Got you. Here’s a single, clean runbook for **every question and sub-task (Q1–Q5)**. It works the same on your laptop or Colab; I’ll show both paths.

**Project root assumption:** you unzipped the bundle and are in the folder vgg6\_cifar\_full/.  
Use python -m ... so imports work from anywhere.

**Pre-flight (once)**

**Local**

cd vgg6\_cifar\_full

python -m venv .venv && source .venv/bin/activate # optional but recommended

pip install -r requirements.txt

**Colab**

# Make sure you're in the project root after unzipping/cloning

%cd /content/vgg6\_cifar\_full

!pip install -r requirements.txt

(For W&B plots)

import wandb, os

# os.environ["WANDB\_API\_KEY"] = "<your\_key>" # optional; or run wandb.login() interactively

wandb.login()

**Q1. Training Baseline (10 pts)**

**Q1(a) Prepare CIFAR-10 with normalization & augmentation**

Run the baseline entry script; it prints the transforms and uses proper normalization.

**Local**

python -m vgg6\_cifar.scripts.train\_baseline \

--data\_dir ./data \

--out\_dir ./runs/baseline \

--epochs 60 --batch\_size 128 --lr 0.1 --optimizer sgd --momentum 0.9 \

--weight\_decay 5e-4 --label\_smoothing 0.0 \

--aug\_hflip --aug\_crop --aug\_cutout --aug\_jitter \

--amp --seed 42

**Colab**

python -m vgg6\_cifar.scripts.train\_baseline \

--data\_dir /content/data \

--out\_dir /content/runs/baseline \

--epochs 60 --batch\_size 128 --lr 0.1 --optimizer sgd --momentum 0.9 \

--weight\_decay 5e-4 --label\_smoothing 0.0 \

--aug\_hflip --aug\_crop --aug\_cutout --aug\_jitter \

--amp --seed 42

**You’ll see printed:**

* Normalization: mean=(0.4914, 0.4822, 0.4465), std=(0.2470, 0.2435, 0.2616)
* Augs used: RandomCrop(32,4), RandomHorizontalFlip, ColorJitter(0.2/0.2/0.2/0.02), RandomErasingSquare(~2%)

**Q1(b) Train one strong configuration**

Already done by the command above (VGG6 + SGD+momentum + cosine LR warmup + AMP).  
Stores best checkpoint by **validation accuracy**.

**Q1(c) Report test top-1 & curves**

**Artifacts produced in --out\_dir:**

* metrics.csv, best.pt, final\_test\_metrics.json (contains test\_top1\_acc), README\_BASELINE.txt

**Generate curves:**

python -m vgg6\_cifar.scripts.plot\_curves \

--metrics\_csv ./runs/baseline/metrics.csv \

--out\_dir ./runs/baseline

# Colab: replace path with /content/runs/baseline

Result: loss\_curves.png, accuracy\_curves.png.

**Q2. Model Performance on Different Configurations (60 pts)**

We use the general runner train\_experiment.py. It supports:

* --activation {relu,sigmoid,tanh,silu,gelu}
* --optimizer {sgd,nesterov-sgd,adam,adamw,adagrad,rmsprop,nadam}
* --batch\_size, --epochs, --lr
* --wandb to log and later draw a W&B parallel-coords plot

**Q2(a) Vary the activation function (20 pts)**

Pick one optimizer/lr/bs, change only activation:

# ReLU

python -m vgg6\_cifar.scripts.train\_experiment \

--data\_dir ./data --out\_dir ./runs/act\_relu \

--activation relu --optimizer sgd --lr 0.1 --batch\_size 128 --epochs 40 \

--aug\_hflip --aug\_crop --aug\_cutout --aug\_jitter --amp --seed 42 --wandb

# SiLU

python -m vgg6\_cifar.scripts.train\_experiment \

--data\_dir ./data --out\_dir ./runs/act\_silu \

--activation silu --optimizer sgd --lr 0.1 --batch\_size 128 --epochs 40 \

--aug\_hflip --aug\_crop --aug\_cutout --aug\_jitter --amp --seed 42 --wandb

# GELU, Sigmoid, Tanh similarly:

# --activation gelu | sigmoid | tanh

**What to report:** compare best\_val\_acc and test\_top1\_acc from each run’s final\_test\_metrics.json, and discuss gradient flow & smoothness differences.

**Q2(b) Vary the optimizer (30 pts)**

Fix activation (e.g., ReLU or GELU), vary optimizers:

# SGD (no Nesterov)

python -m vgg6\_cifar.scripts.train\_experiment --data\_dir ./data --out\_dir ./runs/opt\_sgd \

--activation relu --optimizer sgd --lr 0.1 --batch\_size 128 --epochs 40 \

--aug\_hflip --aug\_crop --aug\_cutout --aug\_jitter --amp --seed 42 --wandb

# Nesterov-SGD

python -m vgg6\_cifar.scripts.train\_experiment --data\_dir ./data --out\_dir ./runs/opt\_nesterov \

--activation relu --optimizer nesterov-sgd --lr 0.1 --batch\_size 128 --epochs 40 \

--aug\_hflip --aug\_crop --aug\_cutout --aug\_jitter --amp --seed 42 --wandb

# Adam, AdamW, RMSprop, Nadam, Adagrad:

# --optimizer adam --lr 0.001 or 0.003

# --optimizer adamw --lr 0.001 (often strong)

# --optimizer rmsprop --lr 0.01

# --optimizer nadam --lr 0.001

# --optimizer adagrad --lr 0.05

**Tip:** adaptive optimizers usually need smaller --lr than SGD.

**Q2(c) Vary batch size, epochs, learning rate (10 pts)**

Run multiple combos; note effects on convergence speed & generalization.

# Smaller batch, more epochs, smaller lr

python -m vgg6\_cifar.scripts.train\_experiment \

--data\_dir ./data --out\_dir ./runs/hp\_bs64\_e80\_lr005 \

--activation relu --optimizer sgd --lr 0.05 --batch\_size 64 --epochs 80 \

--aug\_hflip --aug\_crop --aug\_cutout --aug\_jitter --amp --seed 42 --wandb

# Larger batch, fewer epochs, higher lr (watch for stability)

python -m vgg6\_cifar.scripts.train\_experiment \

--data\_dir ./data --out\_dir ./runs/hp\_bs256\_e40\_lr02 \

--activation relu --optimizer sgd --lr 0.2 --batch\_size 256 --epochs 40 \

--aug\_hflip --aug\_crop --aug\_cutout --aug\_jitter --amp --seed 42 --wandb

**Optional: run a quick grid sweep to build evidence (recommended)**

python -m vgg6\_cifar.scripts.sweep\_grid \

--data\_dir ./data --base\_out ./runs/sweeps \

--epochs 30 \

--batch\_sizes 64,128 \

--lrs 0.1,0.05,0.01 \

--optimizers sgd,nesterov-sgd,adam,adamw,rmsprop,nadam,adagrad \

--activations relu,silu,gelu,tanh,sigmoid \

--amp --wandb --seed 42

Outputs ./runs/sweeps/sweep\_summary.csv summarizing best\_val\_acc and test\_top1\_acc for each config.

**Q3. Plots (10 pts)**

**Q3(a) W&B parallel-coordinate plot**

1. Make sure your runs were executed with --wandb.
2. In your W&B project (vgg6-cifar10-assignment by default), open **Parallel Coordinates**.
3. Add axes: activation, optimizer, batch\_size, lr, best\_val\_acc.
4. Screenshot it for the PDF.

**Q3(b) Validation accuracy vs step (scatter)**

For any run with metrics.csv:

python -m vgg6\_cifar.scripts.plot\_scatter\_valacc\_vs\_step \

--metrics\_csv ./runs/exp\_act\_gelu/metrics.csv \

--out\_png ./runs/exp\_act\_gelu/scatter\_valacc\_vs\_step.png

**Q3(c) Training loss/acc & validation loss/acc**

python -m vgg6\_cifar.scripts.plot\_curves \

--metrics\_csv ./runs/exp\_act\_gelu/metrics.csv \

--out\_dir ./runs/exp\_act\_gelu

Results: loss\_curves.png, accuracy\_curves.png.

**Q4. Final Model Performance (10 pts)**

1. Look at your **W&B parallel-coords** plot; identify **exactly one** config with the best **validation accuracy**.
2. Re-run **exactly that config** to verify reproducibility:

python -m vgg6\_cifar.scripts.train\_experiment \

--data\_dir ./data --out\_dir ./runs/final\_best \

--activation <best\_act> \

--optimizer <best\_opt> \

--lr <best\_lr> \

--batch\_size <best\_bs> \

--epochs <best\_epochs> \

--aug\_hflip --aug\_crop --aug\_cutout --aug\_jitter --amp --wandb --seed 42

1. Include in the PDF:
   * The **exact command** above (your chosen flags).
   * The content of ./runs/final\_best/final\_test\_metrics.json (shows best\_val\_acc, test\_top1\_acc).
   * Curves/scatter as needed.

**Q5. Reproducibility & Repository (10 pts)**

**Q5(a) Clean, modular code (4)**

Already structured:

* models/ (VGG6 + activation registry)
* data/ (CIFAR-10, transforms)
* engine/ (trainer, optimizer factory)
* utils/ (logger, metrics)
* scripts/ (baseline, experiment, sweep, plotting)

**Q5(b) README + exact commands + environment + seed (4)**

* The repo’s README.md already contains:
  + Precise run commands for Q1–Q4
  + Dependencies in requirements.txt
  + --seed flag (default 42), CuDNN deterministic on

**Q5(c) Upload trained model (2)**

* Upload the **final best** checkpoint best.pt from ./runs/final\_best/ to your GitHub repo.
* Put the GitHub repo link **inside your PDF**.

**What files to copy into your PDF**

* **Q1**:
  + Transforms used (from log) + normalization values
  + final\_test\_metrics.json (test\_top1\_acc)
  + loss\_curves.png, accuracy\_curves.png
* **Q2**: brief table/summary of configs vs. best\_val\_acc & test\_top1\_acc; concise discussion for (a), (b), (c)
* **Q3**:
  + W&B parallel-coords screenshot
  + scatter\_valacc\_vs\_step.png
  + (optionally re-use curves from Q1/Q2 runs)
* **Q4**: the **single best** config command + final\_test\_metrics.json + note that it reproduces
* **Q5**: link to GitHub repo (with best.pt uploaded)

**Troubleshooting (fast)**

* **ModuleNotFoundError: vgg6\_cifar** → run from project root using python -m vgg6\_cifar.scripts.<script> ...
* **CUDA OOM** → use --batch\_size 64 (or 32), keep --amp, and reduce --num\_workers 2
* **Colab** → confirm GPU is on; !nvidia-smi; reinstall requirements.txt if needed

If you want, I’ll also generate a **ready-to-run Colab notebook** that executes: baseline → a few activation/optimizer runs → plots → prints best config and saves the final report images in one go.