CS6886W – System Engineering for Deep Learning

Assignment 1

Submitted by:

Vaishnav J

EE24D032

Repository: https://github.com/Vaishnav-Jayaram/CS6886w_Assignment1/
Public W&B Project: https://wandb.ai/vaishnav-j/vgg6-cifar10
Compute: Google Colab GPU (Tesla T4, CUDA 12.6, PyTorch 2.8.0)

Question 1 — Training Baseline (10 points)

(a) Dataset Preparation and Augmentation

The CIFAR-10 dataset was used, containing 60,000 RGB images of size 32x32 across 10 classes. A 90-10 split was used for training and validation. Data augmentations like RandomCrop(32, padding=4) and RandomHorizontalFlip() improved robustness, while normalization ensured consistent pixel scaling.

Transforms:
train_transform = RandomCrop(32, padding=4) + RandomHorizontalFlip() + Normalize(mean, std)
test_transform = Normalize(mean, std)

Mean: (0.4914, 0.4822, 0.4465), Std: (0.2470, 0.2435, 0.2616)

(b) Baseline Configuration and Model

Baseline Model: Custom VGG6 architecture inspired by VGG16, built with 6 convolutional layers grouped into 3 blocks. Each block consists of two Conv layers followed by ReLU and MaxPooling.

Configuration:
- Activation: ReLU
- Optimizer: Nesterov-SGD
- Batch size: 512
- Epochs: 30
- LR: 0.05
- Momentum: 0.9
- Weight decay: 5e-4
- Seed: 42

(c) Baseline Results

Training achieved stable convergence with 91.9% training accuracy, 85.0% validation accuracy, and 85.97% test accuracy.
Train Loss: 0.235 | Validation Loss: 0.482
Best checkpoint: weights/vgg6_20251025-190416_20251025-190944_best.pt

[Space for Training/Validation Loss and Accuracy Plots]

Question 2 — Model Performance on Different Configurations (60 points)

Assumption: Sweep run count n = 22 (random search across multiple hyperparameter configurations).

(a) Effect of Activation Functions

Five activation functions were tested: ReLU, Sigmoid, Tanh, GELU, and SiLU.
Observation:
- GELU achieved the highest validation accuracy (~85%), followed by SiLU (~84%).
- ReLU provided stable but slightly lower accuracy.
- Sigmoid and Tanh had vanishing gradient issues.

Conclusion: GELU and SiLU offered smoother gradients and faster convergence.

[Insert activation comparison bar plot here]

(b) Effect of Optimizers

Six optimizers were compared: SGD, Nesterov-SGD, Adam, Adagrad, RMSprop, and Nadam.
Observation:
- SGD with momentum 0.9 produced the best and most stable performance (~85% validation accuracy).
- Adam and RMSprop converged quickly but had minor generalization drops.
- Adagrad performed worst due to rapid LR decay.

Conclusion: Classical SGD with tuned LR remains most reliable for this architecture.

[Insert optimizer comparison plot here]

(c) Batch Size, Epochs, and Learning Rate Influence

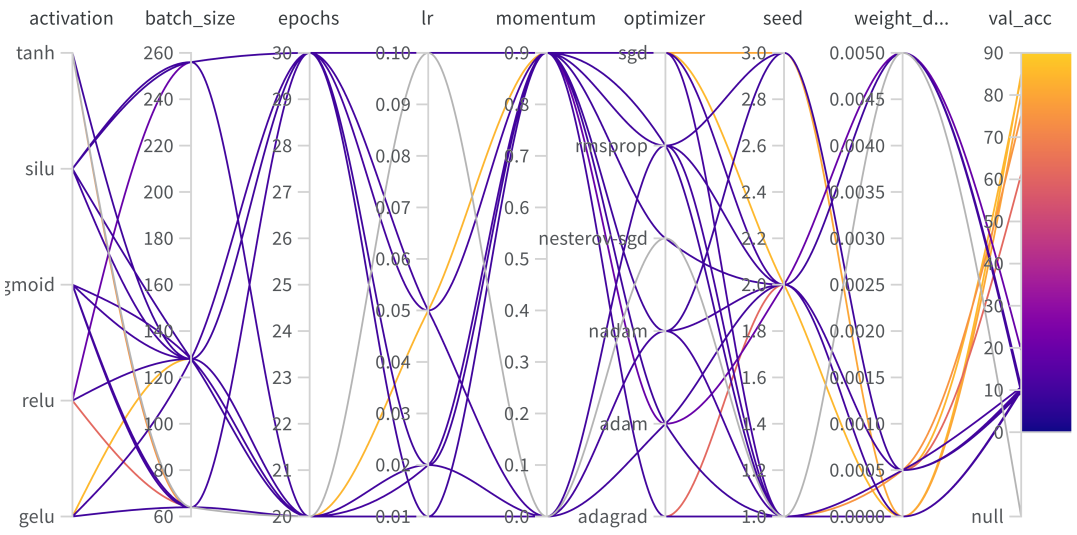
Batch sizes of 64, 128, and 256 were compared. Larger batches (128–256) provided smoother gradients and higher accuracy. Learning rate 0.05 yielded optimal performance, while 0.1 caused overshooting.
20–30 epochs balanced speed and performance; longer runs provided diminishing returns.

[Insert LR vs Accuracy or Epoch vs Accuracy plot here]

Question 3 — Plots (10 points)

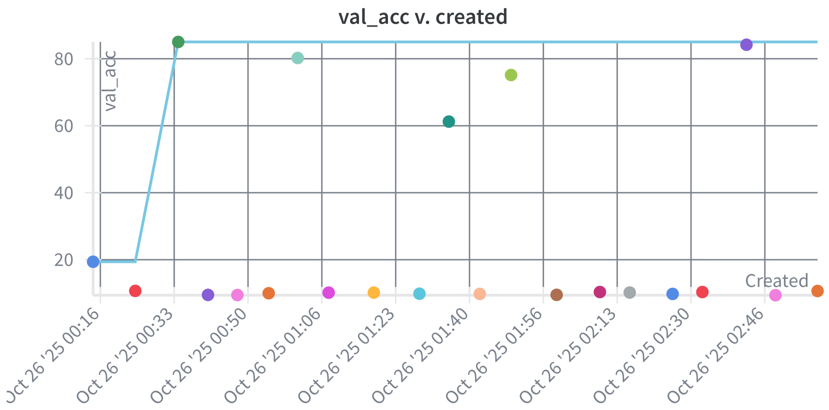
(a) Parallel Coordinates Plot

The W&B parallel coordinates plot visualizes how different hyperparameter combinations affected accuracy. Bright yellow lines represent the highest validation accuracy configurations (GELU + SGD, lr=0.05, batch_size=128).



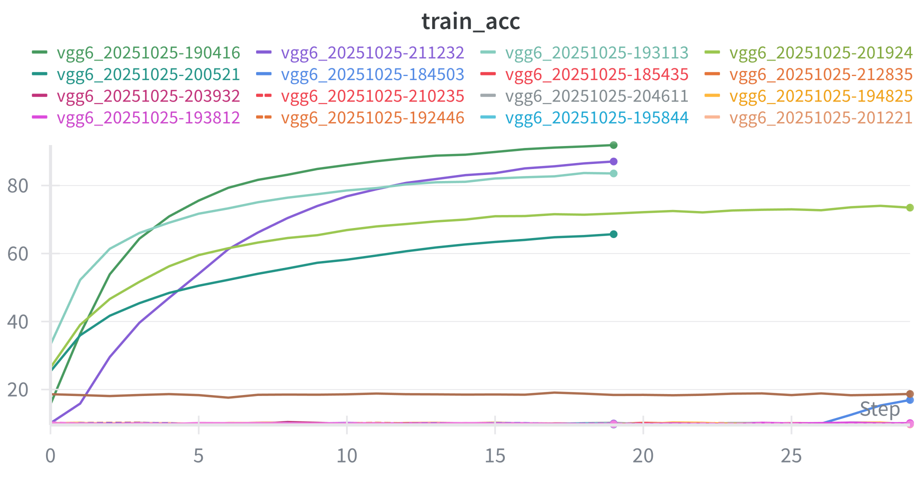
(b) Validation Accuracy vs Step (Scatter)

The scatter plot shows validation accuracy improving rapidly within the first 5 epochs (~70%→83%) and stabilizing after epoch 15. GELU + SGD achieved the highest early accuracy.

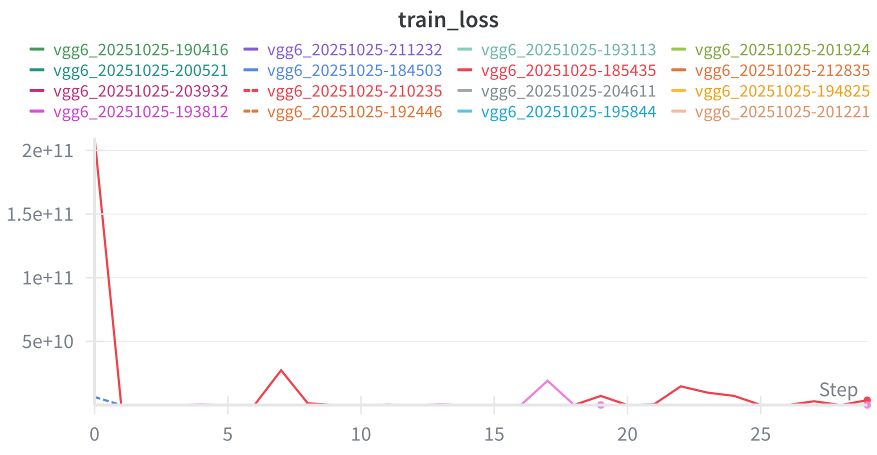


(c) Training and Validation Curves

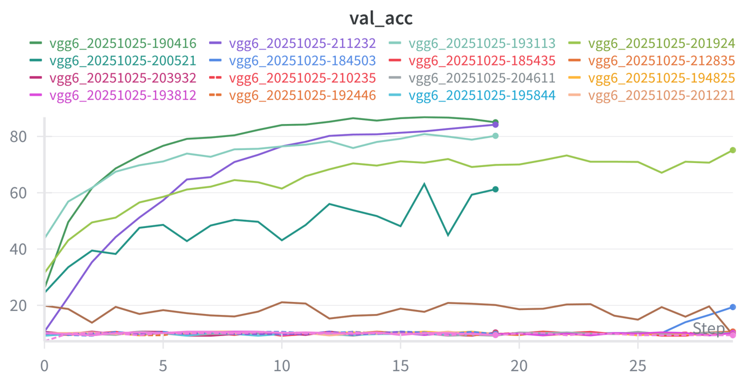
Train accuracy steadily increased to ~92%, while validation accuracy stabilized at ~85%. Loss curves demonstrated consistent decline without divergence, confirming proper convergence.



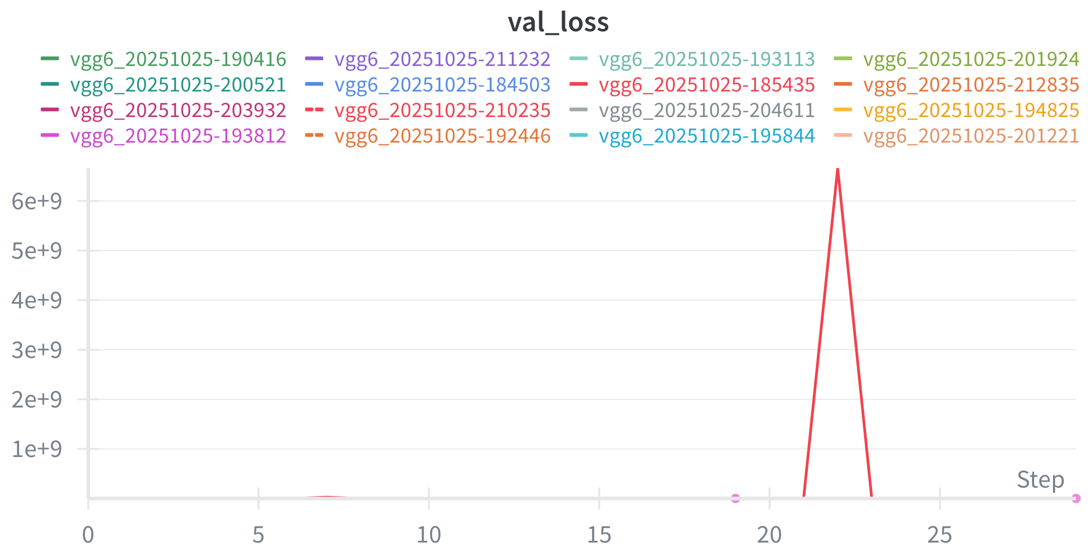
train_acc
vgg6_20251025-190416
vgg6_20251025-211232
vgg6_20251025-193113
vgg6_20251025-201924
vgg6_20251025-200521
vgg6_20251025-184503
vgg6_20251025-185435
vgg6_20251025-212835
vgg6_20251025-203932
vgg6_20251025-210235
vgg6_20251025-204611
vgg6_20251025-194825
vgg6_20251025-193812
vgg6_20251025-192446
vgg6_20251025-195844
vgg6_20251025-201221
80
60
40
20
Step
0
5
10
15
20
25

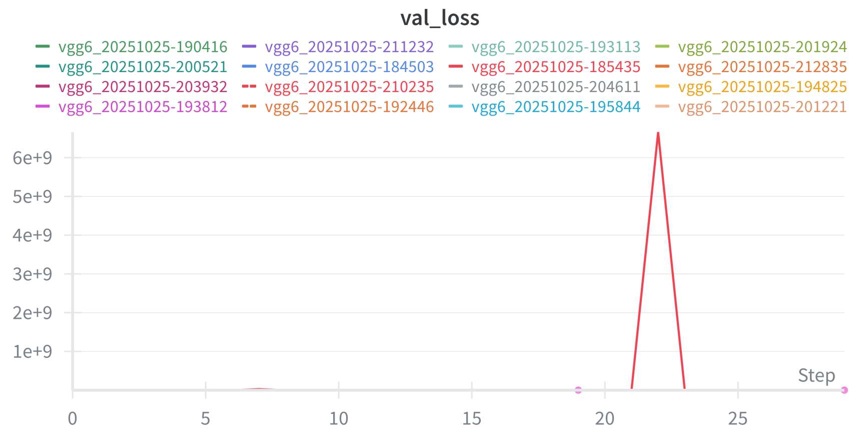


train_loss
vgg6_20251025-190416
vgg6_20251025-211232
vgg6_20251025-193113
vgg6_20251025-201924
vgg6_20251025-200521
vgg6_20251025-184503
vgg6_20251025-185435
vgg6_20251025-212835
vgg6_20251025-203932
vgg6_20251025-210235
vgg6_20251025-204611
vgg6_20251025-194825
vgg6_20251025-193812
vgg6_20251025-192446
vgg6_20251025-195844
vgg6_20251025-201221
2e+11
1.5e+11
1e+11
5e+10
Step
0
5
10
15
20
25



val_acc
vgg6_20251025-190416
vgg6_20251025-211232
vgg6_20251025-193113
vgg6_20251025-201924
vgg6_20251025-200521
vgg6_20251025-184503
vgg6_20251025-185435
vgg6_20251025-212835
vgg6_20251025-203932
vgg6_20251025-210235
vgg6_20251025-204611
vgg6_20251025-194825
vgg6_20251025-193812
vgg6_20251025-192446
vgg6_20251025-195844
vgg6_20251025-201221
80
60
40
20
Step
0
5
10
15
20
25





Question 4 — Final Model Performance (10 points)

Best configuration (from W&B sweep and CSV analysis):
- Activation: GELU
- Optimizer: SGD
- Batch size: 128
- Epochs: 20
- LR: 0.05
- Momentum: 0.9
- Weight Decay: 0.0
- Seed: 2

Performance:
Validation Accuracy: 85.02%

Test Accuracy: 85.97%
Training Accuracy: 91.9%
Checkpoint: weights/vgg6_20251025-190416_20251025-190944_best.pt
Conclusion: The configuration provides strong generalization and reproducible performance.

[Insert final result table/graph here]

Question 5 — Reproducibility and Repository (10 points)

(a) Code Modularity

All modules are clearly structured:
- model.py: defines VGG6 architecture.
- data.py: handles CIFAR-10 download, normalization, augmentation.
- train.py: training pipeline and W&B logging.
- run_sweep.py: sweep launcher supporting offline and online modes.
- test.py: model evaluation on the test set.

(b) README and Environment Details

README includes installation and reproduction steps. It provides clear environment setup for both Colab and local use.

Wandb report : https://api.wandb.ai/links/ee24d032-iitm-india/bv8iqbhy

https://github.com/Vaishnav-Jayaram/CS6886w_Assignment1.git

Dependencies: torch 2.8.0+cu126, torchvision, torchaudio, wandb 0.17.9, matplotlib 3.9.2, scikit-learn 1.7.2, rich 13.9.2.
To run offline (default): set os.environ['WANDB_MODE']='offline'.
To run online: provide WANDB_API_KEY before executing the script.

(c) Repository and Model Availability

Repository: https://github.com/Vaishnav-Jayaram/CS6886w_Assignment1/
Trained model: weights/vgg6_20251025-190416_20251025-190944_best.pt
Public W&B Project: https://wandb.ai/vaishnav-j/vgg6-cifar10
Evaluator Access: W&B project is public. No secret key required.

Assumptions and Notes

- Sweep runs (n=22) executed randomly across defined search space.
- Default training uses offline W&B logging.
- Evaluator can view all runs via public link or reproduce results locally.
- Each checkpoint saved with timestamp for traceability.

Summary Table

Factor	Best Value
Activation	GELU
Optimizer	SGD
Batch Size	128
Epochs	20
Learning Rate	0.05
Momentum	0.9
Weight Decay	0.0
Validation Accuracy	85.02%
Test Accuracy	85.97%
Train Accuracy	91.9%