MaxUp: Lightweight Adversarial Training with Data Augmentation Improves Neural Network Training

Chengyue Gong, Tongzheng Ren, Mao Ye, Qiang Liu
University of Texas, Austin
{cygong, tongzheng, my21, lqiang}@cs.utexas.edu

Abstract

We propose MaxUp*, a simple and effective technique for improving the generalization performance of machine learning models, especially deep neural networks. The idea is to generate a set of augmented data with some random perturbations or transforms, and minimize the maximum, or worst case loss over the augmented data. By doing so, we implicitly introduce a smoothness or robustness regularization against the random perturbations, and hence improve the generation performance. For example, in the case of Gaussian perturbation,* MaxUp *is asymptotically equivalent to using the gradient norm of the loss as a penalty to encourage smoothness. We test* MaxUp *on a range of tasks, including image classification, 3D point cloud classification, and adversarial certification, on which* MaxUp *consistently outperforms the baseline methods, without introducing substantial computational overhead. In particular, we improve ImageNet classification from the top-1 accuracy* 85.5% *without extra data to* 85.8%.

1. Introduction

A central theme of machine learning is to alleviate the issue of over-fitting and improve the generalization performance on testing data. This is often achieved by leveraging important prior knowledge of the models and data of interest. For example, the regularization-based methods introduce penalty on the complexity of the model, which often amount to enforcing certain smoothness properties. Data augmentation techniques, on the other hand, leverage important invariance properties of the data (such as the shift and rotation invariance of images) to improve the performance. Novel approaches that exploit important knowledge of the models and data hold the potential of substantially improving the performance of machine learning systems.

We propose *MaxUp*, a simple yet powerful training method to improve the generalization performance and alleviate the over-fitting issue. Different from standard methods that minimize the average risk on the observed data, *MaxUp* generates a set of random perturbations or transforms of each observed data point, and minimizes the average risk of the *worst* augmented data of each data point. This allows us to enforce robustness against the random perturbations and transforms, and hence improve the generalization performance. *MaxUp* can easily leverage arbitrary state-of-the-art data augmentation schemes [e.g. 5, 6, 8, 26, 33] and substantially improves over them by minimizing the worst (instead of average) risks on the augmented data, without adding significant computational ahead.

Theoretically, in the case of Gaussian perturbation, we show that *MaxUp* effectively introduces a *gradient-norm regularization term* that serves to encourage the smoothness of the loss function, which does not appear in standard data augmentation methods that minimize the average risk.

MaxUp can be viewed as a "lightweight" variant of adversarial training against adversarial input perturbations [e.g. 16, 24], but is mainly designed to improve the generalization on the clean data, instead of robustness on perturbed data (although *MaxUp* does also increase the adversarial robustness in Gaussian adversarial certification as we shown in our experiments (Section 4.3)). In addition, compared with standard adversarial training methods such as projected gradient descent (PGD) [16], *MaxUp* is much simpler and computationally much faster, and can be easily adapted to increase various robustness defined by the corresponding data augmentation schemes.

We test *MaxUp* on several challenging tasks: image classification, point cloud classification, certified defense against adversarial examples [4], and language modeling (see Appendix). We find that *MaxUp* can leverage different state-of-the-art data augmentation methods and boost their performance to achieve new state-of-the-art on a range of tasks, datasets, and neural architectures. In particular, we set up strong results on ImageNet classification without extra data, which improves the 85.5% top-1 accuracy of [28] to 85.8%. For the adversarial certification task, we find *Maxup* allows us to train better certified robust classifiers than prior arts such as the PGD-based method by [18].

2. Main Method

We start with introducing the main idea of *MaxUp*, and then discuss its effect of introducing smoothness regularization in Section 2.1.

ERM Giving a dataset $\mathcal{D}_n = \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$, learning often reduces to a form of empirical risk minimization (ERM):

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_n} \left[L(\boldsymbol{x}, \boldsymbol{\theta})\right], \tag{1}$$

where $\boldsymbol{\theta}$ is a parameter vector of interest (e.g., the weights of a neural network), and $L(\boldsymbol{x}, \boldsymbol{\theta})$ denotes the loss associated with data point $\boldsymbol{x}$ ($\boldsymbol{x}$ can include both label and feature). A key issue of ERM is the risk of over-fitting, especially when the data information is insufficient.

MaxUp We propose *MaxUp* to alleviate over-fitting. The idea is to generate a set of random augmented data and minimize the maximum loss over the augmented data.

Formally, for each data point $\boldsymbol{x}$ in $\mathcal{D}_n$, we generate a set of perturbed data points $\{\boldsymbol{x}_i'\}_{i=1}^m$ that are similar to $\boldsymbol{x}$, and estimate $\boldsymbol{\theta}$ by minimizing the maximum loss over $\{\boldsymbol{x}_i'\}$:

$$\textit{MaxUp:} \qquad \min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_n} \left[\max_{i\in[m]} L(\boldsymbol{x}_i', \boldsymbol{\theta})\right]. \tag{2}$$

This loss can be easily minimized with stochastic gradient descent (SGD). Note that the gradient of the maximum loss is simply the gradient of the worst copy, that is,

$$\nabla_{\boldsymbol{\theta}} \left(\max_{i\in[m]} L(\boldsymbol{x}_i', \boldsymbol{\theta})\right) = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}_{i^*}', \boldsymbol{\theta}), \tag{3}$$

where $i^* = \arg\max_{i\in[m]} L(\boldsymbol{x}_i', \boldsymbol{\theta})$. This yields a simple and practical algorithm shown in Algorithm 1.

In our work, we assume the augmented data $\{\boldsymbol{x}_i'\}_{i=1}^m$ is *i.i.d.* generated from a distribution $\mathbb{P}(\cdot|\boldsymbol{x})$. The $\mathbb{P}(\cdot|\boldsymbol{x})$ can be based on small perturbations around $\boldsymbol{x}$, e.g., $\mathbb{P}(\cdot|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}, \sigma^2\boldsymbol{I})$, the Gaussian distribution with mean $\boldsymbol{x}$ and isotropic variance σ^2. The $\mathbb{P}(\cdot|\boldsymbol{x})$ can also be constructed based on data transformations that are widely used in the data augmentation literature, such as random crops, equalizing, rotations, and clips for images [see e.g 5, 6, 8].

2.1. *MaxUp* as a Smoothness Regularization

We provide a theoretical interpretation of *Maxup* as introducing an extra *gradient-norm regularization* over standard data augmentation to encourage smoothness. Different from the *maximum loss* (2) in *MaxUp*, typical data augmentation methods minimize the *average loss*, that is,

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_n} \left[\frac{1}{m}\sum_{i=1}^m L(\boldsymbol{x}_i', \boldsymbol{\theta})\right]. \tag{4}$$

It turns out the key difference between (2) and (7) is that (2) implicitly introduces an additional smoothness regularization. To set up notation, we define

$$\begin{aligned} L_{\mathbb{P},m}^{max}(\boldsymbol{x}, \boldsymbol{\theta}) &= \mathbb{E}_{\mathbb{P}(\cdot|\boldsymbol{x})} \left[\max_{i\in[m]} L(\boldsymbol{x}_i', \boldsymbol{\theta})\right], \\ L_{\mathbb{P},m}^{avg}(\boldsymbol{x}, \boldsymbol{\theta}) &= \mathbb{E}_{\mathbb{P}(\cdot|\boldsymbol{x})} \left[\frac{1}{m}\sum_{i=1}^m L(\boldsymbol{x}_i', \boldsymbol{\theta})\right], \end{aligned} \tag{5}$$

where $L_{\mathbb{P},m}^{max}(\boldsymbol{x}, \boldsymbol{\theta})$ and $L_{\mathbb{P},m}^{avg}(\boldsymbol{x}, \boldsymbol{\theta})$ denote the expected *MaxUp* and typical average risk on data point $\boldsymbol{x}$ with m augmented copies. By the unbiasedness of averaging, *MaxUp* with $m = 1$ is equivalent to the average loss in exapectation, that is, $L_{\mathbb{P},1}^{max}(\boldsymbol{x}, \boldsymbol{\theta}) = L_{\mathbb{P},m}^{avg}(\boldsymbol{x}, \boldsymbol{\theta})$, $\forall m$. The theorem below shows that *MaxUp* with $m > 1$ effectively introduces a gradient-norm regularization over the averaging-based data augmentation *without explicitly calculating the gradient*.

Theorem 1 (MaxUp as Gradient-Norm Regularization)**.** *Assume $L(\boldsymbol{x}, \boldsymbol{\theta})$ is second-order differentiable w.r.t. $\boldsymbol{x} \in \mathbb{R}^d$. Assume the variance of $\mathbb{P}(\cdot|\boldsymbol{x})$ is bounded by σ^2.*

1) We have for every positive integer m,

$$\begin{aligned} &L_{\mathbb{P},m}^{max}(\boldsymbol{x}, \boldsymbol{\theta}) = L_{\mathbb{P},m}^{avg}(\boldsymbol{x}, \boldsymbol{\theta}) + \Phi(\boldsymbol{x}, \boldsymbol{\theta}) + \mathbf{O}(\sigma^2), \\ &\textit{with} \quad c_{\mathbb{P},m}^- \left\|\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\theta})\right\| \le \Phi(\boldsymbol{x}, \boldsymbol{\theta}) \le c_{\mathbb{P},m}^+ \left\|\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\theta})\right\|, \end{aligned} \tag{6}$$

where $c_{\mathbb{P},m}^+ \ge c_{\mathbb{P},m}^- \ge 0$ are two non-negative coefficients. We have $c_{\mathbb{P},m}^- > 0$ if $\boldsymbol{\alpha}^\top \boldsymbol{x}_1'$ is not deterministic for any $\boldsymbol{\alpha} \in \mathbb{R}^d$ with $\|\boldsymbol{\alpha}\| = 1$. Here $\|\cdot\|$ can be any notion of vector norm in $\mathbb{R}^d$.

2) If $\mathbb{P}(\cdot|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}, \sigma^2\boldsymbol{I})$ and $\|\cdot\|$ is the L2 norm, then the inequality in (6) *becomes equality, and $c_{\mathbb{P},m}^+ = c_{\mathbb{P},m}^- \asymp \sigma\sqrt{\log m}$, where $\asymp$ denotes asymptotical equality.*

Therefore, unless $m = 1$ or $\mathbb{P}(\cdot|\boldsymbol{x})$ is deterministic along some direction, *MaxUp* effectively introduces an extra Lipschitz-like regularization $c_{\mathbb{P},m} \left\|\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\theta})\right\|_2$ over the standard data augmentation, which encourages the smoothness of $L(\boldsymbol{x}, \boldsymbol{\theta})$ w.r.t. the input $\boldsymbol{x}$. The strength of the regularization is controlled by $c_{\mathbb{P},m}$, which increases with the number of samples m. Please refer to appendix for details.

Proof of Theorem 1. Let $\boldsymbol{x}_1''$ be an independent copy of $\boldsymbol{x}_1'$. Using Taylor expansion, we have

$$\begin{aligned} &L_{\mathbb{P},m}^{max}(\boldsymbol{x}, \boldsymbol{\theta}) \\ &= \mathbb{E}\left[\max_{i\in[m]} L(\boldsymbol{x}_i', \boldsymbol{\theta})\right] \\ &= L_{\mathbb{P},m}^{avg}(\boldsymbol{x}, \boldsymbol{\theta}) + \mathbb{E}\left[\max_{i\in[m]} \left(L(\boldsymbol{x}_i', \boldsymbol{\theta}) - L(\boldsymbol{x}_1'', \boldsymbol{\theta})\right)\right] \\ &= L_{\mathbb{P},m}^{avg}(\boldsymbol{x}, \boldsymbol{\theta}) + \mathbb{E}\left[\max_{i\in[q]} \langle \nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\theta}),\ \boldsymbol{x}_i' - \boldsymbol{x}_1''\rangle\right] + \mathbf{O}(\sigma^2). \end{aligned}$$

Algorithm 1 *MaxUp* with Stochastic Gradient Descent

Input: Dataset $\mathcal{D}_n = \{\boldsymbol{x}_i\}_{i=1}^n$; transformation distribution $\mathbb{P}(\cdot|\boldsymbol{x})$; number of augmented data m; initialization $\boldsymbol{\theta}_0$; SGD parameters (batch size, step size η, etc).

Repeat

Draw a mini-batch $\mathcal{M}$ from $\mathcal{D}_n$, and update $\boldsymbol{\theta}$ via

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \mathbb{E}_{\boldsymbol{x}\sim\mathcal{M}} \left[\nabla_{\boldsymbol{\theta}} \left(\max_{i\in[m]} L(\boldsymbol{x}_i', \boldsymbol{\theta}) \right) \right],$$

where $\{\boldsymbol{x}_i'\}_{i=1}^m$ are drawn *i.i.d.* from $\mathbb{P}(\cdot|\boldsymbol{x})$ for each $\boldsymbol{x}$ in the mini batch $\mathcal{M}$. See Equation 3.

Until Convergence.

Define $\boldsymbol{g} := \nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\theta})$, and $\boldsymbol{\alpha} = \boldsymbol{g}/\|\boldsymbol{g}\|$. The second term becomes

$$\begin{aligned}
&\mathbb{E}\left[\max_{i\in[m]} \boldsymbol{g}^\top(\boldsymbol{x}_i - \boldsymbol{x}_1'')\right] \\
&= \|\boldsymbol{g}\| \, \mathbb{E}\left[\max_{i\in[m]} (\boldsymbol{g}/\|\boldsymbol{g}\|)^\top(\boldsymbol{x}_i - \boldsymbol{x}_1'')\right] \\
&= \|\boldsymbol{g}\| \, \mathbb{E}\left[\max_{i\in[m]} \boldsymbol{\alpha}^\top(\boldsymbol{x}_i - \boldsymbol{x}_1'')\right].
\end{aligned}$$

Define

$$\begin{aligned}
c_{\mathbb{P},m}^+ &= \sup_{\boldsymbol{\alpha}:\ \|\boldsymbol{\alpha}\|=1} \mathbb{E}\left[\max_{i\in[m]} \boldsymbol{\alpha}^\top(\boldsymbol{x}_i' - \boldsymbol{x}_1'')\right], \\
c_{\mathbb{P},m}^- &= \inf_{\boldsymbol{\alpha}:\ \|\boldsymbol{\alpha}\|=1} \mathbb{E}\left[\max_{i\in[m]} \boldsymbol{\alpha}^\top(\boldsymbol{x}_i' - \boldsymbol{x}_1'')\right].
\end{aligned}$$

This yields

$$c_{\mathbb{P},m}^- \|\boldsymbol{g}\| \le \mathbb{E}\left[\max_{i\in[m]} \boldsymbol{g}^\top(\boldsymbol{x}_i - \boldsymbol{x}_1'')\right] \le c_{\mathbb{P},m}^+ \|\boldsymbol{g}\|.$$

Obviously, we have $c_{\mathbb{P},m}^+ \ge c_{\mathbb{P},m}^- \ge c_{\mathbb{P},1}^- = 0$.

Consider $c_{\mathbb{P},m}^- = \inf_{\boldsymbol{\alpha}:\ \|\boldsymbol{\alpha}\|=1} \mathbb{E}\left[\max_{i\in[m]} \boldsymbol{\alpha}^\top(\boldsymbol{x}_i' - \boldsymbol{x}_1'')\right]$. Because $\{\boldsymbol{\alpha}:\ \|\boldsymbol{\alpha}\| = 1\}$ is compact and the objective is lower bounded by zero, there must be an $\boldsymbol{\alpha}_*$ such that $c_{\mathbb{P},m}^- = \mathbb{E}\left[\max_{i\in[m]} \boldsymbol{\alpha}_*^\top(\boldsymbol{x}_i' - \boldsymbol{x}_1'')\right]$. Following Lemma 1 below, we must have $c_{\mathbb{P},m}^- > 0$, because otherwise $\boldsymbol{\alpha}_*^\top \boldsymbol{x}_i'$ would be a deterministic random variable, which contradicts with the assumption.

Assume $\boldsymbol{x}_i$ is a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I})$ and $\|\cdot\|$ is the L2 norm. Define $\boldsymbol{z}_{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^\top(\boldsymbol{x}_i' - \boldsymbol{x}_1'')$, then $\boldsymbol{z}_{\boldsymbol{\alpha}}$ follows standard Gaussian distribution, independent of the value of $\boldsymbol{\alpha}$, and hence the inequality becomes equality.

In addition, from the results of [13, 17], when $\mathbb{P}(\cdot \mid \boldsymbol{x})$ is $\mathcal{N}(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I})$, we have $c_{\mathbb{P},m}^+ = c_{\mathbb{P},m}^- \asymp \sigma\sqrt{\log m}$. □

Lemma 1. *Assume Z, $Z_1, \ldots, Z_m$ are i.i.d. random variables, and $m \ge 2$. If $\mathbb{E}[\max_{i\in[m]} Z_i] = \mathbb{E}[Z]$, then Z equals some constant with probability one.*

Proof. Let $Y = \max_{i\in[m]} Z_i$. Then we have $Y \ge Z_1$ deterministically, and hence

$$\mathbb{E}[Y] \ge \mathbb{E}[Z_1] = \mathbb{E}[Z].$$

Therefore, if $\mathbb{E}[Y] = \mathbb{E}[Z]$, we must have $Y = Z_1$ w.r.t. probability one, which implies that $Z_1 = Z_2 = \cdots = Z_m$ with probability one. This suggests that $\mathrm{var}(Z) = 0 = \mathbb{E}[(Z_1 - Z_2)^2]/2 = 0$ and hence Z equals to a constant deterministically. □

3. Related Methods and Discussion

MaxUp is closely related to both data augmentation and adversarial training. It can be viewed as an *adversarial variant of data augmentation*, in that it minimizes the worst (or worse) case loss on the perturbed data, instead of an average loss like typical data augmentation methods. *MaxUp* can also be viewed as a *"lightweight" variant of adversarial training*, in that the maximum loss is calculated by simple random sampling, instead of more accurate gradient-based optimization for finding the adversarial loss, such as projected gradient descent (PGD).

MaxUp is much simpler and faster than the PGD-based adversarial training, and is more suitable for our purpose of alleviating over-fitting on clean data (instead of adversarial defense). Furthermore, PGD-based adversarial training is usually believed to do hurt to generalization. We now elaborate on these connections in depth.

3.1. Data Augmentation

Data augmentation has been widely used in machine learning, especially on image data which admits a rich set of transforms (e.g. translation, rotation, random cropping for image). Recent augmentation techniques, such as MixUp [33], CutMix [31] and manifold MixUp [26] have been found highly useful in training deep neural networks, especially in achieving state-of-the-art results on important image classification benchmarks such as SVHN, CIFAR and ImageNet. More recently, more advanced methods have been developed to find the optimal data augmentation poli-

cies using reinforcement learning or adversarial generative network [e.g. 2, 3, 5, 6, 35].

MaxUp can easily leverage these advanced data augmentation techniques to achieve good performance. The key difference, however, is that *MaxUp* in (2) minimizes the *maximum loss* on the augmented data, while typical data augmentation methods minimize the *average loss*, that is,

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_n}\left[\frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}_i', \boldsymbol{\theta})\right], \tag{7}$$

which we refer to as *standard data augmentation* throughout the paper. It turns out (2) and (7) behave very different as regularization mechanisms, in that (7) does not introduce the gradient-norm regularization as (2), and hence does not have the benefit of having gradient-norm regularization. This is because the first-order term in the Taylor expansion is canceled out due to the averaging in (7).

Specifically, let $\mathbb{P}(\cdot|\boldsymbol{x})$ be any distribution whose expectation is $\boldsymbol{x}$ and $L(\boldsymbol{x}, \boldsymbol{\theta})$ is second-order differentiable w.r.t $\boldsymbol{x}$. Define the expected loss related to (7) on data point $\boldsymbol{x}$:

$$\hat{L}_{\mathbb{P},m}(\boldsymbol{x}, \boldsymbol{\theta}) := \mathbb{E}_{\{\boldsymbol{x}_i'\}_{i=1}^m\sim\mathbb{P}(\cdot|\boldsymbol{x})^m}\left[\frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}_i', \boldsymbol{\theta})\right]. \tag{8}$$

Then with a simple Taylor expansion, we have

$$\hat{L}_{\mathbb{P},m}(\boldsymbol{x}, \boldsymbol{\theta}) = L(\boldsymbol{x}, \boldsymbol{\theta}) + \mathbf{O}(\sigma^2),$$

which misses the gradient-norm regularization term when compared with *MaxUp* decomposition in Theorem 1.

Note that the *MaxUp* update is computationally *faster* than the solving (7) with the same m, because we only need to backpropagate on the worst augmented copy for each data point (see Equation 3), while solving (7) requires to backpropagate on all the m copies at each iteration.

3.2. Adversarial Training

Adversarial training has been developed to defense various adversarial attacks on the data inputs [16]. It estimates $\boldsymbol{\theta}$ by solving the following problem:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_n}\left[\max_{\boldsymbol{x}'\in\mathcal{B}(\boldsymbol{x},r)} L(\boldsymbol{x}', \boldsymbol{\theta})\right], \tag{9}$$

where $\mathcal{B}(\boldsymbol{x}, r)$ represents a ball centered at $\boldsymbol{x}$ with radius r under some metrics (e.g. ℓ_0, ℓ_1, ℓ_2, or ℓ_∞ distances). The inner maximization is often solved by running projected gradient descent (PGD) for a number of iterations.

MaxUp in (2) can be roughly viewed as solving the inner adversarial maximization problem in (9) using a "mild", or "lightweight" optimizer by randomly drawing m points from $\mathbb{P}(\cdot|\boldsymbol{x})$ and finding the best. Such mild adversarial optimization increases the robustness against the random perturbation it introduces, and hence enhance the generalization performance. Adversarial ideas have also been used to improvement generalization in a series of recent works [e.g., 28, 36].

Different from our method, typical adversarial training methods, especially these based PGD [16], tend to solve the adversarial optimization much more *aggressively* to achieve higher robustness, but at the cost of scarifying the accuracy on clean data. There has been shown a clear trade-off between the accuracy of a classifier on clean data and its robustness against adversarial attacks [see e.g., 19, 25, 30, 34]. By using a mild adversarial optimizer, *MaxUp* strikes a better balance between the accuracy on clean data and adversarial robustness.

Besides, *MaxUp* is much more computationally efficient than PGD-based adversarial training, because it does not introduce additional back-propagation steps as PGD. In practice, *MaxUp* can be equipped with various complex data augmentation methods (in which case $\mathbb{P}(\cdot|\boldsymbol{x})$ can be discrete distributions), while PGD-based adversarial training mostly focuses on perturbations in ℓ_p balls.

3.3. Online Hard Example Mining

Online hard example mining (OHEM) [20] is a training method originally developed for region-based objective detection, which improves the performance of neural networks by picking the hardest examples within mini batches of stochastic gradient descent (SGD). It can be viewed as running SGD for minimizing the following expected loss

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathcal{M}}\left[\max_{\boldsymbol{x}\in\mathcal{M}} L(\boldsymbol{x}, \boldsymbol{\theta})\right],$$

which amounts to randomly picking a mini-batch $\mathcal{M}$ at each iteration and minimizing the loss of the hardest example within $\mathcal{M}$. By doing so, OHEM can focus more on the hard examples and hence improves the performance on borderline cases. This makes OHEM particularly useful for class-imbalance tasks, e.g. object detection [20], person re-identification [15].

Different with *MaxUp*, the hardest examples in OHEM are selected in mini-batches consisting of independently selected examples, with no special correlation or similarity. Mathematically, it can be viewed as reweighing the data distribution to emphasize harder instances. This is substantially different from *MaxUp*, which is designed to enforce the robustness against existing random data augmentation/perturbation schemes. Furthermore, standard PGD algorithm breaks the generalization and cannot improve the accuracy.

4. Experiments

We test our method using both image classification and 3D point cloud classification for which a variety of strong regularization techniques and data augmentation methods

Method	Top-1 error	Top-5 error
Vanilla [11]	76.3	-
Dropout [21]	76.8	93.4
DropPath [14]	77.1	93.5
Manifold Mixup [26]	77.5	93.8
AutoAugment [5]	77.6	93.8
Mixup [33]	77.9	93.9
DropBlock [9]	78.3	94.1
CutMix [31]	78.6	94.0
MaxUp+CutMix	**78.9**	**94.2**

Table 1. Summary of top1 and top5 accuracies on the validation set of ImageNet for ResNet-50.

have been proposed. We show that *MaxUp* can outperform all of these methods on the most challenging datasets (e.g. ImageNet, ModelNet40) and state-of-the-art models (e.g. ResNet, EfficientNet, DGCNN). We then apply our method to adversarial certification via Gaussian smoothing [4], for which we find that *MaxUp* can outperform both the augmented data baseline and PGD-based adversarial training baseline. Further, we accelerate *MaxUp* with low-resolution image for image classification tasks. The low-resolution approach, can also be applied to video, point cloud and other data format.

For all the tasks, if training from scratch, we first train the model with standard data augmentation with 5 epochs and then switch to *MaxUp*.

Time and Memory Cost *MaxUp* only slightly increase the time and memory cost compared with standard training. During *MaxUp*, we only need to find the worst instance out of the m augmented copies through forward-propagation, and then only back-propagate on the worst instance. Therefore, the additional cost of *MaxUp* over standard training is m forward-propagation, which introduces no significant overhead on both memory and time cost. Using low-resolution images during the first forward pass, we can further reduce the additional time cost introduced by *MaxUp*.

4.1. ImageNet

We evaluate *MaxUp* on ILSVRC2012, a subset of ImageNet classification dataset [7]. This dataset contains around 1.3 million training images and 50,000 validation images. We follow the standard data processing pipeline including scale and aspect ratio distortions, random crops, and horizontal flips in training. During the evaluation, we only use the single-crop setting.

Implementation Details We test *MaxUp* with $\mathbb{P}(\cdot|\boldsymbol{x})$ defined by the CutMix data augmentation technique [31] (referred to as *MaxUp*+CutMix). CutMix randomly cuts and pasts patches among training images, while the ground truth labels are also mixed proportionally to the area of the patches. *MaxUp*+CutMix applies CutMix on one image for m times (cutting different randomly sampled patches), and select the worst case to do backpropagation.

We test our method on ResNet-50, ResNet-101 [11], as well as recent energy-efficient architectures, including ProxylessNet [1] and EfficientNet [22]. We resize the images to 600×600 and 845×845 for EfficientNet-B7 and EfficientNet-B8, respectively [22], for which we process the images with the data processing pipelines proposed by [23]. For the other models, the input image size is 224×224. To save computation resources, we only fine-tune the pre-trained models with *MaxUp* for a few epochs. We set $m = 4$ for *MaxUp* in the ImageNet-2012 experiments unless indicated otherwise. This means that we optimize the worst case in 4 augmented samples for each image.

For ResNet-50, ResNet-101 and ProxylessNets, we train the models for 20 epochs with learning rate 10^{-5} and batch size 256 on 4 GPUs for 20 epochs. For EfficientNet, we fix the parameters in the batch normalization layers and train the other parameters with learning rate 10^{-4} and batch size 1000 for 5 epochs.

As shown in Table 2, for ResNet-50 and ResNet-101, we achieve the best results among all the data augmentation method. For EfficientNet-B8, we further improve the state-of-the-art result on ImageNet with no extra data.

ResNet-50 on ImageNet Table 1 compares a number of state-of-the-art regularization techniques with *MaxUp*+CutMix on ImageNet with ResNet-50.[1] We can see that *MaxUp*+CutMix achieves better performance compared to all the strong data augmentation and regularization baselines. From Table 1, we see that CutMix gives the best top1 error (78.6%) among all the augmentation tasks, but our method further improves it to 78.9%. DropBlock outperforms all the other methods in terms of the top5 error, but by augmenting CutMix with *MaxUp*, we improve the 94.1% top5 error rate obtained by DropbBlock to 94.2%.

More Results on Different Architectures Table 2 shows the result of ImageNet on ResNet-101, ProxylessNet-CPU/GPU/Mobile [1] and EfficientNet. We can see that *MaxUp* consistently improves the results in all these cases. On ResNet-101, it improves the 79.83% baseline to 80.26%. On ProxylessNet-CPU and ProxylessNet-GPU, *MaxUp* enhances the 75.32% and 75.08% top1 accuracy to 75.65% and 75.42%, respectively. On ProxylessNet-Mobile, we improve the 76.71% top1 accuracy to 77.17%.

For EfficientNet-B7, CutMix enhances the original top1 accuracy 85.0% [by 22] to 85.22%. *MaxUp* further im-

[1] All the FLOPS and model size reported in this paper is calculated by `https://pypi.org/project/ptflops`.

Model	Model Size	FLOPs	+CutMix (%)	+*MaxUp*+CutMix (%)
ResNet-101	44.55M	7.85G	79.83	**80.26**
ProxylessNet-CPU	7.12M	481M	75.32	**75.65**
ProxylessNet-GPU	4.36M	470M	75.08	**75.42**
ProxylessNet-Mobile $\times 1.4$	6.86M	603M	76.71	**77.17**
EfficientNet-B7	66.35M	38.20G	85.22*	**85.45***
Fix-EfficientNet-B8	87.42M	101.79G	85.57*	**85.80***

Table 2. Top1 accuracies of different models on the validation set of ImageNet 2012. The "$*$" indicates that *MaxUp* is applied to the pre-trained model and trained for 5 epochs.

proves the top1 accuracy to 88.45%. On Fix-EfficientNet-B8, *MaxUp* obtains the state-of-the-art 85.80% top1 accuracy. The previous state-of-the-art top1 accuracy, 85.50%, is achieved by EfficientNet-L2.

4.2. CIFAR-10 and CIFAR-100

We test *MaxUp* equipped with Cutout [8] on CIFAR-10 and CIFAR-100, and denote it by *MaxUp*+Cutout. We conduct our method on several neural architectures, including ResNet-110 [11], PreAct-ResNet-110 [12] and WideResNet-28-10 [32]. We set $m = 10$ for WideResNet and $m = 4$ for the other models. We use the public code[2] and keep their hyper-parameters.

Implementation Details For CIFAR-10 and CIFAR-100, we use the standard data processing pipeline (mirror+ crop) and train the model with 200 epochs. All the results reported in this section are averaged over five runs.

We train the models for 200 epochs on the training set with 256 examples per mini-batch, and evaluate the trained models on the test set. The learning rate starts at 0.1 and is divided by 10 after 100 and 150 epochs for ResNet-110 and PreAct-ResNet-110. For WideResNet-28-10, we follow the settings in the original paper [32], where the learning rate is divided by 10 after 60, 120 and 180 epochs. Weight decay is set to 2.5×10^{-4} for all the models, and we do not use dropout.

Model	+ Cutout	+ *MaxUp*+Cutout
ResNet-110	94.84 ± 0.11	$\mathbf{95.41 \pm 0.08}$
PreAct-ResNet-110	95.02 ± 0.15	$\mathbf{95.52 \pm 0.06}$
WideResNet-28-10	96.92 ± 0.16	$\mathbf{97.18 \pm 0.06}$

Table 3. Test accuracy on CIFAR10 for different architectures.

Results The results on CIFAR-10 and CIFAR-100 are summarized in Table 3 and Table 4. We can see that the models trained using *MaxUp*+Cutout significantly outperform the standard Cutout for all the cases.

[2] https://github.com/junyuseu/pytorch-cifar-models

Model	+ Cutout	+ *MaxUp*+Cutout
ResNet-110	73.64 ± 0.15	$\mathbf{75.26 \pm 0.21}$
PreAct-ResNet-110	74.37 ± 0.13	$\mathbf{75.63 \pm 0.26}$
WideResNet-28-10	81.59 ± 0.27	$\mathbf{82.48 \pm 0.23}$

Table 4. Test accuracy on CIFAR100 for different architectures.

m	ResNet-110	WideResNet-28-10
1	73.64 ± 0.15	81.59 ± 0.27
4	75.26 ± 0.21	81.82 ± 0.22
10	75.19 ± 0.13	$\mathbf{82.48 \pm 0.23}$
20	74.37 ± 0.18	82.43 ± 0.24

Table 5. Test accuracy on CIFAR100 with ResNet-110 and WideResNet-28-10, when the sample size m varies.

On CIAFR-10, *MaxUp* improves the standard Cutout baseline from $94.84\% \pm 0.11\%$ to $95.41\% \pm 0.08\%$ on ResNet-110. It also improves the accuracy from $95.02\% \pm 0.15\%$ to $95.52\% \pm 0.06\%$ on PreAct-ResNet-110.

On CIFAR-100, *MaxUp* obtains improvements by a large margin. On ResNet-110 and PreAct-ResNet-110, *MaxUp* improves the performance of Cutout from $73.64\% \pm 0.15\%$ and $74.37\% \pm 0.13\%$ to $75.26\% \pm 0.21\%$ and $75.63\% \pm 0.26\%$, respectively. *MaxUp*+Cutout also improves the standard Cutout from $81.59\% \pm 0.27\%$ to $82.48\% \pm 0.23\%$ on WideResNet-28-10 on CIFAR-100.

Ablation Study We test *MaxUp* with different sample size m and investigate its impact on the performance on ResNet-100 (a relatively small model) and WideResNet-28-10 (a larger model).

Table 5 shows the result when we vary the sample size in $m \in \{1, 4, 10, 20\}$. Note that *MaxUp* reduces to the naïve data augmentation method when $m = 1$. As shown in Table 5, *MaxUp* with all $m > 1$ can improve the result of standard augmentation ($m = 1$). Setting $m = 4$ or $m = 10$ achieves best performance on ResNet-110 , and $m = 10$ obtains best performance on WideResNet-28-10. We can see that the results are not sensitive once m is in a proper range (e.g., $m \in [4 : 10]$), and it is easy to outperform the standard data augmentation ($m = 1$) without much tuning of m. Furthermore, for models with more parameters, we

ℓ_2 RADIUS (CIFAR-10)	0.25	0.5	0.75	1.0	1.25	1.5	1.75	2.0	2.25	2.5	2.75
[4] (%)	60	43	34	23	17	14	12	10	8	6	4
[18] (%)	**74**	**57**	48	38	33	29	25	19	17	14	12
Ours (%)	**74**	**57**	**49**	**40**	**35**	**31**	**27**	**22**	**19**	**17**	**15**

Table 6. Certified accuracy on CIFAR-10 of the best classifiers by different methods, evaluated against ℓ_2 attacks of different radiuses.

can use stronger regularization (larger m).

4.3. Adversarial Certification

Modern image classifiers are known to be sensitive to small, adversarially-chosen perturbations on inputs [10]. Therefore, for making high-stakes decisions, it is of critical importance to develop methods with *certified robustness*, which provide (high probability) provable guarantees on the correctness of the prediction subject to arbitrary attacks within certain perturbation ball.

Recently, [4] proposed to construct certifiably robust classifiers against ℓ_2 attacks by introducing Gaussian smoothing on the inputs, which is shown to outperform all the previous ℓ_2-robust classifiers in CIFAR-10. There has been two major methods for training such smoothed classifiers: [4] trains the classifier with a Gaussian data augmentation technique, while [18] improves the original Gaussian data augmentation by using PGD (projected gradient descent) adversarial training, in which PGD is used to find a local maximal within a given ℓ_2 perturbation ball.

In our experiment, we use *MaxUp* with Gaussian perturbation (referred to as *MaxUp*+Gauss) to train better smoothed classifiers than the methods by [4] and [18]. Like how *MaxUp* improves upon standard data augmentation, it is natural to expect that our *MaxUp*+Gauss can learn more robust classifiers than the standard Gaussian data augmentation method in [4].

Training Details We applied *MaxUp* to Gaussian augmented data on CIFAR-10 with ResNet-110 [11]. We follow the training pipelines described in [18]. We set a batch size of 256, an initial learning rate of 0.1 which drops by a factor of 10 every 50 epochs, and train the models for 150 epochs.

Evaluation After training the smoothed classifiers, we evaluation the certified accuracy of different models under different ℓ_2 perturbation sets. Given an input image $\boldsymbol{x}$ and a perturbation region $\mathcal{B}$, the smoothed classifier is called certifiably correct if its prediction is correct and has a guaranteed lower bound larger than 0.5 in $\mathcal{B}$. The certified accuracy is the percentage of images that are certifiably correct. Following [18], we calculate the certified accuracy of all the classifiers for various radius and report the best results overall of the classifiers. We use the codes provided by [4] to calculate certified accuracy.[3]

Following [18], we select the best hyperparameters with grid search. The only two hyperparameters of our *MaxUp*+Gauss are the sample size m and the variance σ^2 of the Gaussian perturbation, which we search in $m \in \{5, 25, 50, 100, 150\}$ and $\sigma \in \{0.12, 0.25, 0.5, 1.0\}$. In comparison, [18] requires to search a larger number of hyper-parameters, including the number of steps of the PGD, the number of noise samples, the maximum ℓ_2 perturbation, and the variance of Gaussian data augmentation during training and testing. Overall, [18] requires to train and evaluate over 150 models for hyperparmeter tuning, while *MaxUp*+Gauss requires only 20 models.

Results We show the certified accuraries on CIFAR-10 in Table 6 under ℓ_2 attacks for each ℓ_2 radius. We find that *MaxUp* outperforms [4] for all the ℓ_2 radiuses by a large margin. For example, *MaxUp* can improve the certified accuracy at radius 0.25 from 60% to 74% and improve the 4% accuracy on radius 2.75 to 15%. *MaxUp* also outperforms the PGD-based adversarial training of [18] for all the radiuses, boosting the accuracy from 14% to 17% at radius 2.5, and from 12% to 15% at radius 2.75.

In summary, *MaxUp* clearly outperforms both [4] and [18]. *MaxUp* is also much faster and requires less hyperparameter tuning than [18]. Although the PGD-based method of [18] was designed to outperform the original method by [4], *MaxUp*+Gauss further improves upon [18], likely because *MaxUp* with Gaussian perturbation is more compatible with the Gaussian smoothing based certification of [4] than PGD adversarial optimization.

4.4. Point Cloud Classification

Implementation To further verify the performance of our algorithm, we conduct experiments on 3D point cloud classification. We test all the baselines and proposed algorithm on one popular dataset, ModelNet40 [27] using the state-of-the-art model architecture Dynamic graph convolutional neural network (DGCNN) [29]. In existing points processing networks, data augmentation mainly include random rotation about the gravity axis, random scaling, and random jittering [29].

We choose different hyper-parameters (e.g. number of particles, number of neighbours) and different data augmen-

[3] https://github.com/locuslab/smoothing

Model	Dataset	Augmentation	Standard		+*MaxUp*		+*MaxUp* (Low R)	
			Acc(%)	Time	Acc(%)	Time	Acc(%)	Time
ResNet-110	CIFAR-10	Cutout	94.8±0.1	36s	**95.4±0.1**	58s	**95.4±0.1**	47s
ResNet-110	CIFAR-100	Cutout	73.6±0.2	36s	**75.3±0.2**	58s	75.0±0.1	48s
WideResNet-28-10	CIFAR-10	RandAugment	97.1±0.1	72s	**97.5±0.1**	106s	**97.5±0.1**	89s
WideResNet-28-10	CIFAR-100	RandAugment	83.1±0.1	72s	**83.8±0.1**	106s	**83.7±0.1**	89s
ResNet-50	ImageNet	CutMix	78.6±0.0	3.6h	**78.9±0.0**	5.2h	**78.9±0.0**	4.4h

Table 7. Top1 accuracies of different models on the validation set of ImageNet 2012 or on test set of CIFAR-100/10. 'Low R' denotes low resolution, 'Standard' denotes the standard data augmentation method, 'Acc' denotes accuracy and 'Time' denotes per epoch training time estimated on one NVIDIA TITAN V VOLTA.

#Part	#N	Type	Standard (%)	*MaxUp* (%)
1024	20	Gaussian	86.8±0.6	**88.9±0.4**
2048	20	Gaussian	88.4±0.5	**90.4±0.5**
2048	40	Gaussian	92.0±0.5	**92.3±0.4**
1024	20	Jitter	88.1±0.6	**89.2±0.5**
2048	20	Jitter	89.1±0.5	**90.4±0.4**
2048	40	Jitter	92.2±0.5	**92.4±0.5**

Table 8. The results on ModelNet40. '#Part' denotes the number of particles, '#N' denotes the number of neighbours, and 'Type' denotes the type of data augmentation.

tation methods (e.g. Gaussian noise, random jittering) to verify the performance under different settings. For Gaussian blur, we use $\mathcal{N}(0, 0.05)$.

Results Table 8 shows that using *MaxUp* can consistently outperform the standard data augmentation results with different data augmentation and hyper-parameters on ModelNet40. When the number of particles is 1024 and the number of neighbours of the KNN (K-Nearest Neighbor) is 20, *MaxUp* can improve the standard data augmentation by a large margin. When using Gaussian noise, it improve 86.8%±0.6% accuracy to 88.9%±0.4%. When using radfom jittering, it improve 88.1%±0.6% accuracy to 89.2%±0.5%.

4.5. Accelerating Training for Image

A remained problem for *MaxUp* is that, although it only requires extra forward pass, *MaxUp* still includes non-negligible additional time cost. Here, we propose a heuristic approach to address this problem in practice. For image-related tasks, we use low-resolution images when selecting the worst case among sampled augmented images. This strategy can also be applied to video, 3D images and point cloud data.

Implementation We directly followed all the settings in Section 4.3 and 4.2. The only difference is that we use low-resolution down-sampled image to select the worst case. We set the resolution to 16 and 96 for CIFAR and ImageNet, respectively.

Results As shown in Table 7, by using low-resolution image, we include almost no extra training cost compared to standard data augmentation method. Besides, compared to standard *MaxUp*, the low-resolution solution can achieve almost the same performance on the tested cases. For example, when training ResNet-50 on ImageNet, using low-resolution image to select hard augmented examples achieves the same accuracy as *MaxUp* while acquiring less training time. For this case, it reduces the 5.2 hours per epoch to 4.4 hours per epoch.

These results indicates that, for image, the proposed *MaxUp* can be accelerated with using low-resolution images during selection. For other kinds of data type, e.g. point cloud, video, we can also use sub-sampled particles, sub-sampled spatial-temporal pairs.

5. Conclusion

In this paper, we propose *MaxUp*, a simple and efficient training algorithms for improving generalization, especially for deep neural networks. *MaxUp* can be viewed as a introducing a gradient-norm smoothness regularization for Gaussian perturbation, but does not require to evaluate the gradient norm explicitly, and can be easily combined with any existing data augmentation methods. We empirically show that *MaxUp* can improve the performance of data augmentation methods in image classification, language modeling, and certified defense. Especially, we achieve the SOTA performance with extra data on ImageNet when this work is done.

Acknowledgement

This paper is supported in part by NSF CAREER 1846421, SenSE 2037267 and EAGER 2041327. We would like to thank the reviewers for their thoughtful comments and efforts towards improving our manuscript.

References

[1] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2019.

[2] Tianlong Chen, Yu Cheng, Zhe Gan, Jianfeng Wang, Lijuan Wang, Zhangyang Wang, and Jingjing Liu. Adversarial feature augmentation and normalization for visual recognition. *arXiv preprint arXiv:2103.12171*, 2021.

[3] Tianlong Chen, Sijia Liu, Shiyu Chang, Yu Cheng, Lisa Amini, and Zhangyang Wang. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 699–708, 2020.

[4] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *ICML*, 2019.

[5] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *CVPR*, 2019.

[6] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719*, 2019.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE, 2009.

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

[9] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *NeurIPS*, pages 10727–10737, 2018.

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2014.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645. Springer, 2016.

[13] Gautam Kamath. Bounds on the expectation of the maximum of samples from a gaussian. *URL http://www. gautamkamath. com/writings/gaussian max. pdf*, 2015.

[14] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *ICLR*, 2017.

[15] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. Bag of tricks and a strong baseline for deep person re-identification. In *CVPRW*, pages 0–0, 2019.

[16] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ICLR*, 2017.

[17] Francesco Orabona and Dávid Pál. Optimal non-asymptotic lower bound on the minimax regret of learning with expert advice. *arXiv preprint arXiv:1511.02176*, 2015.

[18] Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sebastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. *NeurIPS*, 2019.

[19] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *NeurIPS*, pages 5014–5026, 2018.

[20] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, pages 761–769, 2016.

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, pages 1929–1958, 2014.

[22] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *ICML*, 2019.

[23] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. *arXiv preprint arXiv:1906.06423*, 2019.

[24] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *ICLR*, 2018.

[25] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *ICLR*, 2019.

[26] Vikas Verma, Alex Lamb, Christopher Beckham, Aaron Courville, Ioannis Mitliagkis, and Yoshua Bengio. Manifold mixup: Encouraging meaningful on-manifold interpolation as a regularizer. *ICML*, 2019.

[27] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[28] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan Yuille, and Quoc V Le. Adversarial examples improve image recognition. *arXiv preprint arXiv:1911.09665*, 2019.

[29] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *arXiv preprint*

arXiv:1801.07455, 2018.

[30] Dong Yin, Ramchandran Kannan, and Peter Bartlett. Rademacher complexity for adversarially robust generalization. In *ICML*, pages 7085–7094, 2019.

[31] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *ICCV*, 2019.

[32] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, pages 87.1–87.12. BMVA Press, September 2016.

[33] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[34] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, pages 7472–7482, 2019.

[35] Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. Adversarial autoaugment. *ICLR*, 2020.

[36] Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Thomas Goldstein, and Jingjing Liu. Freelb: Enhanced adversarial training for language understanding. *ICLR*, 2020.