MILS Assignment I Report

Ian Ku

GitHub Repository: `https://github.com/IanKu304/2025S_DL_HW1_RE6131066_Ianku`

Dataset

The mini-ImageNet dataset was used for all experiments. It consists of 50 classes with training, validation, and test sets defined in `train.txt`, `val.txt`, and `test.txt`. All images were resized to 32x32 or 224x224 as appropriate for the network architecture.

Problem A: Dynamic Convolution Module

Design Objective

We designed a convolutional module that:

- Handles arbitrary input channels (e.g., RGB, RG, R)
- Is spatial size invariant
- Dynamically generates convolution kernels

Training Configuration

All models were trained using the same training protocol to ensure fair comparison. The key hyperparameters and strategies are summarized below:

- **Optimizer:** Adam optimizer with an initial learning rate of 1×10^{-3}
- **Loss Function:** Cross-Entropy Loss for multi-class classification
- **Batch Size:** 10
- **Max Epochs:** 40 epochs
- **Early Stopping:** Enabled, with a patience of 5 epochs based on validation loss in Section A and 10 in Section B
- **Learning Rate Scheduler:** Not used
- **Input Size:**

  - 32×32 for DynamicCNN and BaselineCNN (Problem A)
  - 224×224 for ResNet34 and custom 2/4-layer CNNs (Problem B)

- **Channel Robustness:** For Problem A, optional `RandomChannelDrop` was used with probability 0.3 to simulate partial channel inputs (e.g., RG, R)
- **Weight Initialization:** PyTorch default initialization

Architectures

DynamicCNN v1

```
Input Channels: 1{3
→ MLP (input channel as condition) → Dynamic Kernel
→ Conv2D → BN → ReLU → GAP → FC(32→50)
```

DynamicCNN v2

```
Input Channels: 1{3
→ CNN-based kernel generator → Dynamic Kernel
→ Conv2D → BN → ReLU
→ Conv(32→64) → BN → ReLU → GAP
→ FC(64→128) → ReLU → FC(128→50)
```

Baseline CNN

```
Conv(3→32, 3x3) → ReLU → BN → MaxPool
Conv(32→64, 3x3) → ReLU → BN → GAP
FC(64→50)
```

Results on Test Set (32x32)

Without Channel Dropout:

Model	Accuracy	FLOPs	Params
DynamicCNN v1	0.2422	180.39 KMac	31.57 K
DynamicCNN v2	0.1422	19.49 MMac	63.31 K
Baseline CNN	0.2844	5.95 MMac	22.83 K

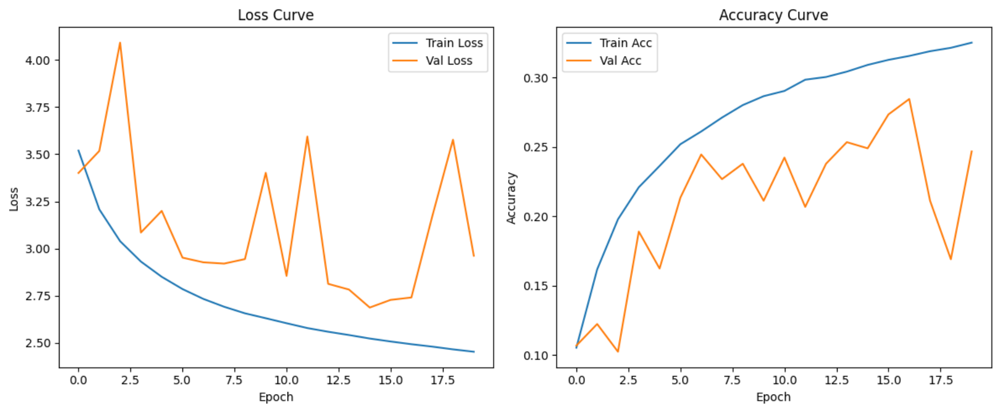

Figure 1: Training plot for Baseline CNN without random channel drop

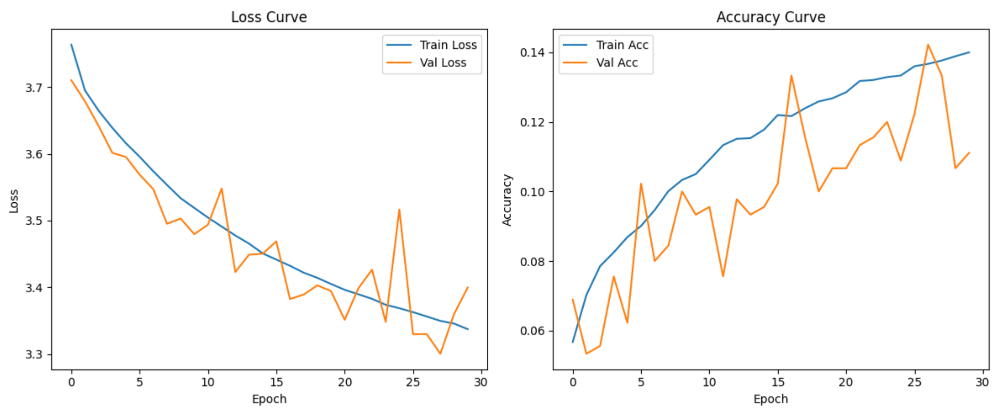

Figure 2: Training plot for DynamicCNN v1 without random channel drop

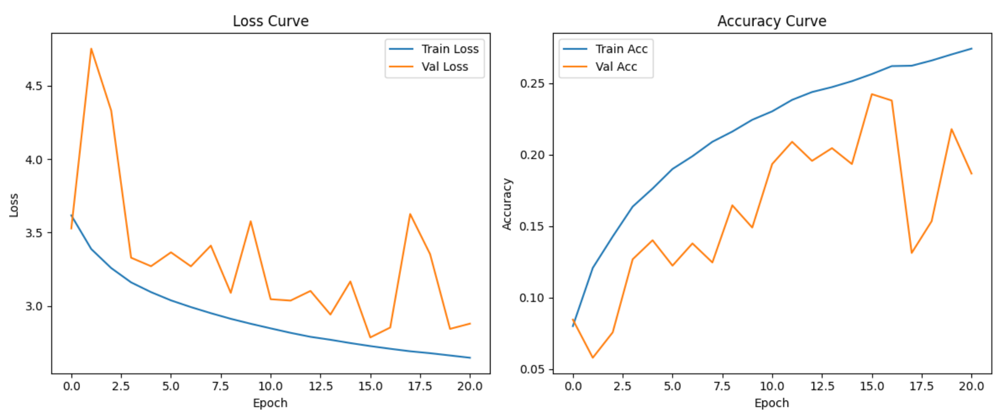

Figure 3: Training plot for DynamicCNN v without random channel drop

With RandomChannelDrop:

Model	Accuracy	FLOPs	Params
DynamicCNN v1	0.1244	180.39 KMac	31.57 K
DynamicCNN v2	0.2244	19.49 MMac	63.31 K
Baseline CNN	0.2422	5.95 MMac	22.83 K

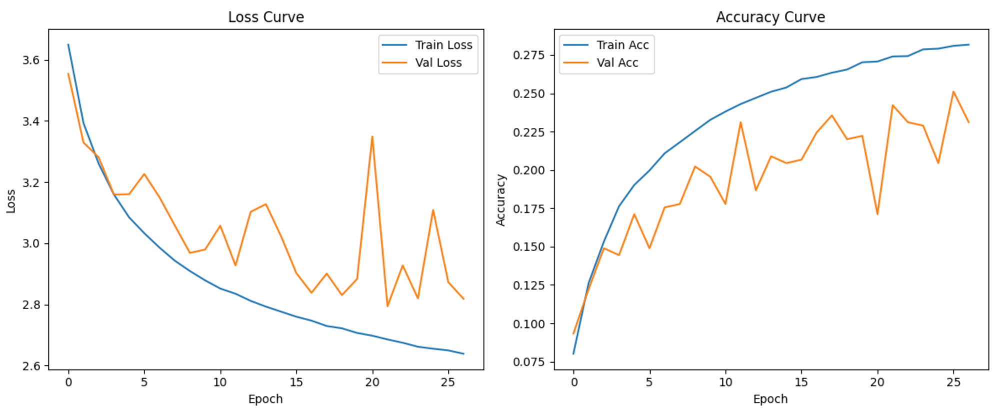

Figure 4: Training plot for Baseline CNN with random channel drop

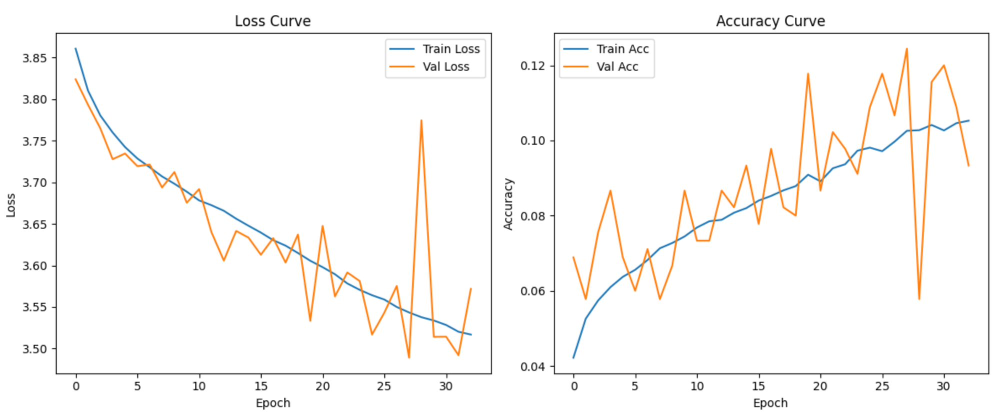

Figure 5: Training plot for DynamicCNN v1 with random channel drop

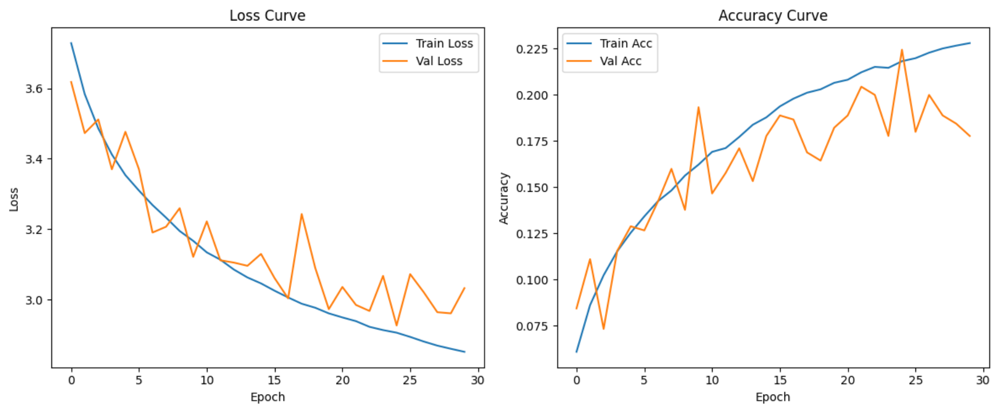

Figure 6: Training plot for DynamicCNN v2 with random channel drop

Problem B: Two-Layer Network

Design Goal

Design a 2–4 effective layer network achieving at least 90% of ResNet34's performance on mini-ImageNet (resized to 224x224).

Architectures

ResNet34 (baseline)

```
Input → Conv(7x7, 64) → MaxPool
[3x BasicBlock(64)]
[4x BasicBlock(128)]
[6x BasicBlock(256)]
[3x BasicBlock(512)]
→ GAP → FC(50)
```

Simple2LayerCNN

```
Conv(3→32, 7x7) → BN → ReLU
Conv(32→64, 5x5) → BN → ReLU → SEBlock(64) → MaxPool
Conv(64→128, 5x5) → BN → ReLU → SEBlock(128) → GAP
FC(128→128) → ReLU → FC(128→50)
```

Simple4LayerCNN

```
Conv(3→32, 3x3) → BN → ReLU
Conv(32→64, 3x3) → BN → ReLU
Conv(64→128, 3x3) → BN → ReLU
Conv(128→128, 3x3) → BN → ReLU
GAP → FC(128→128) → ReLU → FC(128→50)
```

Results (224x224)

Model	Accuracy	FLOPs	Params	Train Time
ResNet34	0.5667	3.68 GMac	21.31 M	234.85 min
Simple2LayerCNN	0.4400	43.66 MMac	237.75 K	451.15 min
Simple4LayerCNN	0.4000	285.83 MMac	301.11 K	470.57 min
SE2LayerCNN	0.4711	43.82 MMac	240.51 K	843.25 min

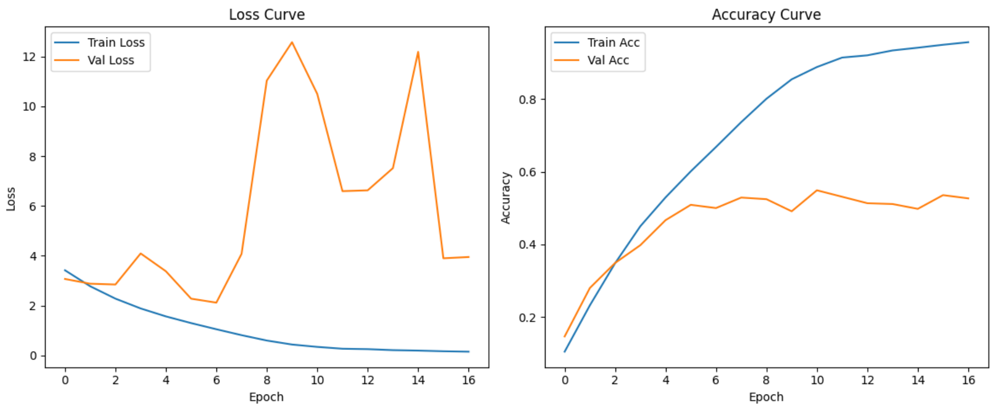

Figure 7: Training plot for ResNet34 model

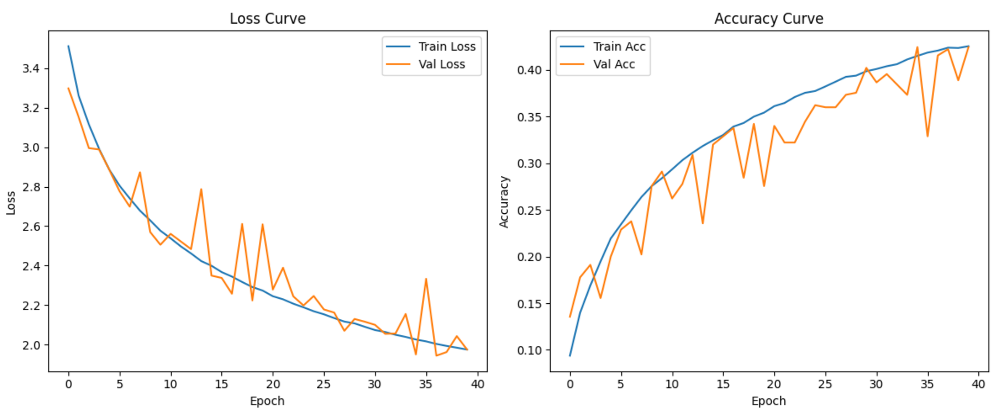

Figure 8: Training plot 1 for Attention model

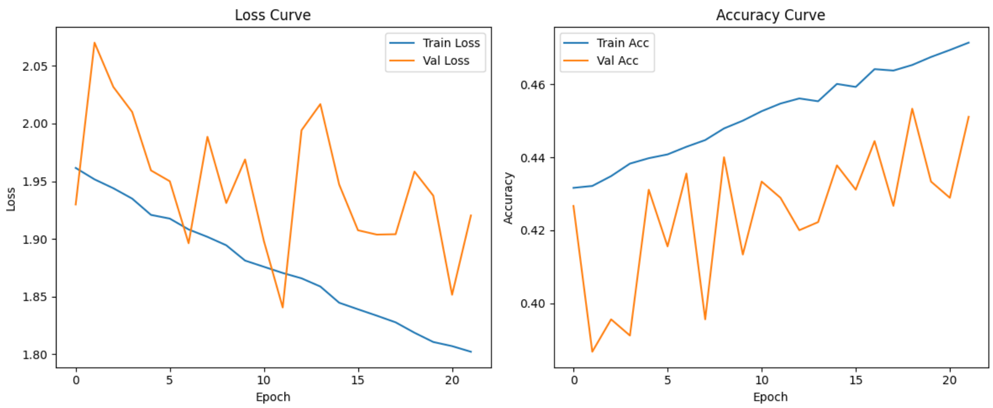

Figure 9: Training plot 2 for Attention model

results/Attention3.png

Figure 10: Training plot 3 for Attention model

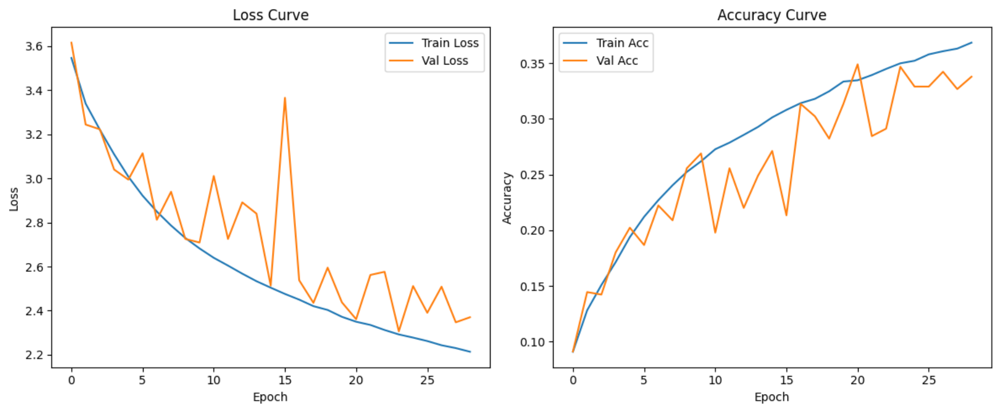

Figure 11: Training plot 1 for S2CNN

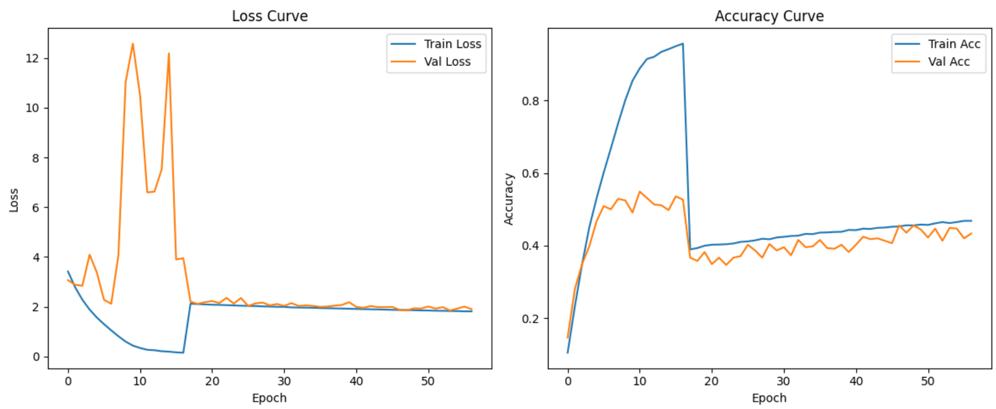

Figure 12: Training plot 2 for S2CNN

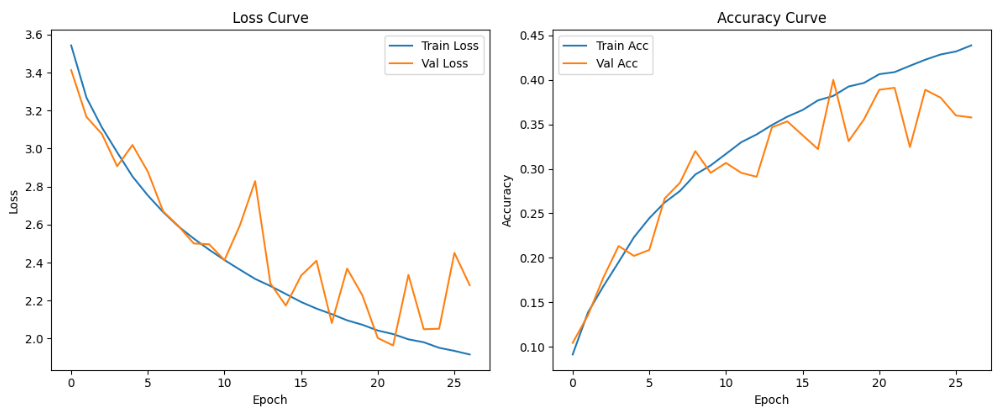

Figure 13: Training plot for S4CNN

Summary and Insights

Problem A

- **DynamicCNN v1** achieves best result without channel dropout.
- **DynamicCNN v2** is more robust under channel variation.
- **BaselineCNN** provides stable performance overall.

Problem B

- **Simple2LayerCNN** achieves 77.6% of ResNet34.
- **SE2LayerCNN** boosts accuracy with minor cost.
- **Simple4LayerCNN** deeper model did not improve performance.

Appendix