YERBABUENA: Securing Deep Learning Inference Data via Enclave-based Ternary Model Partitioning

Zhongshu Gu Heqing Huang Jialong Zhang Dong Su IBM Research ByteDance ByteDance IBM Research

Hani Jamjoom Ankita Lamba Dimitrios Pendarakis Ian Molloy IBM Research IBM Research IBM Cognitive Systems IBM Research

ABSTRACT

Deploying and serving deep learning (DL) models in the public cloud facilitates the process to bootstrap artificial intelligence (AI) services. Yet, preserving the confidentiality of sensitive input data remains a concern to most service users. Accidental disclosures of user input data may breach increasingly stringent data protection regulations and inflict reputation damage. In this paper, we systematically investigate the life cycles of input data in deep learning image classification pipelines and further identify the potential places for information disclosures. Based on the discovered insights, we build YerbaBuena, an enclave-based model serving system to protect the confidentiality and integrity of user input data. To accommodate the performance and capacity limitations of today's enclave technology, we employ a Ternary Model Partitioning strategy that allows service users to securely partition their proprietary DL models on local machines. Therefore, we can (I) enclose sensitive computation in a secure enclave to mitigate input information disclosures and (II) delegate non-sensitive workloads to run out of enclave with hardware-assisted DL acceleration. Our comprehensive partitioning analysis and workload measurement demonstrate how users can automatically determine the optimal partitioning for their models, thus to maximize confidentiality guarantees with low performance costs.

KEYWORDS

Deep Learning; Data Confidentiality; Cloud Security

1 INTRODUCTION

It is convenient to leverage public cloud platforms to serve deep learning (DL) models. Service users can deploy their pre-trained proprietary DL models, in the form of deep neural networks (DNNs), to serve prediction requests. Customized prediction APIs can be exposed to be further integrated into mobile or desktop applications.

However, deploying and serving DL models in public clouds continues to pose security and privacy challenges. DL systems often process large amounts of user input data and depend on the latest innovations in hardware acceleration. Although users expect cloud providers to be trustworthy and dependable, they still remain cautious about the confidentiality of their input data. Accidental disclosures of private user data may violate increasingly stricter data protection regulations and lead to reputation damage.

To address the data confidentiality problem for deploying models in a third-party cloud, researchers proposed approaches based on cryptographic primitives[13, 32, 37] to enable privacy-preserving predictions. Although significant performance improvements have been made, these approaches are still not practical to be deployed in production environments. Distributed machine learning has also been proposed to protect data confidentiality[30, 36, 39, 43]. There, part of the deep learning functionality is delegated to the clients and private data are retained on client machines. However, these approaches introduced additional complexities to the client-side program logic; they also required more computing power on client devices, which are typically resource constrained.

As an alternative, Ohrimenko et al.[38] presented data-oblivious multi-party machine learning algorithms and leveraged Intel Software Guard Extensions (SGX) to make them privacy-preserving. Recent research efforts, such as Chiron[22] and Myelin[23], aim to enable SGX enclave integration for machine learning as a service (MLaaS). However, SGX-based computation is currently performance and memory constrained. Specifically, in-enclave workloads cannot exploit DL accelerators for matrix computation and floating-point arithmetic. It also has a limit of 128MB protected physical memory size for Intel Skylake CPUs¹. This, in turn, makes SGX inadequate to efficiently support running a deep and complex neural network *entirely* within an enclave.

To address the performance and capacity limitations of secure enclaves, it is essential to partition the deep learning workloads and delegate as much computation as possible to run out of enclaves. However, *insecure* model partitioning strategies may lead to information disclosures of the original inputs. Adversaries may leverage the out-of-enclave DL workloads, including both the partial model parameters and the computed outputs, to *reconstruct* or *reveal the properties* of the inputs.

In this paper, we systematically investigate the life cycles of input data in deep learning image classifiers and identify the potential places for information disclosures. Based on the discovered insights, we employ a Ternary Model Partitioning strategy to tightly control both ends, i.e., data *entrance* and *exit*, of deep learning inference pipelines to mitigate information leakages.

Our research prototype consists of two co-operative systems running respectively on users' local machines and in the public cloud. (I) On the user side, we build a Neural Network Assessment Framework to automate the process of evaluating and partitioning their proprietary DL models. Service users can leverage this framework to conduct local model partitioning before deploying the models to the cloud. (II) On the cloud side, we develop Yerbabuena, an enclave-based model serving system to host user-provisioned models and instantiate online image classification services. We leverage

1

 $^{^1\}mathrm{With}$ memory paging support for Linux SGX kernel driver, the size of enclave memory can be expanded with memory swapping. But swapping on the encrypted memory will significantly affect the performance.

Intel SGX to enforce isolated execution with memory access control and encryption protection. By enclosing the sensitive DL workload partitions into a secure enclave and enforcing authenticated encryption at data entrance, we can effectively prevent adversaries from exploiting Input Reconstruction [9, 33] or Model Interpretation [2, 42, 45, 46] techniques to divulge sensitive information of the original inputs. At the same time, we can still delegate non-sensitive workloads to run out of enclave to benefit from hardware-assisted DL acceleration.

We have conducted partitioning experiments on a spectrum of ImageNet-level DNNs with different network depths and architectural complexity, e.g., from the Darknet Reference Model (17 layers), Extraction Model (28 layers), to the deeper and more complex DenseNet Model (306 layers)[21]. Our comprehensive security analysis and workload measurement can be used as a guideline for service users to determine their own principle for partitioning DNNs, thus to achieve maximized security and performance guarantees.

To summarize, the major contributions are as follows:

- (1) A systematic study on the information disclosures of input data in deep learning image classification pipelines;
- (2) A Ternary Model Partitioning strategy derived from the information disclosure analysis;
- A Neural Network Assessment Framework to automate the local model partitioning process covering different DNN architectures;
- (4) An enclave-based deep learning model serving system to protect the confidentiality and integrity of user input data.

Roadmap. First, we briefly introduce the background knowledge in Section 2. Then, we motivate the research problem and derive the security principles for our system in Section 3. Section 4 discusses the threat model. In Section 5, we present the Ternary Model Partitioning strategy and the Neural Network Assessment Framework. We describe the model serving system design in Section 6, the implementation in Section 7, and the evaluation in Section 8. Section 9 discusses our future work and Section 10 surveys the related works. We conclude in Section 11.

2 BACKGROUND

In this section, we give a brief overview of two technologies that are closely related to our work: deep learning and Intel SGX. Deep learning is the key technology behind most state-of-the-art artificial intelligence (AI) services. We leverage Intel SGX as the Trusted Execution Environment (TEE) in YERBABUENA to protect the confidentiality of user input data in deep learning pipelines.

Deep Learning. Conventional machine learning methods are less efficient for processing raw data. Building a machine learning system required non-trivial domain expertise and engineering efforts to transform raw data and extract feature representations. Deep learning is an application of Artificial Neural Networks and is in the family of Representation Learning. Deep learning methods can process raw data directly and automatically discover the representations, with no human interventions. It can learn complex functions by composing multiple non-linear modules to transform representations from low-level raw inputs to high-level abstractions.

The major component of any deep learning inference system is a DNN, which has multiple hidden layers between the input and output layers. Each layer contains multiple neurons. Each connection between two neurons in a DNN has an associated weight. In supervised learning, the weights are learned in the training stage by maximizing the objective function of a neural network. Generally, stochastic gradient descent (SGD) with back-propagation is the most widely used approach for learning the weights of neural networks.

Mathematically, a feedforward DNN can be defined as a representation function F^* that maps an input \mathbf{x} to an output \mathbf{y} , i.e., $\mathbf{y} = F^*(\mathbf{x}; \theta)$. θ represents parameters that are learned during model training. F^* is composed of n (assuming the network has n layers) sub-functions F_i , where $i \in [1, n]$. F_i maps the input \mathbf{x}_i to the output \mathbf{y}_i on layer i. These sub-functions are connected in a chain. Thus, $\mathbf{y} = F^*(\mathbf{x}; \theta) = F_n F_{n-1} ... F_1(\mathbf{x})$. For a classification model, the final output \mathbf{y} is a probability vector generated by the softmax activation function. The top- \mathbf{k} entries are extracted from \mathbf{y} and are mapped with a set of user-defined class labels \mathbf{L} .

In addition, we briefly discuss Convolutional Neural Networks (ConvNets), which have achieved great success in computer vision tasks, including detection, segmentation, and recognition of objects in images and videos. ConvNet is a class of feedforward DNN that is designed for processing data in multi-dimensional arrays. Different from fully connected neural networks, the neurons in each layer of a ConvNet are arranged in three dimensions: width, height, and depth. Each neuron only connects to a receptive field of a previous layer. There are a few distinct types of layers for ConvNets, including convolutional layers and pooling layers. The parameters of a convolutional layer consist of a set of learnable filters. Each filter computes a dot product between their weights and a receptive field it connects to in the input volume, thus producing a 2-dimensional feature map. A convolutional layer usually uses Rectified Linear Unit (ReLU) as its activation function. The output volume of a convolutional layer is stacked feature maps along the depth dimension. The pooling layers perform non-linear downsampling operations to reduce the dimensionality of the feature maps. The convolutional and pooling layers in a ConvNet act as the feature extractors for the input. ConvNets use the same softmax activation function for classification, similar to a regular DNN.

Intel Software Guard Extensions. Intel SGX [35] offers a non-hierarchical protection model to support secure computation on untrusted remote servers. SGX includes a set of new instructions and memory protection mechanisms added to the Intel CPU architecture. A user-level application can instantiate a hardware-protected container, denoted as an enclave. An enclave resides in the application's address space and guarantees confidentiality and integrity of the code and data within it. Privileged software, such as the hypervisor, Basic Input/Output System (BIOS), System Management Mode (SMM), and operating system (OS), is not allowed to access or tamper with the code/data of an initialized enclave.

Here, we briefly discuss the life cycle of an SGX enclave. The detailed explanation and analysis of Intel SGX technology can be found in Costan and Devadas [7]. SGX sets aside a memory region, referred to as the Processor Reserved Memory (PRM). The CPU

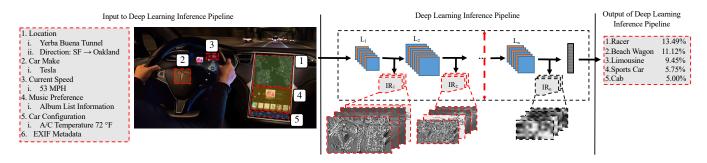


Figure 1: The Information Disclosure of Input Data in a Deep Learning Inference Pipeline

enforces memory access control of PRM to prevent any out-ofenclave memory accesses. Enclave Page Cache (EPC) stores the code and data of an enclave. The state of all pages of the EPC is maintained in the Enclave Page Cache Map (EPCM). A user-mode application can ask system software to create (ECREATE) an enclave. In the loading stage, the untrusted system software requests CPU to copy the code and data (EADD) into the EPC and assigns these memory pages to an enclave. After all memory pages are loaded, an enclave is initialized (EINIT) and the cryptographic hash of the enclave content is finalized. Before provisioning any secrets into an enclave, a client needs to initiate remote attestation [1] to verify the trustworthy level of hardware and the hash of the enclave's contents. An application can only enter its enclave via pre-defined function interfaces by executing a special instruction (EENTER). If enclave code needs to invoke system calls, it should exit (EEXIT) the enclave because ring 0 code is not allowed inside an enclave. To serve interrupts, page faults, or VM exits, the CPU needs to perform Asynchronous Enclave Exits (AEXs) to save the CPU state and transfer control to a pre-specified instruction out of the enclave. In SGX2 [34], Intel will support dynamic memory management inside an enclave, which allows run-time changes to enclave memory while maintaining the security properties.

3 MOTIVATION

Without proper confidentiality protection, input data to the cloud-based model serving systems might disclose user-specific sensitive and private information. We investigate the life cycles of input data instances in the deep learning image classification pipelines and intend to identify the potential places where information might leak. We give a motivating example in Figure 1 to empirically demonstrate such information disclosures. We feed this picture ² into a 1000-class image classification system with a ConvNet model trained on the ImageNet dataset. The output of the system is the top-5 prediction class scores of the picture.

Here we present the potential information disclosures that may occur in the deep learning inference pipelines. Thereafter, we describe the security principles our system needs to achieve to prevent each type of information disclosure respectively.

Information Disclosures in Original Inputs. A picture is worth a thousand words. If the inputs are fed into a DL system in unencrypted forms, adversaries can directly learn rich — sensitive and

private - information. For this specific example, it is obvious that this photo was taken when someone was driving a vehicle through a tunnel. Based on the map and GPS location [1] displayed on the touchscreen, we can infer that this car was in the Yerba Buena Tunnel, which is part of the San Francisco-Oakland Bay Bridge. We can also learn from the red arrow on the map that the driver was driving east from San Francisco to Oakland. The night mode of the map indicates the picture was taken after the sunset. In addition, the steering wheel emblem (2) reveals that this vehicle is a Tesla. The digital speedometer (3) on the dashboard tells us the speed at the time was 53 miles/hour. We can also obtain more personal information about the driver from the album list 4 of the media player and the car settings (5) on the touchscreen. Furthermore, if the picture's EXIF meta-data are not carefully eliminated, we may also retrieve the GPS coordinates, the original date and time when the picture was taken, the device type, and all camera configurations from the EXIF, which may disclose more private information about the user. Therefore, we define the first security principle to prevent information disclosures in the original inputs.

Security Principle I

Original inputs should not be revealed in the public cloud without proper protection.

Information Disclosures in Intermediate Representations. A

deep learning inference procedure extracts feature representations layer by layer. The process can be formulated as a composite transformation function that maps raw inputs to outputs. Each hidden layer performs its own transformation as a sub-function and generates an intermediate representation (IR) as an output. A transformation at each hidden layer helps converge the IRs towards the final outputs. Based on the research efforts in understanding the internal mechanisms of DNNs[44, 53, 54], for an image classification DNN, the shallow layers respond more to low-level photographic information (such as edges, corners, and contours, of the original inputs). In contrast, deep layers represent more abstract and class-specific information related to the final outputs.

Mapping these insights to the security domain, IRs computed out of shallow and deep layers may disclose different input information depending on adversaries' capability. We summarize three types of information disclosures in IRs as follows:

²Image source: https://www.tesla.com/software

(I) Explicit Disclosure. IRs computed out of shallow layers still bestow low-level photographic information of the original inputs. Thus, it is straightforward for humans, acting as adversaries, to understand the IRs by projecting them to pixel space. For example, in Figure 1, by examining the contents of the IR images out of shallow layers, adversaries are able to collect similar amount of information (though with limited information loss) if compared with viewing the original input directly. We consider that such IRs at shallow layers explicitly disclose the information of the original inputs. Therefore, we derive the second security principle intending to prevent the explicit information disclosure in IRs.

Security Principle II

IRs that are similar to the original inputs should not be revealed in the public cloud without proper protection.

By progressing towards deeper layers, the projected IR images in pixel space are not *directly* comprehensible by humans. The information of the original inputs is transformed into high-level features and is encoded via preceding layers as an encoder function. However, the information is still preserved within IRs and is crucial for final classification decisions. If adversaries can obtain *both IRs* and the preceding model parameters, they can still implicitly divulge sensitive information of the original inputs by exploiting Input Reconstruction [9, 33] or Model Interpretation [2, 42, 45, 46] techniques. We analyze both types of implicit disclosures respectively in detail as follows.

(II) Implicit Disclosure via Input Reconstruction. We consider that adversaries intend to reconstruct the original inputs from the IRs. The general input reconstruction problem in deep neural networks can be defined as follows: the representation function at layer *i* of a given DNN is $\Phi : \mathbb{R}^{w \times h \times c} \mapsto \mathbb{R}^d$, wherein $\mathbb{R}^{w \times h \times c}$ is the input space of a specific DNN and \mathbb{R}^d is the output space for the IRs. This process transforms the input **x** to Φ (**x**; θ _{Φ}) and θ _{Φ} represents the model parameters of the first *i* layers. Given an IR = $\Phi(\mathbf{x}; \theta_{\Phi})$, adversaries tend to compute an approximated inverse function ϕ^{-1} to generate $\tilde{\mathbf{x}} = \phi^{-1}(IR)$ that minimizes the distance between \mathbf{x} and $\tilde{\mathbf{x}}$. In practice, in order to derive ϕ^{-1} , adversaries need to have either white-box [33] or black-box [9] access to Φ. White-box access means that adversaries need to retrieve the model parameters θ_{Φ} , while black-box access means that adversaries need to query Φ and generate input-IR pairs. The pairs can be further utilized to approximate a surrogate inverse model ϕ^{-1} .

(III) Implicit Disclosure via Model Interpretation. Instead of reconstructing the original inputs, adversaries can also exploit the Model Interpretation techniques, e.g., Deconv [54], Guided Backpropagation [46], LRP [2], CAM [57], Grad-CAM [42], to interpret the numerical values in IRs. The attack strategy is to connect features in IRs to specific input attributions. Once such feature attribution or interpretation is established, it can be generalized to infer sensitive properties for future inputs. For example, some specific neurons may only be activated by a special attribution in the input, e.g., the steering wheel in Figure 1. If adversaries can confirm the connection between a steering wheel (as an input attribution) with a set of activated neurons, they can infer whether future inputs have

the "steering wheel" attribution by only checking the feature map activation.

Similar as the information disclosure via Input Reconstruction, interpreting the numerical values in IRs also requires *white-box* access to the model parameters. Thus, we can derive the third security principle for both cases above.

Security Principle III

Model parameters for generating IRs should not be revealed or allowed to be queried in the public cloud without proper protection.

Information Disclosures in Semantic Class Mapping. The last place that might disclose the input information is at the exit of the deep learning inference pipeline. The semantic class mapping of a classifier leaks categorical information of the input. From the top-5 prediction results (racer, beach wagon, limousine, sports car, or cab, with different probability scores), we can clearly infer that this picture is highly related to a vehicle, without viewing the original input. Therefore, IRs can be deciphered via forward propagation if models' semantic class labels are left unprotected. To prevent the information disclosure at semantic mapping, we derive the fourth security principle.

Security Principle IV

Semantic labels of models should not be revealed in the public cloud without proper protection.

In summary, information disclosures of user input data might happen at multiple stages in a deep learning inference pipeline, such as data entrance, feature extraction, and semantic class mapping. The design of our system should take all the derived security principles into consideration to prevent potential information leakage.

4 THREAT MODEL

In our threat model, the goal of adversaries is to uncover the contents of the user inputs submitted to deep learning image classification services. We consider that adversaries have the access the cloud machines that serve DL models. This can be achieved in multiple ways. For example, adversaries may exploit zero-day vulnerabilities to penetrate and compromise the system software of the cloud server. Insiders, such as cloud administrators, can also retrieve and leak data from the servers on purpose. Data can be in the form of files on disks or snapshots of physical memory. We assume that adversaries understand the format of the files stored on disks and they are able to locate and extract structured data (of their interest) from memory snapshots.

We assume that service users trust SGX-enabled processor packages and adversaries cannot break into the perimeters of CPU packages to track the code execution and data flow at the processor level. We do not intend to address the side channel attacks against Intel SGX in this paper. We expect that SGX firmware has been properly upgraded to patch recently disclosed micro-architectural vulnerabilities, e.g., Foreshadow[52] and SGXPectre[6], and the

4

in-enclave code has been examined to be resilient to side channel attacks.

We assume that the DL models to be deployed in the cloud are trained in a secured environment and the model parameters are not leaked to adversaries during model training. Users' devices that submit prediction requests are not compromised by adversaries. The keys of service users are properly protected and inaccessible by adversaries. Securing training process and protecting end-point user devices are out of the scope of this paper.

5 TERNARY MODEL PARTITIONING

Based on the four security principles derived in Section 3, we devise a Ternary Model Partitioning strategy to mitigate potential information disclosures in deep learning inference pipelines. The key idea is to partition each model into three functional components, i.e., a FrontNet, a BackNet, and a Semantic Class Mapping Unit.

Partitioning Mechanism. We choose a partitioning layer i where $i \in [1,n)$ in an n-layer DNN. The FrontNet includes layer $1 \to i$ and the BackNet includes the following layers. The function for FrontNet can be represented as $\Phi: \mathbb{R}^{w \times h \times c} \mapsto \mathbb{R}^d$. IR $= \Phi(\mathbf{x}; \theta_\Phi) = F_i F_{i-1} ... F_1(\mathbf{x})$ and its output IR is the intermediate representation computed out of a FrontNet. The function λ for a BackNet is $\mathbf{y} = \lambda(\mathrm{IR}; \theta_\lambda) = F_n F_{n-1} ... F_{i+1}(\mathrm{IR})$, in which IR is the input. The final output \mathbf{y} of a BackNet is a probability vector. The top-k entries are extracted from \mathbf{y} and are mapped with a set of user-defined class labels L. This label matching process is conducted in the Semantic Class Mapping Unit.

Use Case. Service users first partition their to-be-deployed models locally. They can designate that the FrontNet and Semantic Class Mapping Unit should be enclosed within a secure enclave in the cloud, whereas the BackNet can run out of the enclave to benefit from hardware acceleration. Thus, they can encrypt both the FrontNet submodel and labels with their secret keys and provision them to the cloud. The encrypted FrontNet and labels are only allowed to be loaded and decrypted after the enclave initialization and remote attestation. At runtime, service users can submit encrypted inputs to the online prediction service and the inputs are only allowed to be decrypted within the enclave. The semantic class mapping is also conducted within the enclave and the final classification results are sealed before returning back to users.

Revisiting Security Principles. We need to check whether the Ternary Model Partitioning strategy can satisfy all four security principles we defined in Section 3. Apparently, we satisfy Security Principle I because service users only submit encrypted inputs to the cloud. Enclave's memory encryption mechanism can prevent the information leakage of the original inputs. We can also achieve Security Principle III because the model parameters of the FrontNet, which generates the IRs that will be passed to the BackNet, are enclosed within secure enclaves. Adversaries cannot retrieve the FrontNet parameters as they are encrypted out of enclave. Furthermore, we adopt the Galois Counter Mode (AES-GCM) to authenticate the encrypted inputs. Thus, we can prevent adversaries from querying the FrontNet as a black-box. Therefore, we can effectively eliminate the attack surface for exploiting the Input Reconstruction and Model Interpretation methods. Our design also satisfies

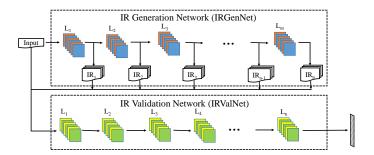


Figure 2: The Architecture of the Neural Network Assessment Framework

Security Principle IV because the Semantic Class Mapping Unit is conducted within the boundary of enclaves. Adversaries cannot decipher the semantic meanings of the probability vector as the labels have already been protected.

By now, the only *missing piece* is whether our design can fulfill the requirement of *Security Principle II*. This also determines how many layers we need to include in the FrontNet, which is model-specific. To translate the *Security Principle II* in the context of Ternary Model Partitioning, we need to find a partitioning layer satisfying the following property: the IRs generated after this layer are no longer similar to the original inputs. In order to address this problem, we develop a Neural Network Assessment Framework for service users to automatically find the optimal partitioning layers for different DL model architectures.

5.1 Neural Network Assessment Framework

The key intuition behind our Neural Network Assessment Framework is that if IRs retain similar visual contents as the original input, they will be classified into similar categories under the same oracle DNN. By measuring the similarity between the classification probability vectors, we can quantitatively determine whether a specific IR is similar to its original input.

In Figure 2, we present the dual-neural-network architecture of our Neural Network Assessment Framework. We submit an input x to the IR Generation Network (IRGenNet) and generate IR $_i$ where $i \in [1, n]$. We place the DL model that is to be deployed in the public cloud as the IRGenNet to generate IRs. Each IR $_i$ contains multiple feature maps after passing layer i (L_i). Then we project feature maps to IR images and submit them to the IR Validation Network (IRValNet), which acts as an oracle to inspect the IR images. The IRValNet can have different model architecture/weights from the IRGenNet. We use the DenseNet Model[21], which is known for its high accuracy in image classification, as the oracle IRValNet. The output of the IRValNet is a N-dimensional (N is the number of classes) probability vector with class scores.

We define dist[x, IR_i] to quantify the similarity between x and IR_i. We use Kullback-Leibler (KL) divergence to measure the similarity of classification probability distributions for both x and IR_i. At each layer i, we select the IR image having the minimum KL divergence (D_{KL}) with the input x: $\forall j \in [1, d(L_i)]$, where $d(L_i)$ is the depth of

the output tensor of layer i.

$$\begin{aligned} \operatorname{dist}[\mathbf{x}, \operatorname{IR}_{\mathbf{i}}] &= \min_{j} (D_{KL}(F^{*}(\mathbf{x}, \theta) \mid\mid F^{*}(\operatorname{IR}_{\mathbf{ij}}, \theta))) \\ &= \min_{j} (\sum_{k=1}^{N} F^{*}(\mathbf{x}, \theta)_{[k]} \log \frac{F^{*}(\mathbf{x}, \theta)_{[k]}}{F^{*}(\operatorname{IR}_{\mathbf{ij}}, \theta)_{[k]}}), \end{aligned} \tag{1}$$

where $F^*(\cdot,\theta)$ is the representation function of IRValNet. To determine the optimal partitioning layer for each neural network, we also compute $D_{KL}(F^*(\mathbf{x},\theta) \mid\mid \mu)$ where $\mu \sim \cup \{1,N\}$, the discrete uniform distribution of the probability vector, and N is the number of classes. This represents that adversaries have no prior knowledge of \mathbf{x} before obtaining IRs and consider that \mathbf{x} will be classified to any class with equal chance. We use it as the baseline for comparison. Thereafter, we compute $\forall i \in [1,n], \ \delta_i = \frac{\text{dist}[\mathbf{x}, |\mathbf{R}_i|]}{D_{KL}(F^*(\mathbf{x},\theta) \mid\mid \mu)}$. We can choose to partition at layer i, if and only if $\forall t \in [i,n], \ \delta_t > 1$. This iff condition is important because KL divergence scores may fluctuate, especially in the situation of skip connections. We will elaborate further about this interesting phenomenon in our DenseNet [21] case study in Section 8.2.

In summary, service users can leverage our Neural Network Assessment Framework to analyze their deep learning models before deployment and automatically determine the optimal Front-Net/BackNet partitioning. Thus, we can satisfy the *Security Principle II* to mitigate explicit information disclosure in IRs.

6 MODEL SERVING SYSTEM DESIGN

The service users can conduct the model partitioning on their local machines and provision the partitioned models to the cloud providers. In this section, we describe the system design of Yerbabuena in the cloud and demonstrate the workflow of establishing a model serving service.

6.1 Partitioned Deep Learning Inference

We adopt the TEE technology to enable isolated execution for security-sensitive computation in deep learning inference. We choose to use Intel SGX [35] as the TEE in our research prototype, but our approach in principle can also be generalized to other TEEs[4, 27]. With the protection of the memory access control mechanism and memory encryption engine (MEE) of SGX, all non-enclave accesses from privileged system software or other untrusted components of systems will be denied. In order to protect the secrecy of user inputs, we enforce enclaved execution on the following three stages in a deep learning inference pipeline:

Enclaved Entrance Control. The raw inputs contain all the sensitive information and users should *not* upload them to the cloud model serving system in plaintext. We allow service users to submit encrypted inputs. We establish an SGX enclave and load the encrypted inputs into this enclave. The enclave can attest to remote parties (i.e., the service users) that it is running on top of a trusted hardware platform with legitimate code/data. After finishing the remote attestation with the enclave, users can provision the keys for input decryption directly into the enclave via a secure communication channel. The user inputs will only be decrypted within the enclave and are invisible to the external computing stacks. Furthermore, we leverage AES-GCM to achieve authenticated encryption.

Thus, we can authenticate legitimate end users and verify the integrity of the user inputs.

Enclaved FrontNet Computation. We let service users partition their proprietary DL models before deploying them to the cloud. Each partitioned model consists of a FrontNet and a BackNet. We allow service users to submit encrypted FrontNets and plaintext BackNets to the model serving system. Same as the user inputs, we also require FrontNet to be decrypted only within the enclave with the key provisioned by the service user.

We leverage the enclave to keep the FrontNet model parameters in confidence. Otherwise, adversaries can exploit the exposed FrontNet via back-propagation to reconstruct original inputs from IRs [9, 33] or interpret the semantic meanings of IRs [2, 42, 45, 46]. In addition, as we authenticate legitimate user inputs through AES-GCM, we can further render Model Stealing Attacks[49] ineffective. For the adversaries who tend to treat the enclave as a black-box service and query to build a surrogate FrontNet model, they need to encrypt their inputs with the proper symmetric keys from the legitimate end users. Assuming that end users' keys are not leaked, we can deny serving these illegitimate requests that fail the integrity check and prevent the leakage of in-enclave FrontNet model information.

Enclaved Semantic Class Mapping. The class labels of a DNN are used to decipher the final probability vector output. However, this class mapping procedure leaks the object type information of the original inputs. In our design, we allow service users to submit encrypted class labels, which are decrypted only within the same enclave. We enforce the enclaved execution for the final class mapping function to mitigate the object information leakage of the original inputs. Furthermore, by controlling both ends of a deep learning inference pipeline, we can effectively mitigate service abusing attacks. The adversaries can no longer generate input-label pairs to infer sensitive attributions within the inputs.

6.2 Workflow

We summarize the workflow of Yerbabuena by explaining the steps in Figure 3 and the corresponding pseudo-code in Algorithm 1. In this case, the service user provides a partially-encrypted pretrained model (with FrontNet encrypted and BackNet in plaintext) and an encrypted label set to set up the online model serving system. Thereafter, the user can request the service with encrypted inputs.

- First, the user partitions a pre-trained proprietary model into a FrontNet and a BackNet. Then she can encrypt the FrontNet with her symmetric model key. Afterward, the user uploads the encrypted FrontNet and the plaintext BackNet to the model serving system in the cloud.
- ② Similarly, the user encrypts the class label set and uploads the encrypted label set to the cloud server.
- **❸** We initialize an Intel SGX enclave (INIT_SGX at line 19). After initialization, we securely copies the encrypted FrontNet and labels (SGX_LOAD_MDL_LBL at line 20) into the enclave.

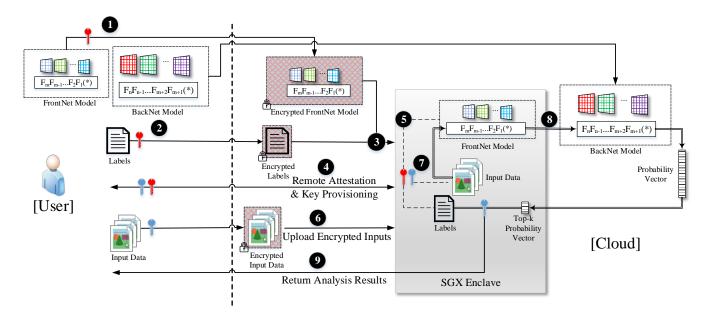


Figure 3: The Workflow of an Image Classification Service via YERBABUENA

Algorithm 1 Partitioned Deep Learning Inference

```
Input:
                fn_enc
                                                 ▶ Encrypted FrontNet sub-model
                                                             ▶ BackNet sub-model
                bn
                lbl enc
                                                         ▶ Encrypted model labels
                                                         ▶ Encrypted image input
                img_enc
                clt

    Client key provisioning server address

                                               > Number of returned predictions
 1: ########### Within SGX Enclave ###########
 2: function SGX LOAD ENC MDL LBL (fn enc, lbl enc, clt)
        tls \leftarrow \text{SGX\_ATTESTATION} (quote, clt)
 3:
        model\_key, img\_key \leftarrow sgx\_get\_keys (clt, tls)
 4:
        fn \leftarrow sgx \ verify \ dec \ (fn \ enc, \ model \ key)
 5
        lbl ← sgx_verify_dec (lbl_enc, model_key)
 6:
 7:
        frontnet ← SGX LOAD DNN (fn)
 8: function SGX INF ENC IMG (img enc)
        img \leftarrow sgx\_verify\_dec (img\_enc, img\_key)
10:
        ir \leftarrow SGX MODEL INF (frontnet, img)
11:
        return ir
12: function sgx_class_mapping(pv_k)
13:
        result \leftarrow sgx\_mapping(lbl, pv\_k)
14:
        enc\_result \leftarrow SGX\_ENC (img\_key, result)
15:
        return enc_result
16:
17: ############ Out of SGX Enclave ###########
18: function initialize_enclave (fn_enc, bn, lbl_enc, clt)
19:
        eid \leftarrow INIT SGX()
20:
        SGX_LOAD_ENC_MDL_LBL (eid, fn_enc, lbl_enc, clt)
21:
        backnet \leftarrow LOAD_DNN(bn)
22:
        return eid, backnet
23: function INF ENC IMG (eid, img enc, k, clt)
        ir \leftarrow \text{SGX\_INF\_ENC\_IMG} (eid, img_enc)
        pv \leftarrow \text{MODEL\_INF}(backnet, ir)
        enc\_result \leftarrow sgx\_class\_mapping (eid, top (pv, k))
```

● The end user and the SGX enclave need to perform remote attestation[1].³ The detailed description of the standard attestation protocol can be found in an example[25] provided by Intel.

After remote attestation, a secure Transport Layer Security (TLS) communication channel is created and the end user can provision symmetric keys (SGX_GET_KEYS at line 4) directly into the enclave in the cloud.

- **⑤** Inside the enclave, we verify the integrity of both the model and the labels by checking their AES-GCM authentication tags, and decrypt the FrontNet model (SGX_VERIFY_DEC at line 5) and the labels (SGX_VERIFY_DEC at line 6) with the provisioned symmetric key from the end user. Then we can build a deep neural network based on the FrontNet (SGX_LOAD_DNN at line 7) within the enclave and the BackNet (LOAD_DNN at line 21) out of the enclave.
- **6** We allow the users to upload their encrypted input data. Similarly, we copy the encrypted input into the enclave and decrypt them after authentication (SGX_VERIFY_DEC at line 9).
- Within the enclave, we pass the decrypted input into the FrontNet model (SGX MODEL INF at line 10) and generate the IR.
- **③** The generated IR is securely copied out of the enclave through a controlled channel of SGX. We pass the IR into the BackNet model and get the probability vector (MODEL_INF at line 25) for data prediction. This vector is an *N*-dimensional vector that represents a probability distribution over *N* different possible classes.
- **9** We extract the top-k entries and pass them into the enclave to find the mapping for their semantic class labels (SGX_CLASS_MAPPING at line 26). The mapped results are encrypted (SGX_ENC at line 14) with the prior symmetric key provisioned by the user. The encrypted result is returned back to the user.

7 IMPLEMENTATION

We build our research prototype Yerbabuena based on Darknet[40], which is an open source neural network implementation in C and CUDA. We also implement the Neural Network Assessment Framework to measure the information leakage of IRs at different layers. It can guide end users to determine, for each specific

³Due to the licensing procedure for registering SGX enclave code and the prerequisites for using the Intel Attestation Server (IAS), we currently skip this step and instantiate a TLS session directly between the end user and the enclave.

deep learning model, the optimal number of layers to include in a FrontNet and run within an enclave. In addition, we port the code from the mbedtls-SGX[55], which is an mbedtls-based implementation of the TLS protocol suite supporting Intel SGX, to enable TLS communication for key provisioning in YERBABUENA. In total, we add 23,333 SLOC in C and 474 SLOC in Python for the system development.

8 EVALUATION

In this section, we first conduct a qualitative security analysis targeting adversaries with different adversarial purposes. Thereafter, we leverage the Neural Network Assessment Framework to guide the partitioning of three ImageNet-based deep learning models. For each partitioned model, we also study the workload allocation based on low-level floating point operations (FLOPs) to see the proportion of computation that can be delegated to run out of enclaves. Finally, we measure the inference performance overhead by enclosing different numbers of layers in the enclave.

8.1 Security Analysis

Here we consider two hypothetical adversaries, \mathcal{A}_1 and \mathcal{A}_2 . They tend to uncover the contents of the original input \mathbf{x} after obtaining \mathbf{x} 's IRs out of the enclave. We consider both adversaries have no prior knowledge of input \mathbf{x} , but they have different attack strategies: \mathcal{A}_1 intends to reconstruct the original inputs from the exposed IRs and \mathcal{A}_2 tries to infer the attribution information belonging to the inputs through Model Interpretation methods.

Input Reconstruction Attacks (\mathcal{A}_1). Here we qualitatively review two representative Input Reconstruction techniques for deep neural networks, analyze the requirements or preconditions for these research works, and demonstrate that we can protect the data confidentiality of user inputs from powerful adversaries equipped with these techniques.

In Mahendran and Vedaldi[33], the authors proposed a gradient descent based approach to reconstructing original inputs by inverting the IRs. Following the formal description of the input reconstruction problem in Section 3, the objective of their approach is to minimize the loss function, which is the Euclid distance between $\Phi(\mathbf{x})$ and IR. Considering that Φ should not be uniquely invertible, they restrict the inversion by adding a regularizer to enforce natural image priors.

The research by Dosovitskiy and Brox[9] has a similar goal of inverting the IRs to reconstruct the original inputs. The major difference is that they do not manually define the natural image priors, but learn the priors implicitly and generate reconstructed images with an up-convolutional neural network. They involve supervised training to build the up-convolutional neural network, in which the input is the intermediate representation $\Phi(\mathbf{x})$ and the target is the input \mathbf{x} . Thus, \mathcal{A}_1 needs to collect the training pairs $\{\Phi(\mathbf{x}), \mathbf{x}\}$.

In our design, the FrontNet models are encrypted by users and are only allowed to be decrypted inside SGX enclaves. Assume \mathcal{A}_1 tends to use Mahendran and Vedaldi's approach[33] for reconstruction, the representation function Φ , which is equivalent to the FrontNet in our case, is not available in plaintext out of the enclave. Thus, we can prevent \mathcal{A}_1 from conducting optimization

to compute both ϕ^{-1} and $\tilde{\mathbf{x}}$. In addition, we can also prevent the adversaries from querying the online FrontNet as a black-box service. The reason is that we use AES-GCM to enable authenticated encryption. The enclave code can deny illegitimate requests, whose authentication tags cannot be verified correctly with users' symmetric keys. Therefore, \mathcal{A}_1 cannot generate training pairs by using Dosovitskiy and Brox's approach[9] to build the up-convolutional neural network. Without the up-convolutional neural network, \mathcal{A}_1 cannot reconstruct the original inputs either.

Input Attributions Inference Attacks (\mathcal{A}_2). All the Model Interpretation methods [2, 42, 46, 54, 57] require back-propagation through the DL model to create connections between inputs and IRs. In our design, we always keep the FrontNet within a secure enclave. Therefore, we can effectively eliminate the possibility for \mathcal{A}_2 to decipher the semantic meaning of these numerical numbers within the IRs. In addition, we also place the final class mapping within the enclave and only send the encrypted prediction results to end users. Thus, we also mitigate the information leakage of predicted classes for the original inputs too.

8.2 Partitioning and Workload Analysis

As an empirical study, we use our Neural Network Assessment Framework to determine the optimal partitioning layers for three ImageNet-level deep neural networks, i.e., Darknet Reference Model (17 layers), Extraction Model (28 layers), and DenseNet Model (306 layers)[21]. Based on the partitioning results, we further analyze the workload allocation after partitioning to measure the computation that can be out-sourced to benefit from DL-accelerated hardware.

Darknet Reference Model. This is a relatively small neural network for ImageNet classification. Its number of parameters is approximately 1/10 of AlexNet[28], while this model still retains the same prediction performance (top-1: 61.1% and top-5: 83.0%) compared to AlexNet.

To determine the optimal partitioning layer for this model, we compare the KL divergence ranges of all layers with the discrete uniform distribution. In Figure 7, we present the KL ranges (black columns) for the IR images of all layers (except the last three layers, i.e., average pooling, softmax, and cost layers, which do not generate IR images). For example, at layer 1 the minimum KL divergence is 3.08 and the maximum is 9.27. The lower the KL divergence score, the more similar it is to the original input. We also highlight the line for the KL divergence of the discrete uniform distribution with regard to the original input x. This KL divergence score is 3.00. We can find that after layer 4, the minimum KL divergence scores surpass the line of discrete uniform distribution's KL. This indicates that exposing IRs after layer 4 to adversaries does not reveal more information of the original input compared to any image classified to a uniform distribution. Thus end users can choose to partition the network at layer 4 and enclose them as a FrontNet to run within an enclave.

To demonstrate that the model assessment result is consistent with human perception, we select the IR images that have the minimum KL divergence scores at the first five layers and display them in Figure 4. For example, for layer 1, the IR image with the minimum KL divergence to the original input is generated by its 7th filter. We can find that, the IR images for this specific model



Figure 4: The List of IR Images with Minimum KL Divergence at Each Layer — Darknet Reference Model

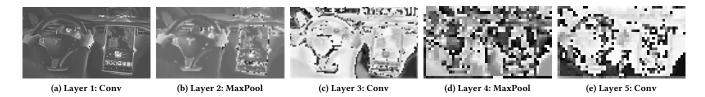


Figure 5: The List of IR Images with Minimum KL Divergence at Each Layer — Extraction Model



Figure 6: The List of IR Images with Minimum KL Divergence at Each Layer — DenseNet Model

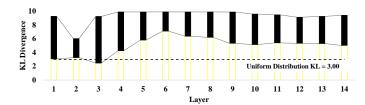


Figure 7: KL Divergence for Intermediate Representations of All Layers — Darknet Reference Model

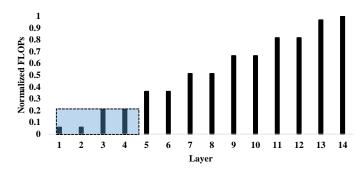


Figure 8: Normalized Cumulative FLOPs — Darknet Reference Model

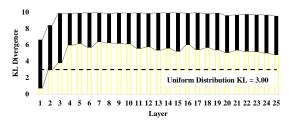


Figure 9: KL Divergence for Intermediate Representations of All Layers — Extraction Model

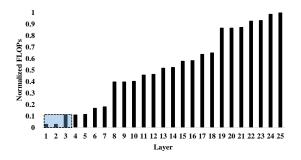


Figure 10: Normalized Cumulative FLOPs - Extraction Model

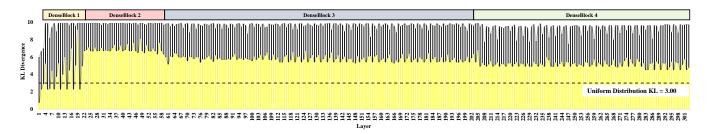


Figure 11: KL Divergence for Intermediate Representations of All Layers — DenseNet Model

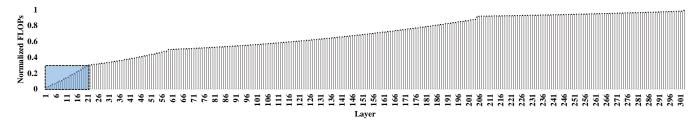


Figure 12: Normalized Cumulative FLOPs — DenseNet Model

retain less and less photographic information of the original inputs when progressing towards deeper layers. From the adversary's perspective, it becomes more difficult to reconstruct the original inputs if he can only obtain IRs generated by deeper layers running out of enclave.

In addition, we also calculate the workloads of FLOPs for each layer and display the normalized cumulative FLOPs in Figure 8. Based on the partitioning decision above, we can enclose the first four layers into an enclave (shown as the blue box), which comprises only 20.88% of the whole workload. The remaining 79.12% workload can still benefit from out-of-enclave hardware DL acceleration.

Extraction Model. Compared to the Darknet Reference Model, the Extraction Model is deeper and can achieve higher prediction accuracy (top-1: 72.5% and top-5: 90.0%).

We present the KL ranges for all layers in Figure 9. We can observe a similar phenomenon that after Layer 3, the KL divergence score ranges exceed the KL divergence of uniform distribution. Thus the safe partitioning point for this neural network can be at Layer 3. We also display the IR images with the minimum KL divergence scores for the first five layers in Figure 5. We can find that adversaries may still observe residual private information of the original input from the IR images at Layer 1 and 2. This is consistent with the quantitative analysis, which shows a subset of IR images before layer 3 have lower KL divergence scores than the uniform distribution.

As demonstrated in Figure 10, these three-layer enclaved execution only comprises 10.9% of the whole workload. The remaining 89.1% workload can be out-sourced to run out of enclaves.

DenseNet Model. In classical ConvNet architectures, each layer obtains the input only from its precedent layer. However, with the increase of network depth, it may lead to the *vanishing gradient problem*[3, 14]. To address this issue, researchers introduced short paths cross layers to make it practical to train very deep neural

networks. The authors of the DenseNet[21] introduced the neural network topology with DenseBlocks. Within each DenseBlock, each layer obtains inputs from *all* preceding layers and also transfers its own IRs to *all* subsequent layers. Between two adjacent DenseBlocks, it contains *transitional layers* to adjust the IR's size. We find that the information disclosure properties of such special model structures, i.e., DenseBlocks and densely connected layers, can be consistently quantified via KL divergence analysis.

We show the KL divergence scores in Figure 11. The DenseNet Model has four DenseBlocks. In each DenseBlock, the minimum KL divergence scores plummet regularly every two layers. The reason behind this phenomenon is that there exist route layers (after every two consecutive convolutional layers) that receive inputs from all preceding layers in the same DenseBlock. For example, the minimum KL divergence of layer 4 (convolutional layer) is 5.24, while at layer 5 (route layer) it drops to 2.27. Lower KL divergence scores indicate higher similarity of two probability distributions. We can obviously find that layer 5 (Figure 6b) preserves more information of the original input than layer 4 (Figure 6a). This result implies that we cannot simply partition in the middle of DenseBlock 1. The IRs generated by deep layers within DenseBlock 1 can still reveal original input's information.

However, there is no densely connected path that *crosses* different DenseBlocks. Although there still exist fluctuations of KL divergence scores in DenseBlock 2, the scores are significantly larger than layers in DenseBlock 1. In Figure 6, we also display the IR images with minimum KL divergence at all transitional layers (layer 21, 59, and 205) between different DenseBlocks. Based on the discrete uniform distribution KL divergence (3.00), the optimal partition point is at layer 21 (the last layer of DenseBlock 1).

Similarly, based on Figure 12, such 21-layer DNN computation accounts for 30.3% of the whole FLOP workload. We can assign the remaining 285 layers to build the BackNet and run it out of the enclave.

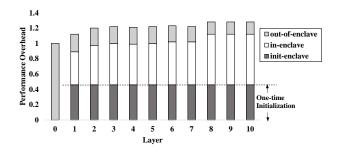


Figure 13: Performance Overhead of Running FrontNet in SGX Enclaves (compiled with -02)

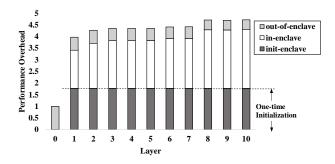


Figure 14: Performance Overhead of Running FrontNet in SGX Enclaves (compiled with -0fast)

8.3 Performance Evaluation

In the performance evaluation, we measure the performance overhead for different system settings and indicate the constraint factors in practice. By understanding the trade-off between security and performance, end users can determine the level of security protection they tend to achieve and the corresponding performance and usability cost they may have to pay. Our testbed is equipped with an Intel i7-6700 3.40GHz CPU with 8 cores, 16GB of RAM, and running Ubuntu Linux 16.04 with kernel version 4.4.0.

We measure the inference performance of YerbaBuena by passing testing samples through the Extraction Model. In the base case, we load the whole neural network without using SGX and obtain the average time for predicting these unencrypted images. To compare with the base case, we partition the network, load multiple layers as the FrontNet inside an SGX enclave and the following layers out of the enclave, and obtain the same performance metrics. We need to emphasize that both images and the FrontNet models are encrypted in these cases and are decrypted at runtime inside SGX enclaves. Due to the SGX memory limitation, we load up to 10 layers of the Extraction Model into the enclave. We compiled YerbaBuena with both gcc optimization level -02 and -0fast (with -03 and -ffast-math enabled) and present the normalized performance results in Figure 13 and 14 respectively. For each individual input, we include the one-time overhead of enclave initialization in the performance measurement. We distinguish the performance overhead contributed by enclave initialization, in-enclave computation, and out-of-enclave computation with bars of different colors. Layer 0 is

the base case with unencrypted inputs and all layers run out of the SGX enclave.

For optimization level at -02, we observe the performance overhead increase from 12% for running one layer inside an enclave to 28% for ten layers. Initializations of enclaves contribute to the most significant portion of the additional performance overhead. However, once the enclave is initialized, we observe that an inference task within an SGX enclave has even lower performance overhead as running out of the enclave. This is mainly due to the characteristic of deep neural network computation, which is computing-intensive and can benefit a lot from using the CPU cache and decrease the rate to read and write the encrypted memory, which is considered to be expensive in SGX. In the cloud scenario, we do not need to initialize and tear down an enclave for each service request, but can run one enclave as a long-time service to serve all client requests. For optimization level at -Ofast, we observe that the absolute time for enclave initialization is at the same level as in Figure 13. The in-enclave FrontNet computation causes 1.64x - 2.54x overhead compared to the base case. The BackNet still conduct inference computation at the same speed as the base case. We speculate that the slow down inside the enclave is due to the ineffective -ffast-math flag for floating arithmetic acceleration. We expect that in the future Intel will release optimized math library within SGX enclave to further reduce the floating arithmetic overhead.

Compared to cryptographic schemes based approaches [13, 32, 37] and running whole neural network within a single enclave [38], the performance overhead of Yerbabuena makes online deep learning inference feasible and adaptable for production-level large-scale deep neural networks. The out-of-enclave BackNet computation can still benefit from hardware/compiler acceleration and we grant end users the freedom to adjust network partitioning strategy to satisfy their specific security and performance requirements.

9 DISCUSSION

Applicable Deep Learning Models. In addition to the widely deployed deep learning models used for classification tasks, there also exist some special information-preserving neural networks designed for specific machine learning purposes. One representative case is the AutoEncoder [20], which is used for efficient encoding, dimension reduction, and learning generative models. AutoEncoder networks are trained to minimize the reconstruction errors between inputs and outputs. Thus, an AutoEncoder's outputs may contain similar sensitive information as its inputs. Our system can be naturally extended to support confidentiality protection of AutoEncoder's inputs. We need to partition each AutoEncoder neural network into three sub-models, i.e., FrontNet, MiddleNet, and Back-Net. Then we can enclose both the FrontNet and the BackNet into an isolated enclave, as both ends of the AutoEncoder may generate IRs similar to the original inputs. End users can use our Neural Network Assessment Framework to determine the number of layers assigned respectively for the FrontNet and the BackNet.

Compression of Deep Learning Models. There is also a line of interesting research works on reducing the storage and computation of deep neural networks without decreasing model accuracy. Denton et al.[8] applied singular value decomposition (SVD) to reduce

ConvNet computation. Han et al.[18] pruned redundant connections for models trained with ImageNet dataset without accuracy loss, e.g., reduce the total number of parameters of AlexNet by a factor of 9x and VGG-16 by a factor of 13x. In Deep Compression [17], the authors further combined pruning, trained quantization, and Huffman coding to reduce the storage of neural networks by 35x to 49x without affecting the accuracy. SqueezeNet[24] can achieve AlexNet-level accuracy with 50x fewer parameters with 0.5 MB model size. These works shed light on deploying production-level deep neural networks on embedded systems and mobile computing platforms. Our system can greatly benefit from their research outcomes to reduce computation and memory footprints within SGX enclaves, and further applying confidentiality protection to deeper neural networks.

10 RELATED WORK

In this section we list the research efforts that are closely related to our work and highlight our unique contributions compared to these works.

Cryptographic Schemes Based Machine Learning. Most of the existing privacy-preserving machine learning solutions are based on cryptographic schemes, such as secure multi-party computation (SMC), fully homomorphic encryptions (FHE)[12], etc. Solutions based on SMC protect intermediate results of the computation when multiple parties perform collaborative machine learning on their private inputs. SMC has been used for several fundamental machine learning tasks [10, 26, 31, 37, 50, 51]. Besides these protocol-based solutions, recently researchers also propose to leverage cryptographic primitives to perform deep learning inference. Gilad-Bachrach et al.[13] proposed CryptoNets, a neural network model that makes predictions on data encrypted by FHE schemes. This approach protects the privacy of each individual input in terms of confidentiality. MiniONN[32] is an approach that transforms existing neural networks to an oblivious neural network that supports privacypreserving predictions.

Considering the significant performance overhead of using cryptographic schemes, we propose to leverage Intel SGX technology to securely execute deep neural network computation on the cloud side. Hence we can protect the confidentiality of user inputs for predictions and can defend against input reconstruction and input attribution inference attacks.

Distributed Deep Learning. Shokri and Shmatikov[43] designed a distributed privacy-preserving deep learning protocol by sharing selective parameters and gradients for training deep neural network in a differentially private way. Ossia et al.[39] proposed a distributed machine learning architecture to protect the user's input privacy. Their framework consists of a feature extractor on the mobile client side and a classifier on the server side. The server side performs inference task on the dimension-reduced extracted features from the mobile client. PrivyNet[30] is a splitting model deep learning training approach. They reused layers of pre-trained models for feature extraction on local machines and train the cloud neural network for the learning tasks based on the feature representations generated by the local neural network.

Different from their works, our approach leverages TEEs in the cloud directly to guarantee the confidentiality of user inputs, class

labels, and the user-provisioned models. Thus, we significantly simplify the client's logic and relieve client devices, which are supposed to have limited computing capacity and power usage restriction, from heavyweight neural network computation. In addition, our approach does not involve transferring intermediate representations through the network, thus eliminating the additional performance overhead for dimension reduction or data compression.

SGX Applications. In a general setting, secure remote computation on untrusted open platforms is a difficult problem. Intel developed SGX technology to tackle this problem by leveraging the trusted hardware on remote machines. A set of new instructions and memory access control have been added since the release of the Intel 6th generation *Skylake* architecture. The general introduction of SGX can be found in [35]. We can also find the technical details about the SGX attestation and sealing mechanisms in [1], dynamic memory allocation of SGX2 in [34], and Memory Encryption Engine in [15]. We have also observed numerous innovative applications leveraging security mechanisms of SGX from academia to industry in recent years to address different research problems.

We have discussed and compared with recent research efforts[22, 23, 38] of employing SGX for privacy-preserving machine learning tasks. To address the performance and capacity limitations of secure enclaves, concurrent research by Tramèr and Boneh[48] explored a similar "workload partitioning" methodology. They proposed to outsource linear layers' computation of DNNs to out-of-enclave GPUs. Different from their approach, we employ a vertical layerwise partitioning strategy to exploit the intrinsic structural properties of deep learning models. These two partitioning strategies are not in conflict and can be deployed together to further reduce the performance overhead of enclaved inference computation with confidentiality protection. MLCapsule [19] is another interesting offline model deployment approach that executes model locally on the client's machine and protects the models' secrecy with SGX enclaves. Thus, they explore how to securely deploy server's machine learning workload to the client, while we investigate in a reverse direction on how to out-source client's computation to the server.

SGX has also been used for efficient two-party secure function evaluation[16], private membership test[47], trustworthy remote entity[29]. SGX technology is also widely researched in cloud scenarios. VC3[41] ran distributed MapReduce computation within SGX enclaves on the cloud to keep the confidentiality of user's code and data. Opaque[56] was a distributed data analytics platform introducing SGX-enabled oblivious relational operators to mask data access patterns. SecureKeeper[5] provided an SGX-enhanced ZooKeeper to protect the sensitive application data. HardIDX[11] leveraged SGX to help search over encrypted data. Different from the goals of these works, our work intends to protect the user input confidentiality from being exposed in public cloud environments.

11 CONCLUSION

We systematically study the information disclosures in deep learning image classifiers and devise a Ternary Model Partitioning strategy to mitigate input data exposure in deep learning inference pipelines. To further help users determine the optimal partitioning layers for their pre-trained models, we design a Neural Network

Assessment Framework to automatically quantify layer-wise information leakage for different neural network architectures. We have also built Yerbabuena, an enclave-based model serving system running on cloud infrastructures, to protect the confidentiality of both user inputs along with user-specified deep neural network layers and semantic class mapping. Security analysis demonstrates our system can effectively neutralize input reconstruction and attribution inference attacks, thus eliminating channels for adversaries to reconstruct user inputs or reveal sensitive input properties.

REFERENCES

- Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. 2013. Innovative technology for CPU based attestation and sealing. In Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy.
- [2] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one* 10, 7 (2015), e0130140.
- [3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural* networks 5, 2 (1994), 157–166.
- [4] Rick Boivie and Peter Williams. 2013. SecureBlue++: CPU Support for Secure Executables. Technical Report. Research report, IBM.
- [5] Stefan Brenner, Colin Wulf, David Goltzsche, Nico Weichbrodt, Matthias Lorenz, Christof Fetzer, Peter R. Pietzuch, and Rüdiger Kapitza. 2016. SecureKeeper: Confidential ZooKeeper using Intel SGX. In Proceedings of the 17th International Middleware Conference.
- [6] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H Lai. 2018. SGXPECTRE Attacks: Leaking Enclave Secrets via Speculative Execution. arXiv preprint arXiv:1802.09085 (2018).
- [7] Victor Costan and Srinivas Devadas. 2016. Intel SGX Explained. IACR Cryptology ePrint Archive (2016), 86.
- [8] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. In Advances in Neural Information Processing Systems. 1269–1277.
- [9] Alexey Dosovitskiy and Thomas Brox. 2016. Inverting Visual Representations with Convolutional Networks. In IEEE Conference on Computer Vision and Pattern Recognition.
- [10] Wenliang Du, Yunghsiang S Han, and Shigang Chen. 2004. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In Proceedings of the 2004 SIAM international conference on data mining.
- [11] Benny Fuhry, Raad Bahmani, Ferdinand Brasser, Florian Hahn, Florian Kerschbaum, and Ahmad-Reza Sadeghi. 2017. HardIDX: Practical and Secure Index with SGX. In Data and Applications Security and Privacy XXXI: 31st Annual IFIP WG 11.3 Conference.
- [12] Craig Gentry. 2009. A fully homomorphic encryption scheme. Stanford University.
- [13] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. 2016. CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. In Proceedings of the 33nd International Conference on Machine Learning.
- [14] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics. 249–256.
- [15] Shay Gueron. 2016. A Memory Encryption Engine Suitable for General Purpose Processors. IACR Cryptology ePrint Archive (2016). http://eprint.iacr.org/2016/ 2004
- [16] Debayan Gupta, Benjamin Mood, Joan Feigenbaum, Kevin Butler, and Patrick Traynor. 2016. Using intel software guard extensions for efficient two-party secure function evaluation. In *International Conference on Financial Cryptography* and Data Security.
- [17] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015).
- [18] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In Advances in Neural Information Processing Systems. 1135–1143.
- [19] Lucjan Hanzlik, Yang Zhang, Kathrin Grosse, Ahmed Salem, Max Augustin, Michael Backes, and Mario Fritz. 2018. Mlcapsule: Guarded offline deployment of machine learning as a service. arXiv preprint arXiv:1808.00590 (2018).
- [20] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. science 313, 5786 (2006), 504–507.
- [21] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In Proceedings of the IEEE Conference

- on Computer Vision and Pattern Recognition.
- [22] Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett Witchel. 2018. Chiron: Privacy-preserving Machine Learning as a Service. arXiv preprint arXiv:1803.05961 (2018).
- [23] Nick Hynes, Raymond Cheng, and Dawn Song. 2018. Efficient Deep Learning on Multi-Source Private Data. arXiv preprint arXiv:1807.06689 (2018).
- [24] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size. arXiv preprint arXiv:1602.07360 (2016).</p>
- [25] Intel 2018. Intel Software Guard Extensions Remote Attestation End-to-End Example. https://software.intel.com/en-us/articles/code-sample-intel-softwareguard-extensions-remote-attestation-end-to-end-example.
- [26] Geetha Jagannathan and Rebecca N Wright. 2005. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining.
- [27] David Kaplan, Jeremy Powell, and Tom Woller. 2016. AMD Memory Encryption. Technical Report. White paper, AMD.
- [28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems.
- [29] Kubilay Ahmet Küçük, Andrew Paverd, Andrew Martin, N Asokan, Andrew Simpson, and Robin Ankele. 2016. Exploring the use of Intel SGX for Secure Many-Party Applications. In Proceedings of the 1st Workshop on System Software for Trusted Execution.
- [30] Meng Li, Liangzhen Lai, Naveen Suda, Vikas Chandra, and David Z Pan. 2017. PrivyNet: A Flexible Framework for Privacy-Preserving Deep Neural Network Training with A Fine-Grained Privacy Control. arXiv preprint arXiv:1709.06161 (2017).
- [31] Yehuda Lindell and Benny Pinkas. 2000. Privacy preserving data mining. In Advances in Cryptology-CRYPTO 2000.
- [32] Jian Liu, Mika Juuti, Yao Lu, and N Asokan. 2017. Oblivious Neural Network Predictions via MiniONN Transformations. In ACM Conference on Computer and Communications Security (CCS).
- [33] Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In IEEE Conference on Computer Vision and Pattern Recognition.
- [34] Frank McKeen, Ilya Alexandrovich, Ittai Anati, Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, and Carlos Rozas. 2016. Intel Software Guard Extensions (Intel SGX) Support for Dynamic Memory Management Inside an Enclave. In Proceedings of the Hardware and Architectural Support for Security and Privacy 2016 (HASP 2016). ACM, New York, NY, USA, Article 10, 9 pages. https://doi.org/ 10.1145/2948618.2954331
- [35] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar. 2013. Innovative Instructions and Software Model for Isolated Execution. In Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy.
- [36] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. 2016. Communication-efficient learning of deep networks from decentralized data. arXiv preprint arXiv:1602.05629 (2016).
- [37] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning.. In 38th IEEE Symposium on Security and Privacy.
- [38] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. 2016. Oblivious Multi-Party Machine Learning on Trusted Processors.. In USENIX Security Symposium.
- [39] Seyed Ali Ossia, Ali Shahin Shamsabadi, Ali Taheri, Hamid R Rabiee, Nic Lane, and Hamed Haddadi. 2017. A Hybrid Deep Learning Architecture for Privacy-Preserving Mobile Analytics. arXiv preprint arXiv:1703.02952 (2017).
- [40] Joseph Redmon. 2013–2016. Darknet: Open Source Neural Networks in C. https://pjreddie.com/darknet/.
- [41] Felix Schuster, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. 2015. VC3: Trustworthy data analytics in the cloud using SGX. In Security and Privacy (SP), 2015 IEEE Symposium on.
- [42] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra, et al. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization.. In ICCV. 618–626.
- [43] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. ACM.
- [44] Ravid Shwartz-Ziv and Naftali Tishby. 2017. Opening the black box of deep neural networks via information. arXiv preprint arXiv:1703.00810 (2017).
- [45] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013).
- [46] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2014. Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806 (2014).

- [47] Sandeep Tamrakar, Jian Liu, Andrew Paverd, Jan-Erik Ekberg, Benny Pinkas, and N Asokan. 2017. The Circle Game: Scalable Private Membership Test Using Trusted Hardware. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security.
- [48] Florian Tramer and Dan Boneh. 2018. Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware. arXiv preprint arXiv:1806.03287 (2018).
- [49] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing Machine Learning Models via Prediction APIs.. In USENIX Security Symposium.
- [50] Jaideep Vaidya and Chris Clifton. 2002. Privacy preserving association rule mining in vertically partitioned data. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.
- [51] Jaideep Vaidya, Murat Kantarcioğlu, and Chris Clifton. 2008. Privacy-preserving naive bayes classification. The VLDB JournalâĂŤThe International Journal on Very Large Data Bases (2008).
- [52] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx.

- 2018. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In *Proceedings of the 27th USENIX Security Symposium*. USENIX Association.
- [53] Jason Yosinski, Jeff Clune, Anh Mai Nguyen, Thomas J. Fuchs, and Hod Lipson. 2015. Understanding Neural Networks Through Deep Visualization. CoRR abs/1506.06579 (2015). http://arxiv.org/abs/1506.06579
- [54] Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and Understanding Convolutional Networks. In Computer Vision ECCV 2014 13th European Conference.
- [55] Fan Zhang. 2018. TLS for SGX: a port of mbedtls. https://github.com/bldck5un/mbedtls-SGX.
- [56] Wenting Zheng, Ankur Dave, Jethro G Beekman, Raluca Ada Popa, Joseph E Gonzalez, and Ion Stoica. 2017. Opaque: An Oblivious and Encrypted Distributed Analytics Platform. In 14th USENIX Symposium on Networked Systems Design and Implementation.
- [57] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2921–2929.