MaxUp: A Simple Way to Improve Generalization of Neural Network Training

Chengyue Gong* [1] Tongzheng Ren* [1] Mao Ye [1] Qiang Liu [1]

Abstract

We propose *MaxUp*, an embarrassingly simple, highly 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, language modeling, and adversarial certification, on which *MaxUp* consistently outperforms the existing best baseline methods, without introducing substantial computational overhead. In particular, we improve ImageNet classification from the state-of-the-art top-1 accuracy 85.5% without extra data to 85.8%. Code will be released soon.

1. Introduction

A central theme of machine learning is to alleviate the issue of overfitting, improving 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 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. Zhang et al., 2018; DeVries & Taylor, 2017; Cubuk et al., 2019a), 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 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 pertubrations (e.g. Tramèr et al., 2018; Madry et al., 2017), 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 4.4)). In addition, compared with standard adversarial training methods such as projected gradient descent (PGD) (Madry et al., 2017), *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 three challenging tasks: image classification, language modeling, and certified defense against adversarial examples (Cohen et al., 2019). We find that *MaxUp* can leverage the 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 a new state-of-the-art result on ImageNet classification without extra data, which improves the best 85.5% top1 accuracy by Xie et al. (2019) to 85.8%. For the adversarial certification task, we find

*Equal contribution [1]UT Austin. Correspondence to: Chengyue Gong <cygong@cs.utexas.edu>.

Maxup allows us to train more verifiably robust classifiers than prior arts such as the PGD-based adversarial training proposed by Salman et al. (2019).

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 $\mathcal{D}_n = \{\boldsymbol{x}_i\}_{i=1}^n$, 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 of interest (e.g., the weights of a neural network), and $L(\boldsymbol{x},\boldsymbol{\theta})$ denotes the loss associated with data point $\boldsymbol{x}$. A key issue of ERM is the risk of overfitting, especially when the data information is insufficient.

MaxUp We propose *MaxUp* to alleviate overfitting. 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 invariant data transformations that are widely used in the data augmentation literature, such as random crops, equalizing, rotations, and clips for images (see e.g Cubuk et al., 2019a; DeVries & Taylor, 2017; Cubuk et al., 2019b).

2.1. *MaxUp* as a Smoothness Regularization

We provide a theoretical interpretation of *Maxup* as introducing a *gradient-norm regularization* to the original ERM objective to encourage smoothness. Here we consider the simple case of isotropic Gaussian perturbation, when $\mathbb{P}(\cdot|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x},\sigma^2\boldsymbol{I})$. For simplifying notation, we define

$$\tilde{L}_{\mathbb{P},m}(\boldsymbol{x},\boldsymbol{\theta}) := \mathbb{E}_{\{\boldsymbol{x}'_i\}_{i=1}^m\sim\mathbb{P}(\cdot|\boldsymbol{x})^m}\left[\max_{i\in[m]} L(\boldsymbol{x}'_i,\boldsymbol{\theta})\right], \tag{4}$$

which represents the expected *MaxUp* risk of data point $\boldsymbol{x}$ with m augmented copies.

Theorem 1 (MaxUp as Gradient-Norm Regularization)**.** *Consider $\tilde{L}_{\mathbb{P},m}(\boldsymbol{x},\boldsymbol{\theta})$ defined in* (4) *with $\mathbb{P}(\cdot|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x},\sigma^2\boldsymbol{I})$. Assume $L(\boldsymbol{x},\boldsymbol{\theta})$ is second-order differentiable w.r.t. $\boldsymbol{x}$. Then*

$$\tilde{L}_{\mathbb{P},m}(\boldsymbol{x},\boldsymbol{\theta}) = L(\boldsymbol{x},\boldsymbol{\theta}) + c_{m,\sigma}\left\|\nabla_{\boldsymbol{x}} L(\boldsymbol{x},\boldsymbol{\theta})\right\|_2 + \mathbf{O}(\sigma^2),$$

where $c_{m,\sigma}$ is a constant and $c_{m,\sigma} = \mathbf{\Theta}(\sigma\sqrt{\log m})$, where $\mathbf{\Theta}(\cdot)$ denotes the big-Theta notation.

Theorem 1 shows that, the expected *MaxUp* risk can be viewed as introducing a Lipschitz-like regularization with the gradient norm $\|\nabla_x L(\boldsymbol{x},\boldsymbol{\theta})\|_2$, 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 $c_{m,\sigma}$, which depends on the number of samples m and perturbation magnitude σ.

Proof. Using Taylor expansion, we have

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

where we assume $\boldsymbol{z}_i = \boldsymbol{x}'_i - \boldsymbol{x}$, which follows $\mathcal{N}(0,\sigma^2\boldsymbol{I})$. The rest of the proof is due to the Lemma 1 below. □

Lemma 1. *Let $\boldsymbol{g}$ be a fixed vector in $\mathbb{R}^d$, and $\{\boldsymbol{z}_i\}_{i=1}^m$ are m i.i.d. random variables from $\mathcal{N}(0,\sigma^2\boldsymbol{I})$. We have*

$$\mathbb{E}\left[\max_{i\in[m]}\langle\boldsymbol{g},\boldsymbol{z}_i\rangle\right] = c_{m,\sigma}\|\boldsymbol{g}\|_2,$$

where $c_{m,\sigma} = \mathbf{\Theta}\left(\sigma\sqrt{\log m}\right)$.

Proof. Define $y_i = \langle\boldsymbol{g},\boldsymbol{z}_i\rangle / \|\boldsymbol{g}\|_2$. Then $\{y_i\}_{i=1}^m$ is *i.i.d.* from $\mathcal{N}(0,\sigma^2)$. Therefore, $c_{m,\sigma} = \mathbb{E}[\max_{i\in[m]} y_i]$, which is well known to be $\mathbf{\Theta}(\sigma\sqrt{\log m})$. See e.g., Orabona & Pál (2015); Kamath (2015) for bounds related to $\mathbb{E}[\max_{i\in[m]} y_i]$. More specifically, we have $0.23\sigma\sqrt{\log m} \leq c_{m,\sigma} \leq \sqrt{2}\sigma\sqrt{\log m}$ following Kamath (2015). □

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 $\eta$, 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
```

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 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 optimizers 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). 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 invariance transforms (e.g. translation, rotation, random cropping). Recent augmentation techniques, such as MixUp (Zhang et al., 2018), CutMix (Yun et al., 2019) and manifold MixUp (Verma et al., 2019) 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 policies using reinforcement learning or adversarial generative network (e.g. Cubuk et al., 2019a;b; Zhang et al., 2020).

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,

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_n} \left[\frac{1}{m} \sum_{i=1}^m L(\boldsymbol{x}_i', \boldsymbol{\theta}) \right], \quad (5)$$

which we refer to as *standard data augmentation* throughout the paper. It turns out (2) and (5) behave very different as regularization mechanisms, in that (5) 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 (5).

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 (5) on data point $\boldsymbol{x}$:

$$\hat{L}_{\mathbb{P},m}(\boldsymbol{x}, \boldsymbol{\theta}) := \mathbb{E}_{\{\boldsymbol{x}_i'\}_{i=1}^m \sim \mathbb{P}(\cdot|\boldsymbol{x})^m} \left[\frac{1}{m} \sum_{i=1}^m L(\boldsymbol{x}_i', \boldsymbol{\theta}) \right]. \quad (6)$$

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 (5) with the same m, because we only need to backpropagate on the worst augmented copy for each data point (see Equation 3), while solving (5) 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 (Madry et al., 2017). It estimates $\boldsymbol{\theta}$ by solving the following problem:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_n} \left[\max_{\boldsymbol{x}'\in\mathcal{B}(\boldsymbol{x},r)} L(\boldsymbol{x}', \boldsymbol{\theta}) \right], \quad (7)$$

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 (7) 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., Xie et al., 2019; Zhu et al., 2020).

Different from our method, typical adversarial training methods, especially these based PGD (Madry et al., 2017), 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., Tsipras et al., 2019; Zhang et al., 2019; Yin et al., 2019; Schmidt et al., 2018). 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) (Shrivastava et al., 2016) 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 (Shrivastava et al., 2016), person re-identification (Luo et al., 2019).

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.

Method	Top-1 error	Top-5 error
Vanilla (He et al., 2016b)	76.3	-
Dropout (Srivastava et al., 2014)	76.8	93.4
DropPath (Larsson et al., 2017)	77.1	93.5
Manifold Mixup (Verma et al., 2019)	77.5	93.8
AutoAugment (Cubuk et al., 2019a)	77.6	93.8
Mixup (Zhang et al., 2018)	77.9	93.9
DropBlock (Ghiasi et al., 2018)	78.3	94.1
CutMix (Yun et al., 2019)	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.

4. Experiments

We test our method using both image classification and language modeling for which a variety of strong regularization techniques and data augmentation methods have been proposed. We show that *MaxUp* can outperform all of these methods on the most challenging datasets (e.g. ImageNet, Penn Treebank, and Wikitext-2) and state-of-the-art models (e.g. ResNet, EfficientNet, AWD-LSTM). In addition, we apply our method to adversarial certification via Gaussian smoothing (Cohen et al., 2019), for which we find that *MaxUp* can outperform both the augmented data baseline and PGD-based adversarial training baseline.

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.

4.1. ImageNet

We evaluate *MaxUp* on ILSVRC2012, a subset of ImageNet classification dataset (Deng et al., 2009). 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 (Yun et al., 2019) (referred to as *MaxUp*+CutMix).

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.

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 (He et al., 2016b), as well as recent energy-efficient architectures, including ProxylessNet (Cai et al., 2019) and EfficientNet (Tan & Le, 2019). We resize the images to 600×600 and 845×845 for EfficientNet-B7 and EfficientNet-B8, respectively (Tan & Le, 2019), for which we process the images with the data processing pipelines proposed by Touvron et al. (2019). 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 (Cai et al., 2019) 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 Tan & Le, 2019) to 85.22%. *MaxUp* further improves 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 (DeVries & Taylor, 2017) on CIFAR-10 and CIFAR-100, and denote it by *MaxUp*+Cutout. We conduct our method on several neural architectures, including ResNet-110 (He et al., 2016b), PreAct-ResNet-110 (He et al., 2016a) and WideResNet-28-10 (Zagoruyko & Komodakis, 2016). 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

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

[2] The code is downloaded from `https://github.com/junyuseu/pytorch-cifar-models`

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.

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.

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 (Zagoruyko & Komodakis, 2016), where the learning rate is divided by 10 after 60, 120 and 180 epochs. Weight decay is set to 2.5^{-4} for all the models, and we do not use dropout.

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.

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, we suggest to use a large m for large models, and a small m for relatively small models.

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.

4.3. Language Modeling

For language modeling, we test *MaxUp* on two benchmark datasets: Penn Treebank (PTB) and Wikitext-2 (WT2). We use the code provided by Wang et al. (2019) as our baseline[3], which stacks a three-layer LSTM and implements a bag of regularization and optimization tricks for neural language modeling proposed by Merity et al. (2018), such as weight tying, word embedding drop and Averaged SGD.

For this task, we apply *MaxUp* using word embedding dropout (Merity et al., 2018) as the random data augmentation method. Word embedding dropout implements dropout on the embedding matrix at the word level, where the dropout is broadcasted across all the embeddings of all the word vectors. For the selected words, their embedding vectors are set to be zero vectors. The other word embeddings in the vocabulary are scaled by $\frac{1}{1-p}$, where p is the probability of embedding dropout.

As the word embedding layer serves as the first layer in a neural language model, we apply *MaxUp* in this layer. We do feed-forward for m times and select the worst case to do backpropagation for each given sentence. In this section, we set a small $m = 2$ since the models are already well-regularized by other regularization techniques.

Implement Details The PTB corpus (Marcus et al., 1993) is a standard dataset for benchmarking language models. It consists of 923k training, 73k validation and 82k test words. We use the processed version provided by Mikolov et al. (2010) that is widely used for PTB.

The WT2 dataset is introduced in Merity et al. (2018) as an alternative to PTB. It contains pre-processed Wikipedia articles, and the training set contains 2 million words.

The training procedure can be decoupled into two stages: 1) optimizing the model with SGD and averaged SGD (ASGD); 2) restarting ASGD for fine-tuning twice. We apply *MaxUp* in both stages, and report the perplexity scores at the end of the second stage. We also report the perplexity scores with a recently-proposed post-process method, dy-

[3] https://github.com/ChengyueGongR/advsoft

Method	Params	Valid	Test
NAS-RNN (Zoph & Le, 2017)	54M	-	62.40
AWD-LSTM (Merity et al., 2018)	24M	58.50	56.50
AWD-LSTM + FRAGE (Gong et al., 2018)	24M	58.10	56.10
AWD-LSTM + MoS (Yang et al., 2018)	22M	56.54	54.44
w/o dynamic evaluation			
ADV-AWD-LSTM (Wang et al., 2019)	24M	57.15	55.01
ADV-AWD-LSTM + *MaxUp*	24M	**56.25**	**54.27**
+ dynamic evaluation (Krause et al., 2018)			
ADV-AWD-LSTM (Wang et al., 2019)	24M	51.60	51.10
ADV-AWD-LSTM + *MaxUp*	24M	**50.83**	**50.29**

Table 6. Perplexities on the validation and test sets on the Penn Treebank dataset. Smaller perplexities refer to better language modeling performance. `Params` denotes the number of model parameters.

Method	Params	Valid	Test
AWD-LSTM (Merity et al., 2018)	33M	68.60	65.80
AWD-LSTM + FRAGE (Gong et al., 2018)	33M	66.50	63.40
AWD-LSTM + MoS (Yang et al., 2018)	35M	63.88	61.45
w/o dynamic evaluation			
ADV-AWD-LSTM (Wang et al., 2019)	33M	63.68	61.34
ADV-AWD-LSTM + *MaxUp*	33M	**62.48**	**60.19**
+ dynamic evaluation (Krause et al., 2018)			
ADV-AWD-LSTM (Wang et al., 2019)	33M	42.36	40.53
ADV-AWD-LSTM + *MaxUp*	33M	**41.29**	**39.61**

Table 7. Perplexities on the validation and test sets on the WikiText-2 dataset. Smaller perplexities refer to better language modeling performance. `Params` denotes the number of model parameters.

namical evaluation (Krause et al., 2018) after the training process.

Results on PTB and WT2 The results on PTB and WT2 corpus are illustrated in Table 6 and Table 7, respectively. We calculate the perplexity on the validation and test set for each method to evaluate its performance. We can see that *MaxUp* outperforms the state-of-the-art results achieved by Frage (Gong et al., 2018) and Mixture of SoftMax (Yang et al., 2018). We further compare *MaxUp* to the result of Wang et al. (2019) based on AWD-LSTM (Merity et al., 2018) at two checkpoints, with or without dynamic evaluation (Krause et al., 2018). On PTB, we enhance the baseline from 55.01/51.10 to 54.27/50.29 at these two checkpoints on the test set. On WT2, we enhance the baseline from 61.34/40.53 to 60.19/39.61 at these two checkpoints on the test set. Results on validation set are reported in both Table 6 and 7 to show that the improvement can not achieved by simple hyper-parameter tuning on the test set.

4.4. Adversarial Certification

Modern image classifiers are known to be sensitive to small, adversarially-chosen perturbations on inputs (Goodfellow et al., 2014). 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, Cohen et al. (2019) 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: Cohen et al. (2019) trains the classifier with a Gaussian data augmentation technique, while Salman et al. (2019) 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

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
Cohen et al. (2019) (%)	60	43	34	23	17	14	12	10	8	6	4
Salman et al. (2019) (%)	**74**	**57**	48	38	33	29	25	19	17	14	12
Ours (%)	**74**	**57**	**49**	**40**	**35**	**31**	**27**	**22**	**19**	**17**	**15**

Table 8. Certified accuracy on CIFAR-10 of the best classifiers by different methods, evaluated against ℓ_2 attacks of different radiuses.

smoothed classifiers than the methods by Cohen et al. (2019) and Salman et al. (2019). 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 Cohen et al. (2019).

Training Details We applied *MaxUp* to Gaussian augmented data on CIFAR-10 with ResNet-110 (He et al., 2016b). We follow the training pipelines described in Salman et al. (2019). 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 Salman et al. (2019), 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 Cohen et al. (2019) to calculate certified accuracy.[4]

Following Salman et al. (2019), 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, Salman et al. (2019) requiers 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, Salman et al. (2019) 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 8 under ℓ_2 attacks for each ℓ_2 radius. We find that *MaxUp* outperforms Cohen et al. (2019) 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 Salman et al. (2019) 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 Cohen et al. (2019) and Salman et al. (2019). *MaxUp* is also much faster and requires less hyperparameter tuning than Salman et al. (2019). Although the PGD-based method of Salman et al. (2019) was designed to outperform the original method by Cohen et al. (2019), *MaxUp*+Gauss further improves upon Salman et al. (2019), likely because *MaxUp* with Gaussian perturbation is more compatible with the Gaussian smoothing based certification of Cohen et al. (2019) than PGD adversarial optimization.

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 SOTA performance on ImageNet.

For future works, we will apply *MaxUp* to more applications and models, such as BERT (Devlin et al., 2019). Furthermore, we will generalize *MaxUp* to apply mild adversarial optimization on feature and label spaces for other challenging tasks in machine learning, including transfer learning, semi-supervised learning.

References

Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2019.

Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. *ICML*, 2019.

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

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. *CVPR*, 2019a.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719*, 2019b.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255. IEEE, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. 2019.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Ghiasi, G., Lin, T.-Y., and Le, Q. V. Dropblock: A regularization method for convolutional networks. In *NeurIPS*, pp. 10727–10737, 2018.

Gong, C., He, D., Tan, X., Qin, T., Wang, L., and Liu, T.-Y. Frage: Frequency-agnostic word representation. In *NeurIPS*, pp. 1334–1345, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *ICLR*, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *ECCV*, pp. 630–645. Springer, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016b.

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

Krause, B., Kahembwe, E., Murray, I., and Renals, S. Dynamic evaluation of neural sequence models. *ICML*, 2018.

Larsson, G., Maire, M., and Shakhnarovich, G. Fractalnet: Ultra-deep neural networks without residuals. *ICLR*, 2017.

Luo, H., Gu, Y., Liao, X., Lai, S., and Jiang, W. Bag of tricks and a strong baseline for deep person reidentification. In *CVPRW*, pp. 0–0, 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *ICLR*, 2017.

Marcus, M., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of english: The penn treebank. 1993.

Merity, S., Keskar, N. S., and Socher, R. Regularizing and optimizing lstm language models. *ICLR*, 2018.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. Recurrent neural network based language model. In *ISCA*, 2010.

Orabona, F. and Pál, D. Optimal non-asymptotic lower bound on the minimax regret of learning with expert advice. *arXiv preprint arXiv:1511.02176*, 2015.

Salman, H., Yang, G., Li, J., Zhang, P., Zhang, H., Razenshteyn, I., and Bubeck, S. Provably robust deep learning via adversarially trained smoothed classifiers. *NeurIPS*, 2019.

Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. Adversarially robust generalization requires more data. In *NeurIPS*, pp. 5014–5026, 2018.

Shrivastava, A., Gupta, A., and Girshick, R. Training region-based object detectors with online hard example mining. In *CVPR*, pp. 761–769, 2016.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, pp. 1929–1958, 2014.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *ICML*, 2019.

Touvron, H., Vedaldi, A., Douze, M., and Jégou, H. Fixing the train-test resolution discrepancy. *arXiv preprint arXiv:1906.06423*, 2019.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses. *ICLR*, 2018.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. Robustness may be at odds with accuracy. In *ICLR*, 2019.

Verma, V., Lamb, A., Beckham, C., Courville, A., Mitliagkis, I., and Bengio, Y. Manifold mixup: Encouraging meaningful on-manifold interpolation as a regularizer. *ICML*, 2019.

Wang, D., Gong, C., and Liu, Q. Improving neural language modeling via adversarial training. *ICML*, 2019.

Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A., and Le, Q. V. Adversarial examples improve image recognition. *arXiv preprint arXiv:1911.09665*, 2019.

Yang, Z., Dai, Z., Salakhutdinov, R., and Cohen, W. W. Breaking the softmax bottleneck: A high-rank RNN language model. In *ICLR*, 2018.

Yin, D., Kannan, R., and Bartlett, P. Rademacher complexity for adversarially robust generalization. In *ICML*, pp. 7085–7094, 2019.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. *ICCV*, 2019.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *BMVC*, pp. 87.1–87.12. BMVA Press, September 2016.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

Zhang, H., Yu, Y., Jiao, J., Xing, E., Ghaoui, L. E., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *ICML*, pp. 7472–7482, 2019.

Zhang, X., Wang, Q., Zhang, J., and Zhong, Z. Adversarial autoaugment. *ICLR*, 2020.

Zhu, C., Cheng, Y., Gan, Z., Sun, S., Goldstein, T., and Liu, J. Freelb: Enhanced adversarial training for language understanding. *ICLR*, 2020.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. *ICLR*, 2017.