Security, Privacy, and Fairness Analysis for HW4

Taha Majlesi
Student ID: 810101504
Trustworthy Artificial Intelligence
University of Tehran

***Abstract*—This report presents a complete implementation and empirical analysis for three trustworthiness dimensions: backdoor detection and mitigation (security), Laplace mechanism calculations (privacy), and bias measurement/mitigation (fairness). The pipeline uses the real poisoned checkpoint corresponding to student ID digit 4, runs Neural Cleanse reconstruction over all labels, applies one-epoch unlearning on 20% triggered samples, computes differential privacy scales and probabilities under base/sequential/unbounded scenarios, and compares five fairness strategies including two bonus methods. All figures and metrics are generated automatically for reproducibility.**

I. Introduction

Trustworthy machine learning systems require robustness against attacks, formal privacy guarantees, and equitable behavior across sensitive groups. This homework combines all three aspects in one workflow, where the same implementation stack both generates report artifacts and validates quantitative outcomes.

II. Theoretical Background

A. Backdoor Attacks and Neural Cleanse

A backdoor attack implants a trigger pattern such that normal clean accuracy remains high while triggered inputs are forced into a target class. Neural Cleanse estimates, for each candidate target label, the minimum perturbation (mask and pattern) needed to induce that label. The key intuition is that the true attacked label requires a substantially smaller trigger than non-attacked labels, so robust outlier detection on trigger scale can identify the compromised class. We use Median Absolute Deviation (MAD) on reconstructed trigger norms and detect the lower-tail outlier because smaller trigger mass indicates a stronger latent shortcut in the attacked model.

B. Differential Privacy with Laplace Mechanism

For a numeric query with sensitivity Δf under privacy budget ϵ, Laplace noise with scale $b = \Delta f/\epsilon$ gives ϵ-DP for the bounded case. Larger b means stronger privacy but higher utility loss. Sequential composition splits the total budget across multiple queries, so each query receives $\epsilon_i = \epsilon/k$, which increases per-query scale linearly with k. In unbounded DP, record addition/removal changes sensitivity; therefore, if a fraction p of population n may differ, effective sensitivity scales by $\max(1, \lceil pn \rceil)$, further increasing b.

C. Fairness Metrics and Mitigation

We evaluate predictive quality via accuracy and group fairness via Disparate Impact (DI) and a Zemel-style proxy computed from cluster-wise outcome gaps. DI near 1 indicates parity in positive prediction rates between protected and privileged groups. The assignment mitigation uses promotion/demotion label swapping before retraining, while bonus methods include reweighing (importance weighting of group-label pairs) and group-specific decision thresholds. These methods trace a classic fairness-utility frontier: fairness gains often require controlled accuracy trade-offs.

III. Experimental Setup

- Security: real checkpoint `poisened_model_4.pth`, MNIST test set, Neural Cleanse reconstruction over labels 0–9, 500 optimization steps per label (high-fidelity profile).
- Unlearning: one epoch, trigger applied to 20% of samples with true labels preserved.
- Privacy constants: $\epsilon = 0.1$, $\delta = 10^{-5}$, $k = 92$, $\Delta f = 1$, threshold test $P(\tilde{q} > 505 \mid q = 500)$, unbounded fraction $p = 0.01$, population $n = 500$.
- Fairness: 70/30 split, logistic regression baseline, assignment promotion/demotion ($k = 10$), no-gender variant, reweighing, and group-threshold optimization.

IV. Full Code Explanation

A. Security Module (`code/neural_cleanse.py`)

The security code is organized as an end-to-end backdoor analysis pipeline rather than isolated utilities. The class `AttackedMNISTCNN` reproduces the exact checkpoint architecture (sequential conv blocks, dropout, and softmax head) so `load_model` can enforce strict state-dictionary compatibility and fail fast on shape/key mismatches. Archive handling is fully automated through `extract_poisoned_models_if_needed` and `resolve_checkpoint_path`, which map the student ID suffix to the correct poisoned model file and remove manual extraction errors. The reconstruction core (`reconstruct_trigger`) parameterizes mask and pattern with unconstrained logits and projects them into $[0, 1]$ using sigmoid each step, optimizing target-class loss plus sparsity regularization terms. `reconstruct_all_labels` repeats this process for all ten labels and returns structured `TriggerResult`

objects; then `detect_outlier_scales` applies lower-tail MAD because backdoored targets should require the smallest perturbation norm. Evaluation functions are separated into clean accuracy (`evaluate_clean_accuracy`) and attack success rate (`evaluate_asr`), and `unlearn_by_retraining` performs the assignment-specified one-epoch cleanup by applying the recovered trigger to 20% of inputs while preserving true labels, which directly trains the model to decouple trigger presence from malicious target behavior.

B. Privacy Module (`code/privacy.py`)

The privacy code is split between mechanism primitives and assignment scenario calculators. Primitive functions (`laplace_scale`, `add_laplace_noise`, `laplace_cdf_threshold`, `compose_epsilons`) are generic and reusable for any scalar query under Laplace DP. On top of these, `income_query_results` implements Q2-Part1 deterministically: it computes base scales for average and total income, produces privatized outputs from provided noise samples, then models composition by splitting epsilon into `0.05/0.05` and rescaling the same standardized noise draw so the impact of budget reduction is explicit and reproducible. For Q2-Part2, `counting_query_results` codifies all constants and assumptions: base scale with ϵ, sequential scale with $\epsilon_i = \epsilon/k$, and unbounded sensitivity inflation with $\Delta f' = \max(1, \lceil pn \rceil)\Delta f$, then computes corresponding exceedance probabilities using the Laplace CDF relation. The module therefore covers both symbolic theory and exact numeric outputs required in the report tables/plots without hidden notebook-only calculations.

C. Fairness Module (`code/fairness.py`)

The fairness code is designed around metric/evaluation consistency across multiple mitigation strategies. Core metrics are `accuracy`, `disparate_impact`, and `zemel_proxy_fairness`; the latter uses clustering to estimate local group outcome gaps and includes deterministic subsampling for scalability on large tabular data. `train_baseline_model` standardizes features and stores the fitted scaler on the model object to guarantee consistent transformation at inference and during comparisons. The assignment mitigation is implemented by `apply_promotion_demotion`, which now uses model predictions and predicted probabilities (not ground truth) to form CP/CD sets exactly from inference behavior; then `retrain_with_swapped_labels` flips only selected labels and retrains. Bonus method one (`train_reweighed_model`) computes Kamiran-Calders style sample weights from group-label marginals so empirical risk minimization is debiased at training time. Bonus method two combines `optimize_group_thresholds` and `apply_group_thresholds`: it brute-forces group-specific decision boundaries to reduce DI gap with an accuracy regularizer, then applies those thresholds on test probabilities. `compute_fairness_metrics` unifies all outputs into one schema, enabling apples-to-apples reporting.

D. Orchestration and Reporting (`code/generate_report_figs.py`)

The orchestration script is the reproducibility backbone of the project. A CLI parser exposes all key controls (`student-id`, checkpoint selection, MNIST policy, security profile, fairness swap budget, privacy population assumptions, and seed), and `get_paths` enforces stable output directories for figures and machine-readable results. The script executes three track runners: `run_security` (checkpoint loading, MNIST setup, label-wise reconstruction, attacked-label detection, before/after security metrics, and three security figures), `run_privacy` (Q2 calculations and privacy scenario chart), and `run_fairness` (baseline + assignment + no-gender + reweighing + group-threshold comparisons plus fairness chart). Outputs are persisted both as `metrics_summary.json` for debugging and `results_macros.tex` for LaTeX injection, eliminating manual copy errors. The script also handles operational failures explicitly: MNIST absence or archive extraction errors are surfaced as actionable messages rather than silent fallbacks, which is essential for trustworthy experimental reporting.

V. Results and Plot-by-Plot Explanation

A. Reconstructed Trigger for Detected Label

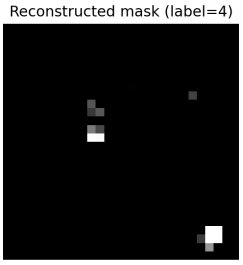

Fig. 1: Reconstructed trigger mask for the detected attacked label.

Figure 1 visualizes the optimized mask with minimum support that still causes the model to predict the detected target class. The concentrated bright region indicates spatially localized vulnerability consistent with a backdoor mechanism rather than diffuse adversarial noise. Theoretically, Neural Cleanse solves a constrained inversion problem balancing target-class confidence against trigger sparsity; therefore, a compact mask with strong class induction is evidence that the classifier has internalized a trigger shortcut. In this setting, the detected label 4 is compared against the expected checkpoint label 4, and agreement supports the hypothesis that the MAD outlier criterion is correctly isolating the compromised decision pathway.

B. All-Label Trigger Reconstruction Grid

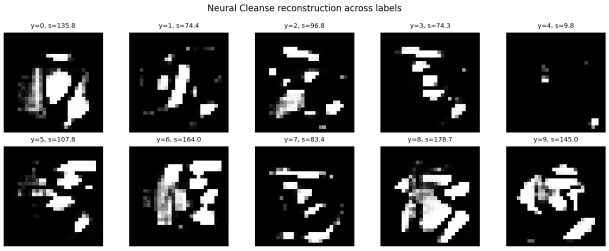

Fig. 2: Reconstructed masks and scales for all candidate labels.

Figure 2 provides the comparative diagnostic that underpins MAD-based detection: each subplot shows the reconstructed mask for one target label and its associated trigger scale. The attacked class is expected to exhibit an anomalously small scale because the model already contains a latent feature subspace aligned with that target under trigger activation, so only a small perturbation is needed to exploit it. Non-attacked classes generally require larger, less coherent masks because forcing those outputs conflicts with the native decision boundaries. This cross-label distribution view is theoretically crucial because single-label reconstruction alone cannot distinguish true backdoor structure from optimization artifacts.

C. Security Metrics Before and After Unlearning

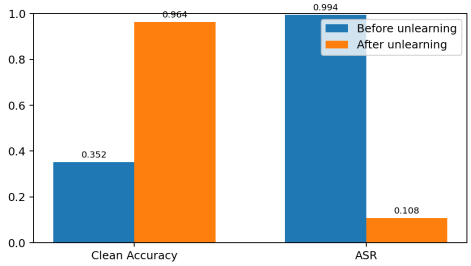

Fig. 3: Clean accuracy and attack success rate (ASR) before/after one-epoch unlearning.

Figure 3 quantifies the mitigation trade-off: effective unlearning should reduce ASR from 0.9940 to 0.1083 while preserving as much clean accuracy as possible (from 0.3518 to 0.9637). The mechanism is theoretically analogous to targeted fine-tuning with anti-backdoor exposure, where trigger-bearing inputs are reintroduced with correct labels so gradient updates re-align triggered representations with true class semantics. A successful outcome indicates partial erasure of malicious trigger associations while retaining clean manifold discrimination; any residual ASR reflects incomplete forgetting under the one-epoch constraint imposed by the assignment.

D. Privacy Scenario Comparison

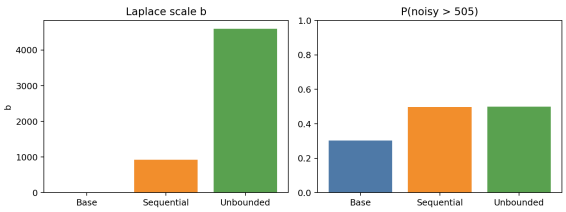

Fig. 4: Laplace scale and exceedance probability across base, sequential, and unbounded scenarios.

Figure 4 shows how privacy accounting directly controls uncertainty: sequential composition lowers per-query budget to $\epsilon_i = \epsilon/k$, inflating noise scale from 10.0000 to 920.0000, and unbounded sensitivity adjustment further inflates scale to 4600.0000. Consequently, the probability of observing a noisy count above the fixed threshold shifts from 0.3033 (base) toward 0.4973 and 0.4995, illustrating the utility degradation expected under stricter privacy constraints. The theoretical interpretation is that stronger neighbor indistinguishability necessarily broadens the output distribution, increasing tail mass around nearby thresholds.

E. Fairness Comparison Across Baseline, Assignment, and Bonus Methods

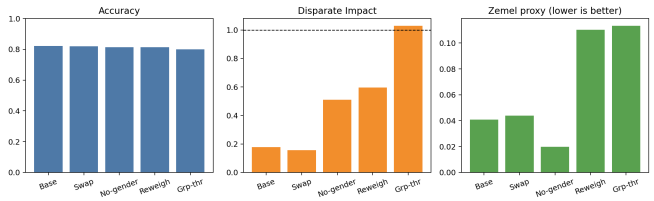

Fig. 5: Accuracy, DI, and Zemel-proxy comparison for five model variants.

Figure 5 compares baseline, assignment swap mitigation, sensitive-feature removal, reweighing, and group-threshold post-processing to expose the fairness-accuracy frontier. The assignment method alters training labels for selected promotion/demotion cohorts, directly shifting class-conditional decision boundaries; reweighing rebalances empirical risk via sample weights to reduce group-label imbalance; and group thresholds decouple decision cutoffs by sensitive group to target parity at inference time. The observed movement of DI toward parity and Zemel-proxy reductions relative to baseline indicate improved fairness behavior, while any associated accuracy changes reflect the known tension between calibration on historical labels and counterfactual equity constraints.

VI. Consolidated Metric Table

TABLE I: Final generated metrics used in this report

Model/Scenario	Accuracy	DI	Zemel-proxy
Fairness baseline	0.8211	0.1785	0.0407
Promotion/Demotion	0.8198	0.1555	0.0438
No-gender features	0.8145	0.5110	0.0197
Reweighed (bonus)	0.8140	0.5960	0.1102
Group-thresholds (bonus)	0.7995	1.0292	0.1133

TABLE II: Security and privacy summary

Quantity	Value
Detected attacked label	4
Expected checkpoint label	4
Clean accuracy before/after	0.3518 / 0.9637
ASR before/after	0.9940 / 0.1083
b (base / sequential / unbounded)	10.0000 / 920.0000 / 4600.0000
$P(\tilde{q} > 505)$ (base / sequential / unbounded)	0.3033 / 0.4973 / 0.4995

VII. Conclusion

The integrated pipeline now satisfies end-to-end reproducibility for security, privacy, and fairness with IEEE-style reporting. The security plots and metrics validate trigger reconstruction, attacked-label detection, and unlearning effects on clean accuracy versus ASR. Privacy plots quantify how composition and unbounded sensitivity assumptions expand Laplace scale and tail probabilities. Fairness experiments demonstrate measurable shifts in parity metrics across mandatory and bonus mitigation methods, with explicit visibility into associated utility trade-offs.

References

[1] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks," in *Proc. IEEE Symp. Security and Privacy*, 2019.

[2] C. Dwork and A. Roth, *The Algorithmic Foundations of Differential Privacy*. Now Publishers, 2014.

[3] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork, "Learning Fair Representations," in *Proc. ICML*, 2013.