RESNET MODEL FOR CIFAR 10

Navya Kriti[1], Cody Wang[1], Nitisha Shetty[1]
[1]New York University

nk3696@nyu.edu, cw3232@nyu.edu, ns6108@nyu.edu

Abstract

Deep Convolutional Neural Networks (DCNNs) have set the benchmark for various computer vision applications. Nonetheless, the size of these state-of-the-art networks often renders them impractical for edge computing due to their extensive resource requirements. In this project, we propose a straightforward strategy to engineer networks with a fewer number of trainable parameters, thereby diminishing their memory requirements:

Through our empirical studies on the CIFAR10 dataset with scaled-down SEResNet architectures, we demonstrate that it is possible to maintain commendable accuracy despite a reduction in parameters.

Introduction

Our objective was to get a balance between the intricacy of models and their computational demands. This led us to evaluate two modified ResNet architectures on the CIFAR-10 dataset, each designed to reduce parameters while potentially increasing accuracy.

The first strategy within our ResNet framework uses a shared parameter technique across multiple residual layers of the network [1], enriching the model without sacrificing depth. The second method integrates Squeeze-and-Excitation (SE) blocks [3] into a streamlined version of the standard ResNet structure [5] with less than five million parameters.

Both ResNet variations are built around the BasicBlock [5], comprising convolutional layers and batch normalization, with dropout included as a discretionary feature. SE blocks [3] are strategically added to help in channel-wise feature recalibration.

Our findings show that the shared-parameter ResNet model reached a notable 93.7% accuracy, while the SE-enhanced ResNet [2], fine-tuned with precise hyperparameters, achieved a 95.6% accuracy.

The complete source code for both models is available for review at our **GitHub repository**: https://github.com/Navya0203/RESNET-ON-CIFAR-10/.

Methodology

Dataset: CIFAR-10

In our approach to training a model on the CIFAR-10 dataset using PyTorch, we've applied a series of data augmentation techniques to the training dataset to enhance the model performance. Specifically, our augmentation pipeline includes Random Resized Crop, which adjusts the scale and aspect ratio of images; Random Rotation, which randomly rotates images by up to 10 degrees to introduce rotational variance; Random Crop with padding, which crops images at random locations after padding them, thereby varying their spatial context; Random Horizontal Flip, which mirrors images horizontally to simulate directional variability; and Color Jitter, which slightly alters the brightness, contrast, saturation, and hue of images to mimic different lighting conditions and color settings.

In addition, we normalized the images by applying a fixed normalization. This standardization uses predefined mean and standard deviation values for the RGB channels (0.4914, 0.4822, 0.4465 for means and 0.2023, 0.1994, 0.2010 for standard deviations, respectively), ensuring that the input data has a consistent scale and distribution.

For the test dataset, we only apply the normalization. This ensures that the model's performance is assessed on data processed similarly to the training data's normalization, providing a clear measure of its ability to generalize from the augmented training data to new, unseen images.

Dense SE Resnet with Shared Parameter

Architecture

In the SE paper, SE-ResNet-110 and SE-ResNet-164 were adapted for CIFAR datasets with each, consisting of 18 and 27 convolutional layers per stage. In our adaptation, we reduced the depth per stage but increased channel counts across the three stages to 64, 128, and 256, keeping the depth of each stage to 6 blocks. This configuration increases the total number of trainable parameters. To explore the impact of parameter efficiency, we conducted experiments with

kernel sharing across different stages. We found that sharing kernels in the first stage, where fewer parameters exist, offers limited benefits. Thus, we experimented with three configurations: sharing kernels across all stages, only in the second and third stages, and exclusively in the third stage, to determine the most effective strategy for balancing network depth with parameter count. We found that our model worked best with only third-stage kernel sharing [1].

Optimizer and Learning Rate Choice

In evaluating optimization strategies for the Dense SE Resnet [1], we compared Adam [4] and Stochastic Gradient Descent (SGD) [6]. Despite Adam's capabilities with adaptive learning rates and handling sparse gradients [4], it underperformed in our dense network architecture, delivering lower accuracy of 85.5%. We attribute this to its less predictable behavior in complex parameter spaces and sensitivity to hyperparameter settings.

Conversely, SGD [6], set with an initial learning rate of 0.01, momentum of 0.9, and weight decay of 5e-4, demonstrated superior performance by achieving an accuracy of 93.83%. Its success is credited to the effective use of momentum, which accelerates convergence along relevant directions and stabilizes updates, and weight decay, which helps prevent overfitting.

For both Adam [4] and SGD [6], we implemented the MultiStepLR scheduler, reducing the learning rate by a factor of 0.1 initially after the first 85 epochs and subsequently every 25 epochs. This strategic reduction in learning rate was crucial in optimizing the training dynamics, enhancing the performance of SGD [6] significantly and attempting to mitigate the limitations observed with Adam [4].

Basic SE Resnet

Architecture

The architecture of the ResNet model implemented in the provided code features three residual layers, each configured with a specific number of BasicBlock [5] units as specified by the num_blocks array [4, 4, 3] . Each BasicBlock within these layers utilizes convolutional layers with kernel sizes defined by the conv_kernel_sizes array, consistently set at 3x3 across all layers [5]. Additionally, the network employs skip connections with a kernel size of 1x1. The model begins with an initial convolution layer that increases the channel size from 3 to 64. The subsequent layers progressively double the number of channels from 64 to 128 and finally to 256, enhancing the network's ability to capture increasingly complex features at each stage. This setup, combined with optional squeeze-and-excitation blocks [3] that further refine the feature maps through adaptive recalibration, allows for a robust network capable of achieving good accuracy.

Optimizer and Learning Rate Choice

In optimizing the Basic SE Resnet [2], we leveraged Stochastic Gradient Descent (SGD) [6] for its robustness and reliability, as demonstrated in previous experiments with the Dense SE Resnet. To enhance SGD's performance, we integrated the Lookahead optimizer, which improves optimization efficacy and convergence stability. Lookahead operates by alternating updates between two sets of weights: fast weights that update regularly, and slow weights that synchronize with the fast weights every five iterations (k=5). This synchronization helps smooth updates and navigate suboptimal local minima more effectively [8].

Additionally, we utilized a CosineAnnealingLR scheduler that dynamically adjusts the learning rate from an initial rate of 0.1 down to 0.001, following a cosine curve. This approach fine-tunes the learning rate as the training progresses toward optimal solutions. We also incorporated gradient clipping [7] with the optimizer to prevent the exploding gradient problem, ensuring that updates remain within reasonable bounds and contribute to stable training.

The combination of SGD [6], Lookahead, gradient clipping, and the CosineAnnealingLR scheduler effectively raised the model's accuracy to 95.6%. This strategy highlights the benefits of hybrid optimization techniques, enhancing training dynamics and overall performance in the complex architecture of the Basic SE Resnet.

Discussion

We undertook a comparative study of two modified ResNet architectures tailored for the CIFAR-10 dataset, each built to improve accuracy. The first model uses a shared-parameter approach [1] within the BasicBlock classes [5] and incorporates Squeeze and Excitation (SE) blocks [3], enabling it to preserve its depth while maintaining a constant parameter count. The second model embeds SE blocks [3] into a more streamlined version of the traditional ResNet framework without any parameter sharing, thus improving its ability to recalibrate features more efficiently while containing less than five million parameters.

Both models are built on the BasicBlock foundation [5], featuring convolutional layers and batch normalization. The integration of SE blocks [3] across both models aims to improve the recalibration of channel-wise features. Performance assessments reveal that the shared-parameter ResNet

model [1] achieved an accuracy of 93.7%, while the SE-enhanced ResNet model [2] , meticulously fine-tuned with optimal hyperparameters, reached an impressive 95.6% accuracy.

Methodologically, we experimented with various data augmentation techniques to boost performance, including Random Resized Crop, Random Rotation, Random Crop, Random Horizontal Flip, and Color Jitter, coupled with a uniform normalization approach. This training methodology enhanced the robustness and generalizability of our model's predictions. We also conducted extensive hyperparameter tuning and experimented with different optimizers for both models to achieve high accuracy. Ultimately, the second model, which consists of SE blocks into a more streamlined version of the traditional ResNet framework, achieved the highest accuracy. Therefore, we recommend this model as the optimal choice for achieving good accuracy on the CIFAR-10 dataset.

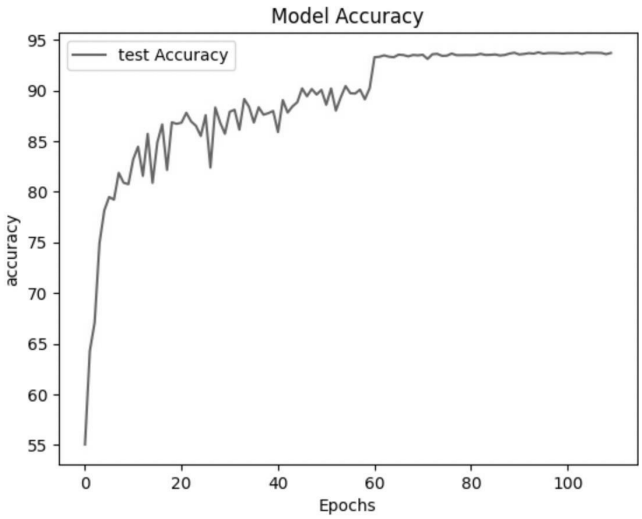

Figure 1: Test Accuracy for Dense SE Resnet with shared parameters, showing SGD's performance with MultiStepLR scheduling.

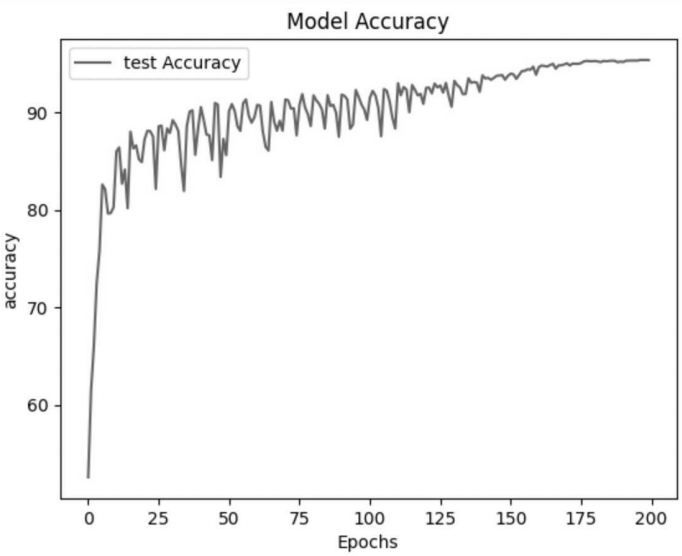

Figure 2: Test Accuracy for Basic SE Resnet, showing SGD along with Lookahead teachnique and CosineAnnealingLR scheduling.

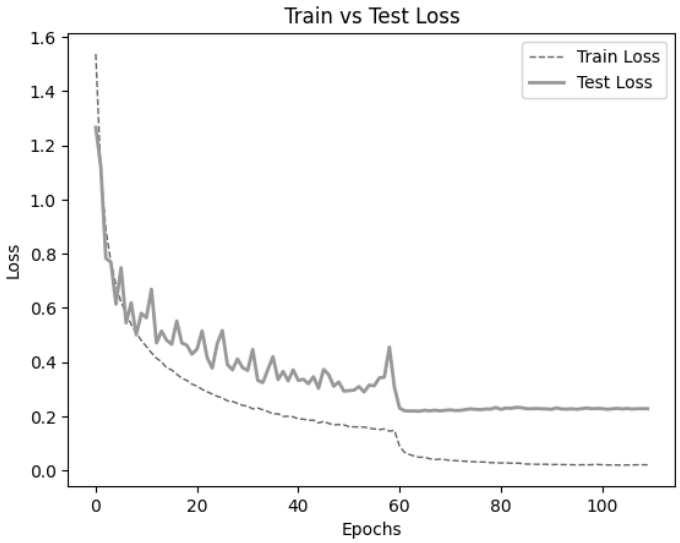

Figure 3: Training and Test Loss for DenseSE Resnet with shared parameters.

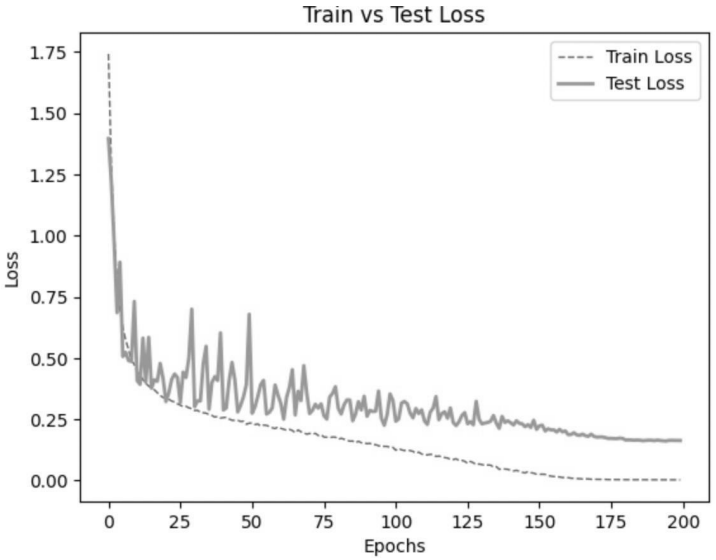

Figure 4: Training and Test Loss for Basic SE Resnet

Results

In our evaluation of different architectures, we refined our model's design using a standard hyperparameter set: 200 epochs, batch size of 128, learning rate of 0.1, weight decay of 5e-4, and momentum of 0.9. Stochastic Gradient Descent (SGD) [6] was the chosen optimizer, coupled with Cross-Entropy loss to optimize our model's weights. This precise configuration led our model to achieve a notable test accuracy of 95.6%. Including advanced methods such as Lookahead optimization [8] and gradient clipping was crucial, enhancing the training process and contributing to rapid and consistent convergence. These decisions, based on a combination of practical findings and established principles, resulted in a high-performing model that exemplifies the thorough and systematic approach of our findings.

Parameter	Our Model
number of residual layers	3
number of residual blocks	[4, 4, 3]
convolutional kernel sizes	[3, 3, 3]
shortcut kernel sizes	[1, 1, 1]
number of channels	[64, 128, 256]
average pool kernel size	8
batch normalization	True
squeeze and excitation	True
Total number of Parameters	4,697,742

Figure 5: Hyperparameters for our finalized model[2]

References

[1] Azadbakht, A. et al 2022. Drastically Reducing the Number of Trainable Parameters in Deep CNNs by Inter-layer Kernel-sharing. arXiv:2306.12100.

[2] Gupta, N.; Chauhan, H.; and Thakur A 2022. Mini-Project 1: Residual Network Design. Available at https://github.com/Nikunj-Gupta/Efficient_ResNets/. Accessed: 2024-04-04.

[3] Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-Excitation Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 7132-7141).

[4] Kingma, D. P., & Ba, J. L. (2015). Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

[5] Liu, K. 2021. Train CIFAR10 with Pytorch. Available at https://github.com/kuangliu/pytorch-cifar. Accessed: 2024-04-04.

[6] Ruder, S. (2017). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

[7] Zhang, J., He, T., Sra, S., & Jadbabaie, A. (2020). Why Gradient Clipping Accelerates Training: A Theoretical Justification for Adaptivity. In Proceedings of the International Conference on Learning Representations (ICLR 2020).

[8] Zhang, M.; Lucas, J.; Ba, J.; and Hinton, G. E. 2019. Lookahead Optimizer: k Steps Forward, 1 Step Back. Advances in Neural Information Processing Systems, 32.

Acknowledgement

We have developed a tailored ResNet model, modifying hyperparameters and experimenting with different optimizers and learning rates for training. Our work is based on foundational code available at [1] (for the Dense SE ResNet featuring parameter sharing) and at [2] and [5] (for the basic SE-enhanced ResNet).

We extend our gratitude for the guidance and resources provided by Professors Chinmay Hegde, the TAs, and the Google Collab team.