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. However, Neural Networks face several security and privacy threats. This is particularly significant in the scenario where Cloud infrastructures deploy a service with Neural Network model at the back end. Here, an adversary can extract the Neural Network parameters, infer the regularization hyperparameter, identify if a data point was part of the training data, and generate effective transferable adversarial examples to evade classifiers. This paper shows how a Neural Network model is susceptible to timing side channel attack. In this paper, a black box Neural Network extraction attack is proposed by exploiting the timing side channels to infer the depth of the network. Although, constructing an equivalent architecture is a complex search problem, it is shown how Reinforcement Learning with knowledge distillation can effectively reduce the search space to infer a target model. The proposed approach has been tested with VGG architectures on CIFAR10 data set. It is observed that it is possible to reconstruct substitute models with test accuracy close to the target models and the proposed approach is scalable and independent of type of Neural Network architectures.**

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

I. Introduction

Of late, Neural Networks have been successfully employed to many diversified areas, namely computer vision, natural language processing and business intelligence[10]. Deep learning architectures have also been deployed for automated critical decision making in security applications like national critical infrastructures, malware and intrusion detection[56]. 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[2]. Banks and financial organisations rely on deep learning to process the massive financial data. Autonomous Driving has attracted several big automotive companies like Audi, Tesla and Waymo to invest billions of dollars into Deep Learning Research[4][3].

Developing and engineering Neural Networks for commercial services requires significant time, money and human effort which ranges from collection of massive data to fine-tuning the hyperparameters of the model to improve its performance. The commercial value of these models make them an important intellectual property for the company due to which the model attributes like number of layers, training algorithms, regularisation hyperparameter, etc. are kept confidential as a black box. These black box models do not reveal any information to the service users other than the output predictions for the input given by the users. This has been commercialised as a business model by several cloud service providers like Google, Amazon, Microsoft and BigML by deploying an end-to-end infrastructure for using Deep Neural Networks as a service[5]. Within Machine Learning as a Service(MLaaS) paradigm, trusted users submit training data to the service providers who spend significant resources to design and train high performing models which are deployed for public use on a pay-per-query basis. Despite its promise, the commercial value of the black box Neural Networks within MLaaS makes them attractive to adversaries to extract the model functionality and attributes, and circumvent the pay-per-query setting of the service. Given the architecture of the Neural Networks, an attacker can further mount various privacy and security attacks like model inversion[20][19][26] and membership inference[47][45] to infer the input and training data instances and generate more accurate adversarial examples to evade classifiers during test time[42]. These attacks violate the privacy of the sensitive data used for training the models and provide a way for attacker to evade security systems like malware and intrusion detection systems by forcing to make incorrect predictions.

A major security question in such a black box setting like MLaaS addressed in this paper is *whether a weak adversary in a black box setting can efficiently infer target Neural Network attributes by exploiting side channels with minimum number of queries?* In this work, a novel model extraction attack in a black box setting is proposed by exploiting timing side channels and efficiently reconstructing a substitute model architecture with functionality close to the target model using a constant number of queries.

Key Challenges in Model Extraction Attacks. Stealing a Neural Network architecture and its functionality is a challenging problem owing to the large number of hyperparameters which makes brute-force infeasible. The rapid growth of Neural Network design space has increased the complexity of architectures making the problem of black-box model extraction more challenging. In the black box setting as in MLaaS, 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 input-output relations to identify the decision boundary of the target model[52][41]. However, such attacks require significant computational resources and a huge time overhead

to search for the substitute model. For instance, given prior knowledge about the number of layers and type of layers, it still takes 40 GPU days to search for a simple 7 layer networks architecture[41]. Further, these attacks do no accommodate state of the art architectures with complex topologies and skip connections[30]. While extracting the model, these attacks require large number of queries which grow with the size of the architecture making the attack highly inefficient[39][52]. An alternative approach for model extraction is to exploit side channels like power consumption[9], memory access patterns[29][28] and cache side channel attacks[57][27] to infer target model attributes. While these attacks give fine grained information about the target model during execution, the threat model requires escalated attacker privileges and strong assumptions like physical access to hardware and shared resources for processes running on the server.

Proposed Approach. The objective of model extraction attack is to search for a substitute model with similar functionality as the target neural architecture. However, the search space for the substitute model is very complex and large due to the large number of hyperparameters in Neural Networks. To make the search tractable and efficient, the adversary has to reduce the search space by identifying some of the attributes of the target Neural Network in a black box setting using minimum queries. In a black box setting, a weak adversary can obtain the output prediction corresponding to a given input image. This paper shows the existence of timing side channels in the black box setting due to the dependence of the total execution time of the Neural Network on the total number of layers or depth of the network. From the total execution time, an adversary can infer the total number of layers(depth) of the Neural Network using a regressor trained on the data containing the variation of execution time with Neural Network depth. This additional side channel information obtained, namely the depth of the network, reduces the search space for finding the substitute model with functionality close to the target model.

To efficiently search for the optimal Neural Network, Reinforcement Learning based Neural Architecture Search is proposed which predicts an architecture using the reward computed from the performance of the previous proposed models[60]. Within the proposed Reinforcement Learning paradigm, knowledge distillation is used to train the substitute models, where the loss function is computed using the predicted labels of the substitute model and the target model instead of the true labels[25]. This ensures that the substitute model learns to mimic the predictions of the target model and hence, increasing the similarity of the two models. The proposed architecture search and reconstruction technique can be used with other attack approaches as well, like cache side channel attacks[27][57].

The proposed approach assumes a weak adversary with only black box access to the target Neural Network and requires a constant number of queries to infer the Neural Network depth which is independent of the architecture size. Further, the objective function of Reinforcement Learning approach maximises the test accuracy of the proposed Neural Network which ensures that the final substitute Neural Network is optimal.

Evaluation. To measure the success of the proposed model extraction attack, the performance of the regressor to correctly infer the depth of the Neural Network given the total execution time is shown. This is followed by the evaluation of the Reinforcement Learning based architecture search by comparing the test accuracy of the reconstructed model with the target model accuracy. The performance of different regressors, to infer the Neural Network depth has been assessed based on the R^2 score and the Mean Squared Error to select the regressor model which captures the maximum variance and accuracy. From the results, ensemble based regressors like random forrest and boosted decision trees outperform their linear counterparts: Ridge regression, Support Vector Machine(SVM) and decision trees. The experiments are performed using deep convolutional Neural Networks similar to VGG architectures[48]. The proposed Reinforcement Learning based architecture search technique can generate a model with test accuracy within 5% of the target model.

Main Contributions. The paper makes the following main contributions:

- Shows that Neural Networks are vulnerable to timing side channel attacks as Neural Network architectures with different depth have different execution time(Section V).
- Proposes 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).
- Proposes an efficient search technique to reconstruct an optimal substitute architecture using Reinforcement Learning and knowledge distillation to obtain functionality similar to 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 several security and privacy issues in adversarial settings[17]. A major security threat in Neural Networks is Adversarial examples, i.e, perturbed data instances that fool the classifier into misclassifying the image by either poisoning the training data or evading the

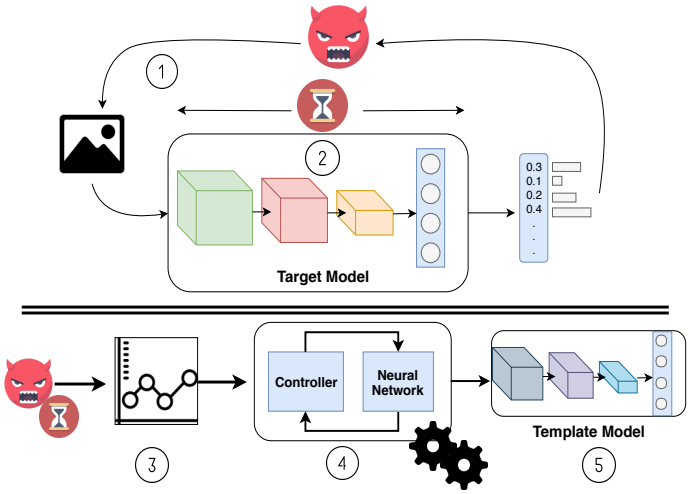

Fig. 1: Model Extraction using Timing Side Channels: 1) The adversary queries the target model by sending an input; 2) Adversary measures the execution time of the target model for the provided input; 3) The execution time is passed to the adversary's regressor model which predicts the depth of the target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

decision logic during inference[42]. For a Neural Network function $f_h(x;\theta)$ with input data point x and parameters θ, an adversary can violate the confidentiality and privacy of input data(x), training data, model parameters(θ) and the model computation($f_h(x;\theta)$). The privacy of input passed to the model can be violated using model inversion attack[19] by extracting the input when the adversary knows the output, parameters and gradients. Another major class of privacy attacks is Membership inference which violate the privacy of individual members of the training dataset by identifying whether a given data point is in the dataset or not[46][45]. Further, the computation of the Neural Networks leak information in the form of side channels which allow the adversary to extract model details or inputs[9][54]. 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[52][54][53]. Some of these attacks assume that the underlying Neural Network architecture is known to the adversary.

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[35]. 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[34] and OpenSSL[13][12].

III. Problem Statement

Model Extraction. Given a black box access to a target Neural Network f_{target}, the goal of the adversary is to search for a substitute Neural Network $f_{substitute} \in S$, where S is the search space for all possible Neural Network models with different hyperparameters, such that the functionality of f_{target} approximates $f_{substitute}$ using minimum possible queries. The metric used to measure the functionality the models, $f_{substitute}$ and f_{target}, is the test accuracy(R_{test}). In other words, the objective is to minimize the difference in test accuracy $R_{test} = \sum_{(x,y)\in D} d(f_{target}(x) - f_{substitute}(x))^2/|D|$ between the two models f_{target} and $f_{substitute}$ for inputs(x, y) sampled from the data(D), i.e, $(x, y) \overset{i.i.d}{\sim} D$ and d is the distance function between the two inputs.

Exploiting Timing Side Channels. To reduce the entropy of search space of possible Neural Network models and make the search more efficient, the adversary exploits the timing side channels to infer the number of layers from the execution time. For this, the adversary collects a dataset(D_A) with execution time(T) and Neural Network depth(K) for various models by changing the number of parameters. Formally, given the attacker dataset $D_A = \{(T_1, K_1), \cdots, (T_N, K_N)\}$ of i.i.d. random variables, for a given depth of the target Neural Network(k) from the total execution time(t), the adversary estimates the regression function $R(k) = E\{K|T = t\}$.

Model Search. The estimated depth is used to constraint the search space to S_k which is the set of all the Neural Network models of depth k. This allows the adversary to search for the substitute model $f_{substitute}$ in the search space S_k instead of search space S where $S_k \subset S$. However, the search space S_k is parameterised by kernel size, stride and number of filters which still make the search space large. To make the search space tractable, the adversary uses Reinforcement Learning paradigm where the accuracy of the target model is included in the objective function as part of the reward, to search for Neural Networks with higher test accuracy.

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. The proposed attack is in a black box setting where 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 is weak with no knowledge of the target model internals.

Hardware. The proposed attack is an inference time attack, where the trained target model has been deployed as a service to the users. During inference, CPUs and Neural Network accelerators are extensively used while GPUs are used for training Neural Network architectures[51]. Majority of machine learning Cloud service providers use CPUs[24]. The attack is evaluated using CPUs but the approach can be extended to other hardware accelerators as well. The adversary requires 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[6][7]. 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. The attack assumes a weak adversary with no knowledge about the training data and only knows the input-output dimensions and range of values they can take. There are two main approaches to reconstruct the training data for the substitute model: Iterative membership inference attacks and data reconstruction attacks. In case of membership inference attack, the adversary samples data points from the underlying data distribution $x \sim D$ which is passed as a query to the target model from which the adversary obtains the model output posterior $f(x)$. Given the output posterior of the model for the input, the attacker checks if the value of the maximum posterior is greater than a threshold($x \in D_T$) or not($x \notin D_T$)[45][59]. This is done iteratively by sampling data points and using membership inference attacks to reconstruct the training data used by the target Neural Network. In data reconstruction attack, an adversary uses a generative adversarial network to reconstruct training data samples from target model by finding the approximate training data distribution[44][21]. Either of the two approaches can be used to reconstruct the dataset and it is a one time operation done during the setup phase of the attack as described in Section VI.

V. Timing Channels In Neural Networks

There exists a direct relation between execution time of Neural Networks and their dependence on various hyperparameters for different Neural Network layers as shown in this section. This dependence of the execution time on the Neural Network hyperparameters allows an adversary to infer the architecture details using the total execution time which forms the basis of the attack. A typical Convolutional Neural Network has three types of layers based on operations: convolution layer, maxpool layer and fully connected layer and each of the layer has stride, kernel size and number of filter as the hyperparameters.

Convolution Layer. Convolution is a weighted sum operation which computes the multiplication of the parameters($\theta_{ij}^{(l)}$) of layer l and the input feature map(z_{ij}^{l-1}) and adds the results, $x_i^{(l)} = \sum_i \sum_j \theta_{ij}^{(l)} \times z_{ij}^{(l-1)}$. The execution time of convolution layers is proportional to the number of multiplications(shown in Figure 2(a)) which 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[1].

Maxpool Layer. Maxpool computes the max of the $k \times k$ region of preceding layer feature map where k is the size of the kernel. Hence, the computation of maxpool depends on the number of filters of the feature map and the kernel size(k). The execution time increases with increasing filter size and increasing kernel size as shown in Figure 2(b). For stride, there is an inverse relation between the execution time and the stride as shown in Figure 2(c). The output size decreases with increasing stride which results in a decrease in the execution time due to fewer number of multiplications.

Fully Connected Layer. Fully connected layer performs a matrix vector multiplication between the parameters of the Neural Network and the input image map. Formally, the matrix vector multiplication of parameters $\theta_{ij}^{(l)}$ for layer l with the activation of previous layer $z_j^{(l-1)}$ is given by $x_i^{(l)} = \sum_j \theta_{ij}^{(l)} \times z_j^{(l-1)}$. Given two fully connected layers of input nodes m and output nodes n, the execution time varies linearly with the total

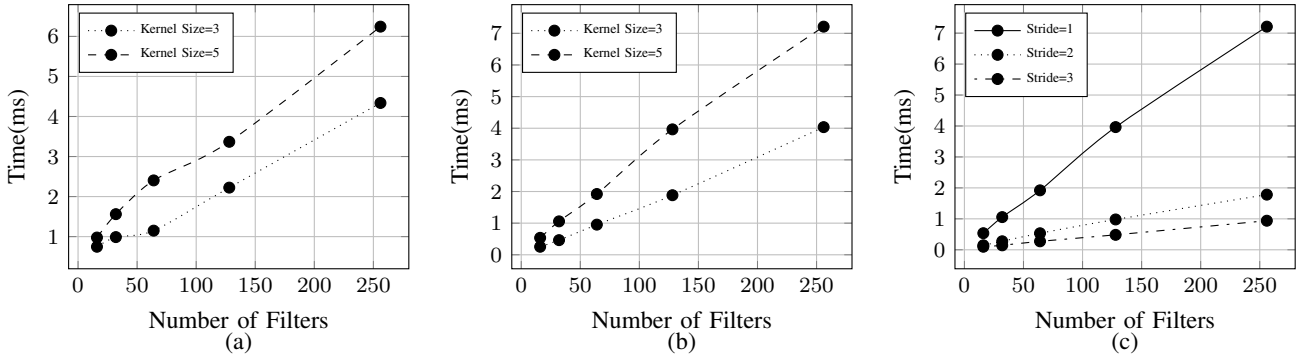

Fig. 2: (a) Convolution Layer: Total Execution Time increases linearly with kernel size and filter size; (b) Maxpool Layer: Total Exeuction Time increases linearly with kernel and filter size; (c) Maxpool Layer: Total Execution Time decreases with increase in stride.

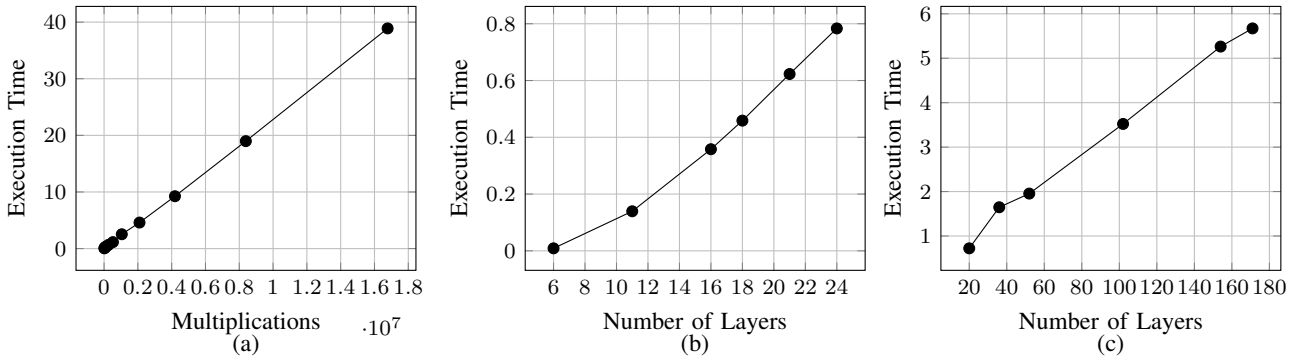

Fig. 3: (a) Fully Connected Layer: Total Execution time varies linearly with the total number of multiplications between the weights of the current layer and the input from the previous layer; (b) Simple Neural Networks Topology: Total Execution Time varies linearly with the number of layers of the neural network architecture; (c) Complex Neural Networks Topology: Total Execution Time varies linearly with the number of layers of the neural network with skip connections between different layers.

number of multiplications, $m \times n$ and the linear relation can be seen in Figure 3(a).

Variation with Depth. Neural Networks are embarrassingly parallel and all the computations in one layer can be executed in parallel to some extent which enables optimisations like model and data parallelism to improve the performance[15][37]. However, due to sequential computation of Neural Networks, the total execution time is the sum of the execution time of individual layers. In Figure 3(b) and Figure 3(c), the execution time increases linearly with the increase in the network architecture depth for both simple and complex topologies. The simple topologies of Neural Networks include LeNet, Alexnet and VGG architectures while the complex topologies include Resnet, Inception net and Densenet architectures which have skip connections between the layers. This linear relation between the number of layers and the total execution time forms the basis of the attack methodology to infer the depth of the Neural Network given the execution time.

Hardware Factors. 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. Since the hardware used for all the models is same, these factors effect the execution time of all the Neural Networks in the same manner.

VI. Attack Methodology

Side channels reveal only a part of the secret in the target system and identifying the rest of the secret can be modelled as a search problem[57]. The proposed attack is broadly divided into three phases as shown in Figure 1:

- **Setup Phase**: Adversary aggregates the dataset by measuring the execution time of multiple models with different hyperparameters, on a particular hardware, to be used in the actual attack. Further, the adversary reconstructs the training dataset using iterative membership inference[45] or dataset reconstruction attack[44]. This is a one time operation required to be performed prior to the attack.
- **Attack Phase**: Adversary queries the target Neural Network and measures the total execution time averaged over all queries. A regressor is trained on the attacker dataset which is used to infer the target Neural Network depth.
- **Reconstruction Phase**: Adversary searches for an optimal Neural Network with test accuracy close to that of the target Neural Network model within the reduced search space by making the depth of the Neural Network constant as inferred from the attack.

A. Setup Phase

The adversary during the setup phase reconstructs the training data with the prediction of target model($f_{target_i}(x)$) as labels instead of true labels for model distillation and creates a dataset containing the execution time of Neural Network architectures with varying hyperparameters. This a one time operation required prior to performing the attack.

Reconstructing Training Data. Since, the attack assumes a weak adversary with no knowledge about the training data, the adversary needs to identify and generate data samples used as the training dataset. For this, the adversary queries the target model with data samples x, and based on the model output posterior, determines if the data sample belongs to the training data D_T of the target Neural Network or not, by selecting a suitable threshold[45]. Given the knowledge of the data samples used as part target model training dataset, the adversary labels the input instances(x) with the predictions of the target model($f_{target}(x)$) instead of the true label(y). The adversary aggregates the training data $(x_i, f_{target}(x_i))$ as using soft target model predictions instead of hard labels ensures that the substitute model learns to mimic the functionality of the target model[25][8].

Creating Attack Model Timing 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 as shown in Table I. Formally, for different model architectures and depth(K), the attacker collects the cor-

responding execution time(T) to generate the attacker dataset $D_A = \{(T_1, K_1), \cdots, (T_N, K_N)\}$. This dataset is specific to a particular hardware and the collected data can then be used for stealing any model run on the same hardware.

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: Sample Attacker Dataset: Depth of the architecture and the number of parameters as input features for the regressor.

B. Attack Phase

Query. The adversary sends queries to the target model and computes the overall execution time averaged across all the queries. For each additional query of model extraction attacks proposed previously, a new bit of information is leaked which reduces the entropy of the target black box model. However, this requires a large number of queries due to the large number of parameters of deep Neural Network architectures. In the proposed attack, each of the query 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.

Regression. The average execution time measured from the target Neural Network(t) is used to estimate the depth of Neural Network(k) using regressor trained on the attack model dataset created during the setup phase, $R(k) = E\{K|T = t\}$. For current experiments, the attacker dataset for training the regressor model is created using 100 Neural Networks with different depth and parameters as shown in Figure 4.

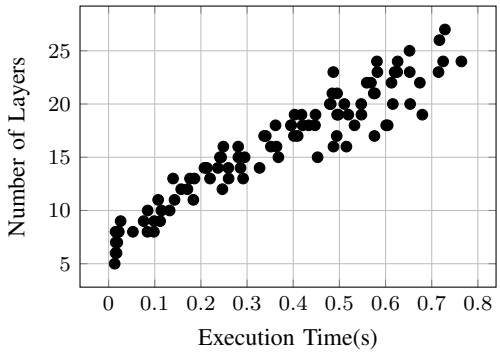

Fig. 4: Scatter Plot of the Adversary's Dataset: Execution time vs Number of layers. Each point in the plot is obtained by varying the parameters of the neural network layers and the number of layers in the neural network. The attacker fits a regressor to predict the depth of the neural network given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

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[60]. 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 then optimised based on performance of the predicted architecture using policy gradient method.

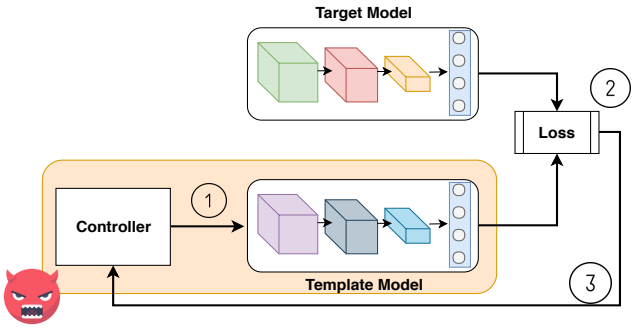

Fig. 5: Reinforcement Learning Based Architecture Search with Distillation: 1) The Recurrent Neural Network controller proposes a Neural Network architecture by selecting values from the state search space based on the the reward from the previous iteration; 2) The proposed model trains by using the predictions of the target model as the output predictions and mimics the behaviour of the target model[25][8]; 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

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 policy function maps the proposed architecture to a real number(accuracy) which is used to compute the reward for the RNN controller. The policy weights are optimised to ensure that newer architectures proposed by the RNN controller have a higher performance. After training the RNN controller for multiple iterations, the proposed substitute architecture is optimal, i.e, has the highest expected accuracy among all the models within the search space. The goal is to maximise the expected accuracy of sampled architecture give by,

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

Since the reward signal R is non-differentiable, we need to use a policy gradient method to iteratively update the parameters of the RNN controller θ_c. This work uses the $Reinforce$ algorithm which allows to directly update the policy weights using stochastic gradient ascent to optimise and update the policy [55]:

$$\bigtriangledown_{\theta_c} J(\theta_c) = \sum_{t=1}^{T} E_{P(a_{1:T};\theta_c)}\big[\bigtriangledown_{\theta_c} \log P(a_t|a_{(t-1):1};\theta_c)R\big]$$

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[25] between the substitute model predictions($y^{template}$) and target model(y^{target}) predictions instead of the true labels(y) for a given data point $x \in D$ and is given as,

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

For each iteration of training the substitute model, the model mimics the predictions of the target model which makes the substitute model similar to the target model in terms of performance evaluated using test accuracy(Figure 5).

VII. Evaluation

Data. For all the experiments, CIFAR10 dataset[36] is used which contains 60,000 32x32 colour images in 10 classes with around 6000 images per class and the classes are mutually exclusive. For training, 50,000 images are used while 10,000 images are used for testing.

Experiment Setup. The processor used for evaluation and experimentation is 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. Accurate timing of the Neural Network inference is done using $process\_time()$ 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. Regression

Ideally, on executing a Neural Network on hardware accelerator, the total execution time solely depends on the number of layers due to the sequential computation. Hence, varying the number of neurons or filters in a particular layer in a deep Neural Network should not change the overall execution time as individual computations within the same layer can be performed in parallel. This property of Neural Networks makes them embarrassingly parallel[51]. However, in practice, a variation in number of parameters within a particular layer shows a deviation in execution time due to inefficient parallelism as seen in Figure 4.

To address this, it is important to train a good regressor model which captures the maximum variance in the timing dataset. A good regressor model should be able to capture and explain the variance of the dataset through its predictions and generalize well over the data samples. The evaluation of different regressor models on the attacker dataset is done using R^2 score metric to measure the variance explained by the model and the mean squared error(MSE) to measure the error in estimating the depth of the network. Based on the results, 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 IV).

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 IV: 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.

The estimated depth of the ensemble regressors on the target deep Neural Networks from the corresponding execution time on VGG based target Neural Network architectures is shown in Table II. Since the output of the regressor is a continuous variable, the prediction of the regressor is rounded to the nearest larger integer. For all the three Neural Networks used for evaluation, the regressors estimate the correct depth from the total execution time.

B. Reconstruction using Reinforcement Learning

Once the adversary has estimated the depth of the Neural Network, the information is used to reduce the search space. Now, the adversary has to search for the optimal substitute Neural Network with test accuracy close to the target Neural Network.

The Reinforcement Learning based architecture search with knowledge distillation is evaluated by fixing the depth of the model architectures inferred using the regressor. The search is further constrained by specifying the convolutional layer

	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 II: Regression Model Predictions: The estimated depth of the neural network is rounded to nearest larger integer. Th depth is correctly estimated using ensemble regressors: Random Forest(RF) and Boosted Decision Trees(BDT). 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 III: Evaluation of Reconstruction using Reinforcement Learning based Architecture Search with knowledge distillation. The test accuracy of the reconstructed model is close to the original accuracy of the target model. The architecture of the model is in format: filters(kernel size) and GAP is the Global Average Pooling

parameter range for the kernel size(k) ϵ {3,5} and the number of filter(n_f) ϵ {32,64,128} which are commonly used hyper-parameters values used in all state of the art networks[48]. 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[49]. 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. For all the three model, the test accuracy of the substitute model generated is within 5% of the target model architecture as shown in Table III.

VIII. Mitigation

The main reason for the manifestation of timing side channels in Neural Networks is the sequential computation of layers which determine the total execution time. It is important to design Neural Networks resistant to timing side channels to prevent model extraction and possible defences are discussed in this section.

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 design Neural Networks without the dependency of the execution time on the number of layers and hyperparameters. Additional noise to the total execution time can be added in the form of latency by including dummy computations and layers. However, this results in a model security and utility tradeoff which is a concern in real time critical applications where the performance of the model is vital.

Adversarial Machine Learning. The second mechanism is a training phase defence where the attack and defence game can be viewed as an adversarial machine learning problem. The goal of the attacker is to fit the best possible curve or function to the attacker dataset for regression while the goal of the defender is to poison the dataset with wrong data instances such that the regressor makes wrong predictions. The defender injects adversarial examples with incorrect timing and depth values resulting in incorrect regressor estimation.

IX. Discussion

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.

Different dataset have images of different sizes and propoerties due to which the intermediate input feature maps require different number of multiplications. For instance, MNIST dataset has images of size 28 $\times$ 28 $\times$ 1, while CIFAR10 dataset has images of size 32 $\times$ 32 $\times$ 3 which results in difference in number of computations to be performed. Hence, 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.

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 the evaluation the attack is on a local model, this can be extended to remote setting like in MLaaS where the model is deployed on a Cloud server. The total round trip time in case of remote setting also includes some additional noise during propagation in the form of jitter as well as the time taken for network propagation[14]. A round trip time model for remote timing attacks is given below where the total response time(t_{res}) is a linear function of the scaled processing time(t_{proc}), network propagation time(t_{net}) and the jitter.

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

To extract the processing time of the Neural Network from the total round trip time, one has to estimate the additional network time and jitter and filter them from the round trip time.

Model Extraction Defences. Several defences have been proposed to mitigate the attacks that exploit the information from output predictions. Suppressing the information provided by output logits reduces the accuracy of the substitute model but degrades the utility[52]. Stateful defence mechanism to monitor and detect a variation in the input query distribution[31] or raise an alert if the information gained by an adversary exceeds a threshold[33]. Trusted hardware like Intel SGX can protect the confidentiality and integrity of the model by moving the model offline to the user's system[23]. For attacks that rely on memory access patterns, implementation using Oblivious RAM could help to hide the access pattern[50]. All the defences mentioned are proposed for attacks that use the output prediction scores but none of these approaches can help to mitigate timing side channels.

X. Related Work

Deep Neural Networks are known to be vulnerable to input images with imperceptible perturbations called adversarial examples[22]. Several applications that rely on deep Neural Networks have been effected by adversarial examples like text classification and sentiment analysis[38], computer vision based autonomous vehicles[11] and audio and speech processing[58]. Further, adversarial examples are transferable, i.e, an example which is misclassified in one model is likely to be misclassified by other models. This allows an adversary to exploit black box models using adversarial examples[42].

Inference attacks are based on the difference in the distribution of output predictions between members and non-members

of the training data[18]. The distinguishability between the two data distributions can be minimised as a min-max game between two Neural Networks to mitigate the training data leakage[40].

Cryptographic techniques like Homomorphic Encryption(HE) and Secure Multiparty Computation(SMC) have been explored for mitigating information leakage about the training data, inputs and parameters. In HE, the underlying Neural Network computations are performed between the encrypted parameters and the encrypted inputs without decrypting them[16]. On the other hand, SMC relies on multiple parties to collaboratively compute a function without the other party inferring the input or parameters[43]. While HE has a high computation overhead, SMC has a high communication overhead and the choice of using one the security primitives depends on the application requirements. However, a combination of both HE and SMPC for different operations in a Neural Network can result in significantly higher performance while preserving privacy[32].

XI. Conclusion

This paper shows 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, the timing channel vulnerability can be exploited to infer the depth of the Neural Network architecture. The evaluation of various regressors on the timing data shows that the ensemble based regressors perform better than their linear counterparts based on the R^2 score and Mean Score Error values. Further, the search problem of extracting a Neural Network architecture by exploiting side channels can be addressed efficiently using Reinforcement Learning with knowledge distillation. This approach can be used with other attacks like cache attacks and memory access pattern monitoring to accurately identify the substitute model close to the target model. The attack is evaluated on VGG like deep learning architectures and it is shown that a substitute model can be reconstructed within 5% of the test accuracy of the target Neural Network.

Acknowledgment

The authors would like to thank Virat Shejwalkar(University of Massachusetts 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] "Convolutional neural networks, cs231n stanford university," http://cs231n.github.io/convolutional-networks/, 2016.

[2] "Artificial intelligence help warfighters many fronts," http://mil-embedded.com/articles/artificial-intelligence-help-warfighters-many-fronts/, 2017.

[3] "Nvidia is powering the world's first level 3 self-driving production car," https://techcrunch.com/2017/07/12/nvidia-is-powering-the-worlds-first-level-3-self-driving-production-car/, 2017.

[4] "Tesla autopilot," https://www.tesla.com/autopilot, 2017.

[5] "Ai trends 2018: Machine learning as a service (mlaas)," https://blog.g2crowd.com/blog/trends/artificial-intelligence/2018-ai/machine-learning-service-mlaas/, 2018.

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

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

[8] 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

[9] 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.

[10] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[11] A. Boloor, X. He, C. D. Gill, Y. Vorobeychik, and X. Zhang, "Simple physical adversarial examples against end-to-end autonomous driving models," *CoRR*, vol. abs/1903.05157, 2019. [Online]. Available: http://arxiv.org/abs/1903.05157

[12] 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

[13] 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

[14] 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

[15] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. aurelio Ranzato, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1223–1231. [Online]. Available: http://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks.pdf

[16] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, pp. 201–210. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045390.3045413

[17] 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

[18] C. Dwork, A. Smith, T. Steinke, and J. Ullman, "Exposed! a survey of attacks on private data," *Annual Review of Statistics and Its Application (2017)*, 2017.

[19] 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.

[20] 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.

[21] S. Gambs, A. Gmati, and M. Hurfin, "Reconstruction attack through classifier analysis," in *Proceedings of the 26th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy*, ser. DBSec'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 274–281. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31540-4_21

[22] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] X. Hu, L. Liang, L. Deng, S. Li, X. Xie, Y. Ji, Y. Ding, C. Liu, T. Sherwood, and Y. Xie, "Neural network model extraction attacks in edge devices by hearing architectural hints," *CoRR*, vol. abs/1903.03916, 2019. [Online]. Available: http://arxiv.org/abs/1903.03916

[29] 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

[30] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[31] 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

[32] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 2018, pp. 1651–1669. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar

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

[34] 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

[35] 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

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

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[38] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," *CoRR*, vol. abs/1812.05271, 2018. [Online]. Available: http://arxiv.org/abs/1812.05271

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

[40] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 634–646. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243855

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

[42] 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

[43] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter, and F. Koushanfar, "Xonn: Xnor-based oblivious deep neural network inference," August 2019. [Online]. Available: https://www.microsoft.com/en-us/research/publication/xonn-xnor-based-oblivious-deep-neural-network-inference/

[44] A. Salem, A. Bhattacharyya, M. Backes, M. Fritz, and Y. Zhang, "Updates-leak: Data set inference and reconstruction attacks in online learning," *CoRR*, vol. abs/1904.01067, 2019. [Online]. Available: http://arxiv.org/abs/1904.01067

[45] 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.

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

[47] 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.

[48] 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

[49] 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

[50] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas, "Path oram: An extremely simple oblivious ram protocol," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 299–310. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516660

[51] 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

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

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

[54] 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.

[55] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992696

[56] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers: A case study on pdf malware classifiers," in *NDSS*, 2016.

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

[58] Z. Yang, B. Li, P. Chen, and D. Song, "Characterizing audio adversarial examples using temporal dependency," *CoRR*, vol. abs/1809.10875, 2018. [Online]. Available: http://arxiv.org/abs/1809.10875

[59] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*, 2018, pp. 268–282. [Online]. Available: https://doi.org/10.1109/CSF.2018.00027

[60] 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