MixProp: Towards High-Performance Image Recognition via Dual Batch Normalisation

Jinghao Zhang[1]
jinghao.zhang@surrey.ac.uk
Zhenhua Feng[1]
z.feng@surrey.ac.uk
Guosheng Hu[2]
guosheng.hu@oosto.com
Changbin Shao[3]
shaochangbin@163.com
Yaochu Jin[4]
yaochu.jin@uni-bielefeld.de

[1] University of Surrey, School of Computer Science and Electronic Engineering Guildford GU2 7XH, UK
[2] Oosto Belfast BT1 2BE, UK
[3] Jiangsu University of Science and Technology, School of Computer Zhenjiang 212100, China
[4] Bielefeld University Faculty of Technology Bielefeld 33619, Germany

Abstract

Recently, Adversarial Propagation (AdvProp) improves the standard accuracy of a trained model on clean samples. However, the training speed of AdvProp is much slower than vanilla training. Also, we argue that the use of adversarial samples in AdvProp is too drastic for robust feature learning of clean samples. This paper presents Mixup Propagation (MixProp) to further increase the standard accuracy on clean samples and reduce the training cost of AdvProp. The key idea of MixProp is to use mixup to generate samples for the auxiliary batch normalisation layer. This approach provides a moderate dataset as compared with adversarial samples and saves the time used for adversarial sample generation. The experimental results obtained on several datasets demonstrate the merits and superiority of the proposed method.

1 Introduction

In recent years, Deep Neural Networks (DNNs) have demonstrated great success in many computer vision tasks [7, 13, 16]. However, a well-trained model could be vulnerable to adversarial attacks that aim to fool the model by adding imperceptible perturbations [30]. These attackers try to bypass or directly attack a target model thus achieving confrontation purposes, leading a trained DNN to change its prediction of a given image completely. To deal with this challenge, many countermeasures have been proposed to detect or defend against adversarial attacks. For instance, Samangouei *et al*. [27] proposed a method using Generative Adversarial Networks (GANs) [8] to deal with adversarial attacks. A contrastive-learning-based mechanism was proposed by Jiang *et al*. [11] to create label-efficient and robust models against adversarial attacks. Xie *et al*. [33] introduced a feature denoising method that uses non-local means and other filters to improve the adversarial robustness of deep networks. As one of the most popular defence methods, adversarial training [12, 19] directly uses augmented adversarial samples to enhance the robustness of a trained deep network.

In spite of its success, the use of adversarial training is not without difficulties. Adversarial training improves the robust accuracy of the trained model on adversarial samples but also reduces its standard

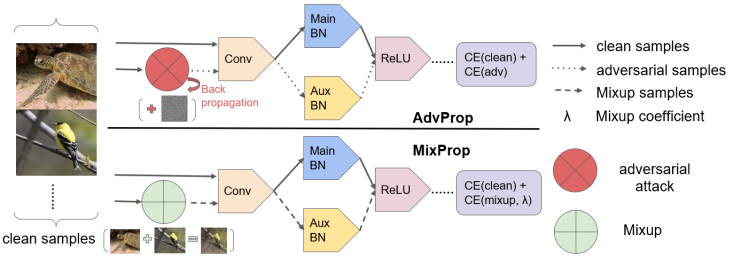


Figure 1: A comparison between AdvProp and MixProp. AdvProp generates adversarial samples for each clean image during the training stage using PGD attacks. However, MixProp uses the mixup of two clean images as the input for the auxiliary BN layers, resulting in efficient network training.

accuracy on clean samples. This is the well-known 'trade-off' problem. Many existing studies found that adversarial training may significantly hurt the generalisation of a model and reduce its standard accuracy [31, 37]. It is extremely difficult to train a model that performs well on both clean and adversarial samples. For this reason, the more noise generated by adversarial attacks 'pollutes' the adversarial training process, the less generalisation capability on clean samples achieves [39].

Instead of improving the robust accuracy of a deep model on adversarial samples, recent studies find that the use of adversarial samples can also improve the generalisation capability and standard accuracy of a trained DNN on clean samples. For example, Adversarial Propagation (AdvProp) [34] manages to achieve the above goals by using two Batch Normalisation (BN) layers, namely main and auxiliary BNs, to process clean and adversarial samples, respectively. As shown in Fig. 1, AdvProp uses all the clean samples to calculate the statistics of the main BN layer, and only uses the adversarial samples for the auxiliary BN layer. On the one hand, the adversarial samples used in AdvProp enhance diverse and robust feature learning for performance boosting on clean data. On the other hand, the use of two BN layers maintains the cleanness of the statistics stored in the main BN layer.

However, the existing AdvProp paradigm has two major issues. First, it requires much more computational resources than vanilla training as the adversarial samples have to be generated on-the-fly. More importantly, we argue that the use of adversarial samples might be too drastic for representation learning even though the auxiliary BN layer mitigates their impact to some extent. To address the above issues, this paper advocates Mixup Propagation (MixProp), which further boosts the performance in standard accuracy and reduces the training cost as compared with AdvProp. Fig. 1 illustrates the design of the proposed MixProp method. Instead of using adversarial samples, the proposed MixProp method synthesises new training samples via Mixup [38], a common data augmentation method that combines two images and their labels with linear interpolation. In MixProp, the main and auxiliary BN layers process clean and mixup samples, respectively. We also introduce a mixup loss function for further performance boosting. Our MixProp framework has the following advantages:

- The training cost of MixProp is significantly reduced compared with AdvProp. AdvProp has to generate adversarial perturbations via a very complicated generation network that is computationally expensive. By contrast, the mixup of two images is computationally very efficient.
- MixProp outperforms AdvProp on both clean images and distorted images without retraining. This demonstrates the robust feature learning capability of the proposed method against various types of image degradation.

2 Related work

Adversarial samples have become a major threat to DNNs since they were discovered. They can fool a trained network and intentionally cause wrong prediction results. There are many types of adversarial attacks in computer vision. For example, unlike a unique perturbation for each image,

different images could share an identical perturbation to fool a classifier. We call such perturbations as Universal Adversarial Perturbations (UAPs) [24]. Some studies explored approaches that try to generate the smallest perturbations, increasing the difficulty of being identified [23]. Under extreme conditions, only one pixel can be attacked to fool a classifier [29]. Researchers have also found that adversarial samples may exist in the real world [15], causing malfunctions in a trained network through physical image or video capturing.

To defend against adversarial attacks, the most widely used method is adversarial training [14, 20]. The key idea of adversarial training is simple and straightforward. It includes the adversarial samples generated by an attack method in the training stage of the network. By this means, the trained model learns the features of the adversarial samples and improves its robustness against adversarial attacks. However, this takes much more time than vanilla training.

Most of the existing adversarial training methods focus on improving the performance (robust accuracy) of a trained model on adversarial samples [2, 6, 20]. However, these approaches may decrease the standard accuracy of the trained model on clean samples [12, 14, 33]. When the robust accuracy goes up, the standard accuracy usually goes down. Maini *et al*. [21] developed a model that is robust to multiple adversarial attack methods, which naturally incorporates different gradient-based perturbation models into a single unified adversary to find the worst-case loss. Wang *et al*. [32] proposed an adversarial training method to adjust the trade-off between standard and robust accuracy at the inference stage. This method does not require retraining the network many times to tune the standard or robust accuracy. It builds on the model-conditional adversarial training framework and uses a balancing hyper-parameter as an input. Therefore, the user could modify standard and robust accuracy by adjusting the balancing hyper-parameter. This approach saves a lot of time, but the upper limits of standard and robust accuracy are not increased. Liu and Jin [18] advocated neural architecture search and found that the searched network can improve both the standard accuracy and robust accuracy. In addition, this method could defend against multiple attacks.

Lin *et al*. [17] proposed filter pruning using GAN, but the training of the GAN model is difficult to converge. Cui *et al*. [3] tried to leverage the clean model to improve the standard accuracy of the adversarial training model. They expected the logit output of an adversarial sample on a robust model to be similar to the logit output of a corresponding clean image on a clean model. By design, the clean model could help the robust model classify the clean images into ground-truth classes. But the standard accuracy still drops a bit by using this approach.

Instead of the promising robust accuracy results achieved by the existing adversarial training methods, many recent studies also tried to improve the standard accuracy of DNNs via the adversarial learning paradigm. For example, Ho and Nvasconcelos [10] proposed contrastive learning with adversarial learning, which uses adversarial samples to maximise the diversity of the pairs, enabling contrastive learning to achieve better performance. Xie *et al*. [34] introduced Adversarial Propagation (AdvProp) that processes clean and adversarial samples separately by using paralleled Batch Normalisation (BN) layers. The dual-BN design could disentangle the distributions of clean and adversarial samples. But AdvProp suffers from low training speed. Mei *et al*. [22] managed to accelerate the training of AdvProp by reducing adversarial attack iterations and decoupling clean and adversarial image pairs. The final training budget is reduced to the level of vanilla training.

3 The Proposed MixProp Method

3.1 Problem Statement and Analysis of AdvProp

As aforementioned, many existing adversarial training methods [28, 31, 32, 36] have a trade-off issue. It is usually difficult for a model to perform well on both clean and adversarial samples. Our assumption about this trade-off issue is that the standard training and adversarial training of a network

may have conflicting objectives. Adversarial samples add noise to the original clean data, shifting the data distribution. However, it is well-known that the feature learning of a deep network relies heavily on the distribution of the training data. By injecting adversarial samples into a clean dataset, it shifts the feature learning results of the trained network significantly, resulting in performance degradation of the model on the clean dataset.

Adversarial Propagation (AdvProp) [34] aims to improve the generalisation capability of a trained deep network on clean samples via the adversarial training paradigm. To address the data distribution shift problem, AdvProp uses two BN layers in parallel for network training. The main BN layers process clean samples and the auxiliary BN layers process adversarial samples, respectively. During the training process of AdvProp, the adversarial samples boost the performance in standard accuracy by enhancing the feature diversity and robustness, and the dual-BN design maintains the statistics of the clean dataset stored in the main BN layers, alleviating the distribution shift problem. AdvProp has demonstrated promising performance not only on the image classification task but also on many other tasks, such as object detection and contrastive learning [1, 10].

One major disadvantage of AdvProp is that its training cost is much higher than vanilla training. The main reason is that AdvProp generates adversarial samples by Projected Gradient Descent (PGD) [20] on-the-fly during the network training stage. AdvProp requires 7 times more forward and backward passes than vanilla training when using PGD-5 [22]. This heavy training cost limits its application to more complicated tasks and larger datasets. The other disadvantage of AdvProp is that the adversarial samples generated by PGD-1 still disturb the standard accuracy. In this paper, we further boost the performance of AdvProp by proposing MixProp which also reduces the training budget significantly.

3.2 MixProp

3.2.1 Data Moderation

As aforementioned, one major issue of AdvProp is that the use of adversarial samples is too drastic and may disturb the feature learning on the clean data. The aim of adversarial attacks is to fool a trained network so the online-generated adversarial samples may shift the features of a sample significantly from the original feature embedding. In this case, the adversarial samples may contain features that do not belong to the clean data domain. Although AdvProp uses a dual-BN architecture for clean and adversarial samples, the adversarial samples still affect the learning of other layers of a network during the feed-forward and back-propagation stages, leading to potential standard accuracy degradation. Therefore, we advocate the use of diverse but moderated samples instead of adversarial samples.

To achieve the above goal, we propose Mixup Propagation (MixProp) that synthesises a new dataset via Mixup [38]. To be specific, MixProp uses the mixup of two arbitrary clean samples to generate a new sample instead of generating online adversarial samples in AdvProp. In MixProp, we also use two BN layers in each block, *i.e.,* one BN layer for clean samples and one BN layer for mixup samples. The use of mixup generates samples that are closer to the clean domain and expands the diversity of the clean dataset. The auxiliary BN layer processes mixup samples, allowing the model to learn more clean sample features and improve its standard accuracy. Note that only the main BN layer is used during the inference stage when testing the performance of MixProp in standard accuracy.

The second problem of AdvProp is the high training cost. Fast AdvProp [22] utilises several tricks to reduce the training cost, including reusing unpaired adversarial samples, reducing adversarial attack iterations and recycling gradients. But Fast AdvProp does not further improve the standard accuracy as compared with AdvProp. In contrast, the proposed MixProp method uses mixup samples that can be generated very efficiently as compared with AdvProp, reducing the training cost significantly. Also, our MixProp further improves the performance in standard accuracy as compared with AdvProp and Fast AdvProp, demonstrating its advantages in terms of both accuracy and efficiency.

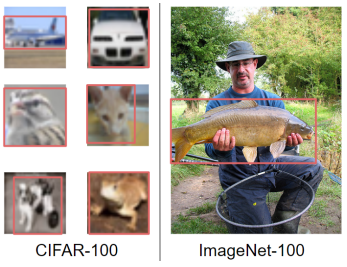


Figure 2: A comparison of the objects in CIFAR-100 and ImageNet-100. An image in ImageNet-100 may contain several objects and the labelled object may only occupy a small part of the whole image.

3.2.2 Attentive Representation Learning

Most adversarial attacks perturb all the pixels in an image, but only some key feature regions are critical to the final decision making [25]. Adversarial perturbations may attract a classifier's attention to some key parts of the target region rather than the background or non-target objects. As shown in [34], the top-1 accuracy of AdvProp obtained on ImageNet and distorted datasets is much better than the results obtained by vanilla training. However, in our evaluation results, the top-1 accuracy of AdvProp is worse than vanilla training when evaluated on the CIFAR-100 dataset, as shown in Table 1. The main difference between CIFAR-100 and ImageNet, except for their resolution, is that an ImageNet image may contain multiple objects and the labelled object may only occupy a small part of the whole image. As demonstrated in Fig. 2, the target object in CIFAR-100 is always in the centre of the image and almost occupies the whole image. We believe the distractions in ImageNet could potentially degrade the performance of a trained network. The use of adversarial perturbations could guide the classifier's attention to the target region when processing samples with a lot of extra background information. This is the main reason why AdvProp performs better than vanilla training on ImageNet.

Note that, the proposed MixProp method has a similar advantage in attentive representation learning. This has also been verified in many existing studies. For example, the experimental results of weakly supervised object localisation in CutMix [35] demonstrate that mixup obtains tighter bounding boxes than those obtained by the baseline method. According to our experimental results in Sec. 4.4, MixProp could guide the feature learning of the trained classifier concentrated on the target in an image.

3.2.3 Loss Function and Network Training

We train the proposed framework in a multi-task optimisation manner. The overall loss function is defined as:

$$Loss = \mathcal{L}_c(\theta, y_c, y_{pred_c}) + \lambda * \mathcal{L}_m(\theta, y_a, y_{pred_m}) + (1-\lambda) * \mathcal{L}_m(\theta, y_b, y_{pred_m}), \quad (1)$$

where $\mathcal{L}_c$ is the cross-entropy loss on clean samples; $\mathcal{L}_m$ is the cross-entropy loss on mixup samples; $\lambda \sim Beta(\alpha, \alpha)$ is the mixing coefficient and $Beta()$ is the Beta distribution; y_c, y_a and y_b are the ground-truth labels of the clean image and two clean images used for mixup, respectively. y_{pred_c} and y_{pred_m} represent the predicted labels of the clean and mixup samples.

As shown in Fig. 1, for each batch of samples, we apply mixup to them and generate mixup samples for the auxiliary BN layer. This procedure takes a copy of the original batch of samples and randomly mixes two arbitrary samples. Then the auxiliary BN layer calculates the statistics of the mixup batch. Meanwhile, the main BN layer just processes the original batch that includes clean samples only. Note that all the layers, except the two BN layers, are optimised jointly with both the clean and mixup samples. After network training, we discard the auxiliary BN layer and only use the main BN layer for image classification.

Table 1: Top-1 accuracy of Preact-ResNet on CIFAR-100

Architecture	Vanilla	Mixup	AdvProp	MixProp
Preact-18	75.68	77.54	75.06	**78.05**
Preact-50	77.08	78.62	76.7	**79.74**

Table 2: Top-1 accuracy of ResNet on ImageNet-100

Architecture	Vanilla	Mixup	AdvProp	MixProp
ResNet-18	83.4	84.1	83.76	**84.58**
ResNet-50	84.84	84.92	**86.18**	85.64

4 Experimental Results

4.1 Implementation Details

Datasets. We train and evaluate our models on CIFAR-100 and ImageNet-100. CIFAR-100 consists of 60000 32x32 colour images of 100 classes. ImageNet-100 is a subset of ImageNet [26] with 100 classes and the other specifications are the same as ImageNet. We also use ImageNet-100-C and Stylized-ImageNet-100 to evaluate the performance of a trained network on distorted images. These two datasets are extracted from ImageNet-C and Stylized-ImageNet. ImageNet-C [9] measures the robustness of a model on 15 diverse corruptions with five severity levels each. The Stylized-ImageNet dataset [4] uses AdaIN style transfer to distort the local textures of an image and retain the global object shapes. The Stylized-ImageNet dataset makes a model to learn more about shapes and less about local textures. Both ImageNet-C and Stylized-ImageNet have 50 images in each class in their validation sets. We reform ImageNet-C and Stylized-ImageNet from 1000 classes to the corresponding 100 classes of the ImageNet-100 dataset and get ImageNet-100-C and Stylized-ImageNet-100.

Experimental Settings. We use Pre-Activation ResNet-18/50 [8] and ResNet-18/50 [7] as the backbone networks. The Pre-Activation ResNet is specially designed for a dataset with very low resolutions, such as CIFAR-100. The use of two different ResNet models can help us understand whether the model size can affect the standard accuracy significantly or not. For ImageNet-100, we use the SGD optimiser with a momentum of 0.9 and weight decay of 5e-4. We train each model for 270 epochs. The learning rate starts at 0.1 and decays by a factor of 0.1 at the 90th, 180th, and 220th epochs. For CIFAR-100, we change training epochs to 320, and decay at the 100th, 180th, and 260 epochs. The batch size is set to 128. We use random crop and random flip for data augmentation. For the coefficient of mixup, we randomly sample λ from the $Beta(1,1)$ distribution for each batch.

AdvProp settings. We use Projected Gradient Descent (PGD) [20] as the attacker to generate adversarial samples on-the-fly for AdvProp. We set the perturbation size ε range from 1 to 3. The number of iterations for the attacker is set to $n = \varepsilon + 1$. Note that when $\varepsilon = 1$, n is still 1. We also fix the attack step size to $\alpha = 1$. Other settings are the same as AdvProp.

4.2 Comparison between AdvProp and MixProp

Table 1 reports the top-1 accuracy of Preact-ResNet-18 and Preact-ResNet-50 obtained on the CIFAR-100 validation set with different methods. We choose PGD-1 and PGD-4 for AdvProp trained with Preact-ResNet-18 and Preact-ResNet-50, respectively, because they have the best performance. The baseline method is 'Vanilla' training that uses a single set of BN layers, which achieves 75.68% top-1 accuracy on CIFAR-100.

In addition to the vanilla training, we also evaluate the single use of mixup. For the 'Mixup' method, we apply mixup to half of the clean samples in each batch under a single set of BN layers. We can see that Mixup further improves the accuracy to 77.54% by replacing 50% of original training samples with mixup samples. This result validates the effectiveness of mixup in deep network training. According

Table 3: The evaluation results obtained on distorted datasets. For ImageNet-100-C, we use the mean Corruption Error (mCE) metric. For Stylized-ImageNet-100, we use the top-1 accuracy.

Architecture	ImageNet-100-C ↓	Stylized-ImageNet-100 ↑
ResNet-18 Vanilla	66.7	18.9
ResNet-18 Mixup	59.1	23.7
ResNet-18 AdvProp (main BN)	61.4	21.8
ResNet-18 MixProp (main BN)	59.2	23.8
ResNet-18 AdvProp (aux BN)	65.7	22.2
ResNet-18 MixProp (aux BN)	**58.6**	**25.8**
ResNet-50 Vanilla	61.8	21.0
ResNet-50 Mixup	56.8	26.0
ResNet-50 AdvProp (main BN)	**54.0**	27.0
ResNet-50 MixProp (main BN)	56.8	24.8
ResNet-50 AdvProp (aux BN)	62.3	26.1
ResNet-50 MixProp (aux BN)	54.4	**29.5**

to the table, we can also find that the use of a more powerful network, Preact-ResNet-50, can achieve better performance when we use the vanilla training method. Preact ResNet-50 achieves 77.08% and 78.62% in terms of the top-1 accuracy when we use vanilla training and Mixup, respectively.

Surprisingly, the top-1 accuracy of AdvProp is lower than that of the vanilla training method, which is against the conclusion obtained by AdvProp [34]. In contrast, the proposed MixProp method outperforms all the other approaches significantly, regardless of the network architecture.

Table 2 reports the performance of different methods in terms of the top-1 accuracy on ImageNet-100 using two network architectures, *i.e.,* ResNet-18 and ResNet-50. Again, Mixup outperforms the vanilla training method. Differently, AdvProp achieves higher top-1 accuracy than vanilla training. However, when we use the ResNet-18 network, AdvProp performs slightly worse than the mixup method. In contrast, MixProp outperforms both vanilla and mixup, regardless of the use of different networks. This demonstrates the robustness of the proposed method in terms of network architecture.

The results obtained on CIFAR-100 and ImageNet-100 validate our argument in Sec. 3.2.2. The adversarial perturbations used in AdvProp not only provide additional training samples but also help attentive feature learning. That is why AdvProp does not work well on CIFAR-100, since the images in CIFAR-100 are already well-cropped. Adversarial attacks cannot improve the attention of a classifier, and perturbations further hurt the generalisation ability of a trained model. Instead, MixProp uses a mixture of clean images, which are harmless to model generalisation.

4.3 Performance on Distorted Images

We also evaluate the performance of the proposed method on distorted images. To this aim, we train different methods on ImageNet-100 and test them on ImageNet-100-C and Stylized-ImageNet-100. For AdvProp and our MixProp, we only use the main BN layers after network training. The experimental results are reported in Table 3. We can see that the proposed MixProp (main BN) method significantly outperforms the vanilla baseline on both distorted datasets when using two different backbones. The improvement is more remarkable than that on the clean datasets. As we can see, MixProp improves mCE by 7.5% on ImageNet-100-C and top-1 accuracy by 4.92% on Stylized-ImageNet-100. Note that these results are obtained when models are not trained with distorted images.

However, MixProp does not always outperform AdvProp (main BN). AdvProp achieves better results on both ImageNet-100-C and Stylized-ImageNet-100 when using ResNet-50. The main reason is that we use mixup samples for the proposed method, which are moderated samples as compared with the adversarial samples in AdvProp. When we use the main BN layers for evaluation, AdvProp

Table 4: A comparison in the training budgets of one epoch.

Perturbation	Vanilla	Mixup	MixProp	AdvProp
PGD-1($\varepsilon=1$)	1N	1N	2N	3N
PGD-3($\varepsilon=2$)				5N
PGD-4($\varepsilon=3$)				6N

Table 5: Actual training time (second) of one epoch on ImageNet-100 using ResNet-18.

Vanilla	Mixup	MixProp	AdvProp(PGD-1)
163s	165s	198s	285s

could perform better than MixProp, especially for a large-capacity network.

4.4 Ablation study

The use of auxiliary BN layers. In AdvProp, only the main BN layers are kept for testing because there is not any 'real' adversarial sample in the test set. The auxiliary BN layers, containing adversarial sample statistics, will not work well on these datasets. The auxiliary BN layers of our method, on the other hand, do not use adversarial samples, but process the mixup of clean samples. The distribution shift between clean and synthesised images is smaller than that between clean and adversarial samples. We can regard the process in the auxiliary BN layers as a kind of 'robustness training' within the clean data distribution. This adjustment makes the auxiliary BN layers capable of being used for robustness evaluation during testing.

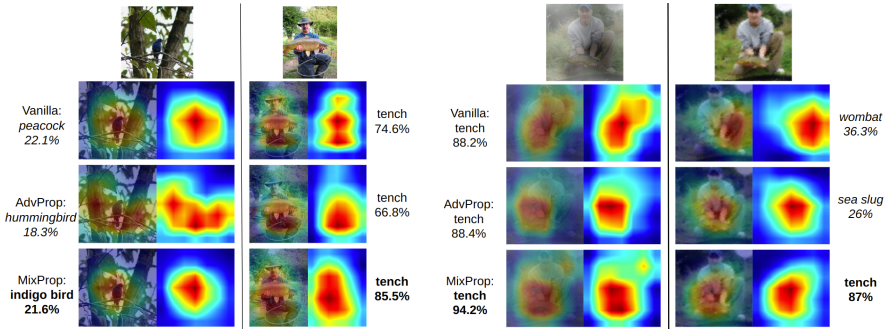
In Table 3, the proposed MixProp (aux BN) method archives better performance. The trained model with ResNet-18 improves the performance of the baseline method by 8.1% in mCE and 6.9% in top-1 accuracy on ImageNet-100-C and Stylized-ImageNet-100, respectively. AdvProp has worse performance when using auxiliary BN layers, which is consistent with the results reported in the paper [34].

A comparison in the training budget. AdvProp requires more training cost than vanilla training because AdvProp needs to generate adversarial samples during the training stage. We compare the training cost of our method with AdvProp following the method used in [22], which denotes the cost of a single forward and backward pass for one image as 1 for simplification and sets the dataset size to N. So, the cost of vanilla training is N for one training epoch. We report the total training cost of different methods in Table 4. Our method, MixProp, only requires $1/3$ to $2/3$ of the AdvProp training cost, depending on the attack iteration.

We also test the actual time used by different methods for one epoch, using a single NVIDIA GTX 3090 card. We set the hyper-parameters the same as in Sec. 4.1. Table 5 presents the actual training time of one epoch on ImageNet-100 using ResNet-18. MixProp takes 21% more time than vanilla training. In contrast, AdvProp with PGD-1 takes 75% more time than vanilla training.

Attentive representation learning. We apply Class Activation Mapping (CAM) [40] to visualise the spatial attentive areas of the classifiers trained with vanilla, AdvProp and MixProp methods. We use ResNet-18 for visualisation. CAM highlights the regions that are discriminative to predict the class by a trained network. As shown in Fig. 3 (a), we can see that the focus region obtained by MixProp is much smaller than that of vanilla and AdvProp when only MixProp has the correct prediction on the left side. Although the dark red region, which we denote as the activation region, is similar between vanilla and MixProp, vanilla has a much wider focus region. AdvProp surprisingly has three focus regions, leading to a wrong prediction.

The performance of MixProp is still the best when all three methods predict correct labels, as shown on the right side of Fig. 3 (a). AdvProp does not make the activation region on the target but concentrates on the fishnet. Despite the fact that both vanilla and MixProp have more precise activation regions, vanilla has more activation regions which split its attention. In contrast, MixProp puts more attention on the target area and gets a higher confidence score.



(a) Clean Images (b) Distorted Images

Figure 3: Visualisation of attentive heatmaps achieved by different methods, using Class Activation Mapping (CAM) [40], on (a) clean images and (b) distorted images. In (a), we show the heatmaps obtained on the 'indigo bird' and 'tench' images. In (b), we show the heatmaps obtained on two 'tench' images of ImageNet-100-C. Each row represents the original images and heatmaps obtained by vanilla, AdvProp and MixProp methods, respectively. The labels and confidence scores are also provided.

We also visualise the results obtained on distorted images of ImageNet-100-C. Vanilla has the smallest activation region when all three methods correctly predict the labels under level three fog distortion on the left side of Fig. 3 (b). However, the fog distortion could make vanilla and AdvProp confused about the location of the target. The activation region of vanilla and AdvProp is away from the target area, focusing on other background objects. MixProp is affected by fog too, but it expands its activation region to get a better sense field. MixProp has the largest activation region but it covers most part of the target area, resulting in the highest confidence score.

The images on the right side of Fig. 3 (b) are affected by level three glass distortion. This time both vanilla and AdvProp misclassify 'tench' to 'wombat' and 'sea slug', respectively. And all the activation regions are misplaced. MixProp achieves a more focused attention region that covers the target area.

Based on our observation, MixProp is able to focus on the target area more precisely. The CAM visualisations validate our statement in Sec. 3.2.2: mixup samples guide the training of a classifier to concentrate on more discriminative features. The improvement is more significant when encountering distorted images. We believe that the mixup process in the auxiliary BN layers guides the parameters in other layers to correctly learn the discriminative features, therefore increasing standard accuracy. At the same time, mixup prevents the model from over-fitting by introducing more possible features. And we can use either the main or auxiliary BN layers for better standard accuracy or robust accuracy.

5 Conclusion

In this paper, we proposed a novel method, called MixProp, that uses a dual-BN architecture for boosting the performance of an image classifier. The proposed method not only boosts the standard accuracy and robustness of the trained model but also reduces the training cost. The proposed method uses the mixup of clean samples to synthesise more comprehensive training samples that are closer to the clean sample distribution than adversarial samples. The synthesised dataset then brings the model's focus to the target area and provides rich and diverse features for better generalisation capability. The proposed method achieves better performance than AdvProp across different datasets without any extra data. However, the choice of the main and auxiliary BN layers during the inference stage is manually chosen in the proposed method. We aim to further address this issue in the future so the BN layers can be adaptively selected for a test sample to achieve good performance on both clean and distorted/adversarial samples.

References

[1] Xiangning Chen, Cihang Xie, Mingxing Tan, Li Zhang, Cho-Jui Hsieh, and Boqing Gong. Robust and accurate object detection via adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16622–16631, 2021.

[2] Minhao Cheng, Qi Lei, Pin-Yu Chen, Inderjit Dhillon, and Cho-Jui Hsieh. Cat: Customized adversarial training for improved robustness. *arXiv preprint arXiv:2002.06789*, 2020.

[3] Jiequan Cui, Shu Liu, Liwei Wang, and Jiaya Jia. Learnable boundary guided adversarial training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15721–15730, 2021.

[4] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2018.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

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

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[9] Dan Hendrycks and Thomas G Dietterich. Benchmarking neural network robustness to common corruptions and surface variations. *arXiv preprint arXiv:1807.01697*, 2018.

[10] Chih-Hui Ho and Nuno Nvasconcelos. Contrastive learning with adversarial examples. *Advances in Neural Information Processing Systems*, 33:17081–17093, 2020.

[11] Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Robust pre-training by adversarial contrastive learning. *Advances in Neural Information Processing Systems*, 33: 16199–16210, 2020.

[12] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[14] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[15] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial Intelligence Safety and Security*, pages 99–112. Chapman and Hall/CRC, 2018.

[16] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE, 2010.

[17] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2019.

[18] Jia Liu and Yaochu Jin. Multi-objective search of robust neural architectures against multiple types of adversarial attacks. *Neurocomputing*, 453:73–84, 2021.

[19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1706.06083.

[21] Pratyush Maini, Eric Wong, and Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *International Conference on Machine Learning*, pages 6640–6650. PMLR, 2020.

[22] Jieru Mei, Yucheng Han, Yutong Bai, Yixiao Zhang, Yingwei Li, Xianhang Li, Alan Yuille, and Cihang Xie. Fast advprop. In *International Conference on Learning Representations*, 2021.

[23] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[24] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[25] Min Ren, Yun-Long Wang, and Zhao-Feng He. Towards interpretable defense against adversarial attacks via causal inference. *Machine Intelligence Research*, 19(3):209–226, 2022.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[27] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.

[28] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. *Advances in neural information processing systems*, 31, 2018.

[29] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[30] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, J. Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[31] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2018.

[32] Haotao Wang, Tianlong Chen, Shupeng Gui, TingKuei Hu, Ji Liu, and Zhangyang Wang. Once-for-all adversarial training: In-situ tradeoff between robustness and accuracy for free. *Advances in Neural Information Processing Systems*, 33:7449–7461, 2020.

[33] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019.

[34] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 819–828, 2020.

[35] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[36] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019.

[37] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, P. Eric Xing, El Laurent Ghaoui, and I. Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, pages 7472–7482, 2019.

[38] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[39] Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. Attacks which do not kill training make adversarial learning stronger. *37th International Conference on Machine Learning, ICML 2020*, PartF16814:11214–11224, 2020.

[40] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.