Using ResNet Model for Image Classification using CIFAR10 dataset

Amaan Elahi, Karan Allagh, Mohammad Maaz Rashid
New York University
ae2950@nyu.edu, ka3527@nyu.edu, mr7374@nyu.edu
Github link

Abstract

Deep convolutional neural networks have achieved remarkable success in image classification tasks. In this project, we develop a lightweight Wide Residual Network (WRN-28-x) tailored for CIFAR-10 classification. Our architecture employs a fractional widening factor to boost representational capacity while keeping the total parameter count under 5 million. Using simple yet effective data augmentation techniques and a cosine annealing learning rate schedule, our model demonstrates a strong trade-off between computational efficiency and classification accuracy. Experimental results show that with optimized network scaling and robust training strategies, high generalization performance (nearly 95% validation accuracy) can be achieved in resource-constrained environments.

Introduction

Deep convolutional networks, notably those using residual connections, have transformed the field of image classification. Despite their success, many state-of-the-art architectures incur high computational costs, which can limit their application in settings with restricted resources. To address this challenge, we propose a lightweight variant of a Wide Residual Network (WRN-28-x) optimized for the CIFAR-10 dataset. Our model balances depth and width through a fractional widening factor, reducing computational overhead while maintaining strong performance. This work examines the impact of architectural modifications, data augmentation, and learning rate scheduling on training stability and convergence speed, providing insights into effective design choices for deployment on low-power devices.

Model Architecture

Our proposed model is based on the WRN-28-x design and includes the following key elements:

- **Residual Blocks:** The network is constructed from residual blocks that learn a residual mapping $F(x)$ with a skip connection, so that the output is given by:

$$y = F(x) + x.$$

- **Wide Convolutional Layers:** Inspired by the WRN design, the network increases the number of filters in each convolutional layer via a fractional widening factor (in our case, approximately 3.5), enabling richer feature extraction.
- **Batch Normalization and ReLU:** Each convolutional operation is followed by batch normalization and a ReLU activation, which help stabilize the training process.
- **Parameter Efficiency:** The final architecture maintains roughly 4.48 million parameters, ensuring the model is lightweight yet powerful.
- **Global Average Pooling:** Before classification, adaptive average pooling is applied to reduce the feature map dimensionality, eliminating the need for fully connected layers.

Data Augmentation Strategies

To enhance the model's generalization capabilities, we applied standard data augmentation techniques using `torchvision.transforms`:

- **Random Cropping:** Randomly extracts subregions from the input images, introducing spatial variability.
- **Horizontal Flipping:** Augments the dataset by mirroring images, promoting invariance to left-right orientation.
- **Normalization:** Standardizes pixel intensity distributions to facilitate faster and more stable convergence.

These augmentations help the model become more robust to variations in image content and positioning.

Mathematical Formulation

The core idea behind our network is to learn residual functions via skip connections. Each residual block can be expressed mathematically as:

$$y = F(x, W) + x,$$

where x is the input, W denotes the learnable weights, and $F(x, W)$ represents the transformation applied by the block (typically two convolutional layers with batch normalization and ReLU activation). In our WRN-28-x model, a sequence of such blocks allows the network to learn complex representations efficiently.

The training objective is to minimize the cross-entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i),$$

where y_i and $\hat{y}_i$ are the ground truth label and predicted probability for the ith sample, respectively. An L2 weight decay term is also added to the loss to mitigate overfitting.

Experiments and Observations

The model was trained on the CIFAR-10 dataset, which comprises 50,000 training images and 10,000 test images across 10 classes. Key aspects of our experimental setup include:

- **Training Duration:** The network was trained for 500 epochs, with detailed performance metrics recorded at each epoch.
- **Learning Rate Scheduling:** We employed a cosine annealing schedule, starting from an initial learning rate of 0.1 and gradually decaying to finer values in later epochs. This strategy facilitated both rapid initial learning and precise fine-tuning.
- **Data Augmentation:** Simple augmentations (random crop and horizontal flip) proved effective in boosting generalization, as evidenced by the steady decline in validation loss.

Analysis of the training log reveals:

- **Early Stages (Epochs 1–10):** The validation accuracy quickly rose from approximately 15% to 62–65%.
- **Mid Training (Epochs 11–25):** The validation accuracy continued to improve, reaching roughly 73.66% by epoch 15 and about 81.28% by epoch 25.
- **Late Stages (Epochs 300–397):** The model achieved over 93% validation accuracy around epoch 300, ultimately approaching nearly 95% by epoch 397. Despite moderate training accuracy (around 60%), the high validation accuracy underscores the effectiveness of our augmentation and learning rate strategies.

Figures 1 and 2 illustrate the training dynamics.

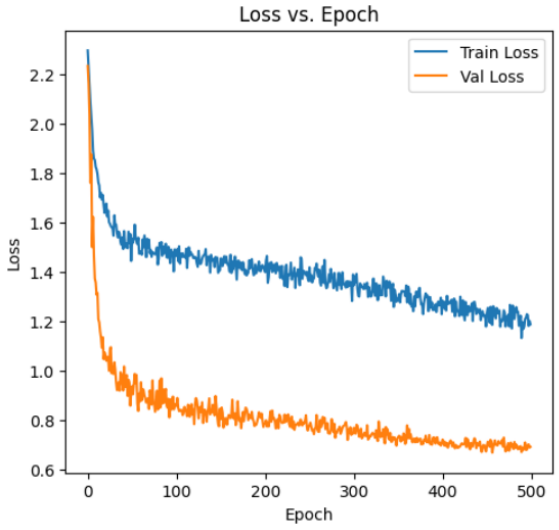

Figure 1: Training and test loss vs. epoch. The smooth downward trend indicates effective optimization.

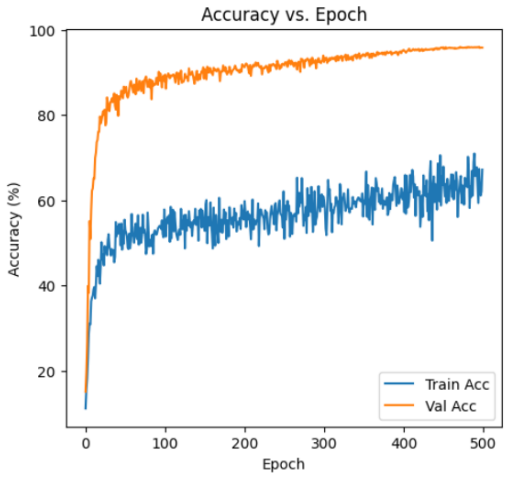

Figure 2: The training and test accuracy across epochs for Narrow ResNet-50 on CIFAR-10, demonstrating that strong regularization helped maintain high test accuracy.

Conclusion

This project demonstrates that a lightweight Wide Residual Network, when optimized through careful network scaling and effective data augmentation, can achieve competitive performance on CIFAR-10. The WRN-28-x model, with its balanced depth and width and a parameter count under 5 million, not only achieves nearly 95% validation accuracy but also maintains computational efficiency. These results underscore the potential for deploying deep learning models in resource-constrained environments without sacrificing classification performance. Future work may explore further refinements in parameter optimization, additional augmentation strategies, or alternative learning rate schedules to push the limits of efficiency and accuracy.

References

[1] https://arxiv.org/pdf/2010.01412 - Deep Learning Optimization Techniques

[2] 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 (CVPR). https://arxiv.org/pdf/1512.03385

[3] Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-Excitation Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://arxiv.org/pdf/1709.01507

[4] Foret, P., Kleiner, A., Mobahi, H., & Neyshabur, B. (2021). Sharpness-Aware Minimization for Efficiently Improving Generalization. In ICLR. https://github.com/davda54/sam

[5] Moskomule. SAM PyTorch Implementation. https://github.com/moskomule/sam.pytorch

[6] Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., & Yoo, Y. (2019). CutMix: Regularization Strategy to Train

Strong Classifiers with Localizable Features. ICCV. https://arxiv.org/pdf/1905.04899

[7] Zhang, H., Cisse, M., Dauphin, Y.N., & Lopez-Paz, D. (2018). Mixup: Beyond Empirical Risk Minimization. ICLR. https://arxiv.org/pdf/1710.09412

[8] PyTorch Vision: Data Augmentation Examples. https://pytorch.org/vision/main/auto_examples/transforms/plot_cutmix_mixup.html

[9] Cubuk, E. D., Zoph, B., Shlens, J., & Le, Q. V. (2019). AutoAugment: Learning Augmentation Policies from Data. CVPR. https://arxiv.org/pdf/1805.09501

[10] Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic Gradient Descent with Warm Restarts. ICLR. https://arxiv.org/pdf/1608.03983