Stealing Neural Networks via Timing Side Channels

Vasisht Duddu*, Debasis Samanta†, D. Vijay Rao‡, Valentina E. Balas§
*Indraprastha Institute of Information Technology, Delhi, India
†Indian Institute of Technology, Kharagpur, India
‡Institute of System Studies and Analysis, Delhi, India
§Faculty of Engineering, Aurel Vlaicu University of Arad, Arad, Romania
vduddu@tutamail.com, dsamanta@iitkgp.ac.in, doctor.rao.cs@gmail.com, valentina.balas@uav.ro

***Abstract*—Deep learning is gaining importance in many applications and Cloud infrastructures are being advocated for this computational paradigm. However, there is a security issue which is yet to be addressed. An adversary can extract the neural network architecture for commercial gain. Given the architecture, an adversary can further infer the regularization hyperparameter, input data and generate effective transferable adversarial examples to evade classifiers.**

We observe that neural networks are vulnerable to timing side channel attacks as the total execution time of the network is dependent on the network depth due to the sequential computation of the layers. In this paper, black box neural network extraction attack by exploiting the timing side channels to infer the depth of the network has been proposed. The proposed approach is independent of the neural network architecture and scalable. Reconstructing substitute architectures with similar functionality as the target model is a search problem. The depth inferred from exploiting the timing side channel reduces the search space. Further, reinforcement learning with knowledge distillation is used to efficiently search for the optimal substitute architecture in the complex yet reduced search space.

We evaluate our attack on VGG architectures on CIFAR10 dataset and reconstruct substitute models with test accuracy close to the target models.

***Index Terms*—Model Extraction Attacks, Timing Side Channels, Security, Deep Learning.**

I. Introduction

Of late, neural networks have been successfully employed to many diversified areas, namely computer vision, natural language processing, business intelligence, etc. These neural networks and machine learning models are deployed in the Cloud as a service to the customers on a pay-per-query basis. Deep learning architectures have also been deployed for automated critical decision making in security applications like national critical infrastructures, malware and intrusion detection. For various military applications like unmanned combat aerial vehicle, automated target recognition and guided missile systems, the underlying decision making depends on state of the art deep learning architectures. Banks and financial organisations rely on deep learning to process the massive financial data.

The commercial value of these models make them an intellectual property for the company and are kept confidential as a black box while deployed as a service to the customers. These black box models do not reveal any information to the service users other than the predictions for the input. The commercial benefits make the black box neural networks attractive to adversaries to extract the models and circumvent the pay-per-query setting of the service or use it to evade security systems like malware and intrusion detection systems. Given the architecture of the neural networks, an attacker can further mount various privacy and security attacks like model inversion[11][10][16] and power side channel attacks[39][5] to extract the inputs and generate more accurate adversarial example[29] to evade classifiers during test time. These attacks violate the privacy of the sensitive data used for training the models or result in wrong decision making.

Key Challenges in Model Extraction Attacks. In the black box setting, the attacker has only access to the output predictions given input and lacks any knowledge about the model and the training data. Previous model extraction attacks have relied on using output predictions to infer the architecture attributes. Nevertheless, this requires a large number of queries[37], large number of models to learn the input output mapping[28] or knowledge about the model structure[29]. Further, querying the gradients of the model requires a significantly large number of queries which grow with the size of the architecture and does not reconstruct the model with a high accuracy[27]. On the other hand, side channels like power consumption[5], memory access patterns[18], cache side channel attacks[40][17], etc. require escalated attacker privileges or physical access to hardware. In this work, an attack in a black box setting to exploit timing side channels and reconstruct a target model architecture with a constant number of queries has been proposed.

Approach. The objective of model extraction attack is to search for a substitute model with similar functionality as the target neural architecture. The search space for the substitute model is very complex and large due to the large number of hyperparameters in neural networks. Our proposed attack reduces the search space by inferring the depth(number of layers) of the neural network by exploiting timing side channels. This is based on the observation that the inference time of the neural network is correlated to the depth of the neural networks, with other hyperparameters fixed, which the attacker can infer using regression. Since most services rely on commonly known state of the art architectures, knowing the depth can help the adversary identify the family of neural network models. Within the reduced search space, there could be multiple models with similar performance and functionality to the target model which is measured using the test

accuracy. To efficiently search for the optimal neural network with similar test accuracy to target model, reinforcement learning based neural architecture search is proposed which predicts an architecture using the reward from the performance of the previous proposed models[41]. Within the proposed reinforcement learning paradigm, knowledge distillation is used to train the substitute models to learn to mimic the target model predictions resulting in similar performance[15]. This reconstruction approach can be used with other attack approaches as well, like cache side channel attacks[17][40].

Observation. We evaluate our attack methodology on deep convolutional neural networks based on VGG architectures[33]. We evaluate various regressors on the timing distribution and observe that ensemble regressors give the best estimate of the depth from the inferred execution time. We show that our reinforcement learning based reconstruction can extract model within 5% of the target model test accuracy.

Main Contributions. In this paper, we make the following main contributions:

- We show that neural networks are vulnerable to timing side channel attacks and neural network architectures with different depth have different execution time(Section V).
- We propose a novel attack to infer the depth of the Neural network using timing side channels in constant number of queries in a black box setting(Section VI).
- We automate the search for optimal substitute architecture using reinforcement learning with knowledge distillation and train the substitute model to mimic the target model(Section VII).

II. Background

A. Neural Networks

Let (x, y) be the data points obtained by sampling from a probability distribution D over the space X of input feature values and space Y of output labels. The mapping from X to Y is captured by a function $f : X \rightarrow Y$. The associated loss function $l_f : X \times Y \rightarrow \mathbb{R}$ captures the error made by the prediction $f(x)$ when the true label is y. Deep Neural networks are modelled as such functions $f_h(x, \theta)$ where θ are the parameters optimised during training to obtain minimum expected loss under the constraint of the hyperparameters h. The set of hyperparameters h includes the depth of the neural network, stride and filter size of convolution and maxpool layers and regularization hyperparameters. The performance of the neural networks is measured by computing the accuracy on test data in classification tasks.

B. Security and Privacy in Machine Learning

Machine learning is known to have various security and privacy issues in adversarial setting[9]. Adversarial examples and model extraction attacks violate the security of machine learning systems. Adversarial examples are data instances that fool the classifier into misclassifying the image by either poisoning the training data or evading the decision logic during inference[29]. Machine learning models can be extracted by adversary to reconstruct a substitute model with similar functionality as the target model and hence violating the intellectual property of the service provider[37][39].

Given learning algorithm f and input data point x, $f_h(x; \theta)$ computes the output prediction label for the optimized parameters θ. Privacy concerns in Machine learning is with regards to the input data(x), training data, model parameters(θ) and the model computation($f_h(x; \theta)$). Model inversion attacks[11][10] try to identify the input data instance for an output given the model details. This attack can be extended in a distributed learning setting and the quality of input data samples extracted can be improved using Generative Adversarial Networks[16]. Power side channel attacks on neural network accelerator allow an adversary to infer the input passed to the model given the knowledge about the neural network architecture[39]. These attacks violate the privacy of the input data instances. Given a black box access to the model, an adversary can identify whether a given data point is part of the training data using membership inference attacks and hence violating the privacy of an individual data points in the training dataset[31][30]. Model extraction attacks and hyperparameter stealing[38] violate the privacy of the parameters. Side channel attacks violate the computational privacy requirement by allowing the adversary to infer machine learning model details during computation[28][5].

Some of these attacks assume that the underlying neural network architecture is known to the adversary. In this work, we provide a way to obtain the architecture to facilitate other privacy attacks.

C. Side Channel Attacks

Implementation and physical characteristics of systems expose information about the underlying computation which can be extracted in the form of side channels. Power consumption and Timing side channels are some common manifestations of side channel attacks overlooked during system design. While side channels like power channels are accurate and reveal significant information about the target architecture, they require expensive equipment and probes to monitor and measure the power consumption[23].

Timing Channels. Timing Channels arise when the program uses branching or conditional statements dependent on the secret information which influences the runtime(external timing attacks) or when the access timing correlates strongly with the program locality dependent on the secret(internal timing attacks). Timing side channels have been used to exploit various cryptographic implementations including RSA, Diffie-Hellman[22] and OpenSSL[7][6].

Termination Channels. Based on when a program terminates, an adversary can infer the secret provided that the termination of program is dependent on the secret. Termination channels arise when there is a strong correlation between the secret and runtime in sequential execution[3]. In sequential programs, termination channels reveal only one bit of

secret about the system for an execution run as compared to other side channels[35], [12]. Termination channels can be considered as a special case of timing channels. In our proposed attack, the depth of the network(secret) determines the termination of the neural network sequential computation which can be exploited to infer the model depth.

III. Problem Statement

Neural Network models have a commercial value as companies invest significant resources and manpower in designing and training them to achieve state of the art performance on various decision making tasks. The knowledge about the model helps the adversary evade the classifiers[29], infer training data samples[30][32], extract regularization hyperparameter[38] and inputs[10][39]. Prior model extraction attacks require large number of queries[27][37], multiple models[28] or escalated adversary privileges[5][18][40][17] to extract model attributes. Our work addresses the inefficiency of model extraction attacks by requiring only constant number of queries during the attack phase in a black box setting.

Model Extraction Attack. A Machine Learning model extraction attack is a search problem where an adversary with a black-box access to a target model f_{target}, searches for a substitute model $f_{substitute}$ that approximates f_{target} using minimum possible queries. We use test accuracy as a metric to measure the similarity of $f_{substitute}$ and f_{target} model for inputs distributed like the training data samples.

To extract models in a black box setting where the adversary does not have the knowledge about the neural network and training data, side channels play an important role in revealing information about the system to reduce the entropy of the secret and hence constrain the search space. To this extent, we exploit the timing side channels in neural networks which does not require elevated privileges and can be leveraged remotely in a black box setting.

Timing Side Channel Attack. The goal of timing side channel attacks is to reduce the entropy of secret(architecture) held by the target system by using the differences in the response time of the system.

In our work, we exploit timing side channels to infer the depth of the neural network architecture and for the same depth, the adversary searches for an optimal substitute neural network $f_{substitute}$ having test accuracy close to f_{target}.

IV. Threat Model

Setting. There are two settings for machine learning in adversarial setting: white box and black box setting, based on the adversary's knowledge about the target system. In white box setting, the adversary has access to the underlying data, learning algorithms, architecture of the model, training parameters and target model architectures which allows the adversary to compute the output of the intermediate layers. On the other hand, an adversary in black box setting has access to the API to send the input image and receive the corresponding output. This is the setting of machine learning as a service.

This work considers a black box setting in which the adversary does not have access to the model internals and can only query the trained model and obtain the corresponding output predictions. The adversary sends an input to the target model and measures the response time corresponding to the execution of the neural network. Hence, we consider a weak adversary with no knowledge about the target model.

Hardware. GPUs are extensively used for training neural network architectures since it is computationally expensive while for inference, CPUs or dedicated neural network accelerators improve the performance of the neural network computation[36]. Majority of machine learning Cloud service providers use CPUs[14]. We consider CPUs as our hardware for our attack methodology but this attack can be extended to other hardware accelerators as well. We assume that the adversary has access to the same processor as the target model which can be openly obtained in most of the ML as a Service(MLaaS) platforms like Amazon Sagemaker and Facebook which heavily rely on CPUs for the inference and provide the hardware specifications[1][2]. The target hardware or the service can be purchased by adversary to run the queries and generate the attack dataset which is a one time operation and done as part of the setup phase for the attack.

Data. We assume that the adversary has knowledge about the underlying data distribution and statistics along with the input-output dimensions and range of values that they can take. The adversary can iteratively generate sample inputs and identify the data points belonging to the training data using membership inference to reconstruct the training data[32][30].

V. Timing Channel In Neural Networks

The execution time of a neural network during inference depends on the hyperparameters of the neural network and various hardware factors. We analyse the execution time and its variation with the hyperparameters for the maxpool, convolution and fully connected layers in the neural networks. This provides us the motivation for relating the neural network execution time with the hyperparameters. We show that the neural networks have timing side channels which can be used to infer the hyperparameters due to the correlation with the inference execution time.

Convolution and Maxpool. The execution time of convolution layers is proportional to the number of multiplications(Figure 1(a)) while for the maxpool it depends on the size of input map and kernel size(Figure 1(b)). The number of multiplications in convolution is given by: $o_w \times o_h \times c_o \times f_w \times f_h \times c_i$ where o_w and o_h are the output matrix width and height, c_i and c_o are the input and output number of channels and f_h and f_w represent the filter width and filter height. We see that the there is an inverse relation between the execution time and the stride(Figure 1(c)). The output size decreases with increasing stride resulting in a decrease in the execution time.

Fully Connected. Given two fully connected layers of input nodes m and output nodes n, the execution time varies linearly with the total number of multiplications, $m \times n$(Figure 2(a)).

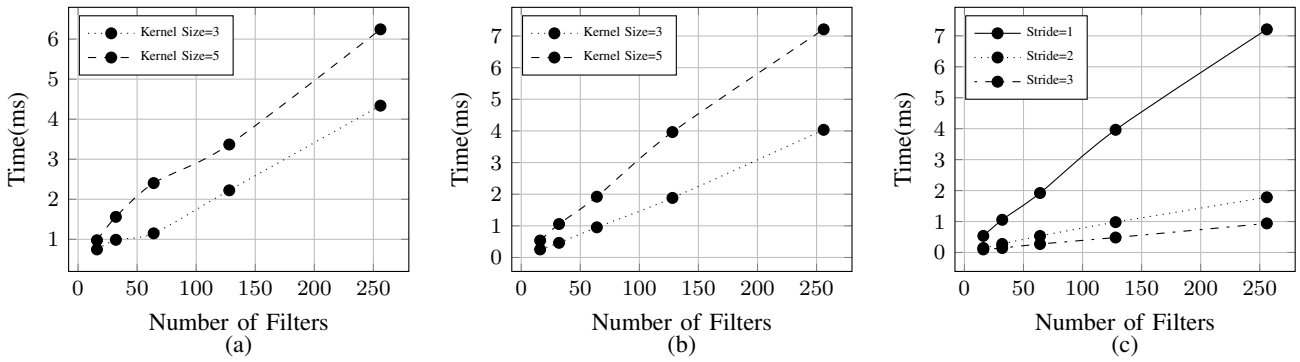

Fig. 1: (a) Convolution: Time increases linearly with kernel and filter size; (b) Maxpool: Time increases linearly with kernel and filter size; (c) Maxpool: Time decreases with increase in stride.

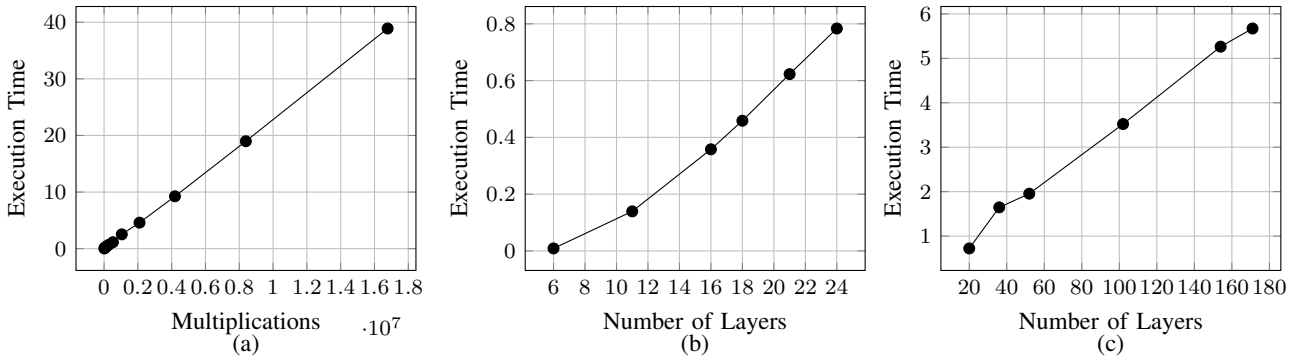

Fig. 2: (a) Fully Connected: Time varies linearly with the total number of multiplications; (b) Neural Networks without skip connections: Time varies linearly with depth; (c) Neural Networks with Skip Connections: Time varies linearly with depth

Variation with Depth. Due to sequential computation of neural networks, the total execution time of the neural network is the sum of the execution time of individual layers. In Figure 1(b) and Figure 2(c), the execution time increases linearly with the increase in the network architecture depth. This forms the basis of our attack methodology which we exploit to infer the depth of the neural network given the execution time.

Hardware Factors. Neural Networks are embarrassingly parallel and all the computations in a particular layer are executed in parallel which enables optimisations like model and data parallelism to improve the performance. While the neural networks fetch the data and parameters from the memory, the program could incur latencies during reading or storing data, contention of threads and inefficient caching. We assume that all these factors effect the execution time of all the neural networks in the same way since we assume the same hardware.

VI. Attack Methodology

Typically, side channels reveal only a part of the secret in the target system and identifying the rest of the secret is a search problem[40].

The proposed attack is broadly divided into three phases as shown in Figure 3:

- **Setup**: Adversary collects the data and model timing details for a particular hardware to be used in the actual attack. This is a one time operation prior to the attack.
- **Attack**: Adversary queries the target neural networks and measures the execution time of the target model to infer the depth.
- **Reconstruction**: Adversary searches for the most optimal neural network with similar test accuracy as the target neural network model from the reduced search space by fixing the depth.

A. Setup Phase

For a given hardware, the setup phase is required to be performed only once and the collected data can then be used for stealing any model run on the same hardware. The adversary in this phase performs two tasks: a) generates data with target model predictions as labels to train the template model using knowledge transfer and b)create a dataset with neural network depth for different execution time to train the regressor model.

a) Substitute Model Training Data. The adversary passes data instance $\mathbf{x}_i$ from the training data, generated using

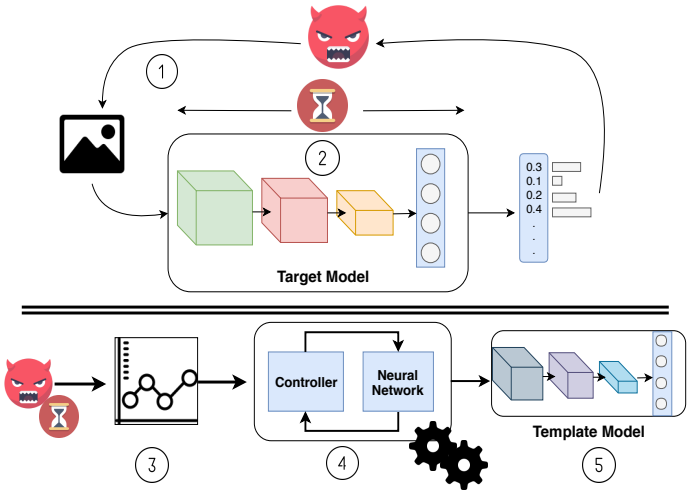

Fig. 3: Model Extraction using Timing Side Channels: 1) The adversary queries the target model by sending an input; 2) Adversary measures the execution time of the target model for the provided input; 3) The execution time is passed to the adversary's regressor model which predicts the depth of the target model; 4) The depth of the neural network is used to reduce the search space for the architecture search within the constrained search space; 5) Architecture Search produces the optimal architecture using reinforcement learning within the given search space which has very close test accuracy as the target model

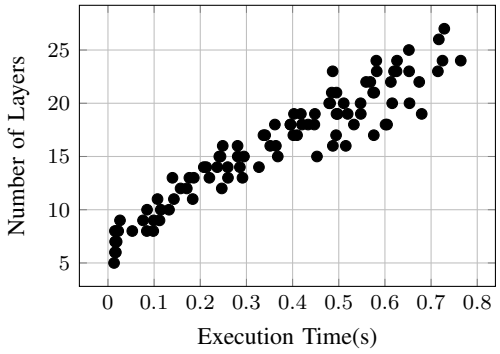

Fig. 4: Scatter Plot of the attacker data: Execution time vs Number of layers. Each point in the plot is obtained by varying the parameters of the neural network layers. The attacker fits a regressor to predict the depth of the neural network given the execution time.

iterative membership inference attack, to the target model to get the corresponding predicted scores $\hat{y}_i$ for each of the classes from the target model. This data point($\mathbf{x}_i$) and corresponding prediction($\hat{y}_i$) from the target model is included as part of the new training data for the substitute model. We train the substitute models using knowledge distillation to mimic the target model and increase the similarity between the parameters[15][4].

b) Creating Attack Model Dataset. The adversary populates the attack model dataset which contains the time taken for inference and corresponding depth of the neural networks along with the number of parameters for the corresponding model. This relies on the assumption that the adversary has access to the same processor as the target model is running. The adversary fits a regressor to the data to learn the correlation between the execution time and the neural network depth. We evaluate the execution time of 100 deep neural network models and compute the inference execution time and plot them with respect to the number of layers(Figure 4). Each point in the plot is obtained by varying the parameters for a selected neural network architecture.

In black box setting, the adversary can only infer the depth of the network from the execution time as the adversary has no access to the model details. Unlike a black box adversary, a white box adversary has access to the model and given the depth of the model and adversary can compute the hyperparameters of the individual layers as well. We however consider a black box setting which is more practical.

B. Attack Phase

Query. The adversary sends queries to the target model and computes the overall execution time averaged across all the queries. Each query reveals the same information(execution time) about the neural networks and hence requires a constant number of queries, independent of the model architecture.

Regression. The average execution time is used to estimate the depth of neural network using regressor trained on the attack model dataset created during the setup phase.

C. Reconstruction Phase

The search space of all possible Neural Networks is very large and complex due to which designing the neural network manually is hard and requires optimal search strategies to reduce and simplify the search space. The depth of the neural network reduces the search space and the model exploration is automated using reinforcement learning which outputs the best model architecture in the constrained search space[41]. A Recurrent Neural Network(RNN) based controller predicts the hyperparameters of each layer in the template neural network sampled using the reward from the previous proposed architecture. The parameters of controller RNN θ_c are optimised based on performance of the predicted architecture using policy gradient method. Formally, the controller predicts the architectures through actions $a_{1:T}$ and the predicted model tries to achieve accuracy R which is used to compute the reward signal to train the controller. The goal is to maximise the expected accuracy of sampled architecture give by,

$$J(\theta_c) = E_{P(a_{1:T};\theta_c)}[R] \tag{1}$$

The training of each proposed model is done using knowledge distillation where the loss function for training the substitute model is the distillation L2 loss function[15] between the substitute model predictions($y^{template}$) and target

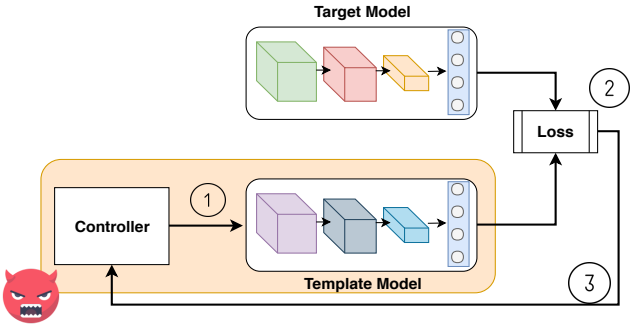

Fig. 5: Reinforcement Learning Based Architecture Search with Distillation: 1) The Recurrent Neural Network controller proposes a neural network architecture by selecting values from the state search space based on the the reward from the previous iteration; 2) The proposed model trains by using the predictions of the target model as the output predictions and mimics the behaviour of the target model[15][4]; 3) The loss of the template and target model predictions is used to compute a reward which is sent to the controller to predict a better performing model

model(y^{target}) predictions instead of the true labels(y) for a given data point $x \epsilon D$ and is given as,

$$L = \sum_{i=1}^{n} \left(y_i^{target} - y_i^{template} \right)^2 \quad (2)$$

The substitute model mimics the target model which makes the search efficient and the model mimics the target neural network model(Figure 5).

VII. Evaluation

Data. For all the experiments, we use CIFAR10 dataset[24] containing 60,000 32x32 colour images in 10 classes with around 6000 images per class where the classes are mutually exclusive. We use 50,000 images as training images and 10,000 for testing.

Experiment Setup. We use Intel Xeon Gold 5115 server processor with a 2.4GHz clock speed, 196GB of main memory and 40 cores. All the reported number are an average of 20 inference runs. For accurate timing of the neural network inference, we used $process\_time()$ from the $time()$ python library which computes the time interval for running the inference using the CPU counter and is not effected by execution of other unrelated processes. The clock has a tick rate(ticks/s) of 10,000,000 which indicates a high resolution of 1e-07.

A. Attacker Model: Regressor

We evaluate our attack methodology on three VGG based convolutional neural networks and measure the performance of various regressor in predicting the depth of the architectures given the inference time. Based on the values of R^2 score and Mean Squared Error(MSE), we observe that the ensemble regressors perform better than the counterparts. Given the attacker dataset $D_A = \{(X_1, Y_1), \cdots, (X_N, Y_N)\}$ of i.i.d. random variables, for fixed x ϵ D_A , our goal is to estimate the regression function R(x) = E{Y |X = x}.

The evaluation of all the layers in done in sequence while the computations of the parameters in a single layer are executed in parallel. For the attacker dataset, we use both the depth of the neural network as well as the number of parameters in the neural networks for the corresponding execution time as shown in Table I. For a good regressor, we require the model to explain more variance about the data and at the same time generalize well over the data samples.

Architecture	Parameters	Inference Time(s)
VGG16	138,357,544	0.59683408
	156,053,800	0.86338694
	133,048,360	0.49502961
VGG19	143,667,240	0.7642632
	168,441,896	1.11189311
	136,588,328	0.55131464

TABLE I: Attacker Dataset: Depth of the architecture and the number of parameters as features for the regressor.

To evaluate the performance of various regressor models, we use R^2 score to measure the variance explained by the model and the mean squared error to measure the error in estimating the depth of the network(Table II). We observe that ensemble approaches like boosted decision trees(BDT) and random forrest(RF) regressor have a higher R^2 score and lower mean squared error to estimate the depth more accurately as compared to the linear models which fail to capture the variance of the attacker dataset(Table II).

Regressor	Mean Squared Error	R^2 Score
Support Vector Machine	7.5405	0.7295
Decision Tree	5.375	0.80719
Linear(Ridge)	4.46533	0.8398
Boosted Decision Tree	4.1947	0.8495
Random Forrest	3.7664	0.8648

TABLE II: Evaluation of Regression Models using R^2 score and Mean Squared Error(MSE): Ensemble approaches estimate the depth better with higher R^2 score and lower MSE.

Once the adversary has estimated the depth of the neural network, he can use that information to reduce the search space to find the optimal model with test accuracy close to the target model. We estimate the depth of the ensemble regressors on the target deep neural networks from the corresponding execution time as shown in Table III. Since the output of the regressor is a continuous variable, we round it to the nearest larger integer and the regressors estimate the correct depth for all the three target neural networks. The target neural network architectures we evaluate are based on VGG convolutional networks.

B. Reconstruction using Architecture Search

For the reconstruction phase, we evaluate our reinforcement learning based architecture search with knowledge distillation based on the test accuracy of the substitute model generated and we observe that our models are within 5% of the target model architecture(Table IV). For constraining the search space to be tractable, we specify the convolutional layer parameter range for the kernel size(k) ϵ {3,5} and

	Architecture	Parameters	Execution Time	Predicted Depth	True Depth
Model 1	[32(3),32(3),MP,64(3),64(3),MP,128(3),128(3),MP]	309,290	0.036057	8.8(RF);8.1(BDT)	9
Model 2	[32(3),32(3),MP,64(3),64(3),MP,128(3),128(3),MP,256(3),MP]	595,242	0.10738	10.15(RF); 10.02(BDT)	11
Model 3	[32(3),32(3),MP,64(3),64(3),MP,128(3),128(3),128(3),MP,256(3),256(3),MP]	1,334,442	0.18594	12.8(RF);12.6(BDT)	13

TABLE III: Regression Model Predictions: The ensemble regressors correctly predict the depth of the network from execution time; The architecture of the model is in format: filters(kernel size) and MP is the Maxpool Layer

	Reconstructed Architecture	Parameters	Original Accuracy	Reconstructed Accuracy
Model 1	[64(3),32(5),128(3),64(5),128(3),64(5),32(5),128(3),GAP]	535,114	88.03%	86.06%
Model 2	[32(5),32(5),64(3),32(3),64(5),128(3),128(3),32(3),64(3),64(5), GAP]	639,978	89.26%	85.65%
Model 3	[64(3),128(3),128(3),32(5),64(5),128(3),128(3),32(3),128(5),128(3),64(5),128(5),GAP]	889,834	90.19%	85.3%

TABLE IV: Evaluation of Reconstruction using Reinforcement Learning based Architecture Search with knowledge distillation; The architecture of the model is in format: filters(kernel size) and GAP is the Global Average Pooling

the number of filter(n_f) ϵ {32,64,128} which are commonly used hyperparameters for most state of the art networks[33]. The architecture search approach explores the space of 50 models, and outputs the architecture corresponding the highest accuracy. To improve the performance of the substitute model, we use fully convolutional net architecture by replacing max-pool layer with convolutional layers with higher stride[34]. The reward used for updating the controller is the maximum validation accuracy of the last 5 epochs cubed which is clipped in the range (-0.05, 0.05) to ensure that the gradients do not overshoot. The controller RNN includes 1 LSTM cell and 32 hidden units and each proposed template model is trained for 20 epochs.

VIII. Discussion

Why does the attack work? The depth of the neural network is an important hyperparameter which determines the performance. The execution time of the neural networks adds up along the depth in sequence which results in a direct relationship between the depth of the neural network and the total execution time which makes them vulnerable to timing side channel attacks. By inferring the depth, we can reduce the overall search space for finding a substitute architecture close to the target black box model.

Constant Number of Queries. Every query for previous model extraction attack leaks a bit of additional information which is used to train and improve the performance of the template model $f_{substitute}$ and make it similar to the target model f_{target} in terms of test accuracy. In our attack, each of the query the attacker makes during the attack phase to the target model, reveals the same bit of information(execution time averaged across all queries) which allows to make constant number of queries, independent of the architecture.

Variation Across Datasets. The timing distribution for neural networks is specific to the training data used. For the same architecture, using different datasets results in different execution time due to different number of computations as shown in Table V. The regressor is specific to a particular timing distribution unique to a dataset and a different attack model has to be trained to fit different timing distribution.

Search Space. Extracting a neural network in a complete black box setting is a difficult search problem where the attacker wants to find the most optimal neural network with the target accuracy similar to the target model in a complex and large search space. While previous attack techniques assume elevated privileges and strong adversary with physical access to the hardware, they are able to reduce the search space drastically and reconstruct neural networks which are very close to the target model. Timing side channels helps the attacker to infer the depth of the neural network in constant number of queries which reveal only limited information. This results in increased cost for searching for the neural network. However, our attack reduces the search space and policy gradient based architecture search makes the exploration of the complex neural network space efficient. Further, the search can be made more efficient by combining other side channel attacks to infer other architectural details and further reducing the search space.

Architecture	Dataset	Inference Time(s)
Alexnet	MNIST	0.28958
	CIFAR10	0.36527
VGG	MNIST	0.44178
	CIFAR10	0.63833

TABLE V: Same architecture trained on different datasets have different timing distribution.

Extending to Remote Setting. While we evaluate the attack on a local model, but this attack can be extended to remote setting. A round trip time model for remote timing attacks has been proposed by Crosby et al.[8], where the total response time(t_{res}) is a linear function of the actual processing time(t_{proc}), network propagation time(t_{net}) and the jitter given as,

$$t_{res} = a \times t_{proc} + t_{net} + jitter \tag{3}$$

where a is a constant. This would require filtering the jitter from the roundtrip latency of sending an input and receiving the corresponding prediction using low order percentiles.

IX. Related Work

Model Extraction Attacks. Model extraction attacks solve a system of linear equation with known and unknown variables using model output predictions[37][26]. This approach, however, requires large number of queries to extract the target model and corresponding parameters. An alternative approach to extract models is by using multiple machine

learning models or "metamodels" to learn the input output relation and output the corresponding model attributes. While this is done in constant number of queries, it requires to train multiple machine learning models to learn the input output mapping[28]. Instead of using model output predictions, querying model explanations like gradients can allow to infer the model architecture in lesser number of queries[27]. However, the number of queries required to infer the model attributes depends on the architecture size and the attack is not practical for large models.

Side Channel Attacks. One research direction to extract machine learning models is to exploit side channels on the system running the model. Side channels reveal only certain part of the secret and extracting the complete secret can be modelled as a search problem. Power consumption of the model during inference can be used to extract the model accurately by using a combination of differential power analysis and simple power analysis[5]. Since, deep neural networks follow a pattern of memory accesses owing to their computations, monitoring the memory access patterns to infer the architectural details of neural networks can help to reduce the overall search space by identifying hyperparameters[18]. However, these attacks require the adversary to have physical access to the hardware and uses additional equipment like oscilloscope and probes. Cache side channel attacks can enable adversary to identify the number of layers and other hyperparameters from the size of matrices being fetched from the memory and difference in fetching time[40][17]. This attack assumes resource sharing in hardware and both the target and victim process should run on the same processor.

Model Extraction Defences. Several defences have been proposed to mitigate the attacks that rely on output predictions. A simple defence is to reduce information leakage from the prediction logits by clipping the confidence values at the cost of utility[37]. To prevent the loss in utility, a stateful mechanism to monitor the variation in query distribution to indicate model extraction has been proposed[20]. Lee et al.[25] add a reverse sigmoid layer to add noise to the output predicts to degrade the performance of the stolen model without varying the performance of the target model. A stateful defence can quantify the extraction status model by continuously monitoring user queries using information gain metrics and alert when the information gain is more than a given threshold[21]. Trusted hardware like Intel SGX can protect the confidentiality and integrity of the model by moving the model offline to the user's system[13].

All the defences mentioned are proposed for attacks that use the output prediction scores but do not consider timing side channels in their threat model.

X. Mitigation

The main reason for the manifestation of timing side channels in neural networks is the sequential computation of layers and the hyperparameters which determine the execution time. We propose possible approaches to defend against and mitigate timing attacks and side channels and design robust neural networks resistant to model extraction using our proposed attack.

Adding Noise to Execution Time. Instead of having a neural network with timing dependent on the depth of the neural network, one defence is to remove the linear dependency of the neural network depth(secret) on the execution time of the neural network. This can be done by designing neural networks by including dummy computations, adding random noise and using additional layers to add latency to the neural network execution time. This approach however effects the utility and hence resulting in a tradeoff between model privacy and the utility.

Adversarial Machine Learning. The second mechanism is a training phase defence where the attack can be viewed as an adversarial machine learning problem. Here, the attacker is trying to fit the best possible curve on the training data to make correct inferences about the depth of the neural network based on the execution time while the defender is trying to poison the attacker dataset so that the regression problem gives wrong predictions of the network depth, hence, degrading the performance of the substitute model. This can be done by using a skewed clock in the hardware to inject adversarial data points into the attacker dataset[19].

XI. Conclusion

We show that neural networks are vulnerable to timing side channels attacks as the total execution time depends on the sequential computation along the number of layers or depth. For a weak adversary in a black box setting, our proposed approach exploits the timing side channels to infer the depth of the neural network architecture. We show that ensemble based regressors perform better than their linear counterparts. Further, extracting a neural network architecture by exploiting side channels is a search problem which can be done efficiently using reinforcement learning. This approach can be used along with other attacks like cache attacks and memory access pattern monitoring where the side channels leak only a part of the secret. We evaluate our attack on VGG like deep learning architectures and show that we can reconstruct the model within 5% of the test accuracy of the target neural network.

Acknowledgment

The authors would like to thank Virat Shejwalkar(UMass Amherst), Sasikumar Murakonda(National University of Singapore) and the reviewers for their feedback and helpful comments which greatly improved the quality of the paper.

References

[1] "Amazon ec2 instance types," https://aws.amazon.com/ec2/instance-types/, 2018.

[2] "Amazon sagemaker instance types," https://aws.amazon.com/sagemaker/pricing/instance-types/, 2018.

[3] A. Askarov, S. Hunt, A. Sabelfeld, and D. Sands, "Termination-insensitive noninterference leaks more than just a bit," in *Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security*, ser. ESORICS '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 333–348.

[4] L. J. Ba and R. Caurana, "Do deep nets really need to be deep?" *CoRR*, vol. abs/1312.6184, 2013. [Online]. Available: http://arxiv.org/abs/1312.6184

[5] L. Batina, S. Bhasin, D. Jap, and S. Picek, "Csi neural network: Using side-channels to recover your artificial neural network information," Cryptology ePrint Archive, Report 2018/477, 2018, https://eprint.iacr.org/2018/477.

[6] B. B. Brumley and N. Tuveri, "Remote timing attacks are still practical," in *Proceedings of the 16th European Conference on Research in Computer Security*, ser. ESORICS'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 355–371. [Online]. Available: http://dl.acm.org/citation.cfm?id=2041225.2041252

[7] D. Brumley and D. Boneh, "Remote timing attacks are practical," in *Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12*, ser. SSYM'03. Berkeley, CA, USA: USENIX Association, 2003, pp. 1–1. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251353.1251354

[8] S. A. Crosby, D. S. Wallach, and R. H. Riedi, "Opportunities and limits of remote timing attacks," *ACM Trans. Inf. Syst. Secur.*, vol. 12, no. 3, pp. 17:1–17:29, Jan. 2009. [Online]. Available: http://doi.acm.org/10.1145/1455526.1455530

[9] V. Duddu, "A survey of adversarial machine learning in cyber warfare," *Defence Science Journal*, vol. 68, no. 4, pp. 356–366, 2018. [Online]. Available: https://publications.drdo.gov.in/ojs/index.php/dsj/article/view/12371

[10] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015.

[11] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing." in *USENIX Security Symposium*, 2014.

[12] S. K. Haider and M. van Dijk, "Revisiting definitional foundations of oblivious RAM for secure processor implementations," *CoRR*, vol. abs/1706.03852, 2017. [Online]. Available: http://arxiv.org/abs/1706.03852

[13] L. Hanzlik, Y. Zhang, K. Grosse, A. Salem, M. Augustin, M. Backes, and M. Fritz, "Mlcapsule: Guarded offline deployment of machine learning as a service," *arXiv preprint arXiv:1808.00590*, 2018.

[14] K. Hazelwood, S. Bird, D. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro *et al.*, "Applied machine learning at facebook: A datacenter infrastructure perspective," in *High Performance Computer Architecture (HPCA), 2018 IEEE International Symposium on*. IEEE, 2018, pp. 620–629.

[15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[16] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017.

[17] 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.

[18] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *Proceedings of the 55th Annual Design Automation Conference*, ser. DAC '18. New York, NY, USA: ACM, 2018, pp. 4:1–4:6. [Online]. Available: http://doi.acm.org/10.1145/3195970.3196105

[19] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," *CoRR*, vol. abs/1804.00308, 2018. [Online]. Available: http://arxiv.org/abs/1804.00308

[20] M. Juuti, S. Szyller, A. Dmitrenko, S. Marchal, and N. Asokan, "PRADA: protecting against DNN model stealing attacks," *CoRR*, vol. abs/1805.02628, 2018. [Online]. Available: http://arxiv.org/abs/1805.02628

[21] M. Kesarwani, B. Mukhoty, V. Arya, and S. Mehta, "Model extraction warning in mlaas paradigm," *arXiv preprint arXiv:1711.07221*, 2017.

[22] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '96. London, UK, UK: Springer-Verlag, 1996, pp. 104–113. [Online]. Available: http://dl.acm.org/citation.cfm?id=646761.706156

[23] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '99. Berlin, Heidelberg: Springer-Verlag, 1999, pp. 388–397. [Online]. Available: http://dl.acm.org/citation.cfm?id=646764.703989

[24] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)." [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[25] T. Lee, B. Edwards, I. Molloy, and D. Su, "Defending against model stealing attacks using deceptive perturbations," *arXiv preprint arXiv:1806.00054*, 2018.

[26] D. Lowd and C. Meek, "Adversarial learning," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, ser. KDD '05. New York, NY, USA: ACM, 2005, pp. 641–647. [Online]. Available: http://doi.acm.org/10.1145/1081870.1081950

[27] S. Milli, L. Schmidt, A. D. Dragan, and M. Hardt, "Model reconstruction from model explanations," *arXiv preprint arXiv:1807.05185*, 2018.

[28] S. J. Oh, M. Augustin, M. Fritz, and B. Schiele, "Towards reverse-engineering black-box neural networks," 2018.

[29] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17. New York, NY, USA: ACM, 2017, pp. 506–519. [Online]. Available: http://doi.acm.org/10.1145/3052973.3053009

[30] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *arXiv preprint arXiv:1806.01246*, 2018.

[31] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 2015.

[32] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Security and Privacy (SP), 2017 IEEE Symposium on*, 2017.

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[34] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for simplicity: The all convolutional net," *CoRR*, vol. abs/1412.6806, 2014. [Online]. Available: http://arxiv.org/abs/1412.6806

[35] D. Stefan, A. Russo, P. Buiras, A. Levy, J. C. Mitchell, and D. Maziéres, "Addressing covert termination and timing channels in concurrent information flow systems," *SIGPLAN Not.*, vol. 47, no. 9, pp. 201–214, Sep. 2012. [Online]. Available: http://doi.acm.org/10.1145/2398856.2364557

[36] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *CoRR*, vol. abs/1703.09039, 2017. [Online]. Available: http://arxiv.org/abs/1703.09039

[37] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *USENIX Security*, 2016.

[38] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," *arXiv preprint arXiv:1802.05351*, 2018.

[39] L. Wei, Y. Liu, B. Luo, Y. Li, and Q. Xu, "I know what you see: Power side-channel attack on convolutional neural network accelerators," *arXiv preprint arXiv:1803.05847*, 2018.

[40] M. Yan, C. Fletcher, and J. Torrellas, "Cache telepathy: Leveraging shared resource attacks to learn dnn architectures," *arXiv preprint arXiv:1808.04761*, 2018.

[41] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *CoRR*, vol. abs/1611.01578, 2016. [Online]. Available: http://arxiv.org/abs/1611.01578