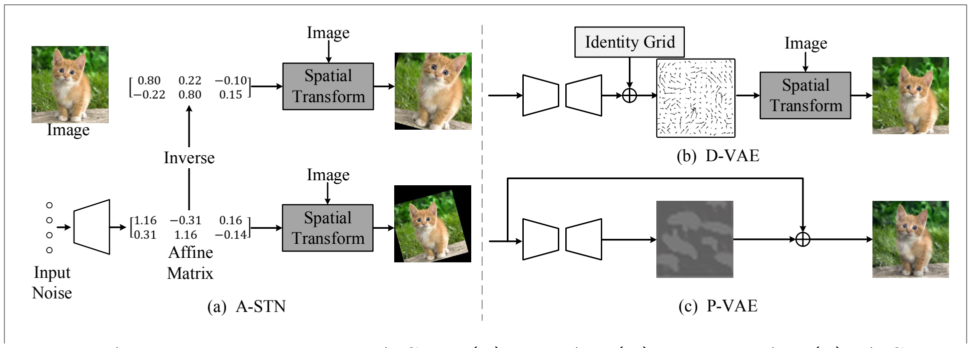

Fig. 2: Augmentation models: A-STN **(a)**, D-VAE **(b)**, and P-VAE **(c)**. A-STN, conditioned on Gaussian noise, generate a transformation matrix. Both the matrix and its inverse are applied to an image for diversity. D-VAE or P-VAE takes an image as input, generating deformation grid maps or additive noise maps. The three models are trainable if plugged in the training scheme in Figure 1.

4 Data Augmentation Models

After introducing the OnlineAugment scheme, we present three different data augmentation models in this section. The three models are motivated by our analysis of possible data transformations. Specifically, we summarize them into three types: global spatial transformation, local deformation, and intensity perturbation. Each transformation corresponds to a model below. They either change pixel locations or values. Note that the pixel in this work refers to an element in a generic feature map, not necessarily an image. Technically, we design the augmentation models based on the spatial transformer network (STN) [19] and variational auto-encoder (VAE) [20].

4.1 Global Spatial Transformation Model

There are several commonly used spatial transformation functions such as rotation, scale, and translation, which can be unified into more general ones, such as affine transformation. It is well-known that STN [19] can perform general spatial transformations. Briefly, STN contains three parts: a localization network, a grid generator, and a sampler. The last two modules are deterministic and differentiable functions, denoted as τ. Therefore, our focus is to design a new localization network ϕ that outputs the transformation matrix.

Augmentation STN (A-STN). Suppose we use affine transformation, the output should be a 6-dimension vector, further re-shaped into a 2×3 transformation matrix. Traditionally, the STN localization network uses images or feature maps as its input. Here we also provide an alternative input of Gaussian noises. Our design is motivated by the observation that the global spatial transformations are transferable in data augmentation. That is, the same spatial transformation is applicable to different images or feature maps in augmenting training data. Therefore, conditioning on the images or feature maps may not be necessary for generating the augmentation transformation.

The architecture of the localization network ϕ depends on the choices of its input. It can be a convolutional neural network (CNN) or a multi-layer perceptron (MLP) if conditioned on the generic feature map or 1-D Gaussian noise. We will give its detailed architectures in the experiment. Moreover, the MLP localization network itself is also transferable as it is unrelated to the target task. However, we may need to craft new CNN localization networks for different tasks. Therefore, it is preferable to implement the localization network ϕ as an MLP in practice.

Double Cycle-consistency Regularization. To apply the spatial transformation model to Algorithm 1, we need to design a proper regularization term $\mathcal{R}$. Empirically, increasing the spatial variance of training data can enhance the generalization power of the model. However, excessive spatial transformations probably bring negative impacts. To constrain the spatial transformations within reasonable scopes, we propose a novel double cycle-consistency regularization, see in Fig. 2 (a). The key idea is to measure the lost information during the spatial transformation process. Mathematically, we compute the double cycle-consistency loss:

$$\mathcal{R}_c^s(\mathbf{x};\phi) = \|\tau_\phi^{-1}(\tau_\phi(\mathbf{x})) - \mathbf{x}\|_2^2 + \|\tau_\phi(\tau_\phi^{-1}(\mathbf{x})) - \mathbf{x}\|_2^2, \tag{3}$$

where $\tau_\phi(\mathbf{x}) = \tau(\mathbf{x}, \phi(\mathbf{z}))$. The deterministic function τ transforms the image or feature map $\mathbf{x}$ using the generated affine matrix $\phi(\mathbf{z})$, where $\mathbf{z}$ is the Gaussian noise. τ_ϕ^{-1} denotes the inverse transformation of τ_ϕ. Ideally, applying a transformation followed by its inverse will recover the original input, and vice versa. In reality, whichever applied first may cause some irreversible information loss. For example, the zoom-in transformation discards the region falling out of scope. Applying the zoom-out transformation afterwards will produce zero or boundary padding, which is different from the original image region. We find a single cycle-consistency loss will lead to biased transformations. The localization network ϕ tends to output zoom-out transformations whose inverse can easily recover the original input. Fortunately, imposing the double cycle-consistency constraint can avoid the biases effectively, thereby producing more diverse transformations.

4.2 Local Deformation Model

Apart from the global spatial transformation model, we propose another complementary deformation model. The global transformation applies the same transformation matrix to all the pixels of an image or feature map. In the local deformation, each pixel, however, has an independent transformation.

Input and Output. It is cumbersome and also unnecessary to produce all the transformation matrices. Recall that STN performs transformations by the grid sampling. A better choice is to predict a grid map directly. For 2D transformations, the grid map has the shape $h \times w \times 2$, where h and w are the height and width of the input feature map. Each location in the grid map indicates the 2D coordinates to sample a pixel. A grid map is personalized to an image or feature map as each pixel has its own transformation. Different from a low-dimension affine matrix, a deformation grid map may be unlikely to transfer. Therefore, our deformation model is conditioned on the image or feature map, generating the grid map.

Deformation VAE (D-VAE). The deformation model ϕ, see in Fig. 2 (b), builds on the Variational Autoencoders (VAE) [20], a popular generative model. A VAE model consists of an encoder and a decoder. Similar to the original VAE, our deformation VAE also uses images or feature maps as the encoder input. However, in our setting, the decoder outputs the deformation grid maps instead of the reconstructed input. We refer to the deformation grid maps as deformation deltas Δ_{ϕ}^{d}. They are added on the grid maps of identity mapping id to perform grid sampling τ. The transformed input $\tau(\mathbf{x}, \Delta_{\phi}^{d} + id)$ serves as the reconstructed input. Following the original VAE, our deformation VAE is also trained to minimize both the reconstruction loss and the KL-divergence between the encoded distribution and the standard normal distribution:

$$\mathcal{R}_{v}^{d}(\mathbf{x};\phi) = \|\mathbf{x} - \tau(\mathbf{x}, \Delta_{\phi}^{d} + id)\|_{2}^{2} + \mathcal{KL}(\mathcal{N}(\mu_{\phi}(\mathbf{x}), \Sigma_{\phi}(\mathbf{x})), \mathcal{N}(0, I)), \tag{4}$$

where $\mu_{\phi}(\mathbf{x})$ and $\Sigma_{\phi}(\mathbf{x})$ are the encoded mean and variance, parameterizing the Gaussian distribution.

Smoothness Regularization. Smooth deformations are essential to preserving the quality of deformed data. Otherwise, the deformed data may become noisy as each pixel is sampled from an independent location in the original image or feature map. This is especially important for location-sensitive tasks such as semantic segmentation. Given an arbitrary pixel i and its neighbours $j \in \mathcal{N}(i)$, we enforce the local smoothness constraint on the deformation deltas:

$$\mathcal{R}_{s}^{d}(\mathbf{x};\phi) = \sum_{i} \sum_{j \in \mathcal{N}(i)} \|\Delta_{\phi}^{d}(i) - \Delta_{\phi}^{d}(j)\|_{2}^{2}. \tag{5}$$

The smoothness regularization can make the deformations consistent for nearby pixels. In the experiment, we use the combination $\lambda_{v}^{d}\mathcal{R}_{v}^{d}(\mathbf{x};\phi) + \lambda_{s}^{d}\mathcal{R}_{s}^{d}(\mathbf{x};\phi)$ to regularize our deformation augmentation model.

4.3 Intensity Perturbation Model

The above two models perform data augmentation by changing the pixel locations. Here we propose another model to manipulate the pixel values instead. As an analogy, the offline data augmentation methods [6,7,18] use some built-in image texture processing functions such as colour and brightness. These functions are designed based on the domain knowledge for natural images. In contrast, our intensity perturbation is more general without domain knowledge.

Perturbation VAE (P-VAE). Specifically, the intensity perturbation model ϕ, see in Fig. 2 (c), conditioned on the image or feature map, generates additive noises Δ_ϕ^p. As the deformation, we use the VAE model to learn the intensity perturbation. The reconstructed input is the sum of the input and generated noises $\mathbf{x} + \Delta_\phi^p$. Therefore, we can compute the VAE loss as:

$$\mathcal{R}_v^p(\mathbf{x};\phi) = \|\Delta_\phi^p\|_2^2 + \mathcal{KL}(\mathcal{N}(\mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x})), \mathcal{N}(0, I)), \tag{6}$$

where $\mu_\phi(\mathbf{x})$ and $\Sigma_\phi(\mathbf{x})$ are the mean and variance of the encoded Gaussian distribution. Note that P-VAE produces deltas Δ_ϕ^p in the image or feature map domain while the deltas Δ_ϕ^d, predicted by D-VAE, lie in the grid map domain. It results in the different reconstruction losses in Equations 6 and 4.

Relation to Adversarial Attacks. Additive noise is a common tool in generating adversarial examples [37,27]. Here we explore its potential in data augmentation. Although the adversarial examples serve as augmented data in adversarial training, they are mainly to improve the model's robustness. Some evidence [28] have shown that adversarial training usually sacrifices the model generalization to clean data. However, our intensity perturbation model can improve the generalization through the OnlineAugment. Recently, concurrent work AdvProp [37] successfully adapts the PGD attack [27] to data augmentation. In contrast, we design the perturbation VAE model to generate additive noises.

5 Experiments

In this section, we empirically evaluate OnlineAugment with the three models.

5.1 Experimental Settings

Applying OnlineAugment is simple in practice. The augmentation models A-STN, D-VAE, and P-VAE requires neither pre-training [23] nor early stopping [29]. Because they can adapt to the target network during online training. Inspired by AdvProp [37], we use multiple batch normalization layers to merge different types of augmentations. A-STN, D-VAE, and P-VAE are trained by Adam optimizer with the same learning rate of $1e-3$, weight decay $1e-4$, and $\beta_1 = 0.5$. Other Adam hyper-parameters are set by default in Pytorch.

A-STN. We design the noise conditioned A-STN as a 6-layer MLP. Specifically, it takes only 1-dimensional Gaussian noises as input and outputs 6-dimensional affine parameters. Each hidden layer generates 8-dimension features. Batch normalization, ReLU, and dropout (0.5) are applied after each linear layer. The loss weights are set as $\lambda_c^s = 0.1$ and $\beta^s = 0.1$.

D-VAE. D-VAE consists of an encoder and a decoder. The encoder maps an image to the 32-dimensional latent space. It includes 5 3×3 convolutional layers and three linear layers. The first convolutional layer increases the channel number from 3 to 32. After that, the feature channels double if the convolution stride is 2. There are two convolutional layers on each resolution. The first linear layer takes the reshaped convolutional features and outputs 512-dimensional latent features. Another two linear layers generate the mean and variance vectors of encoded Gaussian distributions. The decoder is simply the reverse of the encoder with transposed convolutions. The last layer is a 1×1 convolution producing 2-channel grid maps. We use the weights $\lambda_v^d = 1$, $\lambda_s^d = 10$, and $\beta^d = 1e-2$.

P-VAE. It shares almost the same architecture as D-VAE. The only difference is the last layer in the decoder, because P-VAE needs to generate additive noises on the images. The latent space dimension is set to 8 for P-VAE. We also set the hyper-parameters $\lambda_v^p = 1e-3$ and $\beta^p = 10$.

Image Classification. We use datasets CIFAR-10, CIFAR-100, SVHN, and ImageNet with their standard training/test splits. To tune hyper-parameters efficiently, we also sample reduced datasets: R-CIFAR-10, R-CIFAR-100, and R-CIFAR-SVHN. Each of them consists of 4000 examples. The sampling is reproducible using the public sklearn StratifiedShuffleSplit function with random state 0. We report top-1 classification accuracy in most tables except for Table 6 with top-1 errors. The target networks are Wide-ResNet-28-10 [41] (W-ResNet), Shake-Shake network [13], and ResNet-50 [17]. We use Cutout [10] as the baselines in Tables 1, 2, 3, and 4.

Medical Image Segmentation. To test the generalization ability of our approach, we further conduct experiments on the medical image segmentation dataset LiTS [3]. LiTS published 131 liver CT images as a training set with liver and tumor annotations. Since these experiments are to prove the effectiveness of our augmentation algorithm, pursuing the highest performance is not our goal, we use the 2D UNet as the segmentation network to segment CT images on the axial plane slice by slice. We randomly split the 131 CT images into 81 training cases, 25 validation cases, and 25 test cases. We first resample all images to $2 \times 2 \times 2$ mm, then center crop each slice to 256 in size, and finally normalize the image window [-200, 250] to [0, 1] as the input of the network. Only A-STN and D-VAE are presented in this task, since P-VAE has no obvious performance improvement. Compared with classification tasks, segmentation tasks are sensitive to location. Therefore, for A-STN and D-VAE, we not only perform a bilinear grid sample on the images, but also perform a nearest neighbor grid sample for the label masks using the same grid.

We also compared our proposed OnlineAugment with RandAugment [7]. We slightly modify RandAugment to fit our setting. As the number of transformations involved is relatively small, all transformations are used during training. Therefore, for global spatial transformation, the search space is the magnitude of 4 transforms, including rotate, translate-x, translate-y, and scale. For the deformation model, the search space is the magnitude of local deformations. At each iteration, the magnitude of each transformation is uniformly and randomly sampled between 0 and the upper bound for both global spatial transformation and local deformation.

5.2 Experimental Results

The experiments consist of ablation studies for the three models, comparisons with AutoAugment [6], and their orthogonality. The comparisons with state-of-the-art methods are reported on image classification and medical segmentation.

Table 1: Evaluation of the Gaussian noise input and double cycle-consistency regularization in A-STN. We compare them to the image condition and single cycle-consistency. Double cycle-consistency outperforms the single one. With the double one, the noise and image inputs get comparable accuracy.

Dataset	Model	Cutout	Image Input	Image Input	Noise Input	Noise Input
			+1 cycle	+2 cycles	+1 cycle	+2 cycles
R-CIFAR-10	W-ResNet	80.95	83.24	84.76	82.62	**84.94**

Table 2: Evaluation of the smoothness regularization (SR) in D-VAE. We report the results on both image classification and segmentation. The smoothness regularization is more useful for the location-sensitive image segmentation task.

Dataset	Model	Baseline	D-VAE Only	D-VAE + SR
R-CIFAR-10	W-ResNet	80.95 (Cutout)	82.72	**82.86**
Liver Segmentation	U-Net	66.0 (No Aug.)	68.51	**70.49**

A-STN. The A-STN may be conditioned on image or Gaussian noise. We compare these two choices in this ablation study. Besides, we also compare its regularization with single or double cycle-consistency losses. Table 1 gives the comparisons. The double cycle-consistency obtains higher accuracy than the single cycle one. Because it can make A-STN produce more diverse transformations. With the double-cycle consistency regularization, the noise and image conditions achieve comparable accuracy. We use the noise condition A-STN in other experiments since its architecture is transferable between tasks.

D-VAE. The smoothness regularization comes as an additional component in the D-VAE. We evaluate its effectiveness in both CIFAR-10 classification and liver segmentation tasks. Table 2 presents the results. Interestingly, the smoothness regularization has little effect on classification accuracy. However, it makes a difference (%2) for the liver segmentation. Because the liver segmentation is a location-sensitive task. The smoothness regularization can remove the noises along the boundaries of segmentation masks.

P-VAE. The P-VAE generates additive noises to perform data augmentation. AdvProp [28] has a similar goal, but utilizes the iterative gradient methods for the generation. To make adversarial training helpful for the generalization to clean data, AdvProp requires to set proper step length and number in iterative gradient methods. For a fair comparison, we use only adversarial training and the noise regularization in training P-VAE. Table 3 shows the comparisons. P-VAE compares favorably to AdvProp with GD and I-FGSM. On the one hand, P-VAE learns some noise distributions while the iterative methods rely on the deterministic gradient ascent rules. On the other hand, P-VAE generates structured noises, while the iterative approaches produce more complex ones.

Table 3: Comparisons of P-VAE to AdvProp [37] with iterative gradient methods PGD [27], GD, and I-FGSM [4]. Adversarial training plus only the noise regularization can make P-VAE comparable to AdvProp with GD or I-FGSM.

Dataset	Model	Cutout	AP+PGD	AP+GD	AP+I-FGSM	P-VAE
R-CIFAR-10	W-ResNet	80.95	83.00	83.90	83.92	**84.08**

Table 4: Ours *v.s.* AutoAugment (AA). The three models helps separately, and they together may perform on a par with AA. The stochastic shake-shake operations may interfere with the online learning, reducing the improvements.

Dataset	Model	Cutout	AA	A-STN	D-VAE	P-VAE	Comb.
R-CIFAR-10	Wide-ResNet-28-10	80.95	85.43	84.94	82.72	84.18	**85.65**
	Shake-Shake (26 2x32d)	85.42	**87.71**	86.62	86.51	86.34	87.12
R-CIFAR-100	Wide-ResNet-28-10	41.64	47.87	46.55	45.42	47.45	**48.31**
	Shake-Shake (26 2x32d)	44.41	**48.18**	46.81	46.53	46.30	47.27
R-SVHN	Wide-ResNet-28-10	90.16	93.27	92.73	91.32	91.61	**93.29**
	Shake-Shake (26 2x32d)	94.03	**94.63**	94.15	94.06	94.12	94.21

OnlineAugment *v.s.* AutoAugment. AutoAugment is a representative offline augmentation method. Here we evaluate OnlineAugment by comparing with it. Table 4 provides the separate results of three data augmentation models, as well as their combined. Each model, independently, can boost the generalization of two target networks on three datasets. Combining them can bring further improvements, achieving comparable performance as AutoAugment. We can also observe that the improvements for the Shake-Shake network [13] are lower than those of the Wide ResNet [41]. One possible explanation is that the stochastic shake-shake operations may affect the online learning of data augmentation.

OnlineAugment + AutoAugment. Apart from comparing OnlineAugment with AutoAugment, it is more interesting to investigate their orthogonality. Table 5 summarizes the results. We can find that OnlineAugment can bring consistent improvements on top of AutoAugment for different target networks and datasets. Their orthogonality comes from the differences in training schemes and augmentation models. Different from OnlineAugment, AutoAugment learns data augmentation policies in an offline manner. Moreover, the three models (A-STN, D-VAE, and P-VAE) generate different augmentations from the image processing functions in AutoAugment. Note that OnlineAugment is also orthogonal to other offline methods [6,7,18] since they have similar policies as AutoAugment.

Comparisons with State-of-the-art Methods. Besides AutoAugment, we also compare OnlineAugment with other state-of-the-art methods such as Fast AutoAugment (FAA) and Population-based Augmentation (PBA). They all belong to offline data augmentation. OnlineAugment alone gets the lowest test errors on CIFAR-10 and CIFAR-100 and comparable errors on ImageNet. Further, OnlineAugment, together with AutoAugment, achieves new state-of-the-art results on all the three image classification datasets.

Table 5: Ours+AutoAugment (AA). We use A-STN, D-VAE, and P-VAE on top of the AutoAugment polices. Surprisingly, each model can further improve AutoAugment performance. It demonstrates that OnlineAugment is orthogonal to AutoAugment. The three models use more general augmentation operations.

Dataset	Model	AA	+A-STN	+D-VAE	+P-VAE	+Comb.
R-CIFAR-10	Wide-ResNet-28-10	85.43	89.39	87.40	87.63	**89.40**
	Shake-Shake (26 2x32d)	87.71	89.25	88.43	88.52	**89.50**
R-CIFAR-100	Wide-ResNet-28-10	47.87	52.94	50.01	51.02	**53.72**
	Shake-Shake (26 2x32d)	48.18	**50.58**	50.42	50.87	50.11
R-SVHN	Wide-ResNet-28-10	93.27	94.32	93.72	94.17	**94.69**
	Shake-Shake (26 2x32d)	94.63	95.21	94.87	**95.28**	95.06

Table 6: Comparisons with state-of-the-art methods. We compare our OnlineAugment with AutoAugment (AA), PBA [18], and Fast AutoAugment (FAA) [24] on three datasets. OnlineAugment alone obtains comparable test errors. Combining it with AutoAugment produces the lowest errors on three datasets.

Dataset	Model	Baseline	Cutout	AA	PBA	FAA	Ours	Ours+AA
CIFAR-10	Wide-ResNet-28-10	3.9	3.1	2.6	2.6	2.7	2.4	**2.0**
CIFAR-100	Wide-ResNet-28-10	18.8	18.4	17.1	16.7	17.3	16.6	**16.3**
ImageNet	ResNet-50	23.7	-	22.4	-	22.4	22.5	**22.0**

Comparisons with RandAugment on LiTS dataset. Our method has also achieved promising results on the semantic segmentation task, see in Table 7. The proposed method has comparable performance with RandAugment using global spatial transformation and has better results using local deformation. Moreover, OnlineAugment has a further performance boost by combining these two modules, while RandAugment does not benefit from the combination. Specifically, our onlineAugment achieves better results on the tumor images. It is because that OnlineAugment can adaptively generate harder samples for the target network, not just random ones. In Fig. 3, augmented images of A-STN (top), and D-VAE (bottom) along with the training process are shown. One observation is that the predicted augmentation magnitude is relatively large at the beginning of training. As the training progress goes on, the network converges, and the predicted magnitude of augmentation gradually decreases.

6 Conclusion

In this paper, we have presented OnlineAugment - a new and powerful data augmentation technique. Our method adapts online to the learner state throughout the entire training. We have designed three new augmentation networks that are capable of learning a wide variety of local and global geometric and photometric transformations, requiring less domain knowledge of the target task. Our On-

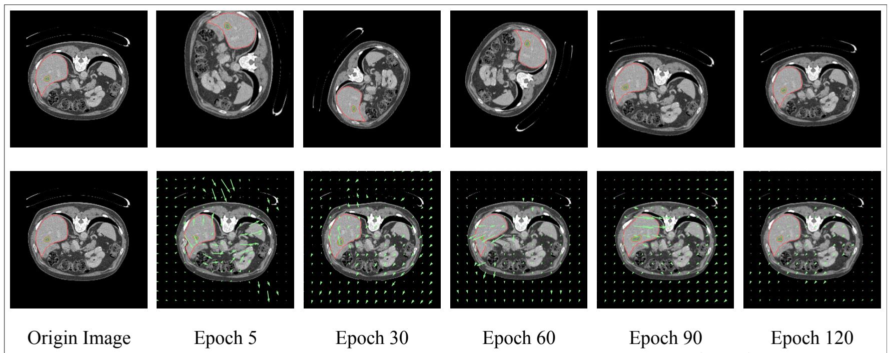

Fig. 3: Visualization of two augmentation modules: A-STN (**top**) and D-VAE (**bottom**). Red line is the contour of liver while green line is the contour of tumor. Our OnlineAugment can generate diverse augmented images. Moreover, it can also adapt to the target network. As the target network converges during training, the magnitude of the augmentation will also decrease.

Table 7: Dice score coefficient comparison (%) of our OnlineAugment with RandAugment on LiTS dataset. Our OnlineAugment has comparable performance with RandAugment in STN, and has much better results in local deformation and the combination of these two.

Method	Liver	Tumor	Average
no Augmentaion	89.04	44.73	66.88
RandAug STN	**93.86**	50.54	72.20
RandAug Deformation	90.49	46.93	68.71
RandAug Combine	91.91	51.11	71.51
Ours A-STN	92.01	52.26	72.13
Ours D-VAE	90.18	50.81	70.49
Ours Combine	93.12	**53.58**	**73.35**

lineAugment integrates both adversarial training and meta-learning for efficient training. We also design essential regularization techniques to guide adaptive online augmentation learning. Extensive experiments demonstrate the utility of the approach to a wide variety of tasks, matching (without requiring expensive offline augmentation policy search) the performance of the powerful AutoAugment policy, as well as improving upon the state-of-the-art in augmentation techniques when used jointly with AutoAugment.

7 Acknowledgment

This work has been partially supported by NSF 1763523, 1747778, 1733843 and 1703883 Awards to Dimitris Metaxas and the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-19-C-1001 to Leonid Karlinsky and Rogerio Feris.

References

1. Antoniou, A., Storkey, A., Edwards, H.: Data augmentation generative adversarial networks. arXiv preprint arXiv:1711.04340 (2017)
2. Baker, B., Gupta, O., Naik, N., Raskar, R.: Designing neural network architectures using reinforcement learning. arXiv preprint arXiv:1611.02167 (2016)
3. Bilic, P., Christ, P.F., Vorontsov, E., Chlebus, G., Chen, H., Dou, Q., Fu, C.W., Han, X., Heng, P.A., Hesser, J., et al.: The liver tumor segmentation benchmark (lits). arXiv preprint arXiv:1901.04056 (2019)
4. Chang, T.J., He, Y., Li, P.: Efficient two-step adversarial defense for deep neural networks. arXiv preprint arXiv:1810.03739 (2018)
5. Ciregan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: CVPR (2012)
6. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation strategies from data. In: CVPR. pp. 113–123 (2019)
7. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. arXiv preprint arXiv:1909.13719 (2019)
8. Dai, J., Li, Y., He, K., Sun, J.: R-fcn: Object detection via region-based fully convolutional networks. In: NeurIPS (2016)
9. DeVries, T., Taylor, G.W.: Dataset augmentation in feature space. arXiv preprint arXiv:1702.05538 (2017)
10. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 (2017)
11. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: ICML. pp. 1126–1135. JMLR. org (2017)
12. Fu, J., Liu, J., Tian, H., Li, Y., Bao, Y., Fang, Z., Lu, H.: Dual attention network for scene segmentation. In: CVPR. pp. 3146–3154 (2019)
13. Gastaldi, X.: Shake-shake regularization. arXiv preprint arXiv:1705.07485 (2017)
14. Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P., He, K.: Detectron (2018)
15. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NeurIPS. pp. 2672–2680 (2014)
16. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
18. Ho, D., Liang, E., Stoica, I., Abbeel, P., Chen, X.: Population based augmentation: Efficient learning of augmentation policy schedules. arXiv preprint arXiv:1905.05393 (2019)
19. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: NeurIPS. pp. 2017–2025 (2015)
20. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NeurIPS. pp. 1097–1105 (2012)
22. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial machine learning at scale. arXiv preprint arXiv:1611.01236 (2016)
23. Lee, D., Park, H., Pham, T., Yoo, C.D.: Learning augmentation network via influence functions. In: CVPR (2020)

24. Lim, S., Kim, I., Kim, T., Kim, C., Kim, S.: Fast autoaugment. In: NeurIPS. pp. 6662–6672 (2019)
25. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)
26. Lorraine, J., Duvenaud, D.: Stochastic hyperparameter optimization through hypernetworks. arXiv preprint arXiv:1802.09419 (2018)
27. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
28. Raghunathan, A., Xie, S.M., Yang, F., Duchi, J.C., Liang, P.: Adversarial training can hurt generalization. arXiv preprint arXiv:1906.06032 (2019)
29. Ratner, A.J., Ehrenberg, H., Hussain, Z., Dunnmon, J., Ré, C.: Learning to compose domain-specific transformations for data augmentation. In: NeurIPS. pp. 3236–3246 (2017)
30. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning (2016)
31. Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J.B., Larochelle, H., Zemel, R.S.: Meta-learning for semi-supervised few-shot classification. arXiv preprint arXiv:1803.00676 (2018)
32. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI. pp. 234–241. Springer (2015)
33. Simard, P.Y., Steinkraus, D., Platt, J.C., et al.: Best practices for convolutional neural networks applied to visual document analysis. In: Icdar. vol. 3 (2003)
34. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
35. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. JMLR **15**(1), 1929–1958 (2014)
36. Taylor, L., Nitschke, G.: Improving deep learning using generic data augmentation. arXiv preprint arXiv:1708.06020 (2017)
37. Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A., Le, Q.V.: Adversarial examples improve image recognition. arXiv preprint arXiv:1911.09665 (2019)
38. Xie, C., Wu, Y., Maaten, L.v.d., Yuille, A.L., He, K.: Feature denoising for improving adversarial robustness. In: CVPR. pp. 501–509 (2019)
39. Yang, M., Yu, K., Zhang, C., Li, Z., Yang, K.: Denseaspp for semantic segmentation in street scenes. In: CVPR (2018)
40. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V.: Xlnet: Generalized autoregressive pretraining for language understanding. In: NeurIPS. pp. 5754–5764 (2019)
41. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
42. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)
43. Zhang, X., Wang, Q., Zhang, J., Zhong, Z.: Adversarial autoaugment. In: ICLR (2019)
44. Zoph, B., Cubuk, E.D., Ghiasi, G., Lin, T.Y., Shlens, J., Le, Q.V.: Learning data augmentation strategies for object detection. arXiv preprint arXiv:1906.11172 (2019)
45. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)