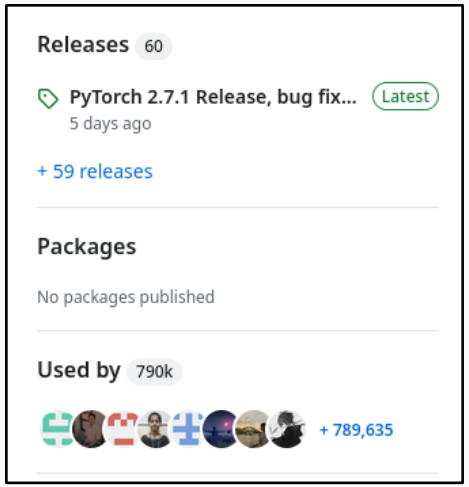
Protecting Deep Learning Models on Cloud Platform with Trusted Execution Environments

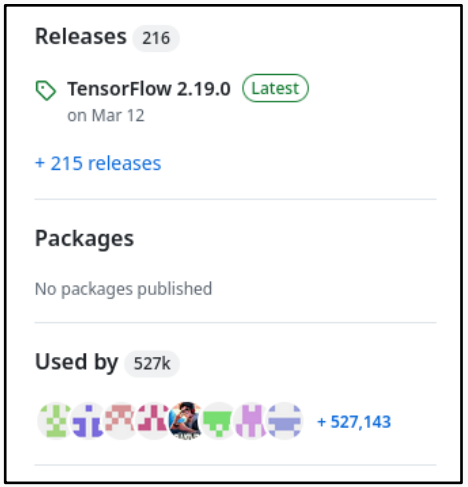
Kripa Shanker
PhD Candidate
Supervisor, Prof. Vinod Ganapathy

Computer Systems Security Laboratory

IISc Bangalore

Deep Learning



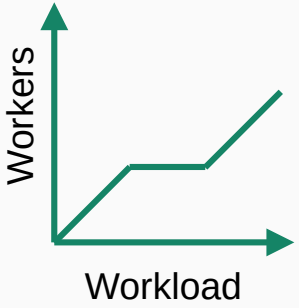


Two popular deep learning frameworks are used in more than 1M+ projects.

Many organizations lack resources to run expensive DL workloads in-house.

Public Cloud Platforms

Benefits



Scale With Demand

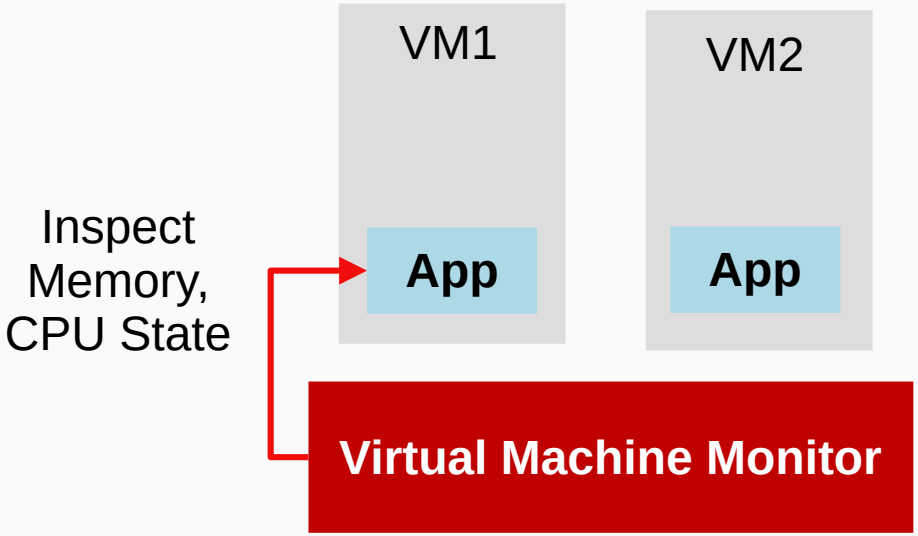
Low Maintenance

Security Upgrades

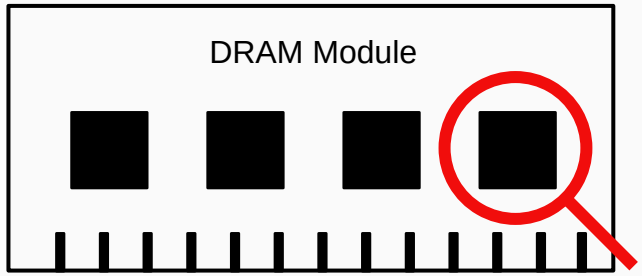
Lower Cost

Attacks on Cloud Workloads

Software Attacks



Physical Attacks



Cold boot attacks
Interfaces snooping attacks

1. LibVMI, A python library for virtual machine introspection.
2. Cold Boot Attacks are Still Hot: Security Analysis of Memory Scramblers in Modern Processors, HPCA 2017

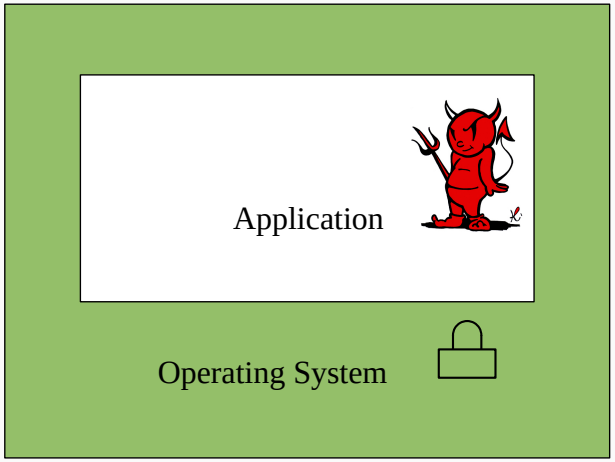
Deep Learning Models

Why protect deep learning modes?

- Well-trained models provide a competitive edge to businesses.
- Training state-of-the-art models is expensive.
- White-box access to the model introduces security and privacy risks.

Explaining and harnessing adversarial examples, ICLR 2015
Membership Inference Attacks against Machine Learning Models, IEEE S&P 2017

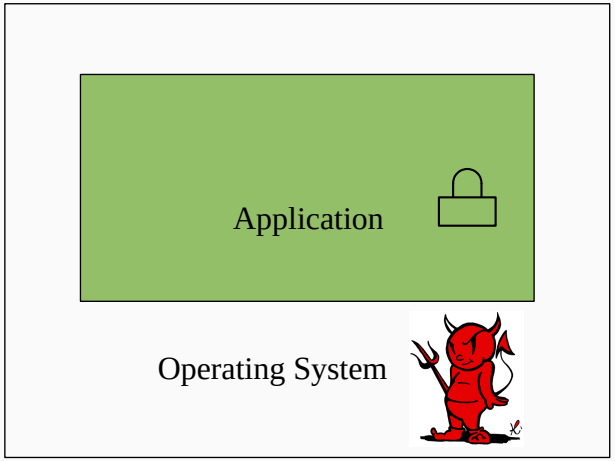
Solution: Trusted Execution Environments



Traditional Security Model

Protects the **host** from the guest.

E.g., Virtual Machines, CPU protection rings, containers, and sandboxes.



TEE Security Model

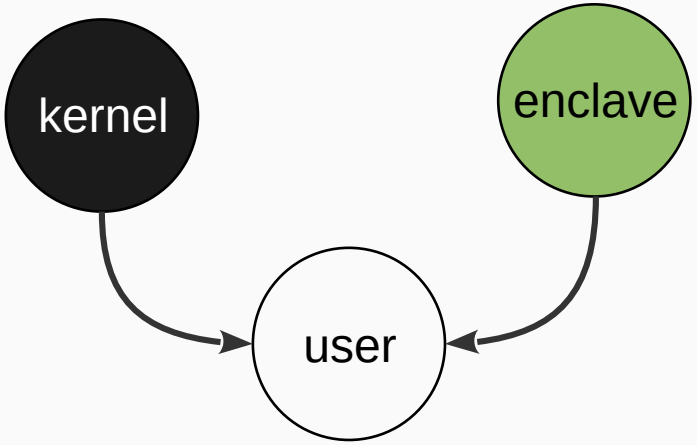
Protects **the guest** from the host.

E.g., Intel SGX, Intel TDX, AMD SEV, and ARM CCA.

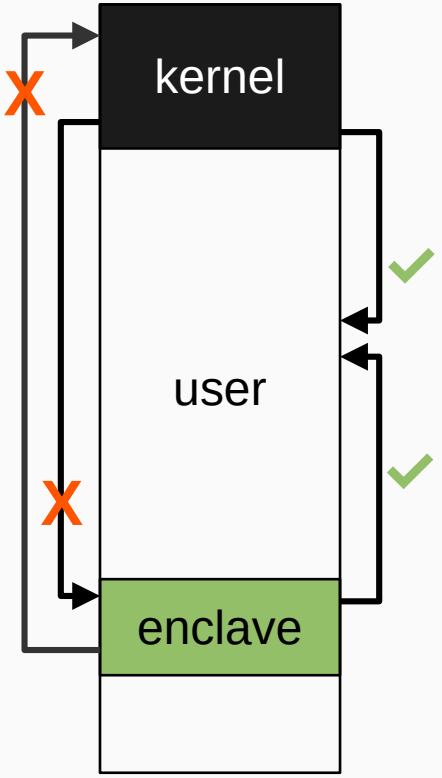
Intel SGX (Software Guard Extensions)

A set of new instructions.

```
ECREATE
EADD
EEXTEND
EINIT
EENTER
EEXIT
ERESUME
EREMOVE
EGETKEY
EREPORT
```



A new (enclave) mode of CPU execution.



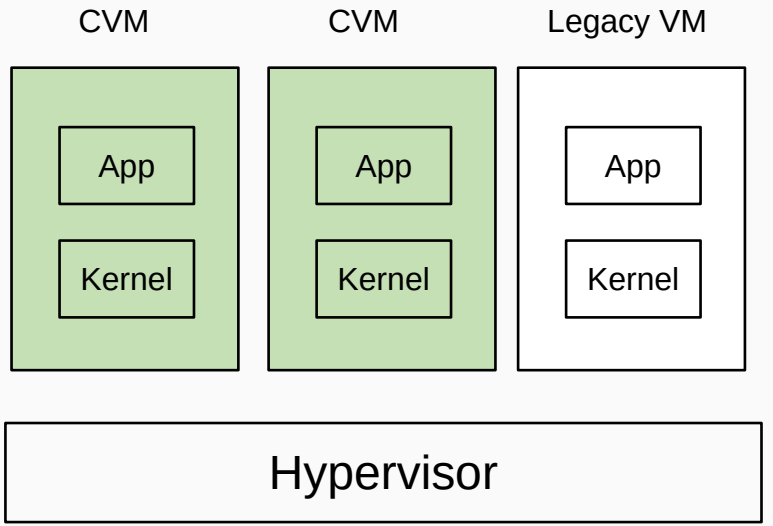
Changes to memory access semantics.

Confidential Virtual Machines

Runs an entire virtual machine with a TEE.

Implementations:

- AMD Secure Encrypted Virtualization (SEV)
- Intel Trust Domain Extensions (TDX)
- Arm Confidential Compute Architecture (CCA)



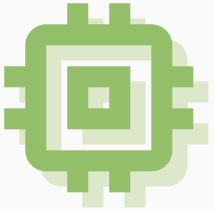
Thesis Statement

Hardware-based trusted execution environments can be leveraged to run private deep learning inference workloads on public cloud platforms with practical runtime performance while protecting the privacy and integrity of the model.

SGX Threat Model

Trusted

- Intel CPU
- Code and Data Within SGX Enclaves



Enclave

Untrusted

- Operating System
- Hypervisor
- BIOS
- Firmware
- Peripheral Devices
- System Buses

OS

Restricted Instructions

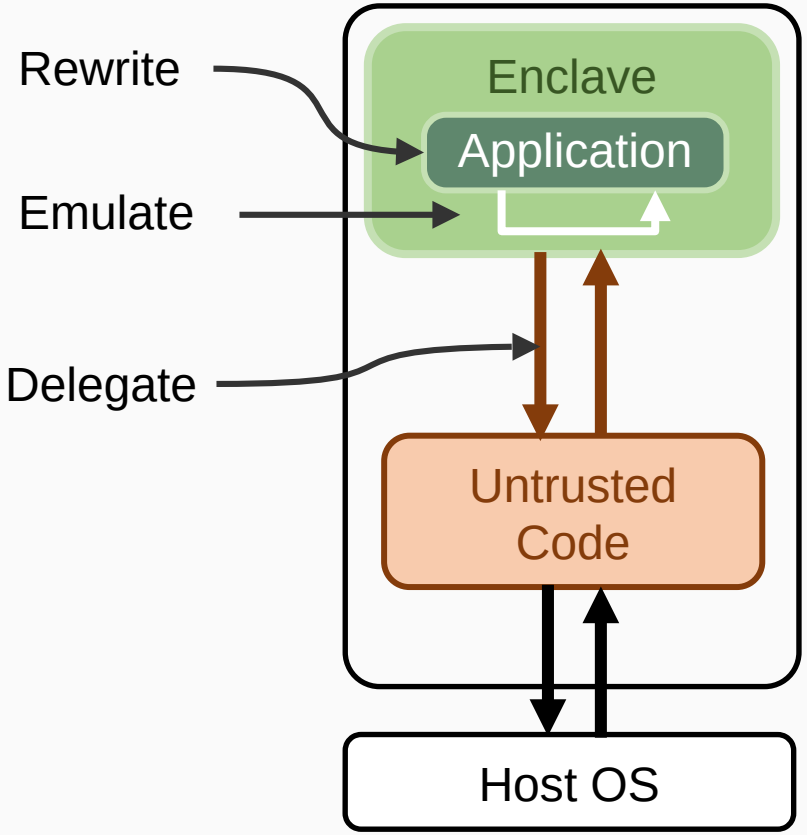
Instructions	Comment
CPUID, GETSEC, RDPMC, SGDT, SIDT, SLDT, STR, VMCALL, VMFUNC	Might cause VM exit.
IN, INS/INSB/INSW/INSD, OUT, OUTS/OUTSB/OUTSW/OUTSD	I/O fault may not safely recover.
IRET, LDS/LES/LFS/LGS/LSS, MOV to DS/ES/SS/FS/GS, POP DS/ES/SS/FS/GS, **SYSCALL**, SYSENTER	Access segment register could change privilege level.
LAR, VERR, VERW	Might provide access to kernel information.
ENCLU[EENTER], ENCLU[ERESUME]	Cannot enter an enclave from within an enclave.

Table: Restricted instructions within an enclave. [Intel Manual]

SYSCALL instruction is frequently used by most applications to request OS services.

Solution

Limit restricted instruction within enclaves



Frameworks to Port Applications to SGX

- Haven [OSDI 2014]
- Scone [OSDI 2016]
- Graphene-SGX [ATC 2017]
- Panolply [NDSS 2017]
- *lxcsgx* [CODASPY 2019]
- SGX-LKL

Which framework to use to port my application workload?

Part 1: An Evaluation of Methods to Port Legacy Code to SGX Enclaves

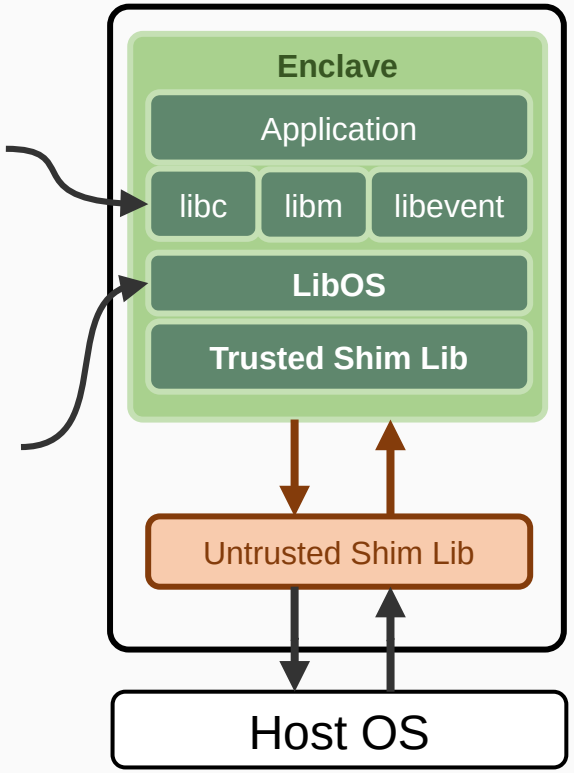
Method 1: Library OS

Runs an entire library OS within the enclave.

- Haven [OSDI 2014]
- Graphene-SGX [ATC 2017]
- SGX-LKL
- Occlum [ASPLOS 2020]

Dependencies are **inside** the enclave.

Library OS is **inside** of the enclave.



Method 2: Library Wrapper

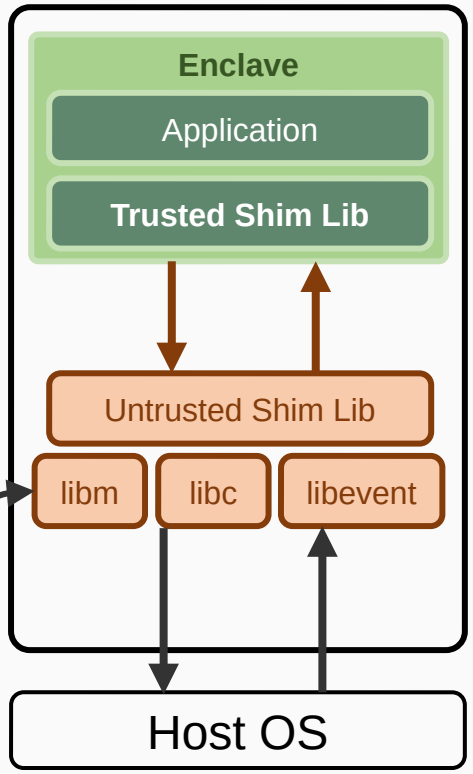
Implements wrappers for functions delegating restricted instructions.

Function wrapper

```
int read(int fd, char *buf, int len){
    return ocall_read(fd, buf, len);
}
```

- Panoply [NDSS 2017]

Dependencies are **outside** the enclave.



Method 3: Instruction Wrapper

Implements wrappers for restricted instructions.

- Scone [OSDI 2016]
- *lxcsgx* [CODASPY 2019]

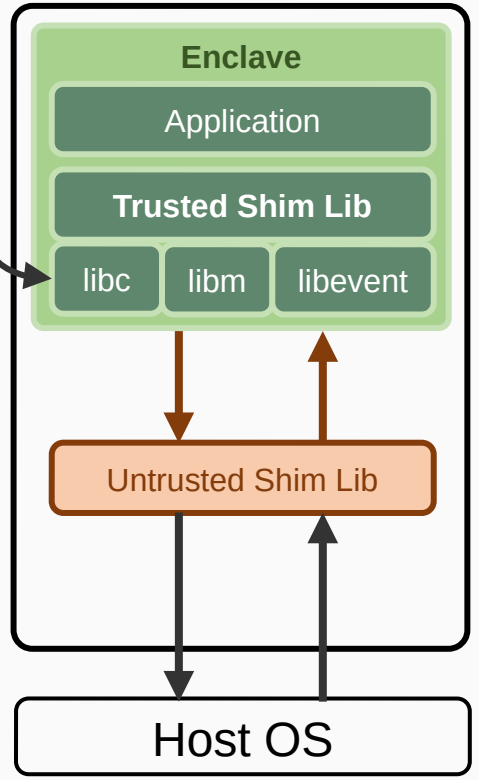
Restricted syscall instruction in libc

```
__syscall:
    movq %rdi,%rax
    movq %rsi,%rdi
    movq %rdx,%rsi
    movq %rcx,%rdx
    movq %r8,%r10
    movq %r9,%r8
    movq 8(%rsp),%r9
    syscall
    ret
```

Instruction wrapper

```
int __syscall(long arg0,
…){
    ...
    ocall_syscall()
    ...
}
```

Dependencies are **inside** the enclave.



Porpoise

- We built Porpoise to port applications to SGX enclaves.
- Porpoise implements wrapper around the `SYSCALL` instruction in *musl*.
- No modification to applications' source code.
- Publicly available at https://github.com/iisc-cssl/porpoise

Research Questions

RQ1. What is the effort required, e.g., code changes, to obtain a working enclave?

RQ2. What is the effort required to re-engineer a working enclave, e.g., by moving code out of or into the enclave?

RQ3. How much trusted code does each method require?

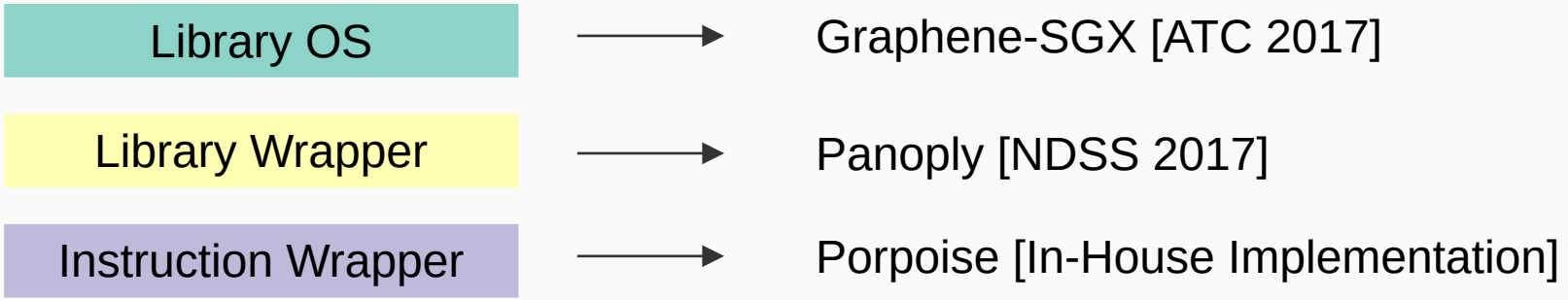
RQ4. What is the performance cost of each of the three methods?

Evaluation

Application	Version	Description
Bzip2	1.0.6	File compression utility
Cpython	3.7	Python interpreter in C
H2O	2.0	HTTP Web server
OpenSSL	1.0.1	OpenSSL Cryptographic library
Memcached	1.5.20	Key-value store

Benchmark application used in evaluation.

Representative Frameworks:



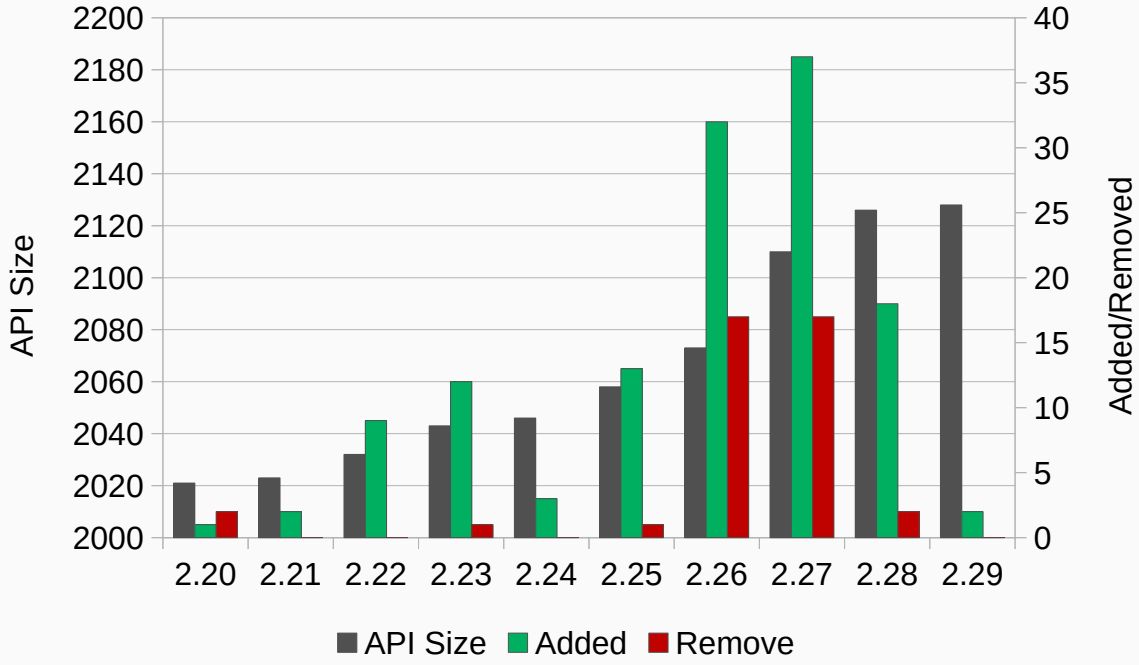
RQ1: Porting Effort

	Bzip2	Python	Memcached	H2O	OpenSSL
Library OS	✅	✅	✅	✅	✅
Library Wrapper	✅	❌	❌	✅	✅
Instruction Wrapper	✅	✅	✅	✅	✅

Challenges with library wrapper model: **Large evolving interface.**

	#Interfaces
Glibc	~2000
Support in Panoply	~250
Addition for Bzip2	10

Evolution of Glibc Interface



Evolution of Syscall Interface

Syscall API is **smaller** and has **few changes** than Glibc API.

Summary: Porting Effort

	Bzip2	Python	Memcached	H2O	OpenSSL
Library OS	✅	✅	✅	✅	✅
Library Wrapper	✅	❌	❌	✅	✅
Instruction Wrapper	✅	✅	✅	✅	✅

Library OS	→	Manifest File	→	Low (Binary Compatibility)
Library Wrapper	→	Implement Missing Wrappers	→	High
Instruction Wrapper	→	Recompile with Porpoise Libraries	→	Medium

RQ2: Re-engineering Effort

Move non-sensitive code or modules outside the enclave.

Bzip2

Execute compression and decompression algorithm within TEE.

- `BZ2_bzcompressInit()`
- `BZ2_bzCompress()`
- `BZ2_bzCompressEnd()`
- `BZ2_bzReadOpen()`
- `BZ2_bzRead()`

OpenSSL

Generate rsa cryptographic keys within enclave.

- genrsa
- ec
- SHA256
- base64
- aes

Cpython

Execute python interpreter loop within enclave.

- PyEval_EvalCode()
- PyArena_New()
- PyArena_Free()
- PyArena_Malloc
- read()

RQ2: Re-engineering Effort

Move non-sensitive code or modules outside the enclave.

e.g. Run only RSA key-generation algorithm within the enclave.

Library OS

Library Wrapper

Instruction Wrapper

Applications cannot be re-engineered

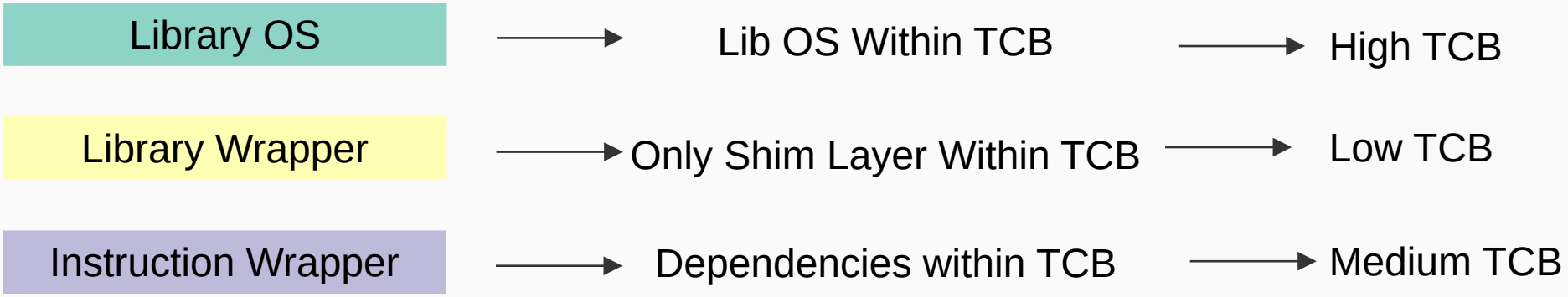
Application	#Interfaces	SLOC Added
bzip	3	29
OpenSSL	1	8
CPython	24	277

Same re-engineering efforts for Library Wrapper and Instruction Wrapper models.

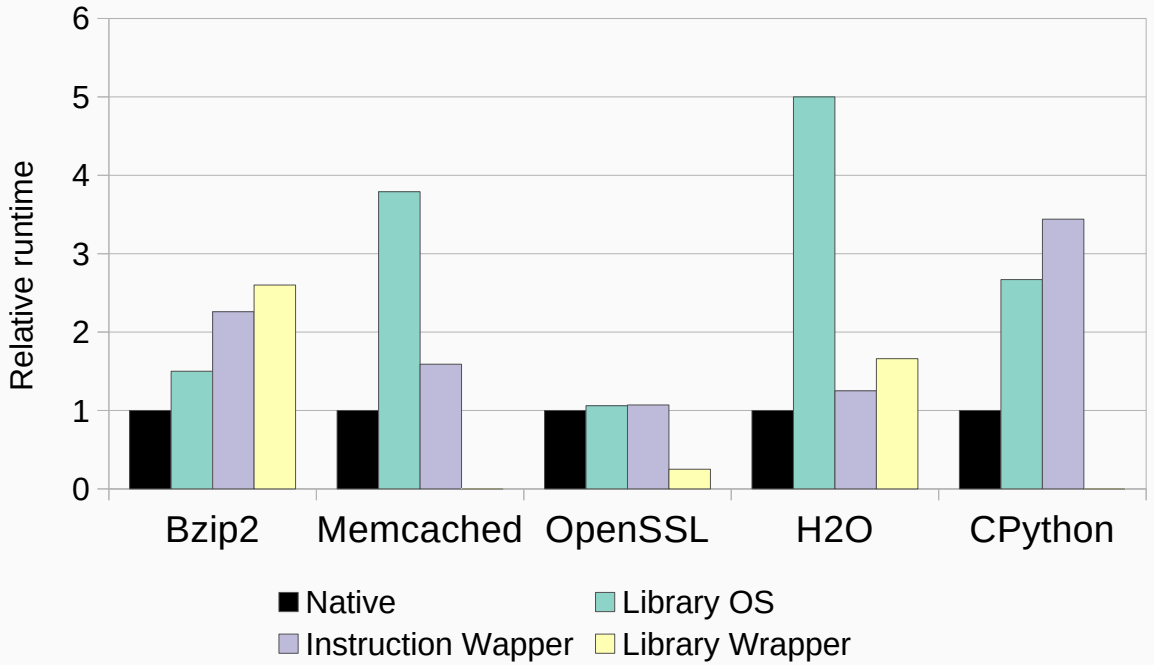
RQ3: Security

	Library OS	Library Wrapper	Instruction Wrapper
Lib OS	31,742	N/A	N/A
libc	1,222,912	N/A	82,978
shim	N/A	14,506	1,934
SDK	N/A	119,545	119,545

Lines of framework code that is part of the TCB.



RQ4: Runtime Performance

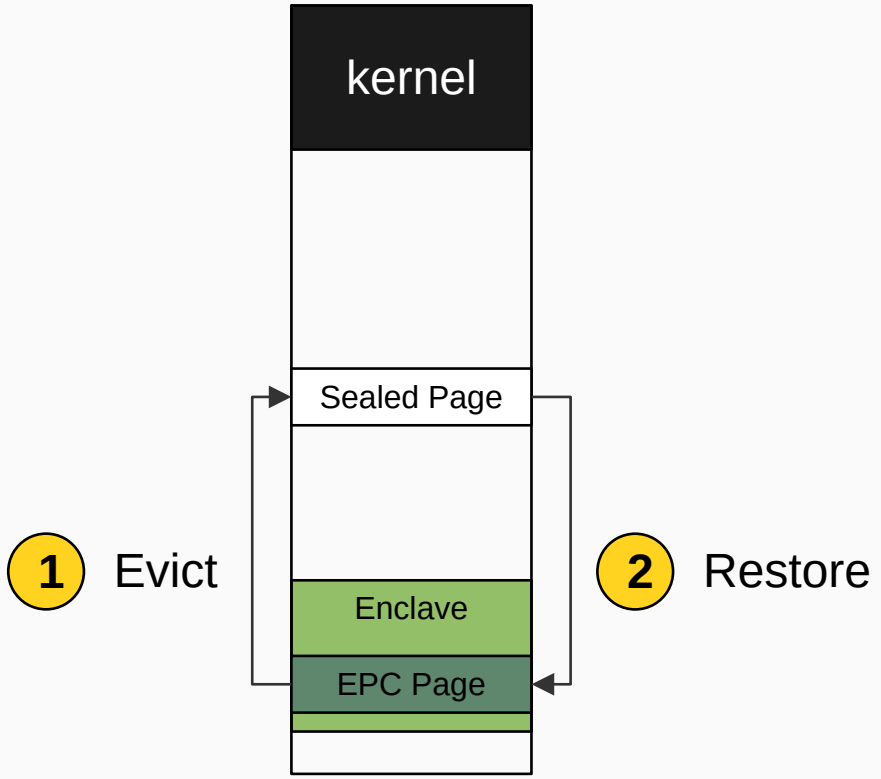


No clear winner in runtime performance.

Part 2: MazeNet: Protecting DNN Models on Public Cloud Platforms With TEEs

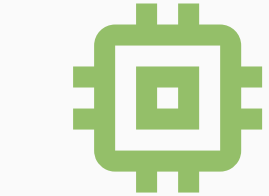
Challenge 1: Fixed Protected Memory

Intel SGXv1 offers 128 MB of cryptographically protected main memory.

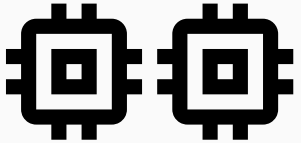


Memory-intensive enclave applications incur performance penalty.

Challenge 2: Underutilisation



CPU With TEE



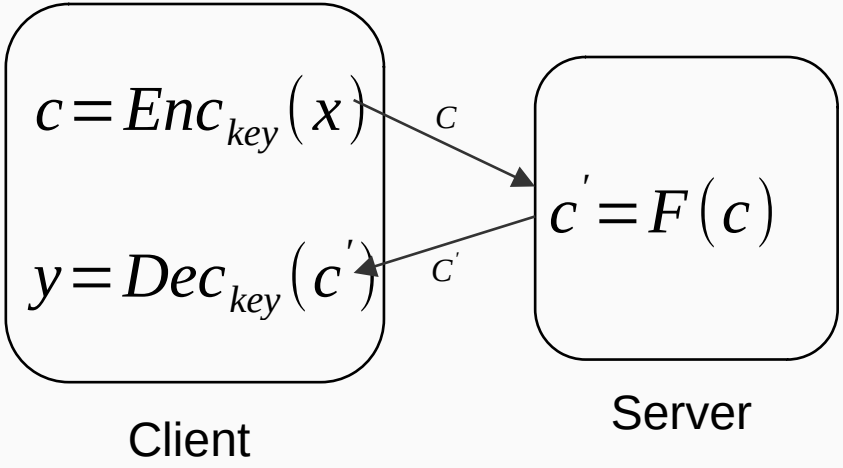
CPUs Without TEE

Hardware Accelerators Without TEE

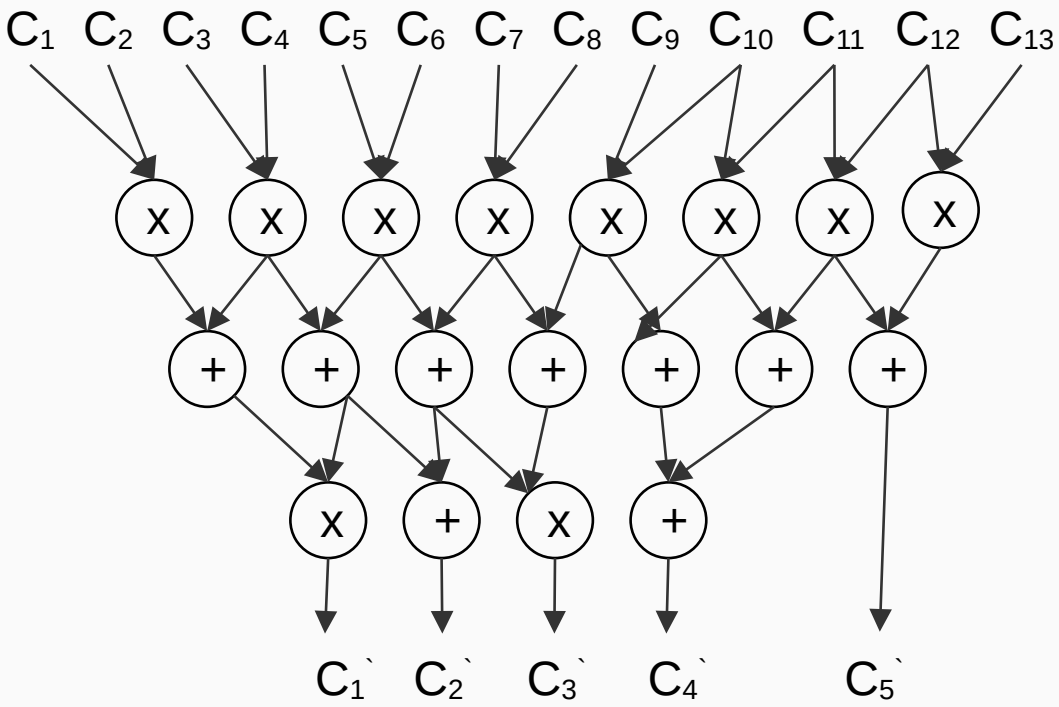
TEE cannot securely utilise untrusted resources.

Prior Approaches

Homomorphic Encryption



CryptoNets [PMLR2016]

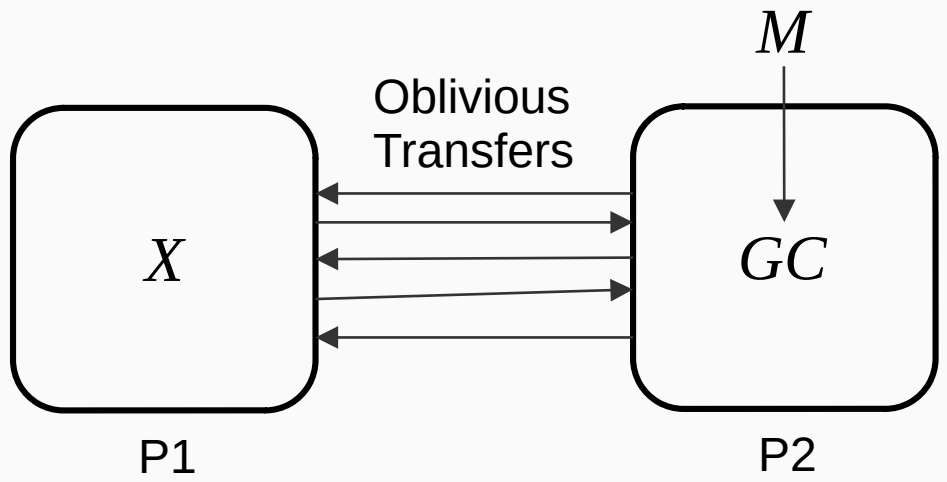


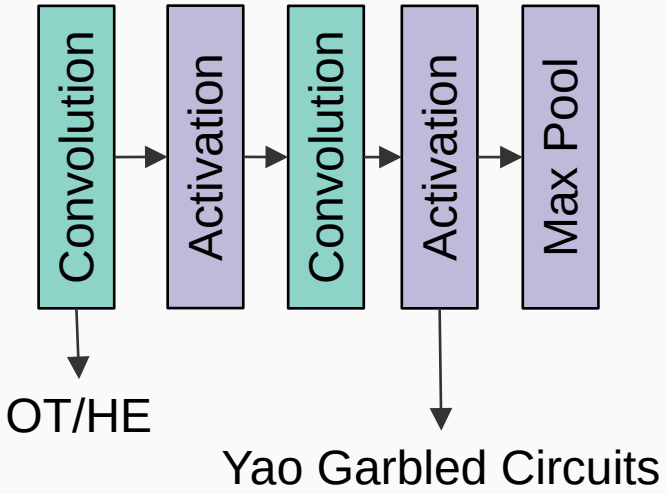
Arithmetic Circuit

Homomorphic Encryption is computationally expensive.

Prior Approaches

Cryptography-based Techniques: Multi-party Computation





DeepSecure [DAC 2017], SecureML [S&P 2017], CryptFlow [S&P 2020], Orca [S&P 2024]

MPC-based techniques incurs high communication cost.

Prior Approaches

Outsourcing Linear Layers to GPUs With Matrix Masking

Offline Phase

$$r_i \leftarrow F^m$$

$$u_i = r_i W_i$$

Online Phase

Blind Input

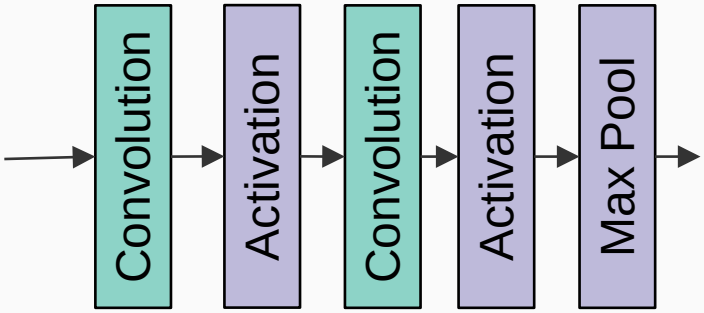
$$\bar{x}_i = x_i + r_i$$

Outsource

$$\bar{y}_i = \bar{x}_i W_i$$

Restore Results

$$y_i = \bar{y}_i - u_i$$



Model	Data Transferred (MB)		
	TEE → GPU	GPU → TEE	Total
VGG16	34.77	51.78	86.48
ResNet50	38.70	40.39	79.09
DenseNet201	91.43	29.96	121.39

Slalom [ICLR 2019], DarkNight [Micro 2021], ShadowNet [S&P 2023], TEESlice [S&P 2024]

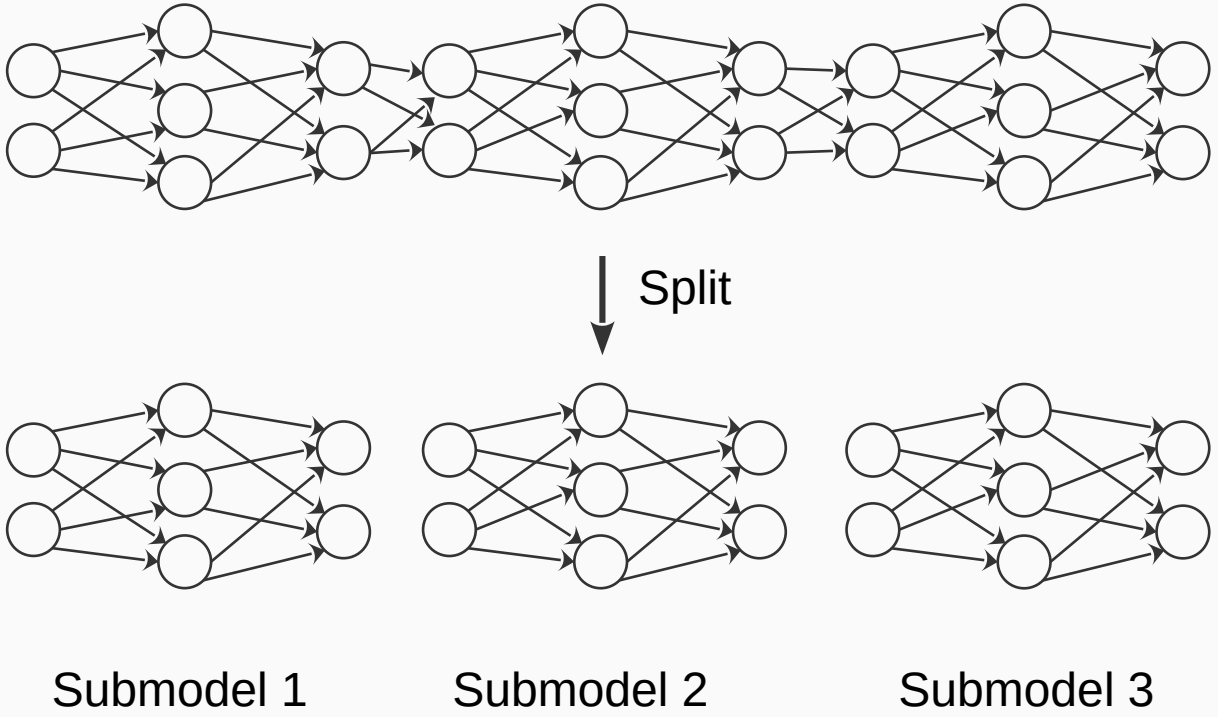
MazeNet

- Transforms pre-trained model into MazeNet models.
- Avoids performance penalty due to fixed protected memory of SGX.
- Improves performance with untrusted resources.
- Protects privacy of pre-trained model.

Method 1: Splitting

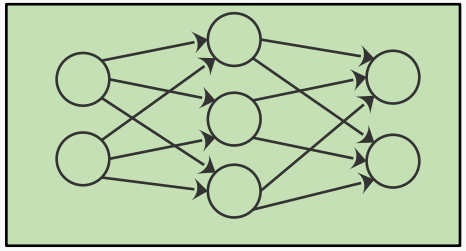
Large models incur performance penalty due to EPC swapping.

Solution: Pre-trained model is split into smaller submodels.

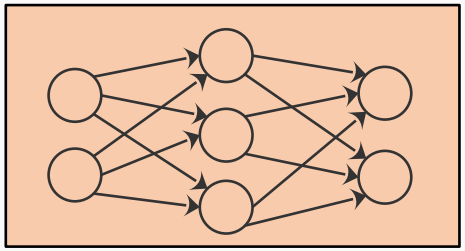


Method 2: Cloaking

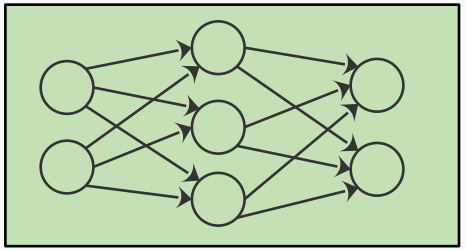
Running submodels only within TEEs results in underutilisation of the overall system.



In-TEE
Submodel 1

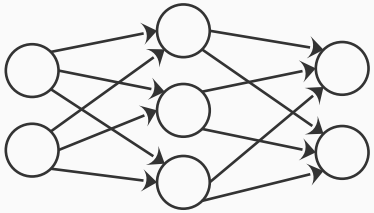


Non-TEE
Submodel 2

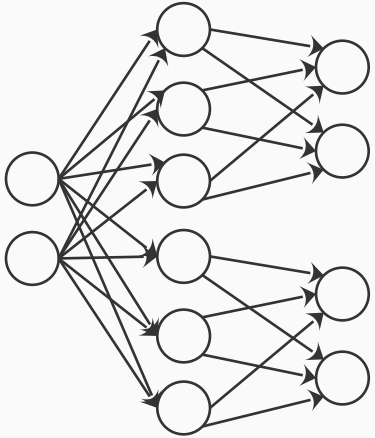


In-TEE
Submodel 3

Non-TEE sudmodels leak weights.

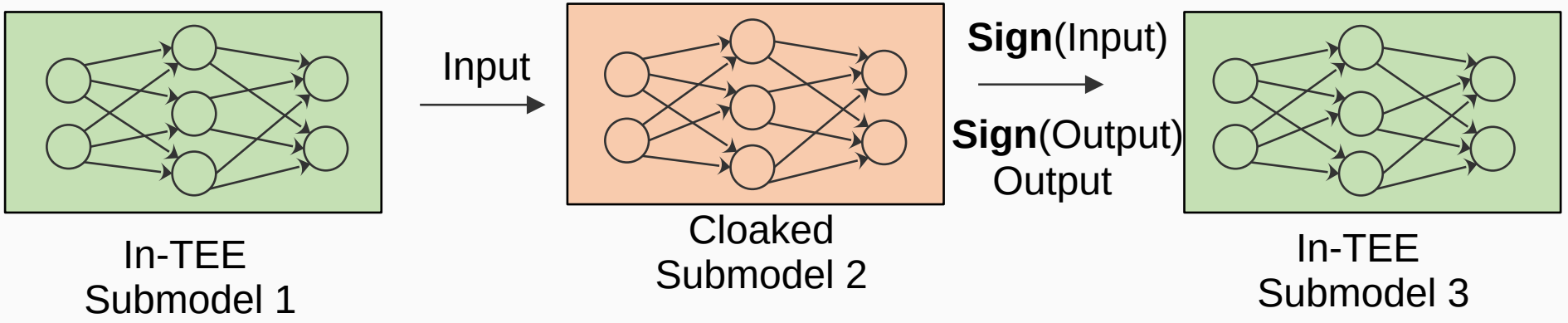


Cloak



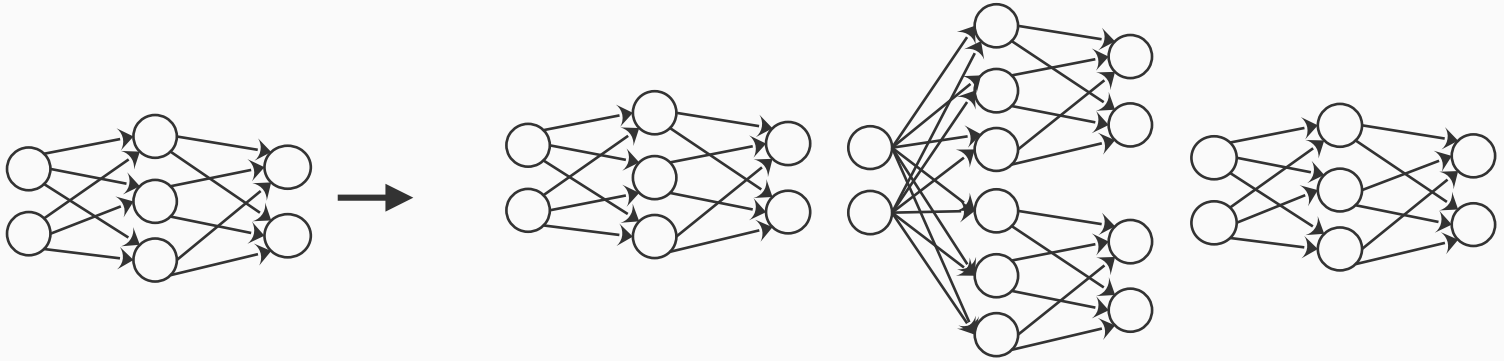
Method 3: Digital Signatures

Adversary can tamper with outsourced computation.



Cloud vendor signs the input and output of cloaked submodels with its private key.

MazeNet Workflow



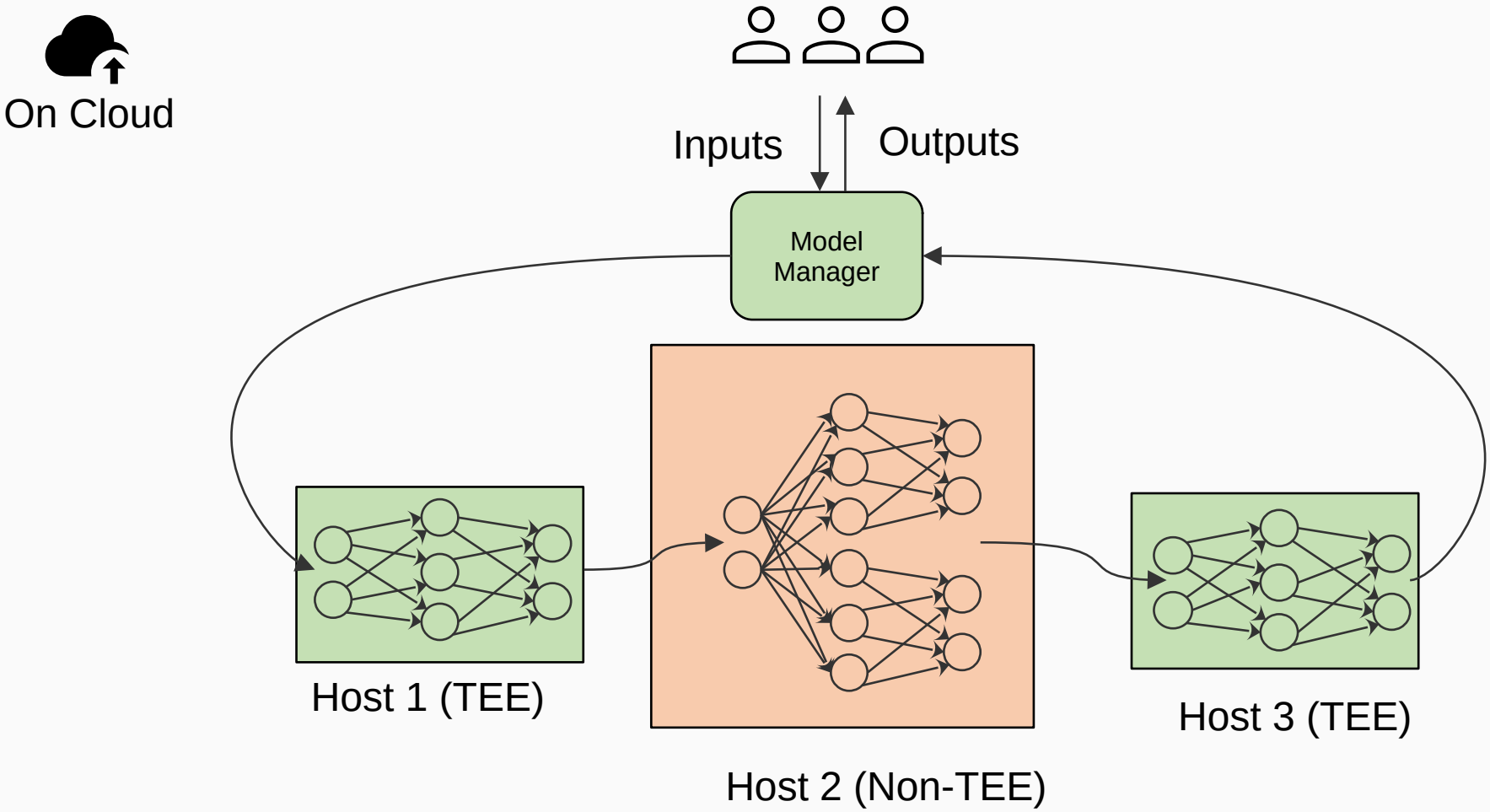
Pre-trained Model

MazeNet Model

On premises

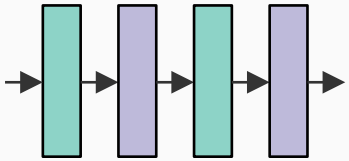
Model Builder

MazeNet Workflow

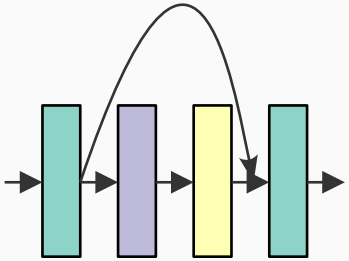


Benchmark Models

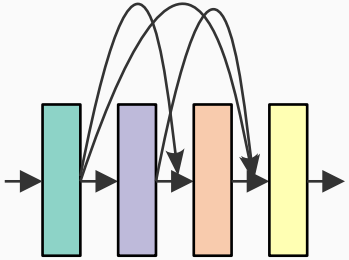
We evaluate the presented techniques on popular convolution neural networks.



VGG16
Sequential Architecture
Computationally expensive



ResNet50
Skip Connections



DenseNet201
Highly Connected Layers
Very deep model

Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015
Deep Residual Learning for Image Recognition, CVPR 2016
Densely Connected Convolutional Networks, CVPR 2017

Implementation

- **Model Builder**: Transforms pre-trained Model into MazeNet model.
- **Model Manager**: Deploy and runs MazeNet models.

Model	TensorFlow Layers
VGG16	Conv2D, Dense, MaxPooling2D, Flatten
ResNet50	Conv2D, Dense, BatchNormalization, Add, GlobalAveragePooling2D,MaxPooling2D, ZeroPadding2D, Activation
DenseNet201	Conv2D, Dense, BatchNormalization, Concatenate, GlobalAveragePooling2D, MaxPooling2D, ZeroPadding2D, AveragePooling2D

Cloaking support for various TensorFlow (v2.7) layers in MazeNet.

Configuration Parameters

Model	Cloak Factor	Submodel Width	In-TEE Layers
VGG16	10%	10	1, 11, 22
ResNet50	10%	10	1-7, 92-102, 174-177
DenseNet201	10%	10	1-7, 49-137, 477-709

Configuration parameters used to generate MazeNet models.

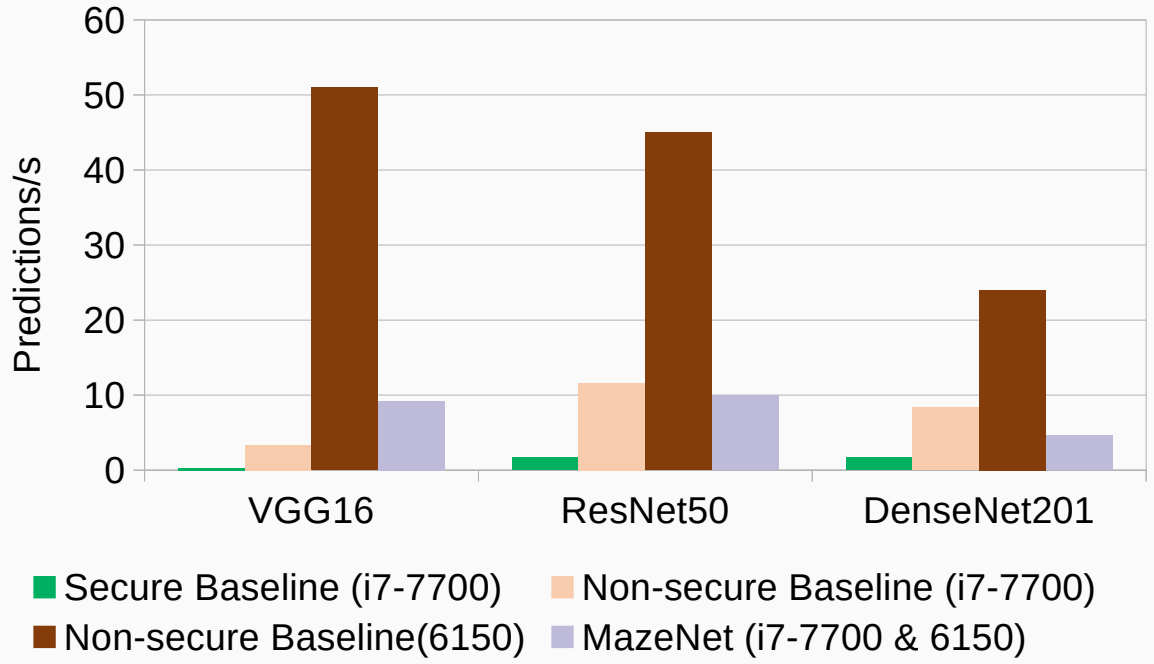
Experimental Setup

Three TEE Systems
- Intel i7-7700 CPU
- 4 Cores
- SGXv1
- 32 GB Main Memory

One Non-TEE System
- Intel Xeon Gold 6150 CPU
- 36 Cores
- 256 GB Main Memory

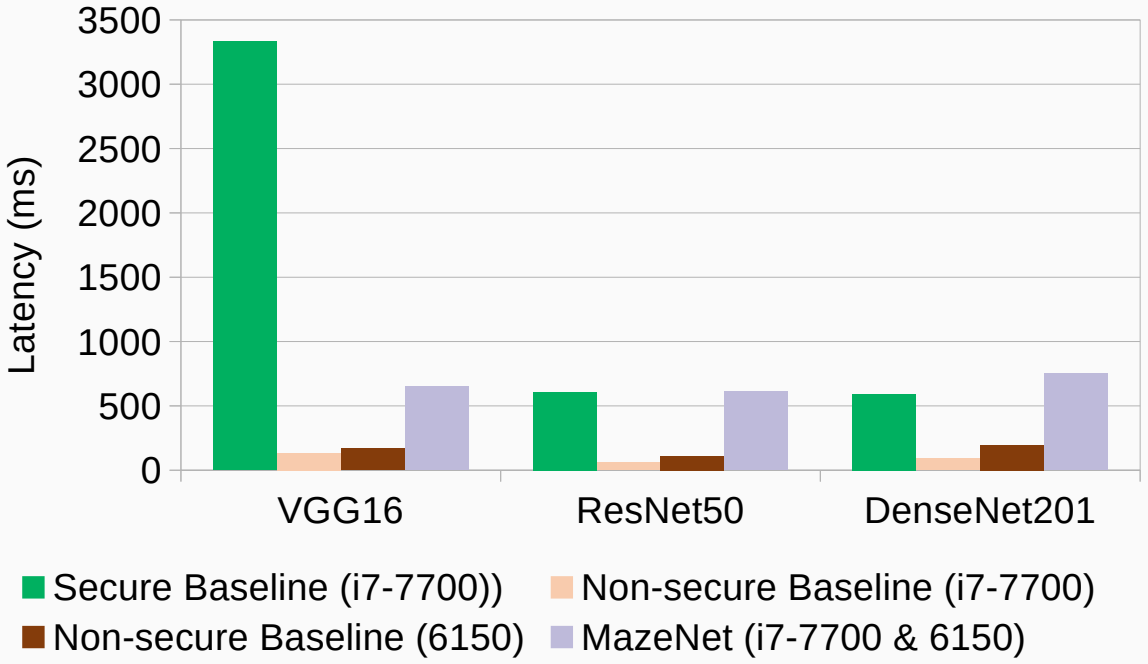
All the hosts are connected through one Gigabit Ethernet.

Throughput Results



MazeNet improves the throughput by 2x to 30x as compared to secure baseline.

Latency Results



MazeNet can improve the latency upto 5x as compared to secure baseline.

Overheads: Synthetic Computations

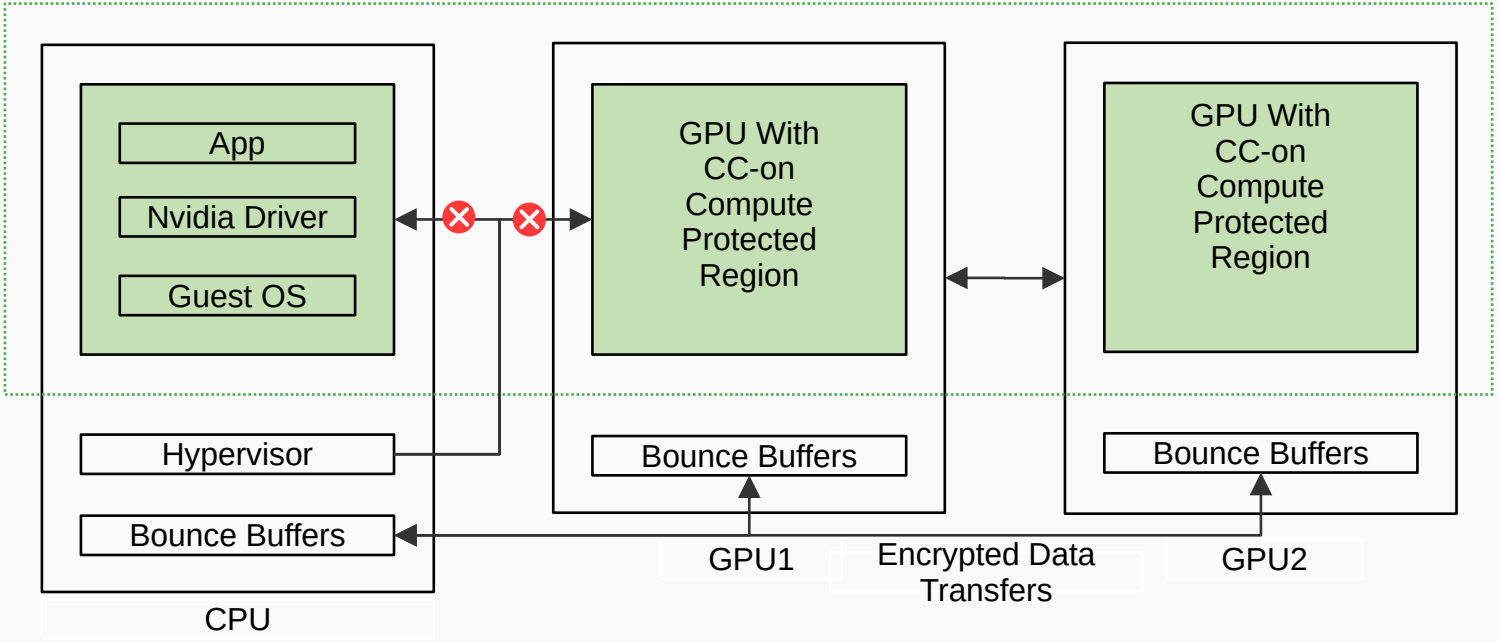
	Standard	MazeNet			Increase
		In-TEE	Cloaked	Total	
VGG16	30.96	1.85	151.76	153.6	5x
ResNet50	7.73	0.684	60.85	61.54	8x
DenseNet201	8.58	3.17	51.18	54.36	6x

Floating point operations (GFLOPs) in standard and MazeNet Models.

Limitations

- Some details of the user inputs are visible to the adversary.
- The model developer needs to provide the suitable split and architecture of synthetic layers.

Nvidia Confidential Compute



Extends CPU TEE trust boundary to GPUs.

Future Work

- Training support for MazeNet.
- Secure outsourcing of large language models to untrusted hardware.

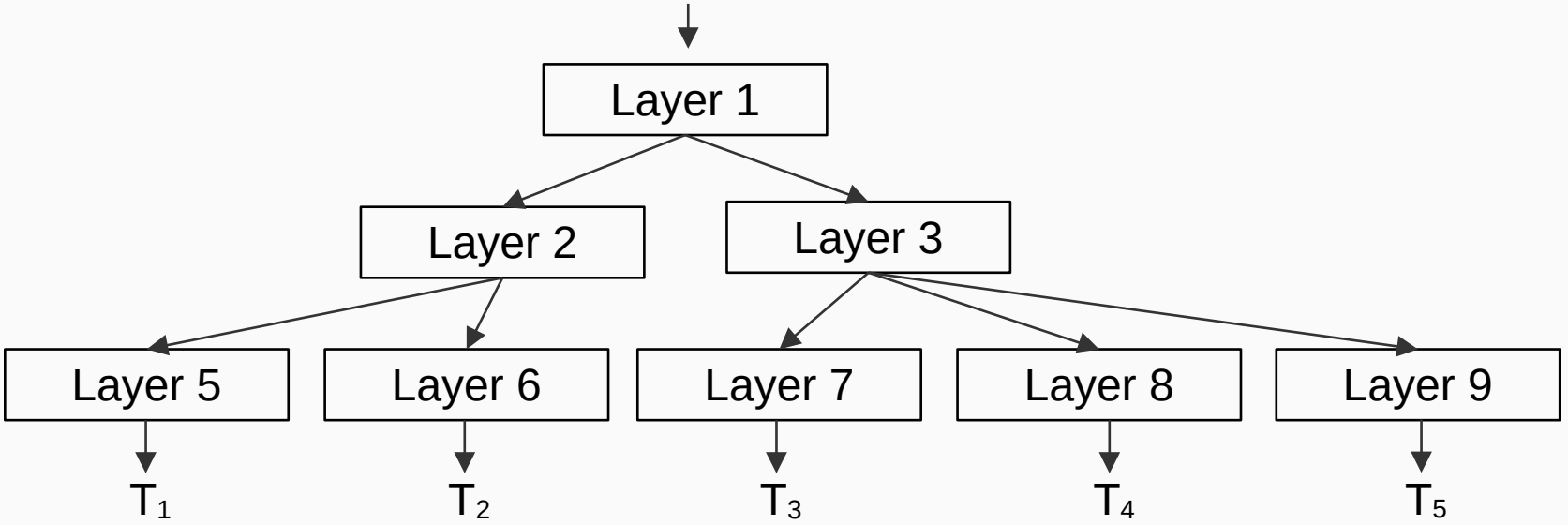
Summary of Contributions

- We study the challenges in porting deep learning workloads to trusted execution environments.
- We build Porpoise to port commodity applications to the Intel SGX enclaves.
- We present MazeNet to run deep learning inference workload on TEEs and improve the performance by outsourcing a portion of computations to untrusted hardware, while protecting the privacy of the model.

Thank You!

Backup Slides

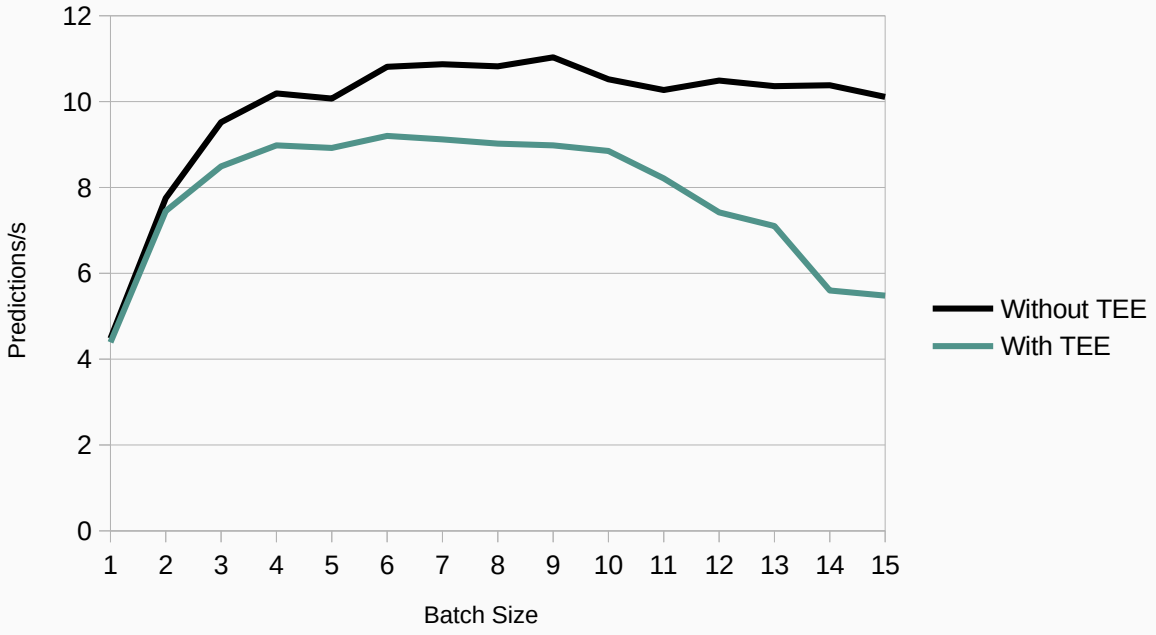
Formal Security Analysis



Only one of the subset of output will be the embedded output.

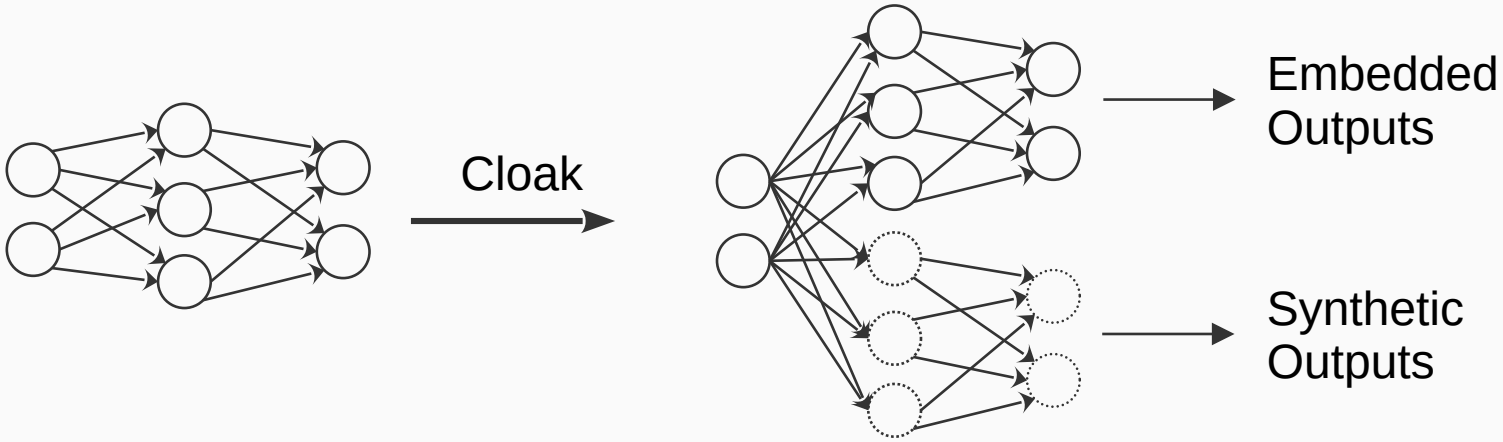
$$P[\text{embedded output}] = \frac{1}{2^{|T|} - 1}$$

Overheads: TEE

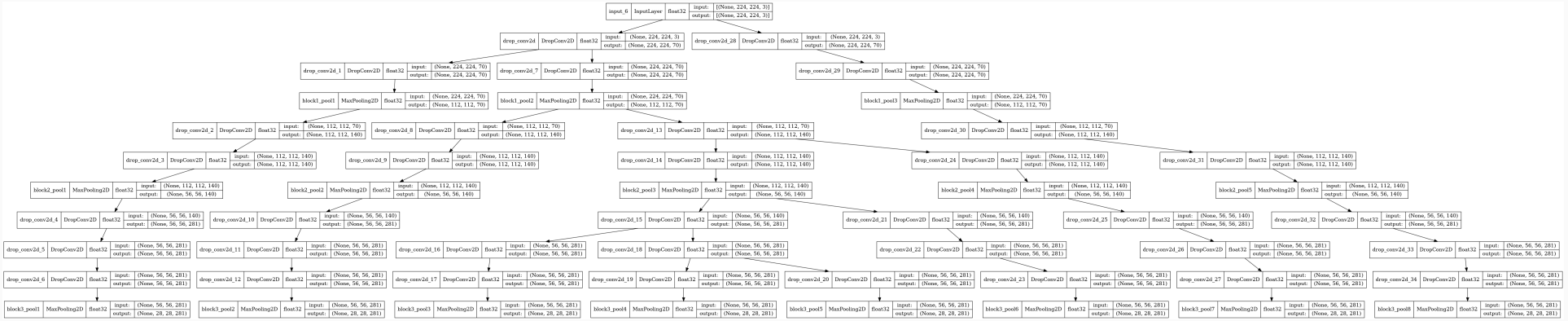


TEE has negligible overheads at low batch sizes, overheads increase at higher batch size.

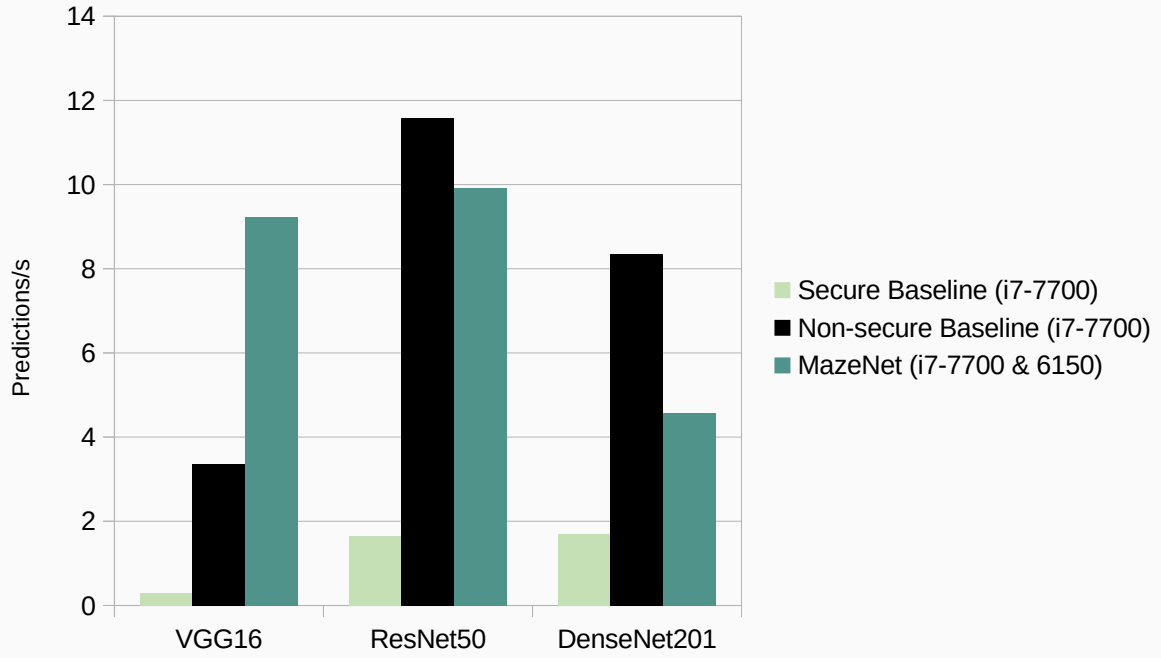
Synthetic layers



VGG16 Cloaked Submodel 1



Throughput Results



MazeNet improves the throughput by 2x to 30x as compared to secure baseline.