Homomorphic Encryption for Neural Network Deployment: An Empirical Security Evaluation

Michael Xiang, Steve Dalla, Gary Wu

April 15, 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.

1 Introduction and Motivation

As machine learning becomes increasingly common in sensitive domains (e.g., healthcare, finance, or government), protecting confidential inputs and proprietary model parameters has become a top priority. Given the increasingly large data and compute demands of modern machine learning, many organizations choose to host models and run sensitive inputs with them on cloud service providers. Traditionally, service providers must trust a cloud platform not to exfiltrate or inspect raw data. However, homomorphic encryption (HE) 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.

Despite these promises, practical questions persist:

- 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 be performed when only the encrypted model is available?

Our work addresses these by implementing a demonstration in TenSEAL[1], comparing an encrypted polynomial MLP with a standard unencrypted ReLU-based MLP on MNIST.

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

2 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.

Privacy-preserving inference resistant to model extraction attacks Byun et al. [5] showed ways to integrate additive Homomorphic Encryption to neural networks but is focused on multi-party computation instead of computation run on one wholly connected system.

Privacy-preserving machine learning: Threats and solutions Al-Rubaie et al. [6] was a seminal landscape paper we're building off of.

Evasion attacks against machine learning at test time Biggio, Battista et al. [7] was a seminal work in the ML attacks space.

3 Experimental Setup

3.1 System Architecture

The system consists of a client and an untrusted cloud server. The client owns an already-trained neural network model and uses homomorphic encryption to protect it and input data:

- **Client Side:** The client holds the secret key for the HE scheme. It encrypts the model's weights and any input data before uploading or sending them to the cloud. After computation, the client will decrypt the results locally.
- **Cloud Server:** The cloud hosts the encrypted model and performs forward inference on encrypted inputs. The activation functions in the neural network are replaced with polynomial or other HE-friendly approximations (since standard ReLU or non-polynomial operations are not directly supported under HE). The cloud never sees raw data or raw model weights; it only deals with ciphertexts. Once the computation is done, the server returns only the final output in encrypted form to the client, who then decrypts it.

By using this architecture, we can ensure that throughout the inference process, all intermediate values (inputs, activations, and weights) remain encrypted on the server. We implement this using the CKKS scheme (a leveled HE scheme for real-number encryption) via the TenSEAL library. Thus, the threat model assumes the cryptographic primitives are correctly implemented and the encryption is semantically secure (IND-CPA secure).

3.2 Models

We consider a setting where an untrusted cloud runs an MLP for digit classification on MNIST. The model is:

- **Plain MLP:** A standard two-layer feedforward neural network (multi-layer perceptron) with ReLU activations, deployed in a normal (unencrypted) manner. This represents the conventional cloud inference scenario where the server has full access to model weights and outputs are returned in plaintext.
- **Homomorphic MLP:** The same network architecture, except with encrypted weights and inputs, and using a polynomial activation function approximation instead of ReLU (to enable computation over ciphertexts). We use the CKKS homomorphic encryption scheme via the TenSEAL library for this implementation. The server performs inference on ciphertexts, and only the final result is decrypted by the client.

Both models are trained on the same task: MNIST handwritten digit classification. This task involves 28×28 grayscale images of digits 0–9, a common benchmark for inference experiments. We chose MNIST for its simplicity and the fact that attacks like membership inference and inversion are feasible to test on this data (and images allow visualizing reconstruction attempts).

3.3 Attacks

For our project, we consider an attacker that operates or has infiltrated the cloud server. We assume that the attacker has the following capabilities:

- Full access to the neural network's encrypted weights and architecture on the server. They know the network structure and see the weights in encrypted form, however, they do not have access to the HE decryption key.
- In the most strict scenario, the attacker sees only encrypted outputs (which are useless without the key). However, we are considering for certain attacks an attacker who learns the decrypted output or some function of it. For example, in a typical machine-learning-as-a-service, the client might send back a prediction result to another party. If the final predictions are not kept secret by the client, an attacker could obtain the plaintext outputs. We have not focused on this as of now, but hope to perform attacks related to this more strict setting.

Given these abilities outlined above, we define the attacker's goals as follows:

- Reconstruct or infer sensitive details about a specific input that was given to the model.
- Reconstruct the model's parameters or an equivalent functioning copy of the model. The attacker may attempt to either extract the exact weights (which is essentially decrypting the model) or train a surrogate model that closely approximates the behavior of the target model.

Thus far, we have conducted two main experiments to evaluate security:

1. **Membership Inference (MI):** We tested the approach of thresholding predicted confidence to distinguish training vs. non-training samples.
2. **Model Extraction (ME):** We used a query-based method to train a surrogate model that mimics the true model's outputs.

We are also working to implement the following attacks:

1. **Gradient Attack:** We plan to performed a standard gradient-based inversion on the plain model (white-box). For the homomorphic model, we are implementing a naive finite-difference approach on encrypted queries, attempting to approximate gradients for input reconstruction.

2. **NN-Indistinguishability Security Game:** To reason formally about the security of inputs under homomorphic encryption, we define an indistinguishability game for the neural network's encrypted inference. This game is similar to the IND-CPA game for encryption, but tailored to the context of an encrypted neural network service:

   - The attacker is given the encrypted model and the system's public parameters. They do not have the secret key. They can also query the model as an oracle: they can submit inputs of their choice and receive the corresponding encrypted outputs. The attacker picks two distinct plaintext inputs, x_0 and x_1, that they want to feed the NN. We then randomly choose one of these inputs, x_b with $b \in 0, 1$ uniformly at random, and homomorphically encrypt it using the client's public key. The encrypted input is fed into the encrypted model, producing an encrypted output $ct_y = \text{Enc}(f(x_b))$, where f is the neural network's inference function. The encrypted output ct_y is returned to the attacker. Based on everything the attacker knows, they must guess whether the output corresponds to x_0 or x_1. Essentially, they output a guess b' for the secret bit b. The attacker wins if they correctly guessed which input was used. We will define our HE-based neural network as secure if no polynomial-time attacker can win the game with a higher probability than randomly guessing.

4 Results and Discussion

4.1 Membership Inference and Model Extraction

Table 1 shows membership inference and model extraction accuracy under both black-box and Poly-HE conditions. We see that the membership inference results differ by only 1–2% for accuracy and precision, which suggests that purely from final outputs, homomorphic encryption does not drastically reduce membership inference in our simple experimental setting. For model extraction, the surrogate model matched the black-box model's predictions with about 87% accuracy, versus 82% for the encrypted polynomial net.

Attack	**Metric**	**Black-Box**	**Poly-HE**	**Difference**
Membership Inference	Accuracy	0.505	0.495	0.010
Membership Inference	Precision	0.503	0.497	0.005
Membership Inference	Recall	1.000	0.990	0.010
Membership Inference	AUC	0.443	0.485	-0.042
Model Extraction	Accuracy	0.872	0.824	0.048

Table 1: Attack results. Overall, performance remains similar under encryption, though model extraction success is somewhat lower for the HE model.

4.2 Gradient-Based Input Recovery Attack

Still working on code and results.

4.3 Indistinguishability Game

Still working on code and results.

5 Conclusion

We presented an end-to-end demonstration of homomorphic encryption for neural network inference, focusing on security evaluations including membership inference, model extraction, and gradient-based input reconstruction. 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. Future work will explore more sophisticated attacks and side-channels, as well as better polynomial approximations of activation functions to reduce accuracy loss in homomorphic settings.

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

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

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