Exploring Data Augmentations for Neural Networks: Evaluating the Impact of PATCH SHUFFLE Data Augmentation on CIFAR-10 Classification Accuracy

Naga Sampath Maddineni
University of North Texas
Denton,Texas - 76201
NagaSampathMaddineni@my.unt.edu

Abhiram Pudi
University of North Texas
Denton,Texas - 76201
AbhiramPudi@my.unt.edu

Abstract

The implementation of deep learning models through Convolutional Neural Networks (CNNs) requires significant amounts of labeled data combined with large datasets to reach prime performance levels while minimizing overfitting occurrences. Data augmentation performs an artificial expansion of the data set through different transformation methods as a way to develop generalized leaner models with better robustness. This project investigates the effectiveness of a combined data augmentation strategy in the CIFAR-10 image classification task. We tested a basic CNN design by training it with standard normalization techniques (baseline) and combining it with additional augmentations such as crop randomization, horizontal mirror effects, color variations and random text cutting and our developed PatchShuffle technique.

1 Introduction

Nowadays the Convolutional Neural Networks (CNN's) represent the essential foundation and basics of modern computer vision because they achieve great results for tasks including image classification as well as object detection and segmentation. CNNs and comparable deep learning models need extensive labeled data as a basis for developing robust and generalizable visual understanding through their deep learning process. The process of data collection and labeling proves to be expensive and time-consuming when applied to real-world settings.

This popular image classification benchmark known as CIFAR-10 contains 60,000 color imagery with dimensions of 32x32 distributed across ten different classes. CIFAR-10 maintains popularity for method evaluation though its database consists of only modest dimensions. CNNs working with CIFAR-10 face an overfitting problem during training because they successfully recognize the training data yet poorly recognize new images.

2 Motivation

Deep learning models require regularization techniques because they easily overfit when receiving limited training data. The simplest yet most effective regularization approach comes through data augmentation. The system applies transformations that maintain labels to training images which enlarges dataset diversity and size through automated processes that do not require new data acquisition. The model must learn more generalizable features when exposed to this technique.

The most common data augmentation methods include geometric transformations with horizontal flips and random crops and color space modifications with brightness and contrast adjustments

Foundation Models for Science Workshop,38th Conference on Neural Information Processing Systems (NeurIPS 2024).

and erasing-based approaches that use cut-out and random erasing methods. Modern techniques in augmentation now include MixUp alongside CutMix while learned augmentation policies consist of AutoAugment and RandAugment. Model augmentation methods provide dual benefits of reducing overfitting along with increased model resilience on complicated problems.

3 Problem Statement

The research investigates the data augmentation technique for improving generalization performance in restricted-data CNN training. The study uses CIFAR-10 as its test platform to evaluate baseline CNN training without augmentation versus different augmentation techniques. A combination of regular transformations and the more unique PatchShuffle method makes up the augmentation process.

Analysis of learned representations along with accuracy evaluation helps show that data augmentation improves both regularization and model robustness according to empirical evidence. The research focuses on answering a fundamental deep learning practical question regarding model training when dealing with restricted datasets.

4 Related Work

The training of computer vision deep learning models depends heavily on data augmentation because it constitutes a fundamental practice. Multiple researchers have analyzed various techniques for data augmentation which holds universal recognition in this field. The works of *Weikang Xiao with Suorong Yang* present extensive guidance about augmentation techniques while research conducted by *Zhang et al* demonstrates data augmentation uses for multi-label robustness and *Goyal et al* analyzes explanation-based augmentation for interpretability growth and improved model effectiveness. Basic geometric transformations along with photometric techniques remain among the original and widespread methods used as of today. The implementation of random flipping together with cropping and rotation and scaling and translation enables models to develop position and orientation and scale invariance. Photometric enhancements that modify brightness and contrast as well as saturation and hue successfully reproduce illumination-related and camera-related changes. The widely used libraries torchvision.transforms and Albumentations provide accessible and efficient implementation of augmentation techniques for contemporary pipelines.

Techniques which explicitly regularize by removing sections of image input have emerged as popular approaches in recent times. Random Erasing and Cutout methods apply rectangle-shaped occlusions to images which prompts the model to study general patterns instead of getting fixated on detailed local features. With GridMask the concept of image masking receives a modification through grid-based systematic execution.The field has adopted two popular methods for image mixing which include Mixup alongside CutMix. Mixup unites two images and their corresponding labels by linear interpolation which results in smoother classification boundaries. CutMix by *Yun et al.* merges spatial occlusion with label mixing by exchanging image regions between two pictures to obtain training regularity and spatial consistency.

Modern approaches in augmentation systems concentrate on developing automated methods of augmenting data. The recent research on AutoAugment, Fast AutoAugment, Population-Based Augmentation (PBA) and RandAugment focuses on developing automatic augmentation policies which optimize performance for specific datasets. The search space requirements make these methods require powerful computing resources to function properly.This research utilizes established augmentation methods including Random Crop and Horizontal Flip together with Color Jitter and Random Erasing while adding our lightweight innovation PatchShuffle. PathShuffle applies computational inexpensive pixel shuffling within image patches to break down local image textures thus providing a novel regularization method. Our augmentation pipeline consists of standard techniques to augment the data which we compare to a baseline for judging its effect on generalization and classification outcomes.

5 Approach and Methodology

This section details the dataset, model architecture, augmentation strategies, training procedure, and evaluation methods used in this project.

5.1 Dataset

The experiments ran their operations on the CIFAR-10 dataset which Krizhevsky introduced in 2009. CIFAR-10 contains 60,000 color images that maintain a 32×32 pixel dimension divided into ten distinct classes including airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset distribution includes 50,000 images in the training portion and 10,000 images in the testing section. The 45,000 images made up the new training subset (90%) while the validation subset contained 5,000 images (10%). The evaluation takes place on the independent set which researchers have kept separate for testing. Standard normalization practices were applied to all images using the CIFAR-10 mean values of **(0.4914, 0.4822, 0.4465)** alongside standard deviation **(0.2023, 0.1994, 0.2010)**.

5.2 Model Architecture

We deployed SimpleCNN using PyTorch following *Paszke et al.'s* description. A series of three convolutional blocks appears sequentially before the fully connected layers. Each convolutional block includes:

- A 2D convolutional layer (nn.Conv2d) with a 3×3 kernel and padding=1
- Batch normalization (nn.BatchNorm2d)
- ReLU activation (nn.ReLU)
- The network applies max pooling (nn.MaxPool2d) using a 2×2 kernel size with a stride value of 2.

The number of output channels increases progressively across the blocks: $3 \rightarrow 32 \rightarrow 64 \rightarrow 128$. After these convolutional layers, the feature map is flattened using nn.Flatten. This is followed by two fully connected (nn.Linear) layers: the first projects the flattened features to a 512-dimensional space, applies ReLU activation and dropout (p=0.5) for regularization, and the second outputs logits for the 10 CIFAR-10 classes.

5.3 Baseline Configuration

The baseline model received input without data augmentation using the essential preprocessing operations found in the standard "torchvision.transforms" library from PyTorch. The plain transform pipeline known as 'transform_plain' underwent the following processing sequence.

- The "transforms.ToTensor()" function performs a conversion of input images either as PIL (Python Imaging Library) objects or NumPy arrays to create PyTorch tensors. The pixel values undergo normalization from the standard 0-255 range to the 0-1 scale that meets requirements for neural network training.
- After the conversion to tensor the image normalization process applies standard values from the CIFAR-10 dataset using mean = '(0.4914, 0.4822, 0.4465)' and standard deviation = '(0.2023, 0.1994, 0.2010)'. To aid training convergence Normalization makes sure that the input data has mean zero and standard deviation of one which stabilizes the models training process.

The baseline model referred to as **Experiment01** received this minimal preprocessing pipeline for its training process. The trainings in this setup did not incorporate any transformation or augmentation methods like cropping or flipping or color jittering. The training protocol of this experiment involved direct learning from raw training images which establishes this configuration as a baseline for future data augmentation assessments.

5.4 Data Augmentation Pipeline Configuration

To reduce overfitting and enhance generalization of the model a combined data augmentation method operated on training data exclusively. The pipeline which underwent the name transform_augmented consisted of both standardized and customized augmentation operations that generated meaningful diversity in training examples. The transformation sequence implemented the following sequence of steps:

1. **Random Crop with Reflection Padding:** The images received 4 pixels of reflection padding (mirrored border extension) which extended along all edges before cropping to 32×32 pixels randomly. The padded image contained a randomly selected 32×32 pixel crop. The random cropping procedure with reflection padding allows the model to learn position-independent features because it modifies object placement while adjusting image size.
2. **Random Horizontal Flip:** The augmentation procedure included horizontal flipping of images at a 50% chance rate. Random horizontal flipping represents an easy yet efficient data augmentation solution that works well with CIFAR-10 since object rotational orientation does not determine class identity (a mirrored car remains recognizable).
3. **Color Jitter:** The brightness and contrast values together with the saturation and hue parameters underwent random adjustments to the image. The brightness received an adjustment of up to 20% while contrast obtained a similar modification and saturation was adjusted by 20% but hue received Jittering of ±0.1. The model develops better color distortion resistance through these lighting adjustment and sensor noise simulation processes.
4. **Conversion to Tensor:** The PIL-based image modifications led to the conversion of the image into a PyTorch tensor through the use of 'ToTensor()'. The process transforms pixel values to range [0, 1] and arranges image channels as C×H×W for PyTorch model compatibility.
5. **PatchShuffle (Custom Transform):** The newly developed augmentation technique known as PatchShuffle operated with a 30% chance of implementation during this work. This approach splits image tensors into sections that do not overlap between each other and use 4x4 dimensions. By applying random pixel shuffles in every patch using NumPy operations the local texture becomes disrupted but the overall color composition together with spatial arrangement stays preserved at a low resolution level. Fine texture cues remain weak under this technique which affects global feature extraction by the model.
6. **Random Erasing:** The process randomly removed a 20% portion of rectangular image regions by converting their pixel values to zero. The augmentation process included erasing areas at different extents (2% to 20% of the complete image size) and through various shapes with aspect ratios from 0.3 to 3.3. The augmentation method introduces simulated obstructed areas to validate that the acquired representations are spatially stable.
7. **Normalization:** The standardized statistics of the CIFAR-10 dataset were applied for normalization of the augmented tensor. The channels received standardized values by applying the mean '(0.4914, 0.4822, 0.4465)' along with standard deviation '(0.2023, 0.1994, 0.2010)'. The model requires inputs to stay within a consistent numerical range for achieving stable training.

A precise application order was implemented where geometric and color-based transformations were executed while the image remained in PIL format to leverage optimized implementations in torchvision. The tensor-based augmentations (including PatchShuffle and Random Erasing) took place after tensor conversion while normalization occurred as the final step. The arrangement of operations provided efficient performance alongside consistent model training reactions.

5.5 Training Details

The comparison between the baseline model which operated without augmentation and the augmented model trained with the combined strategy happened under equal training conditions. The training setup together with hyperparameters and infrastructure components maintained identical conditions in all experiments as described below:

- **Training Duration:** The training period lasted 100 epochs and provided enough time for model convergence but also made overfitting visible in the baseline version. A predetermined number of epochs served to guarantee equivalent training budgets for all experiments.
- **Optimizer:** The Adam optimizer served as the model optimization algorithm because of its popularity and adaptive learning rate mechanism. Adam performs better than AdaGrad and RMSProp by gathering their benefits to speed up training while needing less learning rate adjustment. The chosen initial learning rate at **0.001** serves as a standard value for CNNs when performing vision tasks.

- **Loss Function:** The Cross-Entropy Loss served as the guidance for training through PyTorch's implementation of "nn.CrossEntropyLoss". The Cross-Entropy Loss serves as an optimal choice for multi-class image classification tasks including CIFAR-10 since images contain a single category from ten possible options. The loss function compares predicted probability distributions from logits with actual class labels to calculate differences.
- **Batch Size:** The training used **128** instances as its batch size throughout the entire process. A batch size of 128 provides an optimal combination between training efficiency and secure gradient calculations. Mini-batch training provides efficient GPU utilization while preventing GPU memory from reaching its threshold.
- **Hardware Setup:** The experiments took place on Google Colab utilizing a NVIDIA T4 GPU as hardware to speed up the training process. The training epoch time shortened substantially due to GPU hardware implementation which also enabled quick processing of large batch data along with augmentation functions.
- **Model Checkpointing and Selection:** The model performed assessments on a separate **validation set** which remained isolated during each training epoch. Model parameters representing the best validation accuracy were automatically saved to the disk. The model used the weights that delivered the best validation accuracy for conducting its final test set evaluation to demonstrate its maximum generalization potential rather than its concluding epoch effect.
- **Reproducibility:** Every run required explicit random seed definition for both PyTorch and NumPy and Python's native random module. The controlled randomization methods for data shuffling together with weight initialization and augmentation application decreased performance variability while improving the reliability of performance measurements.

Data augmentation analysis benefits from a solid foundation because of this clear and steady training procedure.

5.6 Evaluation Metrics

An evaluation process with recognized assessment metrics alongside visual diagnostics assisted in performance assessment between baseline and augmented models. The chosen metrics enabled the assessment of both final class prediction outcomes alongside monitoring how training proceeded throughout time.

- **Primary Metric — Test Accuracy:** The performance comparison depends on measuring the accuracy rate when classifying images from the CIFAR-10 test set. The test accuracy represents the number of correct predictions divided by the total 10,000 test images. The ability of a model to predict new data points correctly reflects its test accuracy performance. The metric calculation took place after training finished by applying the best validation accuracy weights from the training period. The implementation of this approach prevents performance bias that results from fitting too closely to training information and final epoch outcomes.
- **Training and Validation Curves:** The recording of training and validation accuracy along with loss values throughout 100 training iterations provided detailed understanding of the learning process. The recorded metrics were transformed into accuracy and loss graphs. Using training curves enables us to monitor convergence speed but validation curves reveal generalization abilities. The combined use of these graphical representations helps identify three main problems:
  1. The model demonstrates underfitting when it shows continuous low performance for training and validation accuracy.
  2. The occurrence of overfitting creates training accuracy growth with simultaneous validation accuracy plateau or decline.
  3. Stability of convergence (smooth vs. noisy curves).
- **Overfitting Analysis:** The difference between training and validation accuracy or loss provides essential information to determine overfitting quality. The model achieves good generalization when the training accuracy difference remains small while excessive model memorization occurs when the accuracy gap becomes large. The analysis holds significant

importance because it shows how data augmentation performs in reducing the gap between training and validation results.

6 Experiments and Results

The assessment of data augmentation impact on model generalization required us to conduct two experiments using SimpleCNN architecture with CIFAR-10 dataset.

1. **Baseline:** The training process lasted 100 epochs with basic preprocessing steps consisting of ToTensor and Normalize. No augmentation techniques were applied.
2. **Augmented:** A 100-epoch training run utilized transform_augmented as its combined augmentation pipeline that combined all geometric, photometric and custom perturbations such as PatchShuffle.

A consistent initialization of random weights was implemented between models through manual parameter reset and fixed random seed usage in all training sessions. The model checkpointing system selected the weights that produced the most accurate validation results so these weights could calculate test set performance.

6.1 Quantitative Results

The final classification accuracy on the CIFAR-10 test set after 100 epochs is summarized in Table 1.

Table 1: Final Test Accuracy on CIFAR-10 after 100 Epochs

Training Method	Train Accuracy (%)	**Test Accuracy** (%)	Training Time(MM:SS)
Baseline (No Augmentation)	98.72	**79.14**	22m 27s
With Data Augmentation	76.32	**84.53**	93m 52s

From the results the improvements observed are as follows:

- Absolute Improvement is +5.39%
- Relative Improvement is +6.81%

The model trained with data augmentation achieved a test accuracy of 84.53%, significantly outperforming the baseline accuracy of 79.14%. This reflects an absolute improvement of 5.39%, and a relative improvement of 6.81%. These results clearly demonstrate the impact of augmentation in enhancing the model's ability to generalize to unseen data.

6.2 Qualitative results(Learning Curves:

Explanation of the learning patterns and regularization aspects requires an analysis of training-validation curves from both models across their 100 epochs. Figure 1, shows the baseline model reaches almost perfect training accuracy of approximately 99% inside just a few epochs. The validation accuracy reaches its peak level at 80.56% during epoch 80 but fails to improve afterward. The training and validation performance discrepancy serves as a standard overfitting indication because the model has learned the training data exhaustively without mastering effective generalization abilities. The augmented model shows a progressive training accuracy rise because it is affected by the difficult features and increased variability from the augmentation methodology. The validation accuracy of the model keeps increasing until it reaches its highest point at 89.52%. The model shows improved generalization capabilities because its training accuracy and validation accuracy maintain a tighter alignment.

Figure 2 shows extra details through the examination of training and validation loss patterns.The baseline model's training loss moves toward zero but the validation loss increases starting from epoch-20 causing overfitting to occur. The augmented model demonstrates higher training loss because augmentation makes the model more complex yet achieves a stable validation loss level of 0.31 whereas the baseline model loses generalization resulting in validation loss exceeding 1.4. The augmentation method clearly functions as a regularization mechanism.

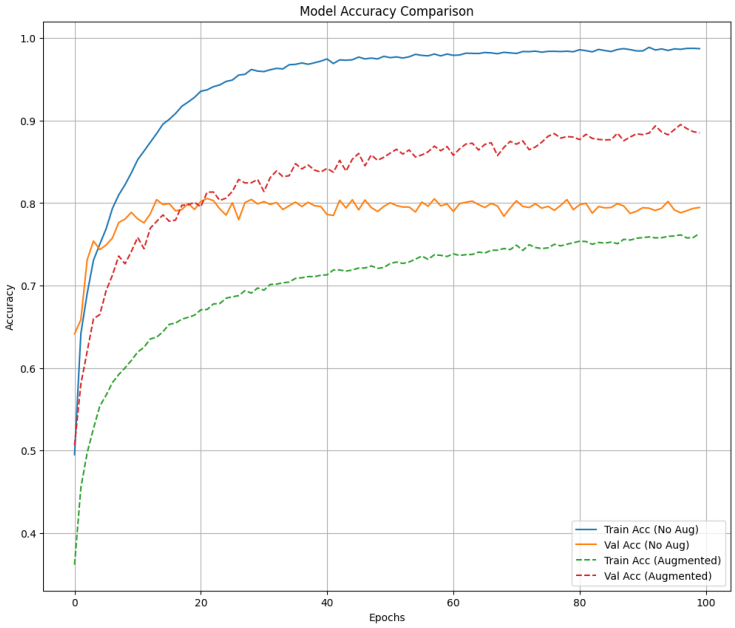

Figure 1: Model Accuracy Comparison: Training and Validation accuracy curves for models trained with and without data augmentation over 100 epochs.

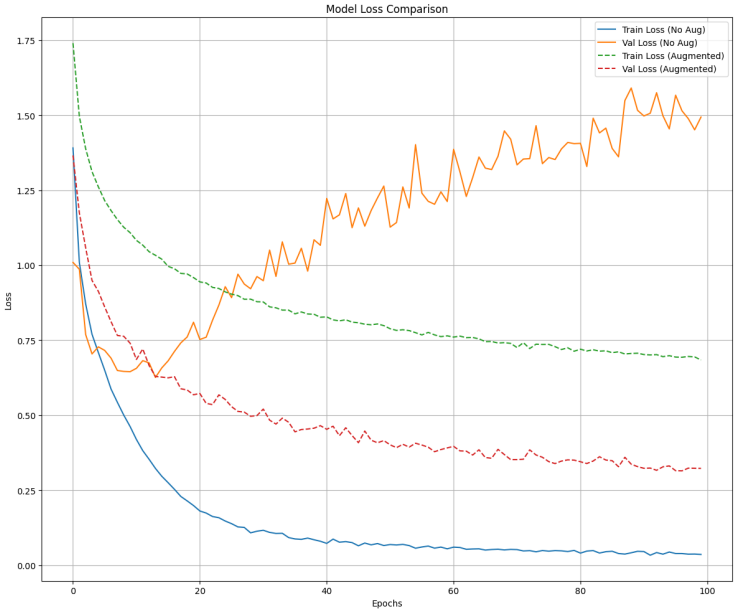

Figure 2: Model Loss Comparison: Training and Validation loss curves for models trained with and without data augmentation over 100 epochs.

7 Analysis and Discussion

The experimental evidence clearly demonstrates the significant positive influence of the applied data augmentation methodology on the performance of the ‘SimpleCNN‘ model for the CIFAR-10 classification problem. The categorical test accuracy gain of 5.39

The primary cause of the improvement is the **regularization effect** provided by data augmentation specifically to counter overfitting. The learning curves (Figures

reffig:accuracy and

reffig:loss) illustrate themselves. The baseline model, which was not augmented during training, memorized the training set instantly, which is evident in its high training accuracy and rapidly decreasing training loss. However, its validation accuracy on unseen data leveled off early, and the validation loss thereafter started increasing, indicating its failure to generalize.

In contrast, the improved model had a harder time training because random transformation was applied to every image. This is evident in its lower-increasing training accuracy and greater training loss. Through the presentation of variability in object location (RandomCrop), orientation (RandomHorizontalFlip), color (ColorJitter), potential occlusions (RandomErasing), and local texture disruption (PatchShuffle), the augmentation technique forced the model to learn more invariant and robust features. It could no longer rely on superficial features peculiar to the original training images. Therefore, the augmented model generalized more effectively, reaching higher validation and test accuracy, and having a lower, more robust validation loss than the baseline. The reduced gap between training and validation metrics for the augmented model also attests to its better capacity for generalization.

The particular mix of augmentations probably took advantage of complementary strengths. Geometric and color augmentation are typical methods to increase invariance. Random Erasing is a strong regularizer against over-relying on small, discriminative regions. The PatchShuffle introduces yet another kind of noise by disrupting local spatial correlation within small patches, perhaps forcing the model to learn from large shapes or patch-patch instead of fine textures. While the specific contribution of PatchShuffle was not unique to this work, the fact that it appears in the successful augmentation pipeline suggests that it may contribute a useful, computationally light form of regularization that is unique compared to traditional methods.

A few things are worth noting as limitations of this work. Experiments were done only on one relatively simple CNN architecture and on a single data set (CIFAR-10). The effectiveness of specific augmentation parameters (probabilities, magnitudes, patch size) was not tested by hyperparameter tuning. Also, an ablation study would be required to precisely quantify the contribution of each individual augmentation method in the multi-strategy combined. However, the results clearly affirm the overall utility of employing a diverse set of data augmentations during training.

8 Conclusion and Future Work

This project successfully demonstrated the impressive benefits of training a Convolutional Neural Network on the CIFAR-10 image classification task using data augmentation. By leveraging a combination of popular geometric, color, and erasing augmentations, along with a new PatchShuffle technique, we could observe significant gain in model performance compared to a baseline trained without any augmentation. The final test accuracy increased from 79.14% of the baseline model to 84.53% of the expanded model, with an absolute increase of 5.39%.

Learning curve analysis confirmed that data augmentation was a powerful regularizer, severely reducing overfitting and the model's ability to generalize to new, unseen data. The augmented model experienced a smaller gap between training and validation performance metrics and converged to a lower, more stable validation loss over the 100 training epochs.

This paper reconfirms the practical importance of data augmentation as a fundamental tool in the deep learning for computer vision toolkit. Even on a straightforward model and a benchmark dataset, carefully chosen augmentations can lead to dramatic performance gains.

Several avenues of future research:

- **More Powerful Augmentation Strategies:** Use and evaluate more powerful techniques like Mixup, CutMix, or pre-fixed AutoAugment or RandAugment policy with the hope to achieve better performance.
- **Hyperparameter Optimization:** Conduct principled hyperparameter optimization of the augmentation parameters (e.g., probabilities, strengths, crop ratio, PatchShuffle patch di-

mension) and the training hyperparameters (e.g., learning schedule, optimizer selection) to ensure optimal performance.

- **Ablation Studies:** Perform ablation experiments to independently isolate and measure the individual contribution of each augmentation technique used in the pipeline, particularly the homebrew PatchShuffle method.
- **Different Architectures and Datasets:** Experiment with the effectiveness of this augmentation scheme on other CNN architectures (e.g., ResNets, VGG) and other image classification datasets to ascertain its more generalizability.
- **Theoretical Understanding:** While beyond the scope of this project, more theoretical work on why particular augmentations (e.g., PatchShuffle or Mixup) are effective would be more enlightening.

Overall, data augmentation is a valuable and highly effective technique for improving CNN performance, and applying several different approaches offers a powerful approach to preventing overfitting and enhancing generalization.

References

[1] Yang, S., Xiao, W., Zhang, M., Guo, S., Zhao, J., & Shen, F. (2023). Image Data Augmentation for Deep Learning: A Survey. arXiv preprint arXiv:2204.08610.

[2] Zhang, X., Wang, Y., Li, P., Chen, J. (2024). Toward robustness in multi-label classification: A data augmentation strategy against imbalance and noise. In Proceedings of the AAAI Conference on Artificial Intelligence, 2024.

[3] Goyal, Y., Wu, Y., Bigham, J. P., Zhang, A. G. (2021). Explanation-based data augmentation for image classification. In Advances in Neural Information Processing Systems (NeurIPS), 2021.

[4] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[6] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

[7] Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2020). Random erasing data augmentation. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 07, pp. 13001-13008).

[8] DeVries, T., & Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552.

[9] Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412.

[10] Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 6023-6032).

[11] Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V., & Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 113-123).

[12] Cubuk, E. D., Zoph, B., Shlens, J., & Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops (pp. 702-703).

[13] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of big data, 6(1), 1-48.

[14] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Advances in neural information processing systems (pp. 8026-8037).

[15] Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: Fast and flexible image augmentations. Information, 11(2), 125.