Cristian Ceja Bolanos
12/01/25
CSCI 167
Final Project

Project Report

GitHub Repository:

https://github.com/CristianEyebrow/167_project

Introduction and Dataset

The dataset I chose to use was the CIFAR-10 dataset. This dataset is commonly used in computer vision and to train models on image. The reason I used this dataset is because it has a variety of configurations that give me enough freedom to test different algorithms/architectures for deep learning. CIFAR-10 is made up of 60,000 color images with a split of 50,000 into training images and the remaining 10,000 into test images. The images are distributed into 10 different categories (animals, vehicles, etc.).

I used torchvision.transforms to preprocess the dataset so that training the model would be smoother. Within the training, I used augmentations, like normalization and unit variance, to reduce overfitting and include variability. PyTorch was used to load the dataset to enable efficient processing.

Model Design

The model design consists of 2 main sections: SimpleCNN and DeeperCNN. Both pf these primary convolutional neural networks were used to evaluate how well they performed in comparison to each other.

The SimpleCNN was used to evaluate a simpler and baseline architecture that is lightweight. It consists of two convolutional layers, ReLU activations, Max pooling, and two fully connected layers for classification. Although this doesn't exactly have a massive number of parameters, it is still very good for helping us understand how the hyperparameters work and how different strategies for optimization affect the results.

The DeeperCNN is essentially a more complex and expressive version of the simpleCNN. With the DeeperCNN, we used increasing channel depth in three convolutional layers. It also consisted of batch normalization, dropout, and a two-layer connected classifier. This model was, as expected, more expressive and allowed for additional depth in the system. It helped the model learn hierarchical feature much better and significantly more capable than the SimpleCNN.

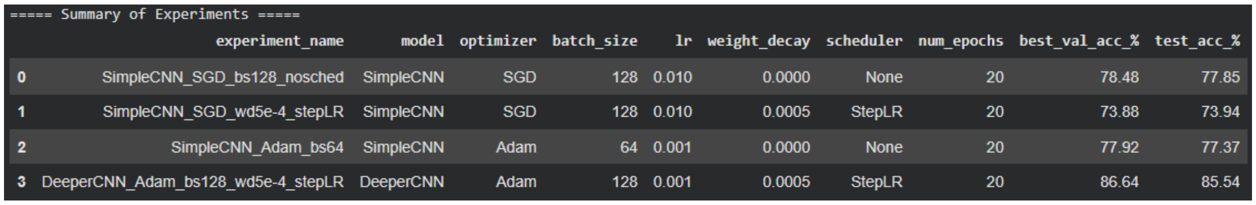
Setup and Hyperparameters

The setup was similar for each of the two main architectures. Since I wanted to reduce the difference between both designs, I tried to keep the training of them as similar as possible so any difference in their performance can be attributed primarily to their system designs, not the training design. The training consisted of 15 epochs, but I tested with up to 20 for more results. The procedure was similar for both architectures. First, the training data (CIFAR-10) was loaded and preprocessed. Next, the hyperparameters and optimizations techniques were trained on each model. Once we got the results from these trainings, we observed the results from each epoch and confirmed the loss/accuracy scores. Then, we evaluated the final performance on the test set to see how well it performed on that set. Finally, we took the results from the test-set performance and recorded the best validation accuracy.

For the hyperparameters, I settled on 3 optimizers: SGD with fixed learning rate, SGD with weight decay and StepLR, and Adam. For the batch size, we used 64 and 128. For the learning rate, we simply had fixed learning rate and stepLR with a factor of 0.5 every 3 epochs.

Results and Analysis

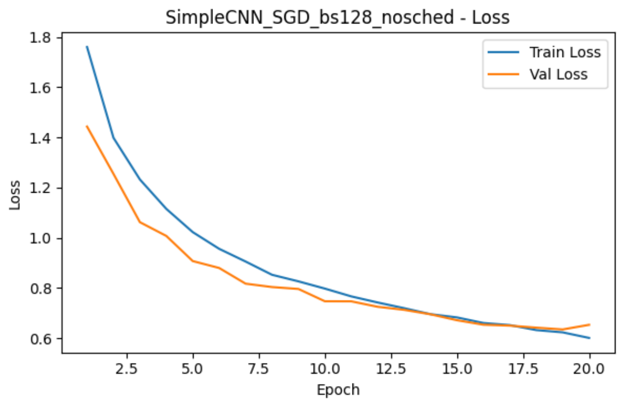
There are the results produced from 20 epochs.

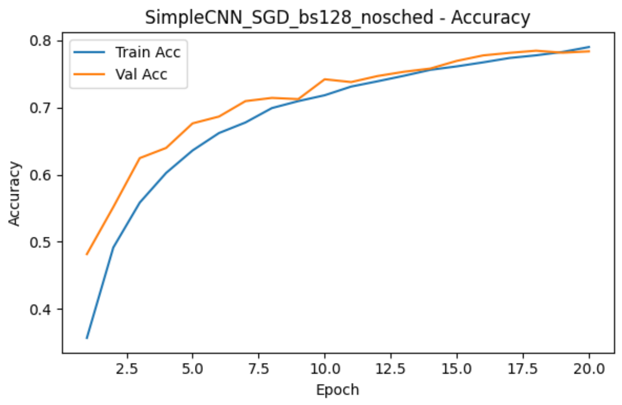


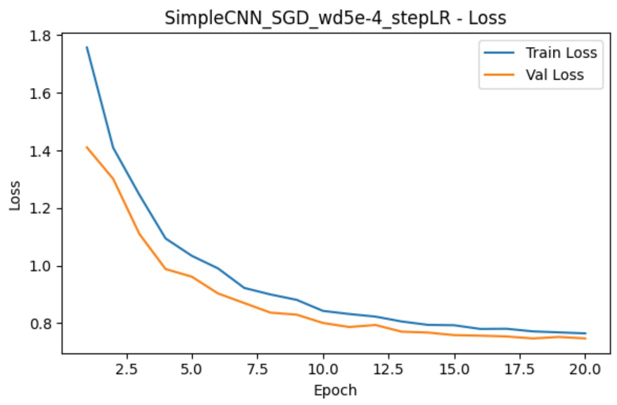
===== Summary of Experiments =====

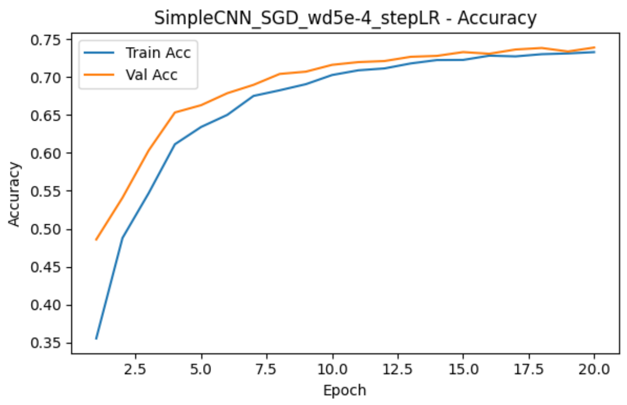
	experiment_name	model	optimizer	batch_size	lr	weight_decay	scheduler	num_epochs	best_val_acc_%	test_acc_%
0	SimpleCNN_SGD_bs128_nosched	SimpleCNN	SGD	128	0.010	0.0000	None	20	78.48	77.85
1	SimpleCNN_SGD_wd5e-4_stepLR	SimpleCNN	SGD	128	0.010	0.0005	StepLR	20	73.88	73.94
2	SimpleCNN_Adam_bs64	SimpleCNN	Adam	64	0.001	0.0000	None	20	77.92	77.37
3	DeeperCNN_Adam_bs128_wd5e-4_stepLR	DeeperCNN	Adam	128	0.001	0.0005	StepLR	20	86.64	85.54

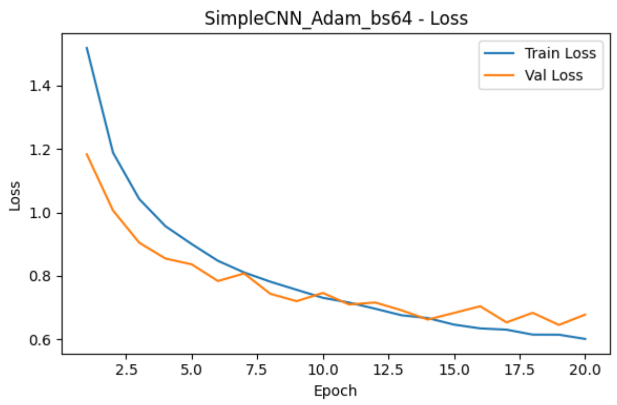
From these results, we can conclude that the DeeperCNN with Adam and StepLR greatly outperforms any of the other SimpleCNN models. DeeperCNN achieved the best validation accuracy of 86.64%, while the best SimpleCNN achieved their best validation accuracy of 78.48%. That is almost a 10% difference, showing a noticeable performance disparity. It should also be noted that the SimpleCNN with Adam and fixed learning rate nearly identical to the SimpleCNN without Adam eventhought they had different batch sizes (128 and 64). This could explain why the SimpleCNN with Adam underperformed given that Adam usually improved performance of structure. That aside, the worst performance was from the SimpleCNN with StepLR. It performed worse than the others big a decent amount, roughly 4% from the next lowest best validation score.

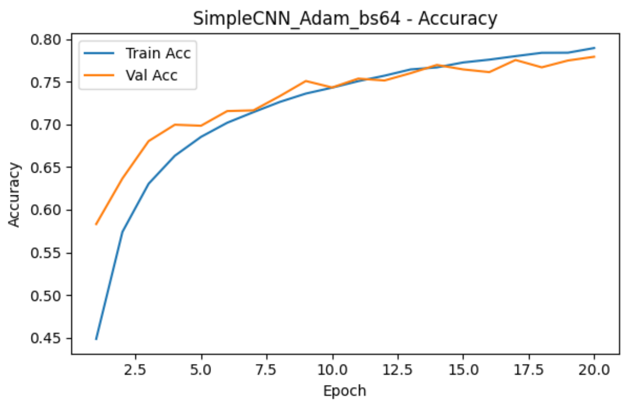


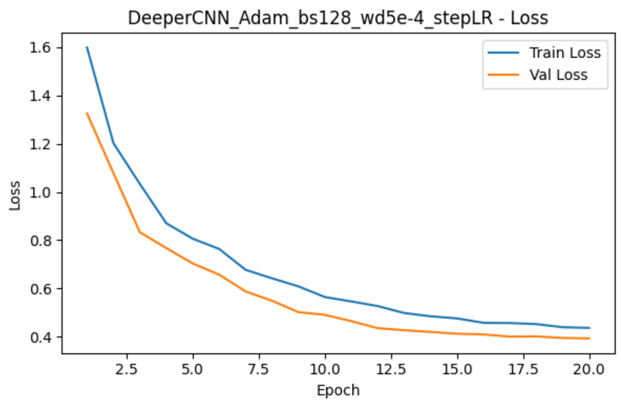


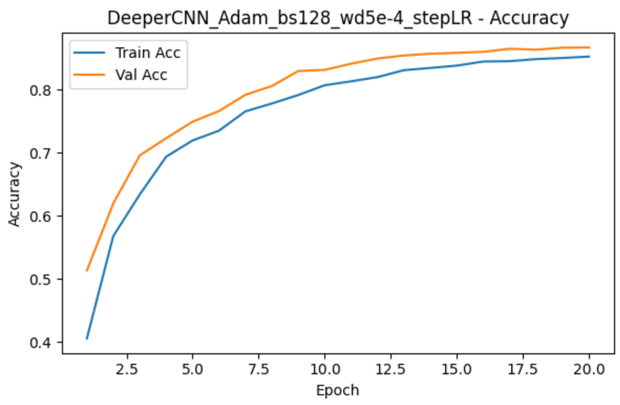












Here we have another representation of the output. These are basic graphs that map the accuracy in respect to each epoch given the training accuracy and validation accuracy/loss. It helps us understand and visualize the results more and how exactly each epoch was performing relative to the learning curve. As we can see, the best models seem to perform at an almost parallel rate when comparing the training and validation accuracy. It shows the consistency and accuracy of the best system. On the other hand, the worse model showed more inconsistencies and spikes in accuracy/loss. Overlapping in the graph from this model (SimpleCNN + Adam) suggests that the test model was performing worse than the training sets, which is not ideal.

Overall, the results aligned with our established belief in deep learning that deeper networks provide more linear regions and greater representation capabilities for image tasks.

Cristian Ceja Bolanos
12/01/25
CSCI 167
Final Project

Final Thoughts

Through systematic experimentation, I was able to demonstrate how different architecture choices influence performance in computer vision. The choice of optimizers, hyperparameters, and algorithms configures the depth and optimizations of training the models. One of the key findings from the experiment was that learning rate scheduling and regularization are crucial for deeper models because they improve accuracy and stability.

For future research, I would consider adding another architecture to further test how much configuration can change depending on the base model. An additional option would also be to include another one or two more datasets to compare a cross-section generalization. There were also some instances where the biases and failed models were a bit hard to concisely pin down, so there could be some work in that regard. Maybe adding more through evaluation metrics for the misclassifications.