



# Auditing Membership Leakages of Multi-Exit Networks

Zheng Li  
CISPA Helmholtz Center for  
Information Security

Yiyong Liu  
CISPA Helmholtz Center for  
Information Security

Xinlei He  
CISPA Helmholtz Center for  
Information Security

Ning Yu  
Salesforce Research

Michael Backes  
CISPA Helmholtz Center for  
Information Security

Yang Zhang  
CISPA Helmholtz Center for  
Information Security

## ABSTRACT

Relying on the truth that not all inputs require the same level of computational cost to produce reliable predictions, multi-exit networks are gaining attention as a prominent approach for pushing the limits of efficient deployment. Multi-exit networks endow a backbone model with early exits, allowing predictions at intermediate layers of the model and thus saving computation time and energy. However, various current designs of multi-exit networks are only considered to achieve the best trade-off between resource usage efficiency and prediction accuracy, the privacy risks stemming from them have never been explored. This prompts the need for a comprehensive investigation of privacy risks in multi-exit networks.

In this paper, we perform the first privacy analysis of multi-exit networks through the lens of membership leakages. In particular, we first leverage the existing attack methodologies to quantify the multi-exit networks' vulnerability to membership leakages. Our experimental results show that multi-exit networks are less vulnerable to membership leakages, and the exit (number and depth) attached to the backbone model is highly correlated with the attack performance. Furthermore, we propose a hybrid attack that exploits the exit information to improve the performance of existing attacks. We evaluate membership leakage threat caused by our hybrid attack under three different adversarial setups, ultimately arriving at a model-free and data-free adversary. These results clearly demonstrate that our hybrid attacks are very broadly applicable, thereby the corresponding risks are much more severe than shown by existing membership inference attacks. We further present a defense mechanism called *TimeGuard* specifically for multi-exit networks and show that *TimeGuard* mitigates the newly proposed attacks perfectly.<sup>1</sup>

## CCS CONCEPTS

• Security and privacy; • Computing methodologies → Machine learning;

<sup>1</sup>Our code is available at <https://github.com/zhenglisec/Multi-Exit-Privacy>.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

CCS '22, November 7–11, 2022, Los Angeles, CA, USA

© 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-9450-5/22/11...\$15.00

<https://doi.org/10.1145/3548606.3559359>

## KEYWORDS

machine learning; multi-exit networks; membership leakages

### ACM Reference Format:

Zheng Li, Yiyong Liu, Xinlei He, Ning Yu, Michael Backes, Yang Zhang. 2022. Auditing Membership Leakages of Multi-Exit Networks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*, November 7–11, 2022, Los Angeles, CA, USA. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3548606.3559359>

## 1 INTRODUCTION

Machine Learning (ML) has established itself as a cornerstone for a wide range of critical applications, such as image classification and face recognition. To achieve better performance, large ML models with increasing complexity are proposed. The improvement in performance stems from the fact that the deeper ML model fixes the errors of the shallower one. This means some samples that are already correctly classified or recognized by the shallow ML model do not require additional complexity. However, such progression to deeper ML models has dramatically increased the latency and energy required for feedforward inference, which further contradicts the research activity in Green AI [1, 48] that aims to decrease AI's environmental footprint and increase its inclusivity. This reality has motivated research on input-adaptive mechanisms, i.e., multi-exit networks, which is an emerging direction for fast inference and energy-efficient computing.

The multi-exit model consists of a backbone model (i.e., a large vanilla model) and multiple exits (i.e., lightweight classifiers) attached to the backbone model at different depths. The backbone model is used for feature extraction and the lightweight classifiers allow data samples to be predicted and to exit at an early layer of the model based on tunable early-exit criteria. Multi-exit architecture can be applied to many critical applications as it can effectively reduce computational costs. For example, exiting early means low latency, which is crucial for operating under real-time constraints in robotics applications, such as self-driving cars. Furthermore, early exiting can improve energy efficiency, which directly influences battery life and heat release, especially on mobile devices.

### 1.1 Our Contributions

**Motivation.** Multi-exit networks, despite their low latency and high energy efficiency, also rely on large-scale data to train themselves, as the way vanilla ML models are trained. In many cases, the data contains sensitive and private information about individuals, such as shopping preferences, social relationships, and health status. Various recent studies have shown that vanilla ML

models, represented by image classifiers, are vulnerable to privacy attacks [33, 35, 37, 42, 46, 49, 51]. One major attack in this domain is membership inference: An adversary aims to infer whether a data sample is part of a target ML model's training dataset.

However, various current designs of multi-exit networks are only considered to achieve the best trade-off between resource usage efficiency and prediction accuracy, the privacy risks stemming from them have never been explored. This prompts the need for a comprehensive investigation of privacy risks in multi-exit networks, such as the vulnerability of multi-exit networks to privacy attacks, the reasons inherent in this vulnerability, the factors that affect attack performance, and whether or how these factors can be exploited to improve or reduce attack performance.

In this paper, we take the first step to audit the privacy risks of multi-exit networks through the lens of membership inference. More specifically, we focus on machine learning classification, which is the most common machine learning task, and conduct experiments with 3 types of membership inference attacks, 6 benchmark datasets, and 8 model architectures.

**Main Findings.** We first leverage the existing attack methodologies (gradient-based, score-based, and label-only) to audit the multi-exit networks' vulnerability through membership inference attacks. We conduct extensive experiments, and the empirical results demonstrate that multi-exit models are less vulnerable to membership inference attacks than vanilla ML models. For instance, considering the score-based attacks, we achieve an attack success rate of 0.5413 on the multi-exit model trained on CIFAR-10 with the backbone model being ResNet-56, while the result is 0.7122 on the corresponding vanilla ResNet-56. Furthermore, we delve more deeply into the reasons for the lower vulnerability and reveal that the reason behind this is that the multi-exit models are less likely to be overfitted.

We also find that the number of exits is negatively correlated with the attack performance, i.e., multi-exit models with more exits are less vulnerable to membership inference. Besides, a more interesting observation is that considering a certain multi-exit model, exit depth is positively correlated with attack performance, i.e., exits attached to the backbone model at deeper locations are more vulnerable to membership inference. These observations are due to the fact that different depths of exits in the backbone model actually imply different capacity models, and that deeper exits imply higher capacity models, which are more likely to be overfitted by memorizing properties of the training set.

**Hybrid Attack.** The above findings render us a new factor to improve the attack performance. More concretely, we propose a novel hybrid attack against multi-exit networks that exploit the exit information as new knowledge of the adversary. The hybrid attack's methodology can be divided into two stages, each of which actually represents one type of attack against ML models:

- **Hyperparameter stealing:** the adversary's goal is to steal the hyperparameters, i.e., the number of exits and the exit depth of a given multi-exit network designed by the model owner.
- **Enhanced membership inference:** the adversary then exploits the stolen exit information as new knowledge to launch more powerful member inference attacks.

In particular, we study three different adversaries for obtaining exit information by starting with some strong assumptions, and gradually relaxing these assumptions in order to show that far more broadly applicable attack scenarios are possible. Our investigation shows that indeed, our proposed hybrid attack can achieve better attack performance by exploiting extra exit information, compared to original membership inference attacks.

**Adversary 1.** For the first adversary, we assume they have direct access to the exit information, i.e., exit depth, as well as train a shadow model of the same architecture (especially the exit placements) as the target model. Further, the adversary trains the shadow models on a shadow dataset that comes from the same distribution as the target dataset. The assumption of same-architecture and same-distribution also holds for almost all existing membership inference attacks [33, 35, 37, 42, 46, 49, 51].

We start by querying the target model using a large number of data samples to determine the number of exits attached to the model. Then we propose different methods (e.g., one-hot encoding) based on the attack models adopted by existing attacks to exploit this exit information. Extensive experimental evaluation shows that extra exit information indeed leaks more membership information about training data. For example, our hybrid attack achieves an attack success rate of 0.7681 on a multi-exit WideResNet-32 trained on CIFAR-100, while the result of the original attack is 0.6799.

**Adversary 2.** For this adversary, we relax the assumption that they have direct access to exit information and keep the assumption of the same architecture and same distribution unchanged. This is a more challenging scenario compared to the previous one.

In this scenario, we propose *time-based hyperparameter stealing* to obtain the exit information. Concretely, we feed a set of samples to the target multi-exit model and record the inference time of these samples. We then propose a simple but effective unsupervised method to cluster the samples based on different inference times. Thus, the number of clusters implies the number of exits, and the index of clusters implies the exit depth.

The intuition is that the goal of multi-exit models is to reduce computational costs by allowing data samples to be predicted and to exit at an early point. Therefore, the inference time for data samples inevitably varies with the depth of the exit, i.e., data samples leaving deeper exit points imply longer inference times. Thus, we can determine the exit depths by observing the magnitude of inference time. Experimental results show that our hybrid attack achieves a strong performance as our *time-based hyperparameter stealing* can achieve almost 100% prediction accuracy of exit depths.

**Adversary 3.** This adversary works without any knowledge about the target models and target datasets, that is, the adversary can only construct a shadow model that is different from the target model or a dataset from a different distribution from the target dataset. Meanwhile, the different architectures between the shadow model and the target model will inevitably lead to different exit placements between them. Encouragingly, our hybrid attack still has better attack performance than the original attacks, suggesting that the extra exit information has a broader range of applicable attack scenarios.

Finally, we propose a simple but effective defense mechanism called *TimeGuard*, which postpones giving the prediction, rather

than giving them immediately. Our in-depth analysis shows that *TimeGuard* can reduce attack performance to a lower bound and maintain high efficiency, i.e., achieve the best trade-off between privacy and efficiency.

Abstractly, our contributions can be summarized as follows:

- We take the first step to audit the privacy risks of multi-exit networks through the lens of membership inference attacks.
- Our empirical evaluation shows that the multi-exit networks are less vulnerable to member inference, and the exit information is highly correlated with the attack performance.
- We propose a hybrid attack that exploits the exit information to improve the attack performance of membership inference.
- We evaluate membership leakage threat caused by hybrid attack under three different adversarial setups, ultimately arriving at a model-free and data-free adversary, which further enlarges the scope of the hybrid attack.
- We propose *TimeGuard* to mitigate privacy risks stemming from our attack and empirically evaluate its effectiveness.

## 2 PRELIMINARIES

### 2.1 Membership Leakages in Machine Learning Models

Membership leakages in ML models emerge when an adversary aims to identify whether a data sample was used to train a certain model or not. It can raise severe privacy risks as the membership can reveal an individual's private information. For example, identifying an individual's participation in a hospital's health analytic training set reveals that this individual was once a patient in that hospital.

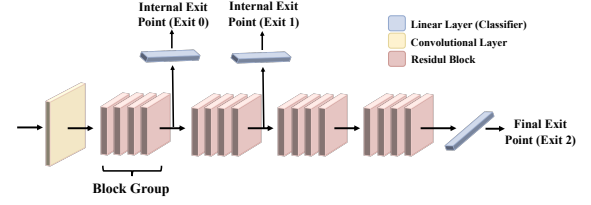
To evaluate the vulnerability of a given ML model to membership leakages, membership inference attacks are used as an auditing tool to quantify the private information that a model leaks about the individual data samples in its training set. In this work, we focus on auditing the membership leakages of multi-exit networks.

### 2.2 Multi-Exit Networks

Multi-exit networks save computation by making input-specific decisions about bypassing the remaining layers, once the model becomes confident. More concretely, a multi-exit network applies multiple lightweight classifiers on a vanilla ML model to allow the inference to preemptively finish at one of the exit points when the network is sufficiently confident with a predefined stopping criterion. See Figure 1 for an illustration of the design of multi-exit networks. In this paper, we focus on the pioneer and, meanwhile the most representative design of multi-exit networks.

**Backbone Initialisation.** As aforementioned, multi-exit networks modify the vanilla ML model by adding multiple lightweight classifiers at certain placements throughout the network. Here, vanilla ML models are also referred to as backbone models. A backbone model can be any regular machine learning model architecture. For example, in this work, we focus on the image classification task, so such backbone models can be convolutional networks applicable to vision, such as VGG [50], ResNet [18], and MobileNet [47].

**Exit Placement.** For simplicity, exit placements are restricted to be at the output of individual network blocks, following an approximately equidistant workload distribution.



**Figure 1: An illustration of multi-exit network with 3 exits inserted, including 2 internal exit points and 1 final exit point.**

**Multi-Exit Network Training.** Given a training dataset, a multi-exit model is optimized by minimizing the loss function of all training samples and exit points. The training process includes two phases: feedforward pass and backward pass. In the former, data samples are passed through the model, including the final exit point and internal exit points, and the network outputs from all exit points are recorded, and then the loss of the network is calculated. In the latter, the weights of the model are updated using gradient descent by the loss backpropagation. In addition to the joint training described above, Kaya et al. [29] also perform exit-only training, i.e., optimizing only the attached lightweight classifiers. In this work, we focus only on joint training, as it allows for better prediction performance.

**Early-Exit Criteria.** Given a data sample, it will leave at one of the exit points when the network is sufficiently confident with a predefined stopping criterion. To quantify confidence, we use the estimated probability that the sample belongs to the top-1 class. If this probability surpasses a threshold  $\tau$ , we consider the prediction to be credible. This threshold helps to instantly adjust early exits based on resource availability and performance requirements. Following most previous works [21, 26, 29, 43, 44, 54], the principle of threshold selection is to guarantee the same or similar classification performance as vanilla models while gaining a lower computational cost. See our technique report [36] for how to select a threshold.

**Application Scenarios.** As introduced in Section 1, with the benefits of low latency and energy efficiency, multi-exit architectures can be used in many real-world applications. In the industry, IT companies leverage multi-exit networks to accelerate forward inference. For instance, Intel [2] has developed multi-exit architectures to reduce the computational cost of DNNs. Microsoft [61] and Huawei [23] conducted research on reducing the computational complexity of BERT for bringing language models to IoT devices. Besides, there is a growing body of applications deploying multi-exit architectures to IoT scenarios and real-time systems [24, 27, 28, 34, 60]. In academia, there has been substantial research focusing on minimizing the computational and energy requirements of multi-exit networks for efficient inference [13, 21, 22, 25, 26, 29, 30, 32, 38, 43, 44, 54–57]. In short, multi-exit networks are gaining significant attention and rapid development in both industry and academia.

### 3 QUANTIFYING MEMBERSHIP LEAKAGE RISKS

In this section, we quantify the privacy risks of multi-exit networks through the lens of membership inference attacks. We start by defining the threat model. Then, we describe the attack methodology. Finally, we present the evaluation results. Note that our goal here is not to propose a novel membership inference attack, instead, we aim to quantify the membership leakages of multi-exit networks. Therefore, we follow the existing attacks and their threat models.

#### 3.1 Threat Model

Here, we outline the threat models considered in this paper. There are three existing categories of scenarios, i.e., white-box scenario, black-box scenario, and label-only scenario.

Given a target model, we assume the adversary has an auxiliary dataset (namely shadow dataset) that comes from the same distribution as the target model's training set. The shadow dataset is used to train a shadow model, the goal of which is to mimic the behavior of the target model to perform the attack. Furthermore, we assume the shadow model has the same architecture as the target model following previous works [33, 35, 37, 42, 46, 49, 51]. In particular, the exit placements of the shadow multi-exit model are also the same as that of the target multi-exit model.

#### 3.2 Attack Methodology

We leverage existing membership inference attacks, which are designed for vanilla ML models, to multi-exit models.

**Gradient-based Attacks.** In gradient-based attacks [33, 42], the adversary obtains all adversarial knowledge and has full access to the target model. Given a shadow dataset  $\mathcal{D}_{shadow}$ , the adversary first splits it into two disjoint sets, i.e., shadow training set  $\mathcal{D}_{shadow}^{train}$  and shadow testing set  $\mathcal{D}_{shadow}^{test}$ . Then the adversary queries the shadow model  $\mathcal{S}$  on each data sample  $x$  from  $\mathcal{D}_{shadow}^{train}$ , and computes the prediction score, the feature of the second to the last layer, the loss in a forward pass, and the gradient of the loss with respect to the last layer's parameters in the backward pass. These computations, in addition to the one-hot encoding of the true label, are concatenated into a flat vector and labeled as a member if  $x$  is in the shadow training set  $\mathcal{D}_{shadow}^{train}$ , otherwise labeled as a non-member. In this way, the adversary can derive all data samples of  $\mathcal{D}_{shadow}$  as an attack training data set. With the attack training dataset, the adversary then trains the attack model, which is a binary classifier. Once the attack model is trained, the adversary can perform the attack to query the target model  $\mathcal{T}$  to differentiate members and nonmembers of the target dataset  $\mathcal{D}_{target}$ .

**Score-based Attacks.** Score-based attacks [35, 46, 49, 51] need to train the shadow model as well. Unlike gradient-based attacks, score-based attacks do not require intermediate features or gradients of the target model, but only access to the output scores of the model. The adversary also derives the attack training dataset by querying the shadow model using the shadow training dataset (labeled as members) and the shadow test dataset (labeled as non-members). The adversary can then use the attack training set to construct an attack model.

**Label-only Attacks.** Label-only attacks [11, 37] consider a more restricted scenario where the target model only exposes the predicted label instead of intermediate features or gradients, or even output scores. Thus, label-only attacks solely rely on the target model's predicated label as their attack model's input. Similar to previous attacks, this attack requires the adversary to train a shadow model. The adversary queries the target model on a data sample and perturbs it to change the model's predicted labels. Then, the adversary measures the magnitude of the perturbation and considers the data samples as members if their magnitude is greater than a predefined threshold, which can be derived by perturbing the shadow dataset on the shadow model.

#### 3.3 Experimental Settings

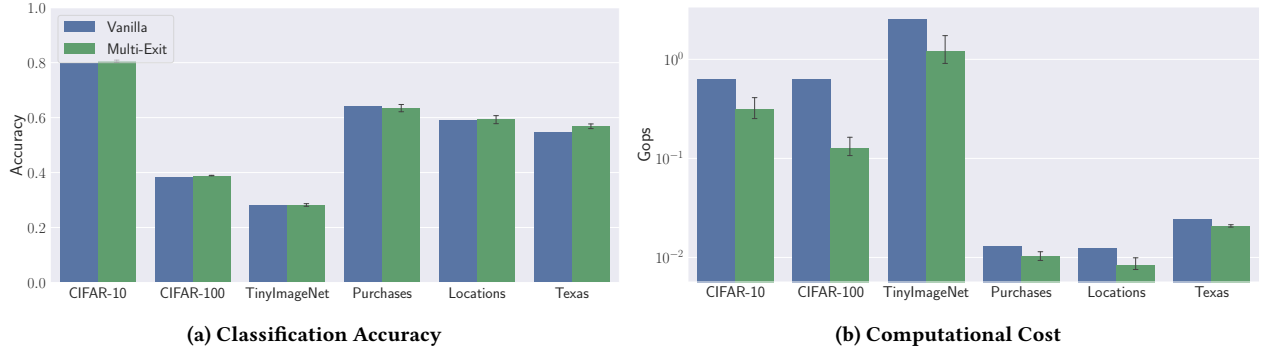
**Datasets.** We consider six benchmark datasets of different tasks, sizes and complexity to conduct our experiments. Concretely, we adopt three computer vision tasks, namely CIFAR-10 [3], CIFAR-100 [3], TinyImageNet [4], and three non-computer vision tasks, namely Purchases [5], Locations [6] and Texas [7]. In particular, the latter three datasets are privacy-sensitive: Purchases dataset relates to shopping preferences, Locations dataset relates to social connections, and Texas dataset relates to health status. Details of all six datasets can be found in our technique report [36].

**Datasets Configuration.** For a given dataset  $\mathcal{D}$ , we randomly split it into four disjoint equal parts:  $\mathcal{D}_{target}^{train}$ ,  $\mathcal{D}_{target}^{test}$ ,  $\mathcal{D}_{shadow}^{train}$ , and  $\mathcal{D}_{shadow}^{test}$ . We use  $\mathcal{D}_{target}^{train}$  to train the target model  $\mathcal{T}$  and treat it as the members of the target model. We treat  $\mathcal{D}_{target}^{test}$  as the non-members of the target model. Similarly, we use  $\mathcal{D}_{shadow}^{train}$  to train the shadow model  $\mathcal{S}$  and treat it as the members of the shadow model. We again treat  $\mathcal{D}_{shadow}^{test}$  as the non-members of the shadow model. We feed all  $\mathcal{D}_{shadow}^{train}$  and  $\mathcal{D}_{shadow}^{test}$  to the shadow model to create an attack training dataset to train the attack models.

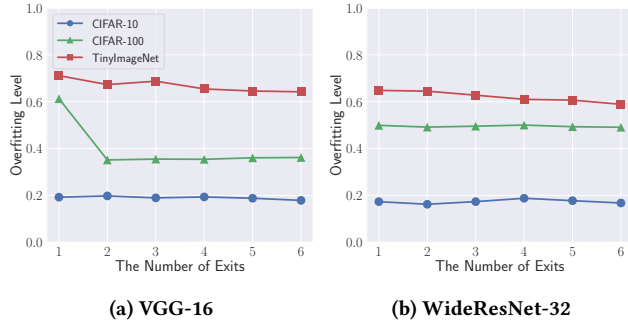
**Attack Model.** Here we establish three types of attack models and each type for one attack.

- **Gradient-based.** This attack has five inputs for the attack model, like the one used by Nasr et al. [42], including the target sample's prediction score, the feature of the second to the last layer, classification loss, gradients of the last layer's parameters, and one-hot encoding of its true label. Each input is fed into a different MLP (2 or 3 layers) and the resulting embeddings are concatenated together as one vector to a 4-layer MLP.
- **Score-based.** The score-based attack utilizes the predicted score as input to the attack model, which is constructed as a 4-layer MLP with one input component.
- **Label-only.** Here, the attack model is not a specific MLP, but a decision function that measures the magnitude of the perturbation and considers data samples as members if their magnitude is larger than a predefined threshold, which can be derived by perturbing the shadow dataset on shadow model.

**Target Model (Multi-Exit Model).** For computer vision tasks, we adopt four popular architectures as the backbone to construct



**Figure 2: The performance of original classification tasks and computational costs for both vanilla models and multi-exit models. computer vision tasks are on VGG-16, and non-computer vision tasks are on FCN-18-1.**



**Figure 3: Comparison of overfitting levels between vanilla and multi-exit model. Note that 1 exit represents the vanilla model and 2-6 exits represent different multi-exit models.**

multi-exit models, including VGG-16 [50], ResNet-56 [18], MobileNetV2 [47], and WideResNet-32 [59]. For non-computer vision tasks, we designed four 18-layer fully connected networks (FCN-18) with different numbers of hidden neural units (1024, 2048, 3072, 4096), named FCN-18-1/2/3/4 throughout the paper. See our technique report [36] for more details about these architectures. For the exit placement, we follow the principle of Kaya et al. [29] by attaching an additional lightweight classifier (2- or 3-layer MLP) as an exit, i.e., exit placements are restricted to be at the output of individual network blocks, following an approximately equidistant workload distribution. In particular, for each backbone model, we construct 5 different target models with the number of exits varying from 2 to 6. Note that here we consider the backbone model’s own classifier as the final exit point and count it in the total number of exits. For early-exit threshold  $\tau$  ( $0 \leq \tau \leq 1$ ), we manually search for suitable  $\tau$  value (among 0 to 1 in 0.05 steps) that achieve the same or similar classification performance as vanilla (backbone) models while gaining lower computational cost. To evaluate computational cost, we calculate the number of mathematical operations (denoted as *ops*) in the feedforward pass process by averaging over 10,000 images. The early-exit thresholds we set for multi-exit models can be found in our technique report [36].

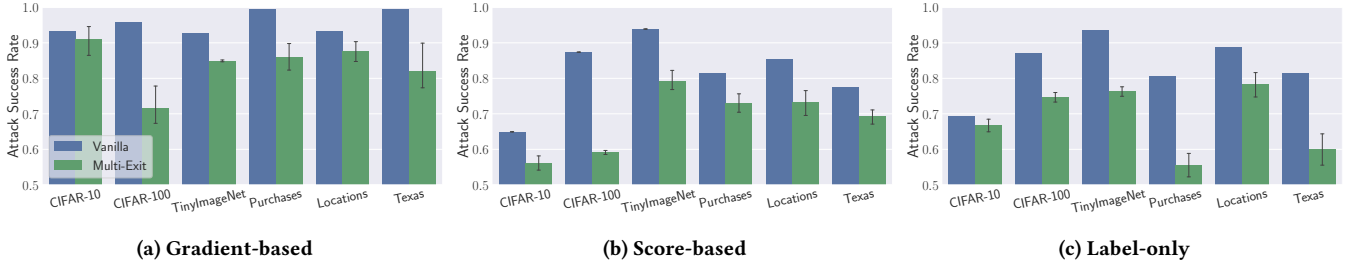
**Baseline (Vanilla Model).** To fully understand the membership leakages of multi-exit models, we further use the vanilla model as the baseline model. We train eight models from scratch for all datasets, including both computer vision and non-computer vision models. In all cases, including vanilla and multi-exit models, we adopt cross-entropy as the loss function and Adam as the optimizer, and train them for 100 epochs. Our code is implemented in Python 3.8 and PyTorch 1.8.1 and runs on an NVIDIA HGX-A100 server with Ubuntu 18.04.

**Metric.** Following previous work, we adopt the accuracy, i.e., attack success rate (denoted as ASR), as the attack model’s training and testing datasets are both balanced with respect to membership distribution. Note that we average the performance of different multi-exit models with the number of exits varying from 2 to 6 and report the mean and standard deviation.

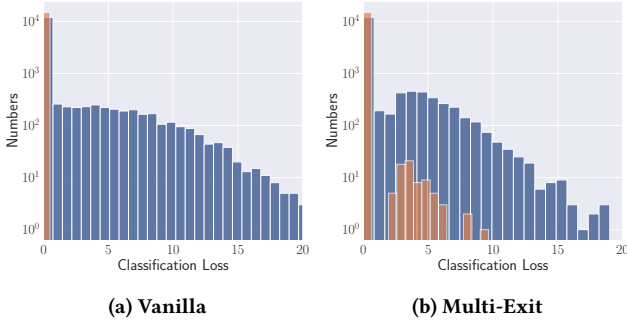
### 3.4 Results

**Classification Accuracy and Computational Cost.** We first show the performance of vanilla and multi-exit models on their original classification tasks and computational costs in Figure 2. See more results in our technique report [36]. We observe that the multi-exit model performs at least on par with the vanilla model on the classification task, but is much better in terms of computational cost. For instance, the multi-exit VGG-16 trained on CIFAR-10 achieves 80.558% accuracy, which is better than 80.04% accuracy of vanilla VGG-16. As for the computational cost, the multi-exit VGG-16 achieves 0.3125 Gops while the vanilla model achieves 0.6283 Gops.

**Attack ASR Score.** Regarding membership inference against vanilla and multi-exit models, we report ASR score on various datasets and model architectures in Figure 4. See more results in our technique report [36]. We can observe that all the multi-exit models have lower ASR than the vanilla models. For example, score-based ASR on vanilla VGG-16 trained on CIFAR-100 is 0.8738, while the mean ASR on multi-exit VGG-16 is only 0.5914. Label-only ASR on vanilla FCN-18-1 trained on Locations is 0.8866, while the mean ASR on multi-exit VGG-16 is 0.7831. However, these results may lead to premature claims of privacy. Section 4 presents that the membership leakage risks stemming from our hybrid attack are much more severe than shown by existing attacks.



**Figure 4: The attack performance of original membership inference attacks against vanilla and multi-exit models. Computer vision tasks are on VGG-16, and non-computer vision tasks are on FCN-18-1.**

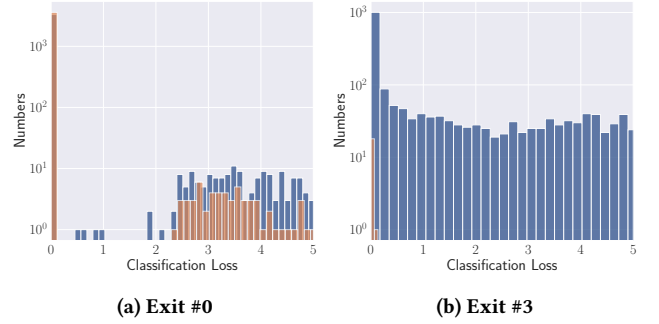


**Figure 5: The distribution of loss with respect to original classification tasks for member and non-member samples between the vanilla VGG-16 and the 4-Exit VGG-16 on CIFAR-10.**

**Overfitting Level.** Here, we delve more deeply into the reasons for the less vulnerability of multi-exit models. As almost all previous works [20, 35, 37, 41, 46, 49, 51, 58] claim that the overfitting level is the main factor contributing to the vulnerability of the model to membership inference, i.e., a lower overfitting level leads to less vulnerability to membership inference. Here, we also relate this to the different overfitting levels of ML models. The overfitting level of a given model is measured by calculating the difference between its training accuracy and testing accuracy, i.e., subtracting testing accuracy from training accuracy, which is adopted by previous works. In Figure 3, however, we see that the overfitting level of multi-exit models remains almost the same compared to the vanilla model, especially in VGG-16 and WideResNet-32 trained CIFAR-10 dataset.

This observation which is contradictory to the previous conclusion inspires us to rethink the relationship between overfitting levels and vulnerability to membership inference. More precisely, we argue that the current calculation, i.e., subtracting test accuracy from training accuracy, is not the best way to characterize overfitting level, which leads to no strong correlation between overfitting level and vulnerability to membership inference, at least for multi-exit models.

**Loss Distribution.** To find a more appropriate way to characterize the overfitting level and also to further investigate why the multi-exit model is less vulnerable to membership inference, we analyze

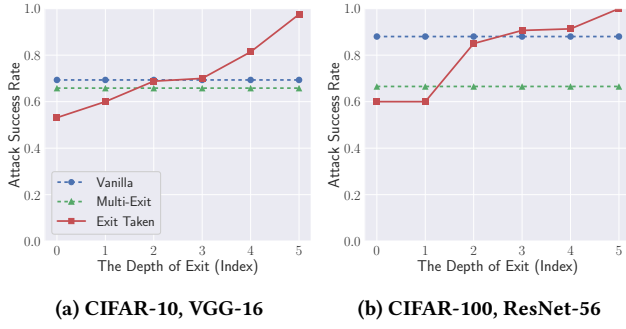


**Figure 6: The distribution of loss with respect to original classification tasks for member and non-member samples of 4-Exit VGG-16 on CIFAR-10.**

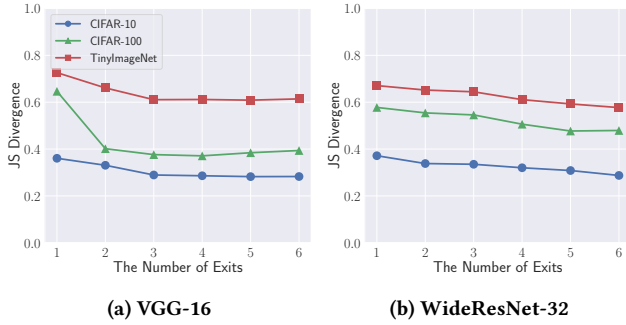
the loss distribution between members and non-members in both vanilla and multi-exit models. Due to space limitations, we only show the results of VGG-16 trained on the CIFAR-10 dataset in Figure 5. A clear trend is that compared to the vanilla VGG-16, the multi-exit VGG-16 has a much lower divergence between the classification loss (cross-entropy) for members and non-members, especially the classification loss of members becomes larger. Note that in Figure 3a, the overfitting level calculated by subtracting test accuracy from training accuracy is almost the same between vanilla and multi-exit VGG-16 trained on CIFAR-10.

We further report the loss distribution and attack performance by “exit taken”, i.e., specific exit depth of one certain multi-exit model. Figure 6 shows the loss distribution for members and non-members by “exit taken” for 4-exit VGG-16. We can see that the divergence between the loss distribution for members and non-members is becoming larger by exit depth. In other words, the early exits are less overfitting while the latter exits are much more overfitting. We further report the attack ASR score of original label-only attack against vanilla, multi-exit, and multi-exit with one specific exit (i.e., “exit taken”) in Figure 7. We can find that latter exits are much more vulnerable to membership leakages, which is consistent with the trend in loss distribution of “exit taken”.

Based on the above observation, we believe that calculating the divergence between the loss distribution of members and non-members can better characterize the overfitting level. More concretely, we leverage Jensen-Shannon (denoted as JS) divergence,



**Figure 7: The attack performance of original label-only attack against vanilla model and 6-exit model with one specific exit (i.e., “exit taken”).**

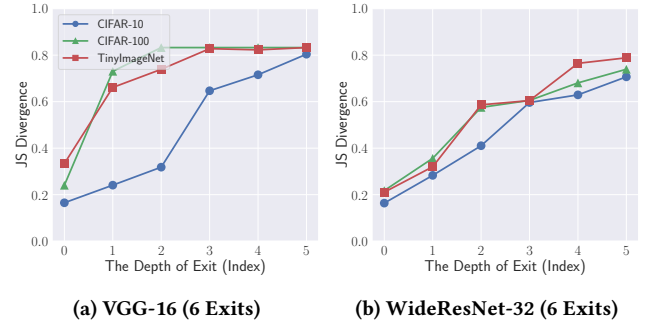


**Figure 8: Comparison of JS divergence between vanilla and multi-exit model. Note that 1 exit represents the vanilla model and 2-6 exits represent different multi-exit models.**

a widely used metric, to measure the distance of two probability distributions [15]. In Figure 8, we display JS divergence between the classification loss of members and non-members with respect to the number of exits. We can see that the JS divergence of multi-exit models is clearly lower than that of vanilla models. These results show that JS divergence is indeed a better way to characterize the overfitting level.

**Effects of the Number of Exits.** We further investigate the effects of the number of exits attached to the backbone models. More interestingly, in Figure 8, we can also find the model with more number of exits leads to lower divergence. This indicates that the number of exits is negatively correlated to the vulnerability to membership leakages. The reason is that more exits attached to the backbone model mean that more data samples leave the earlier exit points than the final exit points, which makes the model less likely to be overfitted.

**Effects of the Depth of Exits.** Here we investigate the effects of the depth of exits attached to the backbone models. Given a backbone model with 6 exits, we use the exit index (from 0 to 5) to represent the depth of exits. We calculate the JS divergence for members and non-members of each exit point separately. As shown in Figure 9, we can see that the JS divergence increases with the depth of exit. These results indicate that the depth of exits is



**Figure 9: The JS divergence of classification loss with respect to the depth of exits. The x-axis represents the depth of exit. The y-axis represents the JS divergence.**

positively correlated to the vulnerability to membership leakages, i.e., the samples leaving from deeper exit points were easier to distinguish between members and non-members. The reason for this observation is that deeper exit points imply higher capacity models, which are more likely to be overfitted to the training set.

## 4 HYBRID ATTACK

After quantifying membership leakages of multi-exit models, we conclude that the multi-exit models are less vulnerable to membership leakages and, more interestingly, we find that exit information is highly correlated with attack performance. The latter motivates us to present a new research question: *Can extra exit information (number and depth) of the multi-exit model leak more membership information about the training set?* Before answering it, more importantly, we need to answer these two-step questions first:

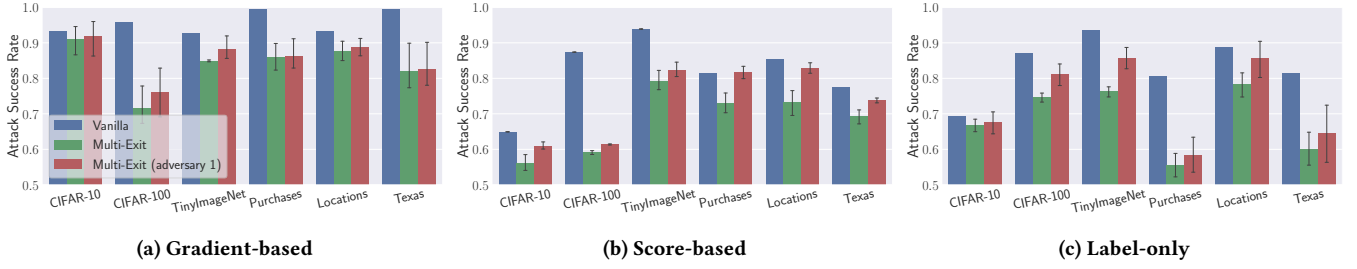
- how to obtain the exit information of target multi-exit models, especially in black-box and label-only scenarios.
- how to leverage the exit information to improve existing membership inference attacks.

In the next section, we propose a novel hybrid attack that first steals exit information and then exploits the exit information as new knowledge for the adversary. In particular, we study three different scenarios for hybrid attacks by starting with some strong assumptions and gradually relaxing them to show that far more broadly applicable attack scenarios are possible.

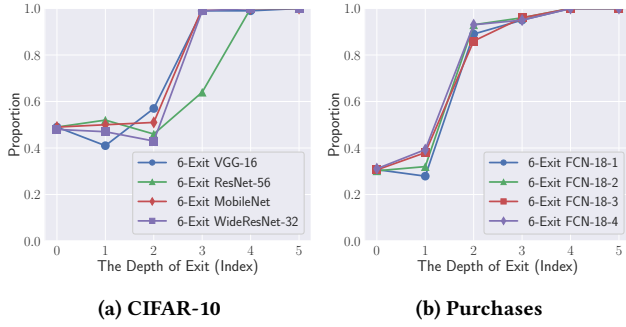
### 4.1 Adversary 1

In this section, we describe our first adversary considered for leveraging exit information to mount membership inference attacks. For this adversary, we mainly make a strong assumption about the adversary’s knowledge. In consequence, the research question of whether extra exit information will leak more membership information can be investigated in a more effective and lower-cost way. In the following, we start by defining the threat model, then describe the adversary’s attack methodology. In the end, we present a comprehensive experimental evaluation.

**Threat Model.** We assume that the adversary has direct access to exit information, i.e., exit depth. More concretely, given a data sample and a 6-exit model, the model outputs not only predictions



**Figure 10: The attack performance of different membership inference attacks on all datasets. The blue and green bars indicate the original attack on the vanilla and multi-exit models, while the red bar indicates our hybrid attack on the multi-exit model. Computer vision tasks are on VGG-16, and non-computer vision tasks are on FCN-18-1.**



**Figure 11: Proportion of non-members in all samples leaving at each exit.**

(score or label) but also exit information, e.g., predictions from the first exit point (*exit 0*) or the sixth exit point (*exit 5*). Note that here, we directly consider the exit index as the exit depth. *exit 0* means the shortest path from the entry point to the first exit, while *exit 5* means the longest path from the entry point to the final exit.

In addition, we make the same assumptions for other settings, such as data knowledge, training knowledge, model knowledge, and output knowledge.

**Methodology.** The methodology is organized into two stages: hyperparameter stealing and enhanced membership inference.

**Hyperparameter Stealing.** The adversary first queries the target model using a large number of data samples, which can come from the shadow dataset or random data samples collected from the Internet. They then count all exit indexes and sort them from smallest to largest. Thus, the largest index implies the number of exits attached to the backbone model.

**Enhanced Membership Inference.** According to the two different types of attack models used in existing attacks, we propose different methods for each attack model to exploit the exit information.

- **MLP Attack Model.** In gradient-based and score-based attacks using MLP as the attack model, given the exit information (number and depth), the adversary first converts it to a one-hot encoding, which is the same as the one-hot encoding of the true label used in the gradient-based attack. They

then provide the one-hot encoding of the exit information and other existing information to the attack model.

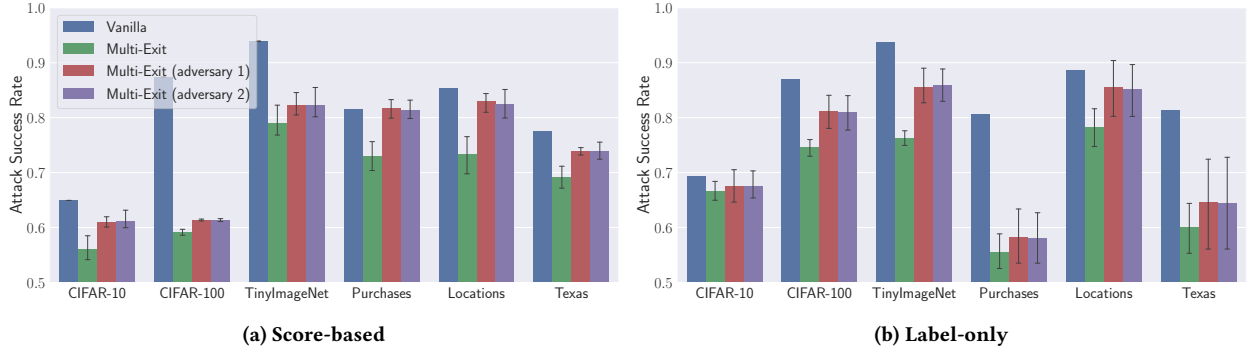
- **Decision Function.** In the original label-only attack, the adversary measures the magnitude of the perturbation and treats the data samples as members if their magnitude is larger than a predefined threshold. Here, instead of performing the above operation directly on all data samples, the adversary first separates the data samples according to their exit depths and then performs the above operation to distinguish members and non-members of each exit depth. The thresholds are also derived in this way on the shadow model.

**Experimental Setup.** For the attack models in the gradient-based and score-based attacks, the adversary has six inputs and two inputs, respectively, where the extra one is the one-hot encoding of the exit depths. Thus the new attack model has one more input component. For the evaluation metric, we again use the attack success rate (denoted as ASR). Note that in label-only attacks, we average ASR scores across all exit depths, as ASR is independent of exit depths.

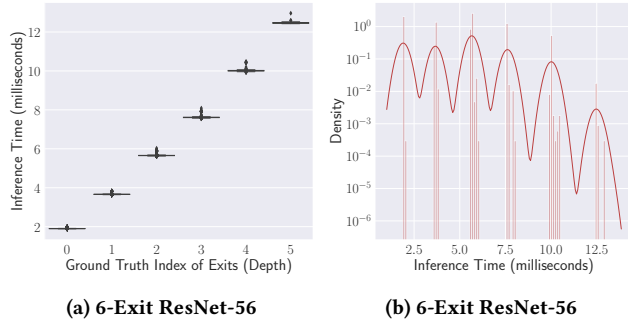
**Results.** Figure 10 depicts the performance of original attacks and our hybrid attacks (see our technique report [36] for more results). Note that, we also average the performance of multi-exit models with the number of exits varying from 2 to 6 and report the mean and standard deviation. Encouragingly, we can observe that hybrid attacks achieve clearly higher ASR score than original attacks, regardless of datasets, architectures and attack types. These results convincingly demonstrate that extra exit information of the multi-exit model leaks more membership information about the training set, compared to original information only.

More interestingly, we also find that compared with gradient-based attacks, the extra exit information used in score-based and label-only attacks can significantly improve the performance of the original attacks. Recall that gradient-based attacks are applicable in white-box scenarios. This indicates that the original gradient-based attack has already exploited almost all the information and thus can achieve the attack performance close to the upper bound. Therefore, in gradient-based attacks, extra exit information can not lead to much higher attack performance gains. In contrast, we can observe that in label-only attacks, extra exit information leads to much higher attack performance gains.

The above results fully demonstrate the efficacy of our hybrid attack. Here, we delve more deeply into the reasons for the success.



**Figure 12: The attack performance of different membership inference attacks against vanilla and multi-exit models. The blue and green bars indicate the original attack on the vanilla and multi-exit models, while the red and purple bars indicate our hybrid attack on the multi-exit model. Computer vision tasks are on VGG-16, and non-computer vision tasks are on FCN-18-1.**



**Figure 13: The inference time with respect to ground truth index of exit (a), and the density estimation by KDE based on inference time (b). They are both obtained from the same model, i.e., 6-exit ResNet-56 trained on TinyImageNet.**

Our insight is that exit depth is a critical indicator for membership inference. Figure 11 shows the proportion of non-members in all samples leaving at each exit. We can see that members tend to exit early, while non-members tend to exit late. In other words, the later exit depth itself indicates that the samples leaving here are likely to be non-members, in contrast to early exits where the samples are likely to be members. Recall that Figure 9 shows that the JS divergence of late exits is much larger than early exits, which further contributes to our hybrid attack. Such separability of members/non-members in terms of exit depth guarantees the efficacy of our hybrid attack.

## 4.2 Adversary 2

In this section, we relax the assumption that the adversary has direct access to exit information.

**Threat Model.** Different from the threat model in Section 4.1, we remove the assumption that the adversary has direct access to exit information, i.e., exit depth. This largely reduces the attack capabilities of the adversary. Given a data sample, the multi-exit model gives a prediction that includes only the score or label and does not include any exit information. This is a more realistic but

also more challenging scenario. Note that we only focus on score-based and label-only attacks, as in this scenario it is unlikely the adversary can obtain gradients or features of target models.

**Methodology.** Recall that the goal of multi-exit models is to reduce computational costs by allowing data samples to be predicted and to exit at an early layer. Therefore, the inference time for data samples inevitably varies with the depth of the exit, i.e., data samples leaving deeper exit points imply longer inference times, as shown in Figure 13a. This renders us a new perspective to determine the exit information, i.e., the magnitude of inference time actually represents the different exit depths. We refer to this method as *time-based hyperparameter stealing*.

The adversary first queries the target multi-exit model using a large number of data samples and records the inference time of these samples. These query samples can come from the shadow dataset, or random data collected from the Internet or any source. The adversary then sorts all recorded inference time as a one-dimensional array. Note that a longer inference time indicates a deeper exit point. Thus the adversary can partition this one-dimensional array into different clusters. Here we leverage Kernel Density Estimation (*KDE*) [45], an unsupervised statistical method for clustering one-dimensional data. Figure 13b shows a set of records of inference time, and we can see that *KDE* fits these records with a smoothed line. Then, several minima of the smoothed line can be used to partition them into different clusters. Thus the number of clusters means the number of exits attached to the target model, and the index of each cluster means the exit depth. The reason why we adopt *KDE* is that we want to cluster one-dimensional arrays (i.e., recorded time), for which *KDE* is well suited, while other popular techniques such as K-means [39], kNN [40] and DBSCAN [14] are multidimensional clustering algorithms.

**Experimental Setup.** We use all the same setups as presented in Section 3.3. All experiments are conducted on an NVIDIA HGX-A100 server with 4-GPU deployed. We run 4 models simultaneously at a time, each on a single GPU. Practically, in order to get a stable inference time, we calculate the inference time by averaging the time of each sample 10 times.

**Table 1: The prediction accuracy of exit depths when we run 4 models simultaneously at a time, each on a single GPU. We averaged the performance with the number of exits varying from 2 to 6 and report the mean and stand deviations.**

Target Model	CIFAR-10	CIFAR-100	TinyImageNet
VGG	0.9998±2e-4	0.9999±1e-5	0.9999±1e-4
ResNet	0.9999±2e-5	1.0±0.0	1.0±0.0
MobileNet	0.9998±8e-5	1.0±0.0	0.9998±2e-4
WideResNet	0.9996±2e-7	0.9999±1e-5	0.9999±1e-5

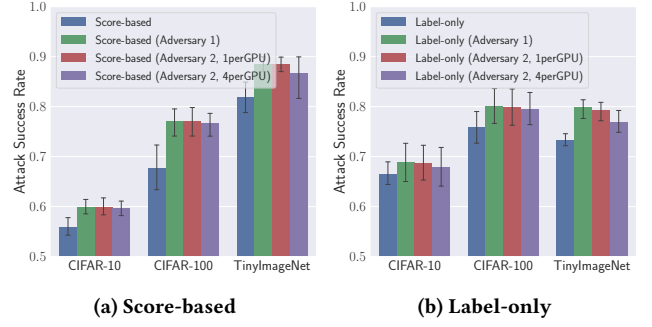
**Table 2: The prediction accuracy of exit depths when we run 16 models simultaneously at a time, every 4 on a single GPU. We averaged the performance with the number of exits varying from 2 to 6 and report the mean and stand deviations.**

Target Model	CIFAR-10	CIFAR-100	TinyImageNet
VGG	0.8027±0.1485	0.9968±0.0019	0.7167±0.0799
ResNet	0.8966±0.2158	0.9118±0.1972	0.8279±0.1859
MobileNet	0.7062±0.4015	0.9008±0.1351	0.8059±0.1336
WideResNet	0.7500±0.3561	0.9622±0.0840	0.9441±0.1105

**Results.** First, we report the prediction accuracy of exit depth over all datasets and model architectures in Table 1. As we can see, our proposed *time-based hyperparameter stealing* can achieve almost 100% accuracy. This indicates that the magnitude of inference time indeed can represent the exit depth, i.e., a longer inference time represents a deeper exit point and vice versa. Consequently, we can observe that our two adversaries achieve very similar performance for various datasets and model architectures in Figure 12. More results can be found in our technique report [36]. These results clearly demonstrate our hybrid attacks are very broadly applicable.

Recall that we run 4 target models simultaneously at a time, each on a single GPU. We further investigate whether a more complex system environment affects attack performance. To this end, we increase the number of models running simultaneously from 4 to 16, and every 4 on a single GPU. We summarize the prediction performance of exit depth in Table 2 and the attack performance in Figure 14. Encouragingly, Figure 14 shows that our hybrid attack still achieves better attack performance than the original attack, albeit a bit worse than other adversaries. Moreover, we believe that programs or models are more likely to run in simple or clean environments, at least not as complex as ours, especially in some real-time constrained applications, such as self-driving cars.

Next, we focus on the practicality of our hybrid attack against remotely deployed models, i.e., Machine Learning as a Service (MLaaS). This is a more challenging scenario where the communication channel can be very noisy. To simulate the complex communication channel, we assume that the noise  $z$  in the channel follows Gaussian distribution  $\mathcal{N}(\mu, \sigma^2)$ . More specifically, we first measure the clean inference time  $t$ , then sample the noise  $z$  from  $\mathcal{N}(\mu, \sigma^2)$ , and finally obtain the noisy inference time  $t' = t + z (z > 0)$ . Here, the  $z > 0$  is to ensure that the noisy inference time  $t'$  is larger than the clean inference time  $t$ . To obtain a stable inference time,



**Figure 14: The attack performance of different membership inference attacks against multi-exit models. 1perGPU means one model running on a single GPU while 4perGPU means 4 models running on a single GPU.**

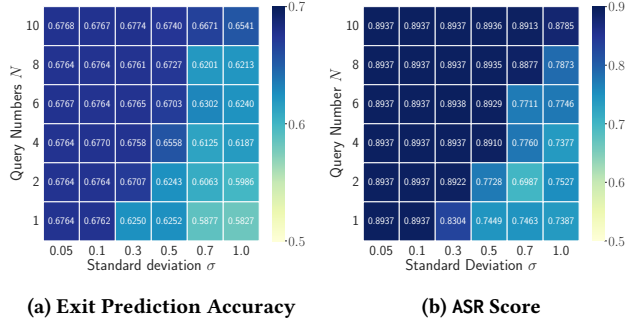
we propose a simple method that computes the inference time by averaging the noisy inference time 10 or more times, i.e., querying the remote model multiple times for each sample. Figure 15 shows the prediction and attack performance under the effect of variance  $\sigma$  of noise and query numbers for each sample. We can see that the highest prediction accuracy and ASR scores can be achieved if the number of queries is large enough, i.e., multiple queries can indeed eliminate the effect of noise. Furthermore, as shown in Figure 15a, we can find that even if the prediction accuracy of the exit depth drops significantly, it still leads to high attack ASR scores.

Furthermore, we delve more deeply into the lower bound of query numbers that can guarantee high attack performance. Consider the noise follows  $\mathcal{N}(\mu, \sigma^2)$ , and two clean inference time  $t_1$  and  $t_2$  from two adjacent exits *exit #1* and *exit #2*, respectively. Thus, the noisy inference time  $t'$  actually follows  $\mathcal{N}(t + \mu, \sigma^2)$ . The research question now is how many query numbers can guarantee the averaged noisy inference time  $\bar{t}'_1$  and  $\bar{t}'_2$  can be distinguished with high confidence. Here, we leverage Z-Test [8], a statistical technique, to determine whether two population means  $\bar{t}'_1$  and  $\bar{t}'_2$  are significantly different. To this end, we first query the target model with one certain sample many times (typically more than 100 times) to estimate the standard deviation  $\sigma$ . Then we calculate the Z-Score by the following formula:

$$Z = \frac{\bar{t}'_1 - \bar{t}'_2}{\sqrt{\frac{\sigma^2}{n_1} + \frac{\sigma^2}{n_2}}} = \frac{(t_1 + \mu) - (t_2 + \mu)}{\sqrt{\frac{\sigma^2}{N} + \frac{\sigma^2}{N}}} = \frac{(t_1 - t_2)}{\sigma \sqrt{\frac{2}{N}}} \quad (1)$$

where  $n_1$  and  $n_2$  represent the query numbers for  $\bar{t}'_1$  and  $\bar{t}'_2$ , and we consider the same query numbers  $N$  for all samples, i.e.,  $n_1 = n_2 = N$ . Besides, as Figure 13 shows, we consider the minimal time difference (ms) between two adjacent exit  $|t_1 - t_2| \in \{3, 5, 7, 9, 11\}$ . To satisfy  $p \leq 0.05$ , i.e., the average noise inference time  $\bar{t}'_1$  and  $\bar{t}'_2$  can be distinguished with more than 95% confidence, we should ensure that  $|Z| \geq 1.96$ .<sup>2</sup> Thus we can derive the relationship between  $N$  and  $\sigma$  as shown in Figure 16. Given  $|t_1 - t_2|$  and  $\sigma$ , the corresponding  $N$  denotes the lower bound of query numbers that can guarantee to divide two adjacent exits with more than 95% confidence. Recall

<sup>2</sup><https://pro.arcgis.com/en/pro-app/2.8/tool-reference/spatial-statistics/what-is-a-z-score-what-is-a-p-value.htm>



**Figure 15: The exit prediction and attack performance under the effect of query numbers  $N$  and standard deviation  $\sigma$ . The model is MobileNet trained on CIFAR-100.**

that as shown in Figure 15a, even if the prediction accuracy of the exit depth drops significantly, it still leads to a high attack ASR score, so the lower bound on the number of queries can also lead to high attack ASR score.

### 4.3 Adversary 3

Previous work [20, 35, 37, 46, 49, 51] has focused on the setup where the adversary trains a shadow model with the same architecture as the target model. Here, we investigate whether the exit information still leaks more membership information when we relax this assumption. In addition, we also investigate the effect of the shadow dataset when we relax the assumption that the shadow dataset and target dataset are identically distributed.

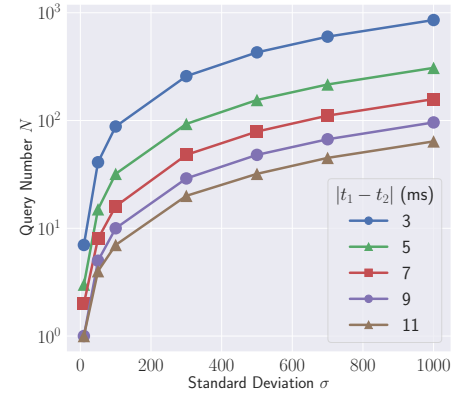
**Threat Model.** To challenge our hybrid attack, we remove the assumption that the adversary can build a shadow model with the same architecture and exit placement as the target model, which largely reduces the attack capabilities of the adversary. In addition, we perform the evaluation of the gain of the exit to attack performance by relaxing the assumption that the shadow and target datasets are identically distributed.

**Methodology.** The strategy of the third adversary is very similar to the second adversary. The only difference is that the third adversary uses a shadow model with a different architecture from the target model, which further inevitably leads to a different exit placement between shadow and target models. For example, given a target model ResNet-56 with 6 exits, the adversary can only train a different model, like VGG-16 with 6 exits, to perform membership inference. In this case, the placement of these 6 exits attached to the backbone model is different between ResNet-56 and VGG-16.

To relax the assumption on the same distribution between the shadow and target datasets, we use different datasets, e.g., CIFAR-10 as the target dataset and TinyImageNet as the shadow dataset, to launch our hybrid attack.

**Experimental Setup.** We use the same settings as described in Section 4.2.

**Results.** Figure 17 shows the attack performance when the shadow models are constructed by different architectures as the target models. First, we observe that the attack performance remains almost



**Figure 16: The relationship between query numbers  $N$  and standard deviation  $\sigma$ . The y-axis  $N$  denotes the lower bound of query numbers that can guarantee to divide two adjacent exits  $t_1$  and  $t_2$  with more than 95% confidence.**

the same in both the original attack and hybrid attack, respectively. More encouragingly, we can also find that the attack performance of our hybrid attack is clearly higher than that of the original attack. For instance, when the target model is WideResNet-32 and the shadow model is VGG-16, the ASR score of our hybrid attack is 0.7935, while that of the original attack is only 0.7205. This indicates that we can relax the assumption that the shadow model has the same model architecture and exit placement as the target model.

Furthermore, we investigate whether we can relax another assumption of the same distribution between the shadow dataset and target dataset. Figure 18 shows the attack performance on MobileNet and FCN-18-1 when the shadow dataset is distributed differently from the target dataset. We observe the performance of our hybrid attack is still better than the original attack even when the target and shadow datasets are different. Such observation hints we can relax the assumption of a same-distribution shadow dataset. See our technique report [36] for more results and the reason why we can relax these two assumptions.

In conclusion, we show that adversary 3 can free the attacker from knowing the target model (especially exit placements) and target dataset, which further enlarges the scope of the hybrid attack. These results convincingly show that the corresponding risks are much more severe under the threats caused by our hybrid attack. Furthermore, the fact that privacy risks are much more severe shown by our hybrid attacks would hinder the process of green AI that aims at fast inference and energy-efficient computing.

## 5 POSSIBLE DEFENSES

In this section, we explore the possible defense and empirically conduct the evaluation. Recall that the adversary determines the exit depths by observing the different magnitude of inference time, thus the intuition of our defense is to hide the difference in inference time for different exit points. We name our defense *TimeGuard*.

**TimeGuard.** The key point is that the multi-exit networks delays giving predictions, rather than giving them immediately. One simple but naive defense mechanism is delaying giving predictions to

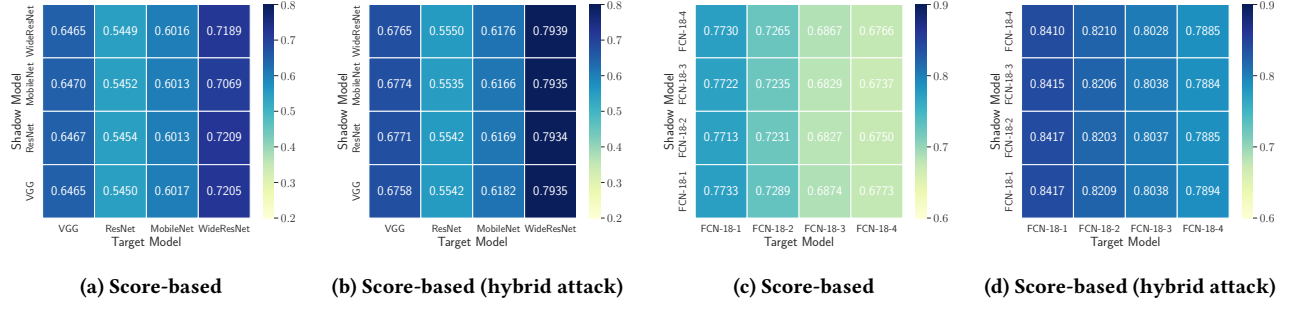


Figure 17: The attack performance when the shadow model has different architecture compared to the target model. These computer vision models (a and b) are trained on CIFAR-100, and these non-computer vision models (c and d) are trained on Purchases.

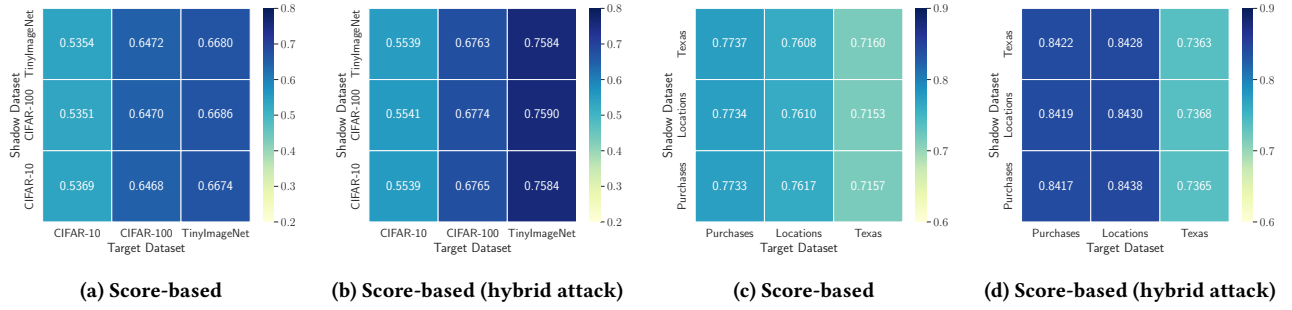


Figure 18: The attack performance when the shadow dataset comes from different distributions of the target dataset. The computer vision model (a and b) we used is MobileNet, and the non-computer vision model (c and d) we used is FCN-18-1.

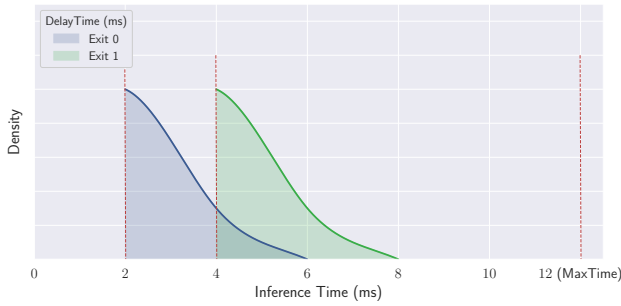


Figure 19: An illustration of how *TimeGuard* works on a multi-exit network with 3 exits. The y-axis represents the density of samples leaving at a certain delaytime among all samples at the same exit. These delaytimes follow the right part of Gaussian distribution  $\mathcal{N}(t, \sigma^2)$ .

the maximum inference time, i.e., a sample passes forward through all the layers of the model, acting like a vanilla model without any exit inserted. This behavior will make it impossible for the adversary to determine the exit information by observing inference time. However, this mechanism preserves privacy perfectly but is less efficient because it destroys one of the core ideas of multi-exit networks, which is to reduce the inference time for certain samples.

To achieve a better trade-off between privacy and efficiency, here we propose a novel mechanism for *TimeGuard* with high efficiency. See Figure 19 for an illustration of *TimeGuard* working on a 3-exit network. More concretely, consider the clean inference time  $t$  of one certain exit, thus the delay inference times of all the samples leave at this exit follow the right part of Gaussian distribution  $\mathcal{N}(t, \sigma^2)$ . See algorithm of *TimeGuard* in Algorithm 1. Here, we leverage ImageHash [9] or Sha512 [12] to calculate the unique hash  $h$ , and leverage HKDF [31] to generate the secret seed for Gaussian noise. In other words, we can obtain a fixed delay inference time  $t'$  for  $x$  since the hash  $h$  is unique (line 1) regardless of the number of queries. Thus, multiple queries on a data sample always give us the same delayed inference time, which is different from the scenario of noise variance reduction.

To further investigate the trade-off between privacy and efficiency under the influence of the standard deviation, we report attack performance and averaged inference time of each sample by varying the standard deviation in Figure 20. We can observe that as the standard deviation increases, the ASR score decreases, while the averaged inference time increases. Since the ASR score of the original attack is the lower bound of the attack performance in both the original and hybrid attacks, the intersection of the two blue lines shown in Figure 20 is the best defense scenario for the model. In other words, the corresponding standard deviation is the optimal setting for *TimeGuard*, which not only reduces the ASR score to the lower bound, but also maintains fast inference.

**Algorithm 1:** *TimeGuard* with high efficiency.

**Input:** a data sample  $x$ , standard deviation  $\sigma$ , multi-exit model  $\mathcal{M}$ , a secret global seed  $S$ .

**Output:** delay time  $t'$  for  $x$ .

- 1 calculate hash  $h$  by  $\text{Hash}(x)$ ; /\* For images/non-images,  $\text{Hash}(x)$  means  $\text{ImageHash}/\text{Sha512}$ . \*/
- 2 set random seed by  $\text{random.seed}(\text{HKDF}(h, S))$ ; /\* the seed of Gaussian noise is secret. \*/
- 3 sample Gaussian noise  $I$  by  $\text{random.normal}(t, \sigma^2, \text{size}=1)$ ; /\*  $I$  is unique and repetitive for  $x$ . \*/
- 4 observe the exit depth where  $x$  leaves by feeding  $x$  to  $\mathcal{M}$ ;
- 5 obtain clean inference time  $t$  of the exit;
- 6 calculate delay time  $t' = t + |t - I|$ ;
- 7 return delay time  $t'$ ;

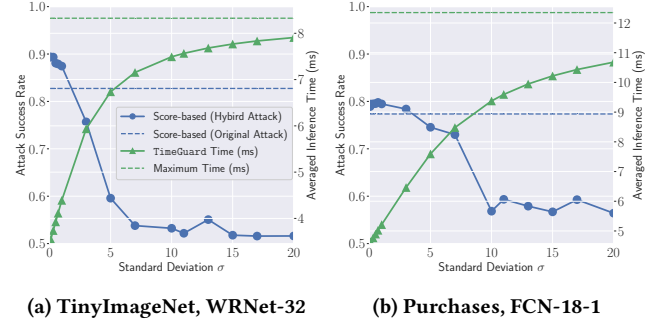
## 6 RELATED WORKS

### 6.1 Multi-exit Networks

Multi-exit architecture is first proposed by Teerapittayanon et al. [54], which adds side branches to the backbone model and aims to reduce inference run-time and energy by allowing some samples to exit early. To make the design more customizable or generic, various modifications to the architecture are proposed, [26] presents MSDNet, which can enable prediction at any time by multi-scale feature maps and dense connectivity; [29] presents a general design, called Shallow-Deep network (SDN), meanwhile identifying that vanilla model are susceptible to overthinking. In addition to various architectural designs, more efforts are focused on the training methods of the multi-exit models. Kaya et al. [29] conduct both joint training and exit-only training; Phuong et al. [43] introduce knowledge distillation [21, 44] into the training procedure, which makes the early exit learn from the last exit; Wang et al. [55] take this a step further by allowing each outlet to learn from all subsequent outlets, while it supports adaptively learning the weights of each distillation loss. More recently, more work has begun to investigate the robustness of multi-exit models. Hu et al. [25] propose that RDI-Nets achieve a “triple win” between high accuracy and low inference cost and robustness to adversarial attacks [10, 16, 53]; Hong et al. [22] propose an attack called DeepSloth to eliminate the computational savings provided by the multi-exit model; Dong et al. [13] give a DNN fingerprint identification method based on inference time to protect the intellectual property of multi-exit models. Besides, multi-exit networks are gaining significant attention and rapid development in industry [2, 23, 61] that leverage multi-exit networks to accelerate forward inference. There is also a growing body of applications deployed with multi-exit architectures, such as IoT scenarios and real-time systems [24, 27, 28, 34, 60].

### 6.2 Membership Inference Attacks

Currently, membership inference is one of the major methods of evaluating privacy risks of machine learning models [17, 19, 33, 42, 46, 49, 52, 58]. Shokri et al. [49] propose the first membership inference attack against ML models. They train multiple shadow models and then construct a dataset to train multiple attack models. These attack models take the posterior of the target sample as input



**Figure 20:** The attack performance and *TimeGuard*'s efficiency under the effect of standard deviation used in *TimeGuard*. Here, WRNet means WideResNet.

and predict its membership status, i.e., member or non-member. Then Salem et al. [46] propose a model- and data-independent membership inference attack by gradually relaxing the assumptions of Shokri et al. [49]. Later, Nasr et al. [42] focus on the privacy risk in centralized and federated learning scenarios and conduct extensive experiments under both black-box and white-box settings. Song et al. [52] investigate the relationship between adversarial samples and privacy risks arising from member inference attacks. Li and Zhang [37] and Choquette-Choo et al. [11] propose the label-only membership inference attack by changing the predicted labels of the target model, then measuring the magnitude of the perturbation. If the magnitude of the perturbation is larger than a predefined threshold, the adversary considers the data sample as a member and vice versa.

## 7 CONCLUSION

In this paper, we take the first step to audit the privacy risk of multi-exit networks through the lens of membership inference. We conduct extensive experiments and find that multi-exit networks are less susceptible to membership leakage and that exits (number and depth) are highly correlated with attack performance. We further propose a hybrid attack to improve the performance of existing membership inference attacks by using exit information as new adversary knowledge. We investigate three different adversarial settings for different adversary knowledge and end up with a model-free and data-free adversary, which shows that our hybrid attack is broadly applicable and thus the corresponding risk is much more severe than that shown by existing attacks. Finally, we present a simple but effective defense mechanism called *TimeGuard* and empirically evaluate its effectiveness.

## ACKNOWLEDGMENTS

We thank all anonymous reviewers for their constructive comments. This work is partially funded by the Helmholtz Association within the project “Trustworthy Federated Data Analytics” (TFDA) (funding number ZT-I-001 4).

## REFERENCES

- [1] <https://greenai.cloud/>.
- [2] <https://twitter.com/IntelAI/status/1089644305445208064>.

- [3] <https://www.cs.toronto.edu/~kriz/cifar.html>.
- [4] <https://www.kaggle.com/c/tiny-imagenet>.
- [5] <https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data>.
- [6] <https://sites.google.com/site/yangdingqi/home/foursquare-dataset>.
- [7] <https://www.dshs.texas.gov/thcic/hospitals/Inpatientpdf.shtm>.
- [8] <https://en.wikipedia.org/wiki/Z-test>.
- [9] <https://github.com/JohannesBuchner/imagehash>.
- [10] Battista Biggio, Iginio Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion Attacks against Machine Learning at Test Time. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 387–402. Springer, 2013.
- [11] Christopher A. Choquette Choo, Florian Tramèr, Nicholas Carlini, and Nicolas Papernot. Label-Only Membership Inference Attacks. In *International Conference on Machine Learning (ICML)*, pages 1964–1974. PMLR, 2021.
- [12] Quynh H. Dang. Secure Hash Standard. *Federal Information Processing Standard*, 2015.
- [13] Tian Dong, Han Qiu, Tianwei Zhang, Jiwei Li, Hewu Li, and Jialiang Lu. Fingerprinting Multi-exit Deep Neural Network Models via Inference Time. *CoRR abs/2110.03175*, 2021.
- [14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231. AAAI, 1996.
- [15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2672–2680. NIPS, 2014.
- [16] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations (ICLR)*, 2015.
- [17] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. LOGAN: Evaluating Privacy Leakage of Generative Models Using Generative Adversarial Networks. *Privacy Enhancing Technologies Symposium*, 2019.
- [18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.
- [19] Xinlei He, Rui Wen, Yixin Wu, Michael Backes, Yun Shen, and Yang Zhang. Node-Level Membership Inference Attacks Against Graph Neural Networks. *CoRR abs/2102.05429*, 2021.
- [20] Xinlei He and Yang Zhang. Quantifying and Mitigating Privacy Risks of Contrastive Learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 845–863. ACM, 2021.
- [21] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the Knowledge in a Neural Network. *CoRR abs/1503.02531*, 2015.
- [22] Sanghyun Hong, Yigitcan Kaya, Ionut-Vlad Modoranu, and Tudor Dumitras. A Panda? No, It's a Sloth: Slowdown Attacks on Adaptive Multi-Exit Neural Network Inference. In *International Conference on Learning Representations (ICLR)*, 2021.
- [23] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. DynaBERT: Dynamic BERT with Adaptive Width and Depth. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.
- [24] Chuang Hu, Wei Bao, Dan Wang, and Fengming Liu. Dynamic Adaptive DNN Surgery for Inference Acceleration on the Edge. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 1423–1431. IEEE, 2019.
- [25] Ting-Kuei Hu, Tianlong Chen, Haotao Wang, and Zhangyang Wang. Triple Wins: Boosting Accuracy, Robustness and Efficiency Together by Enabling Input-Adaptive Inference. In *International Conference on Learning Representations (ICLR)*, 2020.
- [26] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-Scale Dense Networks for Resource Efficient Image Classification. In *International Conference on Learning Representations (ICLR)*, 2018.
- [27] Weiwen Jiang, Edwin H.-M. Sha, Xinyi Zhang, Lei Yang, Qingfeng Zhuge, Yiyu Shi, and Jingdong Hu. Achieving Super-Linear Speedup across Multi-FPGA for Real-Time DNN Inference. *Transactions on Embedded Computing Systems*, 2019.
- [28] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 615–629. ACM, 2017.
- [29] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-Deep Networks: Understanding and Mitigating Network Overthinking. In *International Conference on Machine Learning (ICML)*, pages 3301–3310. PMLR, 2019.
- [30] Alexandros Kouris, Stylianos I. Venieris, Stefanos Laskaridis, and Nicholas D. Lane. Multi-Exit Semantic Segmentation Networks. *CoRR abs/2106.03527*, 2021.
- [31] Hugo Krawczyk and Pasi Eronen. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). *Request for Comments*, 2010.
- [32] Stefanos Laskaridis, Stylianos I. Venieris, Mário Almeida, Ilias Leontiadis, and Nicholas D. Lane. SPINN: Synergistic Progressive Inference of Neural Networks over Device and Cloud. In *Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 37:1–37:15. ACM, 2020.
- [33] Klas Leino and Matt Fredrikson. Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference. In *USENIX Security Symposium (USENIX Security)*, pages 1605–1622. USENIX, 2020.
- [34] En Li, Zhi Zhou, and Xu Chen. Edge Intelligence: On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy. In *ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 31–36. ACM, 2018.
- [35] Jiacheng Li, Ninghui Li, and Bruno Ribeiro. Membership Inference Attacks and Defenses in Classification Models. In *ACM Conference on Data and Application Security and Privacy (CODASPY)*, pages 5–16. ACM, 2021.
- [36] Zheng Li, Yiyong Liu, Xinlei He, Ning Yu, Michael Backes, and Yang Zhang. Auditing Membership Leakages of Multi-Exit Networks. *CoRR abs/2208.11180*, 2022.
- [37] Zheng Li and Yang Zhang. Membership Leakage in Label-Only Exposures. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 880–895. ACM, 2021.
- [38] Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. FastBERT: a Self-distilling BERT with Adaptive Inference Time. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6035–6044. ACL, 2020.
- [39] Stuart P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 1982.
- [40] Subhas C. Nandy, Sandip Das, and Partha P. Goswami. An Efficient K Nearest Neighbors Searching Algorithm for A Query Line. *Theoretical Computer Science*, 2003.
- [41] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine Learning with Membership Privacy using Adversarial Regularization. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 634–646. ACM, 2018.
- [42] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1021–1035. IEEE, 2019.
- [43] Mary Phuong and Christoph Lampert. Distillation-Based Training for Multi-Exit Architectures. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1355–1364. IEEE, 2019.
- [44] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for Thin Deep Nets. In *International Conference on Learning Representations (ICLR)*, 2015.
- [45] Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 1956.
- [46] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.
- [47] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520. IEEE, 2018.
- [48] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Commun. of the ACM*, 2020.
- [49] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *IEEE Symposium on Security and Privacy (S&P)*, pages 3–18. IEEE, 2017.
- [50] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR)*, 2015.
- [51] Liwei Song and Prateek Mittal. Systematic Evaluation of Privacy Risks of Machine Learning Models. In *USENIX Security Symposium (USENIX Security)*. USENIX, 2021.
- [52] Liwei Song, Reza Shokri, and Prateek Mittal. Privacy Risks of Securing Machine Learning Models against Adversarial Examples. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 241–257. ACM, 2019.
- [53] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2014.
- [54] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks. In *International Conference on Pattern Recognition (ICPR)*, pages 2464–2469, 2016.
- [55] Xinglu Wang and Yingming Li. Harmonized Dense Knowledge Distillation Training for Multi-Exit Architectures. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 10218–10226. AAAI, 2021.
- [56] Yue Wang, Jianghao Shen, Ting-Kuei Hu, Pengfei Xu, Tan M. Nguyen, Richard G. Baraniuk, Zhangyang Wang, and Yingyan Lin. Dual Dynamic Inference: Enabling More Efficient, Adaptive, and Controllable Deep Inference. *Journal of Selected Topics in Signal Processing*, 2020.

- [57] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic Early Exiting for Accelerating BERT Inference. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2246–2251. ACL, 2020.
- [58] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *IEEE Computer Security Foundations Symposium (CSF)*, pages 268–282. IEEE, 2018.
- [59] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2016.
- [60] Li Zhou, Hao Wen, Radu Teodorescu, and David H. C. Du. Distributing Deep Neural Networks with Containerized Partitions at the Edge. In *Hot Topics in Edge Computing (HotEdge)*. USENIX, 2019.
- [61] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian J. McAuley, Ke Xu, and Furu Wei. BERT Loses Patience: Fast and Robust Inference with Early Exit. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.