Mix Saturation Attack Data like Cocktail for Robust Learning

Jian Zhao[1], **Xiangbin Liu**[2], **Shifan Lv** [3]

[1] Tsinghua University
[2] Beijing University of Posts and Telecommunications
[3] Ocean University of China
zhaojian20@mails.tsinghua.edu.cn, liuabin@bupt.edu.cn, lvshifan0@163.com

Abstract

Robust learning plays a crucial role in the application of deep neural networks, and many works have achieved impressive results. However, these methods suffer from disadvantages such as high training costs and lack of generality. Finding a low-cost general-purpose robust learning strategy is a worthwhile discussion. Data-centric robust learning is a new exploration, but there is currently little work to explore how to build a general and efficient dataset to train models with good robustness. In this paper, we propose a dataset generated by multiple attack algorithms like a cocktail, to cope with various types of attack data. Experiments demonstrate that the model trained on our dataset improves the score on the test set by 28% compared to the original training dataset. Data-centric robust learning is achieved. The source code is publicly available. [1]

Introduction

In recent years, Deep learning has occupied an important position in machine learning with its outstanding performance (Krizhevsky, Sutskever, and Hinton 2012; He et al. 2016a; Silver et al. 2017). By fitting massive amounts of data, deep learning have been widely used in security-related fields such as face recognition (Wen et al. 2016; Taigman et al. 2014), traffic recognition (Wang et al. 2017), and behavior pattern recognition (Simonyan and Zisserman 2014). However, deep learning models themselves face many security threats. For example, in the recognition problem, artificially creating some subtle perturbations on the image can make the model produce a completely different classification result for the image (Szegedy et al. 2014).

In order to build a safe and reliable deep learning system and eliminate the potential security risks of the model in practical applications, the robustness of deep learning models has attracted widespread attention.

Many methods exist for model robustness analysis and improvement, of which adversarial training remains the most effective method for defending against adversarial attacks (Goodfellow, Shlens, and Szegedy 2014; Madry et al. 2018a; Zhang et al. 2019). However, such methods have some drawbacks. Firstly, adversarial training is costly because the constant generation of adversarial samples as data in training makes model iteration slow. Secondly, adversarial training usually uses a single adversarial attack method to generate adversarial samples instead of using multiple adversarial attack methods simultaneously. The models trained in this way are difficult to defend against complex adversarial attacks (Croce and Hein 2020b). If the dataset is robust enough, a robust model can be achieved even without costly adversarial training. But how to build a common and efficient dataset to train a robust model has not been extensively explored.

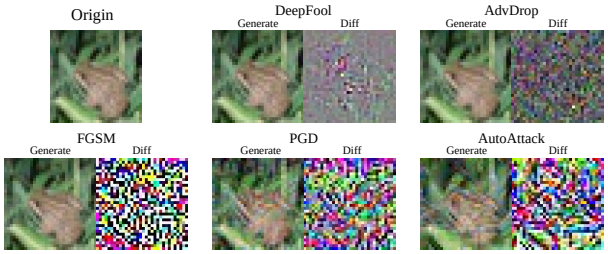

Figure 1: Adversarial samples generated by different adversarial attack methods.

In this paper, we will try to use a variety of adversarial attack methods (Figure 1) to generate adversarial samples for training, including FGSM (Goodfellow, Shlens, and Szegedy 2014), PGD (Madry et al. 2018b), DeepFool (Moosavi-Dezfooli, Fawzi, and Frossard 2016), AdvDrop (Duan et al. 2021), AutoAttack (Croce and Hein 2020b). We will also try to increase the diversity of the training data using common data augmentation algorithms (Figures 2 and 3), including Gaussian noise, SaltPepper (Lin and Yu 2006), Mixup (Zhang et al. 2018) and CutMix (Yun et al. 2019). And we will explore the optimal combination of data through extensive experiments to achieve the highest robust performance of the model on two test sets. Experiments show that our dataset significantly improves the robust performance of the model compared to the original training data, demonstrating the feasibility of data-centric robust learning.

[1] github.com/yoyo-nb/Solution_for_Data-Centric_Robust_Learning

Methods

We use some adversarial attack methods to generate adversarial samples as training data. In this section, we introduce common adversarial attack methods, including FGSM, PGD, DeepFool, AdvDrop, and AutoAttack.

FGSM and PGD

The goal of FGSM(Fast Gradient Sign Method) (Goodfellow, Shlens, and Szegedy 2014) is to increase the loss value of the model. Adversarial samples are generated by calculating the gradient of the loss function with respect to the input images and adding it to the input images:

$$x' = x + \epsilon \cdot sign(\Delta_x \mathcal{J}(x, y)), \tag{1}$$

where $\mathcal{J}(x, y)$ is the classifier cross-entropy loss function value, ϵ is a limiter that limits the maximum perturbation per pixel in the image.

And PGD(Project Gradient Descent) (Madry et al. 2018b) is an iterative attack method. It can be seen as a replica of FGSM — K-FGSM, just multiple iterations in the same direction:

$$x_{t+1} = \prod_{x+s}(x_t + \epsilon \cdot sign(\Delta_x \mathcal{J}(x_t, y))). \tag{2}$$

In general, PGD attacks are stronger than FGSM. Multiple iterations can be used to adapt the direction of gradient changes in the nonlinear model.

DeepFool

DeepFool (Moosavi-Dezfooli, Fawzi, and Frossard 2016) is a non-targed attack method with minimal perturbations based on hyperplane classification. For binary-classification, to change the classification of a sample x, the smallest perturbation is the distance of x from the classification hyperplane. For a multiclassification with classification criterion $\hat{k}(\boldsymbol{x}) = \arg\max_k f_k(\boldsymbol{x})$, the target of the attack is:

$$\arg\min_r \|\boldsymbol{r}\|_2 \quad \text{s.t. } \exists k : f_k(\boldsymbol{x}) >= f_{\hat{k}}(\boldsymbol{x}). \tag{3}$$

The optimal perturbation is obtained by finding a k such that x_0 is closest to the hyperplane distance $f_k(\boldsymbol{x}) - f_{\hat{k}}(\boldsymbol{x}) = 0$. Let $\hat{l}$ be k that satisfies this condition, then the optimal perturbation is:

$$\boldsymbol{r}_*(\boldsymbol{x}) = \frac{\left|f_{\hat{l}(\boldsymbol{x})}(\boldsymbol{x}) - f_{\hat{k}(\boldsymbol{x})}(\boldsymbol{x})\right|}{\left\|\boldsymbol{w}_{\hat{l}(\boldsymbol{x})} - \boldsymbol{w}_{\hat{k}(\boldsymbol{x})}\right\|_2^2}\left(\boldsymbol{w}_{\hat{l}(\boldsymbol{x})} - \boldsymbol{w}_{\hat{k}(\boldsymbol{x})}\right), \tag{4}$$

where $\boldsymbol{w}_i$ is the normal vector of the classified hyperplane f_i. Add $\boldsymbol{r}_*$ to the input image x to get the adversarial sample.

AdvDrop

AdvDrop (Duan et al. 2021) differs from the methods described above. It gets a whole new pattern of constructing the adversarial examples by actively losing information. AdvDrop uses DCT (discrete cosine transform) to transform the images into the frequency domain, discards the high-frequency information based on quantization table q, and then uses IDCT (Inverse DCT) to generate adversarial images:

$$x' = D_I\left(Q_{diff}(D(x), q)\right) \quad s.t. \left\|q - q_{init}\right\|_\infty < \epsilon, \tag{5}$$

where $D(x), D_I(x)$ stands for DCT and IDCT; $Q_{diff}(x, q)$ is a quantization function, A classic formula for $Q_{diff}(x, q)$ is as follows:

$$Q_{diff}(x, q) = (\phi(\frac{x}{q}) + [\frac{x}{q}]) \cdot q. \tag{6}$$

The quantization table q will be updated by with the $sign$ of gradients returned via backward propagation. Formally:

$$q' = q + sign\left(\Delta_q \mathcal{L}_{adv}\left(x', y\right)\right), s.t. \left\|q - q_{init}\right\|_\infty < \epsilon, \tag{7}$$

where ϵ is a limiter to make the resultant adversarial image x' looking indistinguishable from clean image x.

AutoAttack

AutoAttack (Croce and Hein 2020b) is a sophisticated attack algorithm that combines multiple attack methods: APGD-CE, APGD-DLR, FAB (Croce and Hein 2020a) and SquareAttack (Andriushchenko et al. 2020). PGD is a commonly used method for testing model robustness, which is computationally inexpensive and performs well in most cases. But PGD can also fail, overestimating the robustness of the model. This may be due to the fixed step size of the PGD algorithm and the extensive use of cross-entropy loss. Therefore, the Auto-PGD (APGD) algorithm is proposed, which is improved in two directions. On the one hand, the fixed step size is improved to gradient-based step size, and on the other hand, the loss function is changed. Regarding the gradient step size, momentum is added to traditional PGD:

$$z^{(k+1)} = Ps(x^{(k)} + \eta^{(k)} \nabla f(x^{(k)})), \tag{8}$$

$$\begin{aligned} x^{(k+1)} = Ps(x^{(k)} &+ \alpha \cdot (z^{(k+1)} - x^{(k)} \\ &+ (1 - \alpha) \cdot (x^{(k)} - x^{(k-1)})), \end{aligned} \tag{9}$$

where α is 0.75, Ps stands for pixel clipping operation. For changing the loss function, propose Difference of Logits Ratio (DLR):

$$DLR(x, y) = -\frac{z_y - max_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}} \tag{10}$$

where π is the order of the z components in descending order. AutoAttack combines APGD-CE, APGD-DLR, FAB, and SquareAttack to attack. Due to the diversity of components used, it can test the robustness of the model well.

Experiments

Benchmarks

- **Dataset**: We use the 10k test set of CIFAR10 (Krizhevsky 2009) to generate 50k images for training, and use various data augmentation and adversarial attack methods to obtain 130k images and 220k images respectively for testing.

Training Set	Score
10k Ori	73.12
10k Ori + 20k DeepFool + 20k AdvDrop	91.80
10k Ori + 10k DeepFool + 10k AdvDrop + 20k FGSM	92.58
10k Ori + 10k DeepFool + 10k AdvDrop + 10k FGSM + 10k PGD	94.52
10k Ori + 10k DeepFool + 10k AdvDrop + 10k AutoAttack + 10k PGD	**95.09**

Table 1: Experiment results of different adversarial attack methods for generating training sets. (Ori represents the original test set of cifa10, best result in bold)

Eps for AA	Score
0.01	95.04
0.03	**95.10**
0.1	95.09
0.2	94.60

Eps for PGD	Score
0.1	95.10
0.2	95.72
0.25	**95.73**
0.3	95.71

Table 2: Experiment results of different eps for AutoAttack(AA) and PGD. (Best result in bold)

Batchsize	Score
32	97.12
48	**97.28**
64	96.97
Scheduler	Score
MultiStep	97.28
CosineAnnealing	**97.85**

LR	Score
0.05	97.68
0.1	97.85
0.12	**97.90**
0.13	97.81
0.15	97.68
0.2	97.63

Table 3: Experiment results of different batchsize, scheduler and learning rate(LR) for SGD. (Best result in bold)

- **Models**: We first use ResNet50 (He et al. 2016a) and DenseNet121 (Huang et al. 2017) as the backbone networks to find the best training data and optimal settings for the optimizer. The robustness is then evaluated by replacing the backbone network with PreActResNet18 (He et al. 2016b) and WideResNet (Zagoruyko and Komodakis 2016).
- **Metric**: The average classification rate of the models on the test set (higher is better) is used to evaluate the robustness, which is computed by the following formula:

$$Score = \frac{1}{|\mathcal{M}|} \sum_{M_i \in \mathcal{M}} \frac{1}{|\mathcal{X}|} \sum_{(x_j, y_j) \in \mathcal{X}} \mathbf{1}\left(M_i\left(x_j\right) = y_j\right) \tag{11}$$

where $\mathcal{M}$ is the set of all trained models, $\mathcal{X}$ is the evaluation dataset.

Generate Training Set

We use multiple adversarial attack methods to generate the training set. The process of obtaining the training set is divided into four steps, which are determining adversarial attack methods, adjusting algorithm hyperparameters, adjusting optimizer hyperparameters, and replacing the original dataset. We use a test set of size 130k to calculate the scores. Finally, we obtain 50k images as the training set to achieve data-centric robust training.

SaltPepper probability = 0.1

Noise variance	Score
1	98.02
2	98.06
3	**98.08**
3.5	97.98
5	98.00

Noise variance = 3

SaltPepper probability	Score
0	**98.13**
0.05	98.10
0.1	98.08

Table 4: Experiment results of different variance for Gaussian noise and probability for SaltPepper. (Best result in bold)

Augmentation	Score
Gaussian noise	97.97
Mixup	97.36
CutMix	97.46

Table 5: Experiment results of different augmentation for the original dataset.

Determining Adversarial Attack Methods We successively use five adversarial methods to generate adversarial samples to train ResNet50 and DenseNet121. The two models are trained using AdamW (Loshchilov and Hutter 2018) optimizer with learning rate 0.01, $\beta = (0.9, 0.999)$ and batch size 128. We trained 100 epochs and decayed the learning rate to 0.1 times once the number of epochs reached 70 and 90. We first train ResNet50 and DenseNet121 using the original 10k test set of CIFAR10. Then we attack ResNet50 and DenseNet121 using different adversarial attack methods to generate half the number of adversarial samples respectively. Table 1 shows the experiment results, where the training set in the last row has the highest score. We generated 10k adversarial samples with DeepFool, AdvDrop, PGD and AutoAttack, respectively, and combined them with the original 10k test set images to obtain 50k training set images, which were used for the next experiments. Figure 1 shows the adversarial samples generated by different adversarial attack methods.

Adjusting the Algorithm Hyperparameters Next, we adjust eps for PGD and AutoAttack, and the optimizer settings are the same as in Table 1. Table 2 shows the experimental results, illustrating that the highest scores are obtained when the eps of PGD is set to 0.25 and the eps of AutoAttack is set to 0.03.

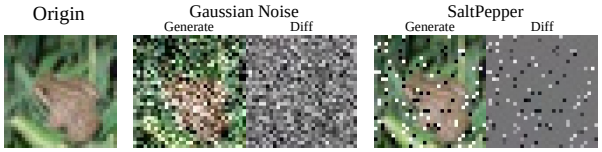

Figure 2: Augmented samples generated by Gaussian noise and SaltPepper.

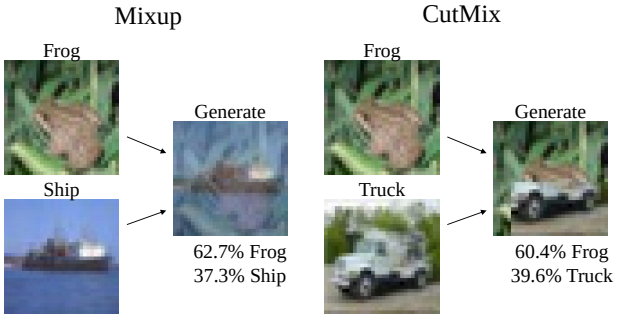

Figure 3: Augmented samples generated by Mixup and CutMix.

Adjusting the Optimizer Hyperparameters Now, we use SGD with Nesterov momentum (Sutskever et al. 2013) as the optimizer and train the model 200 epochs with a learning rate of 0.1 and MultiStepLR scheduler to select the best batchsize. Then we select the best scheduler between MultiStep($gamma = 0.1, milestones = [140, 180]$) and CosineAnnealing($T\_max = 200$) (Loshchilov and Hutter 2017), and finally we select the best learning rate. The experimental results are shown in table 3. The highest score is obtained with batchsize 48, CosineAnnealing scheduler, and learning rate 0.12.

Augmentation for the Original Dataset Finally, we augment half of the 10k original data in the training set with Gaussian noise and the other half with SaltPepper (Lin and Yu 2006), and adjust the variance of the Gaussian noise and the probability of the SaltPepper, respectively. The samples generated by the two methods are shown in Figure 2. Table 4 shows the experiment results, and we found that the highest scores are obtained when the variance of Gaussian noise is equal to 3 and the probability of SaltPepper is equal to 0 (no SaltPepper). Besides, we also tried using Gaussian noise to enhance the full 10k original data or using Mixup (Zhang et al. 2018) and CutMix (Yun et al. 2019) methods. The samples generated by the two mix methods are shown in Figure 3. Table 5 shows the results of the experiment and none of these methods scored as high as the best score in Table 4.

Further Experiments

After extensive experiments, the training data we finally used is shown in Figure 4, consisting of 40k adversarial samples generated by four adversarial attack algorithms, 5k samples generated using Gaussian noise data augmentation, and 5k original data. Table 6 shows the parameters of the optimizer. To further evaluate the robustness of the data, we use a test set of size 220k and use PreActResNet18 and WideResNet as the backbone networks to calculate the score. We used the original training set of CIFAR10 to generate 50k training data as in Figure 4. The optimizer hyperparameters were unchanged except that the learning rate of PreActResNet18 was set to 0.012. Table 7 shows the experimental results, which illustrate that our approach can greatly improve the robustness of the model.

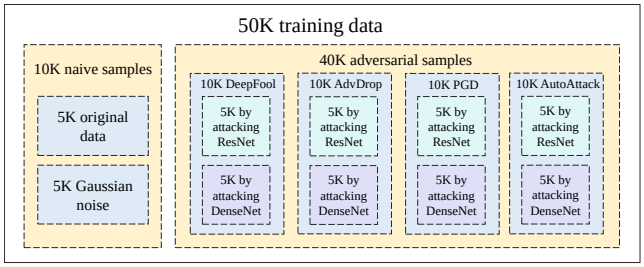

Figure 4: Composition of training data.

Config	Value	Config	Value
optimizer	SGD	nesterov	True
learning rate	0.12	batch size	48
weight decay	1e-4	epochs	200
momentum	0.9	scheduler	CosineAnnealing

Table 6: Detailed settings for the optimizer

Method	Score
50k original train set	66.21
Ours	**84.74**

Table 7: Experiment results on PreActResNet18 and WideResNet. (Best result in bold)

Conclusion

We use multiple adversarial attack methods to generate adversarial samples as training data and obtain high scores of 98 and 84 in our experiments on different backbone networks and test sets, respectively, demonstrating that our approach can achieve good robustness with a small amount of data.

During the experiment, there are more and more components of the training data, and the parameters that need to be fine-tuned are becoming more and more complex. The data composition in experiments is like a layered cocktail, which is being prepared by us using different kinds and densities of wine. Finally, there is a cup of the best flavor cocktail called "Robust Learning".

Acknowledgements

We thank the security AI challenger program "Data-Centric Robust Learning on ML Models" launched by Alibaba Group and Tsinghua University.

References

Andriushchenko, M.; Croce, F.; Flammarion, N.; and Hein, M. 2020. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*.

Croce, F.; and Hein, M. 2020a. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*.

Croce, F.; and Hein, M. 2020b. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*.

Duan, R.; Chen, Y.; Niu, D.; Yang, Y.; Qin, A. K.; and He, Y. 2021. AdvDrop: Adversarial Attack to DNNs by Dropping Information. In *Proceedings of the IEEE International Conference on Computer Vision*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision*.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. *Tech Report*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*.

Lin, T.-C.; and Yu, P.-T. 2006. Salt-pepper impulse noise detection and removal using multiple thresholds for image restoration. *Journal of Information science and Engineering*.

Loshchilov, I.; and Hutter, F. 2017. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.

Loshchilov, I.; and Hutter, F. 2018. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018a. Towards deep learning models resistant to adversarial attacks.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018b. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*.

Simonyan, K.; and Zisserman, A. 2014. Two-stream convolutional networks for action recognition in videos. *Advances in Neural Information Processing Systems*.

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. 2013. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Taigman, Y.; Yang, M.; Ranzato, M.; and Wolf, L. 2014. Deepface: Closing the gap to human-level performance in face verification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Wang, W.; Zhu, M.; Zeng, X.; Ye, X.; and Sheng, Y. 2017. Malware traffic classification using convolutional neural network for representation learning. In *International Conference on Information Networking (ICOIN)*.

Wen, Y.; Zhang, K.; Li, Z.; and Qiao, Y. 2016. A discriminative feature learning approach for deep face recognition. In *Proceedings of the European Conference on Computer Vision*.

Yun, S.; Han, D.; Oh, S. J.; Chun, S.; Choe, J.; and Yoo, Y. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018. mixup: Beyond Empirical Risk Minimization. In *International Conference on Learning Representations*.

Zhang, H.; Yu, Y.; Jiao, J.; Xing, E.; El Ghaoui, L.; and Jordan, M. 2019. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*.