Innovative: International Multi-disciplinary Journal of Applied Technology
(ISSN 2995-486X) VOLUME 02 ISSUE 06, 2024

Color-Spatial Autoaugment another Approach to Autoaugment Policies Found & Implementation

Hussien Hayder Al-Khafaji
Graduate School of Electrical and Computer Engineering, Altınba¸s University Istanbul, Turkey

Abstract:

Data augmentation is a highly effective technique for improving modern image classifiers' accuracy and has continuously improved over the years. In this paper, we propose a different approach to implementing AutoAugment policies known as "Color-Spatial AutoAugment," Our implementation utilizes the best policies dis- covered by AutoAugment for specific datasets. It categorizes them into color and spatial, thereby improving image classification accuracy. Applied to CIFAR-10, CIFAR-100, and SVHN datasets, our method significantly improved CIFAR-10 and CIFAR-100, achieving top-1 accuracies of 91.1% and 60%, respectively. These results mark a 5.54% improvement on CIFAR-10 and a substantial 7.05% increase on CIFAR-100 over the traditional AutoAugment. On the SVHN dataset, how- ever, our approach was a bit short, yielding a top-1 accuracy of 91.15% compared to AutoAugment's 93.55%. These findings highlight the potential of Color-Spatial AutoAugment and the improvement over AutoAugment within the same training conditions.

Keywords: *Computer Vision, Neural Networks, Feature Extraction, Image Classification.*

1. INTRODUCTION

Deep learning networks are highly effective machine learning systems, especially when trained with vast data. Data augmentation stands out as a robust method to enrich the quantity and diversity of data by applying a set of predefined augmentations randomly [1, 15, 23]. Within computer vision, several standard augmentation techniques can alter an image to provide a new perspective without distorting its essential content. These techniques include shifting the image by a few pixels, modifying its contrast, flipping it horizontally or vertically, or cropping a section of it [21, 22, 31].

While the machine learning and computer vision communities have dedicated considerable effort to developing advanced network architectures (e.g., [10, 11, 20, 24– 27, 32]), the exploration of enhanced data augmentation methods that introduce more invariances has not received as much attention. Take, for instance, the data augmen- tation technique used in the AlexNet network [15] unveiled in 2012; it has become somewhat of a standard for ImageNet [6], with only minor

adjustments made since its introduction.

The different invariance of an image can also be hardcoded into a convolutional network structure. However, implementing data augmentation strategies is often more straightforward than embedding these invariances into the model's architecture. Over the years, data augmentation techniques have improved, and many state-of-the-art techniques have been developed, one notable example being AutoAugment [4].

AutoAugment is designed to identify the best augmentation strategies for image classification tasks automatically. The main idea behind AutoAugment is to use a search algorithm to find effective combinations of various data augmentation transfor- mations. However, it is worth noting that even the most optimal augmentation policies discovered for a specific dataset might not perform well on others. For example, hori- zontal flipping of images during training is an effective data augmentation method on CIFAR-10 [14] but not on MNIST [7] due to the different symmetries in these datasets. While AutoAugment can effectively enhance accuracy by finding the best augmen- tation policies for a specific dataset, our implementation, Color-Spatial AutoAugment, takes this further. By applying the discovered policies more selectively and incorpo- rating color and spatial dimensions, we can tailor the augmentation to each image, achieving even more significant performance improvements.

AutoAugment was trained using a subset of CIFAR-10, called "reduced CIFAR- 10," which consists of 4000 randomly selected images. The augmentation policies found on the "reduced CIFAR-10" were then applied to both CIFAR-10 and CIFAR-100 [14] due to the similarity between both datasets [4]. The discovered policies consisted of a sequence of two or three sub-policies, where each sub-policy contains two augmenta- tion operations. These policies specify the sequence of augmentation operations to be applied to each training sample to generate an augmented version of that image; each operation within these sub-policies is associated with specific hyperparameters, such as the magnitude of the transformation.

Color-Spatial AutoAugment refined the approach by taking the sub-policies discov- ered by AutoAugment and categorizing the augmentation operations into two distinct categories: color and spatial. Each augmentation operation is listed uniquely within these categories without any repetitions. Afterward, one operation from each cate- gory is randomly selected and applied to produce an augmented image. This method ensures a diverse and unique application of augmentation techniques to enhance image training sets effectively.

The implementation initial training using a self-supervised learning approach, specifically the SimCLR method[3], on a smaller-scale network, ResNet18 [11]. The training was done on an RTX 2080Ti, utilizing a batch size of 64, a learning rate of 0.01, an SGD optimizer, and a cosine scheduler. The network was then trained for 300 epochs, and the validation was done on the same network structure with a lin- ear classifier added as the final layer for 100 epochs. The network achieved a 17.51% error rate when using Color-Spatial AutoAugment compared to AutoAugment, where the network achieved 22.52%. While for WideResNet 28-10, the error rate was 17.87% when using Color-Spatial AutoAugment compared to AutoAugment with 20.93%.

The ResNet50 network demonstrated our best performance for the final test, uti- lizing a learning rate of 0.025, following the recommended learning rate of the official SimCLR paper sections B.6 and B.7. The training was done for 1000 epochs, result- ing in an error rate of 8.9% while using Color-Spatial AutoAugment compared to AutoAugment, which had an error rate of 14.44%.

The CIFAR-100 dataset also improved performance on the ResNet50 network under the same conditions as CIFAR-10. The network achieved a 60% accuracy rate using Color-Spatial AutoAugment, compared to a 52.95% accuracy rate with AutoAugment.

While the SVHN dataset demonstrated a decrease in performance on the ResNet50 under the same

conditions used for CIFAR-10, but with SVHN-specific policies from the AutoAugment paper, the network achieved a 6.45% error rate with AutoAugment, as opposed to an 8.85% error rate with Color-Spatial AutoAugment.

Our approach was also tested using a Supervised Classification learning network, specifically WideResNet 28-10. Here, Color-Spatial AutoAugment achieved a 2.76% error rate, outperforming AutoAugment, which had a 3.09% error rate. These tests were conducted on the CIFAR-10 dataset over 320 epochs.

2. Related Work

Over recent years, numerous studies have explored the efficiency of augmentation tech- niques in image classification tasks across widely recognized datasets like CIFAR-10, CIFAR-100, MNIST, and ImageNet. One notable study by Krizhevsky et al. (2012)[15] introduced data augmentation as an effective strategy for training deep convolutional neural networks (CNNs). This approach utilized straightforward transformation tech- niques, including random cropping, horizontal flipping, and random alterations of RGB channels, markedly enhancing the CNN models' generalization capabilities on ImageNet.

Simonyan and Zisserman (2014)[24] introduced the now widely adopted VGGNet architecture, demonstrating that data augmentation, combined with a deeper CNN architecture, could yield state-of-the-art results on ImageNet. The study by He et al.

(2016)[11] presented the ResNet architecture, further underscoring the significance of data augmentation in effectively training very deep neural networks.

Justin Lemley, Saeid Bazrafkan, and Peter Corcoran introduced a technique termed Smart Augmentation[16], which enhances the performance of machine learning models by automatically generating augmented data. This is achieved by merging samples from the same class within a dataset.

Zhang et al. (2017)[29] introduced the "mixup" augmentation technique, a method that linearly interpolates between pairs of training examples and their correspond- ing labels. This technique has demonstrated enhanced robustness and improved generalization of models trained on the CIFAR-10 and CIFAR-100 datasets.

Cutout augmentation, introduced by DeVries and Taylor (2017)[8], also gained much attention as it randomly masks out square regions of input images during train- ing, forcing the model to rely on other informative regions. This technique has shown promising results on CIFAR-10, CIFAR-100, and ImageNet datasets.

The Cutout augmentation technique, introduced by DeVries and Taylor (2017)[8], has also gained much attention. It enhances training by randomly masking out square regions of input images, compelling the model to focus on other informative regions. This method has shown promising results on the CIFAR-10, CIFAR-100, and ImageNet datasets.

AutoAugment, introduced by Cubuk et al. (2019)[4], utilizes a search algorithm to automatically identify optimal augmentation policies, achieving state-of-the-art results on the CIFAR-10 and ImageNet datasets through effective transformation com- binations. Subsequent advancements include "RandAugment," proposed by Cubuk et al.[5], simplifying the augmentation process by applying random transformations directly to training data. This approach matched or surpassed the performance of the original AutoAugment but also significantly decreased the computational resources required to discover effective augmentation policies.

TrivialAugment, introduced by Samuel G. Mu¨ller and Frank Hutter (2021)[17], applies a simple approach to image augmentation. Unlike other methods that search for the best combination of augmentations and parameters, TrivialAugment randomly applies an augmentation operation to

each input image during training. This random selection is made from a predefined set of augmentation operations, such as rotation, translation, shearing, or color adjustments. Each operation is applied with a randomly chosen magnitude, ensuring a diverse set of augmented images.

Chenyu Zheng, Guoqiang Wu, and Chongxuan Li (2023)[30] explored using genera- tive models to augment data for machine learning tasks. Generative data augmentation involves creating synthetic data using models such as Generative Adversarial Networks (GANs)[9] and Variational Autoencoders (VAEs)[13]. The effectiveness of generative augmentation depends on the quality of the generated data. Poorly generated samples can mislead the training process, resulting in degraded model performance. Training generative models, especially GANs, is computationally intensive and requires signif- icant resources. The paper calls for further research to address the limitations and optimize generative data augmentation in practical applications.

3. Color-Spatial AutoAugment

Color-Spatial AutoAugment represents our proposed approach to applying the policies discovered by AutoAugment[4]. The core concept of AutoAugment revolves around employing a search algorithm to identify the best combinations of data augmentation transformations within a predefined search space.

In the original AutoAugment framework, the authors used a reinforcement learn- ing algorithm to search for the most effective augmentation policies. The search space was defined to include a variety of predefined sets of transformations, such as rota- tions, translations, shears, flips, and color changes. The algorithm explored various combinations of these transformations to discover policies that enhance performance on the target dataset. In our Color-Spatial AutoAugment approach, we want to force each policy to include at least one color transformation and one spatial transformation. Figures Figure 1 and Figure 2 illustrate the distinct batches of both augmentation methods applied to the first image in the ”Cat” category of the CIFAR-10 dataset.

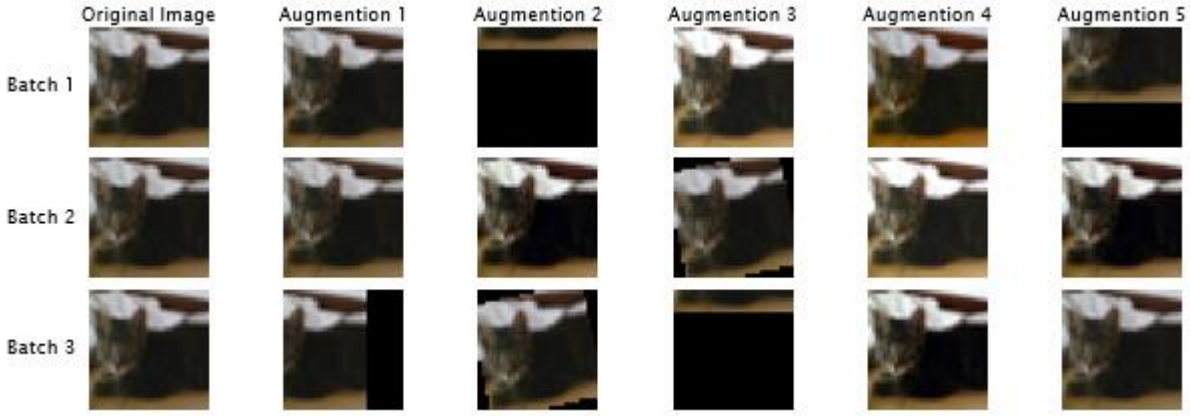

Fig. 1. A different set of batches illustrates the successful policies discovered by AutoAugment on CIFAR-10, applied randomly to each batch.

The policies discovered by AutoAugment [4] consisted of two or three sub-policies, each including two augmentation operations. These policies specify the sequence in which augmentation operations are applied to each training image, thereby generating an augmented version of that image. Each operation within the policy is associated with specific hyperparameters, such as the magnitude of the transformation, to guide the augmentation process effectively.

Color-Spatial AutoAugment organizes all discovered augmentation operations into two distinct categories: color and spatial. These categories include unique augmen- tation operations without repetition, as shown in Table A1 and Table A2. When augmentation operations share the same hyperparameters, only one is utilized in Color-Spatial AutoAugment to ensure diversity and

efficiency in the augmentation process.

Augmentations that modify the image's color, such as Posterize, Solarize, Contrast, Sharpness, Brightness, Equalize, and Invert, are classified under the color category. On the contrary, augmentations that alter the image's spatial orientation, such as ShearX,

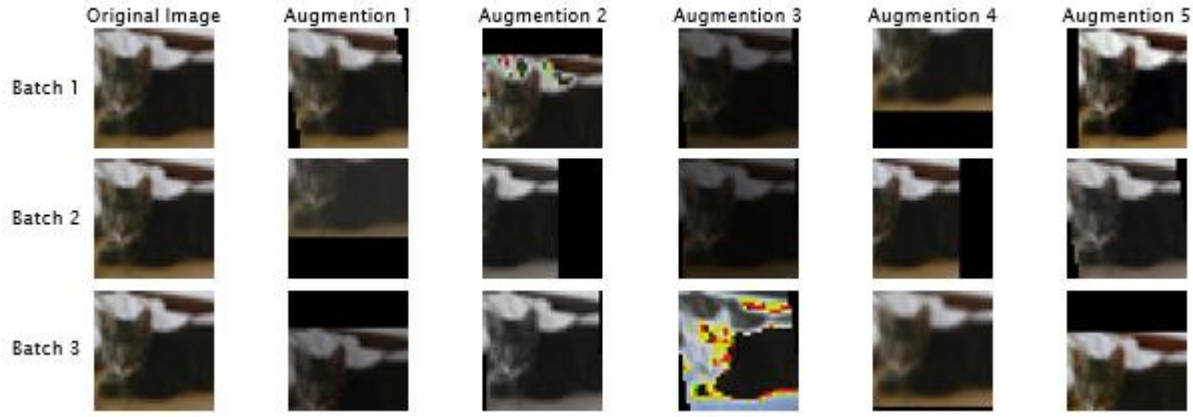

Fig. 2. A different set of batches demonstrates the random augmentations applied by Color-Spatial AutoAugment to the same image, utilizing one color and one spatial augmentation. The image used is the first in the "Cat" category of the CIFAR-10 dataset.

ShearY, TranslateX, TranslateY, Rotate, and Flip, are categorized under the spatial category.

When invoking the method to augment an image, users can select the number of augmentations to apply from each category. In our study, most tests involved apply- ing one color and one spatial augmentation, as shown in Figure 2. A random set of augmentation operations was chosen to generate an augmented version of the image, ensuring variability in the augmentation process.

4. Results

Initial testing was conducted using a self-supervised learning approach, employing the SimCLR [3] methodology on a ResNet18 [11] architecture, paired with the InfoNCE

[19] loss function set at a temperature of 0.20. The chosen optimizer was SGD [2], con- figured with a cosine annealing learning rate strategy starting at 0.01 and decreasing to a minimum of 1e-8, a momentum of 0.9, and a weight decay of 1e-4.

The training was conducted on an RTX 2080Ti, accumulating approximately 2000 GPU hours. All networks used a batch size of 64. The ResNet18 network underwent training for 300 epochs, followed by a validation phase on the same network architec- ture, with a linear classifier as the last layer for 100 epochs. Validation was carried out using 100% of the labels.

4.1. CIFAR-10 Results

4.1.1. Self-Supervised learning Results

The ResNet18 network, when augmented with Color-Spatial AutoAugment, achieved a top-1 accuracy of 82.49% on CIFAR-10, marking a 5.01% improvement over AutoAug- ment. Furthermore, Color-Spatial AutoAugment demonstrated a substantial 24.98% enhancement in performance compared to the network trained with a standard aug- mentation methodology, which includes random crop, random horizontal flip, random color jitter, random rotations, and random shear.

While the WideResNet 28-10 [28] was trained for 300 epochs as well, Color-Spatial AutoAugment achieved a top-1 accuracy of 82.13%, indicating a 3.06% improvement over AutoAugment. The various networks and their corresponding results for CIFAR- 10 are detailed in Table 1, showcasing

the performance comparison.

Table 1. The table shows the accuracy of both AutoAugment and Color-Spatial AutoAugment on the different networks; for the ResNet50, we Show the accuracy for both 300 epochs and 1000.

Epochs	Network	AA	CS-AA
	ResNet18	77.48	82.49
300	ResNet50	79.51	85.74
	WideResNet 28-10	79.01	82.13
1000	ResNet50	85.56	91.1

Afterward, we adopted the learning rate recommendation from the SimCLR paper, outlined explicitly in sections B.6 and B.7 [3], which suggests the learning rate equation $lr = 0.1 \frac{batch\ size}{256}$. Therefore, with a batch size of 64, we employed a learning rate of

0.025 for our subsequent Self-Supervised learning experiments.

0.025 for our subsequent Self-Supervised learning experiments.

The ResNet50 network underwent training and testing for 300, 600, and 1000 epochs, with each phase followed by 100 epochs of validation on 100% of the labels. Across all tests, Color-Spatial AutoAugment demonstrated improvements over AutoAugment. Specifically, at 1000 epochs, Color-Spatial AutoAugment achieved a top-1 accuracy of 91.1%, marking a 5.54% enhancement compared to AutoAugment. Figure 3 presents the validation results for both augmentation methods. Furthermore, Figure 4 illustrates the loss metrics for each method on the ResNet50 network, where, at the 1000th epoch, Color-Spatial AutoAugment recorded a loss of 0.6383, in contrast to AutoAugment's 0.7019.

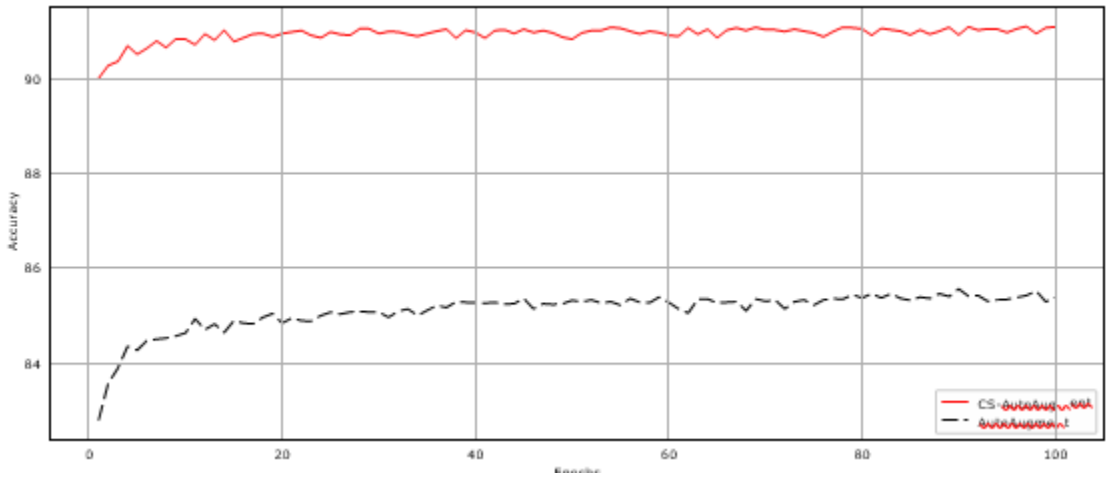

Fig. 3. The top-1 accuracy of both augmentation methods was evaluated using ResNet50 as the backbone. Validation was performed on 100% of the CIFAR-10 test dataset labels, following the network's training for 1000 epochs.

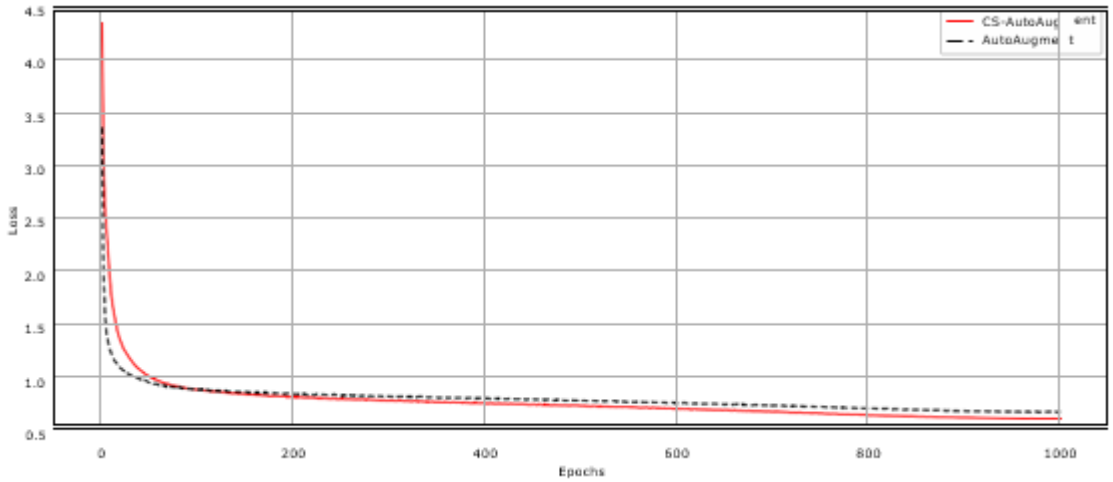

Fig. 4. The loss metrics for both augmentation methods, applied on the ResNet50 architecture over 1000 epochs, for the CIFAR-10 dataset.

4.1.2. Supervised learning Results

Supervised classification learning was conducted on the CIFAR-10 dataset using a WideResNet 28-10, with and without the implementation of Cutout[8], as shown in Table 2. The training utilized a learning rate of 0.1, a weight decay of 5e-4, a momentum of 0.9, and a gamma of 0.2. The learning rate scheduler employed was MultiStepLR, with milestones set at (20,60,120,160,220,260) epochs, and the network was trained for 320 epochs.

Using Color-Spatial AutoAugment, the network achieved an error rate of 3.17%, compared to AutoAugment's 3.31% without Cutout. However, with the inclusion of Cutout, Color-Spatial AutoAugment further reduced the error rate to 2.76%. In Figure 5 and Figure 6, we can observe the accuracy and loss metrics throughout the 320 training epochs for both augmentation methods.

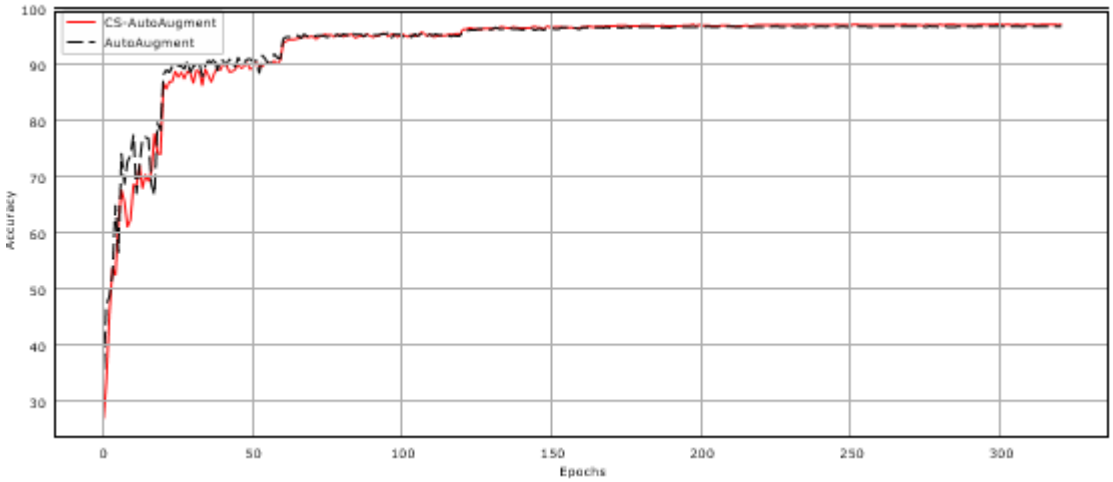

Fig. 5. The accuracy of the supervised training using WideResNet 28-10 as the backbone network for both augmentation methods on the CIFAR-10 dataset, following the network training for 320 epochs.

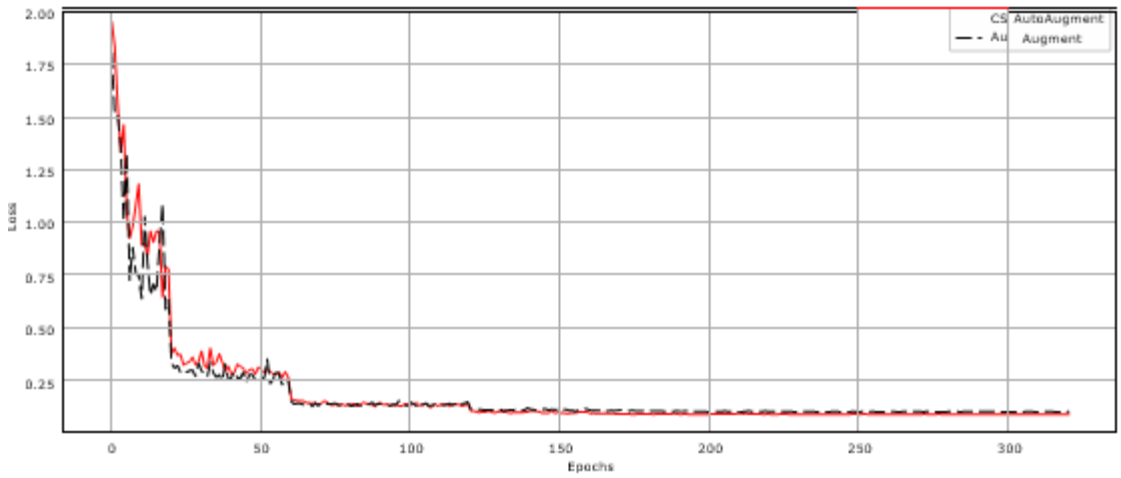

Fig. 6. The loss metrics of the supervised training using WideResNet 28-10 as the backbone network for both augmentation methods on the CIFAR-10 dataset, following the network training for 320 epochs.

Table 2. The error rates of the supervised training on CIFAR-10, using WideResNet 28-10 as the backbone network, for both AutoAugment and Color-Spatial AutoAugment methods, with and without the application of Cutout.

Model	Error Rate	Loss
Baseline	3.82	0.1576
Cutout	3.40	0.1280
AutoAugment	3.31	0.1241
Cutout + AutoAugment	3.09	0.1106
CS-AutoAugment	3.17	0.1161
Cutout + CS-AutoAugment	2.76	0.0977

4.2. CIFAR-100 Result

4.2.1. Self-Supervised learning Results

Given the similarities between CIFAR-10 and CIFAR-100, the same policies were applied to train on CIFAR-100. Employing the ResNet18 network with identical settings to those used for CIFAR-10, the network was trained for 300 epochs and underwent validation for 100 epochs. Using Color-Spatial AutoAugment, the net- work achieved a top-1/top-5 accuracy of 48.69%/79.23%, representing a 4.88%/4.47% improvement over AutoAugment.

While the WideResNet 28-10 was trained for 300 epochs as well, Color-Spatial AutoAugment achieved a top-1/top-5 accuracy of 50.56%/81.13%, indicating a 2.75%/2.19% improvement over AutoAugment. The various networks and their corre- sponding results for CIFAR- 100 are detailed in Table 3, showcasing the performance comparison.

The ResNet50 network underwent training for 1000 epochs, followed by 100 epochs of validation on 100% of the labels. Color-Spatial AutoAugment achieved a top- 1/top-5 accuracy of 60%/86.41%, marking a 7.05%/4.65% enhancement compared to AutoAugment. ?? presents the validation results for both augmentation methods. And Figure 8 illustrates the loss metrics for each method on the ResNet50 network.

Table 3. The table presents the accuracy results of AutoAugment and Color-Spatial AutoAugment across various network architectures when applied to the CIFAR-100 dataset.

Epochs	Network	AA	CS-AA
	ResNet18	43.81/74.76	48.69/79.23
300	WideResNet 28-10	47.81/78.94	50.56/81.13
1000	ResNet50	52.95/81.76	60.00/86.41

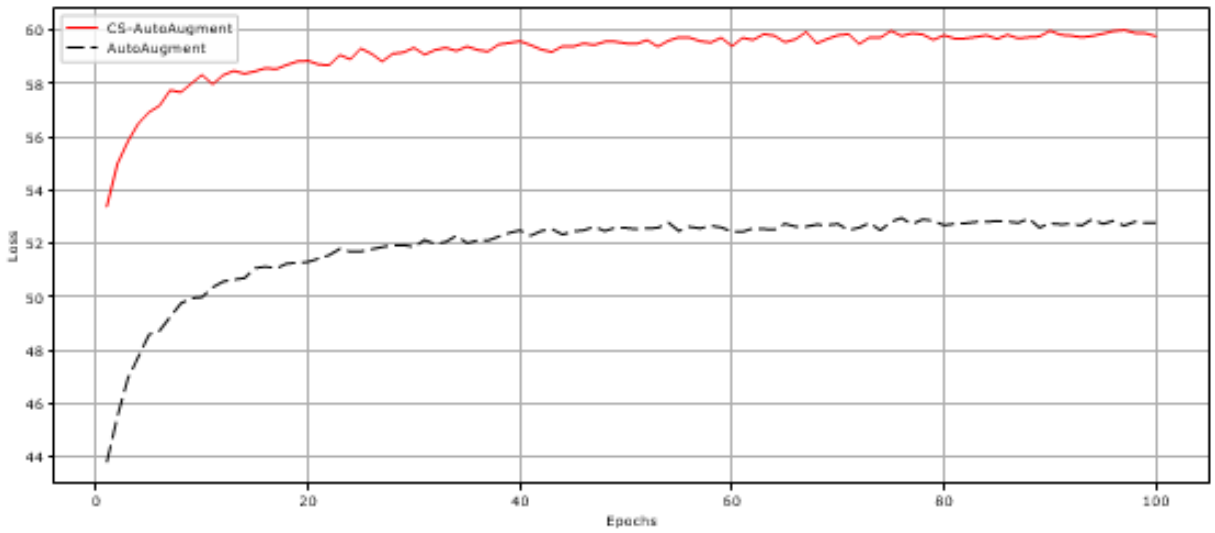

Fig. 7. The top-1 accuracy of both augmentation methods was evaluated using ResNet50 as the backbone. Validation was performed on 100% of the labels from the CIFAR-100 test dataset, following the network's training for 1000 epochs.

4.2.2. Supervised learning Results

Supervised Classification learning was conducted on the CIFAR-100 Dataset using a WideResNet 28-10; a learning rate of 0.1 was employed, weight decay of 5e-4, momen- tum of 0.9, and gamma of 0.2. The learning rate scheduler was a MultiStepLR with a milestone of (20,60,120,160) epochs and trained for 200 epochs. While using Color- Spatial AutoAugment, the network achieved an error rate of 17.87% while using Cutout compared to AutoAugment with 17.88% with the same conditions. In Figure 9 and Figure 10, we can observe the accuracy and loss of the training cycle over the 200 epochs for both augmentation methods.

Supervised classification learning was executed on the CIFAR-100 dataset using a WideResNet 28-10 architecture. A learning rate of 0.1, weight decay of 5e-4, momen- tum of 0.9, and gamma of 0.2. The learning rate scheduler used was MultiStepLR with milestones at (20,60,120,160) epochs, and the network was trained for 200 epochs. While using Color-Spatial AutoAugment with Cutout, the network achieved an error

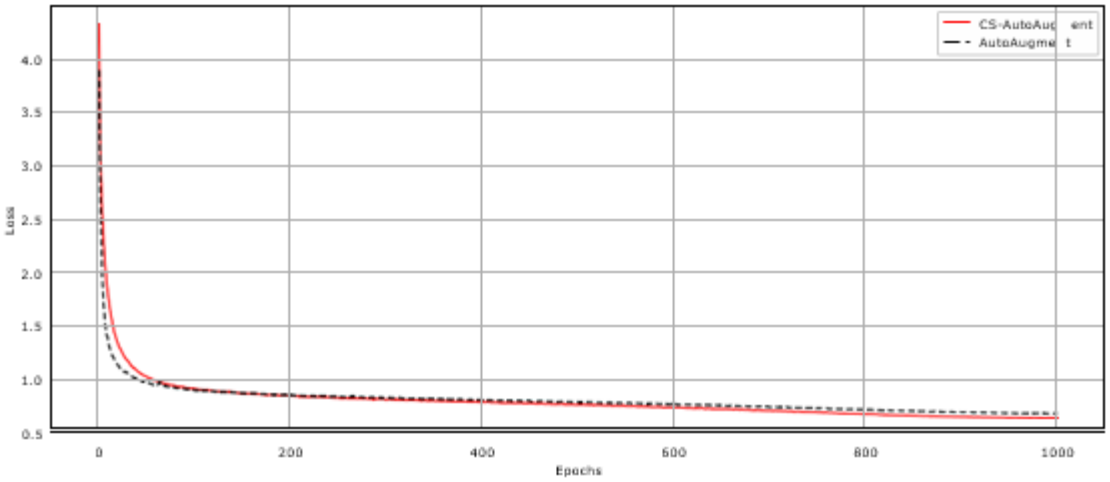

Fig. 8. The loss metrics for both augmentation methods, applied on the ResNet50 architecture over 1000 epochs, for the CIFAR-100 dataset.

rate of 17.87%, a slight improvement compared to AutoAugment's 17.88% under iden- tical conditions. Figure 9 and Figure 10 show the accuracy and loss metrics throughout the 200-epoch training cycle for both augmentation methods.

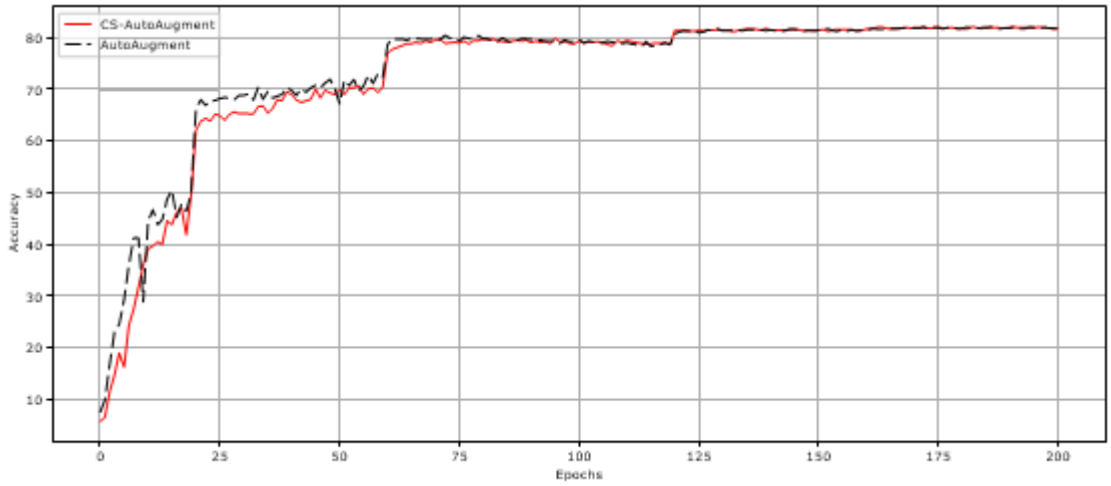

Fig. 9. The accuracy of the supervised training on CIFAR-100, using WideResNet 28-10 as the backbone network, for both augmentation methods after 200 training epochs.

4.3. SVHN Result

4.3.1. Self-Supervised learning Results

The SVHN (Street View House Numbers)[18] Dataset was tested on the ResNet50 for 1000 epochs. With the same parameters used for CIFAR-10 on the ResNet50 network, Color-Spatial AutoAugment achieved 91.15%, falling behind AutoAugment by 2.4%. In Figure 11 and Figure 12 we can observe the accuracy and loss metrics throughout the 1000 training epochs for both augmentation methods.

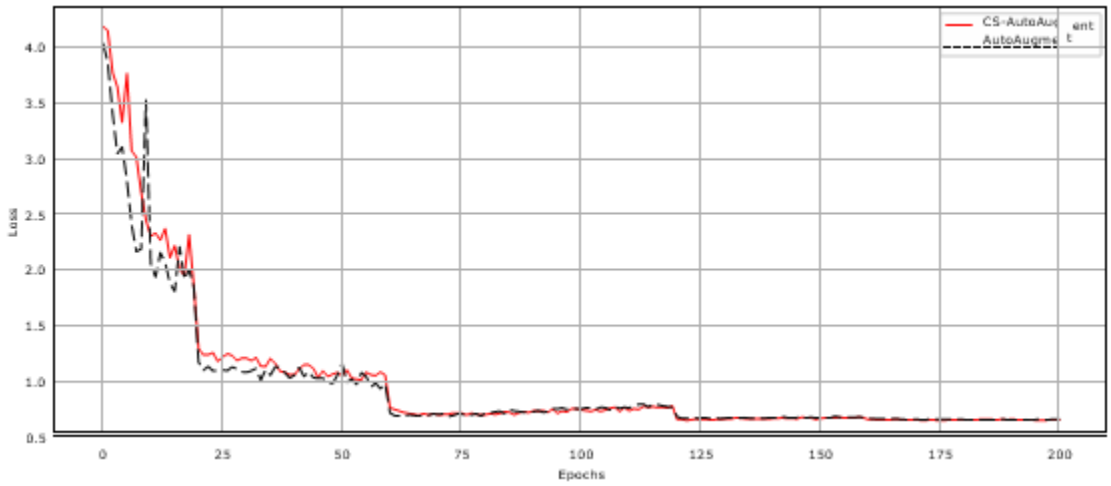

Fig. 10. The loss metrics of the supervised training on CIFAR-100, using WideResNet 28-10 as the backbone network, for both augmentation methods after 200 training epochs.

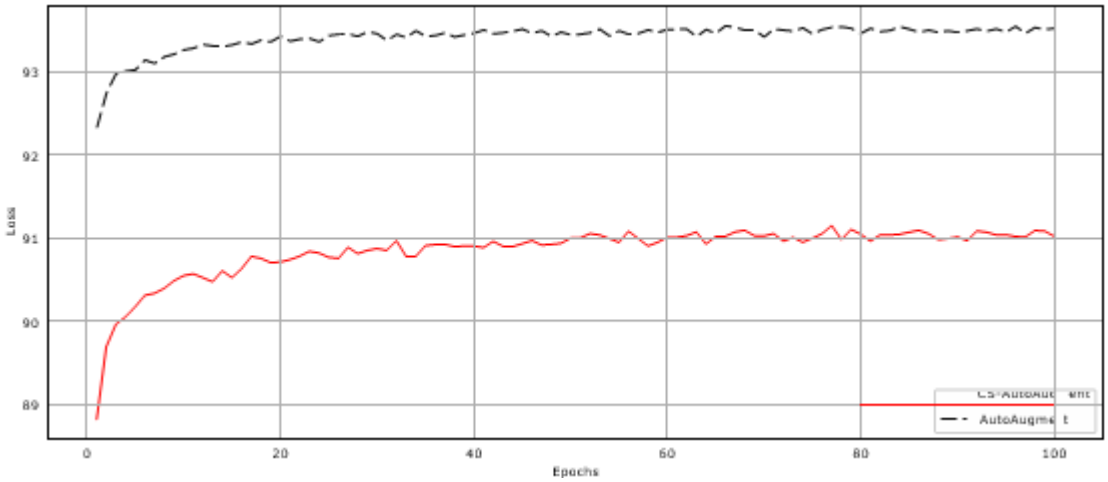

Fig. 11. The top-1 accuracy of both augmentation methods was evaluated using ResNet50 as the backbone. Validation was performed on 100% of the labels from the SVHN test dataset, following the network's training for 1000 epochs.

5. Discussion

In one of the self-supervised experiments on CIFAR-10 using ResNet50 as the backbone model, a batch size of 32 and a learning rate of 0.0125 were used. This configuration achieved a top-1 accuracy of 90.92% after 1000 epochs. Notably, this result was 0.18% lower than our best result, obtained with a batch size of 64 and a learning rate of 0.025. Therefore, further testing with larger batch sizes, such as 512 or 1024, is recommended to evaluate the impact on performance.

The implementation was further tested on the Supervised Contrastive Learning network [12], utilizing the CIFAR-10 dataset over 300 epochs with a cosine annealing learning rate set to 0.6 and an SGD optimizer. Then, the validation was performed on a linear classifier for 50 epochs. Color-Spatial AutoAugment achieved an accu- racy of 95.31%, slightly lower by 0.12%, compared to the performance achieved with AutoAugment.

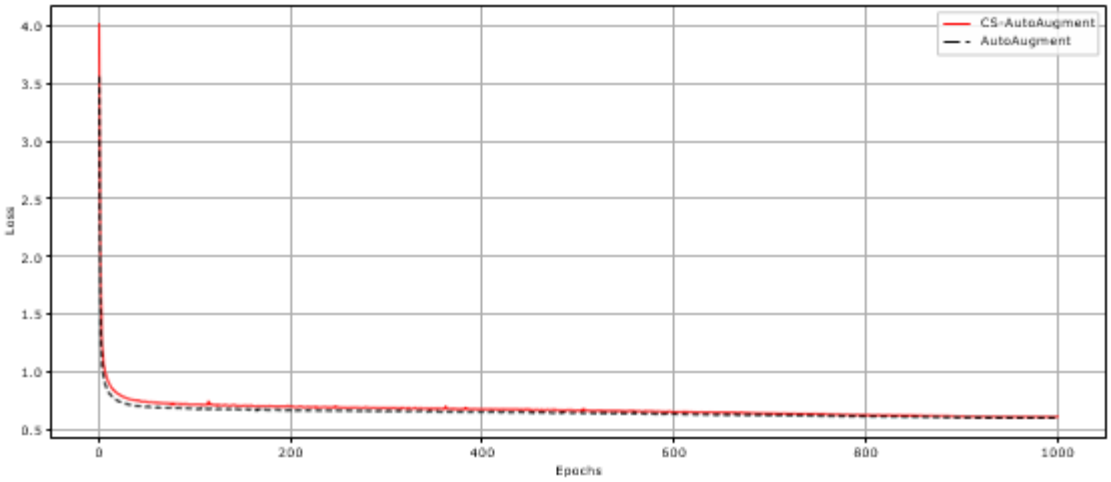

Fig. 12. The loss metrics for both augmentation methods, applied on the ResNet50 architecture over 1000 epochs, for the SVHN dataset.

The accuracy of CIFAR-100 may be improved by extending the number of train- ing epochs. The observations of the ResNet50 on both CIFAR-10 and CIFAR-100 showed significant accuracy improvement when the network underwent training for 1000 epochs. Given the larger size and complexity of CIFAR-100 and SVHN datasets, training for more epochs could be beneficial. However, it is crucial to note that an increase in epochs does not always lead to better results; there is a risk of overfitting, where the model's performance will worsen beyond a certain point in the training process.

The SVHN dataset performed a little worse with Color-Spatial AutoAugment than with AutoAugment, but the limited testing conducted on this dataset makes these results not definitive. Notably, for SVHN, the loss metric at the final epoch was 0.6110 using Color-Spatial AutoAugment, compared to a slightly lower loss of 0.6011 with AutoAugment. This differs from the results on CIFAR-10 and CIFAR-100, where Color-Spatial AutoAugment achieved lower loss values than AutoAugment.

We are also exploring the potential of two colors and one spatial augmentation to determine their effectiveness. Additionally, the possibility of applying Color-Spatial AutoAugment to other datasets, such as MNIST and ImageNet, is under consideration.

6. Conclusion

A central finding of this research is that Color-Spatial AutoAugment consistently improved the accuracy of various models, both in self-supervised and supervised learning settings. This was particularly evident in the in-depth case study involving ResNet50 trained for 1000 epochs, where Color-Spatial AutoAugment demonstrated superior performance to traditional AutoAugment and basic augmentation methods. In self-supervised learning scenarios, Color-Spatial AutoAugment led to the devel- opment of high-quality features, as evidenced by the improved performance of models in downstream tasks. This underscores the potential of Color-Spatial AutoAugment in enhancing the feature representation capabilities of neural networks.

The research highlighted the significance of a balanced augmentation approach. By integrating color and spatial transformations, Color-Spatial AutoAugment pro- vided a comprehensive augmentation strategy that improved learning dynamics and generalization.

Declarations

Conflict of Interest Statement

The author declares no conflict of interest concerning this research. The development and presentation of the ”Color-Spatial AutoAugment” model were conducted impar- tially, free from any financial, personal, or professional interests that could potentially bias the outcomes or conclusions of the study.

Data Availability Statement

The implementation code used in this research is available on GitHub at https:

//github.com/Eng-Null/Color-Spatial-AutoAugment.git. Researchers and interested parties can access the code used to train and evaluate the ”Color-Spatial AutoAug- ment” model. We encourage transparency and reproducibility in research and provide this resource to facilitate further exploration and validation of our findings. Please refer to the GitHub repository for detailed instructions on accessing and utilizing the dataset for your research purposes.

References

1. Baird HS (1992) Document image defect models, Springer, pp 546–556
2. Bottou L, Bousquet O (2010) Large-scale machine learning with stochastic gradient descent. In: Proceedings of the 19th International Conference on Computational Statistics, Springer, pp 177–186
3. Chen T, Kornblith S, Norouzi M, et al (2020) A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:200205709
4. Cubuk ED, Zoph B, Mane D, et al (2019) Autoaugment: Learning augmentation policies from data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 113–123
5. Cubuk ED, Zoph B, Shlens J, et al (2019) Randaugment: Practical automated data augmentation with a reduced search space. 1909.13719
6. Deng J, Dong W, Socher R, et al (2009) Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, Ieee, pp 248–255
7. Deng L (2012) The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine 29(6):141–142
8. DeVries T, Taylor GW (2017) Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:170804552
9. Goodfellow IJ, Pouget-Abadie J, Mirza M, et al (2014) Generative adversarial networks. 1406.2661
10. Han D, Kim J, Kim J (2017) Deep pyramidal residual networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp 6307–6315
11. He K, Zhang X, Ren S, et al (2016) Deep residual learning for image recogni- tion. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 770–778
12. Khosla P, Teterwak P, Wang C, et al (2021) Supervised contrastive learning. 2004.11362
13. Kingma DP, Welling M (2022) Auto-encoding variational bayes. 1312.6114
14. Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images. Tech. rep., University of Toronto

15. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems
16. Lemley J, Bazrafkan S, Corcoran P (2017) Smart augmentation learning an optimal data augmentation strategy. IEEE Access 5:5858–5869
17. Mu¨ller SG, Hutter F (2021) Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp 774–782
18. Netzer Y, Wang T, Coates A, et al (2011) Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning
19. Oord Avd, Li Y, Vinyals O (2018) Representation learning with contrastive predictive coding. arXiv preprint arXiv:180703748
20. Real E, Aggarwal A, Huang Y, et al (2018) Regularized evolution for image classifier architecture search. arXiv preprint arXiv:180201548
21. Sermanet P, Chintala S, LeCun Y (2013) Convolutional neural networks applied to house numbers digit classification. In: Proceedings of the International Conference on Pattern Recognition (ICPR), IEEE, pp 3288–3291
22. Shorten C, Khoshgoftaar TM (2019) A survey on image data augmentation for deep learning. Journal of Big Data 6(1):60
23. Simard PY, Steinkraus D, Platt JC, et al (2003) Best practices for convolu- tional neural networks applied to visual document analysis. In: Proceedings of International Conference on Document Analysis and Recognition
24. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Advances in Neural Information Processing Systems
25. Szegedy C, Liu W, Jia Y, et al (2015) Going deeper with convolutions. In: Pro- ceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
26. Szegedy C, Ioffe S, Vanhoucke V, et al (2017) Inception-v4, inception-resnet and the impact of residual connections on learning. In: AAAI
27. Xie S, Girshick R, Dollar P, et al (2017) Aggregated residual transformations for deep neural networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 5987–5995
28. Zagoruyko S, Komodakis N (2016) Wide residual networks. In: Proceedings of the British Machine Vision Conference
29. Zhang H, Cisse M, Dauphin YN, et al (2017) mixup: Beyond empirical risk minimization. arXiv preprint arXiv:171009412
30. Zheng C, Wu G, Li C (2023) Toward understanding generative data augmenta- tion. 2305.17476
31. Zhou Y, Guo C, Wang X, et al (2024) A survey on data augmentation in large model era. arXiv preprint arXiv:240115422
32. Zoph B, Vasudevan V, Shlens J, et al (2017) Learning transferable architectures for scalable image recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition

Appendix A The Policies used in Color-Spatial AutoAugment

A.1. CIFAR

Table A1. Color-Spatial AutoAugment policies used in training both CIFAR-10 and CIFAR-100. One of the Color sub-policies and one of the Spatial sub-policies selected at random.

Color	Spatial
'factor': color, 'value': (0.4, 3)	'factor': shear_y, 'value': (0.5, 8)
'factor': color, 'value': (0.7, 7)	'factor': shear_y, 'value': (0.2, 7)
'factor': color, 'value': (0.9, 9)	'factor': translate_x, 'value': (0.3, 9)
'factor': color, 'value': (0.2, 8)	'factor': translate_x, 'value': (0.5, 8)
'factor': color, 'value': (0.7, 0)	'factor': translate_y, 'value': (0.7, 9)
'factor': brightness, 'value': (0.6, 7)	'factor': translate_y, 'value': (0.4, 3)
'factor': brightness, 'value': (0.7, 9)	'factor': translate_y, 'value': (0.9, 9)
'factor': brightness, 'value': (0.9, 6)	'factor': rotate, 'value': (0.7, 2)
'factor': brightness, 'value': (0.1, 3)	
'factor': contrast, 'value': (0.2, 6)	
'factor': contrast, 'value': (0.6, 7)	
'factor': auto_contrast, 'value': (0.5, 8)	
'factor': auto_contrast, 'value': (0.4, 8)	
'factor': auto_contrast, 'value': (0.6, 0)	
'factor': auto_contrast, 'value': (0.8, 4)	
'factor': auto_contrast, 'value': (0.9, 3)	
'factor': auto_contrast, 'value': (0.9, 2)	
'factor': auto_contrast, 'value': (0.9, 1)	
'factor': invert, 'value': (0.1, 7)	
'factor': invert, 'value': (0.0, 3)	
'factor': invert, 'value': (0.1, 3)	
'factor': equalize, 'value': (0.9, 2)	
'factor': equalize, 'value': (0.6, 5)	
'factor': equalize, 'value': (0.5, 1)	
'factor': equalize, 'value': (0.3, 7)	
'factor': equalize, 'value': (0.2, 0)	
'factor': equalize, 'value': (0.6, 4)	
'factor': equalize, 'value': (0.6, 6)	
'factor': equalize, 'value': (0.8, 8)	
'factor': solarize, 'value': (0.5, 2)	
'factor': solarize, 'value': (0.2, 8)	
'factor': solarize, 'value': (0.4, 5)	
'factor': solarize, 'value': (0.8, 3)	
'factor': posterize, 'value': (0.3, 7)	
'factor': sharpness, 'value': (0.8, 1)	
'factor': sharpness, 'value': (0.9, 3)	
'factor': sharpness, 'value': (0.3, 9)	
'factor': sharpness, 'value': (0.6, 5)	
'factor': sharpness, 'value': (0.2, 6)	

A.2. SVHN

Table A2. Color-Spatial AutoAugment policies used in the training of SVHN. One of the Color sub-policies and one of the Spatial sub-policies selected at random.

Color	Spatial
'factor': invert, 'value': (0.2, 3)	'factor': shear_x, 'value': (0.9, 4)
'factor': invert, 'value': (0.7, 5)	'factor': shear_x, 'value': (0.7, 9)
'factor': invert, 'value': (0.9, 3)	'factor': shear_x, 'value': (0.7, 2)
'factor': invert, 'value': (0.4, 5)	'factor': shear_x, 'value': (0.1, 6)
'factor': invert, 'value': (0.9, 6)	'factor': shear_y, 'value': (0.7, 6)
'factor': invert, 'value': (0.7, 4)	'factor': shear_y, 'value': (0.9, 8)
'factor': invert, 'value': (0.9, 4)	'factor': shear_y, 'value': (0.9, 5)
'factor': invert, 'value': (0.8, 5)	'factor': shear_y, 'value': (0.8, 8)
'factor': invert, 'value': (0.6, 4)	'factor': shear_y, 'value': (0.3, 7)
'factor': invert, 'value': (0.6, 5)	'factor': shear_y, 'value': (0.8, 5)
'factor': invert, 'value': (0.8, 8)	'factor': shear_y, 'value': (0.8, 4)
'factor': invert, 'value': (0.1, 5)	'factor': translate_x, 'value': (0.9, 3)
'factor': solarize, 'value': (0.6, 6)	'factor': translate_y, 'value': (0.6, 7)
'factor': solarize, 'value': (0.2, 6)	'factor': translate_y, 'value': (0.8, 3)
'factor': auto_contrast, 'value': (0.8, 3)	'factor': translate_y, 'value': (0.0, 2)
'factor': auto_contrast, 'value': (0.7, 3)	'factor': translate_y, 'value': (0.6, 6)
'factor': auto_contrast, 'value': (0.8, 1)	
'factor': equalize, 'value': (0.6, 3)	
'factor': equalize, 'value': (0.9, 5)	
'factor': equalize, 'value': (0.6, 7)	
'factor': equalize, 'value': (0.6, 5)	
'factor': equalize, 'value': (0.6, 1)	
'factor': contrast, 'value': (0.3, 3)	
'factor': rotate, 'value': (0.8, 4)	
'factor': rotate, 'value': (0.9, 3)	
'factor': solarize, 'value': (0.4, 8)	
'factor': solarize, 'value': (0.7, 2)	
'factor': solarize, 'value': (0.3, 3)	