ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

SOTER: Guarding Black-box Inference for General Neural Networks at the Edge

Tianxiang Shen, Ji Qi, Jianyu Jiang, Xian Wang, Siyuan Wen, Xusheng Chen, Shixiong Zhao, Sen Wang, Li Chen, Xiapu Luo, Fengwei Zhang, Heming Cui

The University of Hong Kong

Huawei Technologies

The Hong Kong Polytechnic University

Southern University of Science and Technology

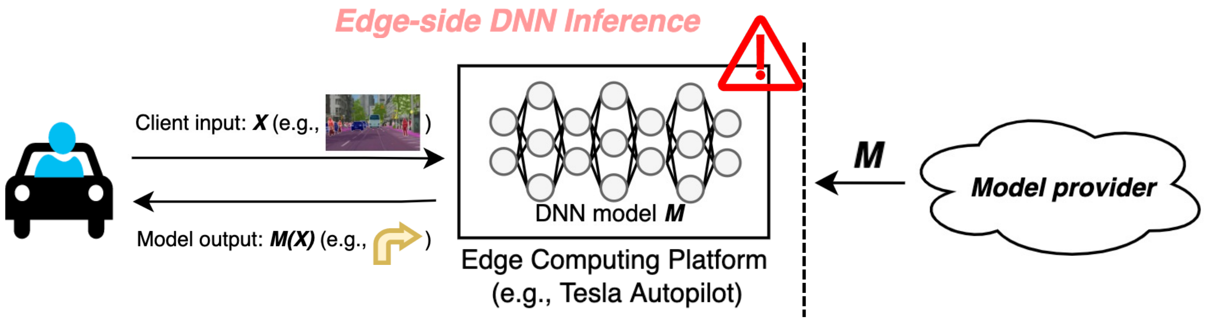
Background: Edge-side DNN Inference

- **Giant companies (e.g., Google) provide well-trained Deep Learning (DL) models to clients**
  - DL models, especially **Deep Neural Networks** (DNN), serve numerous mission-critical AI applications

Autonomous Driving	Home Monitoring	Virtual Assistance	Speech Recognition	…
TESLA	Sentry	Cortana	Google Cloud Speech API	…

  - Giant companies pay **substantial effort** to train **accurate** models, which are **private**

- **To provide high-quality (low-latency) services, DNN models are usually deployed on edge-side user devices**
  - Clients (i.e., users) run *edge-side DNN inference* to get real-time results



  - However, sensitive model parameters are exposed, and inference can be easily interfered at the untrusted edge!

- **In sum, edge-side inference requires low latency, high accuracy with confidentiality and integrity protection**

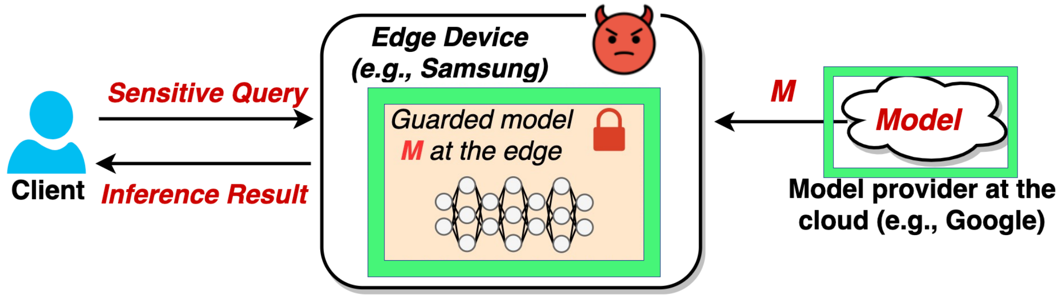
Background: Trusted CPU TEE & Untrusted GPU

- **Trusted Execution Environment (TEE) is promising to protect model confidentiality**
  - TEEs (e.g., Intel SGX, ARM TrustZone) provides **data confidentiality** and **code integrity** guarantees
  - TEEs are widely used to protect edge services
    - E.g., Samsung uses TrustZone to store *payment information;* Trustonic uses TEE to build IoT apps *with trusted-UI*
  - TEE-based inference systems are emerging (more details in page 4-5)

- **Edge-side TEEs are trusted, but edge-side GPUs are untrusted**
  - **CPU TEE does not support GPU, model providers cannot trust third-party GPUs**
    - Current **Trusted GPU**s either require extensive hardware modifications or support only hardware simulators
  - GPU is essential: Numerous edge devices have been integrated with GPU to accelerate edge intelligence
    - E.g., Apple's A15 chip equips *4-core GPU*; Samsung's new mobile processor Exynos2200 includes *AMD GPU*

Requirements for Edge-side DNN Inference

➢ **Deployment scenario**



➢ **An *ideal* edge-side inference system should meet the following requirements:**

Performance

- ***Low latency:*** utilize co-located GPU accelerator to speed up model inference
- ***High accuracy***: retain the same accuracy as the original model

Security

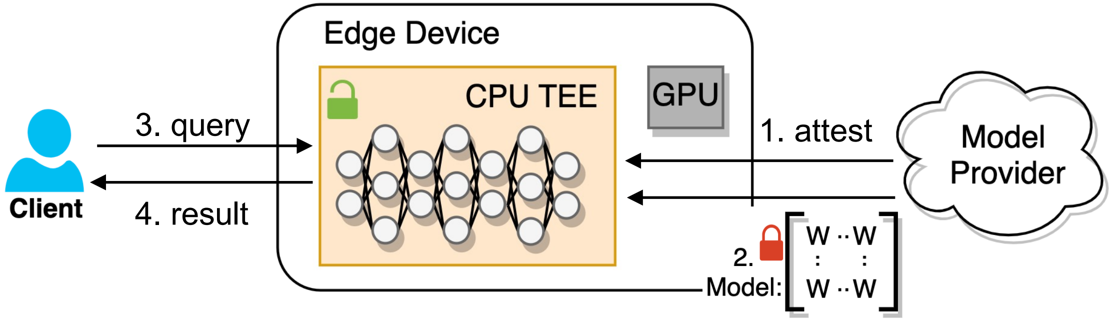
- ***Model confidentiality***: model parameters' plaintexts should be hidden
- ***Inference integrity:*** any attacks (e.g., malicious modifications) on inference results should be detected

Prior work: TEE-shielding Approach

- **Existing TEE-based inference systems include <u>*TEE-shielding*</u> *approach* and <u>*partition-based*</u> *approach***

- ***TEE-shielding* approach (e.g., MLCapsule [CVPR '21])**

  - ***How it works***



1. **Attest** to the TEE-equipped edge device
2. **Offload** and **decrypt** the encrypted model in an attested TEE enclave
3. **Take** client **input** to run inference purely inside the CPU TEE
4. **Return** the **inference** result back to the client

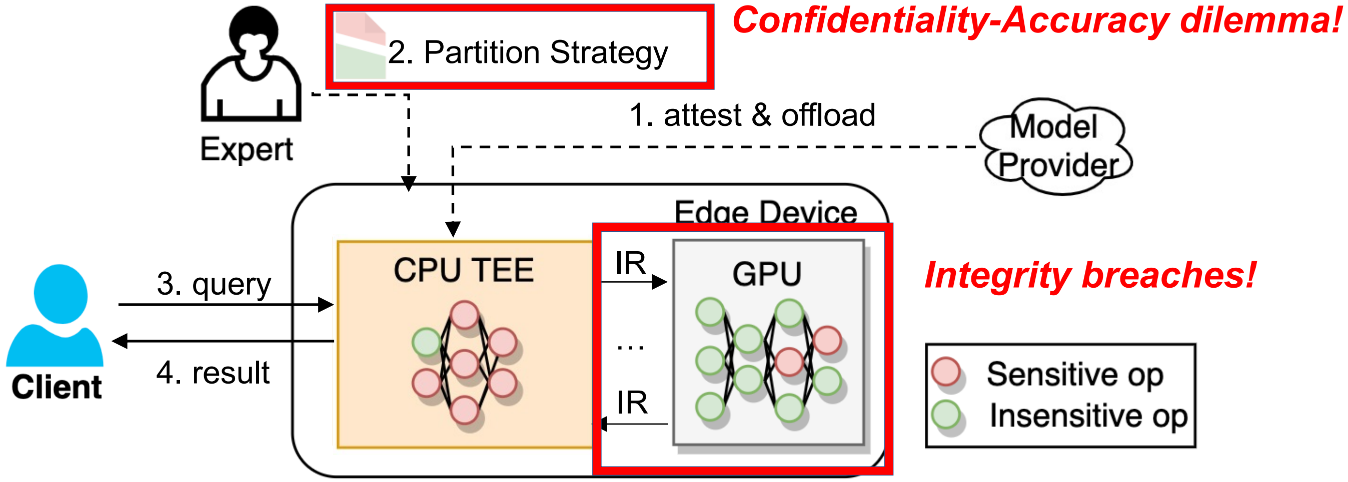
  - ***Advantages:*** Protect model confidentiality and inference integrity; Retain high accuracy

  - ***Limitations:*** No GPU acceleration with extremely high inference latency (up to 36.1*X*) than insecure GPU inference

Prior work: Partition-based Approach

➢ **Partition-based approach (e.g., AegisDNN [RTSS '21], eNNclave [AISec '20])**

- ***How it works***



Sensitive segments **-> trusted-but-slow** CPU TEE

Insensitive segments (with ***plaintext*** or ***retrained* parameters**) -> untrusted-but-fast GPU

- ***Advantages:*** Low latency with GPU acceleration
- ***Limitations***: Incur either confidentiality loss or accuracy loss; Integrity breaches on partitioned model

Goals of Our Solution: SOTER

➢ **SOTER is a partition-based inference system that achieves all desired properties for edge-side DNN inference**

- **Accelerate** heavy-weight computation with GPU and **retain high accuracy** as the original model
- **Protect model confidentiality** by hiding all parameters' plaintexts
- **Detect integrity breaches** (e.g., malicious modifications) on inference results

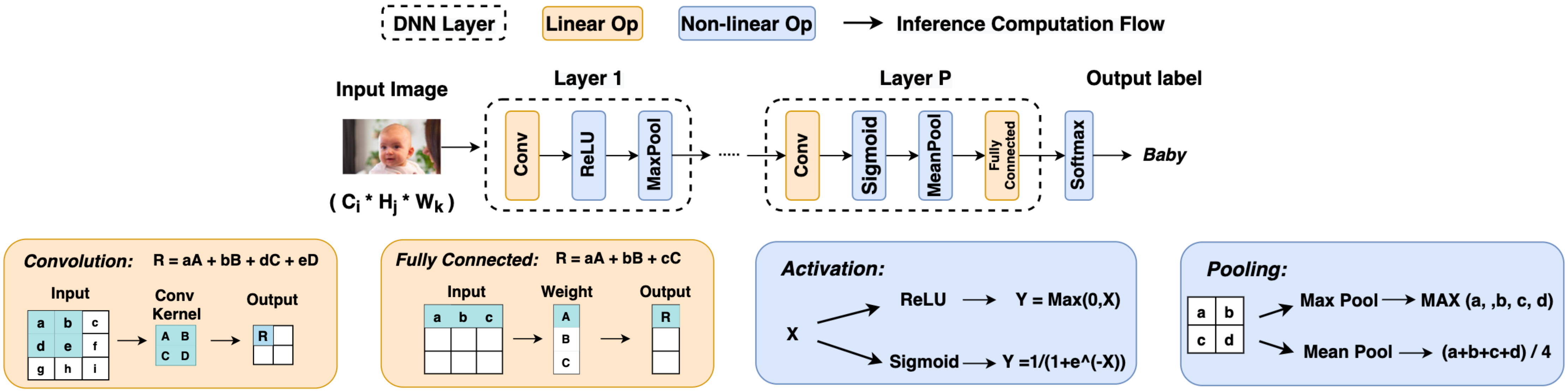
	GPU Acceleration	**No Accuracy Loss**	**Model Confidentiality**	**Inference Integrity**
MLCapsule	😐	😀	😀	😀
eNNclave	😀	😐	😀	😐
AegisDNN	😀	😀	😐	😐
SOTER	😀	😀	😀	😀

➢ **To achieve these goals, SOTER asks two questions:**

- ***Q1:*** *How can we utilize untrusted GPU for acceleration without sacrificing confidentiality or accuracy?*
- ***Q2:*** *How to efficiently detect integrity breaches outside the TEE?*

Recap DNN Model Architecture

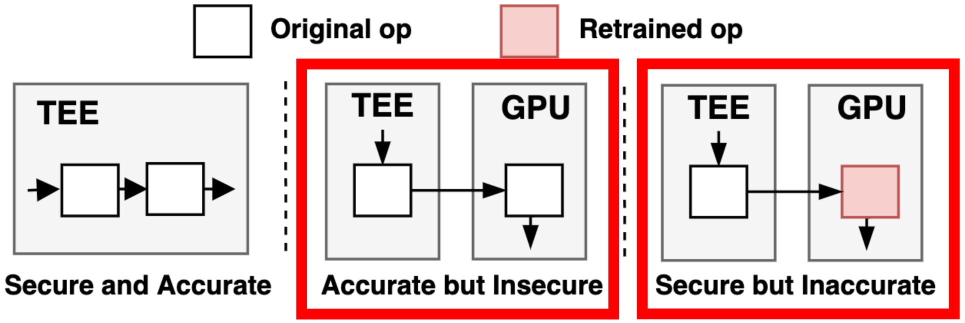
- **Recap DNN model architecture**



- **Associativity of common DNN operators:** (μ^-1 * μ) F(X) = μ^-1 F(μ X)
  - All linear operators (e.g., Conv, FC) satisfy associativity property and they represent a major fraction of model computation
  - Some non-linear operators (e.g., ReLU) are *scale-invariant* and satisfy this property under specific constraints
    - *E.g., ReLU: F*(*x*) = *Max*{0, *X*} is *scale-invariant* when ***μ*** **> 0**, i.e., $\boxed{F(\mu x) = Max\{0, \mu X\} = \mu F(X)}$

Bridging the Confidentiality-Accuracy Gap (Q1)

- **Confidentiality-accuracy dilemma**

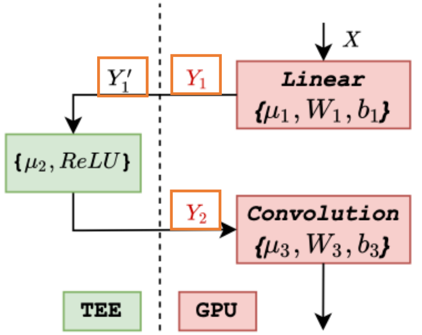


- **SOTER's key weapon: the general associativity property of common inference operators**

sensitive! (μ^-1 * μ) F(X) = μ^-1 F(μ X) insensitive -> GPU

sensitive -> TEE

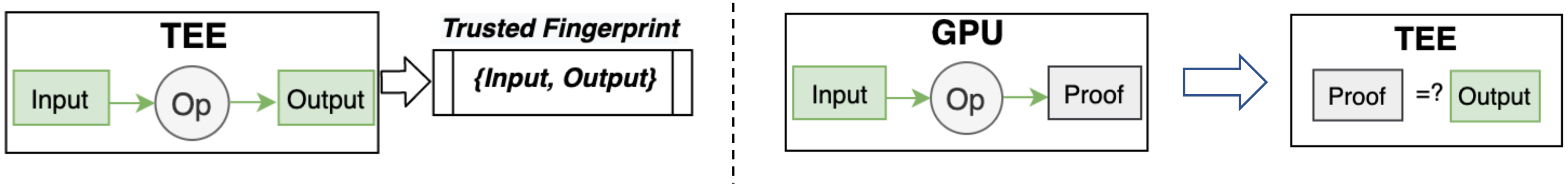
- **Major workflow**
  - **Step 1:** Automatically profile an encrypted model in TEE
  - **Step 2:** Morph a portion of associative operators' parameters with *hidden scalars*
  - **Step 3:** Partition morphed operators to run on GPU
  - **Step 4:** Execute operators in order, transmit IRs between kernels, restore execution results with hidden scalars in TEE



$$Y_1 = \mu_1(W_1 * X + b_1)$$
$$Y_1' = Y_1 * \mu^{-1} = W_1 * X + b_1$$
$$Y_2 = \mu_2 * ReLU(Y_1')$$
$$= 0 \Leftarrow if\ Y_1' \le 0$$
$$= \mu_2 * (W_1 * X + b_1) \Leftarrow if\ Y_1' > 0$$

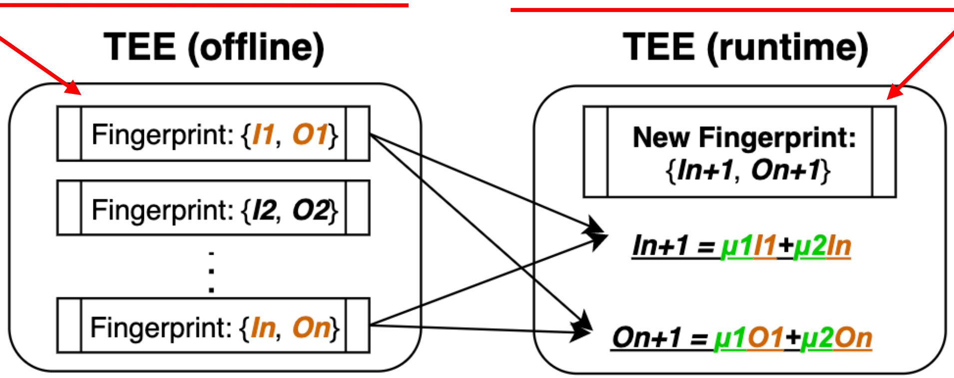
Detecting Integrity Breaches (Q2)

- **Partition-based system *inevitably* open access to integrity breaches outside the TEE**
- **Detect integrity breaches: a straw man *Trusted Fingerprint* (TF) re-computing approach**



- **Key challenge: Obliviousness-timeliness dilemma**
  - If we use fixed TF, the adversary can easily observe and bypass the TF detection
  - If generate new TF as regular user input in CPU TEE, TFs become oblivious to observe, but TF generation (in CPU TEE) becomes the performance bottleneck, leading to slow detection
- **SOTER solves the challenge using the same associativity observation from confidentiality protection**
  - Associativity variant: If F(X1) = Y1; F(X2)=Y2; …; F(Xn)=Yn, then F(μ1X1+ μ2X2+…+ μnXn)= μ1Y1+ μ2Y2+…+ μnYn

Step 1:
Prepare ***cornerstone TFs*** in the preprocessing phase



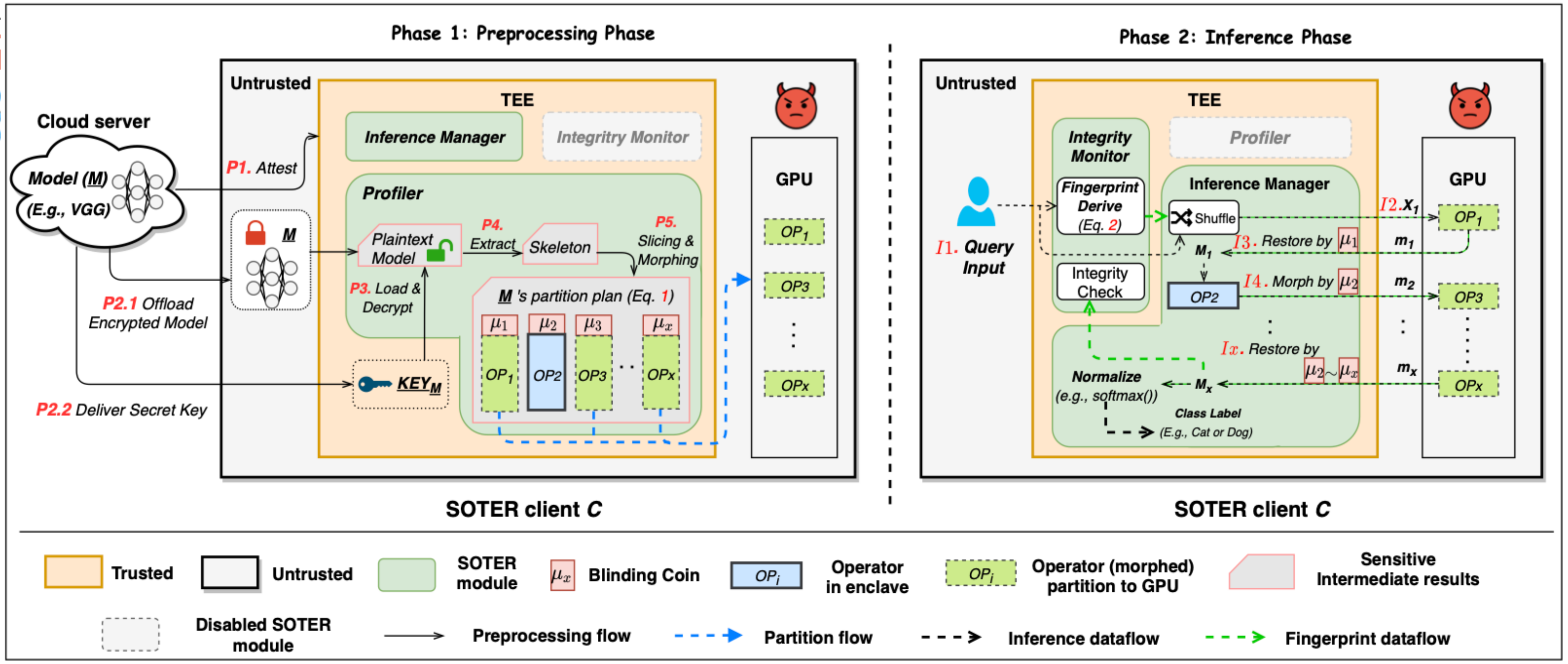
Step 2:
Generate new ***scalars*** and use the ***associativity variant*** to efficiently produce new TFs

SOTER: In a Nutshell

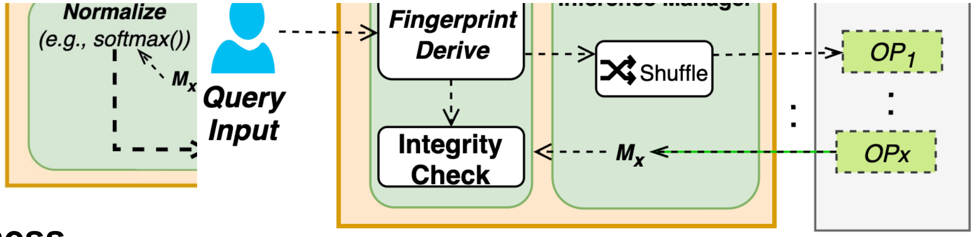
- By using **the same associativity weapon**, **three key modules** run in two phases to collectively provide low latency, high accuracy, model confidentiality, and integrity protection

SOTER Profiler

SOTER architecture



- The ***Profiler*** and ***Inference Manager*** module speeds up model inference with untrusted GPU while protecting parameters' plaintexts
  - Automatically profile and formulate partition plans
  - **Hide partitioned operators' parameters with secret blinding coins**



- The ***Integrity Monitor*** module check partitioned GPU operators' execution results to detect any integrity breaches
  - Top-W operator reserving
  - **Efficiently generate new trusted fingerprints at runtime for obliviousness**

Implementation and Evaluation

- **Implementation Details**
  - Implemented on PyTorch and Graphene-SGX, extensible to any imperative Deep Learning frameworks and TEE codebase
  - Adopted a **two-phase design** for offline preprocessing and online inference
  - Designed a **Morph-Then-Restore protocol** for cooperative executions between kernels (TEE & GPU)
  - Designed a **periodical upgrading mechanism** to prevent chosen plaintext attacks
  - Designed an **on-demand operator prefetching mechanism** to reduce TEE memory footprints

- **Baseline secure inference systems**
  - MLCapsule [CVPR ’21]
  - AegisDNN [RTSS ’21]
  - eNNclave [AISec ’20]

  (The above blue optimization is also incorporated in the three baselines)

- **Evaluation settings in our dedicated cluster**
  - Dell R430 server with 2.60GHZ Intel E3-1280 V6 CPU, 64GB memory, and SGX hardware support
  - A GPU farm with Nvidia 2080Ti GPUs, each GPU had 11GB physical memory
  - Evaluated on VGG19, Alexnet, Resnet152, Densenet121, Multi Layer Perception, and Transformer

Evaluation Questions

- **How is SOTER's end-to-end performance compared to baselines?**
- **How is SOTER's confidentiality protection compared to baselines?**
- **Are SOTER's trusted fingerprint oblivious to the adversary outside the TEE?**
- **How sensitive is SOTER's performance to different partition ratio?**

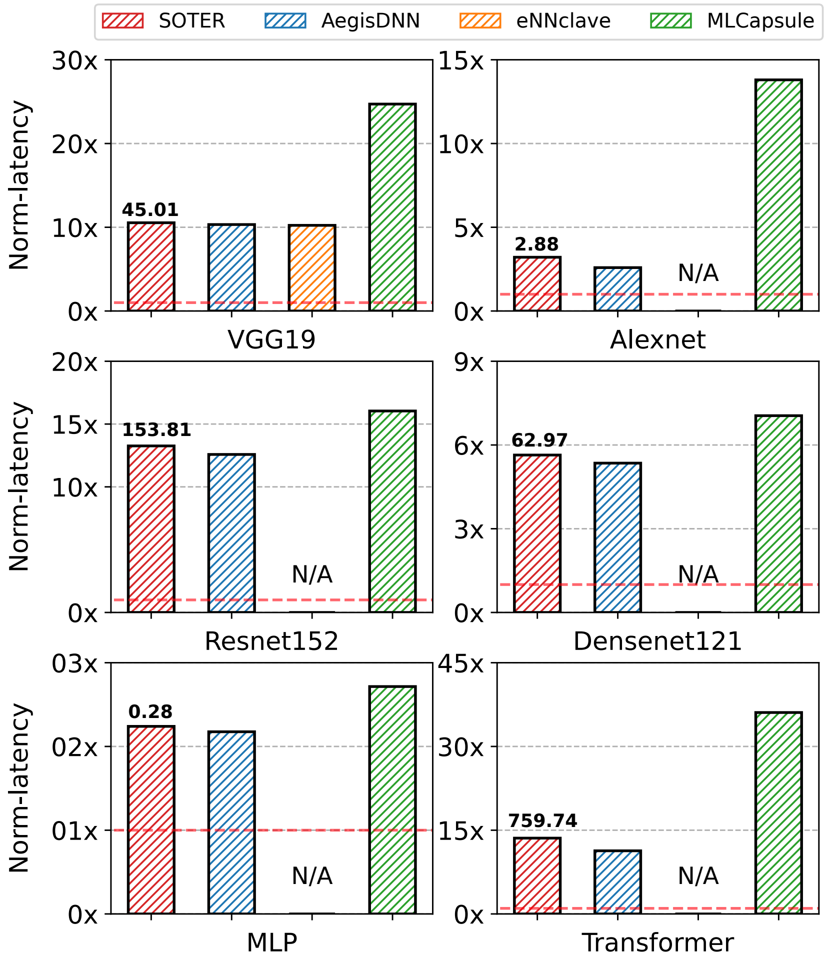
End-to-end performance

➢ **Figure 1 shows the inference latency (normalized to insecure GPU inference, red dotted line) compared to three baselines (SOTA TEE-shielding and partition-based approach) running six prevalent DNN models**

- SOTER achieved **1.21*X* ~ 4.29*X* lower inference latency** than TEE-shielding MLCapsule
- SOTER enforced **integrity protection**, with only 1.03*X* ~ 1.27*X* higher inference latency than partition-based AegisDNN

SOTER's inference results (in milliseconds)						
Model	**MLP**	**AN**	**VGG**	**RN**	**DN**	**TF**
P1: CPU (TEE)	0.19	1.65	25.38	92.18	41.65	439.52
P2: GPU	0.05	0.71	14.24	33.97	13.71	204.93
P3: Kernel Switch	0.01	0.18	0.83	25.98	5.6	41.52
P4: Integrity Check	0.03	0.34	4.56	14.75	6.02	73.77
End-to-end (P1+P2+P3+P4)	0.28	2.88	45.01	153.88	62.97	759.74

Figure 1



Security Evaluation

➢ (**Confidentiality**) Even if SOTER completely hides partitioned operators' plaintexts, an adversary may still conduct ***model stealing attacks*** to train a ***substitute model (SM)***

(A **higher** accuracy/BLEU of SM means **more** confidentiality loss)

- SOTER achieved **stronger confidentiality** protection than AegisDNN
- SOTER achieved **the same strong confidentiality** protection as eNNclave while eNNclave sacrifices inference accuracy

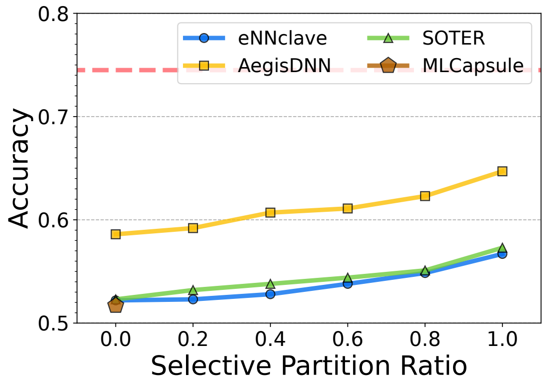


Figure 2.a (on VGG19)

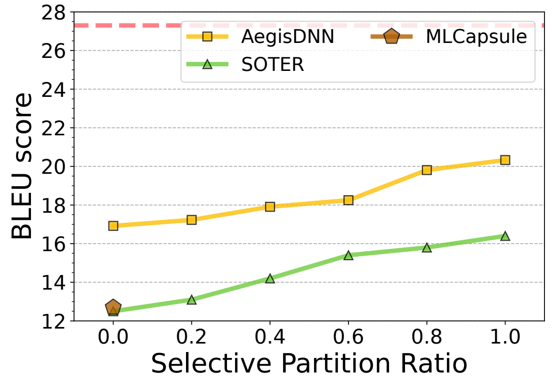


Figure 2.b (on Transformer)

➢ (**Integrity**) Compare SOTER's oblivious trusted fingerprint (Figure 3.a) with the straw man fixed trusted fingerprint approach (Figure 3.b)

- SOTER's fingerprints are **oblivious** to the adversary because the L2 distance distribution of fingerprints shares the same form of normal distribution as client's normal query input

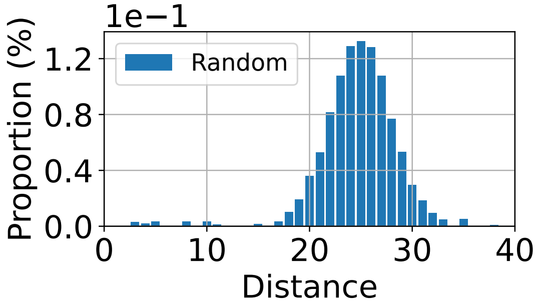


Figure 3.a (w oblivious TF)

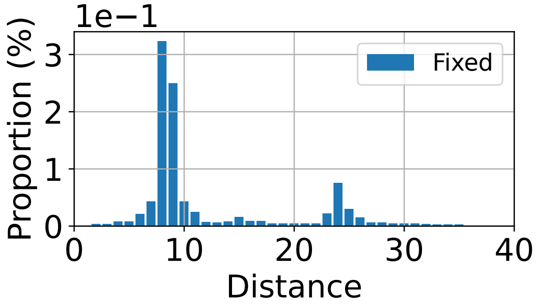


Figure 3.b (w/o oblivious TF)

Conclusion

- In this paper, we present SOTER, the first work that achieves **model confidentiality**, **low-latency** and **high-accuracy** with **integrity protection** for **general** neural network inference
  - Comparable **strong confidentiality** as TEE-shielding approach; Comparable **low latency** as partition-based approach; **High accuracy** same as insecure GPU inference; Overwhelming high probability of obliviously **detecting integrity breaches**
- These features encourage giant companies to develop powerful models and deploy them on third-party edge devices
- SOTER can also help with protecting models on untrusted cloud servers
- SOTER's future work is broad:
  - SOTER can integrate with emerging black-box defenses to further strengthen privacy guarantees
  - SOTER can be extended to multiple GPUs and TEEs for distributed model inference
- SOTER's artifact is available at https://github.com/hku-systems/SOTER

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED