FAST ADVPROP

Anonymous authors
Paper under double-blind review

ABSTRACT

Adversarial Propagation (AdvProp) is an effective way to improve recognition models, leveraging adversarial examples. Nonetheless, AdvProp suffers from the extremely slow training speed, mainly because: a) extra forward and backward passes are required for generating adversarial examples; b) $2\times$ data (*i.e.*, both the original sample and its adversarial counterpart) is used for training. In this paper, we introduce *Fast AdvProp*, which aggressively revamps AdvProp's costly training components, rendering the method nearly as cheap as the vanilla training setting. Specifically, our modifications in Fast AdvProp are guided by the hypothesis that disentangled learning with adversarial examples is the key for performance improvements, while other recipes (*e.g.*, paired clean and adversarial training samples, multi-step adversarial attackers) could be largely simplified.

Our empirical results show that, compared to the vanilla training baseline, Fast AdvProp is able to further model performance on a spectrum of visual benchmarks, *without incurring additional training cost*. Additionally, our ablations find Fast AdvProp scales better if larger models are used, is compatible with existing data augmentation methods (*i.e.*, Mixup and CutMix), and can be easily adapted to other recognition tasks like object detection.

1 INTRODUCTION

Deep neural networks are highly successful for visual recognition. As fueled by powerful computational resources and massive amounts of data, deep networks achieve compelling, sometimes even superhuman, performance on a wide range of visual benchmarks. However, when testing out of the box, these exemplary models are usually get criticized for lacking generalization/robustness—an increasing amount of works point out that deep neural networks are brittle at handling out-of-domain situations like natural image corruptions (Hendrycks & Dietterich, 2018), images with shifted styles (Geirhos et al., 2018; Hendrycks et al., 2020), *etc*.

Adversarial propagation (AdvProp) (Xie et al., 2020), which additionally feeds networks with adversarial examples during training, emerged as one of the most effective ways to train not only accurate but also robust deep neural networks. The key in AdvProp is to apply separate batch normalization (BN) layers (Ioffe & Szegedy, 2015) to clean training samples and adversarial training samples, as they come from different underlying distributions. Later works further explore the potential of AdvProp on training better models for other recognition tasks (*e.g.*, detection, segmentation) (Chen et al., 2021b; Shu et al., 2020; Chen et al., 2021a; Xie & Yuille, 2020; Wang et al., 2020; Gong et al., 2021), under different learning paradigms (*e.g.*, contrastive learning) (Jiang et al., 2020; Ho & Vasconcelos, 2020; Xu & Yang, 2020), *etc*.

However, the benefits brought by AdvProp do not come for "free"—AdvProp introduces a significant amount of additional training cost, which is mainly incurred by generating and augmenting adversarial training samples. For instance, compared to the vanilla training baseline (where only clean images are involved), the default setting in AdvProp (Xie et al., 2020) results in $7\times$ computational cost, *i.e.*, 5/7 from generating adversarial examples, 1/7 from training adversarial examples, 1/7 from training clean images. This extremely high training cost not only limits the further explorations of AdvProp on larger networks (Xie et al., 2019; Brock et al., 2021; Dosovitskiy et al., 2020), with larger datasets (Sun et al., 2017; Kuznetsova et al., 2020), and for different learning tasks, but also makes the direct comparisons against other low-cost learning algorithms (Zhang et al., 2018; DeVries & Taylor, 2017; Yun et al., 2019; Cubuk et al., 2019b;a) seemingly unfair.

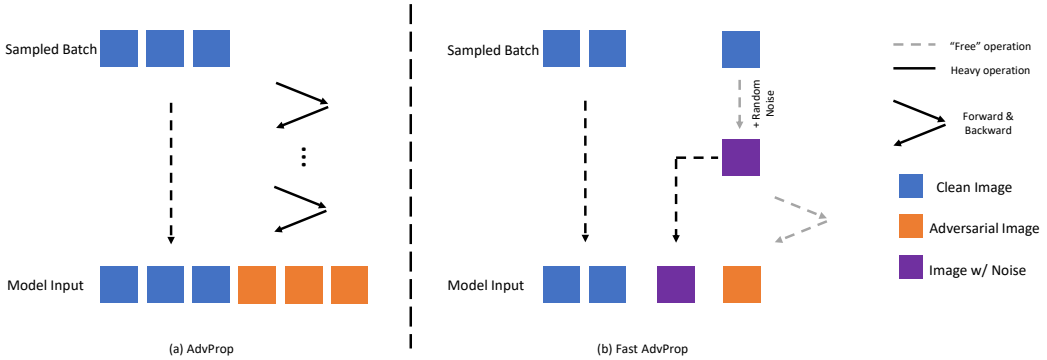

Figure 1: Comparison between AdvProp and Fast AdvProp. (a) AdvProp generates a paired adversarial image for each image in the sampled batch, which incurs heavy additional training cost. Then both the paired adversarial images and the clean images are fed into the network, which involves 2x data compared with the vanilla training. (b) Fast AdvProp uses a small portion of images in a batch for adversarial image generation and the other as clean images. During the generation of the adversarial images, the gradients of network parameters are computed simultaneously with the input images and are reused for network update, thus comes for "free".

In this paper, we present Fast AdvProp, which can run as cheaply as the vanilla training baseline in practice. In particular, noting the heavy computations in AdvProp mainly comes from (1) generating adversarial examples where multiple forward passes and backward passes are additionally required, and (2) training with both clean samples and their adversarial counterparts, Fast AdvProp revamps the original training pipeline as the following:

- Firstly, though both clean training samples and their adversarial counterparts are default components in traditional adversarial training (Goodfellow et al., 2015; Kurakin et al., 2017), we argue such pairing behavior is not a fundamental request by AdvProp. Specifically, in Fast AdvProp, we reposition adversarial examples solely as a *bonus* part for network training, *i.e.*, networks now are expected to train with a mixture of *a large portion of clean images* and *a small portion of adversarial examples*, which helps lower down training cost. Though the number of adversarial examples is reduced in training, our empirical results verify this strategy is sufficient to let networks gain robust feature representations.
- Secondly, we integrate the recent techniques on accelerating adversarial training (Wong et al., 2020; Zhang et al., 2019; Shafahi et al., 2019), which significantly reduces the complexity of generating adversarial examples, into AdvProp. However, this is non-trivial—naively adopting these fast adversarial training techniques will collapse the training, resulting in suboptimal model performance. We identify such failure is caused by the "label leaking" effect (Kurakin et al., 2017), which largely weakens the regularization power imposed by adversarial training samples. We further note this leakage comes from the intra-batch communication, and fix it via Shuffling BN (He et al., 2020). Additionally, we find properly re-balancing the importance between clean training samples and adversarial training samples is another key ingredient for ensuring Fast AdvProp's improvements.

Our empirical results demonstrate Fast AdvProp can successfully improve recognition models for "free". For instance, without incurring any additional training cost, Fast AdvProp helps ResNet-50 (He et al., 2016) outperforms its vanilla counterpart by 0.3% on ImageNet, 2.1% on ImageNet-C, 1.9% on ImageNet-R and 0.5% on Stylized-ImageNet. Furthermore, such "free lunch" can consistently be observed when Fast Advprop is applied to networks at different scales, combined with various data augmentation strategies, and adapted to other recognition tasks. By easing the computational barriers, we hope this work can encourage the community to further explore the potential of AdvProp (or adversarial learning in general) on training robust and accurate deep neural networks.

2 Related Work

Adversarial training. Adversarial training (Szegedy et al., 2014; Goodfellow et al., 2015), which trains networks with adversarial examples that are generated on the fly, is one of the most effective

ways for defending against adversarial attacks. Nonetheless, compared to vanilla training, adversarial training significantly increases the computational overhead, mainly due to the high complexity of generating adversarial examples. To this end, many efforts have been devoted to accelerating adversarial training. Both (Shafahi et al., 2019) and (Zhang et al., 2019) propose to merge the gradient for adversarial attacks and the gradient for network parameter updates into a single forward and backward pass to reduce computations. Wong *et al*. (Wong et al., 2020) alternatively argue that the cheapest adversarial attacker, Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015), actually can train robust classifiers, if combined with random initialization. This work is further enhanced by (Andriushchenko & Flammarion, 2020) to explicitly maximizing the gradient alignment inside the perturbation set for enhancing the quality of the FGSM solution. In this work, we aim to integrate these fast adversarial training techniques into AdvProp, for reducing the overhead of generating adversarial training samples.

Adversarial propagation. It is generally believed that adversarial training hurts generalization (Raghunathan et al., 2019). Adversarial propagation (AdvProp) (Xie et al., 2020), a special form of adversarial training, challenges this belief by showing training with adversarial examples actually can improve recognition models. The key is to utilize an additional set of batch normalization layers exclusively for the adversarial images, as they have different underlying distributions to clean examples. Later works further explore the potential of AdvProp on other recognition tasks (Chen et al., 2021b; Shu et al., 2020; Chen et al., 2021a; Xie & Yuille, 2020; Wang et al., 2020; Gong et al., 2021), under different learning paradigms (Jiang et al., 2020; Ho & Vasconcelos, 2020; Xu & Yang, 2020), with different adversarial data (Merchant et al., 2020; Li et al., 2020), *etc*. In this work, rather than furthering AdvProp on performance improvements, we aim to develop a "free" version of it.

Data augmentation. Data augmentation, which effectively increases the size and the diversity of the training dataset, is crucial for the success of deep neural networks (Krizhevsky et al., 2012; Simonyan & Zisserman, 2015; Szegedy et al., 2015; He et al., 2016). Popular ways for augmenting data include geometric transformations (*e.g*., translation, rotation), color augmentation (*e.g*., brightness, contrast), mixing images (Zhang et al., 2018; Yun et al., 2019; DeVries & Taylor, 2017), *etc*. Training with adversarial examples can be regarded as a special way to augment data—different from traditional data augmentation strategies which are usually fixed and model agnostic, the policy of generating adversarial examples is jointly evolved with the model updating throughout the whole training process. Nonetheless, a significant drawback of augmenting adversarial examples is that the introduced computational overhead is much more expensive than that of traditional augmentation strategies. We hereby aim to make training with adversarial examples as cheap as other data augmentation strategies.

3 Fast AdvProp

In this section, we present Fast AdvProp, which aggressively revamps the costly training components in the original AdvProp. Particularly, our modifications mainly focus on how to reduce the computational overheads stemmed from adversarial examples, while (empirically) still attempt to retain the benefits brought by AdvProp.

3.1 Revisiting AdvProp

Advprop (Xie et al., 2020) demonstrates that adversarial examples could improve recognition models. By realizing adversarial images and clean images have different underlying distributions, AdvProp bridges such distribution mismatch by using two BN scheme—the original BN layers are exclusively for clean images, and the auxiliary BN layers are exclusively for adversarial images, This scheme guarantees that the BN layers are executed on a single data source. More concretely, in each iteration,

- **Step 1:** Advprop first samples a batch of clean images from the training dataset, and then applies the adversarial attack to generate the corresponding adversarial images, using the auxiliary BN layers.
- **Step 2:** The clean images and their adversarial counterparts are then fed into the network as a pair. Specifically, the original BN layers are applied exclusively on clean images, and the auxiliary BN layers are applied exclusively on adversarial images.
- **Step 3:** The loss from adversarial images and clean images are jointly optimized for updating network parameters.

Table 1: The performance of vanilla training, AdvProp, AdvProp on the same training budget, our Fast-AdvProp on various datasets.

	IMAGENET ↑	IMAGENET-C ↓	IMAGENET-R ↑	S-IMAGENET ↑	TRAINING BUDGET
Vanilla Training	76.2	58.5	36.3	7.8	1×
PGD-5 AdvProp	77.0	52.0	42.3	12.0	7×
15-epoch PGD-5 AdvProp	66.8 (-9.4)	66.2	31.7	8.6	1×
PGD-1 AdvProp	77.5	54.3	39.5	8.6	3×
35-epoch PGD-1 AdvProp	73.9 (-2.3)	59.6	36.5	9.3	1×
Fast-AdvProp (ours)	76.5 (+0.3)	56.4	38.2	8.3	1×

Xie *et al*. (Xie et al., 2020) show AdvProp substantially improves both the clean images accuracy, as well as the model robustness. We confirm it in our re-implementation—as shown in the second row of Table 1, AdvProp helps ResNet-50 beats its vanilla counterpart by 0.8% on ImageNet (Russakovsky et al., 2015), 6.5% on ImageNet-C (Hendrycks & Dietterich, 2018), 6.0% on ImageNet-R (Hendrycks et al., 2020) and 4.2% on Stylized-ImageNet (Geirhos et al., 2018).

But meanwhile, we note AdvProp significantly increases the training cost. For example, the AdvProp that we reimplement here requires 7× more forward and backward passes than the vanilla baseline. Such heavy training cost not only limits the broader exploration with AdvProp, but also makes the comparisons to other learning strategies (which are usually "free") seemly unfair.

To reduce the computational cost, we first give a naive attempt to simplify AdvProp's training pipeline. Specifically, given PGD-5 AdvProp here is 7× more expensive than the vanilla baseline, we directly reduce its total number of training epochs by 7 (*i.e*., from 105 epochs to 15 epochs). As shown in the third row of Table 1, this *15-epoch PGD-5 AdvProp* substantially degrades the original AdvProp's performance (*i.e*., 66.8% *vs*. 77.0%), even making the resulted model attains much lower performance than the vanilla training baseline. Moreover, we verify that applying the cheapest PGD-1 training (*i.e*., FGSM + random initialization as in (Wong et al., 2020)) to AdvProp still leads to much inferior performance. These result demonstrate that the task of reducing AdvProp's training cost is non-trivial, therefore motivate us to explore more sophisticated solutions next.

3.2 LIGHTENING ADVPROP

We hereby carefully diagnose the design choice of AdvProp. Particularly, we identify the heavy training cost in AdvProp mainly comes from (1) both adversarial examples and their clean counterparts are used in training, and (2) the generation process of adversarial examples where multiple extra forward and backward passes are needed, which we plan to ablate next.

Note that we always keep the disentangled learning behavior with adversarial training samples (*i.e*., keep separate BN layers for adversarial samples and clean samples) unchanged in all ablations, as we assume this is the key for gaining robust features from adversarial examples.

3.2.1 UNPAIRING TRAINING SAMPLES IN ADVPROP

Let's consider the training cost of one epoch. We denote the cost of forward and backward pass for one image as 1 [1], and the dataset size as N. Then the cost of vanilla training for one epoch is

$$cost(\text{Vanilla}) = N. \tag{1}$$

The cost of AdvProp using PGD-K attack (Madry et al., 2018) for one epoch is:

$$cost(\text{AdvProp}) = N + K * N + N = (K + 2) \times N, \tag{2}$$

where the first part (*i.e*., N) refers to the cost of training with clean samples, the second part (*i.e*., $K * N$) refers to the cost of generating adversarial examples, and the third part (*i.e*., N) refers to the cost of training with adversarial examples. The original AdvProp implementation (Xie et al., 2020) by default use $K = 5$, therefore resulting in 7× training cost compared with the vanilla training baseline. This high training cost makes AdvProp challenging to scale to computationally-intensive settings (Xie et al., 2019; Dosovitskiy et al., 2020).

[1] During a standard network training step, we compute the gradient for all parameters used in the forward process; while for attacking the network, we only compute the gradient with respect to the input images. We ignore the difference here for simplification in cost calculation.

Table 2: ImageNet accuracy (%) on AdvProp with different settings. +ADVPROP denotes the original AdvProp. +1 ITER uses the AdvProp with PGD-1 attacker. +DECOUPLED means decoupled training with 20% of the images for adversarial examples and the others solely as clean images. The last column reports the training budgets of each epoch for different settings.

+ADVPROP	+1 ITER	+DECOUPLED	IMAGENET ↑	IMAGENET-C ↓	TRAINING BUDGET
			76.2	58.5	N
✓			77.0	51.9	7N
✓	✓		77.5	54.3	3N
✓	✓	✓	76.6	56.8	1.2N

Use PGD-1 Attack Firstly, rather than using $K = 5$, we use $K = 1$ as the default setting for reducing the training cost. Note this simplification degrades the PGD attacker to the simple FGSM attacker (Goodfellow et al., 2015) with random noise initialization. As shown in Table 2, compared to the default AdvProp, this simplification increase the top-1 accuracy by 0.5% on ImageNet, but at the cost of sacrificing the robustness on ImageNet-C (*i.e.*, 2.4% higher mCE). Note this step reduces the training budget from $7N$ to $3N$.

Decoupled Training The AdvProp implicitly introduces a constraint that for each clean image, we should generate a paired adversarial image for jointly training, *i.e.*, within each iteration, the paired clean and adversarial images are fed into the network. We empirically demonstrate that the requirement for paired input is not necessary. With such decoupled training, the training cost is:

$$cost(\text{Fast AdvProp}) = p_{clean} * N + p_{adv} * (K + 1) * N \tag{3}$$

where p_{clean} and p_{adv} are the percentages of images used for clean examples and adversarial examples in decoupled training, respectively. We note the standard AdvProp is a special case with $p_{clean} = p_{adv} = 1$. In our implementation, we set p_{adv} of the whole training dataset for adversarial image generations, and the rest as clean examples. The formulation now can be simplifies as

$$cost(\text{Fast AdvProp}) = (1 - p_{adv}) * N + p_{adv} * (K + 1) * N = (p_{adv} * K + 1) * N \tag{4}$$

We use $p_{adv} = 0.2$ in Table 2, which reduces the training budget to $1.2N$. In this setting, we beat the vanilla baseline by 0.4% accuracy on ImageNet and by 1.7% on ImageNet-C.

3.2.2 INCORPORATING FREE ADVERSARIAL TRAINING TECHNIQUES

Decoupled training (using equation 4) allows AdvProp to use the same number of images as the vanilla setting for training, but the resulted strategy still incurs extra training costs. They come from the fact that some forward-backward passes are used for adversarial attacks, which requires the gradient of the input images for generating adversarial perturbations; such gradients are not used in updating network parameters.

Recently, some work focus on accelerating adversarial training by recycling the gradient (Shafahi et al., 2019; Zhang et al., 2019), or using FGSM attacker and cosine learning rate with mixed-precision training (Wong et al., 2020). Inspired by the gradient recycle techniques (Shafahi et al., 2019; Zhang et al., 2019), we expect each forward-backward pass can be fully utilized for updating network parameters. Specifically, we simultaneously generate the gradient for both the input images and network parameters when generating the adversarial examples. The gradients of the network parameters would be used for network updates and the gradients of the input images are used for generating the adversarial images, respectively. In this way, the additional cost of input image gradient generation is marginal compared with the vanilla back-propagation, resulting that the adversarial attack can come for "free". As analyzed by equation 4, the training cost is $(p_{adv} * K + 1) * N$ for one epoch. To let the training budget be exactly the same as the vanilla training, we calibrate the training by reducing the total number of epochs by a factor or $(p_{adv} * K + 1)$.

Nonetheless, interestingly, naively recycling the gradients make the training unstable. We utilize the following techniques to stabilize the network training in Fast AdvProp, which is summarized in Algorithm 1.

Algorithm 1: Pseudo code of Fast AdvProp for T epochs, given some radius ϵ, importance re-weight parameter β, learning rate γ, and ratio of adversarial examples p_{adv}.

Data: A set of clean images with labels;
Result: Network parameter θ
for $epoch = 1 \dots T/(p_{adv} + 1)$ **do**
  Sample a clean image mini-batch X with label Y
  Split X into X_1, X_2 with ratio $1 - p_{adv}$ and p_{adv} respectively
  Generate the adversarial examples X_{adv} using X_2
    $\delta \sim \mathcal{U}(-\epsilon, \epsilon)$
    $g_\delta \leftarrow \nabla_\delta l(x + \delta, y, \theta)$
    $\delta' \leftarrow \delta + \epsilon \cdot sign(g_\delta)$
    $X_{adv} = X_2 + clip(\delta', -\epsilon, \epsilon)$
  Calculate gradients for X_1, X_2, X_{adv}, note g_θ^{noise} and g_δ could be compute simultaneously
    $g_\theta^{clean} \leftarrow \mathbb{E}_{x \in X_1}[\nabla_\theta l(x, y, \theta)]$
    $g_\theta^{noise} \leftarrow \mathbb{E}_{x \in X_2}[\nabla_\theta l(x + \delta, y, \theta)]$
    $g_\theta^{adv} \leftarrow \mathbb{E}_{x \in X_{adv}}[\nabla_\theta l(x, y, \theta)]$
  Fuse the gradients
    $g_\theta \leftarrow g_\theta^{clean} + \beta \cdot g_\theta^{noise} + \beta \cdot g_\theta^{adv}$
  Update θ using gradient descent
    $\theta \leftarrow \theta - \gamma g_\theta$
end

Use non-targeted attack AdvProp uses targeted attack where the target label is random sampled from the wrong classes. The gradients are not reusable as no ground-truth information is provided during the attack process. On the contrary, the loss computation in non-targeted attack is the same as vanilla training, which makes gradient reuse possible. The only difference occurs in the optimization process. In non-targeted attack, we maximize the loss with respect to the input images while in vanilla training we minimize the loss with respect to the network parameters.

Bridge the gap between the running/batch statistics in BNs In the real attack scenarios, the attacker performs the attack using the running statistics. AdvProp also uses the running statistics during the generation of the adversarial images. Nonetheless, we observe unstable training with NaN loss if we reuse the gradient using the running statistics. We conjecture this comes from the inconsistent gradient paths as we use the batch statistics for the clean examples and adversarial examples, and use the running statistics for attacking. To remove that inconsistency, we use the batch statistics instead for adversarial image generation.

Shuffling BN for dealing with information leakage When attack using running statistics and non-targeted attack, we observe the training accuracy is unreasonable high, as shown in Fig 2. The network could classify the images by "cheating" since we use the same batch for adversarial attack and adversarial training, and the intra-batch information exchange introduced by BN leaks the information. We use Shuffling BN (He et al., 2020) to solve this problem. Before feed the adversarial images on each GPU into the network, we shuffle the generated adversarial examples across the multiple GPUs. This ensures the batch statistics for the noisy images and the adversarial images come from different subsets, which solves this problem effectively.

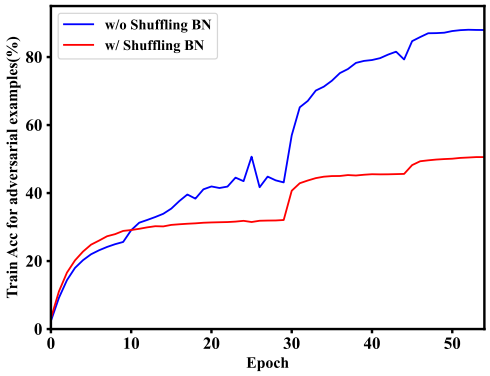

Figure 2: Illustration of the information leakage.

Re-balance the examples By reusing the gradient, the images used for adversarial attack involve two forward and backward passes (adversarial attack and adversarial training), while the clean images involve only one. With the intuition that each image within the batch should have the

same importance, we half the importance for the images used for adversarial attack. In another perspective, the adversarial images and their intermediate product (the noisy images) are two kinds of augmentations on the original images. As in Hoffer et al. (2020), the overall importance for an image should be the same w/ or w/o the repeated augmentations, which is implemented by halving the importance of each augmentation (*i.e.*, $\beta = 0.5$ in Algorithm 1).

Synchronizing the update speed of the parameters The decoupled training strategy enables us to reduce the extra training costs significantly. Nonetheless, it causes problems when combines with the auxiliary BNs scheme. As the noise examples and adversarial examples use the auxiliary BNs and the clean images use the original BNs, the parameters of original/auxiliary BNs receive the gradients from the corresponding examples, while the parameters of the shared layers receive gradients from all examples. The ratio between the gradient magnitude of shared/original BNs/auxiliary BNs is $1 : (1 - p_{adv}) : p_{adv}$. The inconsistent update speed within the network harms the performance. To solve this problem, we re-scale the gradient to ensure the similar parameter update speed.

4 EXPERIMENTS

4.1 EXPERIMENTS SETUP

Dataset We evaluate model performance on ImageNet classification and the robustness on different specialized benchmarks including ImageNet-C, ImageNet-R, and Stylized ImageNet. ImageNet dataset contains 1.2 million training images and 50000 images for validation of 1000 classes. ImageNet-C measures the network robustness on 15 common corruption types, each with 5 severity. ImageNet-R contains stylish renditions like cartoons, art, and sketches of 200 ImageNet classes resulting in 30000 images. Stylized-ImageNet dataset keeps the global shape information while removing the local texture information using the AdaIn style transfer (Huang & Belongie, 2017).

Implementation Details We use the renowned ResNet family (He et al., 2016) as our default architectures. We use a SGD optimizer with momentum 0.9 and train for 105 epochs. The learning rate starts from 0.1 and decays at 30, 60, 90, 100 epochs by 0.1. We use a batch size of 64 per GPU for vanilla training. For decoupled training setting, we use a batch size of $64/(1 - p_{adv})$ per GPU, keeping the same 64 batch size per GPU for the original BNs. p_{adv} is set to 0.2 if not specified.

For a strictly fair comparison, we scaling the total epochs and decay epochs by the relative training budget to vanilla training ($p_{adv} + 1$ in Fast AdvProp, $K + 2$ in AdvProp) in the same budget setting. To generate the adversarial images, we using the PGD attacker with random initialization. We attack for one step and the perturbation size is 1.0. As discussed in Sec. 3.2.2, to ensure the same importance of each example within a batch, we half the loss for the noisy images and the adversarial images by setting $\beta = 0.5$. We re-scale the gradient to achieve the $1 : 1 : 1$ ratio for the magnitude of the gradient between the shared/original BNs/auxiliary BNs parameters.

4.2 MAIN RESULTS

Table 1 compare our method with AdvProp and the vanilla training on ResNet-50. Our re-implementation of AdvProp achieves 77.0% top-1 accuracy on ImageNet, 0.8% higher than the vanilla training baseline. We also evaluate the networks' robustness generalization on ImageNet-C, ImageNet-R, Stylized-ImageNet, which is much more challenging as the vanilla training baseline only achieves a 58.5% error rate on ImageNet-C, 36.3% accuracy on ImageNet-R, and 7.8% accuracy on Stylized ImageNet. The AdvProp also shows compelling robustness compared with the vanilla training, verifying its effectiveness. However, the comparison is unfair as AdvProp using $7\times$ training budget. In the same budget setting, the performance of 15-epoch AdvProp (dividing the total epochs by 7) degrades significantly. The top-1 accuracy on ImageNet is 66.8%, 9.4% lower than the vanilla training baseline. So AdvProp highly relies on the extra training budget to achieve satisfactory performance, which suffers from the extremely slow training speed.

On the contrary, using exactly the same training budget as the vanilla training, our method achieves 76.5% accuracy on ImageNet, 0.3% higher than the vanilla baseline and significantly higher than the 15-epoch AdvProp. It also shows better robustness on ImageNet-C, ImageNet-R, and Stylized-ImageNet compared with the vanilla baseline. With comparable robustness on Stylized-ImageNet, our method beats 15-epochs AdvProp significantly on ImageNet, ImageNet-C, and ImageNet-R.

Table 3: The performance on ImageNet and network robustness on various datasets of vanilla training and our Fast AdvProp.

	IMAGENET ↑	IMAGENET-C ↓	IMAGENET-R ↑	S-IMAGENET ↑
ResNet-50	76.2	58.5	36.3	7.8
+ Fast AdvProp	76.5	56.4	38.2	8.3
ResNet-101	77.8	54.7	40.1	9.1
+ Fast AdvProp	77.8	51.7	41.0	10.8
ResNet-152	78.3	52.3	40.5	10.5
+ Fast AdvProp	78.6	49.8	42.0	12.4

Additionally, our method scales better with larger networks. As shown in Table 3, Fast AdvProp helps the large ResNet-152 achieve 78.6% top-1 accuracy on ImageNet, 49.8% mCE on ImageNet-C, 42.0% top-1 accuracy on ImageNet-R and 12.4% top-1 accuracy on Stylized-ImageNet, beating its vanilla counterpart by 0.3% on ImageNet, 2.5% on ImageNet-C, 1.5% on ImageNet-R and 1.9% on Stylized-ImageNet, respectively.

4.3 ABLATION STUDY

The importance of decoupled training In the Table. 4, we take a close look at the influence of p_{adv}. We can draw a clear conclusion that the larger the p_{adv}, the more inferior top-1 accuracy we achieved. Specifically, using 50% images as adversarial examples only gets 75.4% accuracy, 0.9% lower than the vanilla training. This comes from the fact that as p_{adv} increases, the network goes through fewer epochs for the dataset, as the training budget for each epoch increases as in equation 4.

The decoupled training strategy enables us to train the networks with a mixture of a large portion of clean images and a small portion of adversarial examples, which helps the network go through the dataset with enough epochs. We choose 4:1 as our default setting to keep a reasonable batch size of 16 for the adversarial examples.

Table 4: Ablation study on the influence of the percentage of the adversarial images.

p_{adv}	IMAGENET ↑	EPOCHS
0.20	76.5	87
0.33	76.0	79
0.50	75.4	70

Are the example re-balancing and update speed synchronizing important for Fast AdvProp? Here we focus on the influence of the example re-balancing and the parameter update speed synchronizing. In the default setting, the weight ratio between the clean/noise/adversarial examples is $1:1:1$ without adjustment. The ratio of update speed between the shared/clean only/adversarial only parameters is $1:0.8:0.2$ where $p_{adv}=0.2$. We don't observe any performance gain in this setting as shown in the Table 5. The top-1 accuracy is 76.2%, almost the same as the vanilla training.

We find it is important to simultaneously adopt those two strategies, which achieves 76.5% accuracy. Ignoring the re-balancing for the examples lowers the accuracy by 0.18%, gives us a 76.4% accuracy as some examples are more important than the others, which breaks the assumption that the overall importance for each example should be the same (Hoffer et al., 2020). Removing the update speed synchronizing deteriorates the performance to 76.3%, 0.26% lower than our final setting. This comes from the inconsistent update speed for different parts of the network.

Information leakage problem We find that the shuffling BN is important for gradient reusing. Without Shuffling BN, the training accuracy on the adversarial images reaches 88.0%, even higher than the training accuracy on the clean images, 73.3%, which is abnormal as the adversarial images are much more difficult than the clean images. The shuffling BN technique solves this problem effectively. We observe a smooth and reasonable curve in Figure 2. The validation accuracy boosts from 74.5% to 75.1%, which proves the harm of the information leakage problem.

Table 5: Ablation study on the influence of re-balancing the adversarial examples and the clean examples.

SYNCHRONIZING THE UPDATE SPEED	SAME IMPORTANCE	IMAGENET ↑
		76.20
	✓	76.27
✓		76.35
✓	✓	**76.53**

Combining with other data augmentations Similar to the original AdvProp, our Fast AdvProp could be viewed as a data augmentation method from the perspective of increasing the size and diversity of the dataset using the adversarial examples. Nonetheless, with the help of the adversarial examples, our Fast AdvProp not only regularizes the network to improve the accuracy on the validation set but also increases the robustness to common corruptions. We combine our method with common data augmentation methods including Mixup (Zhang et al., 2018) and CutMix (Yun et al., 2019). Mixup and CutMix require additional training costs compared with vanilla training. For Mixup, we train 180 epochs and decay the learning rate by 0.1 every 60 epochs. For CutMix, we train for 210 epochs and decay the learning rate at 75, 150, 180 respectively. To integrate them with our method, we apply the extra data augmentations on the clean images only. The training budget is the same w/ or w/o our method.

Our method achieves comparable performance on the ImageNet and much better robustness. As shown in Table 6, when combining with CutMix, our method beats the baseline by 2.1% on ImageNet-C, 1.3% on ImageNet-R, 2.5% on Stylized-ImageNet. These results illustrate our method is compatible with existing data augmentation to further improve the network performance and robustness.

Table 6: Robustness when combining our method with existing data augmentation methods.

	IMAGENET-C ↓	IMAGENET-R ↑	S-IMAGENET ↑
CutMix	58.9	35.2	5.5
+Fast AdvProp	55.0	36.5	7.0
MixUp	53.4	41.0	10.0
+Fast AdvProp	52.2	41.7	10.9

Object detection results We implement Fast AdvProp on object detection and evaluate it on the COCO dataset (Lin et al., 2014). We adopt RetinaNet (Lin et al., 2020) as the detection method without freezing the BN's running statistics. We train 24 epochs for the baseline and 20 epochs for Fast AdvProp. We benchmark both the standard performance and the robustness on 15 common corruption types. Our method beats the baseline by 1.4% in mAP corrupted as shown in Table 7.

Table 7: Object detection mAP(%) and robustness measurements on the COCO validation set.

	mAP clean	mAP corr.
Vanilla Training	35.8	17.6
Fast-AdvProp	35.8	19.0

5 CONCLUSION

AdvProp is an effective way to get effective and robust networks. However, it suffers from extremely high training costs. We propose the decoupled training where the adversarial images are only a small portion of the dataset, which reduces the training cost significantly. Also, we succeed in reusing the gradient during the generation of adversarial examples by various techniques. Empirically, our Fast AdvProp algorithm helps the networks on both effectiveness and robustness without additional training cost compared with the vanilla training. We believe the efficiency of our method will speed up the iteration of future works and accelerating the research on leveraging the adversarial examples.

REFERENCES

Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. In *NeurIPS*, 2020.

Andrew Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, 2021.

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*, 2021a.

Xiangning Chen, Cihang Xie, Mingxing Tan, Li Zhang, Cho-Jui Hsieh, and Boqing Gong. Robust and accurate object detection via adversarial learning. In *CVPR*, 2021b.

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

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

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

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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 *ICLR*, 2018.

Xinyu Gong, Wuyang Chen, Tianlong Chen, and Zhangyang Wang. Sandwich batch normalization. *arXiv preprint arXiv:2102.11382*, 2021.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

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

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *arXiv preprint arXiv:2006.16241*, 2020.

Chih-Hui Ho and Nuno Vasconcelos. Contrastive learning with adversarial examples. *arXiv preprint arXiv:2010.12050*, 2020.

Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *CVPR 2020*, pp. 8126–8135. IEEE, 2020.

Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 1510–1519. IEEE Computer Society, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Robust pre-training by adversarial contrastive learning. *arXiv preprint arXiv:2010.13337*, 2020.

Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *ICLR*, 2017.

Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *IJCV*, 2020.

Yingwei Li, Qihang Yu, Mingxing Tan, Jieru Mei, Peng Tang, Wei Shen, Alan Yuille, and Cihang Xie. Shape-texture debiased neural network training. *arXiv preprint arXiv:2010.05981*, 2020.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars (eds.), *ECCV*, volume 8693 of *Lecture Notes in Computer Science*, pp. 740–755. Springer, 2014.

Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *PAMI*, 42(2):318–327, 2020.

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

Amil Merchant, Barret Zoph, and Ekin Dogus Cubuk. Does data augmentation benefit from split batchnorms. *arXiv preprint arXiv:2010.07810*, 2020.

Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C Duchi, and Percy Liang. Adversarial training can hurt generalization. *arXiv preprint arXiv:1906.06032*, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John P. Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *NeurIPS*, 2019.

Manli Shu, Zuxuan Wu, Micah Goldblum, and Tom Goldstein. Preparing for the worst: Making networks less brittle with adversarial batch normalization. *arXiv preprint arXiv:2009.08965*, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

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

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

Haotao Wang, Tianlong Chen, Shupeng Gui, Ting-Kuei Hu, Ji Liu, and Zhangyang Wang. Once-for-all adversarial training: In-situ tradeoff between robustness and accuracy for free. *arXiv preprint arXiv:2010.11828*, 2020.

Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. In *ICLR*, 2020.

Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. In *ICLR*, 2020.

Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L. Yuille, and Quoc V. Le. Adversarial examples improve image recognition. In *CVPR*, 2020.

Qizhe Xie, Eduard Hovy, Minh-Thang Luong, and Quoc Le. Self-training with noisy student improves imagenet classification. *arXiv preprint arXiv:1911.04252*, 2019.

Cong Xu and Min Yang. Adversarial momentum-contrastive pre-training. *arXiv preprint arXiv:2012.13154*, 2020.

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 *ICCV*, 2019.

Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Painless adversarial training using maximal principle. *arXiv preprint arXiv:1905.00877*, 2019.

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

A APPENDIX

A.1 COST ON ADDITIONAL GRADIENT COMPUTATION FOR INPUTS

During the "free" adversarial attack of the small portion of a batch, we simultaneously compute the gradient of the input tensor and the model parameters. In a standard back-propagation, we have already computed the gradients for the model parameters, and the gradient computation for the input only requires using an additional chain rule. Also, ResNet-50 has 25.6M parameters while the input has shape 224x224x3, which is 0.15M only. Thus the additional cost on gradient computation with respect to the inputs is neglectable.