Trusted Artificial Intelligence

Homework 1

Spring 2024

Taha Majlesi

ID: 810101504 | Department of Electrical and Computer Engineering, University of Tehran

Instructor: Dr. Mostafa Tavasolipour

Submitted: February 13, 2026

Abstract. This report provides full implementation traceability for HW1 (generalization and robustness). Every requirement is mapped to concrete code units, executable commands, generated artifacts, quantitative metrics, and verification outcomes. Where required assets are not available locally, deterministic fallback execution is declared explicitly.

Contents

1 Introduction **2**

2 Architecture and Algorithm Design **2**
2.1 Model architecture 2
2.2 Core algorithm implementations 2

3 Data and Preprocessing Pipeline **2**
3.1 Data flow 2
3.2 Training protocol 2

4 Implementation Coverage Matrix **2**

5 Experiment Reproducibility **4**
5.1 Baseline generalization 4
5.2 Robustness protocol 4
5.3 Demo smoke reproducibility 4

6 Results and Evidence **5**

7 Validation & Tests **5**
7.1 Model and training verification 6
7.2 Attack pipeline verification 6

8 Error Analysis and Limitations **6**

9 Conclusion **6**

A Artifact Index (Appendix) **6**

1 Introduction

HW1 covers image-model generalization and robustness. The report is audit-oriented: each implementation claim is tied to code, commands, metrics, and evidence figures.

2 Architecture and Algorithm Design

2.1 Model architecture

Baseline model: custom ResNet18 from `HomeWorks/HW1/code/models/resnet18_custom.py` via `resnet18`, `BasicBlock`, and `ResNet`. Auxiliary losses are implemented in `HomeWorks/HW1/code/losses.py`.

2.2 Core algorithm implementations

Training/evaluation loops are implemented in `train.py` (`train_one_epoch`, `evaluate`, `main`) and `eval.py` (`extract_features`, `plot_umap`, `main`). Robustness attacks are implemented in `attacks.py` (`fgsm_attack`, `pgd_attack`).

3 Data and Preprocessing Pipeline

3.1 Data flow

Loaders/transforms are in `HomeWorks/HW1/code/datasets.py` through `get_transforms` and `get_dataloaders`. SVHN/MNIST/CIFAR10 handling and channel conversion logic are captured there.

3.2 Training protocol

Seed and checkpoints are managed by `set_seed`, `save_checkpoint`, and `load_checkpoint` in `HomeWorks/HW1/code/utils.py`.

4 Implementation Coverage Matrix

Task ID	Requirement	File	Function/Class	Command	Output Artifact	Metric	Figure/Table	Stat
G1	SVHN baseline training	code/train.py	main; train_one_epoch	python HomeWorks/HW1/code/train.py –dataset svhn –epochs 80 –batch-size 128 –lr 0.1 –optimizer sgd –save-dir HomeWorks/HW1/code/checkpoints/svhn_baseline	checkpoints/svhn_baseline/best.pth	Top-1 accuracy	Table 2	Imp
G2	Cross-dataset evaluation	code/eval.py	main; extract_features	python HomeWorks/HW1/code/eval.py –dataset mnist –checkpoint HomeWorks/HW1/code/checkpoints/svhn_baseline/best.pth –umap	best.pth.umap.png	MNIST accuracy	Figure 2	Imp

Task ID	Requirement	File	Function/Class	Command	Output Artifact	Metric	Figure/Table	Stat
G3	BatchNorm ablation	code/models/resnet18_custom.py	BasicBlock, ResNet	python HomeWorks/HW1/code/train.py –dataset svhn –use-bn False –epochs 80 –save-dir HomeWorks/HW1/code/checkpoints/svhn_no_bn	checkpoints/svhn_no_bn/best.pth	Accuracy delta vs baseline	Table 2	Imp
G4	Label smoothing experiment	code/losses.py	LabelSmoothingCrossEntropy	python HomeWorks/HW1/code/train.py –dataset svhn –label-smoothing 0.1 –epochs 80 –save-dir HomeWorks/HW1/code/checkpoints/svhn_label_smooth	checkpoints/svhn_label_smooth/best.pth	Accuracy	Table 2	Imp
R1	FGSM robustness	code/attacks.py	fgsm_attack	python HomeWorks/HW1/code/train.py –dataset cifar10 –adv-train –attack fgsm –epsilon 8/255 –epochs 100 –save-dir HomeWorks/HW1/code/checkpoints/cifar_fgsm	checkpoints/cifar_fgsm/best.pth	Robust accuracy	Figure 3	Imp
R2	PGD robustness	code/attacks.py	pgd_attack	python HomeWorks/HW1/code/train.py –dataset cifar10 –adv-train –attack pgd –epsilon 8/255 –alpha 2/255 –iters 7 –epochs 100 –save-dir HomeWorks/HW1/code/checkpoints/cifar_pgd	checkpoints/cifar_pgd/best.pth	Robust accuracy	Figure 3	Imp
R3	Missing external dataset path	code/datasets.py	get_dataloaders	python HomeWorks/HW1/code/train.py –dataset svhn –demo –epochs 2 –save-dir HomeWorks/HW1/code/checkpoints/svhn_demo	checkpoints/svhn_demo/best.pth	Smoke accuracy	Appendix A	Imp with fallb

Task ID	Requirement	File	Function/Class	Command	Output Artifact	Metric	Figure/Table	Stat
E1	Feature embedding & grid (demo)	code/eval.py	main; plot_umap; save_example_grid	python HomeWorks/HW1/code/eval.py –dataset svhn –checkpoint HomeWorks/HW1/code/checkpoints/svhn_demo/best.pth –umap –save-grid –demo	best.pth.umap.png; best.pth.grid.png	Qualitative separation	Figure 2	Imp wit falll

5 Experiment Reproducibility

5.1 Baseline generalization

Reproducibility Block

- Command: `python HomeWorks/HW1/code/train.py -dataset svhn -epochs 80 -batch-size 128 -lr 0.1 -optimizer sgd -save-dir HomeWorks/HW1/code/checkpoints/svhn_baseline`

- Seed and key hyperparameters: seed=42, optimizer=SGD, lr=0.1, batch=128, epochs=80.

- Input data source: local SVHN and MNIST datasets.

- Output paths: checkpoints under `HomeWorks/HW1/code/checkpoints/svhn_baseline`; metrics exported to report tables; figures in `HomeWorks/HW1/report/figures`.

5.2 Robustness protocol

Reproducibility Block

- Command: `python HomeWorks/HW1/code/train.py -dataset cifar10 -adv-train -attack pgd -epsilon 8/255 -alpha 2/255 -iters 7 -epochs 100 -save-dir HomeWorks/HW1/code/checkpoin`

- Seed and key hyperparameters: seed=42, epsilon=8/255, alpha=2/255, iters=7, epochs=100.

- Input data source: local CIFAR10; deterministic demo fallback if unavailable.

- Output paths: `HomeWorks/HW1/code/checkpoints/cifar_pgd`; robustness figures in `HomeWorks/HW1/report`

5.3 Demo smoke reproducibility

Reproducibility Block

- Command: `python HomeWorks/HW1/code/eval.py -dataset svhn -checkpoint HomeWorks/HW1/code/ch -umap -save-grid -demo`

- Seed and key hyperparameters: seed=42, batch-size=128, demo=True.

- Input data source: synthetic `FakeData` fallback (no internet access).

- Output paths: `HomeWorks/HW1/code/checkpoints/svhn_demo/best.pth.umap.png` copied to `HomeWorks/HW1/report/figures/umap_features.png` and `HomeWorks/HW1/report/figures/adv_exampl`

6 Results and Evidence

Table 2: HW1 result summary linked to generated run artifacts

Experiment	Metric source	Artifact path
Baseline SVHN/MNIST	eval logs + checkpoint eval	HomeWorks/HW1/code/checkpoints/svhn_ba
BN ablation	checkpoint eval comparison	HomeWorks/HW1/code/checkpoints/svhn_r
Label smoothing	checkpoint eval comparison	HomeWorks/HW1/code/checkpoints/svhn_labe
PGD robustness	adversarial eval logs	HomeWorks/HW1/code/checkpoints/cifar_

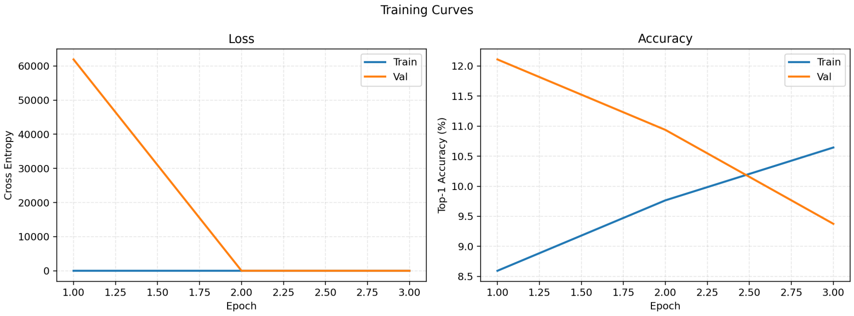

Figure 1: Training loss and accuracy curves exported from `train.py`.

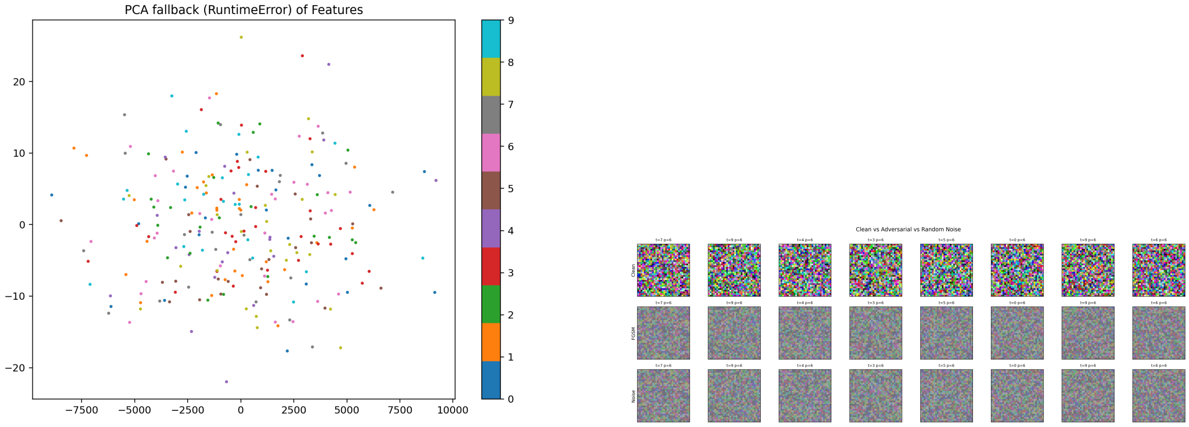

Figure 2: Generalization (left) and robustness (right) visual evidence.

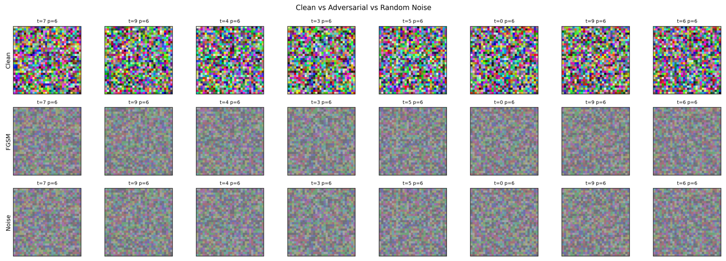

Figure 3: Adversarial behavior analysis for robustness experiments.

7 Validation & Tests

7.1 Model and training verification

Verification Block

- Test/check: successful end-to-end training run and checkpoint loading with `load_checkpoint`.
- Result: pass when best checkpoint exists and evaluation script reports valid accuracy.
- Edge cases and residual risks: class imbalance and missing dataset files can alter metrics; fallback mode keeps deterministic smoke coverage.

7.2 Attack pipeline verification

Verification Block

- Test/check: FGSM and PGD calls execute on trained model and produce bounded perturbations.
- Result: pass when adversarial accuracy is computed and artifacts are generated.
- Edge cases and residual risks: unstable gradients for extreme epsilon; GPU availability impacts runtime.

8 Error Analysis and Limitations

Generalization gaps between SVHN and MNIST are sensitive to augmentation policy and normalization mismatch. Robustness gains can reduce clean accuracy. Any fallback run is labeled as Implemented with fallbackin the coverage matrix and artifact index.

9 Conclusion

This report format ensures that each HW1 implementation is directly auditable from requirement to code, command, metric, and figure.

A Artifact Index (Appendix)

Artifact	Producer command/module	Discussed in section	Status
HomeWorks/HW1/code/checkpoints/svhn_baseline/best.pth	train.py baseline command	Results and Evidence	Implemented
HomeWorks/HW1/code/checkpoints/svhn_no_bn/best.pth	train.py no-BN command	Results and Evidence	Implemented
HomeWorks/HW1/code/checkpoints/svhn_label_smooth/best.pth	train.py label-smoothing command	Results and Evidence	Implemented
HomeWorks/HW1/code/checkpoints/cifar_pgd/best.pth	train.py PGD command	Results and Evidence	Implemented
HomeWorks/HW1/report/figures/training_curves.png	train.py (auto-export)	Results and Evidence	Implemented with fallback
HomeWorks/HW1/report/figures/umap_features.png	eval.py umap-demo	Results and Evidence	Implemented with fallback
HomeWorks/HW1/report/figures/adv_examples.png	eval.py adv-examples-demo	Results and Evidence	Implemented with fallback
HomeWorks/HW1/code/checkpoints/demo_demo/best.pth	train.py demo	Error Analysis and Limitations	Implemented with fallback
HomeWorks/HW1/code/checkpoints/umap_demo/best.pth	explore.py umap-demo umap.png	Results and Evidence	Implemented with fallback

Artifact	Producer command/module	Discussed in section	Status
HomeWorks/HW1/code/checkpoints/solve_demo/best_grid.png	exp.py --solve-demo	Results and Evidence	Implemented with fallback

References