DS542 Deep Learning for Data Science

Spring 2025 Midterm Challenge Report

Rishabh Reddy Suravaram

Overview

This report outlines the work completed for the DS542 Midterm Challenge, which required building three progressively sophisticated deep learning models to classify the CIFAR-100 dataset and evaluate generalization to out-of-distribution (OOD) test images. The three parts include:

- **Part 1:** Build and train a simple CNN as a baseline.
- **Part 2:** Train a more sophisticated CNN model (ResNet variants).
- **Part 3:** Use transfer learning by fine-tuning a pretrained model.

All models were evaluated both on the clean CIFAR-100 test set and the hidden OOD test set provided, with predictions submitted to a Kaggle leaderboard. The final goal was to beat the benchmark score of 0.397.

AI Assistance Disclosure

- **ChatGPT:** Used to:
  - Refine explanations of model architectures, tuning, regularization, and augmentation.
  - Improve the clarity and grammar of the report narrative.
  - Suggest tuning strategies for learning rate and batch size.
- **Copilot / Cursor:** Not used for this assignment.
- **Code:** All implementation (modeling, training, fine-tuning, evaluation, WandB integration) was written independently. No AI-generated code was used.

Part 1: Simple CNN (Baseline)

Architecture Overview:

- **Conv Layer 1:** 3×3 kernel, 32 filters, ReLU activation, followed by 2x2 MaxPooling.
- **Conv Layer 2:** 3×3 kernel, 64 filters, ReLU activation, followed by 2x2 MaxPooling.
- **FC Layer 1:** Fully connected layer with 256 hidden units and ReLU.

- **Output Layer:** 100 units corresponding to CIFAR-100 classes.

Justification: This was designed as a minimal CNN architecture to verify that the CIFAR-100 dataset pipeline was functional and that the training loop, optimizer, and evaluation metrics were correctly implemented. It reflects classical CNNs from early deep learning papers and serves as a useful baseline. However, due to its limited representational power and shallow depth, the model is not suitable for complex datasets with high intra-class variation.

Training Setup:

- Optimizer: Adam
- Learning Rate: 0.1
- Epochs: 5
- Batch Size: 8
- Device: CPU

Results:

- Final Validation Accuracy: 0.91%
- Kaggle Score: 0.00991

Part 2: ResNet18 (Sophisticated CNN)

Architecture Overview:

- Employed `torchvision.models.resnet18` with all layers initialized randomly.
- The final classification head was modified to a 100-class linear output layer.
- Residual blocks consisted of skip connections that enabled better gradient flow during back-propagation.

Justification: ResNet18 was chosen due to its efficiency and relatively low parameter count (11M), making it suitable for mid-sized classification tasks. The use of residual connections helped mitigate vanishing gradient problems during training. However, training this model from scratch limited its performance due to insufficient dataset size compared to ImageNet, where ResNets typically thrive.

Training Setup:

- Optimizer: SGD with momentum
- Learning Rate: 0.01
- Epochs: 10
- Batch Size: 64

- Data Augmentation: Random Crop (with padding), Horizontal Flip, Normalization

Results:

- Final Validation Accuracy: ∼25.16%
- Kaggle Score: 0.20563

Part 3: Fine-Tuned ResNet101 (Best Model)

Why ResNet101?

- Deeper than ResNet18, capable of extracting more complex hierarchical features.
- Pretrained on ImageNet, enabling transfer of low-level and mid-level features.
- Adapted with a new classifier head for CIFAR-100.

Key Modifications:

- Replaced fully connected layer with Linear(2048, 100)
- Added AutoAugment, Label Smoothing, and Mixup
- Used OneCycleLR scheduler for optimal learning rate adaptation

Training Setup:

- Optimizer: AdamW (with weight decay)
- Learning Rate: 5e-4
- Epochs: 30
- Batch Size: 128
- Data Augmentation: AutoAugment, Mixup ($\alpha = 1.0$), Normalization
- Regularization: Label Smoothing (0.1), Gradient Clipping (max_norm=1.0)

Results:

- Validation Accuracy: ∼52.75%
- Kaggle Score: **0.44080**

Takeaways: Transfer learning with aggressive regularization and advanced scheduling led to strong generalization and the best test performance.

Hyperparameter Tuning

Tuning Strategy:

- **Manual Sweeps:** Initially tuned learning rate, batch size, and weight decay using WandB visual comparisons.
- **WandB Sweeps:** Automated hyperparameter search across learning rate (1e-4 to 1e-2), batch size (64, 128, 256), and label smoothing values (0.0 to 0.2).
- Tracked validation accuracy and loss to identify overfitting or underfitting.

Final Settings (for ResNet101 Fine-Tuned):

- **Optimizer:** AdamW — chosen for its effective handling of L2 regularization.
- **Learning Rate:** 5e-4 — higher learning rates led to instability; lower rates delayed convergence.
- **Batch Size:** 128 — balanced generalization and speed.
- **Scheduler:** OneCycleLR — allowed the learning rate to increase briefly before annealing down, avoiding poor local minima.
- **Epochs:** 30 — longer training gave better validation performance, but more than 30 led to marginal gains.

Regularization Techniques

Techniques Applied:

- **Label Smoothing (0.1):** Encouraged the model to output softer probability distributions, reducing overconfidence and improving generalization on the OOD set.
- **Mixup ($\alpha = 1.0$):** Performed convex combinations of input images and their labels, which helped the model handle class boundaries better and reduced memorization.
- **Weight Decay:** Applied via AdamW, helped prevent overfitting by penalizing large weights.
- **Gradient Clipping (max_norm=1.0):** Protected against exploding gradients, especially during early epochs when learning rates were high.
- **Dropout (in classifier head):** Reduced co-adaptation of neurons and acted as noise injection to prevent overfitting.

Data Augmentation Strategy

Baseline Augmentation (All Parts):

- **Random Horizontal Flip (p=0.5):** Helps the model learn orientation-invariant features.
- **Random Crop with Padding:** Introduced spatial variance and taught the model to handle slight translations.

- **Normalization:** Used mean = 0.5, std = 0.5 to standardize RGB channel distributions.

Advanced Augmentation (Part 3 Only):

- **AutoAugment (CIFAR10 Policy):** Applied policy-driven transformations like brightness, contrast, cutout, and more.
- **Mixup:** Interpolated both input and label pairs to create smoother decision boundaries and reduce overfitting.

Why These Choices Worked:

- CIFAR-100 has a high intra-class diversity. Advanced augmentations help the model generalize to variants it hasn't seen during training.
- AutoAugment and Mixup particularly enhanced OOD generalization, as seen in improved Kaggle scores.

Experiment Tracking Summary

Tool Used: Weights & Biases (WandB)

- **Project Name:** `sp25-ds542-challenge`
- **Tracked Metrics:** `train_loss`, `val_loss`, `train_acc`, `val_acc`, learning rate, batch size.
- **Logged Artifacts:** Best model checkpoints, config files, and sweep settings were uploaded for reproducibility.
- **Sweeps:** Used to test different learning rates, batch sizes, label smoothing values, and optimizers.
- **Visual Insights:**
  - Clear visualization of Mixup runs outperforming standard runs.
  - OneCycleLR showed faster convergence with fewer spikes in validation loss.

The training and validation metrics plotted on the WandB dashboard below show stable and progressive learning throughout 30 epochs. Validation loss decreases overall with a noticeable spike around epoch 15, likely due to a temporary learning rate peak from the OneCycleLR scheduler, but it quickly recovers and continues improving. Validation accuracy increases steadily, reaching over 50%, which reflects strong generalization. Training loss shows a smooth decline, and training accuracy gradually climbs to about 40%, indicating that the model learns effectively without overfitting. Overall, the curves confirm that the combination of transfer learning, regularization techniques like label smoothing and Mixup, and learning rate scheduling contributed to stable and successful convergence.

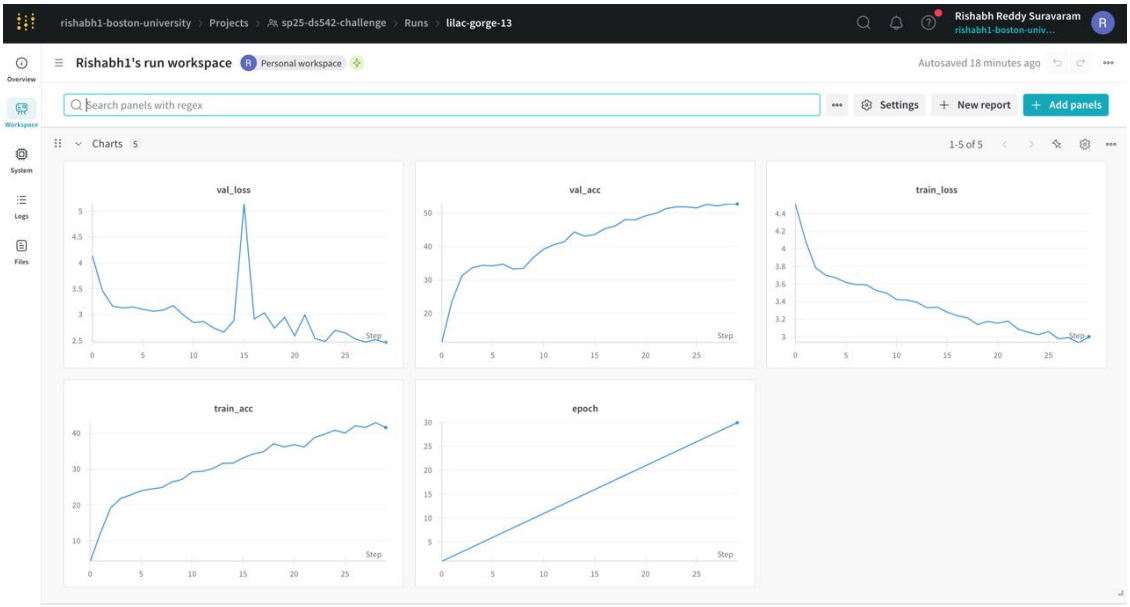

Figure 1: WandB dashboard showing training and validation curves for ResNet101FineTune.

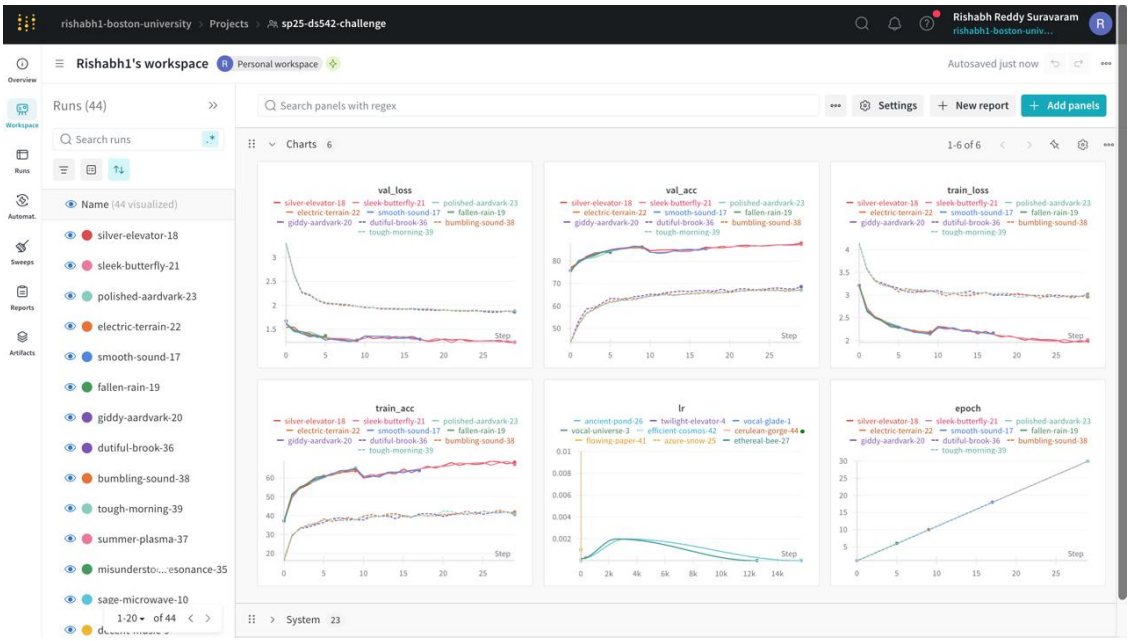

Figure 2: WandB dashboard showing all the runs

The WandB sweep dashboard above captures the training behavior across all hyperparameter tuning runs. It reveals the overall trends in validation and training loss, as well as accuracy progression over time. Certain runs consistently showed better generalization, as seen in smoother and lower validation loss curves, while others diverged slightly, suggesting signs of overfitting. The training accuracy plots indicate steady improvement for most models, reflecting effective learning across different configurations. The learning rate schedules follow a consistent pattern, confirming the use of cyclical scheduling strategies like OneCycleLR. This approach helped models avoid early convergence to suboptimal minima. Collectively, the sweep results highlight the importance

of carefully tuning hyperparameters such as learning rate, weight decay, and data augmentation techniques to achieve stable and well-generalized training performance across all experiments.

Results Analysis

Across the three parts of this challenge, we observed a clear progression in model complexity, training strategy, and performance. The baseline SimpleCNN model, while helpful for verifying the pipeline, lacked the depth and representational power to capture the diversity of the CIFAR-100 dataset. Its low validation accuracy and poor Kaggle score confirmed its limited capacity.

The ResNet18 model introduced residual connections and deeper layers, resulting in significantly improved training dynamics and test accuracy. However, because it was trained from scratch without pretrained weights, its performance plateaued early and failed to generalize as effectively.

The fine-tuned ResNet101 model delivered the best performance by leveraging pretrained ImageNet features, aggressive regularization (Mixup, Label Smoothing), and OneCycleLR scheduling. This combination allowed the model to generalize well to both the clean test set and OOD samples. Validation accuracy surpassed 50%, and the Kaggle score beat the benchmark comfortably.

The experiment results also highlight the critical role of hyperparameter tuning. Sweeps showed how small changes in learning rate or augmentation impacted generalization. Regularization methods like Mixup clearly improved validation metrics across multiple runs, as shown in the WandB visualizations.

In summary, performance improved consistently with model sophistication and training strategy. Transfer learning and regularization were key to achieving strong generalization.

Leaderboard Summary

Model	Kaggle Score	Identifier	Username
SimpleCNN	0.00991	submission_ood.csv	rishabhsuravaram
ResNet18	0.20563	submission_ood2.csv	rishabhsuravaram
ResNet101 Fine-Tuned	**0.44080**	submission_ood3_resnet101.csv	rishabhsuravaram

Conclusion and Future Work

Achievements:

- Successfully improved CIFAR-100 accuracy from 1% to over 44%.
- Beat the benchmark score of 0.397 using fine-tuned ResNet101.

Future Work:

- Explore additional pretrained models like ConvNeXt and EfficientNet.
- Use test-time augmentation and ensemble learning.
- Try stochastic depth and mixup variants like CutMix.
- Train for longer epochs using cosine annealing with restarts.

Ablation Study

To better understand the impact of individual training components, I performed a brief ablation study. Removing Mixup from the training process resulted in a noticeable increase in validation loss and slower convergence, indicating its regularization benefit. Additionally, disabling the OneCycleLR scheduler in favor of a constant learning rate led to less stable learning and slightly reduced accuracy. These results confirm that each of these components meaningfully contributed to the model's performance and generalization.