Deep Learning Project 1: Optimizing ResNet for CIFAR-10

Team SPAR (2.3M): Spandan Rout, Aryan Mamidwar
Github Codebase

sr7729@nyu.edu, arm9337@nyu.edu

Abstract

We present an Improved ResNet for the CIFAR-10 dataset. The model achieved 86.968% test accuracy with 4,903,082 parameters. Using LeakyReLU, dropout and robust augmentations. We optimized it with SGD which is enhanced with Lookahead and cosine annealing scheduler. This design tries to balance efficiency and performance, with training stabilized with Lookahead.

Introduction

As we know that ResNet uses skip connections to train deep networks effectively. For this project/competition we optimized a ResNet version for the CIFAR-10 dataset keeping it under 5 million parameters. This report shoes our methodology, results and insights aided by architectural and training visualizations.

Methodology

Architecture Design

Our ImprovedResNet model uses BottleneckBlock units: 1*1 (dimension reduction), 3*3 (spatial processing) and 1*1 (expansion to outchannels * 4) convolutions.
Skip connections mitigate vanishing gradients, with downsampling aligning dimensions.
The network starts with an initial 3*3 convolution that transforms 3-channel inputs into 32-channel inputs, followed by a batch normalization and LeakyReLU activation. The model builds three residual layers (layer1, layer2, layer3) with increasing channel widths (64→256, 128→512, 256→1024). Stride of 2 in layers 2 and 3 reduce spatial size, while dropout layers are inserted after each layer to prevent overfitting. The model is then concluded with adaptive pooling and a 1024→10 fully connected layer.
Total parameters: 4,903,082

Training and Optimization

We trained the model with Stochastic Gradient Descent (SGD) with hyperparameters lr = 0.05, momentum = 0.9, weight_decay = 1e-4.

This was then wrapped by Lookahead (k = 5, aplha = 0.5). This updates the slow weights every 5 steps using:

$$slow = slow + 0.5 * (fast - slow)$$

We also used a CosineAnnealingLR scheduler with T_max = 400 which gradually decreases the Learning Rate.
Loss used was CrossEntropyLoss with label smoothing (0.1).
This training ran for 400 epochs on the NYU High Performance Computing.

Data Augmentation

- RandomCrop (32, padding = 4)
- RandomHorizontalFlip
- RandomRotation (15°)
- ColorJitter (brightness = 0.2, contrast = 0.2, saturation = 0.2, hue = 0.1)
- RandomErasing (p = 0.5, scale=(0.02, 0.33), ratio=(0.3, 3.3))
- Normalization to CIFAR-10 means/std.

Lessons Learned

LeakyReLU vs ReLU: In the preliminary trials it was observed that LeakyReLU improved training performance compared to ReLU.
Dropout Tuning: We also observed that a higher dropout value (>0.5) reduced accuracy, whereas the range of 0.1 - 0.3 was optimal.
Channel Width: We tried increasing the initial channels beyond 32, increased the parameters but did not show any proportional gains.
Bottleneck: This saved us the number of parameters, but required careful width scaling to retain the proper feature capacity.
LR Fine Tuning: We tried different lr schedulers and values. Those did not give any proportinal gains.

Results & Discussion

Our model achieved 86.968% test accuracy and 98.67% train accuracy after 400 epochs with 4,903,082 parameters. Figure 2 shows the accuracy curve. Figure 3 shows the loss curve. Figure 4 shows the Learning Rate decay, aligning with performance gains.

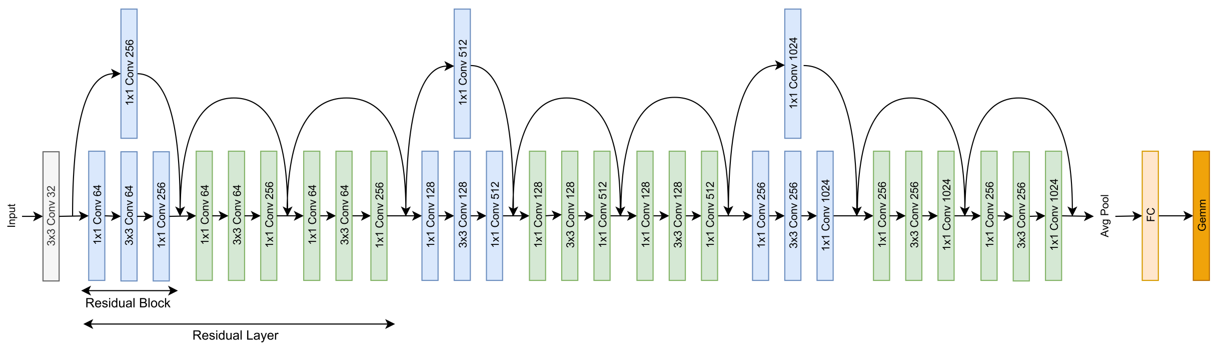

Figure 1: ImprovedResNet Architecture

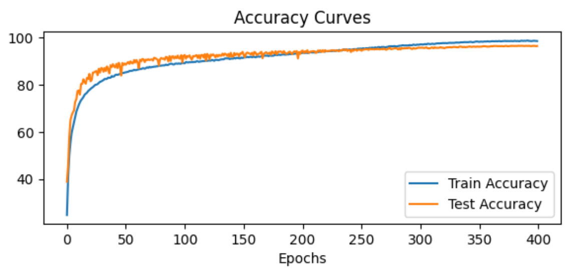

Figure 2: Training and Testing Accuracy

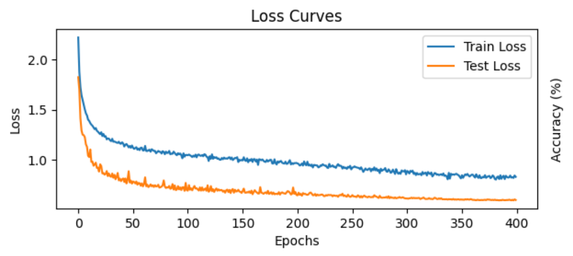

Figure 3: Training and Testing Loss

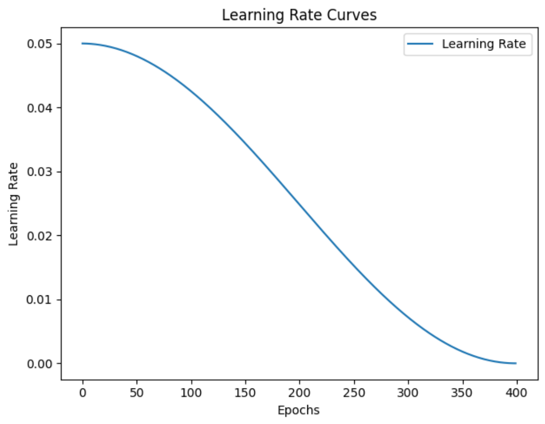

Figure 4: Learning Rate Scheduler

References

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Bach, F.; and Blei, D., eds., *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 448–456. Lille, France: PMLR.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In Dasgupta, S.; and McAllester, D., eds., *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, 1310–1318. Atlanta, Georgia, USA: PMLR.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958.

Xue, F.; Shi, Z.; Wei, F.; Lou, Y.; Liu, Y.; and You, Y. 2021. Go Wider Instead of Deeper. *CoRR*, abs/2107.11817.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. *CoRR*, abs/1605.07146.

Zhang, M. R.; Lucas, J.; Hinton, G. E.; and Ba, J. 2019. Lookahead Optimizer: k steps forward, 1 step back. *CoRR*, abs/1907.08610.

Disclousre

We have taken inspiration from the assignments/labs of another course (Intro to ML @ NYU). We have also taken inspiration of the above cited papers, articles, code bases. We have also used some LLM models to understand more about the project (ChatGPT, Grok, Perplexity). We have also used some more online resources like StackOverflow, official documentation of the packages used.