LEARNING TO AUGMENT INFLUENTIAL DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Data augmentation is a technique to reduce overfitting and to improve generalization by increasing the number of labeled data by performing label preserving transformations; however, it is currently conducted in a trial and error manner. A composition of predefined transformations such as rotation, scaling, and cropping is performed on training samples, and its effect on performance over test samples can only be empirically evaluated and cannot be predicted. This paper considers an influence function that predicts how generalization is affected by a particular augmented training sample in terms of validation loss, without comparing the performance that includes and excludes the sample in the training process. A differentiable augmentation model that generalizes the conventional composition of predefined transformations is proposed. The differentiable augmentation model and reformulation of the influence function allow the augmented model parameters to be updated by backpropagation to minimize the validation loss. The experimental results show that the proposed method provides better generalization over conventional data augmentation methods.

1 INTRODUCTION

In supervised learning, training deep neural networks generally requires large size labeled data. Insufficient labeled data will lead to poor generalization due to overfitting. One simple method to reduce overfitting and to improve generalization is to perform data augmentation which involves creating additional labeled data by modifying each training samples with label preserving transformations. Different augmentation can bring about huge differences in test performance depending on the task and training dataset (Dosovitskiy et al. (2016); Graham (2014)). Unfortunately, it is an area which has not be extensively explored (Simonyan & Zisserman (2014); He et al. (2016); Xie et al. (2016))—training data is often augmented in a manner similar to that performed in training the AlexNet (Krizhevsky et al. (2012)). A composition of predefined transformations such as rotation, translation, cropping, scaling and color perturbation are popular transformation choices; however, determining the strength of each transformation that will lead to best performance is an art and is often set in an empirical manner with trial and error by observing validation loss (Simard et al. (2003); Ciregan et al. (2012)).

An alternative to tuning a composition of predefined transformations is to learn a strategy for composing the transformations (Sixt et al. (2016); Ratner et al. (2017); Lemley et al. (2017); Peng et al. (2018); Cubuk et al. (2018)). In learning of strategy, various criteria have been considered. Sixt et al. (2016) and Ratner et al. (2017) considers a strategy to augment realistic samples without considering the classification model. Given the classification model, Lemley et al. (2017) and Peng et al. (2018) consider antithetical strategies which minimize (Lemley et al. (2017)) or maximize Peng et al. (2018) training loss, respectively. These antithetical studies imply that relating data augmentation with the training loss has not been fully understood concerning on their effectiveness over the performance on test samples. Cubuk et al. (2018) considers small child models to compute validation loss to augment samples which improve generalization; however, learning requires a reinforcement learning framework with the validation loss as a reward signal due to a gradient from the validation loss cannot flow to training samples. Also, the validation loss of the small child model is not examined to be similar with the validation loss of the end-classification model which may have different architecture.

This paper proposes a data augmentation method which directly traces an impact on generalization through an augmentation model. For this, an influence function Cook & Weisberg (1980); Koh &

Liang (2017) is considered to compute how the validation loss is affected by a particular augmented training sample. The influence function predicts validation loss change due to the augmented training sample without comparing the performance that includes and excludes it in the training process. We also propose a differentiable augmentation model that generalizes the conventional augmentation method. Reformulation of the influence function and the differentiable augmentation model enable a gradient of the validation loss to flow through augmentation model and training samples, thus the augmentation model can be learned to directly improve generalization by backpropagation.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related studies, and Section 4 and 3 describe the details of the proposed method. The experimental and comparative results are reported in Section 5. The conclusions are presented in Section 6.

2 Related works

2.1 Data augmentation methods

In this subsection, a brief overview of data augmentation methods is provided in three categories depends on the use of classification model: (i) unsupervised methods that do not consider the classification model during learning[1], (ii) adversarial methods that maximize classification loss and (iii) supervised methods that minimize classification loss.

Unsupervised methods Unsupervised methods include conventional data augmentation method that is a composition of predefined transformations such as rotating, translating, cropping, scaling and color perturbation (Simonyan & Zisserman (2014); He et al. (2016); Xie et al. (2016); Krizhevsky et al. (2012)). The transformations are manually chosen in an empirical way with trial and error by observing validation loss (Simard et al. (2003); Ciregan et al. (2012)). Ratner et al. (2017) considered a generator which generates a sequence of predefined transformations. Given the sequence, training sample is augmented by applying predefined transformations in a consecutive way. The generator is learned in GAN framework (Goodfellow et al. (2014)) that the generated sequence produces realistic augmented sample; however, the classifier is not considered during learning process and an effect of data augmentation is only can be observed in a trial and error.

Adversarial methods Adversarial methods include hard sample mining that is to collect or augment samples which are misclassified by the current classification model. It has been used in training SVM models (Dalal & Triggs (2005)), boosted decision trees (Dosovitskiy et al. (2016)), shallow neural network (Rowley et al. (1998)) and deep neural network (Shrivastava et al. (2016a)). Peng et al. (2018) and Wang et al. (2017a) considered to generate hard samples by adversarially updating ranges of predefined transformations such as occluding (Peng et al. (2018); Wang et al. (2017a), scaling and rotating Peng et al. (2018). In these methods, a simple composition of predefined transformations can only be used due to that more flexible transformations may generate adversarial examples (Peng et al. (2018)). Adversarial examples are obtained by applying small perturbations or transformations to samples that result in the classification model predicts an incorrect answer with high confidence (Szegedy et al. (2013)). Recent studies have shown that adversarial update in convolution based transformation (Baluja & Fischer (2018)) and spatial transformation (Xiao et al. (2018)) can generates adversarial examples.

Supervised methods Lemley et al. (2017) designed an augmentation model that learns how to augment samples during learning of a classification model in a way that reduces the training classification loss, but its effect on performance over test samples can only be empirically evaluated and cannot be predicted. Cubuk et al. (2018) considers small child models to compute validation loss to evaluate several augmentation policies over predefined transformations; however, learning requires a reinforcement learning framework due to a gradient cannot flow from the validation loss to training samples as well as non-differentiable predefined transformations stop the gradient. Also, the validation loss of the small child model may be different from the validation loss of the end-classification model.

[1]Unsupervised methods may observe validation loss; however, they are referred to as unsupervised due to learning is not involved.

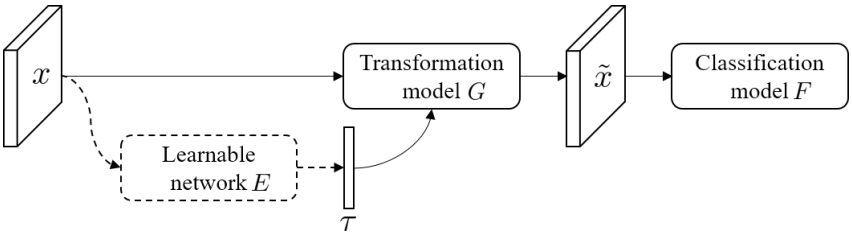


Figure 1: A generic data augmentation framework for the classification task. An input training sample x is transformed to $\tilde{x}$ by a transformation model which is parameterized by τ. Conventional method usually composed by a sequence of predefined transformations with randomly sampled corresponding ranges τ (solid line path). A learnable network for estimating τ from x is considered to obtain the transformed sample $\tilde{x}$ that maximize generalization of the classification model (dashed line path).

2.2 Generative adversarial networks

Goodfellow et al. (2014) proposed generative adversarial network (GAN) which is a framework for training a deep generative network in an adversarial process. The framework considers the simultaneous training of two networks: a generator which generates a sample from random noise and a discriminator which estimates the probability that a sample came from the training data rather than generator. The adversarial process is formed by a minimax two-player game between discriminator learned to distinguish a source of the sample and generator trained to generate an indistinguishable sample by discriminator.

Based on the GAN framework, several studies have shown their potential to use on data augmentation by generating class-conditional images (Odena et al. (2016); Nguyen et al. (2016); Mirza & Osindero (2014)) or improving the realism of synthetic samples (Shrivastava et al. (2016b); Sixt et al. (2016); Wang et al. (2017b)); however, difficulties to learn a generative models (relatively to a classification model) require additional unlabeled (Odena et al. (2016); Nguyen et al. (2016); Mirza & Osindero (2014)) or synthetic (Shrivastava et al. (2016b); Sixt et al. (2016); Wang et al. (2017b)) samples to improve performance.

2.3 Influence Functions

The influence function is a classic technique from robust statistics (Cook & Weisberg (1980)) that estimates how the model parameters change due to up-weighting a particular training sample by a small amount. Cook & Weisberg (1980) focused on removing training points from linear models and Cook (1986); Thomas & Cook (1990); Wei et al. (1998) have been studied with a wider variety of perturbations. Koh & Liang (2017) scaled up the influence function to non-convex and highly-complex models including deep neural networks by using efficient approximations based on Hessian-vector products (Pearlmutter (1994)), conjugate gradients (Martens (2010)) and stochastic estimation (Agarwal et al. (2016)). They also considered the influence on the validation loss as up-weighting a particular training sample that can be computed without retraining process and used for multiple purposes: debugging models, detecting dataset errors and even creating training-set attacks.

3 Augmented Data Evaluation

Tuning or learning an augmentation model involves evaluating an augmented sample (Figure 1). Given the classification model, this is often performed by computing training loss of the sample; however, an effect on test samples cannot be predicted and only be computed in a trial and error manner. Evaluating augmented sample using validation loss requires learning two classifiers (including and excluding the sample during the learning process) which is computationally expensive due to that the models should be fully trained as well as the cost to compute the loss over validation samples. Cubuk et al. (2018) only consider a small child classification model which has a different architecture with the end classification model, and evaluation is not performed on the augmented sample, but an augmentation policy. Rather than repeat this prohibitive process, we proposed the

method to approximate validation loss change due to a particular augmented sample without the retraining process.

3.1 PROBLEM SET UP

In a classification task, given an input space $\mathcal{X}$, an output space $\mathcal{Y}$ and parameter space Θ, a learner aims to a classification model F that maps $\mathcal{X} \mapsto \mathcal{Y}$ and be parameterized by θ. Given a sample $z = (x, y) \in \mathcal{X} \times \mathcal{Y}$ and parameters $\theta \in \Theta$, let $l(z, \theta)$ be the loss evaluated on the sample z at parameters θ. With training data $\mathbf{z}^{\text{tr}} = \{z_i\}_{i=1}^N$, an empirical risk and an empirical risk minimizer are given as follows:

$$\mathcal{L}(\mathbf{z}^{\text{tr}}, \hat{\theta}(\mathbf{z}^{\text{tr}})) \;=\; \frac{1}{N} \sum_{i: z_i \in \mathbf{z}^{\text{tr}}} l(z_i, \hat{\theta}(\mathbf{z}^{\text{tr}})), \tag{1}$$

$$\text{where} \quad \hat{\theta}(\mathbf{z}^{\text{tr}}) \;=\; \arg\min_{\theta \in \Theta} \mathcal{L}(\mathbf{z}^{\text{tr}}, \theta). \tag{2}$$

To measure generalization of the classification model F, validation data $\mathbf{z}^{\text{val}} = \{z_j\}_{j=1}^M$ is often considered, and generalization is approximated by an average of loss values over validation data $\mathbf{z}^{\text{val}}$ at parameter $\hat{\theta}(\mathbf{z}^{\text{tr}})$:

$$\mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\mathbf{z}^{\text{tr}})) \;=\; \frac{1}{M} \sum_{j: z_j \in \mathbf{z}^{\text{val}}} l(z_j, \hat{\theta}(\mathbf{z}^{\text{tr}})). \tag{3}$$

As shown in Figure 1, consider a label preserving transformation G that maps $\mathcal{X} \mapsto \mathcal{X}$, and let τ be control parameters, $\tilde{x} = G(x, \tau)$ be an augmented input, $\tilde{z} = (\tilde{x}, y)$ be an augmented sample, and $\tilde{\mathbf{z}}^{\text{tr}} = \{\tilde{z}_i\}_{i=1}^N$ be an augmented training dataset. Also, consider a learnable network E parameterized by ϕ. E estimates control parameters τ given the input x. Thus, $\tilde{x} = G(x, E(x, \phi))$.

Given the transformation model G, the goal is then to find optimal τ for each input x—otherwise, to find optimal parameters ϕ of E—that minimizes a classification loss over the validation data $\mathbf{z}^{\text{val}}$ when the classification model is learned using $\tilde{\mathbf{z}}^{\text{tr}}$:

$$\phi \;=\; \arg\min_{\phi \in \Phi} \mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\tilde{\mathbf{z}}^{\text{tr}})) \tag{4}$$

$$\hat{\theta}(\tilde{\mathbf{z}}^{\text{tr}}) \;=\; \arg\min_{\theta \in \Theta} \mathcal{L}(\tilde{\mathbf{z}}^{\text{tr}}, \theta). \tag{5}$$

Here, $\tilde{\mathbf{z}}^{\text{tr}} = \{\tilde{x}, y\}_{i=1}^N = \{G(x, E(x, \phi)), y\}_{i=1}^N$. Solving Equations 4–5 requires a bi-level optimization that the inner problem 5 is the optimization problem itself.

3.2 INFLUENCE BY UPWEIGHTING A TRAINING SAMPLE

Consider a change in model parameters θ due to removing a particular training sample z_i. Formally, this change is $\hat{\theta}(\mathbf{z}^{\text{tr}} \backslash z_i) - \hat{\theta}(\mathbf{z}^{\text{tr}})$. Influence functions (Cook & Weisberg (1980); Koh & Liang (2017)) provide an efficient approximation without retraining process to obtain $\hat{\theta}(\mathbf{z}^{\text{tr}} \backslash z_i)$. Let consider the change in model parameters due to upweighting z_i by ϵ amount in the loss function:

$$\hat{\theta}(\mathbf{z}^{\text{tr}} \cup \epsilon z_i) = \arg\min_{\theta \in \Theta} \mathcal{L}(\mathbf{z}^{\text{tr}}, \theta) + \epsilon l(z_i, \theta) \tag{6}$$

Then, from Cook & Weisberg (1980), the following approximation can be derived:

$$-\frac{1}{N} \mathcal{I}_{\text{up, params}}(z_i) \simeq \hat{\theta}(\mathbf{z}^{\text{tr}} \backslash z_i) - \hat{\theta}(\mathbf{z}^{\text{tr}}), \tag{7}$$

where

$$\mathcal{I}_{\text{up, params}}(z_i) \;\triangleq\; \left. \frac{d\hat{\theta}(\mathbf{z}^{\text{tr}} \cup \epsilon z_i)}{d\epsilon} \right|_{\epsilon=0} \tag{8}$$

$$=\; -H(\hat{\theta}(\mathbf{z}^{\text{tr}}))^{-1} \nabla_\theta l(z_i, \hat{\theta}(\mathbf{z}^{\text{tr}})). \tag{9}$$

Here, $H(\theta) \triangleq \frac{1}{N} \sum_{i=1}^N \nabla_\theta^2 l(z_i, \theta)$ is the Hessian evaluated at θ.

By using Equation 9 and applying chain rule, influence of upweighting $z_i \in \mathbf{z}^{\text{tr}}$ on the validation loss at $z_j \in \mathbf{z}^{\text{val}}$ can be approximated (Koh & Liang (2017)):

$$-\frac{1}{N}\mathcal{I}_{\text{up, loss}}(z_i, z_j) \simeq l(z_j, \hat{\theta}(\mathbf{z}^{\text{tr}} \backslash z_i)) - l(z_j, \hat{\theta}(\mathbf{z}^{\text{tr}})), \tag{10}$$

where

$$\begin{aligned}
\mathcal{I}_{\text{up, loss}}(z_i, z_j) &\triangleq \frac{dl(z_j, \hat{\theta}(\mathbf{z}^{\text{tr}} \cup \epsilon z_i))}{d\epsilon}\Big|_{\epsilon=0} && (11)\\
&= \nabla_\theta l(z_j, \hat{\theta}(\mathbf{z}^{\text{tr}}))^\top \frac{d\hat{\theta}(\mathbf{z}^n \cup \epsilon z_i)}{d\epsilon}\Big|_{\epsilon=0} && (12)\\
&= -\nabla_\theta l(z_j, \hat{\theta}(\mathbf{z}^{\text{tr}}))^\top H(\hat{\theta}(\mathbf{z}^{\text{tr}}))^{-1} \nabla_\theta l(z_i, \hat{\theta}(\mathbf{z}^{\text{tr}})). && (13)
\end{aligned}$$

For the validation dataset $\mathbf{z}^{\text{val}}$, Equation 12 is simply expanded by:

$$\mathcal{I}_{\text{up, loss}}(z_i, \mathbf{z}^{\text{val}}) = -\nabla_\theta \mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\mathbf{z}^{\text{tr}}))^\top H(\hat{\theta}(\mathbf{z}^{\text{tr}}))^{-1} \nabla_\theta l(z_i, \hat{\theta}(\mathbf{z}^{\text{tr}})). \tag{14}$$

Equation 11 describes a gradient of $l(z_j, \hat{\theta}(\mathbf{z}^{\text{tr}}))$ w. r. t. ϵ at nearby $\epsilon = 0$, and influence of removing z_i can be approximate by Equation 10.

3.3 Influence by augmentation

With a training sample z_i and the corresponding augmented training sample $\tilde{z}$, let $\hat{\theta}(\mathbf{z}^{\text{tr}} \cup \epsilon \tilde{z}_i \backslash \epsilon z_i)$ be the estimate of θ with downweighting z_i and with upweighting $\tilde{z}_i$ by ϵ amount. And let $\hat{\theta}(\mathbf{z}^{\text{tr}} \cup \tilde{z}_i \backslash z_i)$ be the estimate of θ by replacing z_i with $\tilde{z}_i$. An analogous approximation of Equation 10–13 yields:

$$-\frac{1}{n}\mathcal{I}_{\text{aug, loss}}(z_i, \mathbf{z}^{\text{val}}) \simeq \mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\mathbf{z}^{\text{tr}})) - \mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\mathbf{z}^{\text{tr}} \cup \tilde{z}_i \backslash z_i)) \tag{15}$$

where the influence function $\mathcal{I}_{\text{aug, loss}}(z_i, \tilde{z}_i, \mathbf{z}^{\text{val}})$ is:

$$\begin{aligned}
\mathcal{I}_{\text{aug, loss}}(z_i, \tilde{z}_i, \mathbf{z}^{\text{val}}) &\triangleq \frac{d\mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\mathbf{z}^{\text{tr}} \cup \epsilon \tilde{z}_i \backslash \epsilon z_i))}{d\epsilon}\Big|_{\epsilon=0} && (16)\\
&= \nabla_\theta \mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\mathbf{z}^{\text{tr}}))^\top \frac{d\hat{\theta}(\mathbf{z}^{\text{tr}} \cup \epsilon \tilde{z}_i \backslash \epsilon z_i)}{d\epsilon}\Big|_{\epsilon=0} && (17)\\
&= -\nabla_\theta \mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\mathbf{z}^{\text{tr}}))^\top H(\hat{\theta}(\mathbf{z}^{\text{tr}}))^{-1} \big(\nabla_\theta l(\tilde{z}_i, \hat{\theta}(\mathbf{z}^{\text{tr}})) - \nabla_\theta l(z_i, \hat{\theta}(\mathbf{z}^{\text{tr}}))\big) && (18)\\
&= \mathcal{I}_{\text{up, loss}}(\tilde{z}_i, \mathbf{z}^{\text{val}}) - \mathcal{I}_{\text{up, loss}}(z_i, \mathbf{z}^{\text{val}}). && (19)
\end{aligned}$$

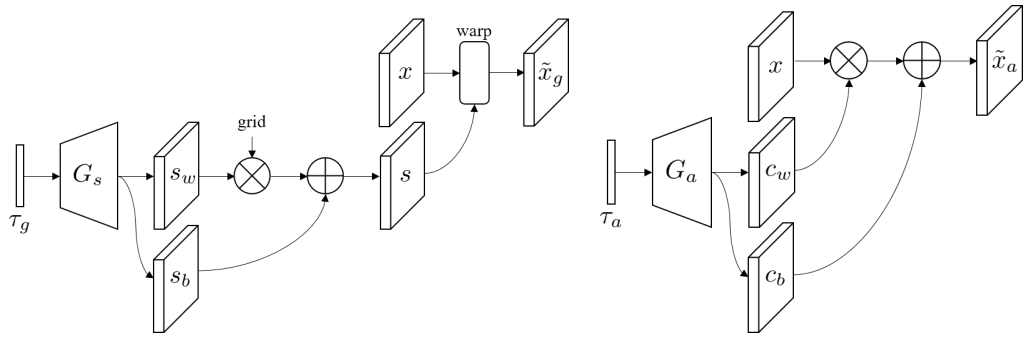
The influence function $\mathcal{I}_{\text{aug, loss}}(z_i, \tilde{z}_i, \mathbf{z}^{\text{val}})$ predicts the influence of the augmented sample compared with the training sample and is the difference of each upweighting influence function values. Also, the influence function can be used to evaluate the augmented data $\tilde{z}_i$.

3.4 Reformulation of influence functions

To compute influence functions efficiently, we adopts techniques such as conjugate gradients and stochastic estimation used in Koh & Liang (2017). Given G and F, Hessian-vector products $s_{\text{val}} = H(\hat{\theta}(\mathbf{z}^{\text{tr}}))^{-1} \nabla_\theta \mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\mathbf{z}^{\text{tr}}))$ is precomputed and cumulated for validation data using conjugate gradients and stochastic estimation techniques. The HVPs are fixed during learning E and are used to compute influence functions of augmented samples. Also, we further approximate the influence function by only considering the top fully connected layer of F, thus the remaining $\nabla_\theta l(z_i, \hat{\theta}(\mathbf{z}^{\text{tr}}))$ can be represented by a simple closed form. This allows to represent $s_{\text{val}} \nabla_\theta l(z_i, \hat{\theta}(\mathbf{z}^{\text{tr}}))$ as a closed form by regarding s_{val} as a fixed vector. Gradient from this flows through fixed F and G, then used to update E by chain rule.

4 Transformation Models

Transformations for augmenting images can be categorized by spatial transformations and appearance transformations. In this section, we describe the proposed transformation models that generalize both transformations. The models are differentiable, thus a gradient from the influence function can be propagate to E during learning.



(a) Spatial transformation model (b) Appearance transformation model

Figure 2: The proposed transformation models for spatial and appearance transformation. (a) Spatial transformation model G_s takes an input τ_g then generates s_w and s_b. Flow field s denotes the coordinates to be transformed and is obtained by operations like as point-wise affine transform by s_w and s_b. Warp operation transforms x to $\tilde{x}$ by interpolating x based on coordinates in s. (b) Appearance transformation model G_a takes an input τ_a then generates c_w and c_b. $\tilde{x}$ is obtained by filtering x by a filter with weights c_w and bias c_b.

4.1 Spatial transformation model

Spatial transformations includes random flip, crop, scaling, rotation, shearing, translation, and affine transformation. These transformations can be defined by the changes in coordinate over image pixel locations. The proposed transformation model for spatial transformation is illustrated in Figure 2-(a). Firstly, we define spatial transformation—in a similar way with Jaderberg et al. (2015); Xiao et al. (2018)—by

$$\begin{bmatrix} s(i,j,1) \\ s(i,j,2) \end{bmatrix} = \begin{bmatrix} s_w(i,j,1) & s_w(i,j,2) \\ s_w(i,j,3) & s_w(i,j,4) \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} s_b(i,j,1) \\ s_b(i,j,2) \end{bmatrix}. \tag{20}$$

Here, i, j denotes a source coordinate, $s_w(i,j)$ and $s_b(i,j)$ denotes multiplication factor and bias, respectively. Note that when we perform global average pooling on s_w and s_b, the transformation is reduced to affine transform. In this formulation, the spatial transformation is fully defined by s_w and s_b and these parameters are what we interest to generate from τ_g.

Spatial transformation model G_g takes an input τ_g then generates s_w and s_b. Flow field s denotes the coordinates to be transformed and is obtained by operations like as point-wise affine transform by s_w and s_b. Warp operation transforms x to $\tilde{x}$ based on bilinear interpolation indexed by s. The model is designed by a stacked transposed convolutional network that its final layer having $4+2$ channels. After we obtain s, average pooling is applied to smoothing s, then image is warped by bilinear interpolation which is differentiable Jaderberg et al. (2015). All computations in this formulation can be implemented by feed forward neural network.

4.2 Appearance Transformation Model

Transformations in appearance include enhance contrast, brightness, color and hue shift. These transformations can be formed by 1×1 spatial dimension filters. Thus, to formulate appearance transformation model, we focused on to generate 1×1 spatial filters. The proposed transformation model for appearance transformation is illustrated in Figure 2-(b). Appearance transformation model G_a takes an input τ_a then generates c_w and c_b. c_w and c_b are global average pooled to form a filter with 1×1 spatial resolution. Transformed image $\tilde{x}$ is obtained by filtering x with filter weights c_w and bias c_b. The model is designed by a transposed convolutional network that its final layer having $1 \times 1 + 1$ channels for grey images and $3 \times 3 + 3$ channels for RGB images.

4.3 Learning transformation models

During learning, G is firstly trained based on the GAN framework and is fixed for further learning process. Then, E is trained to predict τ that maximize the influence function for each x and

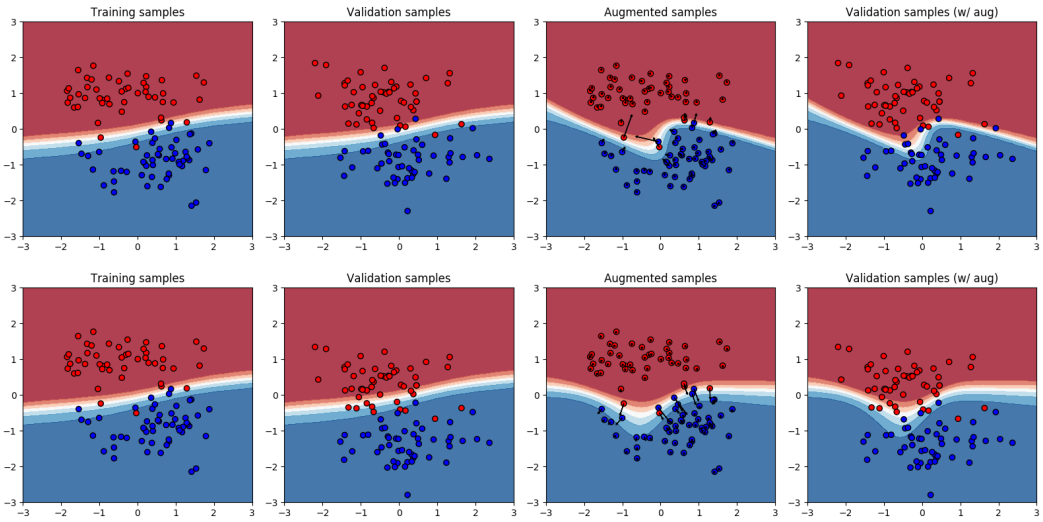


Figure 3: Toy example on a synthetic binary classification dataset. Red and blue dots represent data point of class red and blue, respectively. Contour plot of predictive probability is overlaid. Each column represents: (i) training samples and predictive probability contour plot from model parameters learned using training samples, (ii) validation samples and predictive probability contour plot from model parameters learned using training sample, (iii) augmented samples to maximize influence and predictive probability contour plot from model parameters learned using augmented samples, (iv) validation samples with the predictive probability contour plot from model parameters learned using augmented samples.

$\tilde{x} = G(x, E(x))$ pairs. In practice, we combine the spatial and appearance transformation model by concatenating (τ_s, τ_a) and (s_w, s_b, c_w, c_b). WGAN-GP (Gulrajani et al. (2017)) is used for training G with modifications. Firstly, two discriminators over image space and flow field s space are considered. The discriminator over image space is updated to distinguish $\tilde{x}$ and x, and the discriminator over flow field s space is updated to distinguish $s = G_s(\tau)$ and manually generated flow fields that will result random affine transforms. Also, to prevent the identity transformation, inverse of mean-square error on flow field space and image space are additionally considered when updating G.

5 Experiments

5.1 Toy 2D dataset

To visualize the proposed method, we conduct experiments on the toy 2D dataset. Transitions along x-axis and y-axis are considered to augmentation model, and influence is computed by validation samples. Simple 3-layer neural network is considered as a classification model. In Figure 3, red and blue dots represent data points of class red and blue, and a contour plot of predictive probability is overlaid.

In the first row of Figure 3, an influence value of augmented samples nearby mis-classified validation sample is high, thus augmentation model is learned this direction. In the second row of Figure 3, we generate training and validation data then move validation data to downside to make mismatch with training data. In this setting, Influence value of downside transition is high, thus augmentation model is learned to augment by downside transition.

5.2 MNIST dataset

We ran experiments on the MNIST using only a subset of the class labels to train the end classification models and treating the rest as unlabeled data. We use a four-layer all-convolutional CNN for

Table 1: MNIST accuracies of various data augmentation methods.

	The number of labelled data	
Augmentation method	1%	10%
None	90.2%	97.3%
Heur.	95.9%	99.0%
Ratner (MF)	96.5%	99.2%
Ratner (LSTM)	96.7%	99.1%
Proposed	96.7%	99.3%

classification model. The experiments are conducted in the same setting with Ratner et al. (2017). In Table 1, we list classification accuracies for various data augmentation methods from Ratner et al. (2017). *None* denotes that any augmentation is not applied and *Heur* is the standard heuristic approach of applying random compositions of the given set of transformation operations. *Ratner* denotes the results from Ratner et al. (2017). We used 128 dimensional τ and four-layered convolutional neural network for E and symmetry transposed convolutional neural network for G.

5.3 CIFAR-10 DATASETS

We ran experiments on the CIFAR-10 using a subset of the class labels to train the end classification models and treating the rest as unlabeled data. We use a ResNet-56 (He et al. (2016)). In Table 2, we list classification accuracies for various data augmentation methods from Ratner et al. (2017). We used 128 dimensional τ and four-layered convolutional neural network for E and symmetry transposed convolutional neural network for G.

Table 2: CIFAR-10 accuracies of various data augmentation methods.

	The number of labelled data	
Augmentation method	10%	100%
None	66.0%	87.8%
Heur.	77.5%	92.3%
Ratner (MF)	79.8%	94.4%
Ratner (LSTM)	81.5%	94.0%
Proposed	82.1%	94.8%

6 CONCLUSION

Data augmentation is a technique to avoid overfitting and improve generalization by increasing the size of labeled dataset; however, it is currently conducted in a trial and error manner. Composition of predefined transformations such as rotation, scaling and cropping are performed on training samples, and its effect on performance over test samples can only be empirically evaluated and cannot be predicted. This paper considers an influence function that predicts how generalization in terms of a validation loss is affected by a particular augmented training sample without comparing the performances that includes and excludes it in the training process. A differentiable augmentation model that generalizes the conventional composition of predefined transformations is also proposed. The differentiable augmentation model and reformulation of the influence function allow the augmented model parameters to be updated by backpropagation for minimizing the validation loss. The proposed method is conducive in providing better generalization over conventional data augmentation methods.

REFERENCES

Naman Agarwal, Brian Bullins, and Elad Hazan. Second order stochastic optimization in linear time. *arXiv preprint arXiv:1602.03943*, 2016.

Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. In *Proceedings of AAAI-2018*, 2018. URL http://www.esprockets.com/papers/aaai2018.pdf.

Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3642–3649. IEEE, 2012.

R Dennis Cook. Assessment of local influence. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 133–169, 1986.

R Dennis Cook and Sanford Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pp. 886–893. IEEE, 2005.

Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1734–1747, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pp. 2017–2025, 2015.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation-learning an optimal data augmentation strategy. *IEEE Access*, 2017.

James Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 735–742, 2010.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Anh Nguyen, Jason Yosinski, Yoshua Bengio, Alexey Dosovitskiy, and Jeff Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv preprint arXiv:1612.00005*, 2016.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2226–2234, 2018.

Alexander J Ratner, Henry R Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. *arXiv preprint arXiv:1709.01643*, 2017.

Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1):23–38, 1998.

Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 761–769, 2016a.

Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. *arXiv preprint arXiv:1612.07828*, 2016b.

Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, volume 3, pp. 958–962, 2003.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Leon Sixt, Benjamin Wild, and Tim Landgraf. Rendergan: Generating realistic labeled data. *arXiv preprint arXiv:1611.01331*, 2016.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

William Thomas and R Dennis Cook. Assessing influence on predictions from generalized linear models. *Technometrics*, 32(1):59–65, 1990.

Xiaolong Wang, Abhinav Shrivastava, and Abhinav Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017a.

Xinlong Wang, Mingyu You, and Chunhua Shen. Adversarial generation of training examples for vehicle license plate recognition. *arXiv preprint arXiv:1707.03124*, 2017b.

Bo-Cheng Wei, Yue-Qing Hu, and Wing-Kam Fung. Generalized leverage and its applications. *Scandinavian Journal of statistics*, 25(1):25–37, 1998.

Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*, 2018.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.