Midterm Report

Renjie Fan

30, March 30, 2025

AI Disclosure

In completing this midterm project, I made limited use of AI-based tools such as ChatGPT and GitHub Copilot for auxiliary purposes. These tools were primarily used to help brainstorm high-level ideas, verify certain implementation approaches, and accelerate debugging when encountering unfamiliar error messages.

All core components of the project—including model design, training logic, evaluation, and submission pipeline—were fully developed and implemented by me. I independently wrote the training loops, modified pretrained architectures to fit the CIFAR-100 dataset, and carefully tuned the hyperparameters based on my own experimentation and understanding.

For example, in Part 3, I designed a two-phase transfer learning strategy with staged layer unfreezing, implemented MixUp augmentation, used label smoothing, and fine-tuned the model with OneCycleLR. These decisions were informed by my learning and empirical results, with AI playing only a background role in validating common practices.

Overall, this project reflects my individual work and understanding. Any use of AI tools was strictly supplemental and did not replace original effort, creativity, or problem-solving.

Part 1: Baseline Fully Connected Neural Network

Model Description

To establish a baseline, I implemented a simple fully connected neural network (FCNN) for CIFAR-100 classification. The network consisted of a flattening layer, two fully connected hidden layers with ReLU activations, and a final output layer with 100 units for classification.

Hyperparameter Tuning

Hyperparameters were selected based on initial testing for stable convergence. I used a learning rate of 0.01, hidden layer sizes of [512, 256], a batch size of 128, and optimized the model using SGD with momentum.

Regularization Techniques

Dropout (with p = 0.5) was applied after each hidden layer to reduce overfitting. Additionally, L2 regularization was used via a weight decay of 1e-4 in the optimizer.

Data Augmentation Strategy

Data augmentation was minimal and included normalization with CIFAR-100's mean and standard deviation, along with random horizontal flips during training.

Results Analysis

The model achieved around 40% training accuracy and approximately 30% validation accuracy. Performance was limited due to the lack of inductive bias for image data, which motivated the move to CNNs in the next stage.

Part 2: CNN from Scratch (No Transfer Learning)

Model Description

In Part 2, I implemented a custom convolutional neural network (CNN) from scratch. The model used three convolutional blocks with increasing filter sizes, each followed by batch normalization, ReLU, and max pooling, and concluded with fully connected layers for classification.

Hyperparameter Tuning

I used three convolutional layers with 32, 64, and 128 filters, respectively. The model was trained with the Adam optimizer using a learning rate of 0.001, a batch size of 128, and for 30 epochs.

Regularization Techniques

Dropout (p = 0.3) was used after the fully connected layers to combat overfitting. Batch normalization was applied throughout the network to stabilize learning, and weight decay (1e-4) was also included.

Data Augmentation Strategy

The augmentation strategy included random cropping, horizontal flipping, and normalization. These techniques improved generalization by exposing the model to varied data conditions.

Results Analysis

Validation accuracy improved to around 45%, showing clear gains over Part 1. However, the model still underperformed relative to pretrained networks, particularly due to its limited depth and the small dataset.

Part 3: Transfer Learning with Improved ResNet-50

Model Description

In this part of the challenge, I adopted a pretrained ResNet-50 model as the base architecture for CIFAR-100 classification. Since CIFAR-100 images are only 32×32 in size, I modified the model to better accommodate smaller inputs. Specifically, the original large-kernel convolution (conv1) was replaced with a 3×3 convolution using stride 1, and the initial max pooling layer was removed to preserve spatial information. The final fully connected layer was also replaced with a new 100-class output layer. The training was divided into two phases. In the first phase, I froze all layers except the final classifier (fc) to allow the new head to adapt. In the second phase, I progressively unfroze deeper layers (layer4, layer3, and layer2) at strategically chosen epochs. Parameter groups with different learning rates were used to optimize the learning process during unfreezing.

Hyperparameter Tuning

I selected hyperparameters through a combination of theoretical reasoning and empirical testing. A OneCycle learning rate policy was employed with an initial learning rate of 0.01. The batch size was set to 128 to balance speed and memory usage. I used weight decay with a value of 1e-4 for regularization. To further enhance generalization, label smoothing with a value of 0.1 and MixUp augmentation with alpha set to 0.4 were included. The training included 3 freeze epochs, followed by 70 epochs

of progressive unfreezing, which allowed sufficient time for deep layers to adapt gradually without destabilizing earlier weights.

Regularization Techniques

To prevent overfitting, I implemented several regularization techniques. L2 weight decay was applied via the optimizer. Label smoothing softened the hard classification targets, improving calibration. MixUp was used during training to blend image-label pairs, which encouraged the model to form smoother decision boundaries and improved generalization under distribution shifts. Random erasing was another key technique, as it forced the model to rely on broader contextual cues rather than memorizing specific image regions. Collectively, these regularization strategies contributed to both improved robustness and better test-time performance.

Data Augmentation Strategy

The data augmentation pipeline was tailored for transfer learning. I employed RandomResizedCrop to ensure the model could handle objects at different scales, followed by RandomHorizontalFlip to improve invariance to orientation. ColorJitter was used to simulate variations in brightness, contrast, and saturation, making the model more resilient to lighting conditions. RandomErasing helped simulate occlusions. All inputs were normalized to match the statistics of the ImageNet dataset, ensuring compatibility with the pretrained backbone. This diverse augmentation suite significantly enriched the training set and helped the model generalize well to unseen data.

Results Analysis

The model achieved a best validation accuracy of approximately 64.3% and a test accuracy exceeding 65% in some runs. Training accuracy consistently surpassed 90%, indicating strong fitting ability. Compared to the models from Part 1 and 2, this model demonstrated substantial improvement in both speed of convergence and generalization. The gradual unfreezing strategy was particularly effective in avoiding catastrophic forgetting and enabled the model to adapt to CIFAR-100 without damaging the pretrained knowledge. However, I observed mild overfitting behavior after epoch 60 in some runs, suggesting that either early stopping or stronger regularization could further improve performance. Additional gains might be possible through ensembling or switching to larger pretrained backbones like ViT or EfficientNet.

Experiment Tracking Summary

I used Weights & Biases (WandB) to track all experiments, including model checkpoints, loss curves, and learning rate schedules. The OneCycle learning rate pattern was clearly visible and contributed to stable training dynamics. Runs such as sunny-plasma-6, rich-cherry-5, and pretty-resonance-3 were compared side-by-side in the UI, which made it easier to evaluate which changes were beneficial. Validation accuracy and loss curves demonstrated consistent improvement with minimal fluctuations, and the best model was automatically saved based on validation accuracy monitoring.