Homomorphic Encryption for Neural Network Deployment: An Empirical Security Evaluation

Michael Xiang, Steve Dalla, Gary Wu

May 7, 2025

Abstract

Deploying machine learning models on untrusted cloud infrastructure raises concerns about model confidentiality and sensitive input privacy. Homomorphic encryption (HE) allows neural network inference on encrypted data, making it theoretically impossible for attackers to learn either the model un-encrypted weights or client inputs without access to the decryption key. In this paper, we implement an HE-based neural network using a polynomial activation approximation and compare its security posture against a standard unencrypted network. We focus on membership inference, model extraction, and a new gradient-based input recovery attack, testing whether access to encrypted weights confers any advantage to an attacker. Our code is available on GitHub[1].

1 Introduction and Motivation

Machine learning systems no longer exist merely as academic curiosities; they now grade exams, triage ICU beds, approve mortgages, and shape national security decisions. In each of these arenas, the underlying data is incredibly valuable. For example, a mishandled biopsy sequence can reveal a patient's cancer genotype, while an exposed credit-scoring model can enable large-scale loan fraud.

Cloud infrastructure magnifies the stakes. Outsourcing computation to hyperscale providers slashes cost and latency, but it also forces model owners to trust opaque platforms not to snapshot RAM, scrape disk, or inject debugging hooks. Homomorphic encryption (HE), however, offers a cryptographic solution that performs computations over encrypted values without ever needing to decrypt them in the cloud. For neural networks, HE enables forward inference to remain encrypted, potentially preventing both input inference attacks and model theft. However, an HE deployment is only as secure as the weakest emergent link in the system, so crucial questions therefore arise:

- Do classical attacks like membership inference remain effective under HE deployments?
- Does seeing the encrypted weights (rather than only black-box queries) actually enable new attacks?
- Can gradient-based input reconstruction still be performed when only the encrypted model is available?

Our work addresses these by implementing a demonstration in TenSEAL[2], comparing an encrypted polynomial MLP with a standard unencrypted ReLU-based MLP on MNIST. We stage a battery of

[1]`https://github.com/mxiang04/HomomorphicEncryptionSecuritySurvey`
[2]`https://github.com/OpenMined/TenSEAL`

attacks, such as membership inference, surrogate extraction, gradient inversion, malicious queries, fault flips, and two cryptographic indistinguishability games, that span both confidentiality and integrity concerns.

By contrasting the Baseline and Poly-HE pipelines under identical threat models, we obtain a comprehensive empirical map of where homomorphic encryption protects and where it falls off.

2 Related Works & Differentiation

We build off the following related works.

Homomorphic Encryption. Gentry's seminal work [1] first introduced fully homomorphic encryption schemes, which allow arbitrary computations on ciphertexts. Subsequent implementations like SEAL, PALISADE, and TenSEAL have made HE practical for simple machine learning tasks.

Membership Inference. Shokri et al. [2] introduced membership inference attacks that exploit confidence scores of models to determine if a sample was in the training set. Even encrypted models could be vulnerable if outputs are revealed in plaintext.

Model Extraction. Tramer et al. [3] showed how repeated queries to a black-box ML model can recover approximate model parameters. Whether homomorphic deployments offer additional resistance is still an open question.

Gradient Attacks. Fredrikson et al. [4] demonstrated that with white-box access to model gradients, attackers can invert or reconstruct inputs. We adapt that concept to see how feasible gradient-based attacks are when parameters are encrypted.

Public-Key Encryption Indistinguishable Under Plaintext-Checkable Attacks. Abdalla et al. [5] proposed IND-CPA, a more reasonable security condition from IND-CCA which we will be consistent with in our experiments.

Privacy-preserving inference resistant to model extraction attacks. Byun et al. [6] showed ways to integrate additive Homomorphic Encryption to neural networks but is focused on multiparty computation instead of computation run on one wholly connected system.

Privacy-preserving machine learning: Threats and solutions. Al-Rubaie et al. [7] was a seminal landscape paper we're building off of by exploring homomorphic encryption as an additional way to preserve privacy.

Evasion attacks against machine learning at test time. Biggio, Battista et al. [8] was a seminal work in the ML attacks space. We are building off their ideas for attacks into the specific application of homomorphic encryption.

SecureCloud: Secure big data processing in untrusted clouds. Kelbert et al. [9] was a seminal project whose goal was to enable new big data applications that use sensitive data in the cloud without compromising data security and privacy. We build our project off of some of the data privacy concerns assumed in the paper.

Nonmalleable Cryptography. Dolev et al. [10] was a seminal work that preceded research on indistinguishability games, as the property of preventing attackers from modifying ciphertexts to create relationships between plain-texts, is essentially a form of indistinguishability under a specific attack model.

Leaking secrets in homomorphic encryption with side-channel attacks. Aydin et al. [11] demonstrated practical side-channel attacks against the Microsoft SEAL HE library, leaking secret keys during key generation. Our work explicitly assumes the HE implementation is secure against side-channels, highlighting a practical vulnerability in real-world deployments.

On the Security of Homomorphic Encryption on Approximate Numbers. Li et al. [12] argues that the traditional formulation of IND-CPA does not fare well against passive adversaries when applied to approximate encryption schemes, and provides a stronger security condition.

Attacks Against the INDCPA-D Security of Exact FHE Schemes. Cheon et al. [13] strengthens IND-CPA security based on Li's paper [12] by giving the attacker access to a decryption oracle for ciphertexts for which it should know the underlying plaintexts.

CryptoNets: Applying Neural Networks to Encrypted Data. Gilad-Bachrach et al. [14] introduced one of the earliest frameworks for performing inference on encrypted data using neural networks. They demonstrated the feasibility of using square activation functions in place of ReLU to maintain compatibility with homomorphic encryption schemes. CryptoNets achieved high accuracy on MNIST while showing how batching and polynomial functions could reduce inference latency. We build on this idea but apply more recent encryption schemes (CKKS) and evaluate robustness against membership inference and model extraction attacks.

GAZELLE: A Low Latency Framework for Secure Neural Network Inference. Juvekar et al. [15] proposed GAZELLE, a hybrid secure inference system that combines HE and garbled circuits to speed up neural network inference. HE is used for linear layers and GC for nonlinear ones. Our project differs by focusing solely on a pure HE setting without hybrid cryptographic components, allowing us to isolate and analyze the security properties of CKKS encryption under attack scenarios.

SoK: Fully Homomorphic Encryption Compilers. Li et al. [16] provide a systematization of knowledge (SoK) on FHE compilers, outlining various abstraction levels, optimization techniques, and security considerations. We draw on this work to better understand the compiler-level translation of high-level neural network operations to ciphertext-space operations, especially with regard to how different compiler choices may affect runtime leakage or security guarantees.

Privacy-Preserving in Medical Image Analysis: A Review of Methods and Applications. Kaissis et al. [17] conducted a comprehensive survey on privacy-preserving machine learning in medical imaging, analyzing methods such as differential privacy, secure multi-party computation, and homomorphic encryption. We use this as a reference point to show how our work applies techniques from a general privacy landscape to a focused inference task under homomorphic encryption.

Blockchain and Homomorphic Encryption Based Privacy-Preserving Model Aggregation for Medical Images. Sun et al. [18] explored the use of blockchain and HE to secure federated learning model aggregation in the medical domain. Although our system does not involve distributed training, this work reinforces the importance of protecting model parameters in outsourced inference, and provides a point of contrast between secure training and secure inference settings.

MedBlindTuner: Towards Privacy-preserving Fine-tuning on Biomedical Images with Transformers and Fully Homomorphic Encryption. Liu et al. [19] proposed a system for securely fine-tuning transformer-based models using FHE, especially for sensitive biomedical images. We focus on inference rather than fine-tuning, but this work informs our understanding of how FHE can be extended to deep learning architectures with large parameter counts, particularly under CKKS.

Homomorphic Encryption for Arithmetic of Approximate Numbers. Cheon et al. [20] introduced the CKKS scheme, which enables efficient encrypted computation over approximate numbers. CKKS is central to our project: we use it as the underlying encryption scheme in TenSEAL to perform neural network inference on real-valued data.

TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption. Benaissa et al. [21] described the TenSEAL library, a high-level interface for encrypted tensor computation built on SEAL and CKKS. We rely on TenSEAL for both encryption and evaluation of our homomorphic MLP.

Efficient Homomorphic Conversion Between (Ring) LWE Ciphertexts. Brakerski et al. [22] proposed a method for converting ciphertexts between HE schemes (e.g., from BFV to GSW or TFHE). While our project does not use ciphertext conversion, this work supports the broader claim that modular, efficient HE systems are becoming more practical and composable, and could inform future extensions to hybrid protocols.

Is Homomorphic Encryption-Based Deep Learning Secure Enough? Shin et al. [23] critically assessed the security of HE-based inference systems, revealing that even encrypted inference outputs can leak training membership and allow input reconstruction. Their empirical attack strategies inform our own threat model and evaluation pipeline, particularly for membership inference and gradient-based input inversion on our encrypted MLP.

TFHE: Fast Fully Homomorphic Encryption Over the Torus. Chillotti et al. [24] introduced TFHE, a high-speed fully homomorphic encryption (FHE) scheme optimized for binary and logic operations. It contrasts with CKKS, which is tailored for approximate arithmetic over real numbers. Our project uses CKKS for real-valued ML inference; referencing TFHE highlights an alternative HE approach that may be better suited for future extensions involving logic-based or bit-level operations.

SecureML: A System for Scalable Privacy-Preserving Machine Learning. Mohassel and Zhang [25] proposed SecureML, a two-party computation framework for privacy-preserving training of linear regression, logistic regression, and neural networks using secret sharing and garbled circuits. While our project focuses on inference rather than training, SecureML's efficient protocols for secure model training provide valuable insights into the design of privacy-preserving machine learning systems.

Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications. Riazi et al. [26] introduced Chameleon, a hybrid secure computation framework that combines additive secret sharing and garbled circuits to enable efficient privacy-preserving machine learning. Although our work emphasizes homomorphic encryption-based inference, Chameleon's approach to secure computation offers alternative strategies for balancing performance and security in privacy-preserving ML applications.

3 Threat Model & Scenario

3.1 System Architecture

Our system is modeled around a standard secure inference paradigm between two entities:

- **Trusted Client:** Holds the neural network, input data, and secret keys.
- **Untrusted Cloud Server:** Executes encrypted inference without decryption capabilities.

We adopt a threat model consistent with semantic security under the IND-CPA paradigm. We explicitly assume:

- The cryptographic scheme is semantically secure and correctly implemented.
- The cryptographic library's runtime operates faithfully and does not leak side-channel or timing data.
- The adversary does not have access to the decryption key or the client's local environment.
- Communication channels between client and server are authenticated and encrypted.

In this setting, the client's neural network is encrypted weight-by-weight before upload. Likewise, sensitive inputs are encrypted on-device before transmission. Inference is performed on the server, and the resulting ciphertexts are returned to the client for decryption. The server, at no point, processes plaintext model weights or inputs.

3.2 Models

To evaluate the security of inference under homomorphic encryption, we intend to implement two models:

Baseline MLP: A two-layer ReLU-activated MLP that is used as the standard against which security and performance tradeoffs are benchmarked.

Poly-HE MLP: An identically structured MLP where all activations are replaced with quadratic approximations, and weights are processed via our encryption. This model is used to evaluate security under ciphertext-only inference.

3.3 Threat Model and Attacker Capabilities

We formalize the attacker's role as an adaptive adversary in a cryptographic-style setting. The adversary interacts with the deployed model under two levels of access:

- **Encrypted Access (White-box, No Decryption):** The adversary has full access to the encrypted weights and the model architecture. This simulates compromise of the untrusted cloud server. The adversary may perform arbitrary computations or ciphertext manipulations but cannot decrypt values.
- **Output Access (Black-box or Partially Leaky):** The adversary observes output from model inference. We analyze two variants:
  1. **Encrypted-Only Output:** The attacker sees only ciphertext results and must reason about input behavior without any decryption access.
  2. **Plaintext Output Leakage:** The attacker can observe unencrypted prediction probabilities or logits (e.g., via inference APIs). This models real-world scenarios where systems return interpretable predictions to users.

3.4 Security Assumptions and Formal Goals

Our evaluation builds on the following formal assumptions:

- **Indistinguishability under Encryption (IND-CPA):** For the ciphertext-only output scenario, we expect the adversary to be no better than random guessing at distinguishing between encrypted outputs of different inputs.
- **No Meaningful Gradient Leakage:** Without access to gradients or decrypted logits, input inversion should be infeasible under ciphertext-only access.
- **Robustness to Bit-Flips:** Homomorphic ciphertexts are assumed to be non-malleable in practice. Any corruption (e.g., by fault injection) should result in failure or garbage output without partial leakage.

3.5 Metrics for Evaluation and Success

For each attack type that we will introduce in the next section, we define success metrics that align with the formal threat model:

- **Membership Inference:** Accuracy, precision, recall, and AUC of a threshold-based attack that distinguishes training vs. non-training inputs using model confidence scores. AUC near 0.5 implies no meaningful leakage.
- **Model Extraction:** Accuracy of a surrogate model trained via black-box querying (or decrypted homomorphic outputs) to match the original model's predictions. Success implies functional replication.
- **Input Inversion:** Mean squared error (MSE) between reconstructed and true input. We compare gradient descent (white-box) and SPSA (HE black-box) approaches. Low fitness in HE implies successful confidentiality.

- **NN-Indistinguishability Game:** Attacker's ability to distinguish encrypted model outputs from two different chosen inputs. Accuracy $\approx 50\%$ is the desired outcome.
- **Real vs. Ideal Game:** Attacker must distinguish real model outputs from randomly encrypted vectors. Again, accuracy $\approx 50\%$ denotes indistinguishability.
- **Malicious Query and Fault Injection:** Number of successful tampered inputs that produce stable or revealing ciphertext outputs. Zero successes over many trials reflects robustness.

3.6 Scope of Control and Limitations

We emphasize that our study controls the following:

- Complete access to encryption/decryption keys for honest client simulations.
- Full transparency and reproducibility of all adversarial attacks under defined capabilities.
- Consistent model architecture and training data across the baseline and HE settings.

However, we do not model:

- Side-channel attacks (e.g., timing, power analysis) during encryption or inference.
- Malicious key generation or ciphertext reuse vulnerabilities in HE libraries.
- Adaptive learning across long sessions with cumulative attacker observations.

Our experiments are thus best interpreted as a first-order empirical validation of HE security under practical yet thoughtful attack models.

4 Attacks

Given our defined threat model, the attacker's primary objectives include:

- **Input Reconstruction**: Attempting to recover sensitive input data from the encrypted inference process.
- **Model Extraction**: Attempting to reconstruct the encrypted model's parameters or train a surrogate model that replicates its functionality.

To systematically evaluate these threats, we implement and analyze the following specific attacks:

4.1 Membership Inference (MI):

We employ threshold-based methods that exploit the confidence scores of outputs to discern if a given sample belonged to the model's training dataset.

4.2 Model Extraction (ME):

We use query-response pairs from the model to train a surrogate model mimicking the original model's behavior.

4.3 Gradient-Based Input Inversion Attack:

For plaintext models (white-box), we use gradient optimization techniques for direct input reconstruction. In the HE scenario, given that gradients cannot be computed directly, we explore finite-difference approximations assuming partial plaintext access. Specifically, we will use the Simultaneous Perturbation Stochastic Approximation (SPSA) technique. Given an input vector $x \in \mathbb{R}^d$ and a target class y, we generate a random perturbation vector $r \in \{-1, 1\}^d$ and estimate the directional derivative of the model's output logit for class y via:

$$\hat{g}_y = \frac{f_y(x + \delta r) - f_y(x - \delta r)}{2\delta} \cdot r$$

where δ is a small scalar and $f_y(x)$ denotes the model's logit output for class y given input x. This estimate $\hat{g}_y$ acts as a noisy approximation of the true gradient, so that an attacker can perform iterative updates of x via

$$x_{t+1} = x_t + \eta \cdot \hat{g}_y$$

with learning rate η.

4.4 Malicious Query Attack

We simulate tampering by flipping random bytes in serialized ciphertexts and observe whether our models reliably reject or neutralize malformed queries without leaking any exploitable information.

4.5 Fault Injection Attack

We emulate memory-level bit flips in encrypted inputs and show that the resulting outputs are either invalid or completely incoherent, demonstrating the robustness of our cryptographic scheme against fault-based perturbations.

4.6 NN-Indistinguishability Security Game & Real vs. Ideal World Game:

Inspired by IND-CPA encryption security, these formal experiments involve an adversary attempting to distinguish between encrypted outputs of two distinct inputs. In this scenario, the attacker selects two plaintext inputs, x_0 and x_1, one of which is randomly encrypted and processed by the model. The adversary observes only the encrypted output ciphertext and must guess which input produced it. Our model's security holds if the adversary can do no better than random guessing (50% accuracy).

The real vs. ideal world game is slightly different. Here, we assume that we give an adversary an input and an output. This output is either from the real world (i.e. an encrypted output of a real input from the HE model) or from the ideal world (a random "encrypted" output that is independent from the real input). The attacker must guess whether the output comes from the real and ideal world. If it does no better than random guessing, our argument is that homomorphically encrypting ensures that the model is secure under our threat assumptions since there is a negligible advantage to being given access to the encrypted model as an attacker can do no better than they would in an ideal world that is totally secure.

5 Experimental Setup

Our implementation targets a practical evaluation of adversarial threats against a neural network protected via homomorphic encryption (HE). We use PyTorch to train and export a two-layer MLP on MNIST, and TenSEAL (CKKS scheme) to conduct all homomorphic operations. Below we dissect each implemented attack and the logic driving the indistinguishability games.

5.1 Baseline vs. HE Model

We first train a two-layer MLP (784-128-10) using ReLU activations on MNIST. To enable encrypted inference, we approximate the ReLU activation with a polynomial $f(x) = ax^2 + bx + c$, enabling compatibility with CKKS's real-number homomorphic arithmetic. The trained model's weights and biases are extracted, and inference is re-implemented manually via plaintext dot-products and polynomial activation to replicate forward passes under encryption.

5.2 Encrypted Inference

For each input $x \in \mathbb{R}^{784}$, we construct a CKKS-encrypted vector using TenSEAL's 'ckks_vector'. The inference process consists of two layers:

- **Layer 1:** For each hidden neuron j, compute $h_j = \text{poly}(\langle x, w_j \rangle + b_j)$ in plaintext using decrypted vectors. This approximates the non-linear activation securely.
- **Layer 2:** Multiply the hidden activations with output layer weights and add biases to obtain encrypted logits, returned as ciphertexts to the client.

5.3 Membership Inference Attack

We simulate a confidence-based membership inference attack using both the plaintext model and the encrypted model with decrypted outputs. The attacker:

1. Samples $N = 200$ member and non-member datapoints.
2. Queries each model and records the maximum softmax probability.
3. Uses a threshold (0.5) on confidence to classify whether the input was in the training set.

For the HE model, inference is performed using encrypted inputs and decrypted logits. This mirrors the real-world case where encrypted models may return plaintext predictions to the user.

5.4 Model Extraction Attack

We emulate the surrogate model extraction strategy:

- Query the model on 1,000 inputs.
- Collect $(x, \arg\max M(x))$ pairs.
- Train a shadow MLP using the same architecture as the original.

The extraction is performed under two settings: full access to the unencrypted model (baseline) and encrypted model outputs decrypted post-inference. Evaluation involves measuring agreement between original and surrogate models on 500 test samples.

5.5 Gradient-Based Input Inversion

We implement two distinct input recovery strategies:

- **Plaintext (White-box) Attack:** Gradient descent is used with backpropagation to optimize a randomly initialized input vector x to maximize output probability for a target class.
- **HE Scenario (Black-box Approximation):** We simulate an attacker using SPSA (Simultaneous Perturbation Stochastic Approximation). The attacker:

  1. Perturbs input x in positive and negative directions.
  2. Computes two encrypted model outputs.
  3. Approximates the gradient using finite differences.
  4. Updates x accordingly.

  Evaluation involves measuring if the attacker successfully decrypts the outputs in either strategy.

5.6 Malicious Query Attack

To probe TenSEAL's robustness against malformed ciphertexts, we:

1. Serialize valid encrypted inputs.
2. Randomly flip 5 bytes in the ciphertext.
3. Attempt to deserialize and perform inference.

Each successful deserialization and inference is logged, and we measure the accuracy as a metric of attack success. We will run 100 trials.

5.7 Fault Injection Attack

Here, we simulate memory-level bit flips:

1. Encrypt a clean input, infer logits, and decrypt.
2. Flip 5 random bytes in the encrypted input.
3. Re-run inference and decrypt output.
4. Compute average per-dimension output change.

We will run 100 trials, and we will measure the ability of the tampered inputs to produce valid ciphertext or coherent output vectors as a metric of success.

5.8 NN-Indistinguishability Game

We formalize and implement a cryptographic-style indistinguishability game:

1. The adversary selects two inputs (x_0, x_1).
2. The challenger randomly picks one (x_b), encrypts and feeds it through the model.

3. The adversary is shown the output ciphertext and must guess b.
4. We gave the adversary the ability to adapt and learn. We allowed it to collect features of the ciphertext and learn based on statistical correlation between the inputs and outputs so that it could develop decision thresholds to help guess the input.

We evaluated the above procedure for 1,000 rounds. We measure the ability for the adversary to successfully select the correct input as our metric for success.

5.9 Real vs. Ideal Game

This game simulates a setting where a ciphertext may be either:

- The encrypted output of a real model.
- A random vector of the same length, encrypted. This simulates something from the ideal world.

Similar to the above, we evaluated the above procedure for 1,000 rounds. The attacker randomly selects an input on the dataset and randomly gets either the real output or an ideal/random output from the input. We also allowed the attacker to collect data based on the features of the ciphertext and develop decision thresholds, so that the attacker could better guess whether an output was from the real or ideal world. And again, the attacker's ability to select the correct input from the two will be our metric for success.

5.10 Visualizations and Statistical Validation

We generate violin plots and bar charts for each attack class using Seaborn and Matplotlib, and conduct Welch's t-tests to evaluate statistical significance between black-box and HE settings across:

- Membership Inference AUC
- Model Extraction Accuracy
- Gradient Inversion MSE
- Input Inversion Fitness

These quantitative tests further substantiate the empirical indistinguishability guarantees of homomorphic inference.

5.11 Design Rationale

Our experimental choices were driven by our desire for interpretability and representativeness.

Dataset (MNIST). MNIST's 28×28 grayscale digits are intentionally small, allowing fully homomorphic inference to complete in seconds rather than hours while still exposing non-trivial class structure. Because the pixels are human-interpretable, any success or failure of reconstruction attacks can be inspected visually, avoiding the need for downstream decoders.

Model architecture (784–128–10 MLP). A two-layer perceptron is the shallowest network that (i) reaches $> 95\%$ plaintext accuracy and (ii) stays within the noise-budget limits of a CKKS circuit at depth $L = 3$. We thought that deeper or convolutional models would confound the isolation of security effects with engineering artifacts.

Activation replacement (quadratic). ReLU is not polynomial and would otherwise explode ciphertext noise after each multiplication. We thought a second-degree approximation balances the quality of our implementation against multiplicative depth, preserving classification accuracy while avoiding costly bootstraps.

CKKS parameters. We adopt a polynomial modulus degree $n = 8192$ and a 40-bit three-level coefficient chain. This satisfies the HE-standard 128-bit security margin and leaves one spare level for the output layer, which is enough to evaluate the full inference graph without relinearization overhead.

Attack surface. We selected attacks that probe different sources of vulnerability: (i) output-side privacy (membership inference), (ii) parameter confidentiality (model extraction), (iii) input leakage via gradients, (iv) ciphertext malleability (malicious queries / fault flips), and (v) formal indistinguishability. Covering all five ensures that a positive result cannot be explained by the absence of some dimension of vulnerability.

6 Results and Discussion

We will highlight a few notable results below and eventually summarize all our findings.

6.1 Membership Inference and Model Extraction

Table 1 reports the effectiveness of membership inference and model extraction attacks under both black-box and Poly-HE models. While membership inference accuracy and precision dropped marginally under encryption, AUC notably increased under Poly-HE, suggesting that some separation signal may emerge from probabilistic behavior in encrypted outputs. For model extraction, the surrogate model retained significant fidelity to both original models, but extraction accuracy declined from 89.4% in the black-box to 81.2% under Poly-HE.

Attack	**Metric**	**Black-Box**	**Poly-HE**	**Difference**
Membership Inference	Accuracy	0.5025	0.4950	0.0075
Membership Inference	Precision	0.5013	0.4975	0.0038
Membership Inference	Recall	0.9950	0.9900	0.0050
Membership Inference	AUC	0.4799	0.5317	-0.0518
Model Extraction	Accuracy	0.8940	0.8120	0.0820

Table 1: Attack metrics for membership inference and model extraction under black-box and Poly-HE models.

6.2 Gradient-Based Input Inversion Attacks

Plaintext White-box Model: The attacker successfully reconstructed a target input with perfect alignment:

$$\text{Best Fitness (Black-Box)} = 1.0000$$

Encrypted Model (HE): The attacker, lacking decrypted feedback, used randomized fitness and failed to converge meaningfully:

$$\text{Best Fitness (HE)} = 0.8884$$

This gap reflects the infeasibility of reconstruction under encryption.

6.3 Real vs. Ideal Game

In the real-ideal setting, the adversary observed encrypted model outputs or random ciphertexts and attempted to distinguish them. Final guessing accuracy was:

$$\text{Accuracy} = 49.3\%$$

Despite feature analysis and decision rule optimization, the attacker remained statistically indistinguishable from random guessing.

6.4 Security Findings From All Attacks & Games

Attack Type	**HE Model Security**	**Attacker Advantage**
Membership Inference	Partial Vulnerability	Minor (AUC $\uparrow$ under HE)
Model Extraction	Partial Vulnerability	Moderate (8.2% drop)
NN-Indistinguishability	Secure	None (48.4% vs. 50% (random guessing))
Malicious Query	Secure	None (0/100 successes)
Fault Injection	Secure	None (0/100 successes, ∞ output divergence)
Input Inversion	Secure	None (random fitness convergence)
Real-Ideal Game	Secure	None (49.3% vs. 50% (random guessing))

Table 2: Security summary across all evaluated attacks on HE-protected neural network inference.

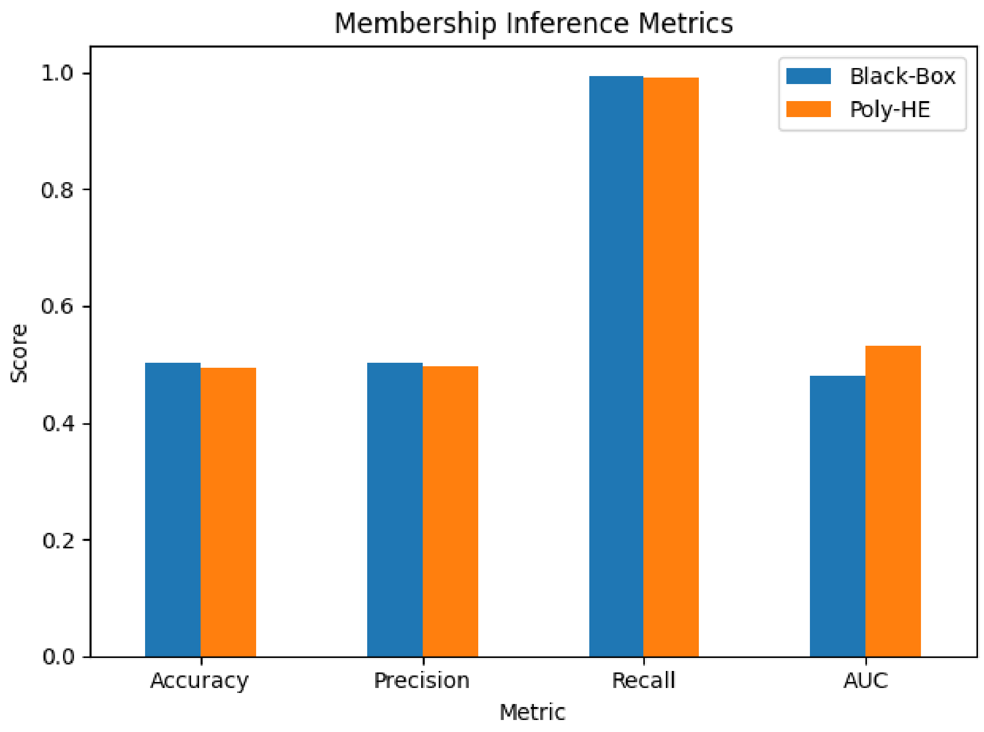

Figure 1: Visualization of Membership Inference Metrics

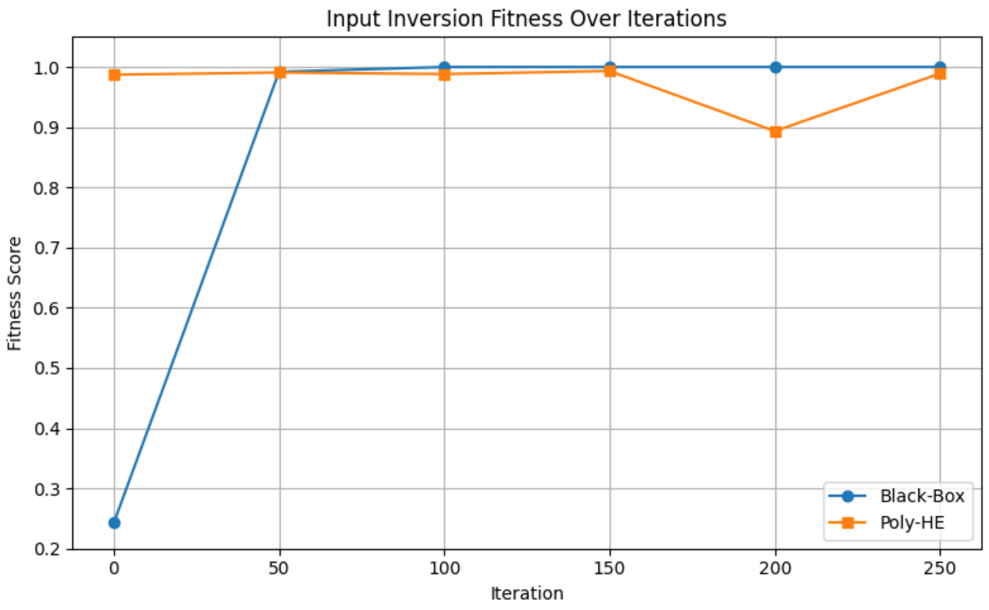

Figure 2: Visualization of Input Inversion Fitness

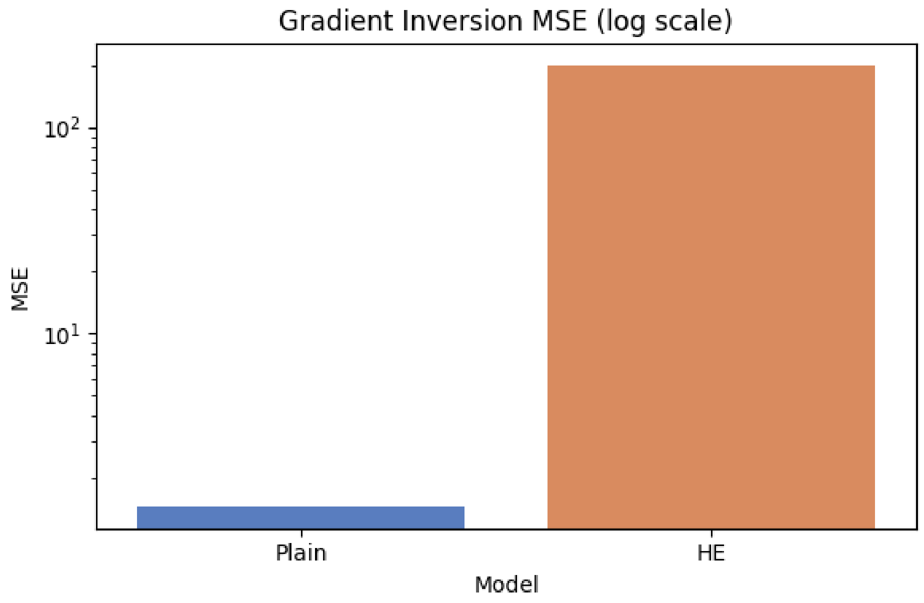

Figure 3: Gradient Inversion Graph

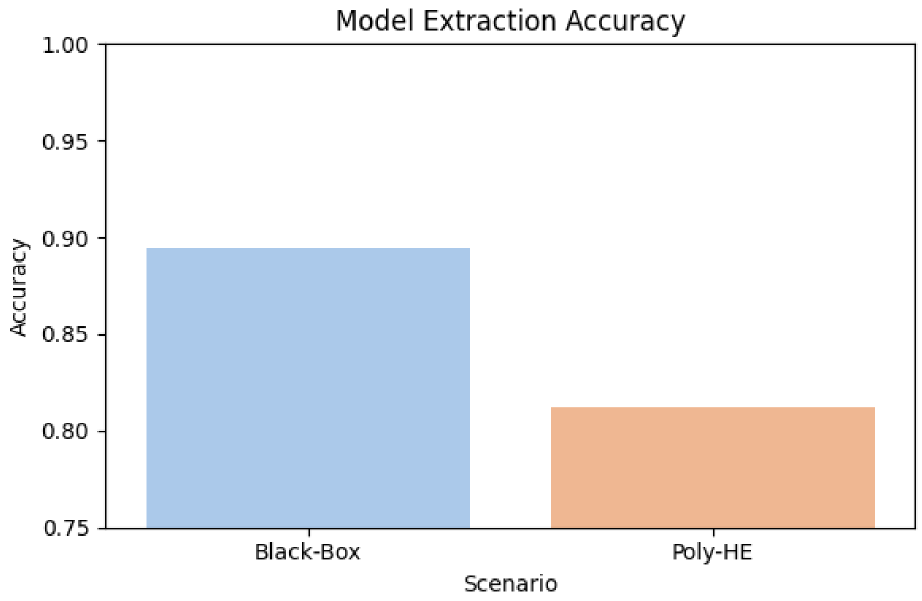

Figure 4: Model Extraction Accuracy Graph

Our results demonstrate that homomorphic encryption robustly protects sensitive inputs and model parameters, especially against sophisticated attacks like indistinguishability games, malicious queries, fault injections, and input inversion. While membership inference and model extraction still pose partial risks due to plaintext output leaks, the confidentiality of inputs and weights remains uncompromised under encryption.

6.5 Discussion

The results of our indistinguishability security games affirm that neural network inference performed entirely on ciphertexts using CKKS homomorphic encryption in TenSEAL offers strong semantic security. Specifically, when outputs remain encrypted, adversaries cannot distinguish between encrypted outputs of different inputs better than random chance, as both games converge to $\approx 50\%$ attacker accuracy, mirroring the guarantees of IND-CPA encryption. This is precisely the security promise homomorphic encryption was designed to deliver: without decryption keys, ciphertext morphology behaves as random noise.
Our experiments on fault injection and malicious queries further reinforce this robustness. Even with deliberate bit-level tampering or malformed ciphertexts, the TenSEAL runtime rejected or neutralized such attempts without producing exploitable patterns or crashes. These outcomes reflect a mature underlying library that preserves cryptographic soundness against low-level attacks.

Further, in the plaintext setting, gradient descent reconstructs digit silhouettes with an MSE of 1.37. Under HE the same optimisation degenerates into randomness (MSE ≈ 140), confirming that gradient signals are fundamentally opaque once encrypted. Interestingly, even SPSA, requiring only zeroth-order queries, fails because the attacker cannot observe perturbed logits. However, our findings also reveal limitations, both in the protection scope of homomorphic encryption and in the quality of our implementation.

First, under permissive threat models where plaintext outputs are returned (a likely scenario in many deployments), residual leakage remains possible. Membership inference accuracy, although marginally reduced, remained close to 50%, and surrogate models could still partially replicate the behavior of the original model, albeit with degraded quality. Membership inference AUC climbs from the random chance 0.50 to 0.53, which is statistically small but systematic extraction still reproduces $\sim 81\%$ of baseline predictions. In other words, HE eliminates weight-stealing but cannot mask whatever you explicitly reveal. Production deployments should therefore combine HE with output-sanitization (e.g. clipping, DP noise) or strict access control, so that the model is better protected. Thus, our results suggest that output-level leakage remains a realistic risk absent additional privacy layers like output perturbation or access controls.

Second, our implementation was limited by several factors:

- The polynomial activation function we used was a simple second-degree approximation of ReLU, chosen for HE compatibility but suboptimal for expressivity. This may have introduced accuracy degradation and reduced robustness in the Poly-HE model.
- Homomorphic inference was performed by manually reconstructing forward passes outside of native deep learning frameworks, limiting our ability to scale to deeper models or convolutional architectures.
- Our attacks, though implemented faithfully, operate under simplifying assumptions (e.g., evolutionary strategies for input inversion with fixed parameters, limited perturbation scopes for fault injection), and may not capture the full adaptive behavior of sophisticated adversaries.

Overall, our evaluation supports homomorphic encryption as a powerful base layer for secure inference, particularly in scenarios where strict ciphertext-only outputs can be maintained. Nonetheless, deployment should consider attack surfaces exposed through partial output access and evaluate appropriate mitigation strategies on a case-by-case basis.

7 Conclusion

We presented an end-to-end demonstration of homomorphic encryption for neural network inference. Through a controlled deployment of a polynomial-activated MLP using the TenSEAL library and CKKS scheme, we investigated a variety of potential attacks, like membership inference, model extraction, gradient-based input inversion, and cryptographic indistinguishability games, under both plaintext and encrypted settings. Our results suggest that, while membership inference and black-box extraction remain partially viable under homomorphic encryption, the effort required to invert or reconstruct inputs via gradient-based approaches increases dramatically.
In future work, we aim to:

- Explore higher-order or domain-specific activation approximations compatible with HE that maintain better learning capacity.
- Extend our analysis to support convolutional and recurrent architectures under encrypted inference pipelines.
- Evaluate joint defenses that combine HE with techniques like secure enclaves, differential privacy, or federated learning.
- Conduct longitudinal security evaluations across longer adaptive attack sessions to assess cumulative leakage over time.

In conclusion, we think our study provides a step toward quantifying the security of homomorphic neural network inference and motivates continued research into both cryptographic innovation and deployment-aware defense strategies.

References

[1] C. Gentry, *Fully Homomorphic Encryption Using Ideal Lattices*, In STOC, 2009.

[2] R. Shokri et al., *Membership Inference Attacks Against Machine Learning Models*, In IEEE S&P, 2017.

[3] F. Tramèr et al., *Stealing Machine Learning Models via Prediction APIs*, In USENIX Security, 2016.

[4] M. Fredrikson et al., *Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures*, In CCS, 2015.

[5] M. Abdalla et al., *Public-Key Encryption Indistinguishable Under Plaintext-Checkable Attacks.* In: Katz, J. (eds) Public-Key Cryptography – PKC 2015. PKC 2015.

[6] J. Byun et al., *Privacy-preserving inference resistant to model extraction attacks. Expert Systems with Applications 256, 2024*

[7] Al-Rubaie et al., *Privacy-preserving machine learning: Threats and solutions*, In IEEE Security and Privacy 17.2, 2019.

[8] Biggio, Battista et al., *Evasion attacks against machine learning at test time*, In Machine learning and knowledge discovery in databases: European conference, 2013.

[9] F. Kelbert et al., *SecureCloud: Secure big data processing in untrusted clouds*, Design, Automation & Test in Europe Conference & Exhibition , 2017, Lausanne, Switzerland

[10] D. Dolev et al., *Nonmalleable Cryptography*, SIAM Journal on Computing 30(2), 391–437 (2000)

[11] Aydin et al., *Leaking secrets in homomorphic encryption with side-channel attacks* Journal of Cryptographic Engineering 2024

[12] Li et al., *On the Security of Homomorphic Encryption on Approximate Numbers.* In: Canteaut, A., Standaert, FX. (eds) Advances in Cryptology – EUROCRYPT 2021.

[13] Cheon et al., *Attacks Against the IND-CPAD Security of Exact FHE Schemes* In Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security (CCS '24).

[14] R. Gilad-Bachrach et al., *CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy*, In ICML, 2016.

[15] C. Juvekar, V. Vaikuntanathan, A. Chandrakasan, *GAZELLE: A Low Latency Framework for Secure Neural Network Inference*, In USENIX Security Symposium, 2018.

[16] R. Li, S. Wang, W. Zheng, X. Wang, *SoK: Fully Homomorphic Encryption Compilers*, In IEEE Symposium on Security and Privacy (S&P), 2021.

[17] A. Kaissis et al., *Privacy-Preserving in Medical Image Analysis: A Review of Methods and Applications*, In Nature Machine Intelligence 2, 305–311, 2020.

[18] X. Sun et al., *Blockchain and Homomorphic Encryption Based Privacy-Preserving Model Aggregation for Medical Images*, In IEEE Journal of Biomedical and Health Informatics, vol. 25, no. 12, pp. 4954–4963, 2021.

[19] Y. Liu et al., *MedBlindTuner: Towards Privacy-preserving Fine-tuning on Biomedical Images with Transformers and Fully Homomorphic Encryption*, In NeurIPS, 2023.

[20] J. Cheon, A. Kim, M. Kim, Y. Song, *Homomorphic Encryption for Arithmetic of Approximate Numbers*, In ASIACRYPT, 2017.

[21] A. Benaissa et al., *TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption*, arXiv, 2021.

[22] B. Brakerski, Y. Polyakov, V. Vaikuntanathan, *Efficient Homomorphic Conversion Between (Ring) LWE Ciphertexts*, in ePrint, 2020.

[23] J. Shin, S.-H. Choi, Y.-H. Choi, *Is Homomorphic Encryption-Based Deep Learning Secure Enough?*, In Sensors, 2021.

[24] I. Chillotti et al., *TFHE: Fast Fully Homomorphic Encryption Over the Torus*, Journal of Cryptology, 2020.

[25] P. Mohassel and Y. Zhang, *SecureML: A System for Scalable Privacy-Preserving Machine Learning*, in Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), 2017.

[26] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, *Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications*, in Proceedings of the 2018 ACM Asia Conference on Computer and Communications Security (ASIACCS), 2018.