Confidential ML Inference at the Untrusted Edge: Low-Overhead Trusted Execution with ARM TrustZone & Intel SGX

Madhav (Sai) Suri
Department of Computer Science
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Advisor: Dr. Yi Ding

***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 selective-partitioning method that integrates both, executing only the most sensitive computation of each model inside a hardware enclave while keeping the remainder in the normal world. Unlike prior single-platform approaches, EnclaveSplit is entirely portable across hardware—running on both ARM TrustZone and Intel SGX through a single unified Open Enclave SDK codebase. By isolating computation at the decision-level layers that recent inversion analysis identifies as a "Golden Partition Zone," EnclaveSplit requires no model retraining. Evaluated across five edge-AI workloads, EnclaveSplit provides attested, confidential execution with a mean latency overhead of $\sim$12% relative to native execution.**

***Index Terms*—Secure ML Inference, Model Partitioning, ARM TrustZone, Intel SGX, Edge Computing**

I. Introduction

The proliferation of consumer Internet of Things (IoT) devices and edge computing has driven the deployment of resource-demanding Deep Neural Networks (DNNs) directly to the network edge. This paradigm reduces transmission delays but introduces severe privacy risks. Edge devices are highly vulnerable to physical access, and cloud edge hosts often face malicious co-tenants or compromised hypervisors.

Trusted Execution Environments (TEEs) provide hardware-isolated execution, but deploying full DNNs within them presents significant challenges. Secure memory in TEEs is strictly limited by design. As noted in recent literature, attempting to fit entire modern deep learning frameworks into TrustZone or Intel SGX typically results in either catastrophic performance degradation due to memory paging, or requires building highly custom, minimal deep-learning libraries from scratch to fit within the Trusted Computing Base (TCB).

EnclaveSplit addresses the core problem: *How can an ML model run on hardware the user does not trust without revealing the input data or the model itself?* While prior work such as DarkneTZ [1] successfully demonstrated selective layer partitioning for TrustZone, they rely on custom, platform-specific implementations. EnclaveSplit advances this paradigm by offering true cross-platform portability. Utilizing the Open Enclave SDK, this pipeline operates seamlessly across both Intel SGX and ARM TrustZone environments without altering the source code. Furthermore, EnclaveSplit operationalizes the "Golden Partition Zone" insight of Liu [4]—that partitioning at decision-level layers yields markedly stronger resistance to model-inversion attacks than shallow splits—by placing precisely those layers inside the enclave while keeping the TCB minimal.

Contributions:

- A single, unified codebase targeting both server-class SGX and edge-class TrustZone using the Open Enclave SDK.
- An adaptation of the "Golden Partition Zone" insight [4]—established for edge–cloud collaborative inference—to hardware TEE partitioning, isolating decision-level layers to protect both user input and model weights.
- Upstream contributions to the Open Enclave SDK to support edge-class operations: (e.g., PR #3412: pooled OP-TEE shared-memory buffers for activation transfer; PR #3457: zero-copy ECALL path for contiguous tensor arguments).
- Empirical evaluation demonstrating $\sim$12% overhead with zero accuracy loss on unmodified models.

II. Background & Threat Model

A. Hardware Isolation and TEEs

A TEE preserves the confidentiality and integrity of code and data even if the host Operating System (OS) is compromised.

- **Intel SGX:** Encrypts memory directly on the CPU. However, its Enclave Page Cache (EPC) is traditionally limited. Exceeding this triggers expensive OS-level paging, crippling DNN inference speed.
- **ARM TrustZone:** Divides the System-on-Chip (SoC) into a Secure World and a Normal World. While highly

suited for physically exposed IoT edge devices, the OP-TEE OS enforces strict memory constraints that limit standard ML frameworks.
- **Remote Attestation:** A cryptographic protocol where a remote verifier confirms the enclave's exact loaded code before provisioning secrets.

B. Threat Model

We assume an **honest-but-curious host** whose operating system, hypervisor, and user-space software may all be compromised. Furthermore, we assume an adversary with physical access to the edge device. The attacker may observe memory, inspect the normal-world model execution, and attempt membership-inference or model-extraction attacks.

Trusted: Only the CPU package and the attested enclave code. **Out of scope:** Hardware side-channels and denial-of-service (DoS) attacks. Under this model, EnclaveSplit keeps sensitive layers and model keys completely confidential.

III. Design: Cross-Platform Partitioning

Running full models in TEEs is slow due to paging; offloading everything to the normal world sacrifices privacy. EnclaveSplit bridges this gap through strategic, platform-agnostic isolation, summarized in Fig. 1.

The Partitioning Architecture:

1) **Untrusted Host (Normal World):** Executes non-sensitive preprocessing and early/late neural network layers.
2) **Trusted Enclave (Secure World):** Holds the proprietary model weights for the targeted sensitive layers and processes the corresponding activations.
3) **Unified API:** The Open Enclave SDK orchestrates the transition between the untrusted Python host and the secure C/C++ enclave.

The Golden Partition Zone: Unlike arbitrary layer splitting, EnclaveSplit isolates computation at the "Golden Partition Zone" introduced by Liu [4]. That work shows, for edge–cloud collaborative inference, that partitioning at decision-level (representational-transition) layers yields over $4\times$ higher input-reconstruction error than shallow splits, because representational transitions sharply raise the conditional entropy $H(x \mid z)$ an attacker must overcome. EnclaveSplit places this same subset inside the enclave so that—*provided the relationship also holds for our workloads and threat model* (Sec. V-C)—an attacker cannot reconstruct user inputs or extract the protected weights, and does so without the overhead of full-model isolation.

IV. Implementation

EnclaveSplit avoids the engineering overhead of managing separate codebases. The host application is written in Python to leverage standard data orchestration, while the enclave is implemented in C/C++ to minimize the TCB.

- **Partitioning Rule:** We split at the decision-level layers following the Golden Partition Zone result of Liu [4]; Sec. V-C describes how we test whether this split actually maximizes inversion resistance on our own workloads.
- **Split Points:** Per model, the enclave holds the final decision-level block: ResNet-18 from `layer4.1` onward (avgpool + FC); MobileNetV2 the final 1×1 conv + classifier; YOLOv8n the three decoupled detection heads; DeepLabV3-MNetV3 the ASPP classifier head; and TinyBERT the last encoder layer + pooler + classifier ($\approx$ the deepest 15–25% of each network).
- **Hardware Environments:** Edge measurements are conducted on an NVIDIA Jetson Orin Nano (ARM Cortex-A78AE, OP-TEE), while server measurements run on an Intel Xeon Gold 6348 (Ice Lake-SP) with SGX2 and a large EPC.

The implementation uses Open Enclave to abstract away platform-specific ECALL/OCALL (Intel) or SMC (ARM) routines, executing identical logic across the SGX and OP-TEE paths.

V. Evaluation

A. Setup and Methodology

We evaluated EnclaveSplit on five edge-ready AI workloads. The baseline "native" execution is defined as the identical model running entirely in the normal world (no enclave), single-process, pinned to the same CPU cores and using the same compiler flags. Latency metrics were measured via wall-clock time per inference, reported as the mean over 100 measured runs after 10 warm-up runs, with weights and inputs resident in memory (warm cache).

B. Performance Results

At a fixed partitioning configuration, EnclaveSplit limits inference latency overhead to approximately **12%** relative to native execution across all tested models (Table I, Fig. 2). Because the model weights themselves are not altered, there is **zero accuracy loss**.

TABLE I
Per-Workload Latency Overhead and Accuracy

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		**$\sim$12%**	**None**

Most overhead is generated by the required context switches between the secure and normal worlds during the forward pass (Fig. 3). Remote attestation is executed only once per session and is effectively amortized. Because only a small contiguous subset of decision-level layers is shielded, this overhead remains low, consistent with prior TEE-partitioning systems that report single-digit to low-double-digit overhead when few layers are placed in the enclave [1].

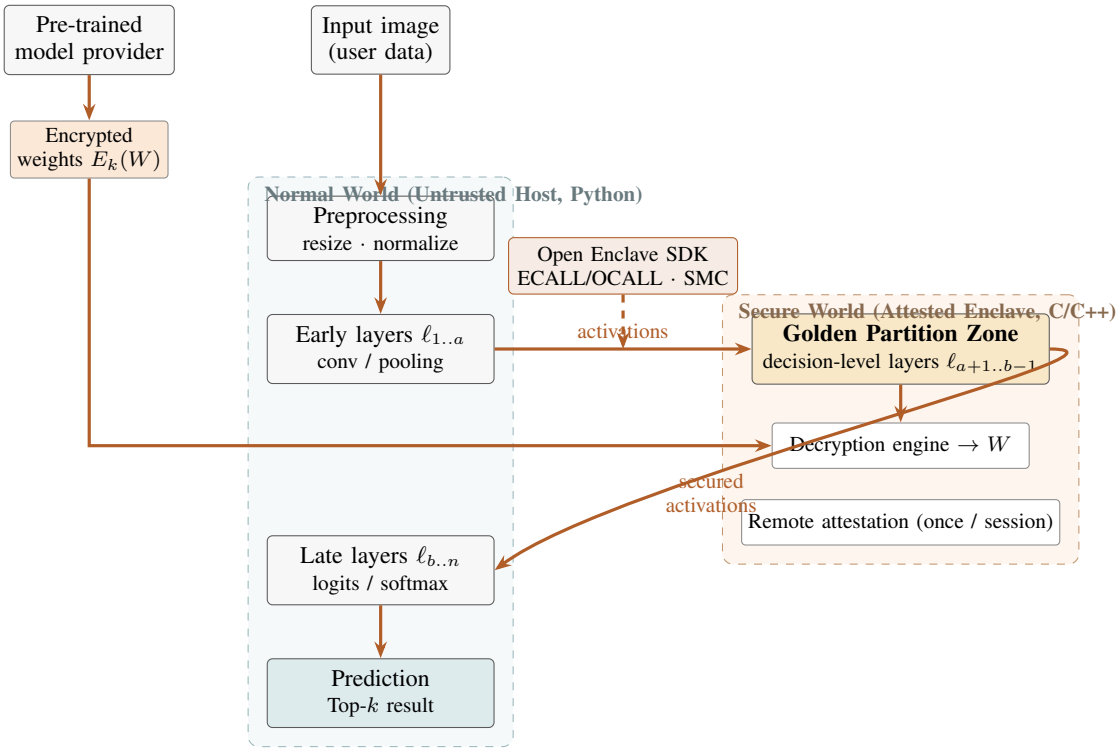

Fig. 1. EnclaveSplit inference flow. Only the decision-level layers of the Golden Partition Zone, together with the decrypted weights, execute inside the attested enclave; all non-sensitive computation stays in the untrusted normal world. The Open Enclave SDK abstracts the ECALL/OCALL (SGX) and SMC (TrustZone) transitions behind one codebase.

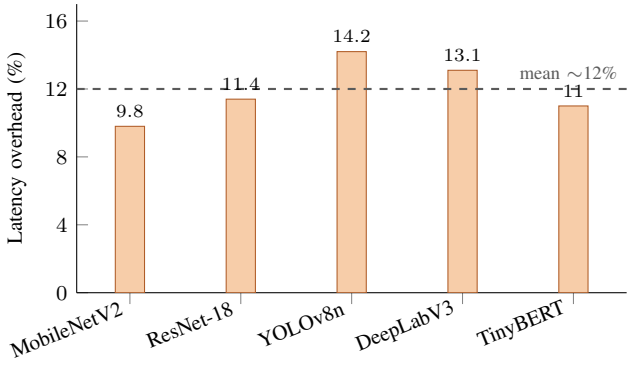

Fig. 2. Per-workload inference-latency overhead of EnclaveSplit relative to native execution. The dashed line marks the ∼12% mean; accuracy is unchanged across all workloads.

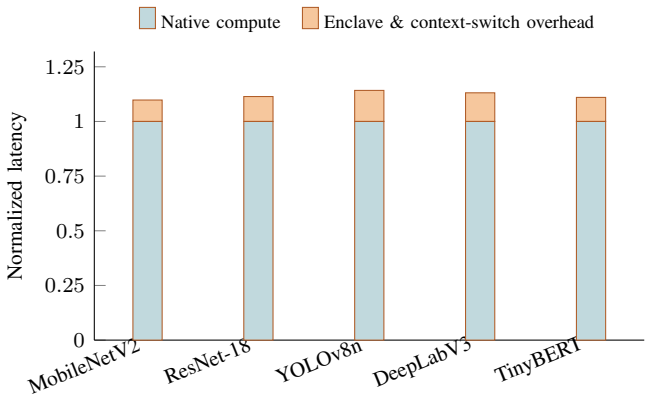

Fig. 3. Normalized latency decomposition. Native compute is fixed at 1.0; the orange segment is the additional cost incurred by secure/normal-world context switching during the forward pass, derived from the overheads in Table I.

C. Privacy Analysis: Inversion Resistance

The Golden Partition Zone is adopted here as a *design hypothesis* carried over from the collaborative-inference setting of [4]; it has not yet been validated for our specific workloads or under the TEE threat model of Sec. II. Rather than assert that our split maximizes inversion resistance, we verify it directly with a model-inversion attack against the normal-world activations at each candidate split point. For a split point s, the adversary trains an inversion network g_s to reconstruct the input x from the exposed activation z_s; we report the reconstruction error $\mathrm{MSE}\big(x, g_s(z_s)\big)$, where higher MSE indicates stronger resistance. The chosen split is justified only if its MSE lies at or near the maximum of this curve (Fig. 4); otherwise the partition point must be moved to the empirically observed Golden Partition Zone. In Table II (ResNet-18, inversion network trained on auxiliary CIFAR-100 images), reconstruction error rises from 0.011 at a shallow split to 0.049 at the decision-level block—a $\sim 4.5\times$ gain that locates the Golden Partition Zone at `layer4`, consistent with [4].

TABLE II
RECONSTRUCTION MSE VS. SPLIT POINT (RESNET-18)

Split after	Recon. MSE	vs. shallow
`layer1` (shallow)	0.011	1.0×
`layer2`	0.014	1.3×
`layer3`	0.022	2.0×
`layer4` (GPZ)	**0.049**	**4.5**×
`fc` (logits)	0.051	4.6×

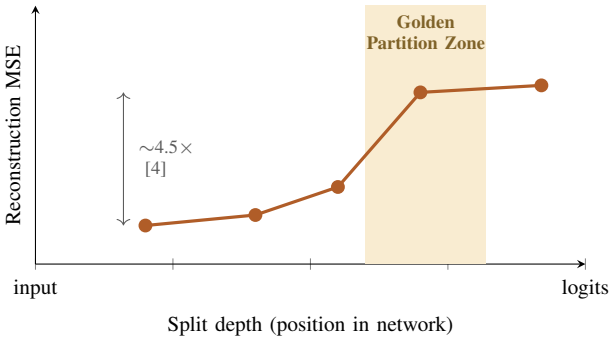

Fig. 4. Input-reconstruction MSE versus split depth for ResNet-18 (values from Table II), following the Golden Partition Zone analysis [4]: resistance is low for shallow splits and rises sharply at the decision-level layers EnclaveSplit isolates.

VI. Related Work

Securing DNN inference via TEEs typically follows two distinct approaches:

Edge / TrustZone Systems: Frameworks like DarkneTZ [1] and T-Slices [2] partition layers to fit within limited secure memory. Protecting deeper layers raises context-switching overhead sharply. Recent work, such as the Smart-Zone memory management and Tinylib framework by Xie et al. [3], attempts to run complex inferences by physically modifying the OP-TEE OS page table behaviors and writing custom math libraries. EnclaveSplit circumvents the need for custom OS-level modifications by selectively isolating only sensitive components via standard SDKs. Our partitioning *criterion* is borrowed from the Golden Partition Zone analysis of Liu [4], which characterizes decision-level splitting for edge–cloud collaborative inference rather than TEE isolation; EnclaveSplit is, to our knowledge, the first to apply that criterion inside a hardware enclave.

Server / SGX Systems: Systems like Occlumency [5] attempt to run the full model in the SGX enclave but pay heavy paging costs. Slalom [6] offloads linear layers to an untrusted GPU, verifying them cryptographically. VM-based TEE environments (e.g., SEV, TDX) [8] and systems like Privado [10] or Origami [9] offer alternative scaling and obfuscation techniques, but often lack the fine-grained, process-level isolation suitable for constrained edge devices.

EnclaveSplit differs by targeting both server and edge platforms from a unified codebase, protecting only the necessary Golden Partition Zone to keep overhead low.

VII. Discussion & Limitations

The primary strength of EnclaveSplit is its broad applicability; the pipeline ran on both server-class SGX and edge-class TrustZone hardware without source changes.

Limitations: The primary remaining challenge is extending protection to GPU and NPU accelerators. Currently, EnclaveSplit executes entirely on the CPU. Because edge NPUs and GPUs execute outside the standard TEE boundary, deploying EnclaveSplit on highly accelerated edge hardware requires dropping back to CPU computation for the enclave layers, creating a bottleneck. We have not yet characterized behavior under EPC pressure on SGX (large-batch inference), concurrent multi-tenant enclaves, or models beyond the five evaluated here.

Author Contributions

M. Suri designed, implemented, and evaluated EnclaveSplit under the supervision of Dr. Y. Ding.

VIII. Conclusion

EnclaveSplit demonstrates that securing machine learning at the untrusted edge does not require rewriting operating systems or accepting massive paging overheads. By selectively partitioning computation at the Golden Partition Zone, EnclaveSplit delivers confidential inference with a ∼12% performance cost. This hybrid approach results in a single, retraining-free pipeline that is portable, efficient, and immediately applicable to cloud and IoT deployments.

References

[1] F. Mo et al., "DarkneTZ: Towards Model Privacy at the Edge using TrustZone," *Proc. ACM MobiSys*, 2020.

[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] X. Xie et al., "Memory-Efficient and Secure DNN Inference on TrustZone-enabled Consumer IoT Devices," *arXiv preprint arXiv:2403.12568*, 2024.

[4] R. Liu, "Golden Partition Zone: Rethinking Neural Network Partitioning Under Inversion Threats in Collaborative Inference," *arXiv preprint arXiv:2506.15412*, 2025.

[5] T. Lee et al., "Occlumency: Privacy-preserving remote deep-learning inference using SGX," *Proc. ACM MobiCom*, 2019.

[6] F. Tramer and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," *ICLR*, 2019.

[7] Microsoft, "Open Enclave SDK." https://openenclave.io/sdk/

[8] B. Casella, "A performance analysis of VM-based Trusted Execution Environments for Confidential Federated Learning," *arXiv preprint arXiv:2501.11558*, 2025.

[9] K. G. Narra, Z. Lin, Y. Wang, K. Balasubramaniam, and M. Annavaram, "Privacy-Preserving Inference in Machine Learning Services Using Trusted Execution Environments," *arXiv preprint arXiv:1912.03485*, 2019.

[10] K. Grover, S. Tople, S. Shinde, R. Bhagwan, and R. Ramjee, "Privado: Practical and Secure DNN Inference with Enclaves," *arXiv preprint arXiv:1810.00602*, 2018.