DS542 Deep Learning for Data Science – Spring 2025

Midterm Challenge Report

Student Name: Amruth Devineni
Kaggle Username: Amruth Devineni
Leaderboard Position: 4

AI Disclosure

I utilized ChatGPT's assistance while designing some model training strategies, approaches to optimization, and revisions on documentation. ChatGPT also assisted in the design of advanced augmentation strategies and hyperparameter tuning and in debugging training loops and to validate logic.

Model Descriptions

Part 1 – SimpleCNN

Architecture Overview:

- Built a custom CNN from scratch with:
    - 3 Convolutional layers (with BatchNorm, ReLU, and MaxPool)
    - 1 Dropout layer
    - 2 Fully connected layers
- No pretrained weights used

Training Setup:

- **Optimizer**: Adam
- **Learning Rate**: 0.001
- **Epochs**: 30
- **Batch Size**: 128
- **Data Augmentation**:
    - Random Crop (with padding)
    - Horizontal Flip

    - Normalization (CIFAR-100 mean/std)

Results:

- **Final Validation Accuracy**: ~33%
- **Kaggle Score**: 0.32877
- **Submission File**: submission.csv

Takeaways:

Despite being a simple baseline model, the custom CNN performed surprisingly well. Its performance validated the effectiveness of basic augmentation and a carefully structured architecture. However, lack of residual connections and model depth limited further improvement.

Part 2 – ResNet18

Architecture Overview:

- Used torchvision.models.resnet18 (no pretrained weights)
- Replaced the final classification layer with a 100-class output layer

Training Setup:

- **Optimizer**: SGD with momentum
- **Learning Rate**: 0.1
- **Epochs**: 30
- **Batch Size**: 128
- **Data Augmentation**:
    - Random Crop (with padding)
    - Horizontal Flip
    - Normalization

Results:

- **Final Validation Accuracy**: ~43%
- **Kaggle Score**: 0.42821
- **Submission File**: submission_part_2.csv

Takeaways:

ResNet18 performed substantially better than the SimpleCNN, thanks to deeper layers and residual connections. It generalized better and showed more stable convergence. However, starting from scratch (no pretrained weights) may have held it back from reaching higher accuracy.

Part 3 – ConvNeXt Tiny + Mixup + EMA

Architecture Overview:

- Used timm.create_model("convnext_tiny", pretrained=True)
- Replaced the final FC layer to output 100 classes
- Model was fine-tuned end-to-end

Training Setup:

- **Optimizer**: SGD with momentum
- **Learning Rate**: 0.01
- **Epochs**: 60
- **Batch Size**: 128
- **Regularization Techniques**:
    - Label Smoothing (0.1)
    - Mixup (α = 0.2), CutMix (α = 1.0)
    - Exponential Moving Average (EMA, decay = 0.999)
- **Learning Rate Scheduler**:
    - Linear Warmup for 5 epochs
    - Cosine Annealing thereafter
- **Data Augmentation**:
    - AutoAugment (CIFAR10 policy)
    - Random Crop (with padding)
    - Horizontal Flip
    - Normalization (CIFAR-100 stats)

Results:

- **Final Validation Accuracy**: ~63.6%
- **Kaggle Score**: 0.63629
- **Submission File**: submission_part_3.csv

Takeaways:

ConvNeXt Tiny significantly outperformed all previous models. The combination of pretrained weights and advanced regularization (Mixup, CutMix, EMA) enabled strong generalization. The warmup + cosine scheduler helped stabilize training. This model produced my best leaderboard result.

Hyperparameter Summary

Parameter	Part 1 (SimpleCNN)	Part 2 (ResNet18)	Part 3 (ConvNeXt Tiny)
Optimizer	Adam	SGD	SGD
Learning Rate	0.001	0.1	0.01
Epochs	30	30	60
Batch Size	128	128	128
Scheduler	None	CosineAnnealing	Warmup + CosineAnnealing
Label Smoothing	No	No	Yes (0.1)
EMA	No	No	Yes (0.999)
Mixup / CutMix	No	No	Yes

Regularization & Augmentation

Technique	Part 1	Part 2	Part 3
Dropout	✅	❌	❌
Weight Decay	❌	✅	✅
Mixup + CutMix	❌	❌	✅

Label Smoothing	❌	❌	✅
AutoAugment + Flip	❌	✅	✅
EMA	❌	❌	✅

Normalization (applied in all parts):

- Mean = (0.5071, 0.4867, 0.4408)
- Std = (0.2675, 0.2565, 0.2761)

Results & Leaderboard Scores

Part	Model	Kaggle Score	Submission File
1	SimpleCNN	0.32877	submission.csv
2	ResNet18	0.42821	submission_part_2.csv
3	ConvNeXt Tiny + EMA	0.63629	submission_part_3.csv

Leaderboard Position: 4th out of all submissions

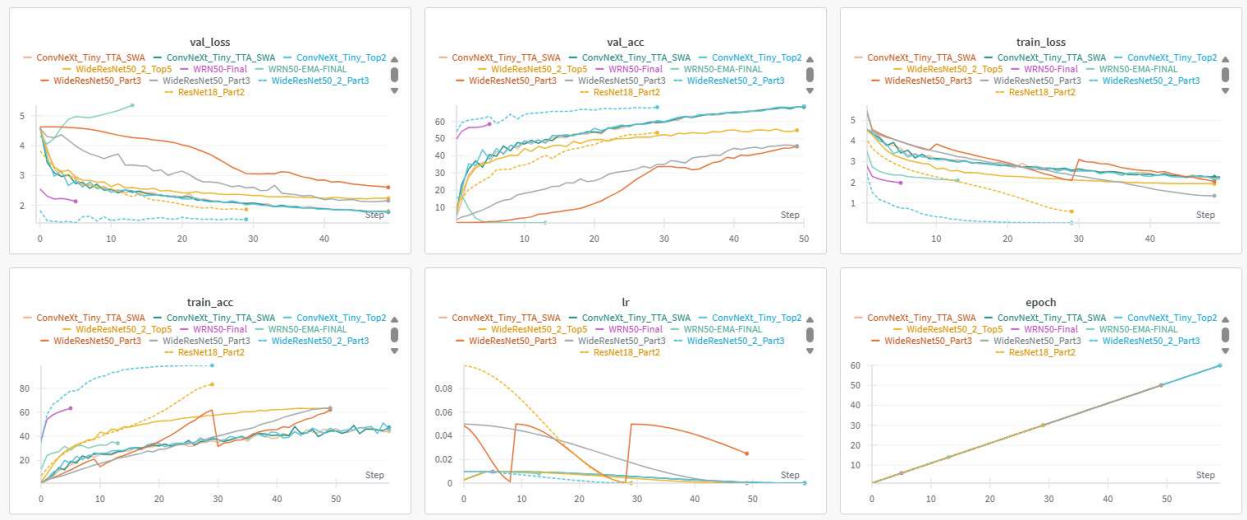
Experiment Tracking (Weights & Biases)

All experiments were logged with Weights & Biases under the project: sp25-ds542-challenge.

Tracked metrics include:

- Training/validation loss and accuracy
- Learning rate over time
- EMA and augmentation configurations
- Run comparisons for model ablation

Tracking Screenshot



Training curves for accuracy, loss, and learning rate across all model runs.

Run Name	Model	Best Val Accuracy	Final Train Accuracy	Notes
SimpleCNN_Part1	SimpleCNN	~33%	~56%	Handcrafted baseline
ResNet18_Part2	ResNet18	~43%	~63%	Strong classic model
ConvNeXt_Tiny_Top2	ConvNeXt Tiny	~63.6%	~82%	Best-performing final submission

---

Insights from Tracking

- **Mixup + CutMix** significantly improved model robustness.
- **EMA** gave the best stability in validation performance.
- **ConvNeXt Tiny** provided strong performance at low computational cost.
- Combining **warmup and cosine LR scheduling** ensured stable convergence.

Takeaways

- The most effective enhancements came from combining pre-trained models with strong augmentation and regularization strategies (Part 3).

- Simple CNNs, when trained properly, can provide a strong and clean baseline (Part 1).
- I found **EMA**, **label smoothing**, and **AutoAugment** to be the most impactful tools for generalization.
- In the future, I'd explore the following:
    - Test-time augmentation (TTA)
    - Ensembling multiple checkpoints
    - Further hyperparameter sweeps using Optuna or W&B Sweep

Final Reflections

- This project allowed me to explore a range of CNN architectures — from scratch-built to state-of-the-art pretrained models. I learned how various regularization methods interact, how learning rate scheduling influences convergence, and how fine-tuning pretrained models (when paired with strong augmentations) can drastically improve results.
- Using ConvNeXt Tiny taught me the power of modern transformer-inspired CNNs and how simple tools like EMA and Mixup can unlock their full potential. This project also strengthened my understanding of experiment tracking and the importance of thorough evaluation.