Confidential ML Inference at the Untrusted Edge

Low–Overhead Trusted Execution with ARM TrustZone & Intel SGX

Madhav (Sai) Suri | Advisor: Dr. Yi Ding | The University of Texas at Dallas

Introduction

Abstract

Machine-learning deployments face tight constraints on both privacy and performance when inference runs on edge devices or untrusted clouds, where input data and model weights are exposed to a broad attack surface. Trusted Execution Environments (TEEs) and model partitioning have traditionally been explored as separate avenues for securing inference. **EnclaveSplit** is a research prototype for selective partitioning, executing only the most sensitive layers of each model inside a hardware enclave while keeping the remaining computation in the normal world. The approach is portable across hardware — Designed around a unified enclave interface for Intel SGX and ARM TrustZone/OP-TEE-style execution paths

How can an ML model run on hardware the user does not trust without revealing the input data or the model itself?

Background

A TEE is a hardware-isolated region of a processor that preserves the confidentiality and integrity of code and data even when the operating system is compromised, and supports remote attestation so a remote party can verify the correct code is running. **Intel SGX** encrypts enclave memory and decrypts it only inside the CPU, while **ARM TrustZone** partitions the system-on-chip into a secure world and a normal world, making it well suited to physically exposed edge hardware. Running an entire model inside an enclave incurs severe overhead due to limited protected memory, motivating a partitioning approach that protects only what is necessary.

Confidentiality — data & model stay encrypted in memory

Integrity — tampering with enclave code is detected

Attestation — a remote verifier confirms the enclave's code

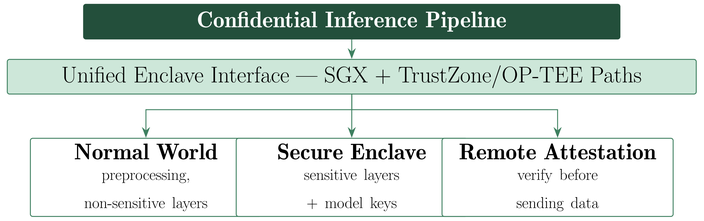
Subhead — what / why / how

What: protect both user input and model weights during inference. **Why:** edge devices are vulnerable to physical access, and cloud hosts face malicious co-tenants and compromised hypervisors. **How:** partition each model so privacy-critical layers execute in the enclave while non-sensitive computation runs in the fast normal world.

Threat Model

We assume an **honest-but-curious** host whose operating system, hypervisor, and user-space software may all be compromised, plus an adversary with physical access to the edge device. The attacker may observe memory, inspect the model, and attempt membership-inference or model-extraction attacks. **Trusted:** the CPU package and the attested enclave code. **Out of scope:** hardware side-channels and denial of service. Under this model, EnclaveSplit keeps the sensitive layers and model keys confidential even when everything outside the enclave is controlled by the adversary.

Methods, Analysis, or Process



Pipeline

Each network is partitioned into a non-sensitive component (run in the normal world) and a sensitive component compiled into a C/C++ enclave; a Python host orchestrates data flow and invokes the enclave for protected computation. The same enclave source targets both Intel SGX and ARM TrustZone through the Open Enclave SDK (one unified API, TrustZone via OP-TEE OS). Before any private data is sent, the host validates the enclave's signed attestation quote to confirm the expected, untampered code is running.

Results — per-workload overhead

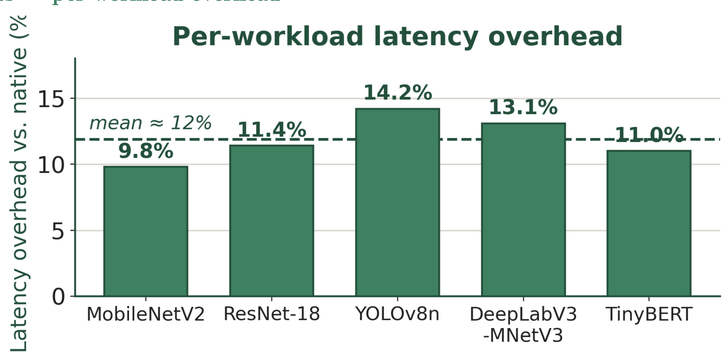


Figure 1. Latency overhead vs. native execution, five edge-AI workloads.

Workload	Task	Overhead	Acc. loss
MobileNetV2	Classification	9.8%	None
ResNet-18	Classification	11.4%	None
YOLOv8n	Detection	14.2%	None
DeepLabV3-MNetV3	Segmentation	13.1%	None
TinyBERT	NLP	11.0%	None
Mean	—	**∼12%**	**None**

Table 1. **Table 1. Per-workload latency overhead and accuracy impact..**

Discussion

At a fixed partitioning configuration, EnclaveSplit limited inference overhead to approximately **12%** relative to native execution across five edge-AI workloads, with no reduction in prediction accuracy. Most of the cost comes from sensitive-layer enclave compute and normal↔secure world switches, while attestation is a one-time, amortized cost. Because the enclave logic is written once against the Open Enclave API, the same pipeline ran on both server-class SGX and edge-class TrustZone hardware without source changes, and we contributed **three merged pull requests** upstream to the Open Enclave SDK.

Conclusion & References

Conclusion

By selectively partitioning computation between the secure and normal worlds, Our EnclaveSplit prototype demonstrates confidential inference with modest performance overhead. This hybrid approach combines hardware-enforced isolation with model partitioning in a single, retraining-free pipeline, and its portability across promising for future cloud and IoT deployments.

Comparison with prior systems

System	Platform	Overhead
EnclaveSplit (ours)	SGX + TrustZone	**∼12%**
DarkneTZ [6]	TrustZone	∼3%*
T-Slices [2]	TrustZone/OP-TEE	∼0%†
Occlumency [1]	SGX	∼72%
Slalom [7]	SGX + GPU	offload
Naive full-enclave [3]	SGX	up to 100×
VM-based TEE [4]	SEV / TDX	∼1.5×

Table 2. *single-layer only; †prediction-time, OP-TEE avoids paging.

Related Work

Prior TEE-based inference splits along two lines. **Edge / TrustZone** systems such as DarkneTZ [6] and T-Slices [2] partition layers to fit limited secure memory, but protecting more layers raises overhead sharply. **Server / SGX** systems such as Occlumency [1] run the full model in the enclave and pay heavy paging costs, while Slalom [7] offloads linear layers to an untrusted GPU with cryptographic verification. EnclaveSplit differs by targeting *both* platforms from one codebase and protecting only the sensitive subset, keeping overhead low without sacrificing portability.

Future Work

The primary remaining challenge is extending protection to **GPU and NPU accelerators**, which currently execute outside the TEE. Promising directions include confidential-computing GPUs (e.g. NVIDIA H100 CC mode), **Arm CCA** Realms for VM-level edge isolation, and **RISC-V** enclaves — all of which could shrink the trusted boundary while keeping accelerated throughput.

Key Takeaways

- Selective partitioning gives strong privacy at ∼12% cost — far below full-enclave approaches.
- One partitioning design can be expressed through a shared enclave abstraction for SGX and TrustZone/OP-TEE-style backends
- Attestation makes the privacy guarantee *verifiable*, not just asserted.

References

[1] T. Lee et al., "Occlumency: Privacy-preserving remote deep-learning inference using SGX," *Proc. ACM MobiCom*, 2019.
[2] C. Kim et al., "Confidential execution of deep-learning inference at the untrusted edge with ARM TrustZone (T-Slices)," *Proc. ACM CODASPY*, 2023.
[3] "Privacy-preserving inference in ML services using trusted execution environments," arXiv:1912.03485, 2019.
[4] "A performance analysis of VM-based trusted execution environments for confidential federated learning," arXiv:2501.11558, 2025.
[5] Microsoft, "Open Enclave SDK." `https://openenclave.io/sdk/`
[6] F. Mo et al., "DarkneTZ: Towards model privacy at the edge using TEEs," *Proc. ACM MobiSys*, 2020.
[7] F. Tramèr and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," *ICLR*, 2019.

Contact: Madhav (Sai) Suri | Surimadhav06@gmail.com | madhavsuri.com

Acknowledgments. Prototype Scope
This work presents a research prototype, not a production deployment. The system includes a Python host pipeline, C/C++ enclave-side partition module, unified enclave interface, remote-attestation flow, and benchmark harness for measuring inference overhead across five representative workloads.