Learning Augmentation Network via Influence Functions

Donghoon Lee Hyunsin Park Trung Pham Chang D. Yoo
Korea Advanced Institute of Science and Technology (KAIST)
{iamdh, hs.park, trungpx, cd_yoo}@kaist.ac.kr

Abstract

Data augmentation can impact the generalization performance of an image classification model in a significant way. However, it is currently conducted on the basis of trial and error, and its impact on the generalization performance cannot be predicted during training. This paper considers an influence function that predicts how generalization performance, in terms of validation loss, is affected by a particular augmented training sample. The influence function provides an approximation of the change in validation loss without actually comparing the performances that include and exclude the sample in the training process. Based on this function, a differentiable augmentation network is learned to augment an input training sample to reduce validation loss. The augmented sample is fed into the classification network, and its influence is approximated as a function of the parameters of the last fully-connected layer of the classification network. By backpropagating the influence to the augmentation network, the augmentation network parameters are learned. Experimental results on CIFAR-10, CIFAR-100, and ImageNet show that the proposed method provides better generalization performance than conventional data augmentation methods do.

1. Introduction

In supervised learning, deep neural networks generally require large amounts of labeled data for training. An insufficient number of labeled data will lead to poor generalization performance due to overfitting. One simple method to reduce overfitting and improve generalization performance is to perform data augmentation, whereby each training sample is transformed with a label-preserving transformation to create additional labeled data. Different augmentations can result in significant differences in generalization performances depending on the task [7, 11, 14, 49]; however, this has not yet been extensively explored. Even in a well-studied image classification task [15, 34, 46], training data is often augmented in a manner similar to that performed in training AlexNet [21]. The composition of predefined transformations, such as rotation, translation, cropping, scaling, and color perturbation, is a popular choice; however, choosing the transformations and determining the strength of each transformation, *e.g*. rotation angle, that result in the best performance is often conducted empirically through observations of validation loss [3, 33].

Most recently, strategies for composing the transformations through learning [6, 22, 28, 29, 35] rather than tuning by trial and error have been studied. To learn such a strategy, various learning criteria have been considered. Without considering a classification model, Ratner *et al*. [29] and Sixt *et al*. [35] adopted a strategy for augmenting realistic samples. Given a classification model, [22] and [28] consider antithetical strategies for augmenting samples that minimize and maximize training loss, respectively. It is not yet fully understood why these antithetical studies, which relate data augmentation to training loss, are effective in improving performance on test samples. Cubuk *et al*. [6] considered small child models to compute validation loss for augmenting samples to improve generalization performance. Here, learning requires a reinforcement learning framework with the validation loss as a reward. This necessitates the classification model to be learned from scratch for every update of the augmentation model parameters, requiring thousands of GPU hours for learning.

This paper proposes a data augmentation method that links the impact of augmentation to the validation loss. To predict impact, an influence function [5, 20] is incorporated to compute the effect of a particular augmented training sample on the validation loss. Without a leave-one-out retraining process, the influence function approximates the change in validation loss due to the inclusion or exclusion of the augmented training sample. A differentiable augmentation network is proposed to augment the sample based on its predicted impact on the validation loss. The transformation space of the proposed network encompasses compositions of predefined transformations. The influence function and the differentiable augmentation model enable the gradient of validation loss to flow to the augmentation model.

The remainder of this paper is organized as follows. Section 2 briefly reviews some of the most relevant literature related to the proposed method, while Sections 3 and 4 describe the details of the proposed method. Experimental and comparative results are reported in Section 5. Section 6 summarizes and concludes the paper.

2. Related work

2.1. Data augmentation methods

This sub-section provides a brief overview of data augmentation methods in the following three categories: (i) unsupervised methods that do not involve labels 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 methods, which use a composition of predefined transformations, such as rotating, translating, cropping, scaling and color perturbation [15, 21, 34, 46]. The transformations are manually chosen through trial and error by empirically observing validation loss [3, 33]. Ratner *et al*. [29] considers a generator that generates a sequence of predefined transformations. Given the sequence, the training sample is augmented by consecutively applying predefined transformations. The generator is learned in the generative adversarial network (GAN) framework [13]. The generated sequence produces a realistic augmented sample; however, the classifier is not involved during the learning process and the effect of data augmentation can only be observed through trial and error.

Adversarial methods Adversarial methods include hard sample mining that collects or augments samples that are misclassified by the current classification model. They have been used in training support vector machines (SVM) [8], boosted decision trees [11], shallow neural networks [30], and deep neural networks [31]. Wang *et al*. [42] and Peng *et al*. [28] selected hard samples by adversarially updating the ranges of predefined transformations, such as occluding [28, 42], scaling, and rotating [28].

Adversarial examples are slightly perturbed or transformed samples that result in a classification model predicting an incorrect answer with high confidence [38]. For these methods, flexible and complex transformation models [2, 45] cannot be used to generate adversarial examples [28]. Training the classification model with adversarial examples will improve robustness to those adversarial examples but may degrade performance on clean test samples [40, 41]. Recent studies have shown that adversarial updates in convolution-based transformations [2] and spatial transformations [45] can potentially generate adversarial examples.

[1]Unsupervised methods by chance can lead to validation loss; however, augmentation is conducted without any supervision to optimize an objective.

Supervised methods Lemley *et al*. [22] designed an augmentation model that learns to augment samples that reduce training classification loss, but the performance on test samples can only be empirically evaluated and not be predicted. Cubuk *et al*. [6] considered small child classification models for computing validation loss to evaluate several augmentation policies over predefined transformations. However, learning requires a reinforcement learning framework as predefined transformations are non-differentiable and cannot be backpropagated. The child model must be learned from scratch for every update of the augmentation model parameters and thus requires thousands of GPU hours. Furthermore, the validation loss of the small child model may not be a good predictor of the validation loss of the final classification model.

2.2. Generative adversarial networks for data augmentation

Goodfellow *et al*. [13] proposed a framework for training a deep generative network in an adversarial manner referred to as the generative adversarial network (GAN). The framework simultaneously trains two networks: a generator that generates a sample from random noise and a discriminator that estimates the probability that a sample came from the training data rather than from the generator. An adversarial process is formed by a two-player minimax game in which a discriminator tries to distinguish the source of a sample while a generator tries to generate an indistinguishable sample.

Several studies have demonstrated the potential of using the GAN framework in data augmentation by either improving the realism of synthetic samples [32, 35, 43] or by generating class-conditional images [24, 26, 25]. However, generative models are generally known to require more data to train than a classification model, and for training, additional synthetic [32, 35, 43] and/or unlabeled [24, 26, 25] samples are required.

2.3. Influence functions

The influence function is a function from robust statistics [5] to estimate how model parameters change due to up-weighting a particular training sample. Cook and Weisberg [5] developed influence function of removing training data in learning a linear model, and in [4, 39, 44], influence functions concerning a wider variety of perturbations were studied. Koh and Liang [20] considered influence function to non-convex and highly-complex models, including deep neural networks, by using efficient approximations based on Hessian-vector products (HVPs) [27], conjugate gradients

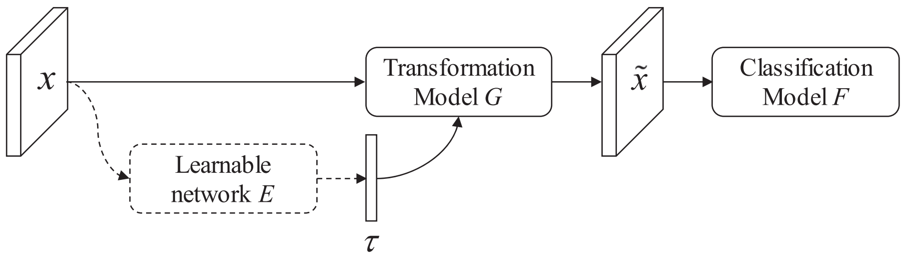


Figure 1: A generic data augmentation framework for a classification task. An input training sample x is transformed to $\tilde{x}$ by a transformation model, which is parameterized by τ. The conventional method usually defines G as a composition of predefined transformations based on randomly sampled range τ (solid line path). In this paper, G is a differentiable network. A learnable network E estimates τ given x to obtain the transformed sample $\tilde{x}$, where $\tilde{x}$ maximizes the generalization performance of the classification model (dashed line path).

[23], and stochastic estimation [1]. They also considered the influence of up-weighting a particular training sample on validation loss. In [20], the influence functions are used for various purposes: debugging models, detecting dataset errors, and creating training-set attacks.

3. Augmented data evaluation

Figure 1 depicts a general framework for tuning or learning an augmentation model that may involve evaluating an augmented sample. Given a classification model, the evaluation is often performed by computing the training loss of the sample; however, the effect on test samples cannot be predicted and can only be empirically evaluated. To evaluate the impact of an augmented sample using validation loss requires learning two classifiers: one that includes and the other that excludes the sample during the learning process for comparing their performances. Learning both classifiers is computationally expensive as the models need to be fully trained from scratch and evaluated over all validation samples. Rather than repeating this prohibitive process, a method is proposed to approximate the validation loss difference due to a particular augmented sample, thus eliminating the retraining process.

3.1. Problem set up

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

$$\hat{\theta}(\mathbf{z}^{\text{tr}}) = \underset{\theta \in \Theta}{\arg\min}\, \mathcal{L}(\mathbf{z}^{\text{tr}}, \theta), \tag{1}$$

where the empirical risk is given as

$$\mathcal{L}(\mathbf{z}^{\text{tr}}, \theta) = \frac{1}{N} \sum_{i: z_i \in \mathbf{z}^{\text{tr}}} l(z_i, \hat{\theta}). \tag{2}$$

To measure the generalization performance of the classification model F, validation data $\mathbf{z}^{\text{val}} = \{z_j\}_{j=N+1}^{(N+M)}$ is often considered. Generalization performance is approximated as the average loss over validation data $\mathbf{z}^{\text{val}}$ with parameter $\hat{\theta}(\mathbf{z}^{\text{tr}})$ as

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

Consider a label preserving transformation G that maps $\mathcal{X} \mapsto \mathcal{X}$ such as the one shown in Figure 1. Let τ be the 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. In addition, consider a learnable network E parameterized by ϕ. It estimates τ given x. Thus, $\tilde{x} = G(x, E(x, \phi))$.

Given the transformation model G, the goal is to find the optimal τ for each input x; otherwise, find optimal parameters ϕ of E that minimizes the validation loss when the classification model is learned using $\tilde{\mathbf{z}}^{\text{tr}}$. This is mathematically represented as follows

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

where

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

Solving Equations 4–5 requires a bilevel optimization where one problem is embedded (nested) within another.

3.2. Influence by upweighting a training sample

Consider a change in model parameters θ due to the exclusion of a particular training sample z_i. Formally, this change is given as $\hat{\theta}(\mathbf{z}^{\text{tr}} \backslash z_i) - \hat{\theta}(\mathbf{z}^{\text{tr}})$. Influence functions [5, 20] provide an efficient approximation without a retraining process to obtain $\hat{\theta}(\mathbf{z}^{\text{tr}} \backslash z_i)$. Let us consider the change in model parameters due to upweighting z_i by an amount of $\epsilon l(z_i, \theta)$ 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 [5], 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

$$\begin{aligned} \mathcal{I}_{\text{up, params}}(z_i) &\triangleq \frac{d\hat{\theta}(\mathbf{z}^{\text{tr}} \cup \epsilon z_i)}{d\epsilon}\Big|_{\epsilon=0} && (8) \\ &= -H(\hat{\theta}(\mathbf{z}^{\text{tr}}))^{-1}\nabla_\theta l(z_i, \hat{\theta}(\mathbf{z}^{\text{tr}})). && (9) \end{aligned}$$

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

Using Equation 9 and applying the chain rule, the influence of up-weighting $z_i \in \mathbf{z}^{\text{tr}}$ on the validation loss at $z_j \in \mathbf{z}^{\text{val}}$ can be approximated [20] as shown below:

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

where

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

For the validation dataset $\mathbf{z}^{\text{val}}$, Equation 12 can be expanded given by:

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

Equation 11 describes a gradient of $l(z_j, \hat{\theta}(\mathbf{z}^{\text{tr}} \cup \epsilon z_i))$ with respect to ϵ at nearby $\epsilon = 0$. The influence of excluding z_i can be approximated by Equation 10.

3.3. Influence by augmentation

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

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

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

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

The influence function $\mathcal{I}_{\text{aug, loss}}(z_i, \tilde{z}_i, \mathbf{z}^{\text{val}})$ predicts the differences between the influences of the augmented sample and the original sample. In addition, the influence function can be used to evaluate the effectiveness of the augmented sample $\tilde{z}_i$ compared to that of the original sample.

3.4. Reformulation of influence functions

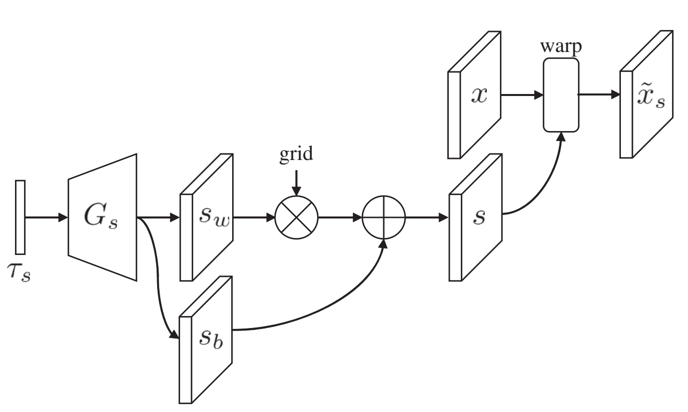
To compute the influence functions efficiently, techniques such as conjugate gradients and stochastic estimation are adopted, as used in [20]. Given F, inverse Hessian-vector products (iHVPs) $s_{\text{val}} = H(\hat{\theta}(\mathbf{z}^{\text{tr}}))^{-1}\nabla_\theta \mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\mathbf{z}^{\text{tr}}))$ are precomputed and cumulated for validation samples using conjugate gradients and stochastic estimation techniques. The iHVPs are fixed during learning G and E and are used to compute influence functions of augmented samples. The influence function is further approximated through considerations of only 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 makes it possible to represent $s_{\text{val}}\nabla_\theta l(z_i, \hat{\theta}(\mathbf{z}^{\text{tr}}))$ in a closed form by regarding s_{val} as a fixed vector. The gradient from this flows through fixed F and is then used to update G and E by the chain rule.

4. Transformation models

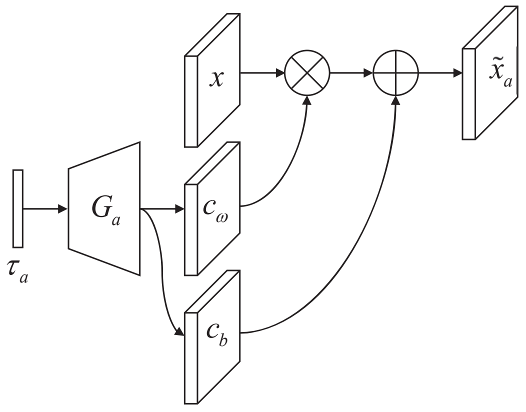
Transformations for augmenting images can be categorized as either spatial transformations or appearance transformations. In this section, the proposed transformation models that generalize both transformations are described. As the models are differentiable, the gradient from the influence function can be propagated to G and E during learning.

4.1. Spatial transformation model

Spatial transformations include random flip, crop, scaling, rotation, shearing, translation, and affine transformations. These transformations can be defined by the coordinate change in pixel locations. The proposed transformation model for spatial transformation is illustrated in Figure 2-(a). Spatial transformation is defined—in a way similar to



(a) Spatial transformation model



(b) Appearance transformation model

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

[17, 45]—by

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

Here, i, j denotes the source coordinate, and $s_w(i,j)$ and $s_b(i,j)$ denote the multiplication factor and bias, respectively. Note that when global average pooling is performed on s_w and s_b, the transformation is reduced to the affine transformation. In this formulation, the spatial transformation is fully defined by s_w and s_b that are obtained by feeding τ_s into G_s.

Spatial transformation model G_g takes an input τ_s and then generates s_w and s_b. Flow field s denotes the coordinates to be transformed and is obtained by performing operations such as the point-wise affine transform by s_w and s_b. The warp operation in Figure 2-(a) transforms x to $\tilde{x}$ based on the bilinear interpolation indexed by s. The model is designed by a stacked transposed convolutional network with $4 + 2$ channels in its final layer. Once s is obtained, 4×4 average pooling is applied to smoothing s. The image is then warped by bilinear interpolation, which is differentiable operation [17]. All computations in this formulation can be implemented by a feed-forward neural network.

4.2. Appearance transformation model

Transformations in appearance include alterations of contrast, brightness, color, and hue. These transformations can be formed by 1×1 spatial dimension filters. Thus, to formulate the appearance transformation model, generating 1×1 spatial filters is of primary concern. The proposed transformation model for the appearance transformation is illustrated in Figure 2-(b). The appearance transformation model G_a takes an input τ_a and then generates c_w and c_b, which are average pooled in the same way as the spatial transformation model. The transformed image $\tilde{x}$ is obtained by $x + \delta x$, where δx is obtained by filtering x with filter weights c_w and bias c_b. The model is designed by a transposed convolutional network with $3 \times 3 + 3$ channels for RGB images in its final layer. For grey images, the appearance transformation model is not considered.

4.3. Learning transformation models

During learning, G is trained based on the GAN framework. Then E and G are trained to predict τ which is used to augment inputs, which maximize the influence function for each x and $\tilde{x} = G(x, E(x))$ pair. In practice, the spatial and appearance transformation models are combined by concatenating (τ_s, τ_a) and (s_w, s_b, c_w, c_b). By sharing conv (E) and convtr (G) blocks for the spatial and appearance transformations, the combined transformation model has single τ (implicitly include τ_s and τ_a) and outputs s_w, s_b, c_w, c_b. Thus, the spatial and appearance transformation models are learned together during entire learning process. The relativistic average GAN (RaGAN) [18] is used for training G with modification. To avoid trivial solutions (identity transformation), a simple trick (with standard loss function) to match distributions of transformed image and baseline augmented image (instead of original image) is used. Using this method, G was pretrained to mimic the baseline augmenta-

Dataset	%	Model	None	Heur.	Ratner MF [29]	Ratner LSTM [29]	Proposed
MNIST	1	4 layer CNN	9.8	4.1	3.5	3.3	3.1
	10	4 layer CNN	2.7	1.0	0.8	0.9	0.8
CIFAR-10	10	ResNet-56 [15]	34.0	22.5	20.2	18.5	17.7
CIFAR-10	100	ResNet-56 [15]	12.2	7.7	5.6	6.0	5.2
CIFAR-100	100	ResNet-56 [15]	36.3	31.6	-	-	29.6

Table 1: Test set error rates (%) of classification models trained on subsamples of the labeled training data. The experiments are conducted under the same setting as [29].

Dataset	Model	Baseline	Cutout [10]	AutoAug. [6]	Proposed
CIFAR-10	Wide-ResNet-28-10 [48]	3.9	3.1	2.6	2.8
	Shake-Shake (26 2x96d) [12]	2.9	2.6	2.0	2.1
	PyramidNet+ShakeDrop [47]	2.7	2.3	1.5	1.7
CIFAR-100	Wide-ResNet-28-10 [48]	18.8	18.4	17.1	17.3
	Shake-Shake (26 2x96d) [12]	17.1	16.0	14.3	14.6
	PyramidNet+ShakeDrop [47]	14.0	12.2	10.7	11.9

Table 2: Test set error rates (%) on CIFAR-10 and CIFAR-100 datasets. The experiments are conducted under the same setting as [6].

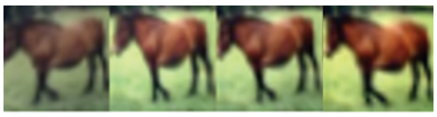


Figure 3: Diverse transforms can be obtained from the same input image of CIFAR-10 dataset using dropout τ.

tion method. Two discriminators over the image space and feature space—feature map in the last fully-connected layer of F—are considered. The discriminator over the feature space is defined by single fully-connected layer. Given F, the learning steps for training E and G are arranged as follows: (1) pretrain G in the GAN framework, (2) compute iHVPS, and (3) train E and G to maximize the influence. Omitting the pretrain stage does not affect accuracy. Including this stage provides better initialization for G which leads to a more stable learning and faster convergence during stages (2) and (3). Classification model F is retrained using augmented samples from learned E and G thereafter. During learning E and G, dropout [36] with probability 0.5 is applied to τ. Dropout is retained during retraining F to obtain diverse augmented samples as shown in Figure 3.

5. Experiments

5.1. MNIST dataset

Experiments were performed on the MNIST using only a subset of class labels to train the classification models and treating the rest as unlabeled data. A 4-layer all-convolutional neural network was used for classification. Experiments were conducted under the same settings as [29]. In Table 1, classification errors on test set are listed for various data augmentation methods from [29]. *None* indicates that no augmentation is applied, and *Heur* is the standard heuristic approach of applying random compositions of the given set of transformation operations. *Ratner* denotes the results from [29]. A 128 dimensional τ and 4-layer convolutional neural network was used. For G, a 4-layer transposed convolutional neural network was used. For the proposed method, the random cropping technique is used as did in [29]. The proposed method shows lower test error than those of [29]. Architectures and hyper-parameters in the optimization are shown in Supplementary-B.

5.2. CIFAR-10 and CIFAR-100 datasets

Performance comparison is made among the proposed method, heuristic, [29], and [6] under the exact equal settings. For fair performance comparison, benchmark script made available by the authors of [29] and [6] was used to obtain exact identical baseline performance. A 128 dimensional τ and 5-layer convolutional neural network were used. For G, a 5-layer transposed convolutional neural network was used. All of the reported results are averaged over 5 runs. The detailed settings of the hyper-parameters are described in Supplementary-B.

For comparison with [29], a subset of the class labels was used to train the classification models and the rest was treated as unlabeled data. A ResNet-56 [15] was used. For the proposed method, random cropping and horizontal flip

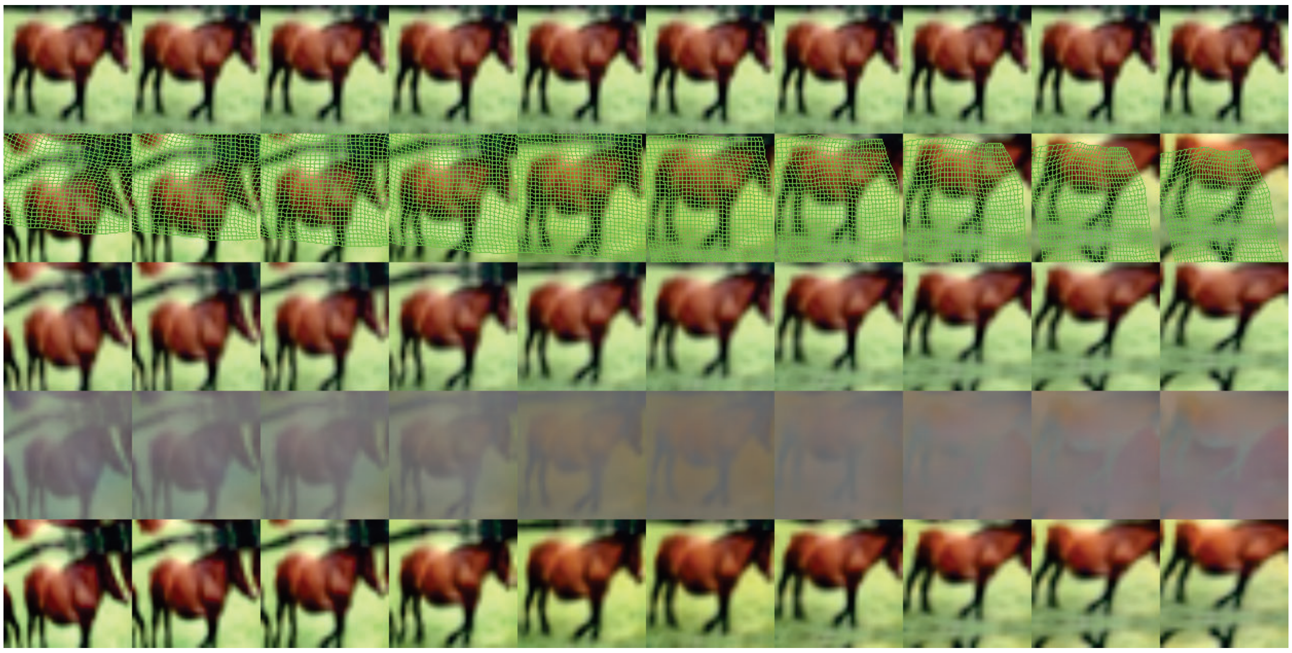


Figure 4: Images are transformed by applying ten linearly interpolated τ's to the same training sample in the CIFAR-10 dataset. Each row represents: (i) the training image, (ii) spatial transformation model outputs, (iii) spatially transformed images, (iv) appearance transformation model outputs, and (v) final transformed images.

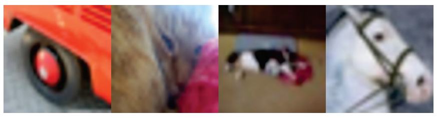


Figure 5: Some corrected test samples of CIFAR-10 dataset.

were used as was used in [29]. For the comparison with [6], the experiments were conducted with the same setting as [6]. Wide-ResNet-28-10 [48], Shake-Shake [12], and Shake-Drop [47] were used as classification models. The baseline augmentation follows the convention for state-of-the-art CIFAR-10 models: standardizing the data and using horizontal flips with 50% probability, random crops, and finally Cutout [10] with 16x16 pixels. The proposed augmentation was applied in addition to the baseline as used in [6].

In Table 1 and Table 2, test errors are listed to compare the proposed method with [29] and [6], respectively. For CIFAR-10, detailed in Table 1, the proposed method shows performance improvement over [29][2]. In Table 2, the proposed method achieved comparable performances to [6] but approximately 600 times faster than [6] in terms of GPU hours (Table 4).

In Figure 4, transformed images from the same input image by traversing in the τ-space are shown. Images are transformed by applying ten linearly interpolated τ's to the same training sample in the CIFAR-10 dataset. Each row represents: (i) the training image, (ii) spatial transformation model outputs, (iii) spatially transformed images, (iv) appearance transformation model outputs, and (v) final transformed images. Given an input, we note that learned spatial transformations are (relatively) specific while learned appearance transformations allow a wider range of transformations (Figure 3). Corrected test samples by using learned augmentation model (Figure 5) show some similarities: (1) object is only partially shown or the size is small and (2) the color of the object is rare (*e.g.* white horse).

To analyze the effect of appearance and spatial transform, We conducted experiments on CIFAR-10 using Res-56 and the following results (error rates) are obtained: (1) Heur. (7.7%, Table 1), (2) pretrained G (6.4%), (3) spatial transform only (6.1%), (4) appearance transform only (5.8%), (5) combined transform (in serial) (5.4%), and (6) combined transform (in parallel) (5.2%, Table 1).

For the dimensions of τ, the results are as follows: (1) 32 dim (5.8%), (2) 64 dim (5.4%), (3) 128 dim (5.2%, Table 1), and (4) 256 dim (5.2%).

[2]Note that the performance gap due to the presence of label supervision during the learning of an augmentation module is smaller than that during the learning of a classification module. Supervision of the augmentation module may not show significant performance gain because the performance is measured by the end classifier, which already includes label supervision.

Model	Baseline [6]	Baseline (ours)	AutoAug. [6]	Proposed
ResNet-50 [15]	76.3 / 93.1	76.1 / 93.0	77.6 / 93.8	77.1 / 93.4
ResNet-200 [15]	78.5 / 94.2	78.1 / 94.0	80.0 / 95.0	79.0 / 94.6

Table 3: Validation set Top-1 / Top-5 accuracy (%) on ImageNet dataset. The experiments are conducted under the same setting as [6]. All results are obtained using 1-crop testing.

Dataset	AutoAug. [6]	Proposed
CIFAR-10	5,000	8
ImageNet	15,000	40

Table 4: GPU hours comparison of AutoAugment and the proposed method. Ours are estimated with Titan-X Pascal.

5.3. ImageNet dataset

The experiments on the ImageNet dataset [9] were conducted focusing on the comparisons between baseline, AutoAugment [6], and the proposed method. To train the augmentation model, 50,000 samples were split from the original training set to create a hyper-validation set, and the remaining was used as the training set. The hyper-validation set was used to compute the influence function. The ResNet-50 and ResNet-200 [15] were used as backbone architectures. During pretraining the transformation models, the progressive learning technique proposed in [19] was adopted to stabilize the GAN training process. In training F, a batch size of 4096 and a learning rate of 1.6 were used. The learning rate was decayed by 10-fold at epochs 90, 180, and 240 as used in [6]. Due to the unavailability of ImageNet performance reproduction script of [6], our system had to run with in-house script resulting in a slightly lower benchmark performance than that reported in [6]. For baseline augmentation, the standard Inception-style preprocessing was used, which involves scaling pixel values to [-1,1], horizontal flips with 50% probability, scaling, cropping, aspect ratio change, and random distortions of colors [16, 37].

The trained ResNet-50 and ResNet-200 [15] were then used to compute the influence function. A 128 dimensional τ and 8-layer convolutional neural network and 8-layer transposed convolutional neural network were used for E and G, respectively. Refer to Supplementary-B for the details. For the proposed method, inputs were preprocessed by the baseline augmentation method as did in [6]. In Table 3, classification accuracies are listed for the comparison with AutoAugment [6]. The proposed method achieved the comparable performance to [6], and the training time was reduced from 15,000 GPU hours to 40 GPU hours as shown in Table 4 (375 times faster).

6. Conclusion

To improve data augmentation, the influence function that predicts the effects of a particular augmented training sample on generalization performance is proposed. The differentiable augmentation network is learned to generate augmented samples that maximize the influence function, thereby minimizing the validation loss. The influence function provides an approximation of the change in validation loss without comparing the performances that include and exclude the augmented training sample in the training process. The differentiable augmentation network and the reformulation of the influence function allow augmentation network parameters to be updated by backpropagation. Also, the proposed augmentation network can generalize the conventional composition of predefined transformations. The experimental results confirmed that the proposed method provides better generalization than conventional data augmentation methods do on various datasets including CIFAR-10, CIFAR-100, and ImageNet.

Acknowledgements

This work was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2016-0-00563, Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion) and Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01396, Development of framework for analyzing, detecting, mitigating of bias in AI model and training data).

References

[1] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization in linear time. *stat*, 1050:15, 2016. 3

[2] Shumeet Baluja and Ian Fischer. Learning to attack: adversarial transformation networks. In *AAAI Conference on Artificial Intelligence*, 2018. 2

[3] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1, 2

[4] R Dennis Cook. Assessment of local influence. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 133–169, 1986. 2

[5] R Dennis Cook and Sanford Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980. 1, 2, 3, 4

[6] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: learning augmentation policies from data. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 6, 7, 8

[7] Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 23(9):1469–1477, 2015. 1

[8] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 2

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 8

[10] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 6, 7

[11] 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 (TPMAI)*, 38(9):1734–1747, 2016. 1, 2

[12] Xavier Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017. 6, 7

[13] 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 (NeurIPS)*, 2014. 2

[14] Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014. 1

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 6, 8

[16] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 8

[17] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 5

[18] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*, 2018. 5

[19] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations (ICML)*, 2018. 8

[20] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*, 2017. 1, 2, 3, 4

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012. 1, 2

[22] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, 5:5858–5869, 2017. 1, 2

[23] James Martens. Deep learning via hessian-free optimization. In *International Conference on Machine Learning (ICML)*, 2010. 3

[24] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2

[25] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: conditional iterative generation of images in latent space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[26] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International Conference on Machine Learning (ICML)*, 2017. 2

[27] Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994. 2

[28] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[29] Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 1, 2, 6, 7

[30] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 20(1):23–38, 1998. 2

[31] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[32] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[33] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, 2003. 1, 2

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 1, 2

[35] Leon Sixt, Benjamin Wild, and Tim Landgraf. Rendergan: Generating realistic labeled data. *Frontiers in Robotics and AI*, 5:66, 2018. 1, 2

[36] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, 2014. 6

[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 8

[38] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014. 2

[39] William Thomas and R Dennis Cook. Assessing influence on predictions from generalized linear models. *Technometrics*, 32(1):59–65, 1990. 2

[40] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: attacks and defenses. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[41] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations (ICLR)*, 2019. 2

[42] 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 (CVPR)*, 2017. 2

[43] Xinlong Wang, Mingyu You, and Chunhua Shen. Adversarial generation of training examples for vehicle license plate recognition. *arXiv preprint arXiv:1707.03124*, 2017. 2

[44] Bo-Cheng Wei, Yue-Qing Hu, and Wing-Kam Fung. Generalized leverage and its applications. *Scandinavian Journal of statistics*, 25(1):25–37, 1998. 2

[45] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018. 2, 5

[46] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[47] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6, 7

[48] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*, 2016. 6, 7

[49] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 1