Branchy-TEE: Deep Learning Security Inference Acceleration Using Trusted Execution Environment

Yulong Wang, Kai Deng, Fanzhi Meng, Zhi Chen, Mingyong Yin*
Institute of Computer Application
China Academy of Engineering Physics
Mianyang, China
{wangyulong, dengkai, mengfz, chenzhi, yinmy}@caep.cn

Run Yang
School of Cyberspace Science
Harbin Institute of Technology
Harbin, China
runyoung@hit.edu.cn

***Abstract*—Deep Learning as a Service (DLaaS) has become a remarkable trend in modern data-driven online services. Both data holders and service providers need to build on trust in third-party cloud infrastructure platforms. However, once the trust is broken, data holders' sensitive data and service providers' intellectual property rights will face significant security and privacy risks. In this paper, we propose a secure and efficient inference framework for deep learning in untrustworthy cloud platforms, termed Branchy-TEE, which aims to protect the confidentiality and integrity of data and models of multiple participating actors throughout the inference process using the Trusted Execution Environment (TEE). Branchy-TEE dynamically loads the inference network into the TEE on-demand based on early-exit mechanism, expecting to break the hardware performance bottleneck of the TEE. Moreover, a joint training method based on knowledge distillation for multi-exit networks is proposed, by flowing "knowledge" from the final exit with high accuracy to the early branch exit with lower accuracy. Finally, the effectiveness and efficiency of Branchy-TEE are verified through extensive experiments in real environments, while achieving an optimal balance between performance and hardware resources.**

***Index Terms*—trusted execution environment, early-exit mechanism, security and privacy, knowledge distillation**

I. Introduction

In recent years, Deep Neural Networks (DNNs) have revolutionized the fields of computer vision, autonomous driving, and natural language [1]. This trend has driven the development of DLaaS as a novel computing paradigm, prompting many service providers in specialized fields to deploy their models on top of third-party cloud infrastructure platforms, aiming to provide more convenient and cost-effective services to users worldwide [2].

To address the issue of data security and privacy protection when deploying models in third-party cloud infrastructure platforms, existing cryptography-based approaches can provide sufficient confidentiality and integrity protection for data in transit, computation, and at rest [3, 4]. However, complex and frequent cryptographic computations introduce significant computational and communication overheads, which remain impractical for scenarios with high real-time requirements and limited communication bandwidth.

* Corresponding author: Mingyong Yin (yinmy@caep.cn)
DOI reference number: 10.18293/SEKE2023-131

Meanwhile, security protections in any layer of the computing stack can be bypassed by vulnerabilities in the underlying layers, which has driven the need for lowest-layer (hardware-layer) security solutions. The TEE technologies [5], represented by Intel Software Guard Extensions (SGX)[6], have received increasing attention to ensuring the confidentiality and integrity of data and code by providing a hardware-protected secure area (termed Enclave) [7]. Although SGX provides strong security in an untrusted computing environment, it also suffers from performance degradation under certain conditions. The inference latency has undoubtedly become the most severe obstacle for SGX to develop deep learning systems in the cloud [8, 9]. Taegyeong Lee [10] found that deep learning inference inside an Enclave is up to 6.4 times slower than running outside the Enclave. The main reason for the performance degradation is the hardware design limitation. SGX has limited EPC (Enclave Page Cache) memory capacity, which leads to expensive secure page-swapping operations when the memory required by the model exceeds the EPC size [11].

Previous studies have been proposed to address performance degradation. The SLALOM framework proposed by Tramèr [12] delegates all linear layer computations of deep neural networks from TEE to untrustworthy but faster GPUs, resulting in a 6 to 20-fold increase in inference throughput. Similarly, Gu proposed DeepEnclave [13], which attempts to reduce memory usage peaks by dividing the original DNN model into FrontNet (in EPC) and BackNet (in regular memory). However, the above methods cannot guarantee the confidentiality and integrity of the intermediate outputs of the model, resulting in ineffective defense against DNN reconstruction attacks [14]. In conclusion, it remains challenging to achieve secure DNN inference acceleration while efficiently utilizing valuable EPC memory resources [15].

In this paper, we propose Branchy-TEE, a secure and efficient deep learning inference framework for untrusted cloud platforms, to achieve confidentiality and integrity of data and models throughout the inference process. Specifically, Branchy-TEE achieves a balance between security and efficiency under limited EPC resources by building an early-exit network with multiple exits to load different branch networks into the Enclave on demand. Extensive experiments on DNN models of various scales show that Branchy-TEE

reduces secure inference latency by up to 84.37% compared to native SGX, while introducing a minimum accuracy loss of only 8.39%. In addition, a joint training method of multi-exit networks based on knowledge distillation is proposed, which can compensate the loss of inference accuracy caused by the early exit mechanism to some extent.

II. Problem Statement

A. System Model

The system model of Branchy-TEE is shown in Figure 1 and involves three different entities: a third-party cloud service provider, a model provider, and a data holder.

(1) Third-party cloud service provider (CSP): CSP provide powerful computational and storage resources for DNN model inference computation. In addition, CSP supports hardware-protected trusted execution environment (SGX Enclave) for deep learning inference computation on the cloud.

(2) Model provider:(MP): The MP offloads its model and parameters to the Enclave through a secure communication channel and instantiates its model inside the Enclave and hosts the inference service.

(3) Data holder:(DH): The DH verifies the inference service is running in the Enclave through remote attestation, and then offloads its private input over a secure communication channel. After the inference service is executed, the output will be returned to the corresponding DP through the established secure channel.

B. Threat Model

With the same trust assumption as the existing research work [16], the only trusted entity in Branchy-TEE is the Intel SGX-enabled CPU on the cloud server, which must support Local/Remote Authentication mechanisms (LA/RA) in addition to providing a trusted execution environment. Apart from that, everything in the infrastructure is untrustworthy, and the potential threats considered in Branchy-TEE are mainly from: Honest-But-Curious cloud providers, malicious co-located cloud tenants, and privileged attackers. Branchy-TEE can preserve confidentiality and integrity for both parties against above potential threats: (1)DH's private input is not leaked to CSP; (2) The MP's DNN model (e.g., model architecture and parameters) is not revealed to the CSP and DH in the computation.

III. The Branchy-TEE System Design

A. Overall Architecture of Branchy-TEE System

Fig. 1 illustrates an overview of the secure deep learning inference system based on TEE provided by the Branchy-TEE framework.

Phase 1. Initially, the MP instantiates the model inside a real Enclave in a third-party cloud infrastructure via the remote attestation mechanism RA (step ❶). Besides, the model and parameters are offloaded to the Enclave via a secure channel (step ❷).

Phase 2. During the instantiation of the received model, Branchy-TEE loads the first branch network into **Enclave 1**

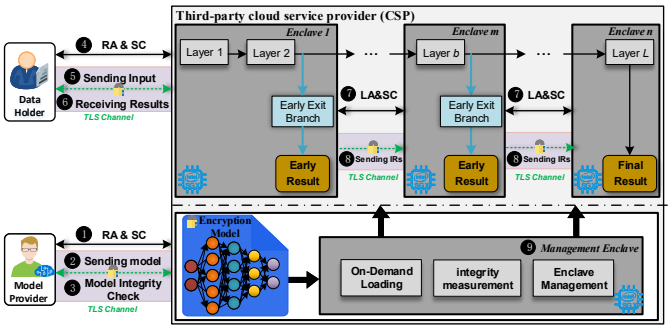


Fig. 1: Overall architecture of proposed system Branchy-TEE.

via the Layer On-Demand Loading module (LOL). When this early branch fails to make a valid decision, the LOL module loads the second branch network into **Enclave 2**. It takes the Intermediate Representations (IRs) output from the previous branch network as input to continue the inference computation until the network gives the final inference result. In order to ensure the confidentiality and integrity of IRs transmission, Branchy-TEE implements the authentication among Enclave in the same platform through the Local Attestation Mechanism (LA) provided by SGX (step ❼). After the authentication is passed, a secure TLS communication channel is established to transmit the network intermediate representation IRs(step ❽).

Phase 3. Similar to step ❶ of MP, DH verifies that the inference service is running in the hardware-protected Enclave through the RA mechanism, and establishes a secure communication channel after the verification is passed (step ❹). The DH's private input is then submitted over this secure channel to the Enclave (step ❺), and once the inference is completed the DH accepts the inference result over the secure channel (step ❻).

B. Layer On-Demand Loading based on Early-exit Mechanism

State-of-the-art SGX-based systems still suffer from significant performance overhead caused by limited EPC memory. Therefore, Branchy-TEE proposes the Layer On-Demand Loading module (LOL) based on early-exit Mechanism to load different branching networks layer by layer in a sequential manner, aiming to expect the model to make inference decisions at an early stage and minimize the EPC memory usage.

1) Relevant Definitions: The network structure of DNN inference tasks is hierarchically structured. We can define DNN inference formally as a function: $y = f^*(x) = f_{\theta_n} f_{\theta_{n-1}} \cdots f_{\theta_1}(x)$. The input x is mapped to the output y by a layer-by-layer nonlinear transformation, where f_{θ_i} and θ_i denote the i_{th} layer subfunction and its corresponding parameters, respectively, and n is the number of the DNN layers. The intermediate representation of the ith hidden layer can be denoted as $IR_i = f_{\theta_i} f_{\theta_{i-1}} \cdots f_{\theta_1}(x)$. Thus, we can give a formal definition of early-exit network with multiple exits and other related definitions as follows:

Definition 1. *(Early-Exit Network) Given a DNN model with $|\mathcal{M}|$ branch exits $y = f^{\mathcal{M}}(x, \theta) : \mathbb{X} \to \mathbb{P}^{|\mathcal{M}|}$, where $\mathcal{M} = \left(e_1, e_2, \cdots, e_{|\mathcal{M}|}\right)$ denotes the layer of the backbone network where the branch exit is located. For any branch exit $\mathcal{M}_i$, its classification probability distribution is calculated $P_{\mathcal{M}_i} : \mathbb{IR} \to \triangle^K$ as follows.*

$$P_{\mathcal{M}_i} = c_{\mathcal{M}_i}\left(IR_{e_i}, \tilde{\theta}_{\mathcal{M}_i}\right) \tag{1}$$

where e_i denotes the layer of the backbone network where the branch exit $\mathcal{M}_i$ is located, $c_{\mathcal{M}_i}$ and $\tilde{\theta}_{\mathcal{M}_i}$ denote the branch classifier and the corresponding parameters, respectively, and $\triangle^K$ denotes the probability distribution of the model output y corresponding to K classes.

Definition 2. *(Branch decision confidence) For any current branch exit $\mathcal{M}_i$, the output probability distribution of that branch classifier $c_{\mathcal{M}_i}$ is $P_{\mathcal{M}_i}$, and the confidence degree $\mathcal{H}(\cdot) \in (0, 1]$ of that branch decision can be measured by calculating the normalized entropy of $P_{\mathcal{M}_i}$, as follows:*

$$\mathcal{H}\left(P_{\mathcal{M}_i}\right) = -\frac{1}{\log K}\sum_{i=1}^{K} p_{\mathcal{M}_i} \log p_{\mathcal{M}_i} \tag{2}$$

When $\mathcal{H}\left(P_{\mathcal{M}_i}\right) < \beta_i$, it means that the branch is confident enough to perform an early exit on the network, otherwise the inference continues to the next branch execution, where $p_{\mathcal{M}_i} \in P_{\mathcal{M}_i}$ and β_i is the pre-set decision threshold of the current branch.

Branchy-TEE is mainly oriented towards feed-forward DNN for classification tasks, where the inference process is performed by extracting feature representations (i.e., IR) layer by layer. Therefore, the computation of each layer is semantically independent, and the IR is dynamically generated by each layer and is only accessed by the layer that generated it and the connected next layer in a short time. It leads to very low reusability of EPC memory and triggers a high frequency of secure paging in EPC. This feature motivates Branchy-TEE to cut the original network into M independent mutually exclusive branch networks loaded into the secure Enclave on demand.

2) Secure Inference Acceleration Algorithm in Enclave: In this subsection, we first detail the security inference acceleration algorithm, termed FINE (Fast Inference IN Eclaves) in Branchy-TEE, as shown in Algorithm 1 below. In summary, based on the early-exit mechanism, the FINE algorithm is built on the BranchyNet[17]. After decrypting the user's sensitive input data within the first Enclave (line 4 of the algorithm), any $Enclave_{\mathcal{M}_i}$, provides a secure computation for its associated branch network $branch_{\mathcal{M}_i}$. The $branch_{\mathcal{M}_i}$ is consisted of the backbone network $f^{\mathcal{M}}_{\theta_{e_i}} \cdots f^{\mathcal{M}}_{\theta_{e_{i-1}+1}}$ (line 7 of the algorithm) and the branch classifier $c_{\mathcal{M}_i}$ (line 8 of the algorithm) in series. When $\mathcal{H}\left[c_{\mathcal{M}_i}\left(IR_{e_i}, \theta_{\mathcal{M}_i}\right)\right] < \beta_i$, Branch-TEE supports network early exit and the final inference result is given by this branch, and the output of $\mathcal{M}_i$ will be sent to the DH via a secure channel (lines 12 to 14 of the algorithm). The confidence degree of the current branch network decision can be measured by calculating the entropy of the inference result (soft-label) (as shown in definition 2); If $\mathcal{H}\left[c_{\mathcal{M}_i}\left(IR_{e_i}, \theta_{\mathcal{M}_i}\right)\right] > \beta_i$, the intermediate representation of the backbone network IR_{e_i} will be sent as input to $Enclave_{\mathcal{M}_{i+1}}$ via a secure channel (line 16 of the algorithm), where the following branch network is located, to continue the inference task layer by layer.

Algorithm 1 FINE: Fast Inference IN Eclaves

Input: $\bar{x}$, encrypted user input; $f^{\mathcal{M}}_{\theta}$, a DNN model with $|\mathcal{M}|$ branch exits; $\mathcal{M}$, branch exit points; θ, parameters of each layer; $\tilde{\theta}$, corresponding parameters of branch classifier $c_{\mathcal{M}}$; $B = \left\{\beta_1, \beta_2, \cdots, \beta_{|\mathcal{M}|}\right\}$, set of branch exit decision thresholds; K, number of categories in the model output.

Output: $\bar{y}$, encrypted inference result

1: **for** $i = 1$ to $|\mathcal{M}|$ **do**
2: Initialize decision confidence $confidence = 0$
3: **if** $i == 1$ **then**
4: $x \leftarrow Decryption\left(\bar{x}\right)$
5: $IR_{e_1} = f^{\mathcal{M}}_{\theta_{e_1}} \cdots f^{\mathcal{M}}_{\theta_1}(x)$
6: **else**
7: $IR_{e_i} = f^{\mathcal{M}}_{\theta_{e_i}} \cdots f^{\mathcal{M}}_{\theta_{e_{i-1}+1}}\left(IR_{e_{i-1}}\right)$
8: $P_{\mathcal{M}_i} = c_{\mathcal{M}_i}\left(IR_{e_i}, \tilde{\theta}_{\mathcal{M}_i}\right)$
9: **for** $p_{\mathcal{M}_i}$ in $P_{\mathcal{M}_i}$ **do**
10: $confidence = confidence + p_{\mathcal{M}_i} \log p_{\mathcal{M}_i}$
11: $confidence = -\frac{1}{\log K} * confidence$
12: **if** $confidence < \beta_i$ or $i == |\mathcal{M}|$ **then**
13: $\bar{y} \leftarrow Encryption\left(\arg\max P_{\mathcal{M}_i}\right)$
14: **return** $\bar{y}$
15: **else**
16: $Enclave_{\mathcal{M}_{i+1}} \leftarrow Sending\left(Enclave_{\mathcal{M}_i}, IR_{e_i}\right)$
17: **continue**

C. Joint Training based on Knowledge Distillation for Multi-exit Networks

In order to compensate for the loss of inference accuracy introduced by the early exit mechanism, a joint training method based on knowledge distillation is proposed to transfer "knowledge" from the last exit (teacher network) with high classification accuracy to the early branch exit (student network) with lower accuracy.

As shown in Fig. 2, the loss function of the knowledge distillation network (illustrated in Eq. 3) consists of the distillation loss extracted from the Soft-target by the teacher network and the original classification loss of the student network (Hard-target), both jointly weighted. The "distillation" method incorporates a "temperature" parameter T into the softmax function, causing the output of the softmax layer to become smoother as T increases. As a result, the information carried by negative labels is amplified relative to positive labels, and the model training focuses more on negative labels [18]. Specifically, given a training set $\{(x_n, y_n)\}_{n=1}^{N}$, the overall loss of the joint training is calculated as follows:

$$\mathcal{L}^{total} = \frac{1}{N}\sum_{n=1}^{N}\left[\mathcal{L}^{class}\left(x_n, y_n\right) + \lambda \mathcal{L}^{dist}\left(x_n\right)\right] \tag{3}$$

where $\mathcal{L}^{class}$ is the classification loss of the multi-exit network (dashed arrow in Figure 2), $\mathcal{L}^{dist}$ is the distillation loss (solid arrow in Figure 2), and λ is the superparameter that balances between the two classes of losses.

The distillation loss $\mathcal{L}^{dist}$ is utilized to transfer knowledge to each of the early exits s_{e_i}, improving the prediction accuracy of the early branches. Specifically, the last exit is used as the teacher network t, and the cross-entropy between the $softmax$ distribution of the branch network and the $softmax$ distribution of the teacher network is calculated as the branch distillation loss $\mathcal{L}^{\tau}_{t\to s_{e_i}}$ under the temperature $T=\tau$ condition. The losses of each branch are then weighted and summed to obtain the overall distillation loss $\mathcal{L}^{dist}$.

$$\mathcal{L}^{dist} = \sum_{i=1}^{|\mathcal{M}-1|}\mathcal{L}^{\tau}_{t\to s_{e_i}} = \sum_{i=1}^{|\mathcal{M}-1|}\left[-\sum_{k=1}^{K} p_k^{\tau}\log\left(q_k^{e_i,\tau}\right)\right]$$
$$where\ p_k^{\tau} = \frac{exp\left(v_i/\tau\right)}{\sum_{j=1}^{K} exp\left(v_j/\tau\right)},\ q_k^{e_i,\tau} = \frac{exp\left(z_k^{e_i}/\tau\right)}{\sum_{j=1}^{K} exp\left(z_j^{e_i}/\tau\right)} \tag{4}$$

where p_k^{τ} denotes the probability of the $softmax$ output of the teacher network t at temperature τ on the i_{th} class, v_i denotes the $logits$ of the teacher network output, $q_k^{e_i,\tau}$ denotes the probability of the $softmax$ output of the branch network e_i at temperature τ on the i_{th} class, and $z_k^{e_i}$ denotes the $logits$ of the branch network output.

As the teacher network may also have a certain error rate, equation 3 introduces a classification loss $\mathcal{L}^{class}$ to mitigate the propagation of errors from the teacher model to the student model by using the ground truth label. The classification loss is defined as follows:

$$\mathcal{L}^{class} = \sum_{i=1}^{|\mathcal{M}|}\left[-\frac{1}{\tau^2}\sum_{k=1}^{K} c_k \log q_k^{e_i,T=1}\right],$$
$$where\quad q_k^{e_i,T=1} = \frac{exp\left(z_k^{e_i}\right)}{\sum_{j=1}^{K} exp\left(z_j^{e_i}\right)} \tag{5}$$

where $c_k \in \{0,1\}$ denotes the ground-truth value on the k_{th} class and $q_k^{e_i,\tau=1}$ denotes the $softmax$ calculation for each branch without distillation ($T=1$). Since the introduction of temperature τ into the $softmax$ function in the $\mathcal{L}^{dist}$ to soften the probability distribution, the size of the gradient generated by the distillation loss is reduced by τ^2 compared to the categorical loss, we multiply the coefficients of $\frac{1}{\tau^2}$ in Eq. 5 to ensure that the distillation loss and the classification loss contribute essentially the same amount of gradient.

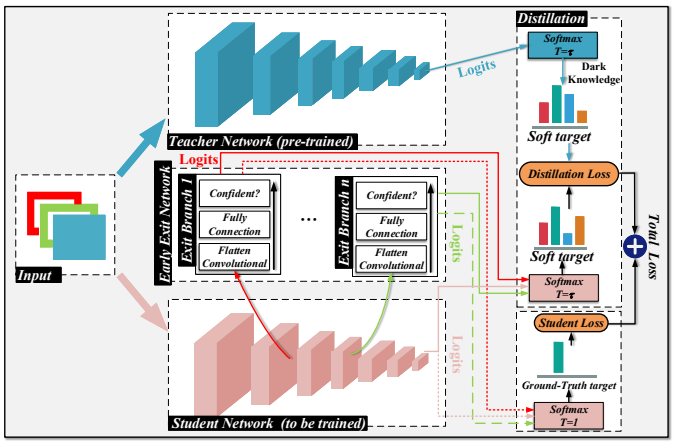


Fig. 2: A multi-exit joint training method based on knowledge distillation.

IV. EXPERIMENTS

A. Experimental Setup

Environment. Branchy-TEE is implemented based on the SGX Library Operating System Graphene-sgx[19], with a higher degree of compatibility. The processor supports SGX feature and has 128MB of EPC memory.

Models and datasets. We employ DNN models of different scales to evaluate Branchy-TEE: including AlexNet, VGG19, MobileNet v1, ResNet50, ResNet101, and ResNet152. Regarding the dataset, we used the CIFAR-10 [20] to compare and validate the performance of our proposed secure inference algorithm.

Baseline. Comparison with Branchy-TEE by the following baseline methods: **(1) Native-DNN**, DNN inference tasks are running in unprotected memory outside of Enclaves; **(2)Native-SGX**, DNN inference tasks are forced to be executed in the native SGX Enclave without providing any optimization mechanism.

B. Result Analysis

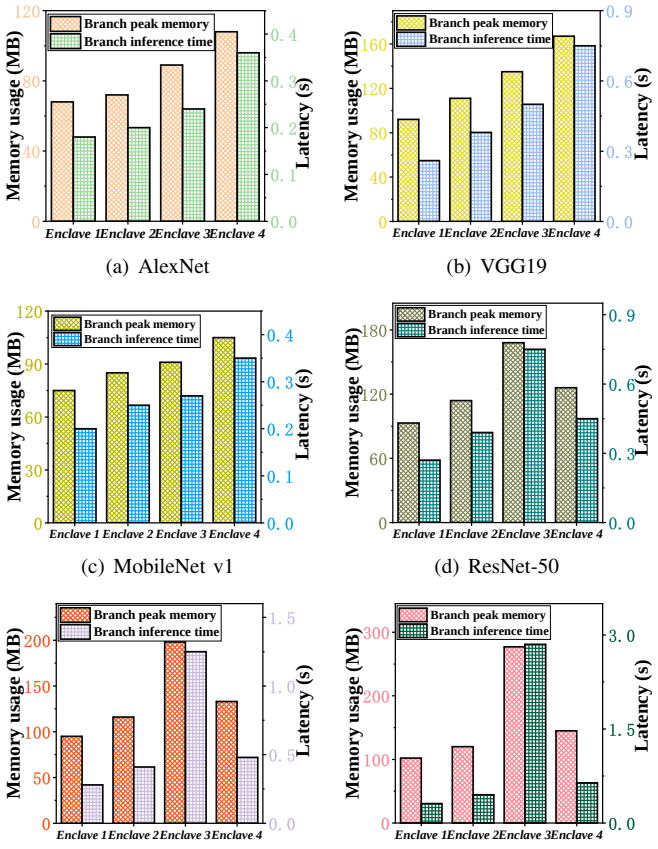
First, the performance overhead of inference in and outside the Enclave for DNN models of different scales is illustrated in Table I. It is evident from the table that the inference latency of Native-SGX is an order of magnitude higher than that of Native-DNN, with an average latency of 33x. As mentioned earlier, the reason for such severe performance degradation is that all models run with a peak memory much larger than the upper limit of EPC memory, triggering a large number of EPC Secure Paging operations, which significantly reduces the inference speed of the models within the Enclave.

The conclusions drawn in Fig. 3 reveal that the more backward branching network has a higher demand for EPC memory, and the cost of an EPC page swap is quite expensive (up to 40K cycles) [21]. It motivates us to assign a more loose branch decision threshold B in Algorithm 1 to encourage the network to give a final decision at an early stage.

Furthermore, Table II compares the secure inference performance between Branchy-TEE and Native-SGX in terms of

TABLE I: Performance overhead introduced by executing deep inference tasks in Native-SGX.

models	model sizes	Peak memory	Inference Latency		
			Native-DNN	Native-SGX	Overhead
AlexNet	492KB	161MB	0.022s	1.617s	73.17x
VGG19	77MB	335MB	0.146s	4.244s	29.06x
MobileNet v1	13MB	190MB	0.069s	2.470s	35.71x
ResNet50	91MB	339MB	0.287s	5.871s	20.43x
ResNet101	163MB	484MB	0.472s	10.144s	21.51x
ResNet152	223MB	618MB	0.694s	18.639s	26.87x



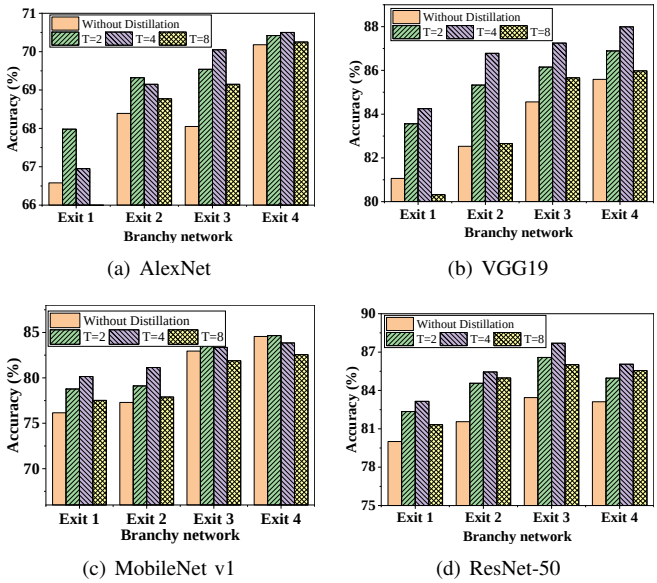
(a) AlexNet (b) VGG19 (c) MobileNet v1 (d) ResNet-50 (e) ResNet-101 (f) ResNet-152

Fig. 3: The performance of each early-exit branch of different model running separately in Enclave.

average peak memory usage and average inference latency, respectively. Overall, Branchy-TEE significantly outperforms Native-SGX for all target models. For the AlexNet model with the slightest performance improvement, the average peak memory requirement of Branchy-TEE decreases by 16.28% compared to Native-SGX. In contrast, for ResNet152, with the highest EPC memory requirement, this decrease is as high as 83.06%. Correspondingly, the average inference latency generated by the AlexNet is reduced by up to 33.10% compared to Native-SGX, and up to 84.15% in ResNet152. However, the model accuracy loss introduced by the early-exit mechanism is not negligible, ranging from 8.39% for VGG19 to 12.46% for ResNet-101, respectively.

TABLE II: Security performance comparison between Branchy-TEE (ours), Native-SGX, and Native-DNN.

Models	Accuracy loss (Native-DNN)	Average peak memory		Average inference latency	
		Ours	Native-SGX	Ours	Native-SGX
AlexNet	↓9.15%	129.95MB	↓16.28%	1.08s	↓33.10%
VGG19	↓8.39%	128.72MB	↓61.57%	0.66s	↓84.37%
MobileNet v1	↓10.87%	102.64MB	↓45.98%	0.63s	↓82.80%
ResNet50	↓11.13%	139.75MB	↓58.78%	1.04s	↓82.23%
ResNet101	↓12.46%	144.41MB	↓70.16%	1.18s	↓79.88%
ResNet152	↓11.07%	160.34MB	↓83.06%	2.95s	↓84.15%



(a) AlexNet (b) VGG19 (c) MobileNet v1 (d) ResNet-50

Fig. 4: Performance impact of knowledge distillation on the accuracy of each branchy with different temperature.

Fortunately, the joint training method based on knowledge distillation mentioned above is able to compensate to some extent for the accuracy loss introduced by the multiple exit early retirement network. As shown in Fig. 4, different branching networks show a certain degree of upward trend in Top-1 accuracy after knowledge distillation. In addition, different distillation temperatures have different effects on different models and branches. For some simple models, such as AlexNet and Vgg19, the accuracy improvement effect of distillation temperature $T = 4$ is not better than that at $T = 2$. Even at the distillation temperature $T = 8$, the accuracy of AlexNet and VGG19 is slightly lower than that of the architecture without knowledge distillation.

Conversely, for the ResNet50 network, the accuracy improvement for the network showed an increasing trend with increasing distillation temperature. In general, the choice of distillation temperature T has a certain relationship with the size of the model. The model with a relatively small number of parameters cannot learn all the knowledge of the teacher

network, and a relatively low temperature cannot amplify the information of the soft-target, but can effectively ignore the noise generated by some negative labels.

V. CONCLUSIONS

In this paper, we propose a secure and efficient inference framework for deep learning in untrustworthy cloud platforms, termed Branchy-TEE, which aims to protect the confidentiality and integrity of data and models throughout the inference process using SGX. Moreover, Branchy-TEE dynamically loads the inference network into the TEE on-demand based on early-exit mechanism, breaking the hardware performance bottleneck of SGX. Moreover, a joint training method based on knowledge distillation for multi-exit networks is proposed to compensate for the loss of inference accuracy introduced by the early exit mechanism.

REFERENCES

[1] Yoshua Bengio, Yann Lecun, and Geoffrey Hinton, "Deep learning for ai," *Communications of the ACM*, vol. 64, no. 7, pp. 58–65, 2021.

[2] Google, "Cloud automl.," https://cloud.google.com/automl, Accessed:2022-10-22.

[3] Brian Knott and Shobha et al Venkataraman, "Crypten: Secure multi-party computation meets machine learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4961–4973, 2021.

[4] Payman Mohassel and Yupeng Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 19–38.

[5] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah, "Trusted execution environment: what it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/ISPA*. IEEE, 2015, vol. 1, pp. 57–64.

[6] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar, "Innovative instructions and software model for isolated execution.," *Hasp@ isca*, vol. 10, no. 1, 2013.

[7] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, Andre Martin, Christian Priebe, Joshua Lind, Divya Muthukumaran, Dan O'keeffe, Mark L Stillwell, et al., "{SCONE}: Secure linux containers with intel {SGX}," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 689–703.

[8] Yuepeng Li, Deze Zeng, Lin Gu, Quan Chen, Song Guo, Albert Zomaya, and Minyi Guo, "Lasagna: Accelerating secure deep learning inference in sgx-enabled edge cloud," in *Proceedings of the ACM Symposium on Cloud Computing*, 2021, pp. 533–545.

[9] Tu Dinh Ngoc, Bao Bui, Stella Bitchebe, Alain Tchana, Valerio Schiavoni, Pascal Felber, and Daniel Hagimont, "Everything you should know about intel sgx performance on virtualized systems," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 1, pp. 1–21, 2019.

[10] Taegyeong Lee, Zhiqi Lin, Saumay Pushp, Caihua Li, Yunxin Liu, Youngki Lee, Fengyuan Xu, Chenren Xu, Lintao Zhang, and Junehwa Song, "Occlumency: Privacy-preserving remote deep-learning inference using sgx," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–17.

[11] Meysam Taassori, Ali Shafiee, and Rajeev Balasubramonian, "Vault: Reducing paging overheads in sgx with efficient integrity verification structures," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, 2018, pp. 665–678.

[12] Florian Tramer and Dan Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," *arXiv preprint arXiv:1806.03287*, 2018.

[13] Zhongshu Gu, Heqing Huang, Jialong Zhang, Dong Su, Ankita Lamba, Dimitrios Pendarakis, and Ian Molloy, "Securing input data of deep learning inference systems via partitioned enclave execution," *arXiv preprint arXiv:1807.00969*, 2018.

[14] Zecheng He, Tianwei Zhang, and Ruby B Lee, "Attacking and protecting data privacy in edge–cloud collaborative inference systems," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9706–9716, 2020.

[15] Kyungtae Kim, Chung Hwan Kim, Junghwan" John" Rhee, Xiao Yu, Haifeng Chen, Dave Tian, and Byoungyoung Lee, "Vessels: Efficient and scalable deep learning prediction on trusted processors," in *Proceedings of the 11th ACM Symposium on Cloud Computing*, 2020, pp. 462–476.

[16] Tyler Hunt, Zhiting Zhu, Yuanzhong Xu, Simon Peter, and Emmett Witchel, "Ryoan: A distributed sandbox for untrusted computation on secret data," *ACM Transactions on Computer Systems (TOCS)*, vol. 35, no. 4, pp. 1–32, 2018.

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

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[19] Chia-Che Tsai, Donald E Porter, and Mona Vij, "{Graphene-SGX}: A practical library {OS} for unmodified applications on {SGX}," in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, 2017, pp. 645–658.

[20] A Krizhevsky, "Learning multiple layers of features from tiny images," *Master's thesis, University of Tront*, 2009.

[21] Meni Orenbach, Pavel Lifshits, Marina Minkin, and Mark Silberstein, "Eleos: Exitless os services for sgx enclaves," in *Proceedings of the Twelfth European Conference on Computer Systems*, 2017, pp. 238–253.