Eric Chun

Ejchun@ucsd.edu

Abstract

This project explores the performance of different convolutional neural network architectures such as LeNet, VGG, and a base model using the CIFAR-10 dataset. Hyperparameter tuning and different optimization methods to optimize model performance. Results show that VGG outperforms LeNet and the base model, achieving an accuracy of 76%. Hyperparameter tuning was used on the base model to improve its accuracy by 2%. The findings highlight the importance of architecture choice and hyperparameter optimization in improving results on image classification. The base model is a modified version of the CNN model used in my A4 homework for cogs181. LeNet and VGG models were adapted from PyTorch for Cifar at https://github.com/kuangliu/pytorch-cifar

Introduction

Convolutional neural networks are a standard for image classification tasks. The CIFAR-10 dataset that contains 60,000 32x32 color images across 10 classes with 6000 images per class, serves as a benchmark for evaluating CNN performance. This project aims to compare the performance of two classic CNN architectures—LeNet and VGG—against a base model, with a focus on hyperparameter tuning to optimize accuracy.

Method

Models:

Base: The base model consists of two convolutional blocks. Each convolutional block has one convolutional layer, one ReLU activation layer, and one average pooling layer. The intermediate output after two convolutional blocks has the shape $20 \times 8 \times 8$. Output is then flattened then passed through one linear layer, one ReLU activation layer, and then another linear layer. Cross-entropy is used as the loss function and the stochastic gradient descent is the optimizer.

LeNet: Base LeNet architecture was modified to include batch normalization and dropout for regularization to reduce overfitting. Hyperparameters were taken from the modified base model to increase accuracy.

VGG: A VGG11 architecture was implemented, with adjustments to the fully connected layers to suit the CIFAR-10 dataset. It was also modified to include batch normalization and dropout for regularization. Hyperparameters were taken from the modified base model to increase accuracy.

Hyperparameter tuning and modifications.

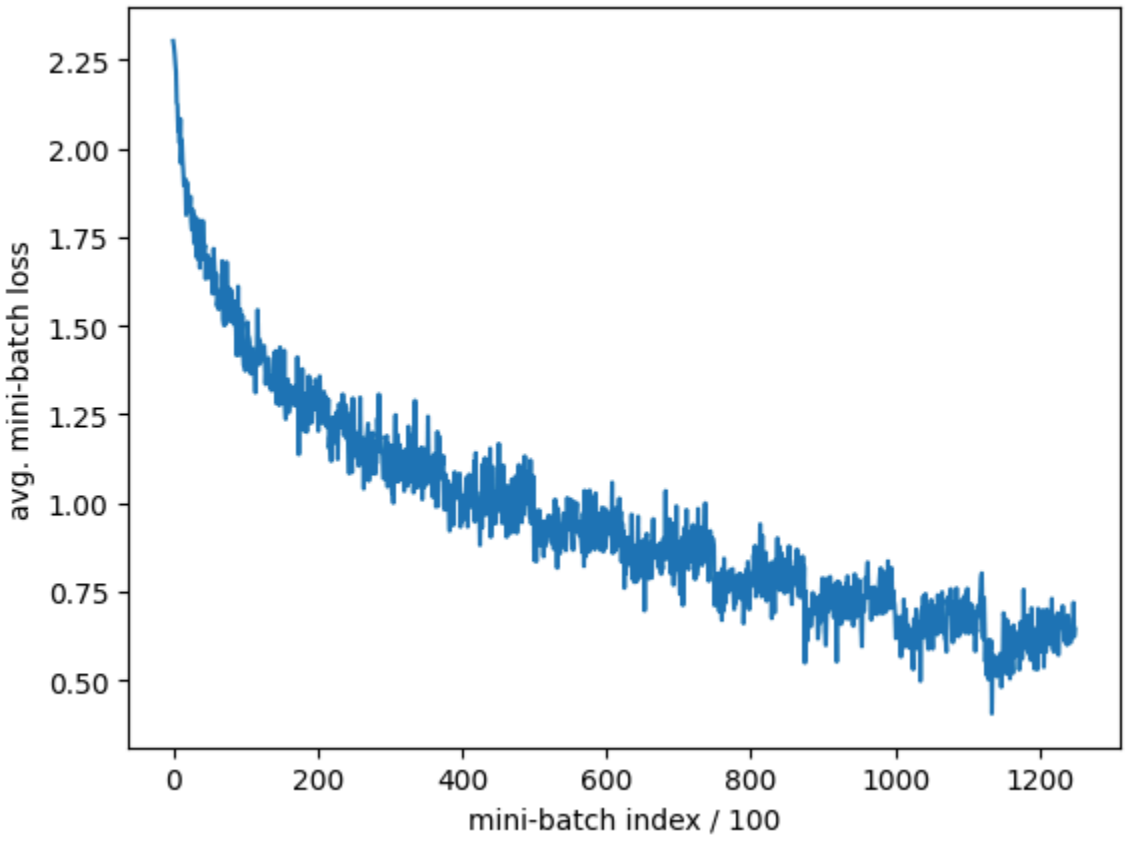
Initially, learning rate, batch size, optimizer, and weight decay values were experimented with. After many failed attempts only epoch size and batch size were tuned to optimize model performance. Adam was initially tried to see if it would improve the model however, it only improved the model with a lower epoch size. SDG with weight decay, batch normalization, and dropout were also experimented with and used to reduce overfitting.

Training

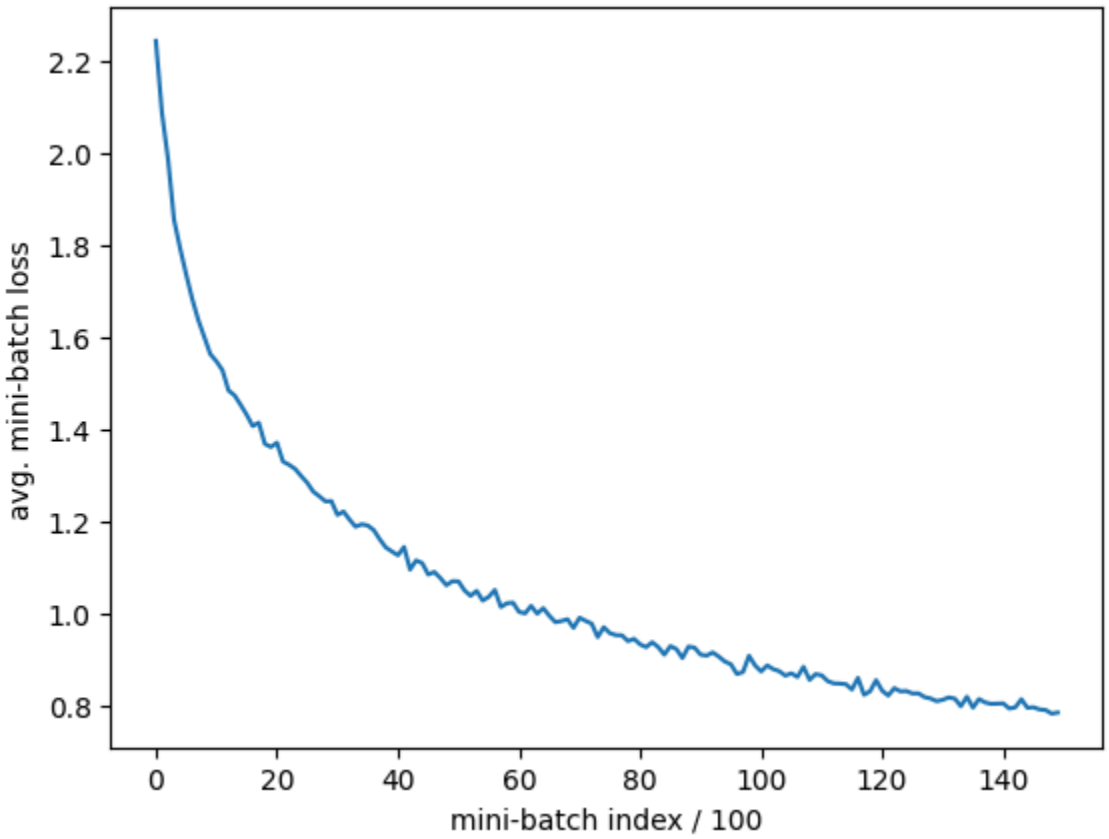
Most models typically were trained for 50 epochs using cross-entropy loss and the SDG optimizer. Some experimented with Adam and lower epochs but were switched to SDG.

Results

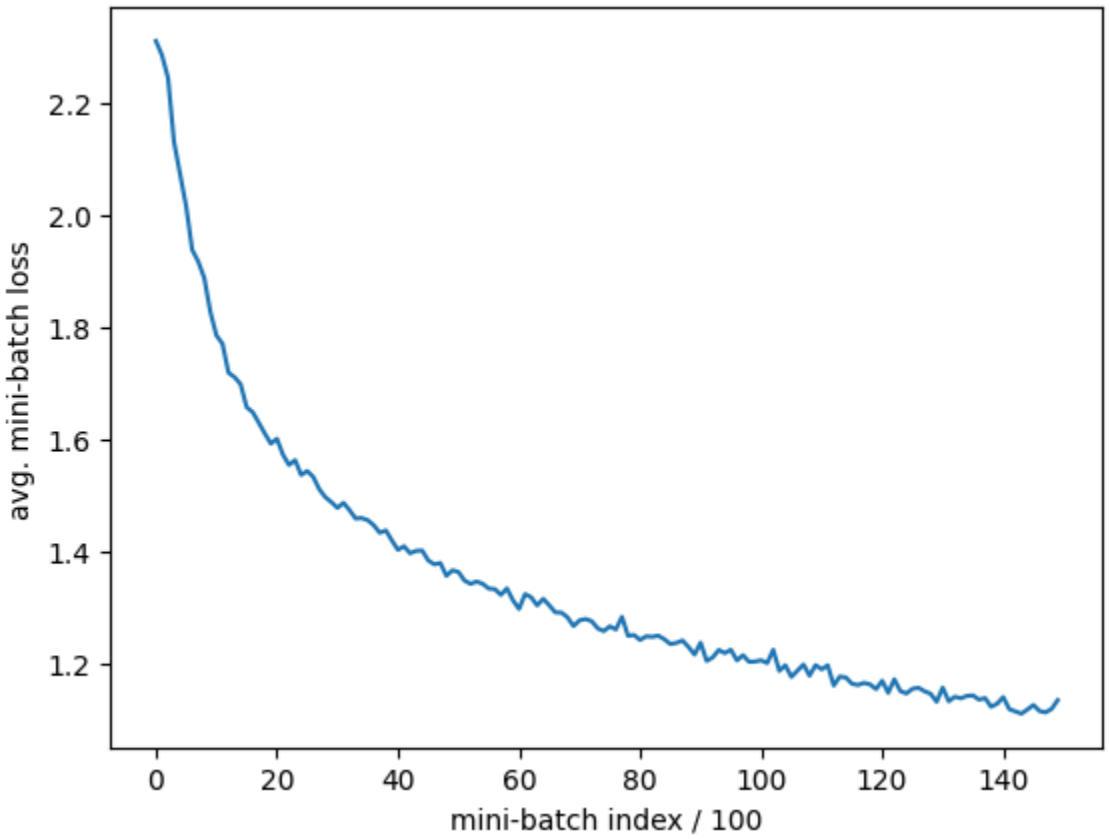
Base Model: Test accuracy of 66%. The base model performed generally well. Results were as expected based on A4 results



Base model with hyperparameter tuning: Test accuracy of 67%. I expected higher test accuracy after changing so many hyperparameters and adding batch weight optimization and dropout.

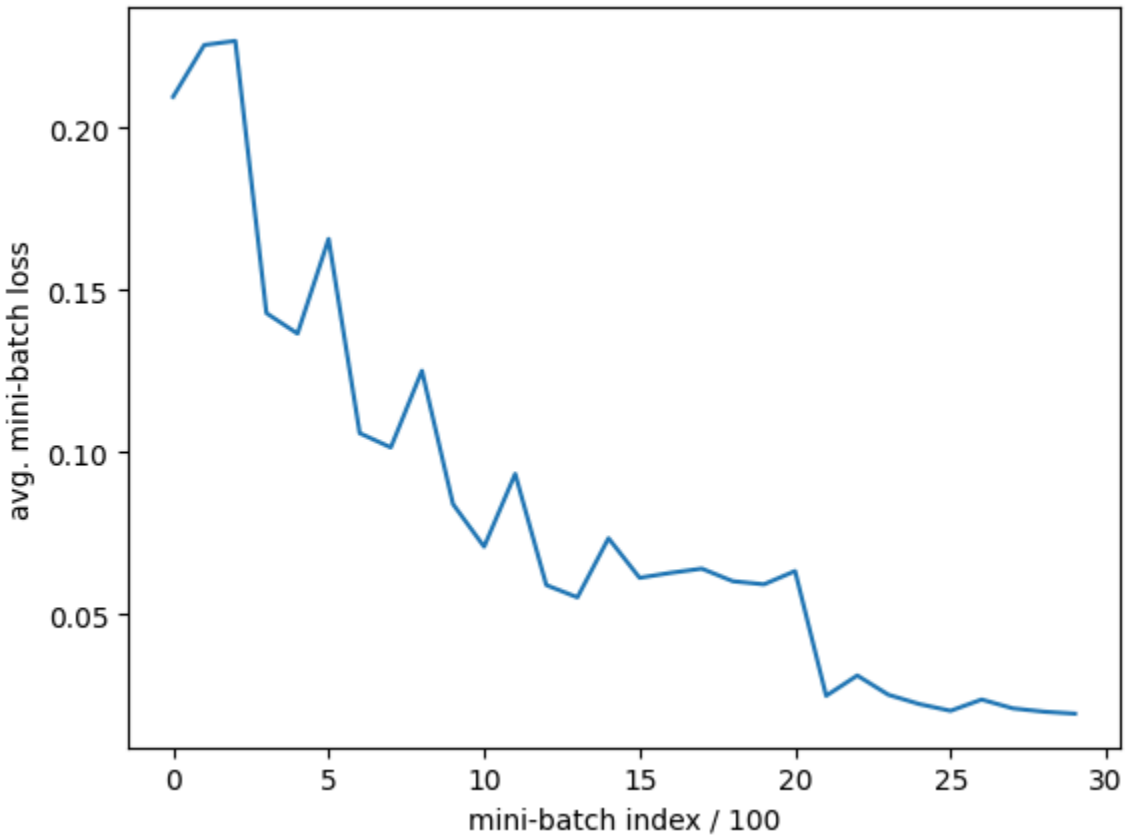


LeNet: Test accuracy 58%. Original LeNet accuracy before hyperparameter tuning was around 30%. Raising epoch count was a large factor in the improvement of this model.



VGG: Test accuracy 76%. This was the model with the highest accuracy. I would have liked to test it before and after hyperparameter tuning but unfortunately even running the model on VGG11 and reduced epoch size it still took over an hour to finish training. Perhaps running it on a GPU instead of a CPU with a stronger processor would have allowed for more efficient testing.

The model was run on 15 epochs on VGG11.



Conclusion

This project explored the impact of architecture choice and hyperparameter tuning on CNN performance. VGG11 achieved the highest accuracy, while LeNet provided a decent performance but a really fast training time. While Hyperparameter tuning proved essential for maximizing performance in the LeNet model, it didn't seem to help the base model all that much. Perhaps different optimizations could have improved the base model's accuracy way more. I would have liked to see how much better I could have gotten the VGG model but I didn't want to wait for multiple additional hours for the model to finish training. It was interesting to see how much the accuracy improved in the times between LeNet and VGG. I am curious as to how much we can continue to improve and create new architectures with both speed and precision.

References

kuangliu. (2017). *GitHub - kuangliu/pytorch-cifar: 95.47% on CIFAR10 with PyTorch*. GitHub. https://github.com/kuangliu/pytorch-cifar

Cogs181 A4 homework assignment 2025