DS542 Midterm Report

Namika Takada
March 31, 2025
CDS DS 542

Namika Takada

Created on March 28 | Last edited on March 28

Model Description

Part one of my project was a SimpleCNN architecture which was designed as a baseline model for the CIFAR-100 image classification. My model consisted of two convolutional layers - 3x3 kernels - each with 16 and 32 filters. It had two fully connected layers: 256 units and then 100 output classes. This architecture had the ability to identify hierarchical features, and the convolutional layers extracted local patterns. The fully connected layers placed at the end of the CNN processed the feature maps learned by the two convolutional layers.

For part two of my project, I chose ResNet18 - a more sophisticated convolutional neural network architecture designed for image recognition. ResNet18 has 18 layers and leverages residual connections to address the graudent problem to deep networks. The use of the predefined

Resent18 architecture from torchvision with the modification of the final fully connected layer to output 100 classes greatly improves performance on image datasets.

In the last part of the project, I used a pretrained ResNet50 model fine tuned for the CIFAR-100 dataset. Similar to my previous model, both are convolutional neural network architectures, but ResNet50 is significantly deeper with 50 layers. This model performs transfer learning to adapt to CIFAR-100. I replaced the final fully connected layer to output 100 classes for CIFAR-100.

Hyperparameter Tuning

The following are the metrics for my part one CNN model. Batch size: 8, learning rate: 0.1, epochs: 5, weight decay: 0.0001. Using a CosineAnnealingLR schedule to manage the learning rate can help improve the model convergence.

The following are the metrics for my part two ResNet18 model. Batch size: 32, learning rate: 0.01, epochs: 5. I tried to find a balance between memory constraint and convergence speed of batch size and finalized on 32 as a sweet spot.

The following metrics for my part three ResNet50 model were chosen based on practices for fine-tuning pretrained models. Batch size: 128, learning rate: 0.001, Epochs: 10. I chose the Adam optimizer with a learning rate of 0.001. I chose 0.001 as the learning rate as a starting point but chose to follow through with it for the rest of my runs. I increased the batch size from my previous model to offer even more efficient training on the GPU.

Regularization Techniques

In part one, the weight decay in the Adam optimizer is 0.001. The learning rate scheduling in the CosineAnnealingLR, as previously mentioned, reduces the learning rate. Both of these techniques prevent the model from overfitting and potentially better generalization.

In part two, the CosineAnnealingLR used to reduce learning rate over time helps prevent overfitting.

In part three, I chose a weight decay of 0.0001 in the Adam optimizer to prevent overfitting. I also implemented the same CosineAnnealingLR to gradually reduce the learning rate for the scheduler.

Data Augmentation Strategy

In the model for part one, I did not include any data augmentation for the training set. Only ToSensor() conversion and the normalization with mean (0.5, 0.5, 0.5) and standard deviation (0.5, 0.5, 0.5) is applied. I wanted to have the most basic cnn model for part 1; however, I did include data augmentation for part 2 and 3.

In part two of my model, I implemented one data augmentation strategy for the CIFAR-100. The random horizontal flip applied to the training data helps the model become less sensitive to the orientation. As for the normalization, I kept it the same from part 1. The augmentation is minimal but effective for image processing.

In part three, I implemented more data augmentation to my training set to greatly improve the performance of the model. I used Random Horizontal Flip which exposes the model to a wider range of variations in the data. I resized the images to 256x256, so the model has a simplistic input to train on. I also used Random Crop to 224x224 with padding. By doing so, the model is exposed to different parts of the images during training. As a result, the model is more robust to variations in object placement. Lastly, I used the ImageNet normalization of mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225). This helps ensure that the pixels for the images are on a similar scale to speed up training and improve model performance.

Result Analysis

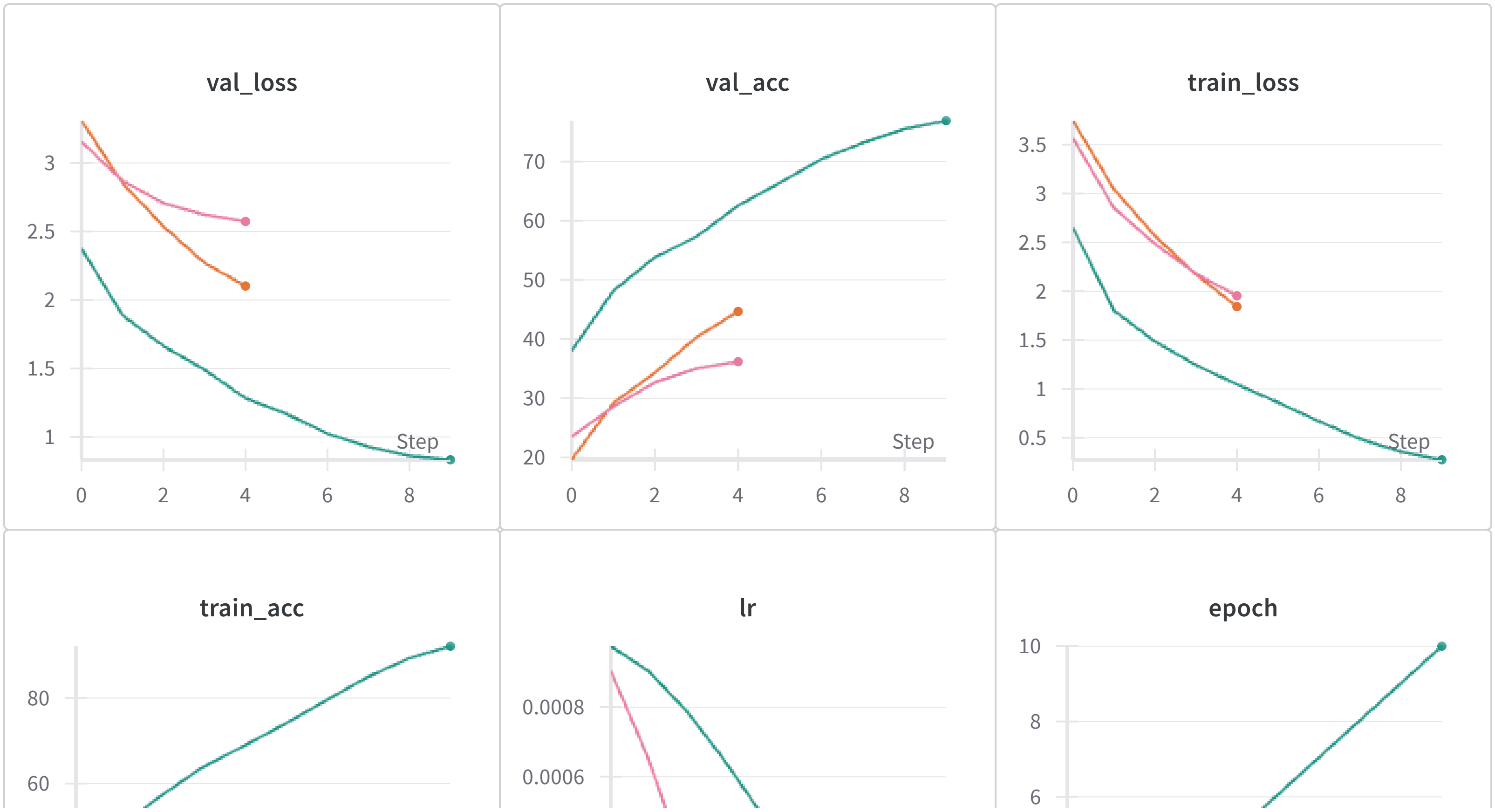
For my part one, I had a score of 0.203 with approximately 50% of the test data. The strengths of the CNN model is the simple architecture combined with the efficient training from the Adam optimizer and learning rate scheduling. This model has several weaknesses - the simplicity of the model can struggle with the complexity of CIFAR-100. The lack of data augmentation can lead to poor generalization since the model is not exposed to a wide enough range of variations in the training data. Some areas of improvement could be implementing data augmentation techniques and experimenting with different architectures which is done for part 2 and 3.

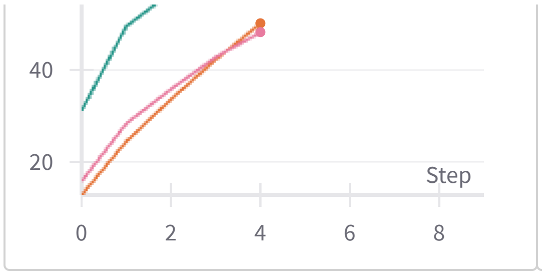
For my part two, I had a score of 0.252 with approximately 50% of the test data. This was a slight improvement from my previous model. The strengths of my model is the sophistication of ResNet18 to handle learning complex features. The architecture is designed to be computationally efficient, which requires less parameters than other models and also prevents the vanishing gradient problem. The use of one data augmentation reduces overfitting and improves overall model performance. One weakness is the minimal data augmentation strategy that might not leverage the model's full capabilities. To improve my model, I can increase the number of training epochs and implement more data augmentation for a better performance.

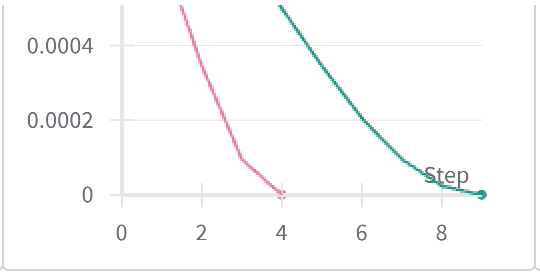
In part three of my model, I achieved an accuracy of 0.49 with approximately 50% of the test data and 0.76 when fine tuned locally on the CIFAR-100 dataset. This model performed the best and this can be attributed to the utilization of transfer learning from a pretrained model. The pretrained ResNet50 model leveraged feature representation from large datasets like ImageNet. As a result, there was faster convergence and better generalization on the CIFAR-100 dataset. The implementation of several data augmentation strategies contributed to the model's performance. The techniques helped expose the model to various transformations of the input data. Furthermore, the learning rate scheduler, CosineAnnealingLR, aided in a gradual decrease in learning rate to effectively fine-tune the pretrained weights. However, my model can greatly improve in various areas. First and foremost, in the experiment tracking of all the runs completed down below, there was one model with 30 epochs. Although that model seemed to perform the best, my computer crashed before it could create a submission csv file, so I did not increase my epoch above 10. This greatly limited the model's ability to fully adapt to the CIFAR-100 dataset. Additionally, the fixed learning rate across all layers may not be optimal for fine-tuning in pretrained model. Implementing a layer-wise learning rate could lead to a better performance overall. Last but not lease, I had to apply the same transformations as in the training pipeline in the evaluation file for consistency. To ensure that the input data is processed in the same way during training and inference, it went under specific transformations. I ensured that the OOD test images are processed to match the format and scale of the training data. To account for this, I made the necessary adjustments for the eval_ooo.py file for part 3.

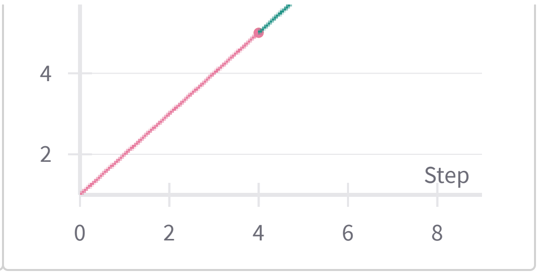
Experiment Tracking Summary

Best from each model

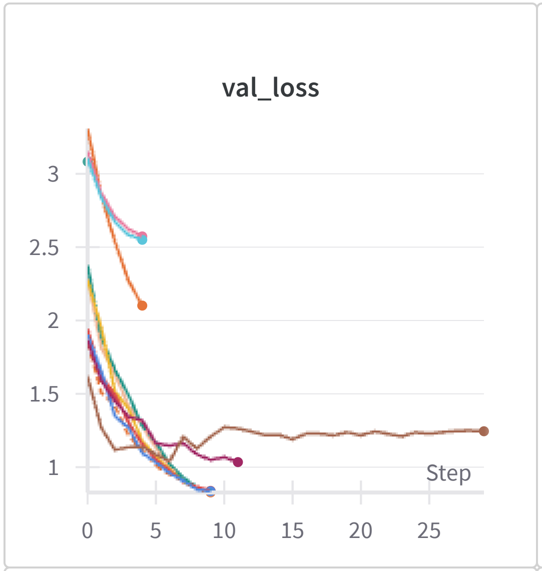


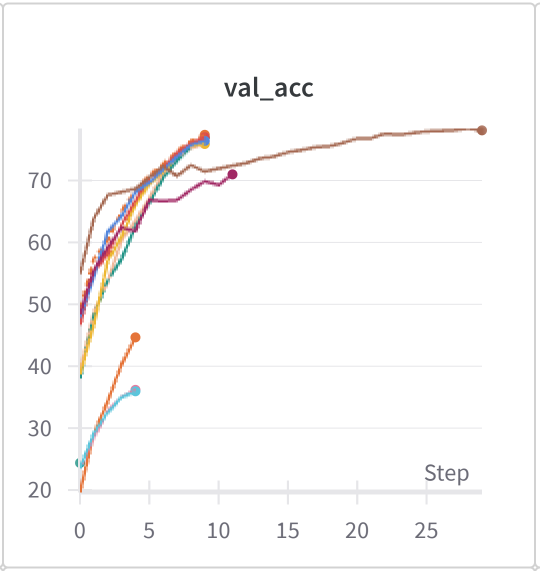


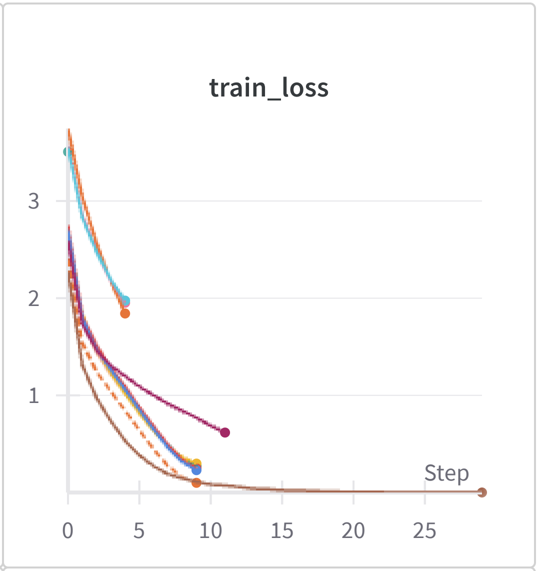


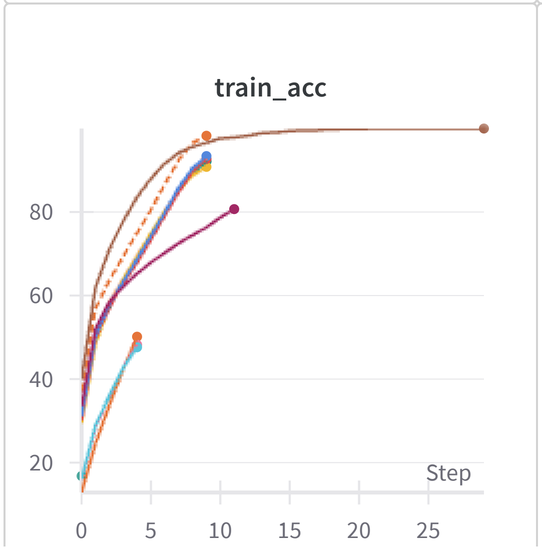


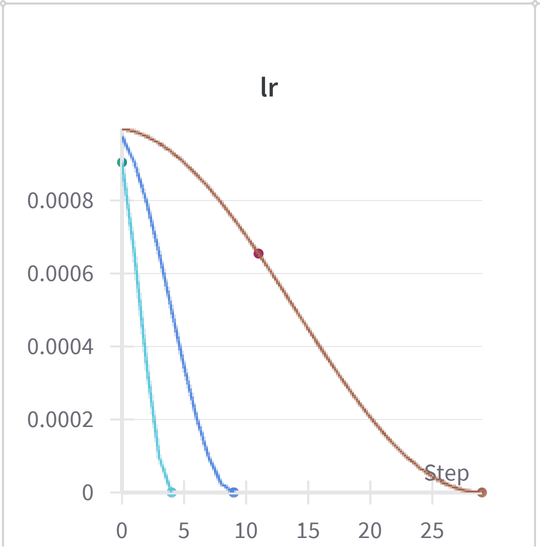
From all runs

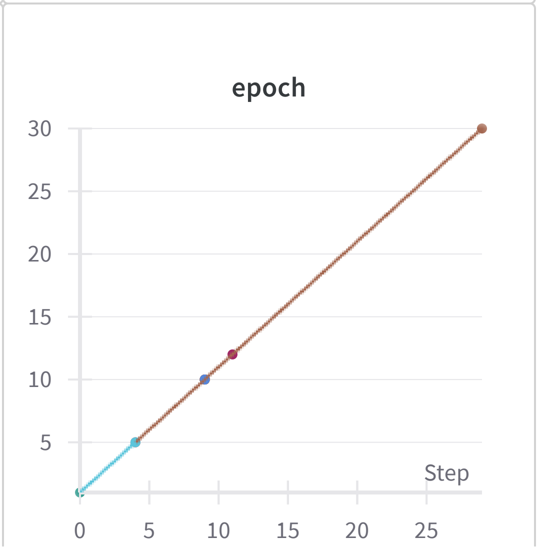












AI Disclosures

For this midterm, I used ChatGPT to assist in selecting the best optimizer and scheduler for each model in the project. I asked ChatGPT about suitable optimizer for CNNs on image classification. It suggested using Adam optimizer with learning rate 0.001 and weight decay of 0.0001. Before fully implementing ChatGPT's recommendation to my model, I researched different optimizers and schedulers to explore my options. By leveraging ChatGPT, I was able to greatly enhance the performance of my model while verifying the information provided. I learned what optimizer/scheduler was best for each model.