An Investigation on Data Augmentation

Yu Hang Ian Jiang
ian.jiang@mail.utoronto.ca
1005415668

Salman Shahid
s.shahid@mail.utoronto.ca
1005125847

Abstract

Large convolutional neural networks are powerful but they tend to overfit the training data. One popular strategy to combat overfitting is to use data augmentation. We will examine two popular data augmentation techniques, Mixup and RandAugment, and apply them to Fashion MNIST. We will perform hyperparameter sensitivity analysis and compare their effectiveness. In the end, we combine both techniques together to improve model generalization even further.

1 Introduction

Since the introduction of convolutional neural networks, researchers have been focused on engineering more intricate model architectures to achieve state-of-the-art results. GoogLeNet (Szegedy et al. (2015)) from 2015 contains approximately 7 million trainable parameters, over 100 times more than LeNet-5 (Le Cun et al. (1989)) from 1998. With the added complexity, overfitting becomes a relevant issue. With limited training samples and highly expressive models, our models start to pickup on noise in the data, leading to strong performance on training data, but inability to generalize as well to unseen data (See Salman and Liu (2019)). In order to combat overfitting, researchers have developed their own bag of tricks, including data augmentation, weight decay, early stopping, and stochastic regularization (dropout, Srivastava et al. (2014)). Data augmentation transforms training data to improve model generatlization. The first algorithm we are analyzing, Mixup Zhang et al. (2018), combines two training samples together by linear interpolation and updates based on the loss on the mixed label. The second algorithm, RandAugment Cubuk et al. (2020), samples from a list of image transformations and applies them sequentially, to create a diverse training set. Our contributions are:

1. We extend these two techniques to a new dataset, Fashion MNIST. Both techniques were evaluated on datasets like CIFAR-10 or ImageNet, so we chose Fashion MNIST (Xiao et al. (2017)), which is in grayscale and relatively small, as a novel dataset to evaluate them on.
2. We perform sensitivity analysis on the hyperparameters and determine which approach provides better generalization results.
3. We integrate both Mixup and RandAugment to show that the test accuracy will improve even further. In doing so, we provide insights on the performance of combining multiple state-of-the-art data augmentation techniques.

2 Related Works

There are many related works on data augmentation techniques. Simple geometric approaches, including rotating, flipping, and random cropping, have seen their success in Krizhevsky et al. (2012). Alternatively, we can use a photometric approach to alter the light intensity or color mapping of the original images, known as color jittering, demonstrated in Wu et al. (2015). In recent years, there is also a surge of learned data augmentation techniques which can adapt to different learning tasks. AutoAugment (Cubuk et al. (2018)) is an automatic policy searching algorithm that uses a reinforcement learning controller and small proxy tasks to select the optimal data augmentation policy. Hataya et al. (2020) extends AutoAugment by optimizing the data augmentation policies

Preprint. Under review.

using a gradient-based approach, while Gudovskiy et al. (2021) uses scalable probabilistic implicit differentiation to improve AutoAugment's robustness for biased data with label noises. Raghu et al. (2020) extends Mixup with the use of commentaries that learn the optimal Mixup coefficient for each pair of output classes using gradient updates approximated from implicit differentiation and Neumann series outlined in Lorraine et al. (2021). Lastly, since the introduction of Generative Adversarial Networks (GAN) in 2014 (Goodfellow et al. (2020)), researchers are also able to generate additional data from a GAN trained on the training set.

3 Method and Algorithm

Our code implementation can be found at https://github.com/jiangyuhang0930/CSC413-Team-52. Most of the code is taken from the original implementation of Mixup by Zhang et al. (2018) and a PyTorch re-implementation of RandAugment.

3.1 Mixup

The first data augmentation technique we are going to examine is Mixup. Essentially, Mixup randomly takes pairs of training samples $(x_i, y_i),\ (x_j, y_j)$ to create a brand new training point:

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j$$
$$\tilde{y} = \lambda y_i + (1-\lambda)y_j$$
$$loss(y_{pred}, \tilde{y}) = \lambda loss(y_{pred}, y_i) + (1-\lambda) loss(y_{pred}, y_j)$$

where $x_i,\ x_j$ are input vectors, $y_i,\ y_j$ are one-hot target encodings, and $\lambda \in [0,1]$, $\lambda \sim Beta(\alpha, \alpha)$ for some $\alpha \in (0, \infty)$. The model is entirely trained by minimizing the loss on Mixup training samples, but Mixup is not applied to the test set. Mixup improves generalization because it encourages the model to predict linearly between examples. This linear characteristic prevents unwanted oscillations when we try to generalize outside of our training set. In the experiment section, we will show that Mixup significantly improves the prediction accuracy when compared to the baseline.

3.2 RandAugment

The second data augmentation technique we are exploring is RandAugment by Cubuk et al. (2020). RandAugment transforms each training image by consecutively applying N operations, each with M magnitude, where N and M are hyperparameters. The N operations are sampled (with replacement) with equal probability from a set of 16 transformations:

- AutoContrast	- Posterize	- Contrast	- ShearY
- Equalize	- Solarize	- Brightness	- Cutout
- Invert	- SolarizeAdd	- Sharpness	- TranslateX
- Rotate	- Color	- ShearX	- TranslateY

Each operation has its own magnitude parameter. For instance, for Cutout, the magnitude refers to how much of the image is cut out, whereas for Rotate, the magnitude refers to degree of rotation. A range is defined for the magnitude parameter of each operation, and the higher the M hyperparameter, the more the image is distorted by each transformation. Since we apply N transformations out of $K = 16$ predefined transformations, RandAugment samples from K^N variations of each training image. Given the large number of variations of each image, the training set as a whole becomes very diverse. This image diversity leads to the notable performance gains yielded by RandAugment.

4 Results and Discussion

4.1 Experiment Details

For all experiments, we used ResNet-18 to train on the Fashion MNIST dataset, which has 60000 training samples and 10000 test samples. We did a [55000, 5000] train-to-validation split to create a validation set to evaluate various hyperparameter settings. We follow the conventional data augmentation procedure by applying random crop, random horizontal flip and normalization. When using RandAugment, it is applied after the horizontal flip but before normalization. The learning rate starts out at 0.1, then is divided by 10 when training is 50% complete, and again when training is 75% complete.

Table 1: Left: hyperparameter analysis on α (Epoch = 100, Batch Size = 128). Right: hyperparameter analysis on Epoch (α = 1, Batch Size = 128). Reported values are validation accuracy at the end of training.

Description	Validation Error (%)
$\alpha = -1$ (No Mixup)	6.10
$\alpha = 0.1$	5.10
$\alpha = 0.5$	4.60
$\alpha = 1$	**4.34**
$\alpha = 2$	4.54
$\alpha = 4$	4.76

Description	Validation Error (%)
Epoch = 50	4.70
Epoch = 75	4.58
Epoch = 100	4.34
Epoch = 150	**4.26**
Epoch = 200	4.74

Table 2: Left: hyperparameter analysis on N (M = 5, Epoch = 150, Batch Size = 128). Right: hyperparameter analysis on M (N = 3, Epoch = 150, Batch Size = 128). Reported values are validation accuracy at the end of training.

Description	Validation Error (%)
No RandAugment	5.82
$N = 1$	**4.28**
$N = 2$	4.34
$N = 3$	4.6
$N = 4$	5.76

Description	Validation Error (%)
$M = 1$	4.62
$M = 2$	4.74
$M = 3$	**4.4**
$M = 4$	4.58
$M = 5$	4.6
$M = 6$	4.58

4.2 Hyperparameter Sensitivity Analysis on Mixup

From the results in Table 1, the model's performance is not sensitive to changes in α. Since λ is sampled from a $Beta(\alpha, \alpha)$ distribution, the larger that α is, the closer λ is to 0.5. This would lead to an image that equally resembles the two images that were mixed together. Setting a large α creates an aggressive Mixup strategy where the training task is much harder, requiring the model to recognize both underlying images and their respective proportions. When $\alpha = 1$, we are sampling from an uniform distribution, and when α is close to 0, λ will be close to 0 or 1, implying that Mixup image will be primarily one original training example with some hints of another one mixed in. This is more conservative, and the training task is similar to the one without Mixup. For our model, the accuracy is lowest when Mixup is not applied. As α increases up to 1, the validation accuracy also increases. This is likely due to Fashion MNIST being an easy dataset because of its small image size and single channel. On more complex datasets, if we set a large α, the model will focus on improving at a task that differs too much from the original task such that the accuracy will suffer. This is why the authors, Zhang et al. (2018), suggest that for CIFAR-10 and CIFAR-100, $\alpha \in \{0.1, 0.4\}$ works the best, and larger α leads to underfitting. In our experiment, as α increases past 1, accuracy falls. For the second experiment, we see that model accuracy is not very sensitive to epoch. As epoch increases, accuracy also increases up to 150 epochs. If we set epoch past 150, we observe that the model accuracy starts to drop in the last 10-15% of the training while training accuracy continues to increase, implying that the model is starting to overfit.

4.3 Hyperparameter Sensitivity Analysis on RandAugment

For the "number of transformations" hyperparameter N, our results (listed in Table 2) show that there is a notable improvement in performance between not using RandAugment at all, versus using RandAugment with small values such as $N = 1, 2, 3$. As we experiment with larger N, the performance boost starts to diminish, and with $N = 4$, the performance is almost the same as if we had not applied RandAugment at all. The train error rate without RandAugment approaches just 2% towards the end of training, whereas with $N = 4$, it is quite high, around 22%. This suggests that without RandAugment, the model is overfitting to the training data and on the other hand, with RandAugment with $N = 4$, the model is underfitting the training data. As hypothesized by the authors, Cubuk et al. (2020), this underfitting might be a result of low signal-to-noise ratio once we apply too many transformations to the training images. Conversely, by applying RandAugment with $N = 1, 2, 3$, we create a diverse dataset with high variance, which prevents our model from overfitting to the training set without drowning out real signals in the data. The closeness of the results for $N = 1, 2, 3$ suggest

that RandAugment is not too sensitive to the number of transformations N.
The performance of RandAugment seems to be rather robust with respect to the magnitude hyperparameter, M. For all values of M tested, the validation error is between 4.4% and 4.74%. This minimal difference is likely just attributable to variance between different experiments. While these relatively small values of M suffice for a dataset of the size of Fashion MNIST, Cubuk et al. (2020) noted that larger datasets typically required more distorted images i.e. higher magnitude M to see notable accuracy gains. We conjecture that this is because larger datasets are already naturally more diverse relative to small datasets, so small values of M do not significantly enhance dataset diversity.

4.4 Final Results and RandAugment with Mixup

Table 3: Final performance of various models on Fashion MNIST. Reported values are accuracy at the end of training. We used batch size = 128 and epoch = 150 for all models.

Model	Validation Error (%)	Test Error (%)
Baseline (Random Crop and Random Horizontal Flip Only)	5.82	6.12
RandAugment Only (N=1, M=5)	4.28	4.61
Mixup Only ($\alpha = 1$)	4.34	4.47
RandAugment with Mixup (N=1, M=2, $\alpha = 0.5$)	**3.82**	**4.39**

Lastly, we combine RandAugment with Mixup. Images go through a random sequence of transformations and then are mixed to create a diverse training set. Table 3 shows the final validation and test performance for the optimal hyperparameter combinations, chosen based on the best validation performance. We obtained the best results by combining RandAugment with Mixup. As discussed previously, RandAugment aims to improve dataset diversity whereas Mixup encourages the model to predict linearly between samples. We believe that combining both techniques yields further performance gains because they serve different purposes. For optimal results, we reduced the distortion strength of each technique (i.e. used $N = 1, M = 3$ instead of $N = 1, M = 5$ for RandAugment and $\alpha = 0.5$ instead of $\alpha = 1$ for Mixup) when applying both together. Combining both augmentations at their full strength resulted in suboptimal performance which is likely attributable to low signal-to-noise ratio due to too much distortion. From the test error curve, we can see that both methods and their combination provide a significant performance boost from the baseline. We also see that RandAugment with Mixup performs better than both RandAugment and Mixup individually as its test error is consistently lower after 120 epochs. More details on hyperparameter choice are in the Appendix.

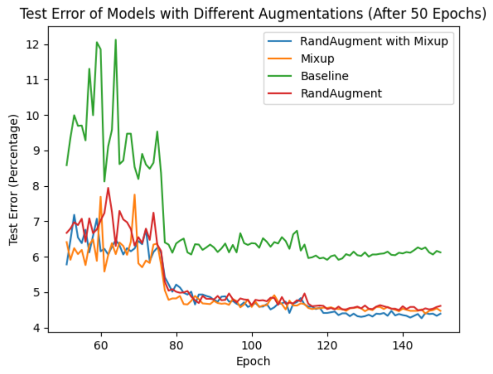

Figure 1: Test Error Curve of the Four Main Models (After 50 Epochs)

5 Conclusion

Overall, we have observed that both Mixup and RandAugment are highly effective data augmentation techniques, even when extended to a relatively small, grayscale dataset like Fashion MNIST. Further, both techniques have proven to be robust to hyperparameter changes. Our experiments have demonstrated the effectiveness of combining multiple state-of-the-art data augmentation techniques with the insight that the techniques' hyperparameters may require additional tuning when combined. We hypothesize that it is most effective to combine augmentation techniques which strive to serve different purposes. Raghu et al. (2020) and Hataya et al. (2020) extend Mixup and RandAugment, respectively, by learning the optimal parameters for each augmentation technique, using gradient approximation proposed by Lorraine et al. (2021). Future works could similarly leverage gradient approximation to learn the optimal combinations of different data augmentations for different classes and learn the parameters for each technique such that they complement each other best.

References

E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018. URL http://arxiv.org/abs/1805.09501.

E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, pages 3008–3017. IEEE, 2020. doi: 10.1109/CVPRW50498.2020.00359. URL https://doi.org/10.1109/CVPRW50498.2020.00359.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, 2020. doi: 10.1145/3422622. URL https://doi.org/10.1145/3422622.

D. A. Gudovskiy, L. Rigazio, S. Ishizaka, K. Kozuka, and S. Tsukizawa. Autodo: Robust autoaugment for biased data with label noise via scalable probabilistic implicit differentiation. *CoRR*, abs/2103.05863, 2021. URL https://arxiv.org/abs/2103.05863.

R. Hataya, J. Zdenek, K. Yoshizoe, and H. Nakayama. Meta approach to data augmentation optimization. *CoRR*, abs/2006.07965, 2020. URL https://arxiv.org/abs/2006.07965.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012. URL https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, pages 396–404, 1989.

J. Lorraine, D. Acuna, P. Vicol, and D. Duvenaud. Complex momentum for learning in games. *CoRR*, abs/2102.08431, 2021. URL https://arxiv.org/abs/2102.08431.

A. Raghu, M. Raghu, S. Kornblith, D. Duvenaud, and G. E. Hinton. Teaching with commentaries. *CoRR*, abs/2011.03037, 2020. URL https://arxiv.org/abs/2011.03037.

S. Salman and X. Liu. Overfitting mechanism and avoidance in deep neural networks. *CoRR*, abs/1901.06566, 2019. URL http://arxiv.org/abs/1901.06566.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. URL http://arxiv.org/abs/1409.4842.

R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun. Deep image: Scaling up image recognition. *ArXiv*, abs/1501.02876, 2015.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL http://arxiv.org/abs/1708.07747.

H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

Appendix

Table 4: Hyperparameter analysis on α, N, and M (Epoch = 100, Batch Size = 128) for the RandAugment with Mixup model. Reported values are validation accuracy at the end of training.

Setting	Validation Error (%)
$\alpha = 1, N = 1, M = 1$	4.10
$\alpha = 1, N = 1, M = 2$	4.16
$\alpha = 1, N = 2, M = 1$	4.34
$\alpha = 1, N = 2, M = 2$	4.40
$\alpha = 1, N = 3, M = 2$	4.46
$\alpha = 1, N = 3, M = 5$	5.04
$\alpha = 0.5, N = 1, M = 1$	4.02
$\alpha = 0.5, N = 1, M = 2$	**3.82**
$\alpha = 0.5, N = 2, M = 1$	4.28

The optimal hyperparameter choice is $\alpha = 0.5, N = 1, M = 2$.

Contributions

Ian Jiang:

1. Brainstorm for research ideas
2. Draft up 50% of research proposal
3. Completed Abstract, 50% of Introduction, Related Works, Mixup section in Method and Algorithm, Mixup portion of Results and Discussion
4. Ran experiments on Mixup models and RandAugment with Mixup models and created tables and charts
5. 50% of the Reference section
6. Some of the modification to the codebase and the GitHub repository

Salman Shahid:

1. Brainstorm for research ideas
2. Draft up 50% of research proposal
3. Completed 50% of Introduction, RandAugment section in Method and Algorithm, RandAugment and RandAugment with Mixup sections of Results and Discussion, Summary
4. Ran experiments on RandAugment models and created tables
5. 50% of the Reference section
6. Most of the modification of the codebase and the GitHub repository