Midterm Challenge Report

Mingyu Chen

March 31, 2025

1 AI Disclosure

I used ChatGPT to get suggestions on hyperparameter tuning, such as how to choose the range of learning rates, batch sizes, etc.

2 Model Description

Part 1 In this part, we design a simple CNN (namely *SimpleCNN*), which consists of three convolutional blocks, each including a convolutional layer followed by batch normalization, a ReLU activation, and max pooling. This design helps model progressively extract hierarchical features while reducing spatial dimensions. The number of feature channels increases from 32 to 128 across the blocks, enabling the model to learn richer representations. Finally, a fully connected layer maps the flattened feature maps to 100 output classes, making it suitable for CIFAR-100 task.

Part 2 In this part, we use *ResNet-101* from torchvision as the pretrained model. ResNet-101 is a CNN network that consists of 101 layers and uses shortcut connections to allow gradients to flow more easily during backpropagation, which avoids the vanishing gradient issue common in deep network optimization.

Part 3 In this part, we still use *ResNet-101* from torchvision as the pretrained model. Unlike Part 2, we applied several regularization techniques, including data augmentation, label smoothing, and cosine learning rate scheduling. With these methods, we significantly improve the performance.

3 Hyperparameter Tuning

Learning rates We tune over the learning rates from $\{0.01, 0.001, 0.0001\}$ for the two models. We finally select 0.01 for *SimpleCNN* and 0.0001 for *ResNet-101*.

Batch sizes We tune over the batch sizes from $\{32, 64, 128\}$ for the two models. We finally select 64 for *SimpleCNN* and 128 for *ResNet-101*.

4 Regularization Techniques

Weight Decay We add a weight decay on the optimizer. Such a penalty term is added to the loss function to discourage large weights and reduce overfitting.

Dropout We add a dropout in the last fully connected layer of Resnet-101. Such a design increase robustness of the training.

Batch Normalization Batch Normalization is built into *SimpleCNN* and *ResNet-101* architecture. Such a normalization helps stabilize and speed up training.

Label Smoothing We set label smoothing as 0.1 in the CrossEntropyLoss. Softens the target labels slightly to prevent the model from becoming overfitting.

5 Data Augmentation Strategy

Random Crop Randomly crops the image with padding to simulate slight shifts and improve spatial robustness.

Random Horizontal Flip By flipping the image horizontally with 0.5 probability, we improve the model's generalization to mirrored inputs.

Color Jitter *ColorJitter* randomly changes brightness, contrast, saturation, which simulates different lighting conditions and prevent overfitting.

Normalize We use CIFAR-100-specific mean and std to standardize each color channel. This helps stabilize and speed up training.

6 Results Analysis

Results Comparison Below is the best score and the corresponding learning rate/batch size of the three parts. As shown in the table, we achieve a better result than the best benchmark performance of 0.397 in Part 3 experiments. It is worth noting that the score in Part 2 is lower than in Part 1. This is mainly because *ResNet-101*, compared to *SimpleCNN*, has a much deeper architecture (101 layers) and therefore requires more sophisticated regularization techniques and learning rate scheduling. After carefully designing

the regularization methods, as shown in the Part 3 results, *ResNet-101* outperformed *SimpleCNN*.

Part	Learning Rate (lr)	Batch Size (bs)	Score
Part 1	0.001	64	0.29656
Part 2	0.0001	64	0.24560
Part 3	0.0001	128	0.46520

Table 1: Comparison of different training settings and their scores.

Models Comparison The comparisons are shown in Table 2. *SimpleCNN* is a shallow and lightweight model with low computational complexity and fast training speed, making it suitable for small-scale tasks. However, it offers limited representational capability and feature extraction power. In contrast, *ResNet-101* is a much deeper model with high computational demands and slower training, but it has more capabilities to learn complex features and representations. While *ResNet-101* provides superior performance for large-scale vision tasks, it also has a higher risk of overfitting if not properly regularized, as shown in Part 2 results above.

Feature	SimpleCNN	ResNet-101
Network Depth	Shallow	Deep
Computational Complexity	Low	High
Training Speed	Fast	Slow
Representational Capability	Weak	Strong
Overfitting Probability	Low	High
Feature Extraction	Moderate	Strong

Table 2: Comparison between SimpleCNN and ResNet-101

Potential Improvements For *SimpleCNN*, potential improvements include increasing depth with additional convolutional blocks and integrating residual connections. For *ResNet-101*, improvements mainly involve reducing training cost and optimizing the trade-off between accuracy and efficiency.

7 Experiment Tracking Summary

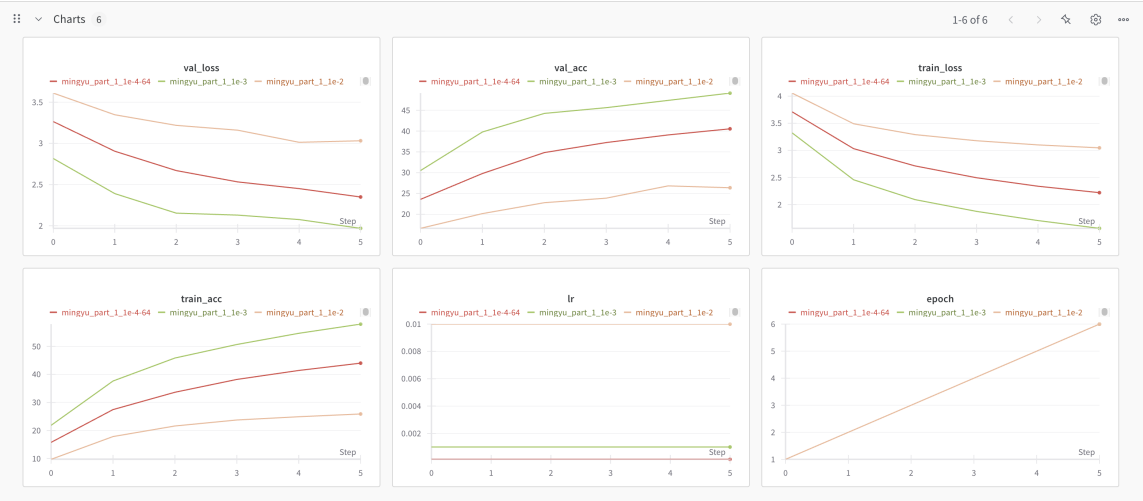

Figure 1: Part 1 5 Epochs Results

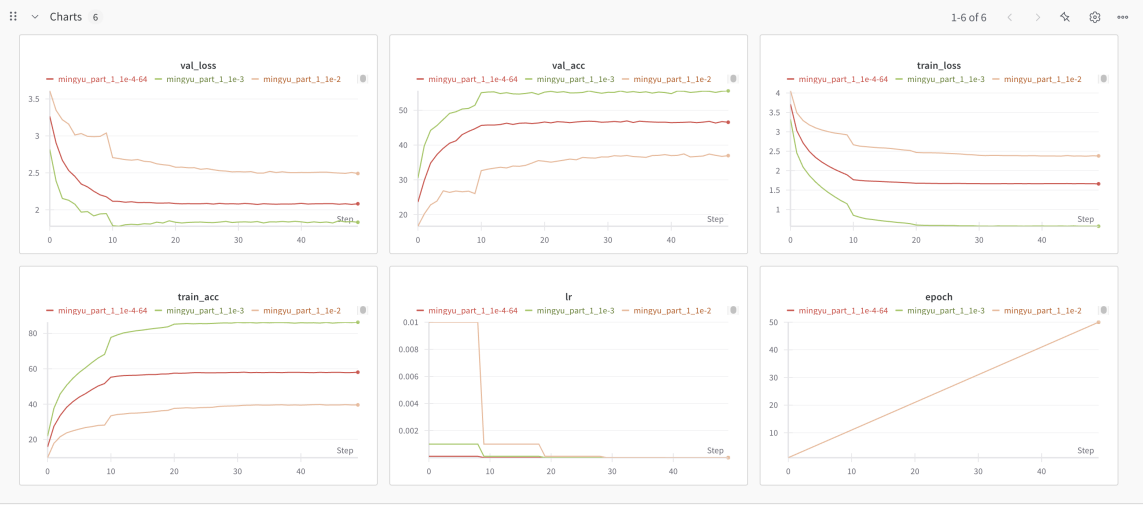

Figure 2: Part 1 50 Epochs Results

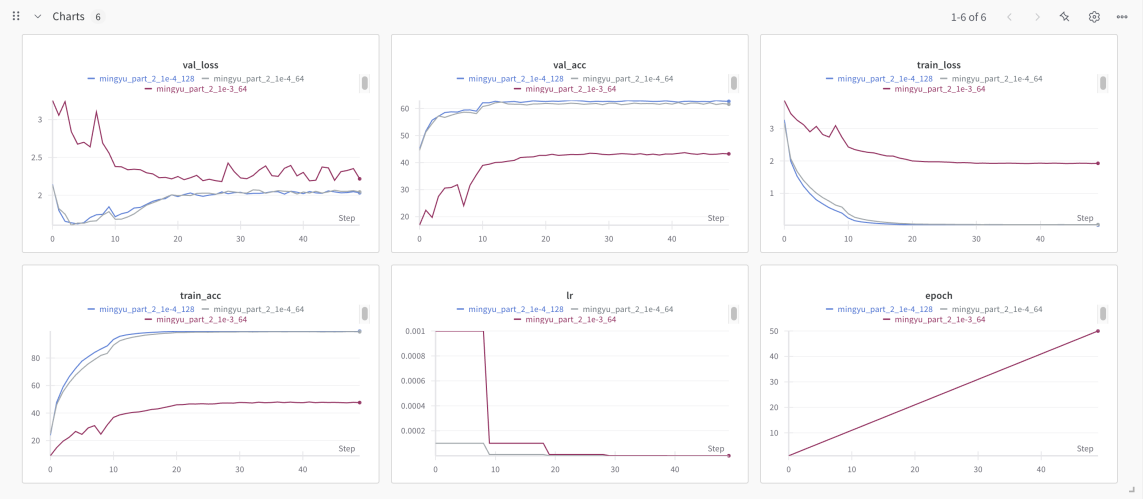

Figure 3: Part 2 50 Epochs Results

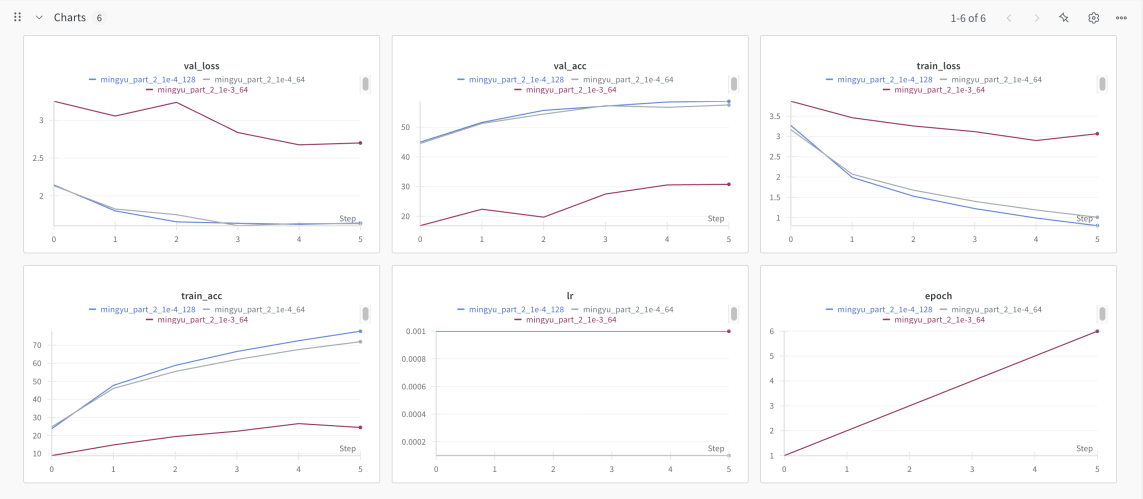

Figure 4: Part 2 5 Epochs Results

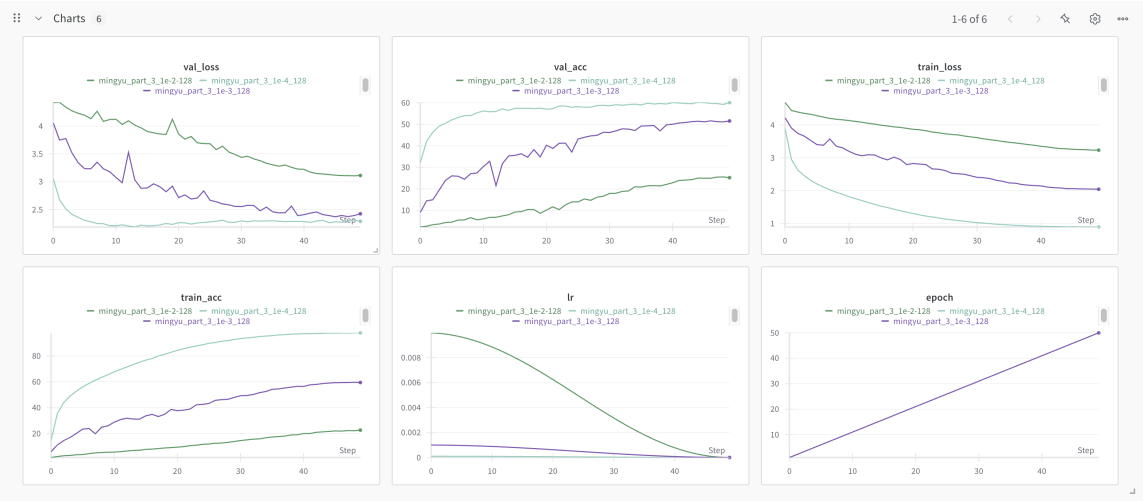

Figure 5: Part 3 50 Epochs Results