FHE-MENNs: Accelerating Fully Homomorphic Private Inference with Multi-Exit Neural Networks

Lars Wolfgang Folkerts
Electrical And Computer Engineering
University of Delaware
Newark, DE, USA
folkerts@udel.edu

Nekatrios Georgios Tsoutsos
Electrical And Computer Engineering
University of Delaware
Newark, DE, USA
tsoutsos@udel.edu

***Abstract*—With concerns about data privacy growing in a connected world, cryptography researchers have focused on fully homomorphic encryption (FHE) for promising machine learning as a service solutions. Recent advancements have lowered the computational cost by several orders of magnitude, but the latency of fully homomorphic neural networks remains a barrier to adoption. This work proposes using multi-exit neural networks (MENNs) to accelerate the FHE inference. MENNs are network architectures that provide several exit points along the depth of the network. This approach allows users to employ results from any exit and terminate the computation early, saving both time and power. First, this work weighs the latency, communication, accuracy, and computational resource benefits of running FHE-based MENN inference. Then, we present the TorMENNt attack that can exploit the user's early termination decision to launch the first-ever concrete side-channel on MENNs. We demonstrate that the TorMENNt attack can predict the private image classification output of an image set for both FHE and plaintext threat models. We discuss possible countermeasures to mitigate the attack and examine their effectiveness. Finally, we tie the privacy risks with a cost-benefit analysis to obtain a practical roadmap for FHE-based MENN adoption.**

***Index Terms*—Fully Homomorphic Encryption, Multi-Exit Neural Networks, Privacy-Preserving Machine Learning.**

I. Introduction

As the world becomes more connected, cloud services have become an increasingly popular solution for businesses. In this paradigm, users send their data to the cloud for processing, allowing the user to offload the computational cost and utilize the cloud service provider's proprietary algorithms. For Machine Learning as a Service (MLaaS), these proprietary algorithms are trained neural networks, which require lots of data and processing power to develop. It is not always feasible for users to train neural networks on specialized datasets, making MLaaS a prevalent solution [1], [2].

Traditional neural network architectures have grown deeper as researchers try to obtain higher accuracies. While accuracy gains improved at first, they reached a wall in recent years. This means that as neural networks get deeper and deeper, accuracy gains decrease, and computational costs increase. Another drawback is that deep neural networks suffer from the *vanishing gradient problem*, making it challenging to train deep networks all at once [3]–[5].

Multi-exit neural networks (MENNs) were introduced to address these issues. The central concept of MENNs is that easy-to-process inputs can take preliminary results from an early exit and terminate the computation quickly. For example, [4] shows that many ImageNet dataset inputs can have a good prediction after a few layers and can terminate earlier; other inputs may need to execute the remaining layers for further processing to obtain more accurate results. As many inputs can exit early, MENNs provide the opportunity to save computation time without large drops of precision. As an additional benefit, MENNs executed to the end provide an ensemble of neural network outputs. This ensemble of multiple exits can be combined to increase the prediction accuracy beyond a single-exit network [6]–[8], or improve the confidence score [9]. Finally, MENNs can address the vanishing gradient problem when implemented as several cascaded neural networks, trained one by one.

MENNs easily adapt to the machine learning as a service (MLaaS) paradigm. In this case, users send their data to the cloud, and the cloud runs the first segment of the network and additional exit layers to give the user an early result. If the user is happy with the *confidence of the result*, they can tell the cloud to terminate the computation. However, if the result has low confidence, the user tells the cloud to continue to the next network segment [10], [11].

Unencrypted MENNs have been exhibited in applications such as autonomous driving [12] and speech recognition [13]. They can either operate in fast edge computing [14], MLaaS [10], and low-power computing environments [15]. One promising, yet unexplored environment of MENNs is fully homomorphic encryption (FHE)-based privacy-preserving machine learning (PPML). Single exit FHE-based PPML uses state-of-the-art encryption techniques to execute computations on ciphertexts, allowing neural networks to perform inference on encrypted user data. This paradigm allows the cloud to maintain full control of its model IP and MLaaS service, while protecting user data from being revealed to a curious cloud service provider. The main drawback of FHE-based PPML is that it is computationally expensive, resulting in large inference times and high dollar costs for the user. For example, a nine layer CIFAR-10 network takes 76 minutes to compute on a 96-CPU core r5.24xlarge AWS server [16]. Due to this slow computational speed, MENNs are a potential technique to obtain faster FHE-based PPML inference.

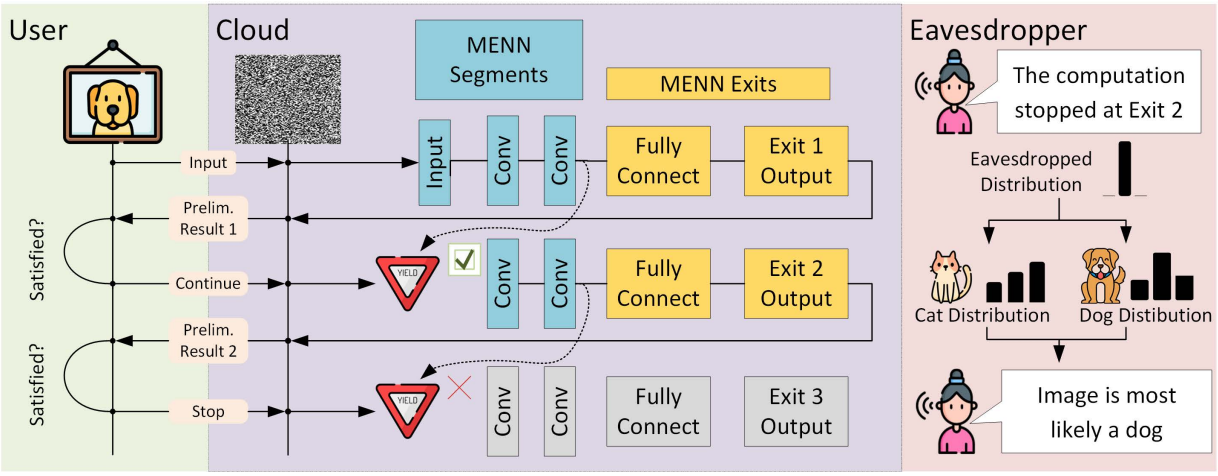


Fig. 1. **FHE-MENNs and the TorMENNt Attack:** In a multi-exit neural network (MENN), preliminary results can be given back to the user. A user can decide, based on the confidence score (e.g., entropy) of the preliminary results, whether they wish to terminate the computation early. An attacker can easily exploit this information to predict the user's inputs. This attack can affect both a plaintext and an encrypted privacy-preserving cloud model.

This work analyzes the potential of FHE-based MENNs to save computational cost and latency. As part of this analysis, we present TorMENNt, a weak side-channel attack on multi-exit neural networks that enables the attacker to recover the classification results of the user. TorMENNt is based on the user inherently leaking information by deciding to terminate the computation early, as shown in Figure 1. The TorMENNt attack is applicable to both unencrypted and FHE-based PPML MENNs.

Overall, our contributions are summarized as follows:

- We analyze the use of MENNs for FHE-based PPML across a variety of architectures and PPML frameworks;
- We expose TorMENNt, a new side-channel attack that impacts a broad variety of modern MENNs;
- We demonstrate how MENNs can be adapted for FHE-based PPML and evaluate the performance gains and the security risks of several MENN inference schemes.

The rest of the paper is organized as follows: Section II covers important background information about MENNs and PPML, while Section III defines the two threat models to be attacked. Section IV examines the benefits of using FHE-based MENNs, while Section V defines the theoretical foundations of the proposed TorMENNt attack and demonstrates its execution and mitigation strategies. Section VI elaborates on how these findings can be adapted to real-world applications. Finally, Section VII presents related works and our concluding remarks are discussed in Section VIII.

II. Background

In this section, we provide the necessary background to issue the TorMENNt attack. In Section II-A, we first present an overview of multi-exit network architectures and how they can be used. Then, in Section II-B, we present an overview of homomorphic encryption schemes. Finally, in Section II-C, we present current FHE-based PPML speedup techniques.

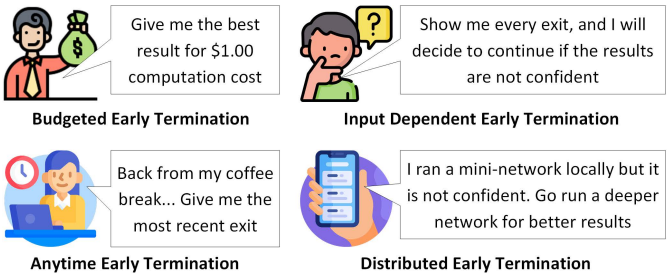


Fig. 2. **Multi-Exit Early Termination Schemes:** A simplified version of the four early termination schemes, where the users are talking to the cloud.

A. Multi-Exit Learning

1) Overview of Multi-Exit Neural Networks: Multi-Exit Neural Networks (MENNs) were developed to speed up early termination in real-time embedded systems, such as video detection. They are sometimes called by different names, including early-exit neural networks [17] and cascaded networks [4], [18]. The original goal of MENNs was to limit **overthinking**, which occurs when deep neural networks are passed easily classified inputs and process them far longer than necessary. Early works found that if the neural network can determine the difference between easy and challenging inputs, then easily classified inputs can exit early in a smaller network, whereas challenging inputs can be processed through additional layers [17], [18]. Other works proved that training of multi-exit neural networks helps improve overfitting and mitigates the vanishing gradient problem [4], [19].

2) Multi-Exit Early-Termination Schemes: There are four main types of early termination schemes to determine which neural network exit is best to take during inference. These types of early termination are summarized in Figure 2.

In the **budgeted** early termination scheme, the user or system supplies the cutoff constraint for running the neural network *upfront*. For example, [15] determines whether to run the complete neural network based on the remaining battery

life for the embedded device. In contrast, [14] implies a time-based constraint for its networks, where the system can request the best result for a predetermined amount of time.

In the **anytime** early termination scheme, the cloud will attempt to run the complete network with all possible exits. The user can interrupt this process early, based on external constraints, and ask for the results from the most recent exit. After the first exit is computed, the cloud should always have a result for the user, and this result has increased the accuracy of being correct over time [7], [10], [14]. Unlike the budgeted early termination scheme, the anytime early termination scheme does not require the budget constraints to be known upfront.

Next, the **input-dependent early termination scheme** outputs a confidence score of how accurate the result is and lets the user decide if they want higher confidence. The user can specify they want 90% confidence in their result, and the network can run until an exit exhibits their confidence score. Different proxy functions can be used to determine this confidence score [10], [14], as discussed in the next section.

Lastly, the **distributed early termination scheme** is similar to the input-dependent early termination scheme, except one network is run locally. This scheme uses a shallow neural network for local computation and a deep neural network for cloud computation; no data needs to be sent to the cloud if the user is satisfied with the confidence score after local computation. The intent is to use this type of early termination on devices with poor network connections; if the local computation is accurate, a fast and low-power result is achieved without needing to wait for a reliable network connection [11].

Our work is primarily concerned with the input-dependent early termination scheme, but all FHE-MENN early termination schemes are discussed in Section VI-A.

3) Confidence Scores: An effective confidence score for the input-dependent early termination scheme is the subject of recent works. Many works [3], [6], [17], [18] use an entropy-based confidence score

$$C_{entropy} = -\sum_{i=1}^{num\ classes} c_i * \log c_i, \tag{1}$$

where c_i is the softmax output of the i-th class.

A second common confidence score used is a max-min difference between values [8], [11], [18], [20], [21]. This is represented as

$$C_{max-min} = max(c_0, ..., c_i) - min(c_0, ..., c_i), \tag{2}$$

and is simplified in some works to just the max function

$$C_{max} \approx max(c_0, ..., c_i). \tag{3}$$

Another option is to use a secondary neural network to determine entropy [9], [18], and is defined as

$$C_{ML} = \sigma(f(x)), \tag{4}$$

where $\sigma()$ is the sigmoid function, x is an intermediate feature representation, and $f()$ is a separately trained neural network confidence predictor function.

B. PPML Schemes

MLaaS is a common application that privacy-preserving computation looks to solve and has become known as PPML. Researchers are investigating ways to develop fast, deep, and private neural networks that require little computational power and bandwidth from users. An overview of different PPML encryption schemes and techniques is discussed below.

1) Leveled Homomorphic Encryption: Leveled Homomorphic Encryption (LHE) is a cryptographic technique that offers the ability to perform operations, such as addition and multiplication, on a ciphertext. LHE-based encryption schemes rely on lattice cryptography and the learning with errors (LWE) problem [22]. Specifically, the LWE problem states that if small amounts of noise are added to a transformed high-dimensional value in a lattice, it is NP-hard to reverse the transformation of the value into a plaintext. This transformed value is the ciphertext, and its noise is essential to its post-quantum resilience. However, this noise grows as HE operations (i.e., addition and multiplication) are performed; once the noise grows beyond a certain level, the ciphertext cannot be decrypted correctly with the decryption key. Therefore, LHE only supports a limited number of operations known as its *multiplicative depth*.

For HE-based PPML, the users typically send their data to the cloud for neural network classification using the cloud's own weights, and the cloud returns the computation result. The user is only involved in the encryption and decryption process, which is a major benefit for HE-based PPML schemes. LHE-based schemes support *packing* of multiple inputs into a single ciphertext during the encryption process. This allows for vectorized computation and high throughput image classification. Nevertheless, a major drawback of LHE is the multiplicative depth limits the computation to shallow neural networks, while supported activation functions can only be expressed as polynomial approximations.

Popular LHE-based encryption libraries include HELib [23], which supports the BGV [24] and CKKS [25] encryption schemes, as well as SEAL [26] that supports the BFV [27] encryption scheme.

2) Fully Homomorphic Encryption: Fully Homomorphic Encryption (FHE) is an extension of LHE, adding a process called bootstrapping to reduce ciphertext noise and allow for unlimited operations. Notably, bootstrapping is computationally expensive, so its use must be limited as much as possible. Recent works have added support for programmable bootstrapping (PBS) [28], or univariate function evaluation during a bootstrap, and bi-directional bridging between binary and integer ciphertexts [16]. These two improvements separately enable fast inference on neural networks.

Popular FHE libraries for PPML include Concrete [29] and (RED)cuFHE [16], both of which use the TFHE [30] encryption scheme.

3) Multi-Party Computation: Multi-Party Computation (MPC) is a cryptographic protocol where different parties work together on a shared computation. This can be achieved using secret sharing, garbled circuits [31], and partially homomorphic encryption. For MPC-based PPML between two parties, users typically send their data to the cloud for convolution using the cloud's weights, and the cloud sends the data back to the users for computing activation functions. Notably, this is a *user-centric approach*, as users are actively involved in the computation. MPC solutions also incur higher communication overheads between users and the cloud. These requirements can make it infeasible for certain applications, where edge devices have low computation power or slow network connections.

In MPC-based PPML schemes with secret sharing, the user data can be divided between two (or more) servers, removing the user from most of the computation. However, this approach is based on a somewhat weaker threat model that assumes the cloud servers can never collude. It is also less practical for an end-user to contract two non-colluding cloud servers instead of one. This loss of security and loss of ease of implementation make this multi-cloud scheme less ideal for high-security applications. Popular MPC-based PPML solutions include Cheetah [32], Gazelle [33], and MiniONN [34].

C. Techniques for Efficient PPML in FHE

Privacy-preserving machine learning has different computational costs than plaintext computation. In general, encrypted additions are low cost, encrypted multiplications are medium cost, and encrypted comparisons are high cost. In this work, we analyze the performance of PPML networks based on the TFHE encryption scheme. TFHE can efficiently perform Boolean operations, such as bit shifts and logic gates.

Approximation and discretization can be used to speed up PPML computation times. There exists two main frameworks for TFHE-based PPML: REDsec [16] and Concrete [29]. In this section, we discuss what approximations these frameworks use to achieve fast FHE-based ML inference.

1) REDsec Optimizations: REDsec obtains its speedups with a technique called *bidirectional bridging*, which converts integer ciphertexts into several binary ciphertexts represented as encrypted bits {0,1}. This allows implementing efficient neural networks with ternary weights $\{-1,\ 0,\ +1\}$. Here, inputs multiplied by -1 use the univariate NOT gate for 1's complement and add 1 for 2's complement later on. Inputs multiplied by 0 are ignored. Finally, inputs multiplied by $+1$ stay as-is. These operations are extremely PPML-friendly and speed up computation significantly over PPML multiplications.

REDsec further utilizes binary ciphertexts for non-linear activation functions. Here, the sign function is used

$$sign(x) = \begin{cases} -1 & x < 0 \\ +1 & x \geq 0 \end{cases} \tag{5}$$

to restrict the convolution layer inputs to ± 1, further simplifying computational cost. This reduces binary weight multiplication to a single XNOR gate. In REDsec this is a special XNOR gate that takes inputs in $\{-1, +1\}$ instead of the traditional $\{0, +1\}$, and it can be viewed as a multiplication function. ReLU is also possible by ANDing the inverse sign bit with the remaining ciphertext bits [16], [35], [36].

2) Concrete Optimizations: Instead of bidirectional bridging, Concrete opts to use a technique called programmable bootstrapping (PBS) for non-linear operations. PBS can be viewed as an encrypted lookup table, where a multiplexer circuit can be inserted during a bootstrap operation on a ciphertext [28].

For multiplication and addition operations, Concrete uses low-precision TFHE integer ciphertexts, where most networks cap the resulting ciphertexts to 13-bits to balance speed and efficiency. For non-linear activation operations, PBS can be used for a wide range of activations: a PBS lookup table can handle any univariate function, allowing for discretized ReLU, hard tanh, and sigmoid to be used [29].

III. Threat Model

A. Definitions

The TorMENNt side-channel attack leaks information about the user input data based on the MENN exit taken. In order to model this information leakage, we assume a **semantically secure inference** game, defined as follows: Suppose a user (assuming the role of a challenger) publishes two plaintext images of identical bitsize, m_0 and m_1. The user selects a random bit b, securely encrypts m_b as C_b and uploads it to a cloud service for encrypted processing and classification, and ultimately receives an encrypted result R. Then, an eavesdropping attacker $\mathcal{A}$ with the ability to query the model (either within the cloud service or monitoring the communication channel) predicts bit $b' = \mathcal{A}(m_0, m_1, C_b, R)$, such that $\varepsilon = |Prob[b' = b] - 0.5|$. We say that $\mathcal{A}$ wins the SSI game if and only if ε is non-negligible (i.e., the eavesdropper can predict bit b with probability better than a random guess). If $\mathcal{A}$ cannot win the SSI game, the system does not leak any information and is semantically secure.

Anytime a decision is made based on derivatives of input data m_b, an attacker can gain information and break semantic security. The TorMENNt attack, which will be presented in Section V, demonstrates that deciding which exit to take will break the semantic security of MENN systems. This work further evaluates mitigation techniques in Section V-D, making it more difficult for adversaries to win the semantic security game.

B. Privacy Preserving Inference Model

The MLaaS inference model assumes an *honest but curious cloud*, identical to most FHE-based PPML works. Here a user employs FHE to protect their input from a cloud service provider who owns the neural network model. The cloud service will run the neural network inference correctly but will attempt to leak information about the user input (e.g., be incentivized to use it for advertising purposes).

C. Malicious FHE Cloud

We will also consider a malicious cloud threat model in our FHE-MENN discussion in Section VI. A malicious cloud enables the cloud to manipulate data to leak information. For FHE-based models, a malicious cloud can run an encrypted test circuit to see if a certain attribute is present. The cloud can saturate values or use an encrypted multiplexer to send false values to the user based on the results of the cloud's encrypted test.

IV. FHE MENN Performance

Due to their large computational complexity constraints, privacy-preserving machine learning technologies are good candidates for early exit models. However, despite growing interest in PPML, the use of privacy-preserving MENNs has not yet been adopted to the best of our knowledge. This section discusses the performance benefits of MENNs when applied to FHE.

A. Network Architectures and Training

For our early exit architectures, we used the Cifar-10 [37] benchmark, which contains 32×32 RGB images sorted across 10 classes. We also measured performance on TinyImageNet, which has 64×64 RGB images sorted across 200 classes. We further adapted several network architectures to incorporate multi-exits, including BinaryNet [38], [39], VGG [40] and Resnet [41].

BinaryNet was constructed using REDsec as its foundation. Exits were inserted every two layers to assess the performance of the FHE-MENN. In developing this network, it was discovered that the default MaxPooling layers in BinaryNet contributed to significant latency overhead. To address this, exits were introduced before the MaxPooling operation, allowing for early exits before encountering these high-latency computations.

Our VGG network was built using Concrete and has a structure similar to our BinaryNet network. Concrete recommends SumPooling as no MaxPooling operation is available. Since SumPooling is more homomorphically friendly in Concrete, we were able to place the early exits after the pooling layer. We also opted for fewer exits to achieve lower latency and higher security (as discussed in Section V). Finally, our ResNet architecture was also built using Concrete. We placed the early exits after each residual addition, where they seemed to naturally fit into the model.

The training process for all architectures followed a two-step approach, beginning with the initial focus on training the backbone of the network while disregarding all but the final exit layers. Then, each exit was individually trained while the backbone was concurrently fine-tuned. We used the cross-entropy loss function, which is standard for supervised learning classification tasks.

B. Selecting an Early Exit Decision Boundary

Figure 3 illustrates the accuracy versus timing outcomes for our networks. The single-exit networks exhibit fixed architectures featuring only one exit, while multi-exit architectures present accuracy and timing results based on entropy and maximum decision thresholds.

TABLE I

Network Comparisons: Comparison of performance and attack vulnerability of our different networks. The first two result columns show the classification accuracy and average time for a single image using the baseline threshold. The next two columns demonstrate how well an attacker can guess the class of a user's images based on exit data. The last column reports the user accuracy drop after the isorecall mitigation.

Model	Dataset	MENN Acc.[3]	Avg. Time[1]	Attack (Single[4])	Attack (Batch[2,4])	Isorecall Acc.[3]
Binary	Cifar	59.3%	1502s	23.5%	47.4%	52.4%
VGG	Cifar	76.1%	811s	15.8%	45.8%	73.6%
Resnet	Cifar	71.0%	1367s	18.7%	67.3%	65.7%
VGG	TImg	56.2%	5018s	1.35%	6.79%	46.1%
Resnet	TImg	56.2%	9094s	1.58%	7.53%	44.7%

[1] **Avg. Times** are for a FHE-based PPML implementation, using the base entropy discussed in Section IV-B.
[2] **Batch** averages attack results of 90-100 pictures per batch for CIFAR10 and 20-25 images per batch for TinyImageNet.
[3] **Performance Accuracy** refers to a user's classification accuracy, both raw (MENN Acc) and after the mitigation (Isorecall Acc).
[4] **Attack Accuracy** refers to how frequently an attacker correctly guessed a user's image class.

To determine these thresholds in our inference methodology, we initially used an entropy cost score described in Eq. 1 with a base threshold of $max_{entropy}/2$. This decision boundary worked well for Cifar10, although was slightly high, with many images exiting later than needed. Therefore, we reduced this threshold by 0%, 10%, 20% and 30% for our timing results in Figure 3.

For Tiny ImageNet, we found that the maximum decision metric (Eq. 3) performs slightly better than entropy. Based on our analysis, we observe that with more classes the network was less able to make confident predictions, and confusion between a few top classes combined with fluctuations of lower-ranked irrelevant classes led to noise in the overall entropy measurements. Instead we used the maximum decision boundary used a base threshold of 0.25 and modified the values in Figure 3 to be $\{0.15, 0.20, 0.25, 0.30\}$. Contrary to entropy, a higher maximum threshold means images exit later with higher accuracy and slower latency.

We also summarize our performance results across different networks in Table I; for the performance metrics, the base thresholds were used.

C. FHE-based PPML Performance

In terms of *accuracy* assessment, REDsec's plaintext mode was employed, utilizing discretized weights of $\{-1, 0, +1\}$. For Concrete, the FHE simulation mode was utilized, with weights and activations discretized to 6 bits. Notably, these simulation modes are essential when evaluating the accuracy across thousands of images. Moreover, all *timing* measurements were conducted on a c5a.24xlarge instance, and the reported times are averages based on three inference measurements in encrypted mode.

As illustrated in Figure 3, when our selected architectures operate as single-exit networks, they exhibit longer runtimes

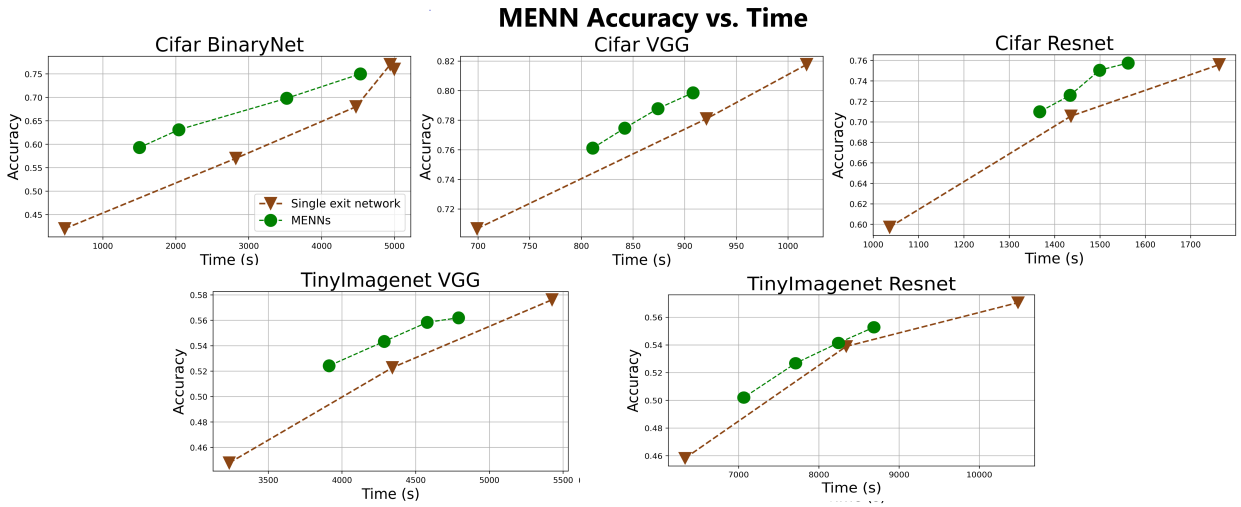


Fig. 3. **MENN Performance:** Different accuracy-timing trade-offs for PPML networks. For Cifar10 MENNs, we used a base entropy of $max\_entropy/2$. Our initial results showed that the base entropy for Cifar-10 was slightly high, so we evaluated the network three more times reducing this threshold by 10%, 20%, and 30%. For entropy, a lower threshold means later exits, resulting in slower latency but higher accuracy. TinyImagenet used the maximum softmax output value as its decision boundary, with values {0.15, 0.20, 0.25, 0.30} serving as the thresholds. For maximum, a higher threshold means later exits, resulting in slower latency but higher accuracy.

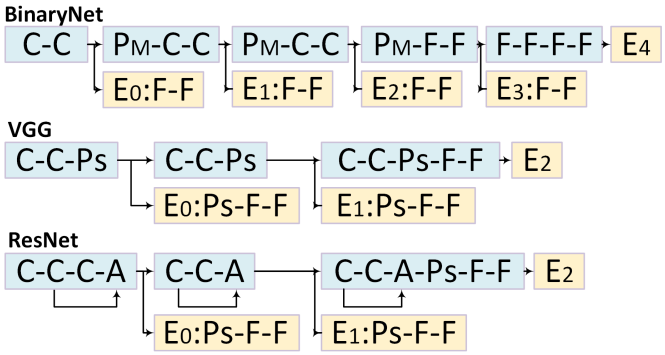


Fig. 4. **Network Architectures:** We utilized three different networks across two different frameworks. First, we adopted the CIFAR-10 BinaryNet from REDsec to have multiple exits. Then, for the Concrete library, we adapted a VGG-style architecture and built a Resnet-style architecture for our experiments. Activation functions (with REDsec employing the sign function and Concrete using hard tanh) and Batch Norm are implied for each Convolution and Fully Connected Layer. Exit layers are shown in yellow, where the backbone of the network skips over the exit layers.
C: Convolution **F:** Fully Connected P_M**:** MaxPool P_S**:** SumPool
A: ResNet Addition **E:** Exit

compared to the MENNs with similar accuracy. MENNs capitalize on the advantage of most images being classified with an early exit, achieving lower latency without sacrificing accuracy. Only images with high entropy outputs necessitate further processing, and these instances propagate through the network.

It is surprising these trends held up across different architectures and FHE frameworks. The greatest improvement came from BinaryNet using REDsec, showing about 7% accuracy improvement for the same latency with using MENNs. This architecture exited before the expensive MaxPool operation which plagued the single-exit networks. The Concrete framework had a more efficient SumPool implementation, allowing single exit networks to run faster. However, MENNs still achieve an impressive 1-3% accuracy improvement for the same latency.

Another advantage of MENNs is that the user can pick their accuracy-latency trade-off by adjusting the threshold. Thus, if they need results quicker, they can raise the entropy-based threshold (or lower the maximum-based threshold) to allow more images to exit early. Conversely, if they want a higher accuracy for their inference, they can lower the entropy-base threshold (or raise the maximum-based threshold) to encourage the image to propagate deeper into the network if the result is uncertain. These MENN architectures allow the user to decide this trade-off dynamically in this MLaaS system. A more in-depth discussion of these results is presented in Section VI.

V. TorMENNt Attack

A. Attack Overview

The main issue with any input-dependent multi-exit neural network scheme is that a user must make a runtime decision. At a high-level, any decision made on data that is visible to an adversary can leak information and therefore break semantic security. For MENNs in particular, the decision to terminate computation early based on preliminary results will reveal a correlation between the execution time and the underlying class of the input. Thus, in the TorMENNt attack, an adversary will be able to predict the classification of user's input with a probability better than a random guess.

To enable the TorMENNt attack, we observe that different classification results have different recall rates when evaluated at a constant recall threshold (Figure 5-left). For example, based on our BinaryNet CIFAR-10 results, plane pictures have lower entropy than other classes and can exit early from the network. In contrast, CIFAR-10 truck pictures are harder to classify and will most likely take the last exit of the MENN. Therefore, if an inference operation terminates early,

an attacker can predict that the input is more likely to be a plane than a truck.

In the privacy-preserving inference model the cloud itself trains the neural network and owns the corresponding weights; therefore the cloud also knows the recall rates for the model. In this case, the user communicates directly with the honest-but-curious cloud to stop the computation, so the cloud has the ability to execute this template attack.

We also remark that it is possible for a third party eavesdropper to extract the same information, although it requires some stronger assumptions. First, the eavesdropper must have knowledge of the model and recall rates for individual classes, which requires at minimum several queries to the target inference service. Second, we assume the eavesdropper has access to the users decision via a side-channel, such as via decreasing communication packets, power consumption, or memory access times on the cloud server. Luckily, there are known techniques and mitigation strategies for these types of side channels [42]–[47].

B. Attack Methodology

In the privacy preserving inference model, the cloud can process encrypted user data such that user data is never directly exposed to the cloud. The cloud does, however, directly know which exit the user terminated computation and can compare this information to its test set data. In this way, the cloud can implement the TorMENNt attack and extract user information.

To launch the attack, we use the entropy-based confidence score on CIFAR10 and the maximum-based confidence score for Tiny Imagenet. We use 70% of the test set to generate the recall rates

$$r_{e,c,test} = \frac{exited_{e,c,test}}{\sum_{e' \in E} exited_{e',c,test}}, \tag{6}$$

where $exited_{e,c,test}$ indicates the number of images belonging to class c that terminated at exit e in our test set. This will be the template to compare with user data for the TorMENNt attack. For visualization, we use cumulative recall rates

$$cumulative\ r_{e,c,test} = \sum_{e' \leq e} r_{e',c,test} \tag{7}$$

summarized in Figure 5 (left and middle) for each of our five networks. The distortion of the circles shows different recall rates for different classes. From these results it is evident that the recall rates are different for each class, thus our hypothesis about information leakage is confirmed.

To automate the interaction of a user in our analysis, we employ client-side scripts that send through the neural network a random subset of n images from a single class of an attack validation dataset. In this case, the cloud service is asked to terminate the MENN inference early when the entropy confidence score is below the $C_{entropy}$ or C_{max} thresholds. At the same time, a server-side attack script observes the early termination exit to apply the TorMENNt attack. In particular, the attack calculates the recall of user data $r_{e,user}$ using an equation similar to Eq. 6, by replacing test data with actual user data and also dropping class c (since it is not known).

We use a bayesian model to predict the classification, where

$$P(c|e) = \frac{P(c) \cdot P(e|c)}{\sum_{\forall c' \in C} P(c') \cdot P(e|c')}. \tag{8}$$

Here, $P(e|c)$ is approximated by the test set template $r_{e,c,test}$ from Equation 6, and $P(c)$ is a prior distribution of classes. For multiple predictions, these probabilities can be determined with the expected value

$$\mathbb{E}[\#\ of\ images\ from\ class\ c] = \sum_{images} P(c|e = E). \tag{9}$$

The expected value estimates the number of images in class that the user sends.

If an attacker wants to test several different distributions, Kullback-Liebler divergence can be used to test out the attacker's guessed distribution, performed as:

$$KL_{P(\mathcal{C})}(r_{user} \| r_{attack}) = \\ - \sum_{c:P(c)>0} \sum_{e \in E} r_{e,user} \ \cdot \ \log \frac{P(c|e, P(\mathcal{C}))}{r_{e,user}}. \tag{10}$$

Here, $r_{e,user}$ is measured user recall obtained by the eavesdropper, and $P(\mathcal{C})$ is the attacker's initial guess of the classification distribution; this should be used for the prior distribution in Eq. 8. The lower the KL divergence between the user output data and the attacker's predicted recall, the more likely the attacker's guess is accurate. The attacker can therefore determine the classification of private user information by finding the lowest KL divergence as Attacker Prediction $= argmin(KL_{c \in \mathcal{C}})$.

C. TorMENNt Attack Results

The results of this prediction method are summarized in Figure 5 and Table I. Here, we define the attacker success rate based on the number of times the attack *correctly* guesses the class divided by the total number of times the attacker guesses, as follows:

$$Attacker\ Success\ Rate = \frac{correct\ predictions}{total\ predictions}. \tag{11}$$

For a single image, the attacker can guess the correct class 23.5% for BinaryNet of the time, which is significantly higher than the 10% rate expected from random guessing; even the worst performing attack for a single image on VGG was 15.8%. This divergence also demonstrates that the semantic security of the system is compromised based on our threat model definition. Notably, with enough images, the success rate of the attacker's guess can reach 45% to 67%.

D. Attack Analysis

The average number of bits leaked from early exit decisions is limited to

$$avg\ leaked \leq \log_2(num\ exits) \tag{12}$$

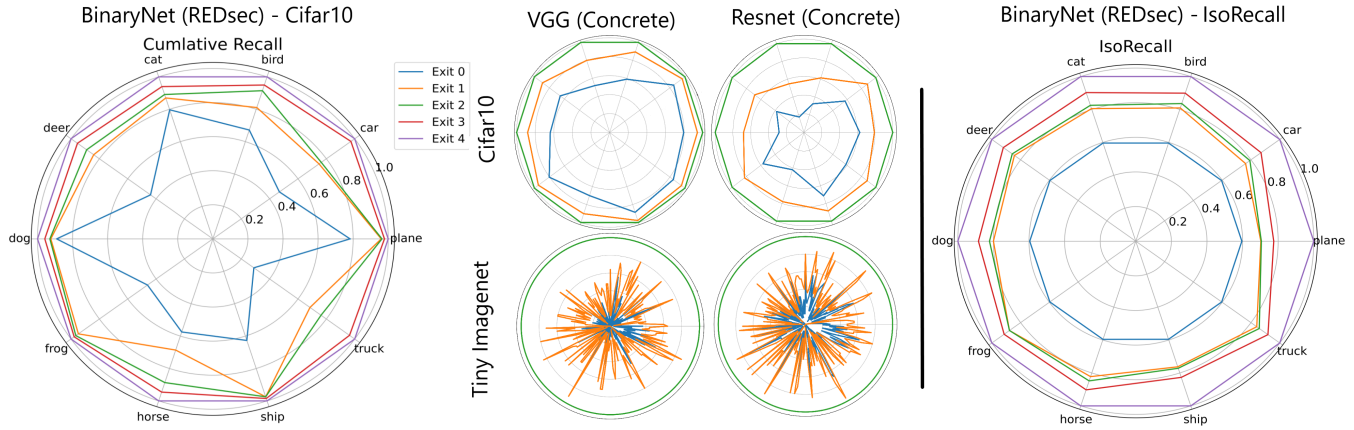
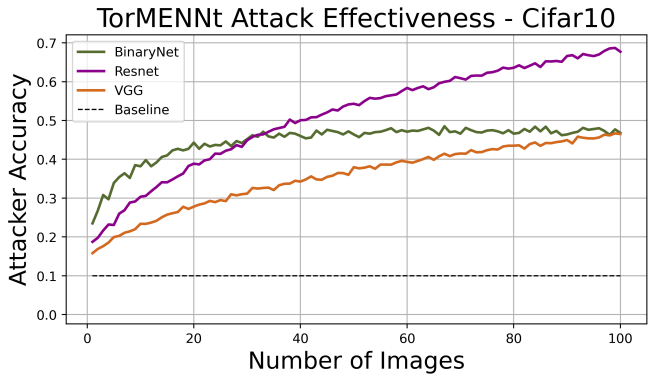
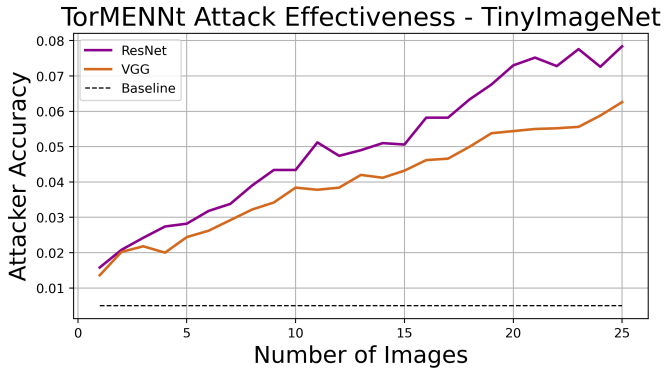


Fig. 5. **Recall across Classes:** These radar charts show the inconsistencies of exits taken among different classes. On the left, we show BinaryNet recalls for each of the classes. As shown by the irregularities of the circles, different classes tend to exit the network at different exit points, leaking information about a user's input. We utilize this information in the TorMENNt attack (Figure 6). Our Iso-recall mitigation sets the entropy levels to allow for fixed recall across classes, shown on the right. This, however, is difficult to scale to many classes and cannot fully guarantee security.



(a) Attacks on Cifar10



(b) Attacks on TinyImageNet

Fig. 6. **TorMENNt attack:** Due to the inconsistencies of recall across classes, it is possible to extract the predicted class based on which exit the user terminates. The attack success rate is the percentage of times the attacker correctly deduced the class of user input data, where the baseline is random guessing. We note that TinyImageNet had fewer images in the validation set, so the trend did not level off as in 6a. Even for a single image, the attacker had a much better than random chance of predicting the correct class based on the exit taken. The attacker gains more information as the user sends more images to the cloud.

bits for a single image input. Therefore, in our BinaryMENN inference example of a single image input, an attacker can only leak a maximum of 2.32 bits of information, which is not sufficient to distinguish between all ten classes of CIFAR-10, and far from sufficient from extracting out TinyImageNet top class. In the privacy-preserving MENN example, the user may risk a curious cloud attacker discovering they sent a picture of a car, but the exact make, model or license plate information are not likely to be leaked by the cloud. Therefore, compared to the privacy guarantees of plaintext-only MENNs, encrypted MENNs can still provide a higher assurance level during inference. Nevertheless, if the attacker can accumulate multiple MENN inputs, the amount of aggregate information grows linearly as

$$avg\ leaked \leq \sum^{num\ images} \log_2(num\ exits) \qquad (13)$$

bits. As a result, the TorMENNt attack becomes more impactful when the attacked users provide a lot of input data or a group of users employ the target service. This is evident by the differences in single and batch results in Table I and Figure 6.

E. Iso-Recall Mitigation

Since information about classification is leaked by comparing to test-set recall, one can eliminate the backdoor by ensuring the exits have *iso-recall across all exits*. This method would prevent an attacker from leaking information about the classification, as no unique information is provided per class.

While this approach mitigates the attack, it still suffers from several limitations. First, enforcing iso-recall still leaves side channels for any image parameters that are not subject to iso-recall. For example, an attacker may still be able to leak the brightness of an image since it is not directly controlled by classification recall. While our experimental results show that classification iso-recall did in fact help mitigate a brightness TorMENNt attack for CIFAR-10, we conjecture that this may not always be guaranteed. Moreover, this mitigation is limited as it assumes that the test set matches real-world applications. If this is not the case, or there are regional variations in user data, then different real-world recall rates can exist. Thus, an

attacker can still leak information about the classes with these differing real-world recall rates. For example, isorecall graph in Figure 5 (right) still shows some noisy distortions due to differences in the test and attack sets. This mitigation also requires information about a template set, which a user would not typically have access to.

Finally, the accuracy of the neural network degrades significantly, as shown in Table I. In our BinaryNet CIFAR-10 example, enforcing an iso-recall equal to the average optimal recall for individual classes leads to a 7% decrease in classification performance. For TinyImageNet, this mitigation is possible but decreased the accuracy lower than the single exit accuracy. Thus it does not scale well to large numbers of classes.

F. Avoiding Early Terminations

One simple semantically secure defense is for the users to ask the cloud service to evaluate the entire MENN, even if they are satisfied with the preliminary results. This solution does not reduce the computational cost of the MENN, but does ensure the user data is kept private. Specifically, even though the users are responsible for the evaluation cost of the entire model, they still get several benefits of using a multi-exit neural network. First, accuracy is improved when using the multiple exits as an ensemble [8]. Second, the users do get a preview of the neural network results early on. This feature is secure only if the users do not leak any information when they are locally satisfied with the results, for example, through a direct response (or even a side channel on the edge device).

VI. Discussion of our Findings

A. TorMENNt with Alternative Termination Schemes

Our attack model is primarily focused on the input-dependent MENN early termination scheme. Here, we discuss how the attack model applies to the early termination schemes introduced in Section II-A2. This discussion will help establish the necessary criterion for other MENN termination schemes and broadens the possibilities for secure MENN implementations.

The budgeted early termination scheme requires users to declare the exit upfront. Specifically, users can choose the exit considering their application's timing, monetary, and accuracy requirements. If this decision is made independent of the input data, then no information about the input data is leaked. This method can give users optimal performance based on their requirements without hurting security. The flexibility that FHE MENNs offer in this respect makes them a valuable feature for end-users.

Similarly, in anytime early termination, if the user's decision to terminate computation is independent of the input data and intermediate early-exit results, then the system remains semantically secure. Since this decision is not made upfront, more responsibility is put on the user to not make decisions based on output data. To remove this control from an unknowledgeable user, implementations of this scheme can enforce that the users can only ask for the result once. This inference scheme can be helpful for systems where the timing requirements to calculate the result vary, such as human-interactive technologies.

Finally, the distributed early termination scheme is the only scheme to offer semantically secure early termination of easy-to-process inputs. Since the data is processed locally, the decision to terminate early is not shared with the cloud, which conceals user inputs from potential eavesdroppers. However, the user must still ensure that this information is not leaked through an edge device side-channel or the absence of sending data. For example, if the edge device is known to generate one picture per minute, the absence of sending data may make the user's decision observable, which is significant depending on the context (e.g., plane images are never sent to the cloud).

B. Balancing FHE-PPML Security and Efficiency

The TorMENNt attack shows that information about sensitive MENN inputs could be leaked through early exit decisions in input-dependent inference. In order to improve security, much of the timing benefits discussed must be sacrificed. However, privacy-preserving MENNs are still a promising technology if their features and drawbacks are properly understood.

First, privacy-preserving MENNs can employ any of the other termination schemes discussed in Section VI-A. The underlying assumption of these schemes is that the early termination is not dependent on the user data, including the early exit preliminary results. Secure implementations of these early termination schemes can allow users more flexibility to choose the appropriate accuracy-timing tradeoff without sacrificing security.

A second strategy is to simply *accept the security risks* of the TorMENNt attack. For a single isolated image, an attacker can extract *on average* a few bits of information about the user based on Eq. 12. For an honest but curious cloud, this information is severely limited, especially for a single input. Also, the TorMENNt attack has strong assumptions that the template dataset has similar inputs compared to real world deployment. Our template and attack datasets did match our benchmarks, but often in real-world applications, this is not the case, which further protects the user.

In order to accept the risk of MENNs, it is likely best for users to assume a malicious cloud threat model. A malicious cloud can more directly leak attributes from users. Here, the malicious cloud can generate both a low entropy and high entropy result. They can select which value to send to the user based on the result of an encrypted test by using an encrypted multiplexer. Therefore, users need to clearly understand the security implications of their early exit decisions, and be OK with leaking one bit of information per exit about their input data. An alternative approach is to make a local decision based on early exit feedback but continue running the MENN in the cloud to guarantee security.

C. MENN implementations on non-FHE PPML

MENNs may also be suitable for other PPML solutions (not only FHE-based PPML). We provide a brief discussion of the

challenges we forsee with such potential implementations, as this remains an exciting research direction.

LHE-based schemes could be used to protect MENNs, but several drawbacks make them less ideal candidates than FHE-based schemes. First, LHE has limited multiplicative depth, making the usefulness of MENNs questionable. It is infeasible for the LHE-MENNs to require that the data be sent back to the user to decrypt and re-encrypt at every exit, since the amount of intermediate data that would need to be sent back to the user would be very large, creating a network bandwidth bottleneck. In the BinaryMENN network, the first exit of BinaryNet for CIFAR10 is of size $30 \cdot 30 \cdot 128$, which corresponds to 230kB of `int16` plaintext data. When this data is encrypted with a modern library, such as SEAL [26] with a 128-bit security level, it would yield 1.7TB bytes of ciphertext data to send back and forth to the user to process.

MPC-based schemes would also be good candidates for MENNs. MPC-based solutions are 2-3 orders of magnitude faster than FHE-based solutions [16], [33], but have other tradeoffs to consider. An important drawback of MPC-based solutions is that they require the user to also engage in non-linear pooling and activation functions. Such involvement requires the user to apply their local computational resources and network bandwidth, although the impact is less than the iterative LHE-based solution mentioned above. For example, the CIFAR-10 neural network evaluated with MPC in MiniONN [34] and Gazelle [33] incurs a communication overhead of 9.8GB and 1.2GB, respectively, which is still high, especially considering that this network is less than half the size of BinaryNet.

VII. Related Works

Many works look to steal model IP through various timing [42]–[45], electromagnetic [46] or power [47] side channels. In these works, an attacker sends inputs through the system and tries to recover the model weights. Unlike TorMENNt, these works do not look to protect user data during inference but are instead concerned with model IP (i.e., the network weights).

Several defenses against these attacks have been developed. MaskedNet [48] claims to be the first hardware inference engine that protects against model side channels. Its attacks are focused on binarized neural networks. The authors employ a masking technique where circuit-level inputs (e.g., an adder) are divided into two parallel circuits so that any information leaked is not useful to the adversary. Other generic side-channel obfuscations contribute to a large body of literature. The most relevant works include Zhang *et. al.* [49] that adds random timing mitigation to program outputs, Raccoon [50] that introduces decoy paths into the program flow, while others focus on secure execution environments for side-channel prevention [51]–[53], as well as cloud-based side-channel prevention [49], [54], [55].

TorMENNt is different from the aforementioned side-channel attacks since it only considers whether certain MENN layers are executed or not. For this reason, TorMENNt is more accessible compared to these side-channel attacks, as it does not require fine-grained power or electromagnetic traces. This also makes TorMENNt impervious to previously proposed obfuscation and masking defenses that aim to distort fine-grained details of the side channels but do not obfuscate or mask whether a computation occurs. TorMENNt also is applicable to the emerging PPML scenarios, which is not the case for the aforementioned attacks. Finally, these side-channel attacks are only intended for single exit networks and are geared towards stealing model weights/IP, not the user inputs.

One notable example is the work by Wei *et. al.* [47] that attempts to steal *inference data* via a power side-channel attack. Their FPGA-based setup uses oscilloscope data to predict model classification on the MNIST digit dataset [56]. Their threat model assumes that the attacker has access to a high-resolution oscilloscope or power monitoring Trojan horse. Overall, the authors report a high model prediction attack success rate of 89% for their MNIST dataset. However, TorMENNt has a different threat model that involves a less intrusive adversary using and targets MENNs. In particular, the attack by Wei *et. al.* requires fine-grained power traces and is subject to obfuscation and masking defenses, which is not the case with TorMENNt. Therefore, TorMENNt can support high noise tolerance in the side channel. Nevertheless, Wei *et. al.* report 89% attack success rate, whereas TorMENNt is limited by Eq. 12 and achieves about 22% attack success rate using 5 exits. Notably, PPML techniques would defend against the side channel of Wei *et. al.*, whereas the TorMENNt attack is applicable on both encrypted and plaintext inference.

Finally, Dong *et. al.* [20] proposed fingerprinting MENN timings to prevent IP theft. They found that MENNs with input-dependent exit termination schemes release information about the model IP (e.g., the network weights). Therefore, they could determine the difference between independent and stolen MENN models. Our TorMENNt attack leverages a similar principle (i.e., early exits leak information), but exploits this leakage to predict the user inputs *for a given model IP*. Conversely, Dong *et. al.* utilize this leaked information in an attempt to recover the model IP *given the user inputs*.

To the best of our knowledge, this is the first work to investigate MENNs for FHE, and the first to expose a high-level ML-based side-channel that is immune FHE privacy protections.

VIII. Conclusion

This work introduces the possibility of using multi-exit neural networks in the context of fully homomorphic encryption. We found that there are signifigant accuracy-latency benefits for using FHE-MENNs over single exit networks, achieving up to 7% accuracy benefits for the same latency compared to single exit networks.

We also explore the security of MENNs, developing the TorMENNt attack to leak information on a users input data. We demonstrate how this attack can leak classification information about a user's input, be remark that this attack can only leak one bit of information per exit per image. We discuss how this could be acceptable for many PPML applications

and users. We further provide suggestions on how to minimize the TorMENNt security vulnerability. Our Iso-recall mitigation can control for a limited number of inputs, but incurs an accuracy drop that increases the more classes or variables we try to control. Despite the TorMENNt vulnerability, we see FHE-MENNs as a promising speed-up alternative for PPML and recommend developers to consider this technology for their applications.

REFERENCES

[1] M. Ribeiro, K. Grolinger, and M. A. Capretz, "MLaaS: Machine learning as a service," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2015, pp. 896–902.

[2] E. Hesamifard, H. Takabi, and M. Ghasemi, "Cryptodl: Deep neural networks over encrypted data," *arXiv preprint arXiv:1711.05189*, 2017.

[3] S. Scardapane, M. Scarpiniti, E. Baccarelli, and A. Uncini, "Why should we add early exits to neural networks?" *Cognitive Computation*, vol. 12, no. 5, pp. 954–966, 2020.

[4] E. S. Marquez, J. S. Hare, and M. Niranjan, "Deep cascade learning," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 11, pp. 5475–5485, 2018.

[5] Y. Hu, A. Huber, J. Anumula, and S.-C. Liu, "Overcoming the vanishing gradient problem in plain recurrent networks," *arXiv preprint arXiv:1801.06105*, 2018.

[6] T.-K. Hu, T. Chen, H. Wang, and Z. Wang, "Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference," *arXiv preprint arXiv:2002.10025*, 2020.

[7] M. Phuong and C. H. Lampert, "Distillation-based training for multi-exit architectures," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1355–1364.

[8] M. Wołczyk, B. Wójcik, K. Bałazy, I. T. Podolak, J. Tabor, M. Śmieja, and T. Trzcinski, "Zero Time Waste: Recycling Predictions in Early Exit Neural Networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2516–2528, 2021.

[9] L. Qendro, A. Campbell, P. Lio, and C. Mascolo, "Early exit ensembles for uncertainty quantification," in *Machine Learning for Health*. PMLR, 2021, pp. 181–195.

[10] S. Laskaridis, A. Kouris, and N. D. Lane, "Adaptive Inference through Early-Exit Networks: Design, Challenges and Directions," in *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*, 2021, pp. 1–6.

[11] C. Lo, Y.-Y. Su, C.-Y. Lee, and S.-C. Chang, "A dynamic deep neural network design for efficient workload allocation in edge computing," in *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 2017, pp. 273–280.

[12] W. Ju, W. Bao, D. Yuan, L. Ge, and B. B. Zhou, "Learning early exit for deep neural network inference on mobile devices through multi-armed bandits," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 2021, pp. 11–20.

[13] S. Chen, Y. Wu, Z. Chen, T. Yoshioka, S. Liu, J. Li, and X. Yu, "Don't shoot butterfly with rifles: Multi-channel continuous speech separation with early exit transformer," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6139–6143.

[14] A. Kouris, S. I. Venieris, S. Laskaridis, and N. D. Lane, "Multi-exit semantic segmentation networks," *arXiv preprint arXiv:2106.03527*, 2021.

[15] Y. Li, Y. Gao, M. Shao, J. T. Tonecha, Y. Wu, J. Hu, and I. Lee, "Implementation of multi-exit neural-network inferences for an image-based sensing system with energy harvesting," *Journal of Low Power Electronics and Applications*, vol. 11, no. 3, p. 34, 2021.

[16] L. Folkerts, C. Gouert, and N. G. Tsoutsos, "REDsec: Running Encrypted Discretized Neural Networks in Seconds," *Cryptology ePrint Archive, Paper 2021/1100*, 2021.

[17] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2464–2469.

[18] X. Wang, Y. Luo, D. Crankshaw, A. Tumanov, F. Yu, and J. E. Gonzalez, "Idk cascades: Fast deep learning by learning not to overthink," *arXiv preprint arXiv:1706.00885*, 2017.

[19] C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, and S. Yang, "Adanet: Adaptive structural learning of artificial neural networks," in *International conference on machine learning*. PMLR, 2017, pp. 874–883.

[20] T. Dong, H. Qiu, T. Zhang, J. Li, H. Li, and J. Lu, "Fingerprinting multi-exit deep neural network models via inference time," *arXiv preprint arXiv:2110.03175*, 2021.

[21] H. Inoue, "Adaptive ensemble prediction for deep neural networks based on confidence level," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1284–1293.

[22] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.

[23] S. Halevi and V. Shoup, "Bootstrapping for HElib," in *Annual International conference on the theory and applications of cryptographic techniques*. Springer, 2015, pp. 641–670.

[24] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.

[25] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 409–437.

[26] H. Chen, K. Laine, and R. Player, "Simple encrypted arithmetic library-seal v2. 1," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 3–18.

[27] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption." *IACR Cryptol. ePrint Arch.*, vol. 2012, p. 144, 2012.

[28] I. Chillotti, M. Joye, and P. Paillier, "Programmable bootstrapping enables efficient homomorphic inference of deep neural networks," Cryptology ePrint Archive, Report 2021/091, 2021., Tech. Rep., 2020.

[29] Zama, "Zama concrete v0.1.0," https://github.com/zama-ai/concrete, 2021, zama AI, France.

[30] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachene, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," in *international conference on the theory and application of cryptology and information security*. Springer, 2016, pp. 3–33.

[31] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 1986, pp. 162–167.

[32] B. Reagen, W. Choi, Y. Ko, V. Lee, G.-Y. Wei, H.-H. S. Lee, and D. Brooks, "Cheetah: Optimizations and methods for privacy preserving inference via homomorphic encryption," *arXiv preprint arXiv:2006.00505*, 2020.

[33] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1651–1669.

[34] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 619–631.

[35] A. Sanyal, M. Kusner, A. Gascon, and V. Kanade, "TAPAS: Tricks to accelerate (encrypted) prediction as a service," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4490–4499.

[36] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *Annual International Cryptology Conference*. Springer, 2018, pp. 483–512.

[37] A. Krizhevsky, "CIFAR10 Dataset," https://www.tensorflow.org/datasets/catalog/cifar10, 2009.

[38] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *Advances in neural information processing systems*, vol. 28, 2015.

[39] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[42] V. Duddu, D. Samanta, D. V. Rao, and V. E. Balas, "Stealing neural networks via timing side channels," *arXiv preprint arXiv:1812.11720*, 2018.

[43] S. Hong, M. Davinroy, Y. Kaya, S. N. Locke, I. Rackow, K. Kulda, D. Dachman-Soled, and T. Dumitraş, "Security analysis of deep neural networks operating in the presence of cache side-channel attacks," *arXiv preprint arXiv:1810.03487*, 2018.

[44] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[45] M. Isakov, L. Bu, H. Cheng, and M. A. Kinsy, "Preventing neural network model exfiltration in machine learning hardware accelerators," in *2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, 2018, pp. 62–67.

[46] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI-NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 515–532.

[47] L. Wei, B. Luo, Y. Li, Y. Liu, and Q. Xu, "I know what you see: Power side-channel attack on convolutional neural network accelerators," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 393–406.

[48] A. Dubey, R. Cammarota, and A. Aysu, "Maskednet: The first hardware inference engine aiming power side-channel protection," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2020, pp. 197–208.

[49] D. Zhang, A. Askarov, and A. C. Myers, "Predictive mitigation of timing channels in interactive systems," in *Proceedings of the 18th ACM conference on Computer and communications security*, 2011, pp. 563–574.

[50] A. Rane, C. Lin, and M. Tiwari, "Raccoon: Closing Digital Side-Channels through Obfuscated Execution," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 431–446.

[51] C. Liu, A. Harris, M. Maas, M. Hicks, M. Tiwari, and E. Shi, "Ghostrider: A hardware-software system for memory trace oblivious computation," *ACM SIGPLAN Notices*, vol. 50, no. 4, pp. 87–101, 2015.

[52] A. Ahmad, B. Joe, Y. Xiao, Y. Zhang, I. Shin, and B. Lee, "Obfuscuro: A commodity obfuscation engine on Intel SGX," in *Network and Distributed System Security Symposium*, 2019.

[53] M. Maas, E. Love, E. Stefanov, M. Tiwari, E. Shi, K. Asanovic, J. Kubiatowicz, and D. Song, "Phantom: Practical oblivious computation in a secure processor," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 311–324.

[54] T. Kim, M. Peinado, and G. Mainar-Ruiz, "Stealth-mem: System-level protection against cache-based side channel attacks in the cloud," in *21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 189–204.

[55] Y. Zhang, A. Juels, A. Oprea, and M. K. Reiter, "Homealone: Co-residency detection in the cloud via side-channel analysis," in *2011 IEEE symposium on security and privacy*. IEEE, 2011, pp. 313–328.

[56] Y. Lecun, L. Bottou, Y. Bengie, and P. Haffner, "MNIST Dataset," https://www.tensorflow.org/datasets/catalog/mnist, 1994.