DS542 Midterm Report

Josh Yip (Kaggle Username: joshyippie)

March 2025

AI Disclosure Statement

This project involved the use of publicly available pretrained models, large-scale datasets, and computational tools that leverage artificial intelligence (AI) and machine learning (ML). The following outlines how AI was utilized during the course of this project:

- Training techniques such as label smoothing, augmentation strategies (e.g., RandAugment, Gaussian Blur, RandomErasing), learning rate scheduling (OneCycleLR), and early stopping were used to enhance model performance and stability. These were informed by GPT as possible increasers for accuracy.
- Debugging PyTorch-related errors, including issues with installation, module loading, environment activation, and runtime exceptions (e.g., torchvision version compatibility, PyTorch GPU utilization).
- Model architecture selection and recommendations (e.g., choosing ResNeXt50, modifying initial convolution layers for CIFAR-100 image sizes).
- Hyperparameter tuning strategy advice (learning rates, optimizer settings, OneCycleLR scheduling).
- Advanced data augmentation recommendations (RandAugment, Gaussian Blur, RandomErasing, ColorJitter).
- Best practices for experiment tracking using `Weights & Biases (wandb)`.
- Documentation and report-writing suggestions (structuring LaTeX sections clearly, writing comprehensive disclosures and analysis sections).

Explicit Breakdown of Code Authorship:

- **Written entirely by me:**
  - Core training and validation loops (`train` and `validate` functions).
  - Data loading, pre-processing, and dynamic normalization logic.

  - Early stopping logic and checkpoint.
  - Initial transformation steps (random cropping, resize, normalize).

- **Written with partial AI assistance:**
  - Selection and adaptation of the pretrained model (`get_model` function).
  - Advanced augmentation pipeline setup (RandAugment, RandomErasing).
  - Hyperparameter tuning guidance (learning rate, epochs, optimizer choice, learning rate scheduler settings).
  - Troubleshooting PyTorch environment issues (import errors, CUDA settings, cluster environment management).

- **Directly AI-suggested improvements:**
  - LaTeX document structuring, including detailed AI disclosure, design justification, results analysis, and ablation sections.
  - Suggested clarifications for training progress interpretation (accuracy expectations, early stopping recommendations).

Code Commenting and Documentation:

- All sections of the provided Python code contain clear, detailed comments describing their explicit purposes, as can be seen directly within the code files.
- Comments written by me explicitly indicate what each function and line does clearly, as recommended by GPT-4 to enhance readability.

Ethical and Fair Use: No confidential, proprietary, or sensitive data was used. All third-party models, datasets, and APIs were used in accordance with their respective licenses.

AI support contributed to rapid iteration, improved model interpretability, and enhanced documentation quality.

1. Experimental Overview

This report presents a series of experiments conducted on the CIFAR-100 image classification dataset. The goal was to progressively improve test accuracy using architectural enhancements, data augmentation, optimization techniques, and transfer learning.

2. Observations and Analysis of Results

2.1 Baseline (Simple CNN + SGD)

- **Optimizer:** SGD (learning rate = 0.001)
- **Test Accuracy:** 27.96%

2.2 Simple CNN + Adam

- **Optimizer:** Adam (learning rate = 1e-3)
- **Test Accuracy:** 40.44%
- **Analysis:** The initial simple CNN model established baseline performance. Switching from SGD to Adam resulted in substantial accuracy improvements (27.96% to 40.44%), highlighting the importance of adaptive optimization in early training stages.

2.3 ResNet18 (Pretrained, Frozen)

- **Test Accuracy:** 44.57%
- Even without fine-tuning, pretrained models showed improved performance over scratch models.

2.4 ResNet18 + Augmentation

- **Epochs:** 5
- **Test Accuracy:** 55.32%
- Stronger augmentation boosted generalization significantly.

2.5 DenseNet121 + Augmentation + Adam

- **Epochs:** 10
- **Test Accuracy:** 76.00%
- DenseNet provided better representational power and convergence.

2.6 ResNeXt50_32x4d + Label Smoothing + Advanced Augmentation

- **Epochs:** 50
- **Local Test Accuracy:** 82.71%
- **Leaderboard Submission Accuracy:** 54.48%

- Achieved the highest test accuracy locally; however, performance dropped on final submission, likely due to domain shift or overfitting to local validation.

3. Ablation Study

Model Variant	CIFAR-100 Test Accuracy
Simple CNN + SGD	27.96%
Simple CNN + Adam	40.44%
ResNet18 (pretrained, frozen)	44.57%
ResNet18 + Aug + Dynamic Norm	55.32%
DenseNet121 + Adam + 10 Epochs	76.00%
ResNeXt50_32x4d + Aug + Smoothing	**82.71%**

Table 1: Comparison of model variants and their corresponding performance on CIFAR-100.

4. Design Justification

This section outlines the rationale behind the final configuration used in the model pipeline for CIFAR-100 classification.

4.1 Model Architecture: ResNeXt50_32x4d

- **Why ResNeXt?** ResNeXt50_32x4d is a powerful extension of ResNet that employs grouped convolutions to increase model capacity without significant computational overhead.
- **Dynamic Normalization:** Normalizing using CIFAR-100 dataset-specific mean and standard deviation led to more stable training and better performance than using generic $(0.5, 0.5, 0.5)$.
- **Pretraining Benefits:** The model was initialized with `ImageNet1K_V1` pretrained weights to leverage transfer learning, which significantly boosts generalization performance on small datasets like CIFAR-100.
- **Input Resolution Compatibility:** CIFAR-100 uses 32x32 images, so the first convolution layer (originally designed for larger inputs) was modified with kernel size 3 and stride 1. Additionally, the aggressive downsampling via max-pooling was removed to preserve spatial details. Considered upscaling, but believed accuracy impact vs training time wasn't ideal for upscaling over freezing pooling layer.
- **Classifier Adaptation:** The final fully connected layer was replaced with a new head projecting to 100 classes.

4.2 Data Augmentation Strategy

- **Purpose:** Improve model robustness and generalization to unseen variations by simulating distortions during training.
- **Techniques Used:**
  - **Random Crop & Horizontal Flip** — Standard spatial augmentations for image recognition tasks.
  - **ColorJitter** — Simulates lighting variation.
  - **Gaussian Blur & RandomErasing** — Encourage spatial and texture invariance.
  - **RandAugment** — Plug-and-play policy to introduce strong randomized augmentations.
- **Validation & Test Sets:** No augmentation was applied to these sets to ensure fair performance evaluation.

4.3 Training Configuration

- **Optimizer:** SGD with momentum (0.8) and weight decay to promote generalization and stability.
- **Learning Rate Scheduler:** OneCycleLR was selected to allow for an aggressive warm-up and decay, which improves convergence speed and test performance.
- **Loss Function:** CrossEntropyLoss with label smoothing (0.05) helps prevent overconfidence and improves calibration.
- **Early Stopping:** A patience of 50 epochs with early stoppage was chosen, saving computation and reducing overfitting in cases of unproductive epochs.

4.4 Experiment Management

- **Tracking:** `Weights & Biases (wandb)` was used to log accuracy, loss, and learning rate in real-time, enabled rapid debugging and model comparison.
- **Reproducibility:** A fixed seed (42) was set, and all models were saved conditionally based on validation performance.

5. Conclusion

Each successive experiment demonstrated incremental improvements through better architecture choices, transfer learning, augmentation, and optimization

techniques. The final model, ResNeXt50_32x4d, achieved the best performance locally, though further tuning and regularization are necessary to bridge the gap between local and public leaderboard results.