Achieving Zen: Combining Mathematical and Programmatic Deep Learning Model Representations for Attribution and Reuse

David Oygenblik, Dinko Dermendzhiev, Filippos Sofias, Mingxuan Yao, Haichuan Xu, Runze Zhang, Jeman Park, Amit Kumar Sikder, Brendan Saltaformaggio

Georgia Tech

davido@gatech.edu
davidoygenblik.github.io

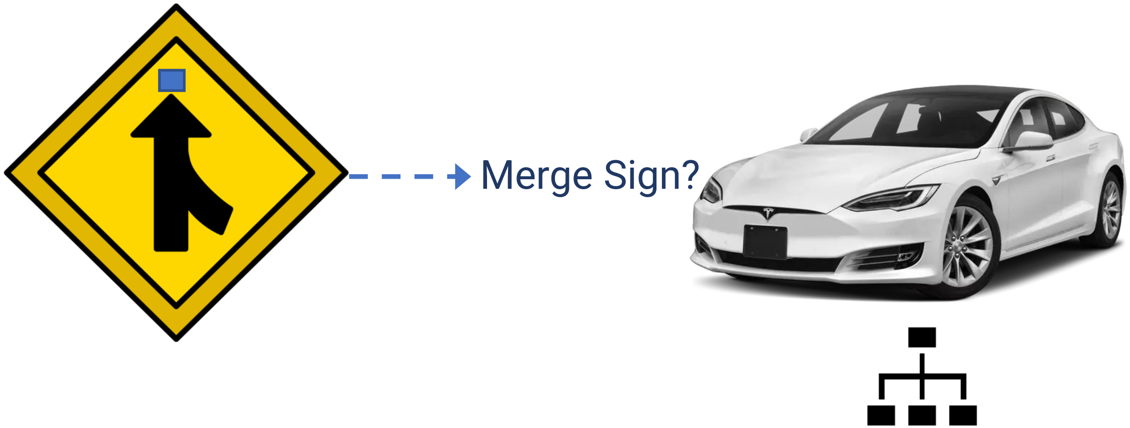
Georgia Tech | Cyber Forensics Innovation Lab

Artifact Evaluated
NDSS SYMPOSIUM
Available
Functional
Reproduced

Self-Driving Car Investigation

Proprietary Sign
Recognition Model

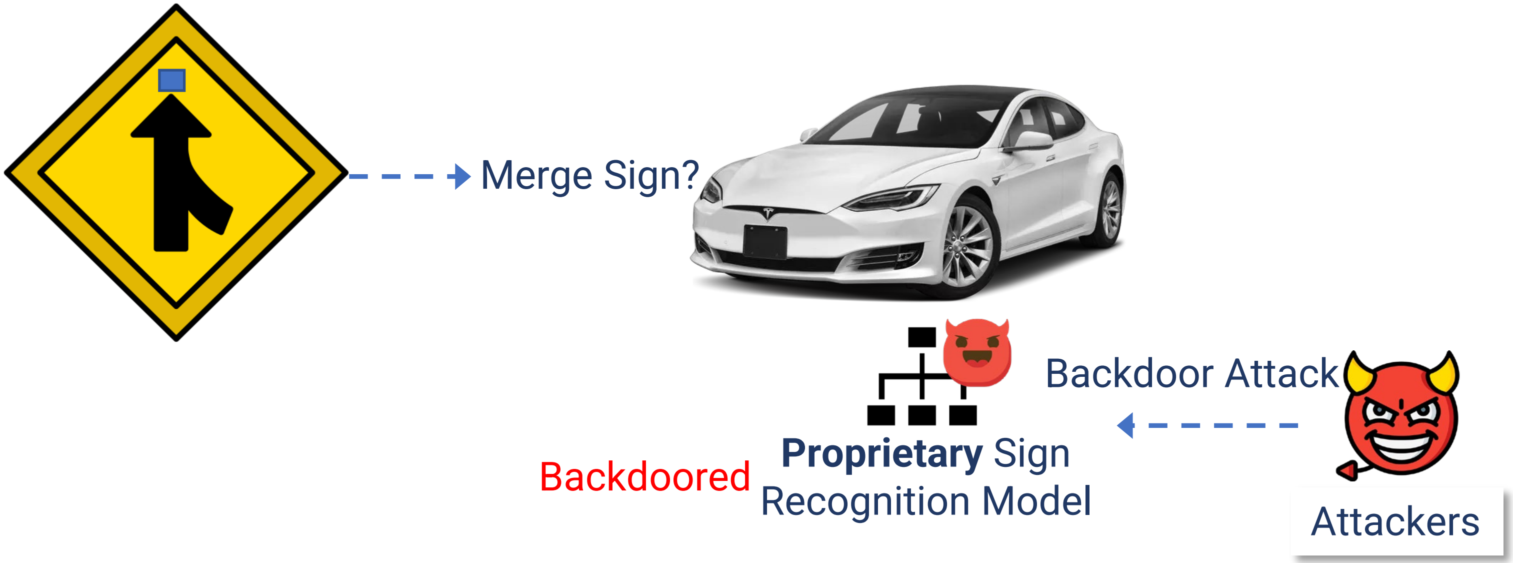
Self-Driving Car Investigation



Merge Sign?

Proprietary Sign Recognition Model

Self-Driving Car Investigation



Merge Sign?

Backdoor Attack

Backdoored **Proprietary** Sign Recognition Model

Attackers

Self-Driving Car Investigation

ge Sign?

Backdoor Attack

Backdoored

Proprietary Sign Recognition Model

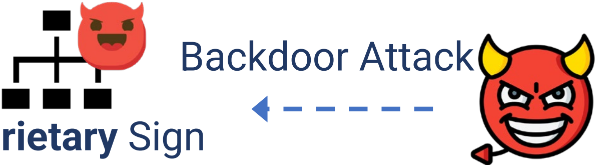
Attackers

Self-Driving Car Investigation

ge Sign?

Apply white-box techniques!

Detective Pika to the rescue



Backdoor Attack

Backdoored **Proprietary** Sign Recognition Model

Attackers

Self-Driving Car Investigation

ge Sign?

Detective Pika to the rescue

The developer decided to mix an open-source model with proprietary layer implementations!

Backdoored **Proprietary** Sign Recognition Model

Backdoor Attack

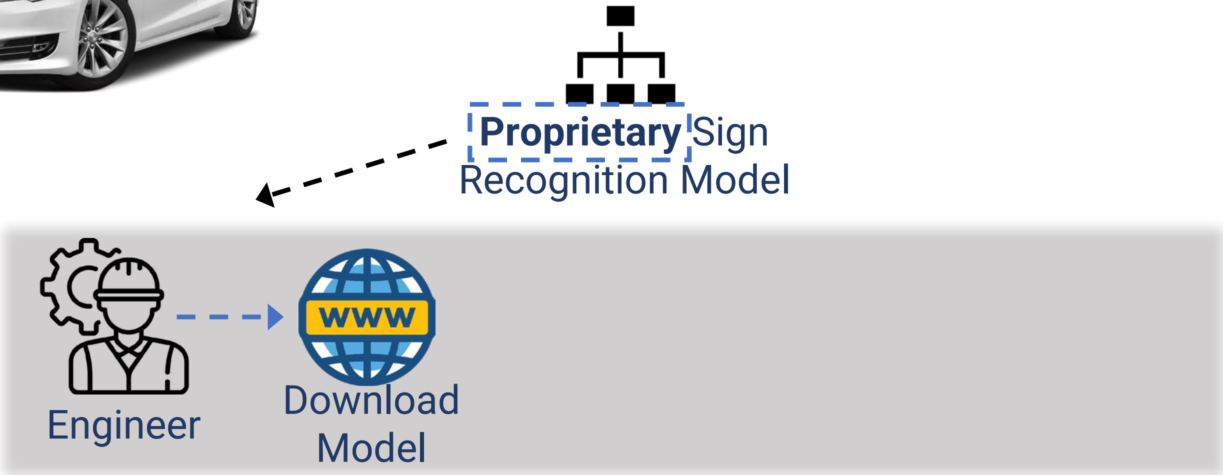
Attackers

Self-Driving Car Investigation

ge Sign?

Detective Pika to the rescue

The developer decided to mix an open-source model with proprietary layer implementations!



Proprietary Sign Recognition Model

Engineer

Download Model

Self-Driving Car Investigation

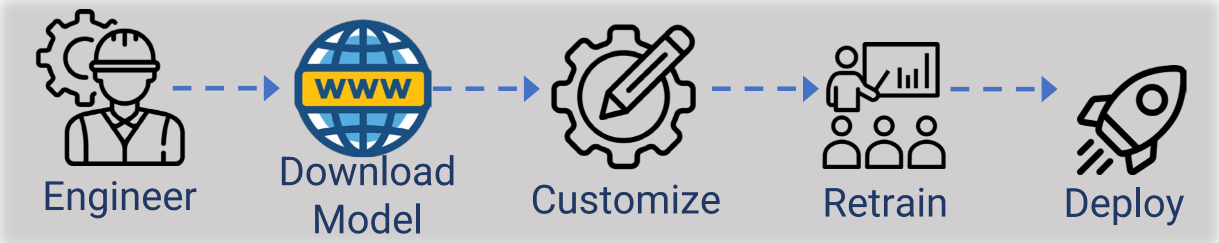
Apply white-box techniques!

Detective Pika to the rescue!

ge Sign?

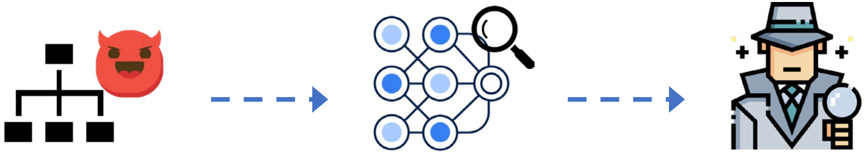
The developer decided to mix an open-source model with proprietary layer implementations!

Proprietary Sign Recognition Model



Investigative Goal

I need to run the model...

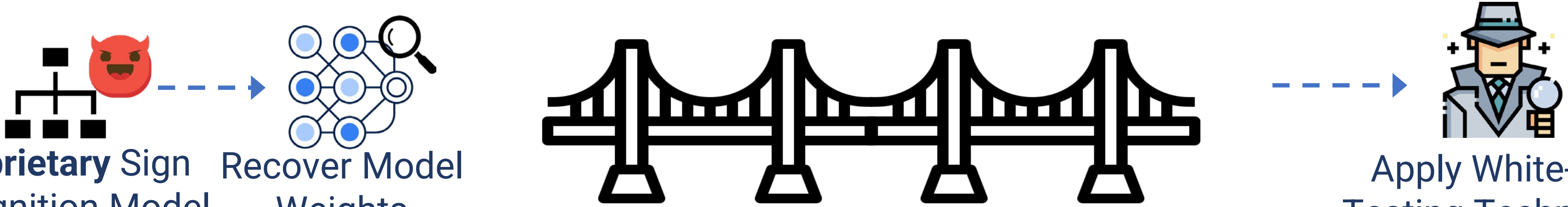


Proprietary Sign Recognition Model

Recover Model Weights

Apply White-box Testing Techniques

Investigative Goal



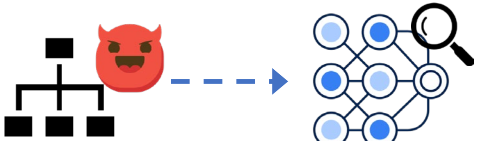
Proprietary Sign Recognition Model

Recover Model Weights

Apply White-box Testing Techniques

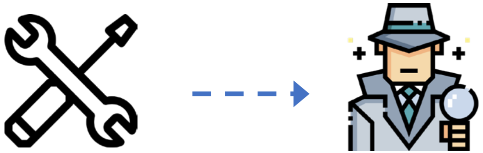
In reality, there is a huge gap that needs to be bridged to make this goal possible!

Investigative Goal



Proprietary Sign Recognition Model

Recover Model Weights

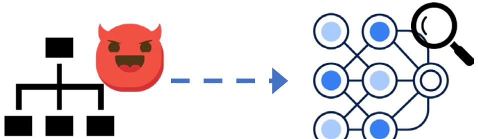


Requires Code Instrumentation

Apply White-box Testing Techniques

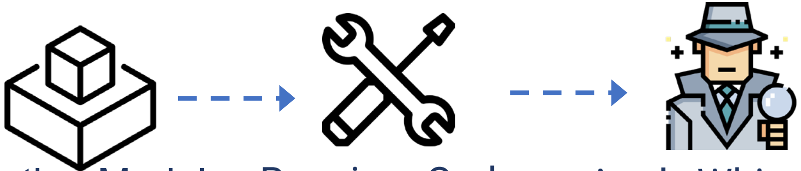
Need to make changes at the source-code level to invoke the white-box tools on the model.

Investigative Goal



Proprietary Sign Recognition Model

Recover Model Weights



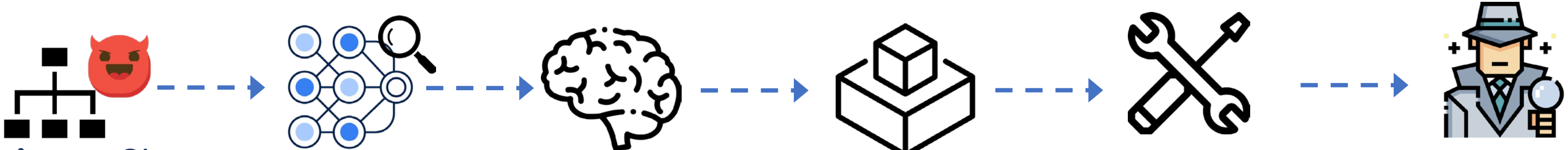
Testing Model Environment

Requires Code Instrumentation

Apply White-box Testing Techniques

Source-code level modifications require an environment to make those modifications where the model's layers, inference code, etc. are defined.

Investigative Goal



Proprietary Sign Recognition Model

Recover Model Weights

Recover Known Layer Types

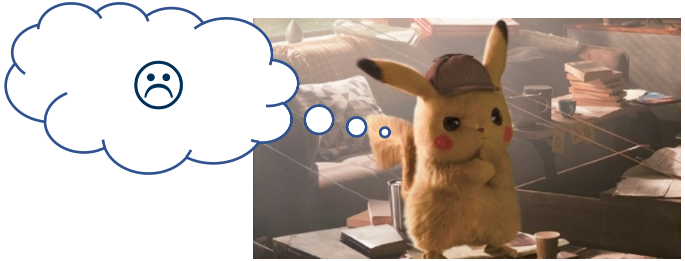
Testing Model Environment

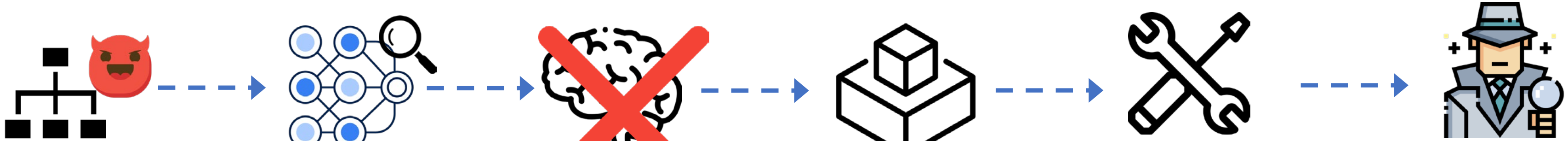
Requires Code Instrumentation

Apply White-box Testing Techniques

The developer used SOTA layer implementations! Their implementation and functionality needs to be known to use in the testing model environment!

Investigative Goal

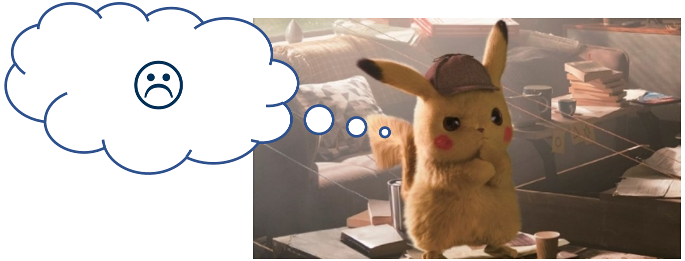


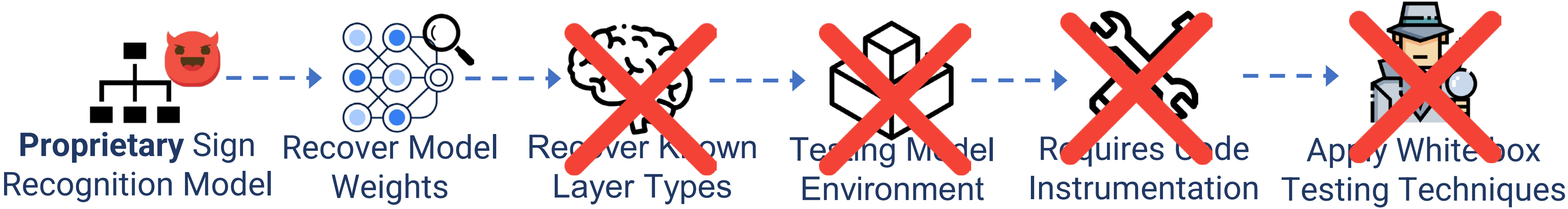


Proprietary Sign Recognition Model → Recover Model Weights → ~~Recover Known Layer Types~~ → Testing Model Environment → Requires Code Instrumentation → Apply White-box Testing Techniques

The developer used SOTA layer implementations! Their implementation and functionality needs to be known to use in the testing model environment!

Investigative Goal





Proprietary Sign Recognition Model

Recover Model Weights

~~Recover Known Layer Types~~

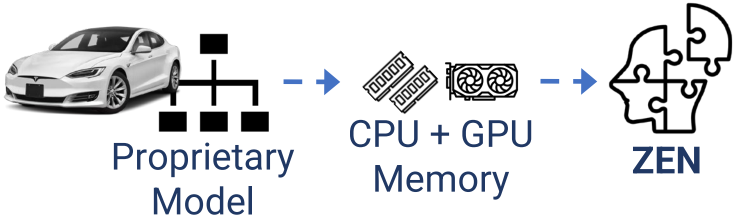
~~Testing Model Environment~~

~~Requires Code Instrumentation~~

~~Apply White-box Testing Techniques~~

Without those SOTA layer implementations, the requirements for the remaining steps to apply white-box techniques are unsatisfiable…

ZEN Bridges the Gap to Investigation!

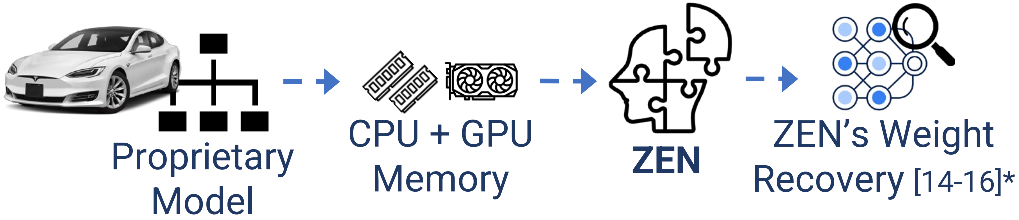


Proprietary Model

CPU + GPU Memory

ZEN

ZEN Bridges the Gap to Investigation!



Proprietary Model

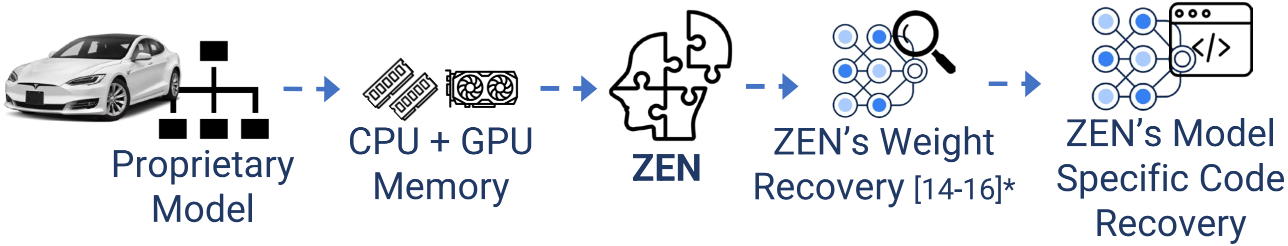
CPU + GPU Memory

ZEN

ZEN's Weight Recovery [14-16]*

*Recovery of weights is prior work and can be found in our paper's matching citations.

ZEN Bridges the Gap to Investigation!



Proprietary Model

CPU + GPU Memory

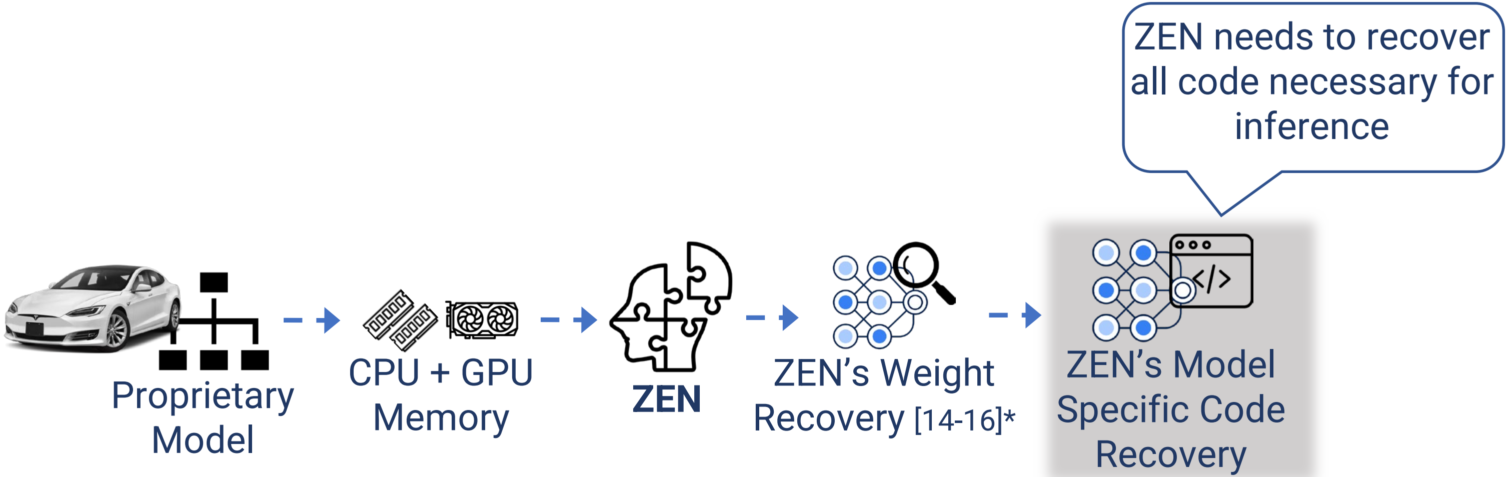
ZEN

ZEN's Weight Recovery [14-16]*

ZEN's Model Specific Code Recovery

*Recovery of weights is prior work and can be found in our paper's matching citations.

ZEN Bridges the Gap to Investigation!



ZEN needs to recover all code necessary for inference

Proprietary Model

CPU + GPU Memory

ZEN

ZEN's Weight Recovery [14-16]*

ZEN's Model Specific Code Recovery

*Recovery of weights is prior work and can be found in our paper's matching citations.

ZEN Bridges the Gap to Investigation!

ZEN needs to recover all code necessary for inference

Model Specific Code Recovery

```
"conv_3x3_bn": ["models.ghostnet", "conv_3x3_bn", "conv_3x3_bn"],
    func_attr: [3, 3, ["inp", "oup", "stride"]]
"conv_1x1_bn": ["models.ghostnet", "conv_1x1_bn", "conv_1x1_bn"],
    func_attr: [2, 2, ["inp", "oup"]]
"__init__": ["utils.logger", "Logger.__init__", "Logger"],
    func_attr: [4, 8, ["self", "fpath", "title", "resume",
            "name", "_", "numbers", "i"]]
.....................................................................
.....................................................................
.....................................................................
"forward": ["models.ghostnet", "InvertedResidual.forward", "InvertedResidual"],
    func_attr: [2, 2, ["self", "x"]],
"__init___1": ["utils.logger", "LoggerMonitor.__init__", "LoggerMonitor"],
    func_attr: [2, 5, ["self", "paths", "title", "path", "logger"]]
"forward_1": ["models.ghostnet", "MobileNetV2.forward", "MobileNetV2"],
    func_attr: [2, 2, ["self", "x"]]
```

Function Enumeration and Signature Extraction (e.g., args, local vars, etc.)

ZEN Bridges the Gap to Investigation!

ZEN needs to recover all code necessary for inference

Model Specific Code Recovery

```
"conv_3x3_bn": ["models.ghostnet", "conv_3x3_bn", "conv_3x3_bn"],
    func_attr: [3, 3, ["inp", "oup", "stride"]]
"conv_1x1_bn": ["models.ghostnet", "conv_1x1_bn", "conv_1x1_bn"],
    func_attr: [2, 2, ["inp", "oup"]]
"__init__": ["utils.logger", "Logger.__init__", "Logger"],
    func_attr: [4, 8, ["self", "fpath", "title", "resume",
            "name", "_", "numbers", "i"]]
.......................................................................
.......................................................................
.......................................................................
"forward": ["models.ghostnet", "InvertedResidual.forward", "InvertedResidual"],
    func_attr: [2, 2, ["self", "x"]],
"__init__1": ["utils.logger", "LoggerMonitor.__init__", "LoggerMonitor"],
    func_attr: [2, 5, ["self", "paths", "title", "path", "logger"]]
"forward_1": ["models.ghostnet", "MobileNetV2.forward", "MobileNetV2"],
    func_attr: [2, 2, ["self", "x"]]
```

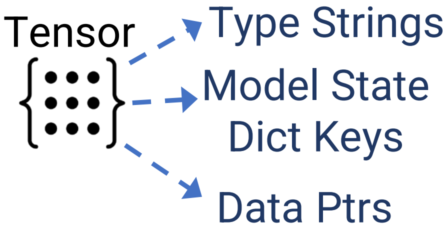
Function Enumeration and Signature Extraction (e.g., args, local vars, etc.)

ZEN Bridges the Gap to Investigation!

ZEN needs to recover all code necessary for inference

Model Specific Code Recovery

```
"conv_3x3_bn": ["models.ghostnet", "conv_3x3_bn", "conv_3x3_bn"],
    func_attr: [3, 3, ["inp", "oup", "stride"]]
"conv_1x1_bn": ["models.ghostnet", "conv_1x1_bn", "conv_1x1_bn"],
    func_attr: [2, 2, ["inp", "oup"]]
"__init__": ["utils.logger", "Logger.__init__", "Logger"],
    func_attr: [4, 8, ["self", "fpath", "title", "resume",
            "name", "_", "numbers", "i"]]
.............................................................
.............................................................
.............................................................
"forward": ["models.ghostnet", "InvertedResidual.forward", "InvertedResidual"],
    func_attr: [2, 2, ["self", "x"]],
"__init___1": ["utils.logger", "LoggerMonitor.__init__", "LoggerMonitor"],
    func_attr: [2, 5, ["self", "paths", "title", "path", "logger"]]
"forward_1": ["models.ghostnet", "MobileNetV2.forward", "MobileNetV2"],
    func_attr: [2, 2, ["self", "x"]]
```



Tensor

Type Strings

Model State Dict Keys

Data Ptrs

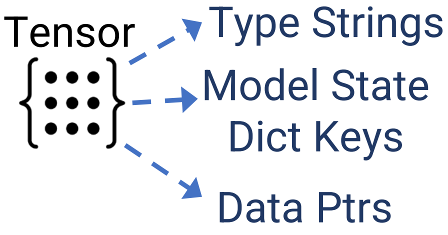
Architecture and Layer Informed Code Filtering

ZEN Bridges the Gap to Investigation!

ZEN needs to recover all code necessary for inference

Model Specific Code Recovery

```
"conv_3x3_bn": ["models.ghostnet", "conv_3x3_bn", "conv_3x3_bn"],
    func_attr: [3, 3, ["inp", "oup", "stride"]]
"conv_1x1_bn": ["models.ghostnet", "conv_1x1_bn", "conv_1x1_bn"],
    func_attr: [2, 2, ["inp", "oup"]]
~~"__init__": ["utils.logger", "Logger.__init__", "Logger"],~~
~~    func_attr: [4, 8, ["self", "fpath", "title", "resume",~~
~~            "name", "_", "numbers", "i"]]~~
.....................................................................
.....................................................................
.....................................................................
"forward": ["models.ghostnet", "InvertedResidual.forward", "InvertedResidual"],
    func_attr: [2, 2, ["self", "x"]],
~~"__init___1": ["utils.logger", "LoggerMonitor.__init__", "LoggerMonitor"],~~
~~    func_attr: [2, 5, ["self", "paths", "title", "path", "logger"]]~~
"forward_1": ["models.ghostnet", "MobileNetV2.forward", "MobileNetV2"],
    func_attr: [2, 2, ["self", "x"]]
```



Type Strings

Model State Dict Keys

Data Ptrs

Architecture and Layer Informed Code Filtering

ZEN Bridges the Gap to Investigation!

ZEN needs to recover all code necessary for inference

Model Specific Code Recovery

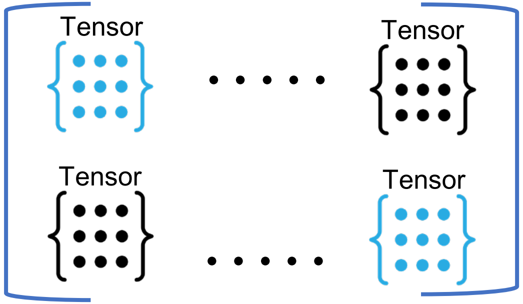
```
"conv_3x3_bn": ["models.ghostnet", "conv_3x3_bn", "conv_3x3_bn"],
    func_attr: [3, 3, ["inp", "oup", "stride"]]
"conv_1x1_bn": ["models.ghostnet", "conv_1x1_bn", "conv_1x1_bn"],
    func_attr: [2, 2, ["inp", "oup"]]
~~"__init__": ["utils.logger", "Logger.__init__", "Logger"],~~
~~    func_attr: [4, 8, ["self", "fpath", "title", "resume",~~
~~            "name", "_", "numbers", "i"]]~~
........................................................................
........................................................................
........................................................................
"forward": ["models.ghostnet", "InvertedResidual.forward", "InvertedResidual"],
    func_attr: [2, 2, ["self", "x"]],
~~"__init___1": ["utils.logger", "LoggerMonitor.__init__", "LoggerMonitor"],~~
~~    func_attr: [2, 5, ["self", "paths", "title", "path", "logger"]]~~
"forward_1": ["models.ghostnet", "MobileNetV2.forward", "MobileNetV2"],
    func_attr: [2, 2, ["self", "x"]]
```

→ *Analyze Function's Bytecode*

Ensure that we recover all code that 'forward' depends on (recursive)

Call Graph Traversal & Dependency Analysis

ZEN Bridges the Gap to Investigation!

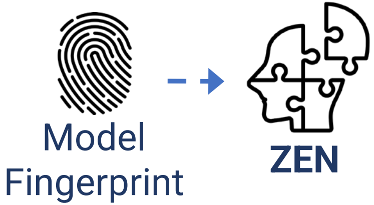


```
"conv_3x3_bn": ["models.ghostnet", "conv_3x3_bn", "conv_3x3_bn"],
    func_attr: [3, 3, ["inp", "oup", "stride"]]
"conv_1x1_bn": ["models.ghostnet", "conv_1x1_bn", "conv_1x1_bn"],
    func_attr: [2, 2, ["inp", "oup"]]
~~"__init__": ["utils.logger", "Logger.__init__", "Logger"],~~
~~    func_attr: [4, 8, ["self", "fpath", "title", "resume",~~
~~                "name", "_", "numbers", "i"]]~~
.............................................................
.............................................................
.............................................................
"forward": ["models.ghostnet", "InvertedResidual.forward", "InvertedResidual"],
    func_attr: [2, 2, ["self", "x"]],
~~"__init___1": ["utils.logger", "LoggerMonitor.__init__", "LoggerMonitor"],~~
~~    func_attr: [2, 5, ["self", "paths", "title", "path", "logger"]]~~
"forward_1": ["models.ghostnet", "MobileNetV2.forward", "MobileNetV2"],
    func_attr: [2, 2, ["self", "x"]]
```

Create model fingerprint by combining information from weight recovery and model-specific code recovery!

ZEN’s Model Attribution!

ZEN now needs something to compare the model fingerprints to…

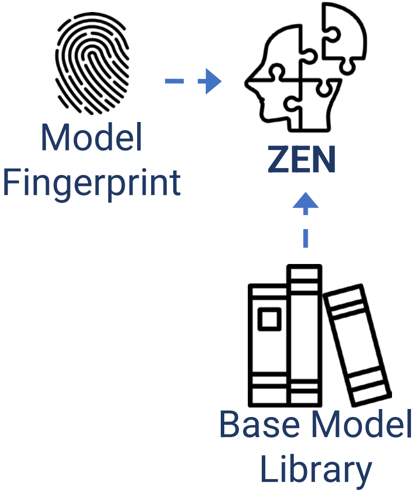


Model Fingerprint

ZEN

ZEN’s Model Attribution!

We created a base model library which contains fingerprints for SOTA open-source models!



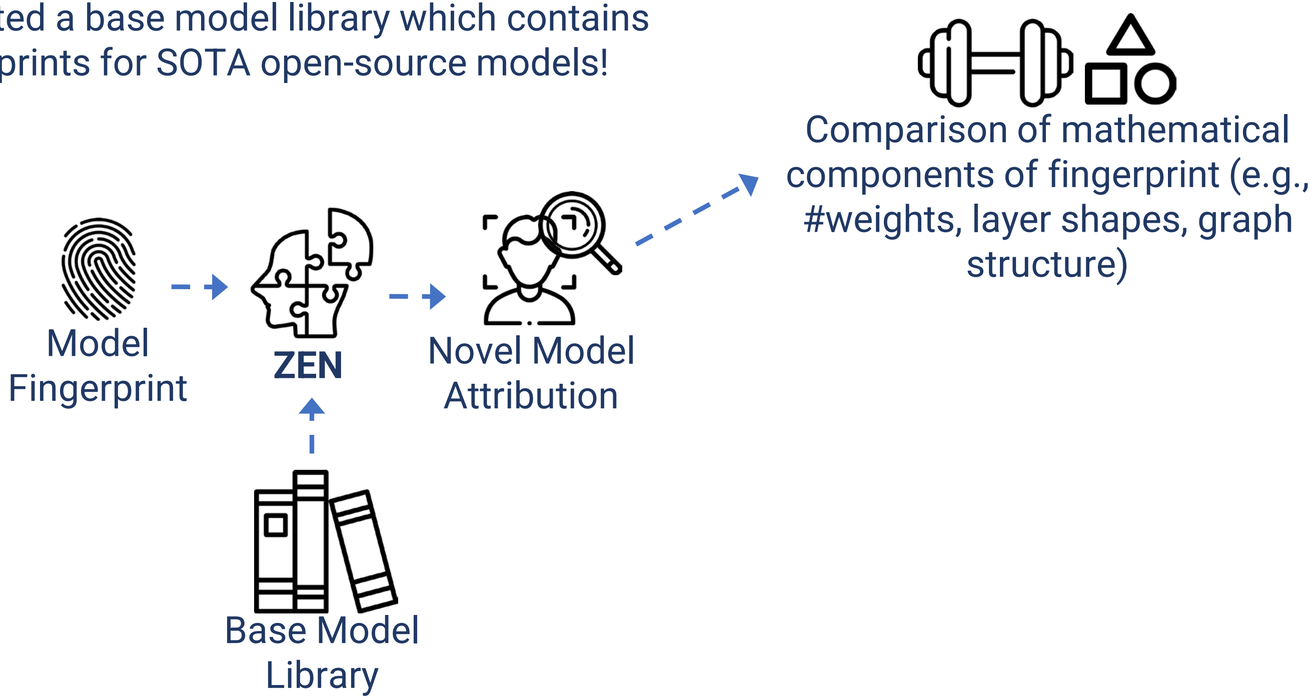
Model Fingerprint

ZEN

Base Model Library

ZEN’s Model Attribution!

We created a base model library which contains fingerprints for SOTA open-source models!



Model Fingerprint

ZEN

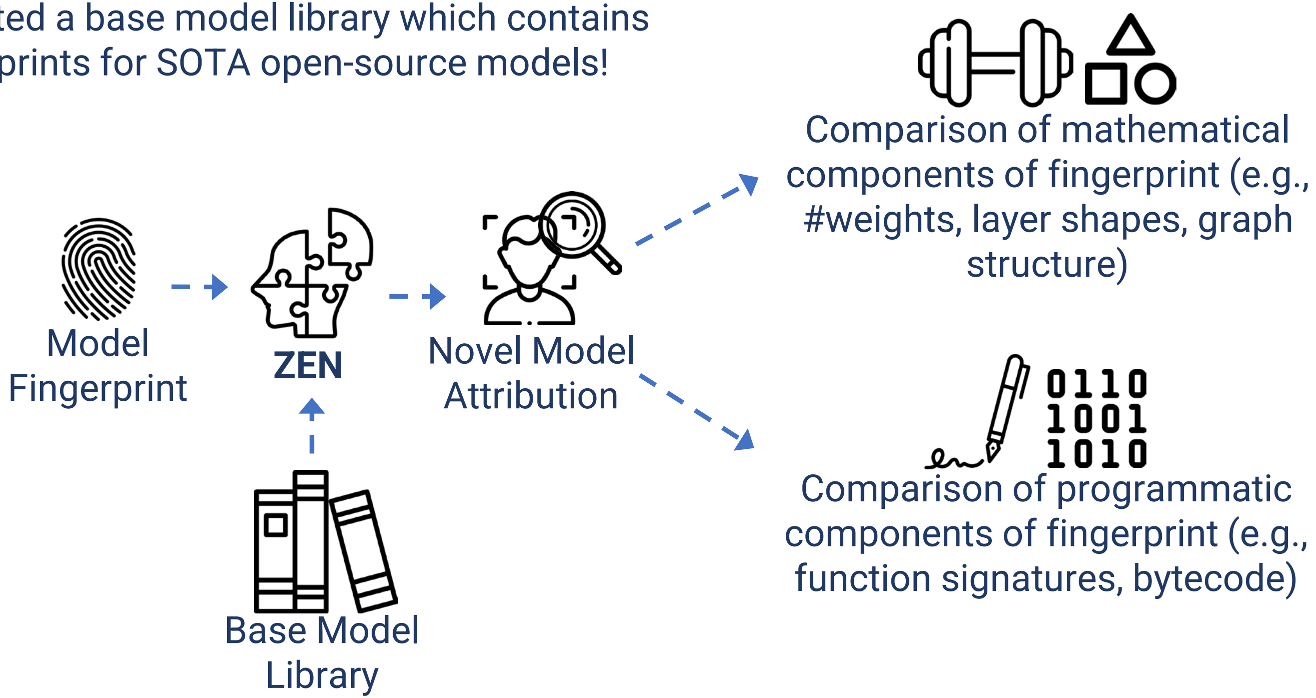
Novel Model Attribution

Base Model Library

Comparison of mathematical components of fingerprint (e.g., #weights, layer shapes, graph structure)

ZEN’s Model Attribution!

We created a base model library which contains fingerprints for SOTA open-source models!



Model Fingerprint

ZEN

Base Model Library

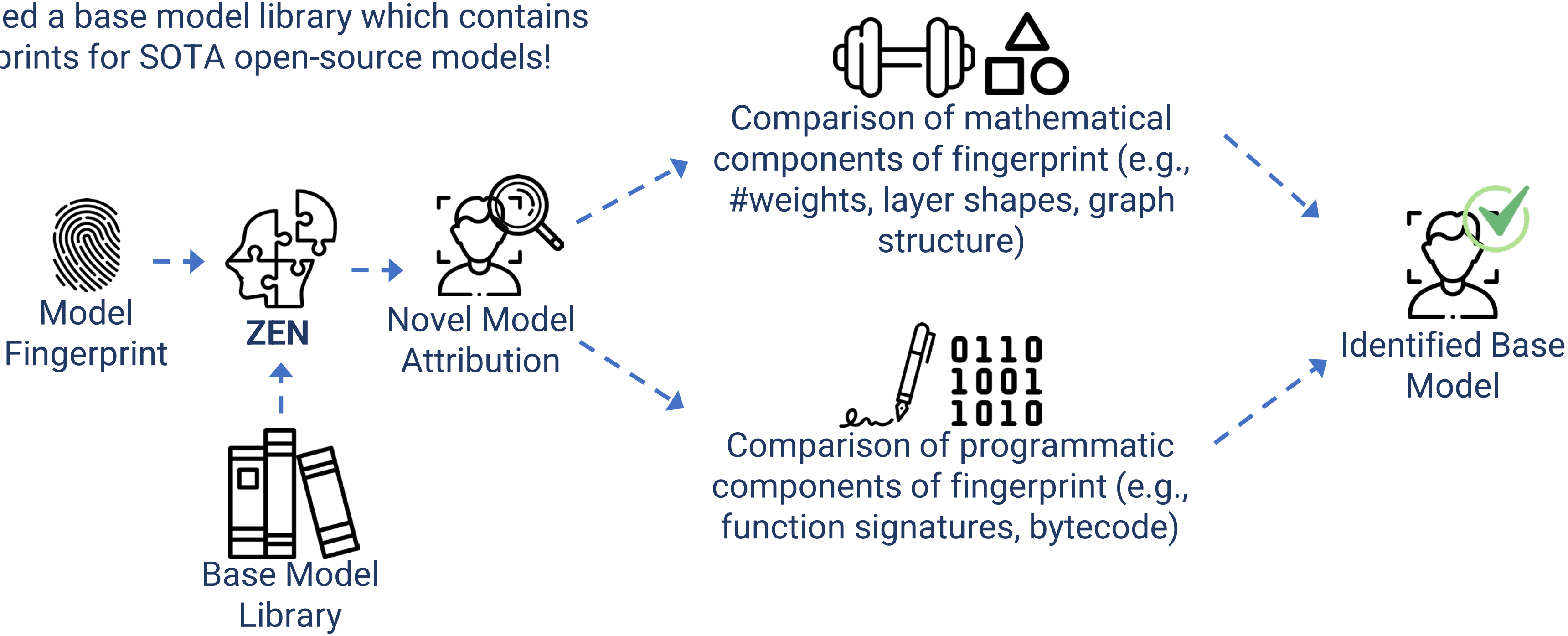
Novel Model Attribution

Comparison of mathematical components of fingerprint (e.g., #weights, layer shapes, graph structure)

Comparison of programmatic components of fingerprint (e.g., function signatures, bytecode)

ZEN’s Model Attribution!

We created a base model library which contains fingerprints for SOTA open-source models!



Model Fingerprint

ZEN

Base Model Library

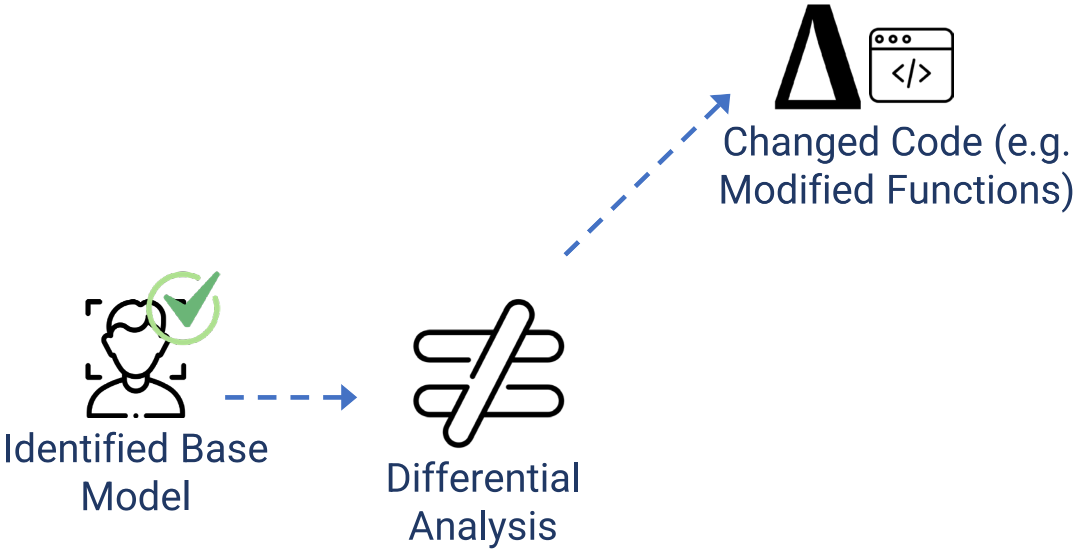
Novel Model Attribution

Comparison of mathematical components of fingerprint (e.g., #weights, layer shapes, graph structure)

Comparison of programmatic components of fingerprint (e.g., function signatures, bytecode)

Identified Base Model

ZEN's Differential Analysis and Patching

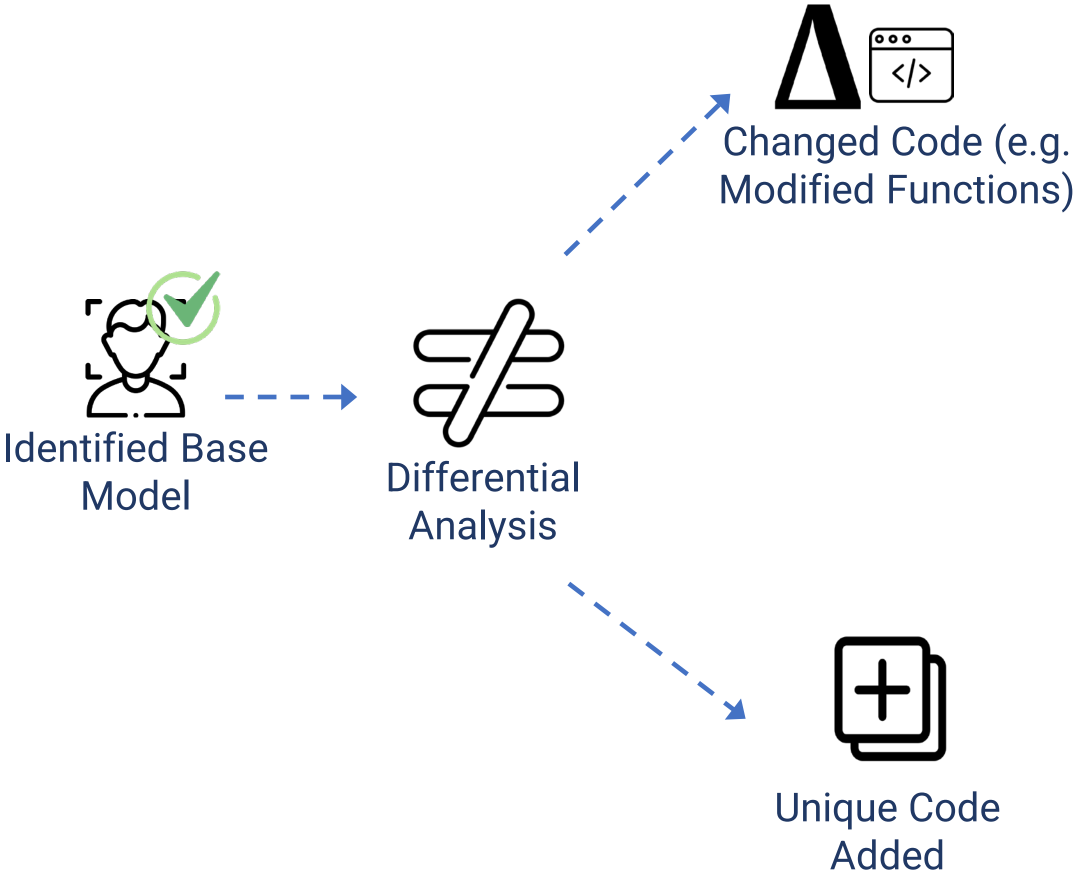


Identified Base Model

Differential Analysis

Changed Code (e.g. Modified Functions)

ZEN’s Differential Analysis and Patching



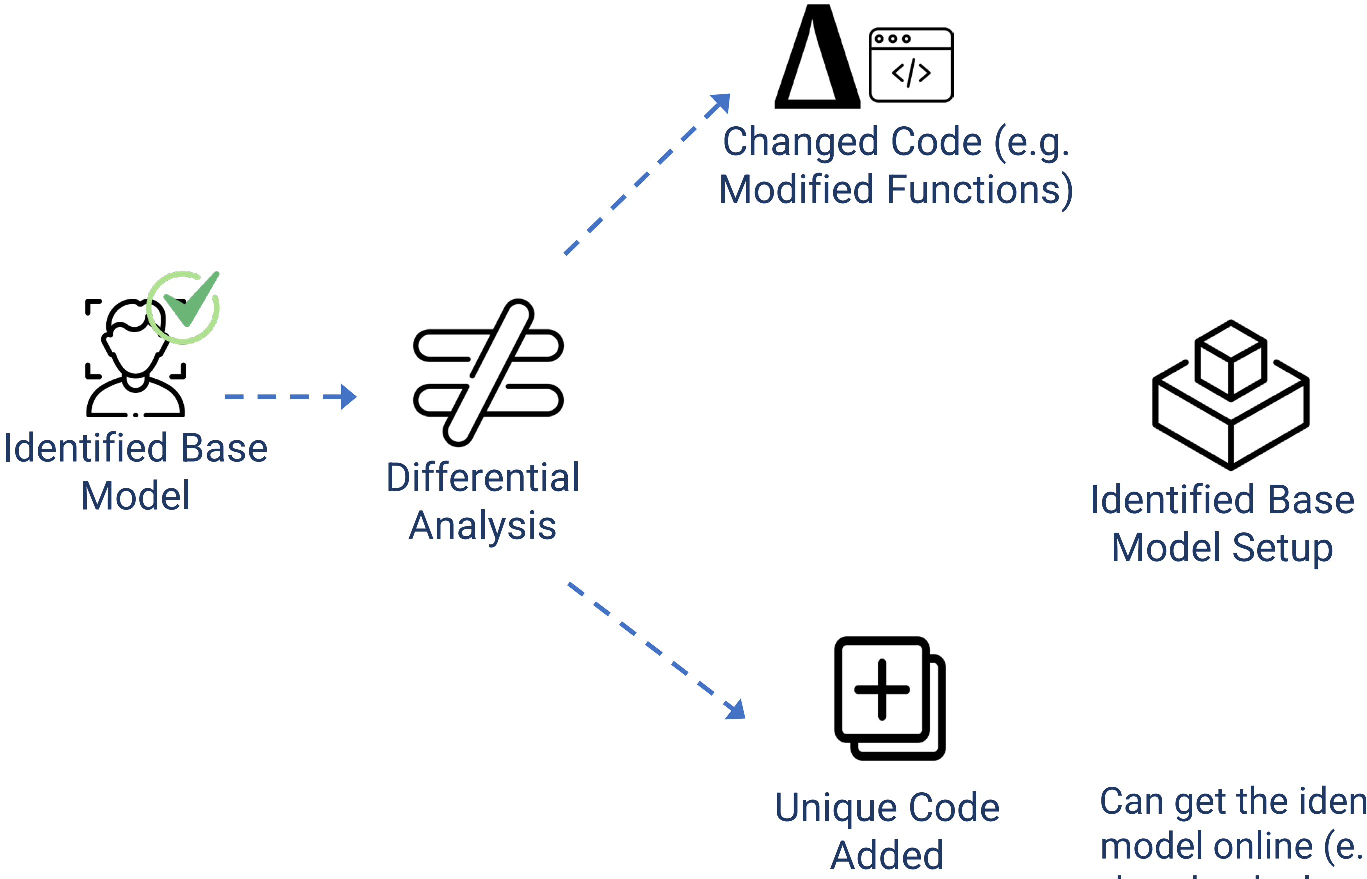
Changed Code (e.g. Modified Functions)

Identified Base Model

Differential Analysis

Unique Code Added

ZEN's Differential Analysis and Patching



Identified Base Model

Differential Analysis

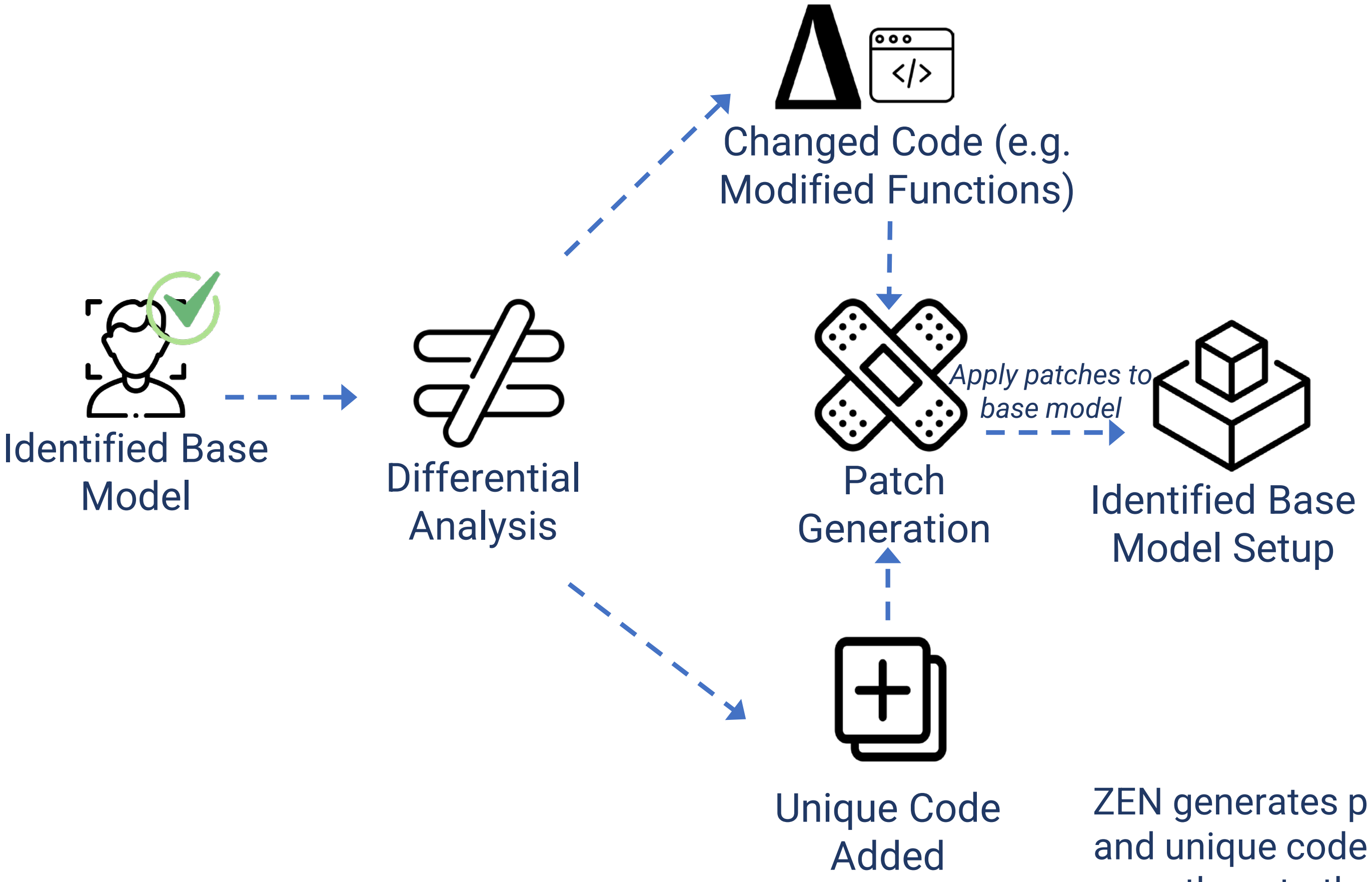
Changed Code (e.g. Modified Functions)

Identified Base Model Setup

Unique Code Added

Can get the identified SOTA open-source model online (e.g., Github, Huggingface), but the deployed model was different…

ZEN's Differential Analysis and Patching



Identified Base Model

Differential Analysis

Changed Code (e.g. Modified Functions)

Unique Code Added

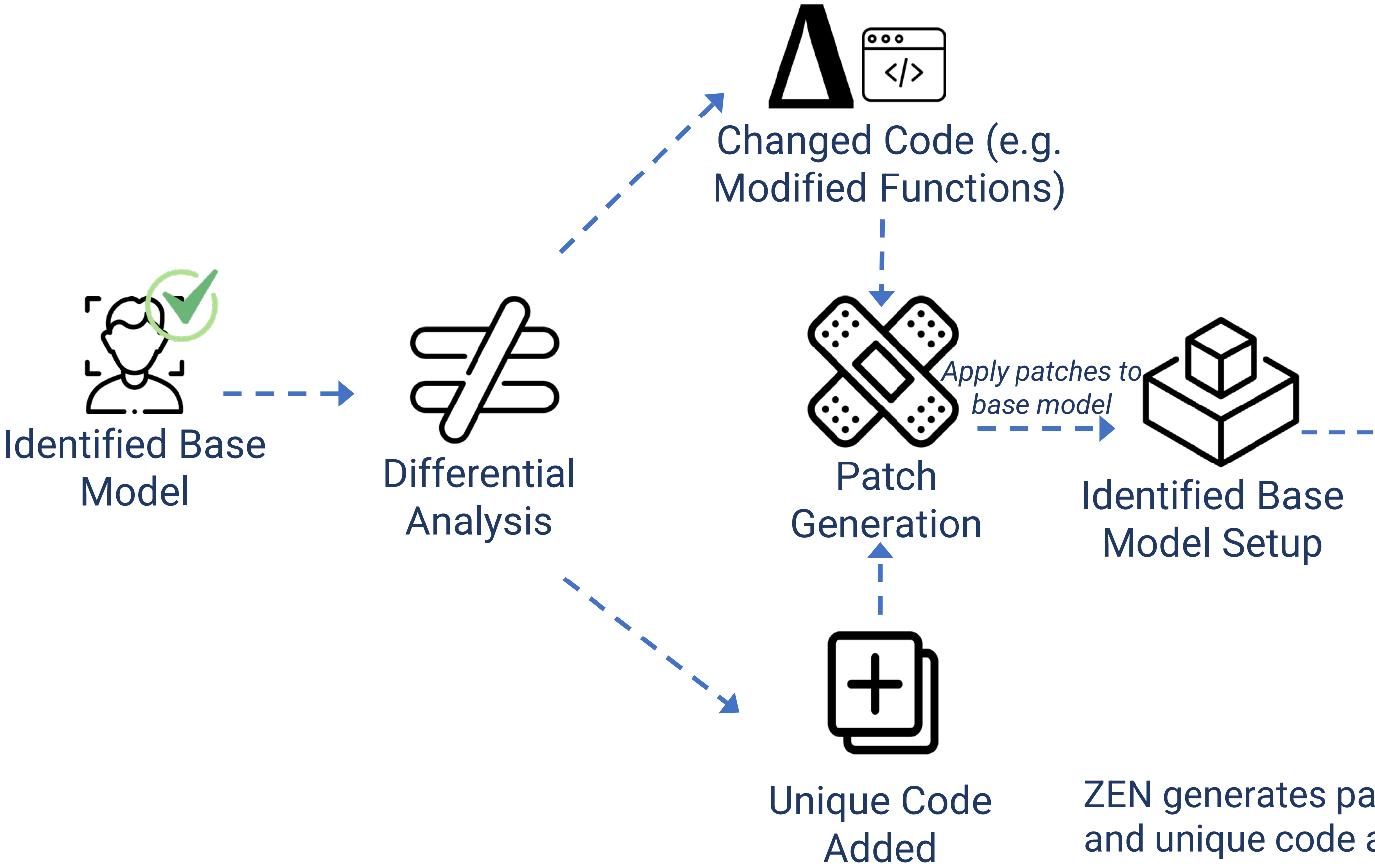
Patch Generation

Apply patches to base model

Identified Base Model Setup

ZEN generates patches for changed code and unique code added and directly apply them to the base model setup!

ZEN's Differential Analysis and Patching



Changed Code (e.g. Modified Functions)

Identified Base Model

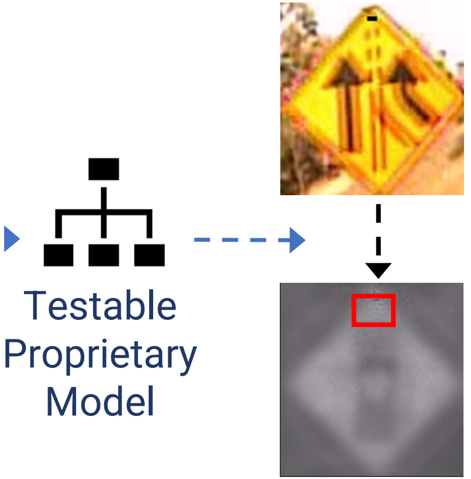
Differential Analysis

Patch Generation

Apply patches to base model

Identified Base Model Setup

Unique Code Added



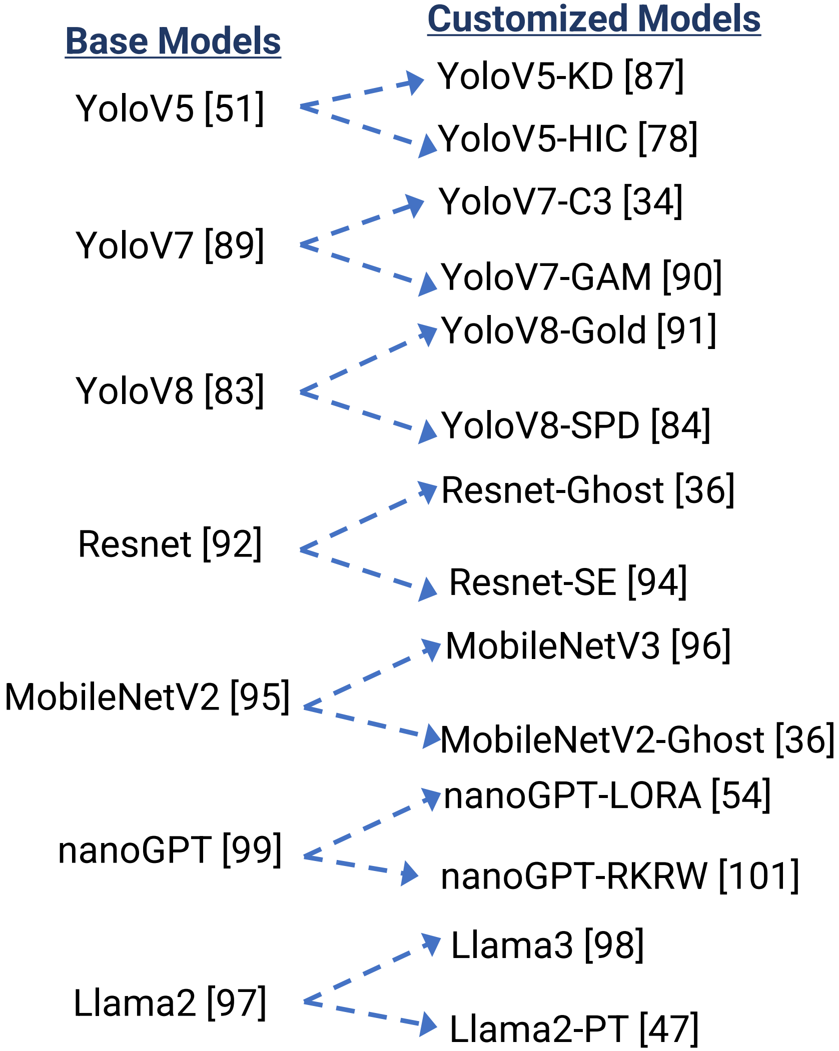
Testable Proprietary Model

White-box testing enabled! ☺

ZEN generates patches for changed code and unique code added and directly apply them to the base model setup!

Experiment Setup

Models: 7 model families for object detection, image classification, and text generation.

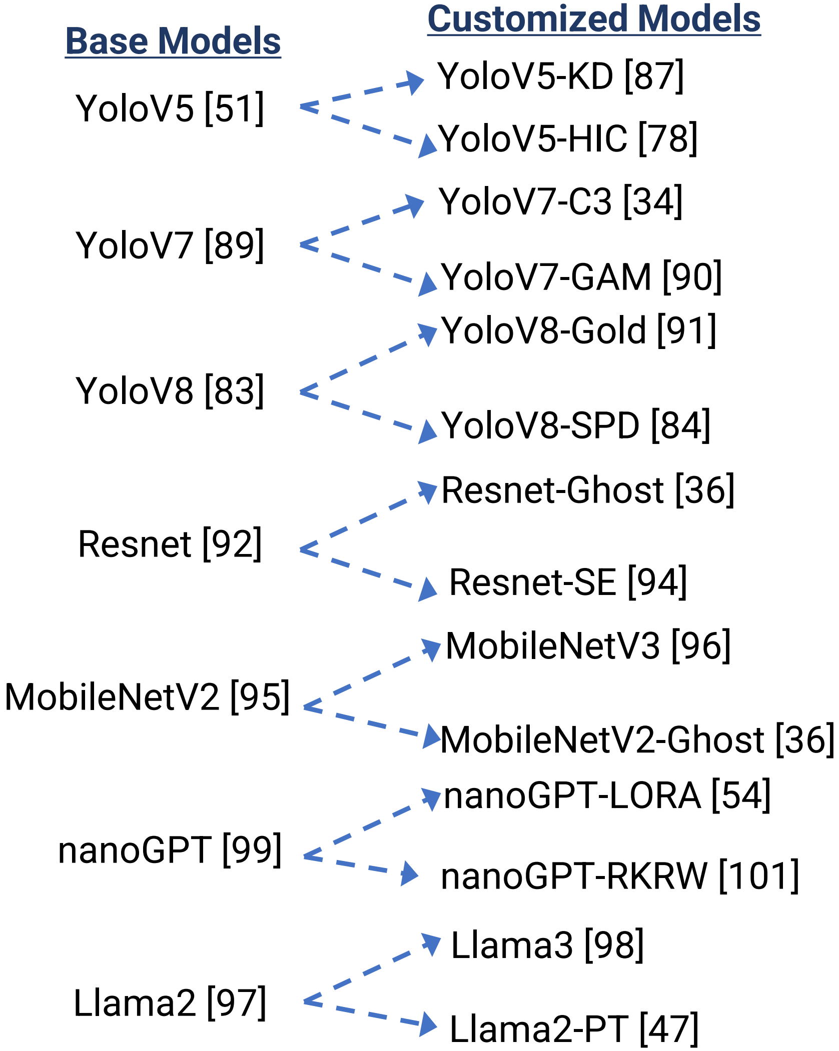


Base Models	Customized Models
YoloV5 [51]	YoloV5-KD [87] YoloV5-HIC [78]
YoloV7 [89]	YoloV7-C3 [34] YoloV7-GAM [90]
YoloV8 [83]	YoloV8-Gold [91] YoloV8-SPD [84]
Resnet [92]	Resnet-Ghost [36] Resnet-SE [94]
MobileNetV2 [95]	MobileNetV3 [96] MobileNetV2-Ghost [36]
nanoGPT [99]	nanoGPT-LORA [54] nanoGPT-RKRW [101]
Llama2 [97]	Llama3 [98] Llama2-PT [47]

Experiment Setup

Models: 7 model families for object detection, image classification, and text generation.

Datasets: Models trained on **COCO, VisDrone, Cifar10, and OpenWebtext** datasets.



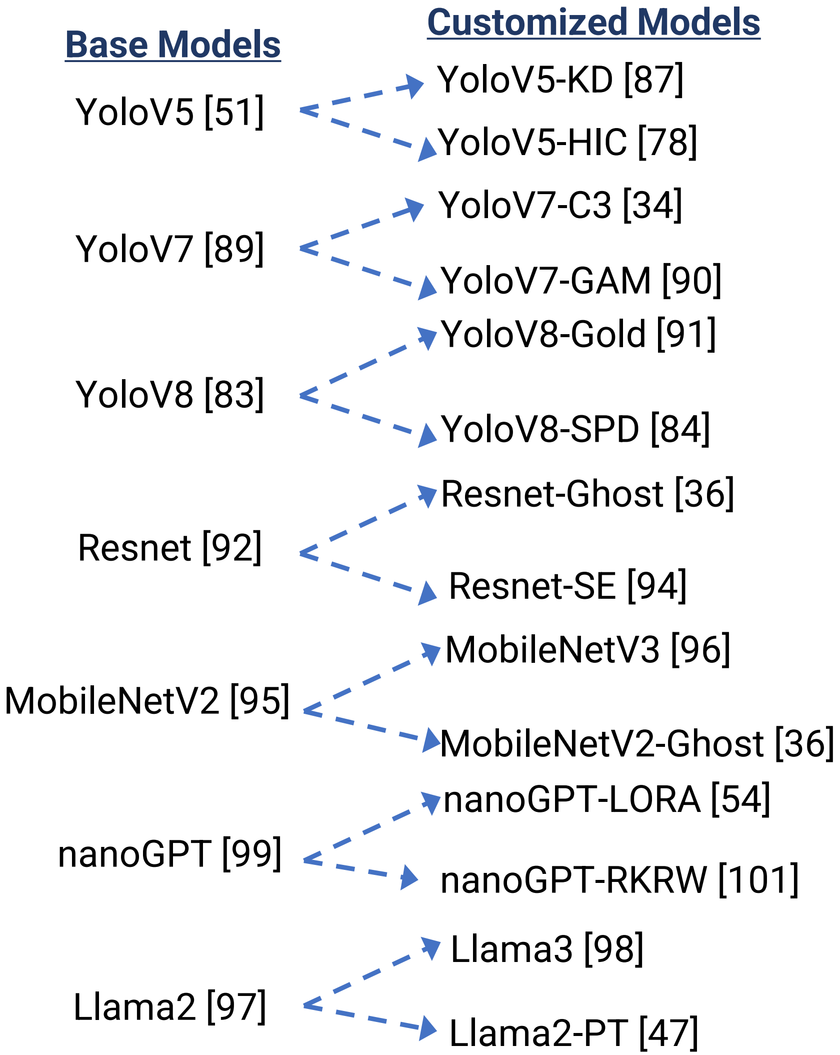
Base Models	Customized Models
YoloV5 [51]	YoloV5-KD [87] YoloV5-HIC [78]
YoloV7 [89]	YoloV7-C3 [34] YoloV7-GAM [90]
YoloV8 [83]	YoloV8-Gold [91] YoloV8-SPD [84]
Resnet [92]	Resnet-Ghost [36] Resnet-SE [94]
MobileNetV2 [95]	MobileNetV3 [96] MobileNetV2-Ghost [36]
nanoGPT [99]	nanoGPT-LORA [54] nanoGPT-RKRW [101]
Llama2 [97]	Llama3 [98] Llama2-PT [47]

Experiment Setup

Models: 7 model families for object detection, image classification, and text generation.

Datasets: Models trained on **COCO, VisDrone, Cifar10, and OpenWebtext** datasets.

ZEN Input: CPU/GPU memories for each of the custom model deployments.



Base Models

Customized Models

YoloV5 [51]

YoloV5-KD [87]

YoloV5-HIC [78]

YoloV7 [89]

YoloV7-C3 [34]

YoloV7-GAM [90]

YoloV8 [83]

YoloV8-Gold [91]

YoloV8-SPD [84]

Resnet [92]

Resnet-Ghost [36]

Resnet-SE [94]

MobileNetV2 [95]

MobileNetV3 [96]

MobileNetV2-Ghost [36]

nanoGPT [99]

nanoGPT-LORA [54]

nanoGPT-RKRW [101]

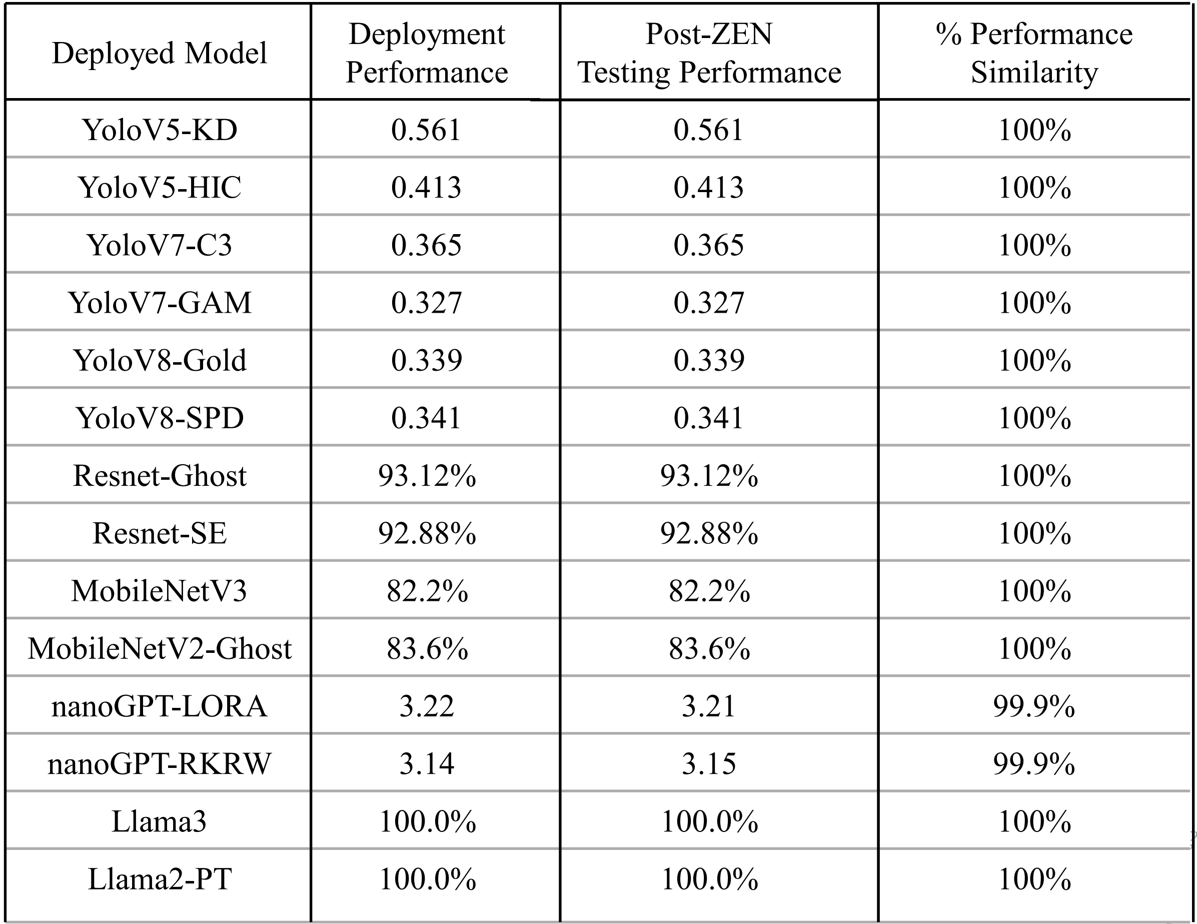
Llama2 [97]

Llama3 [98]

Llama2-PT [47]

ZEN Post Patching Performance

To check the success of model rehosting, we once again measure the pre/post deployment performance

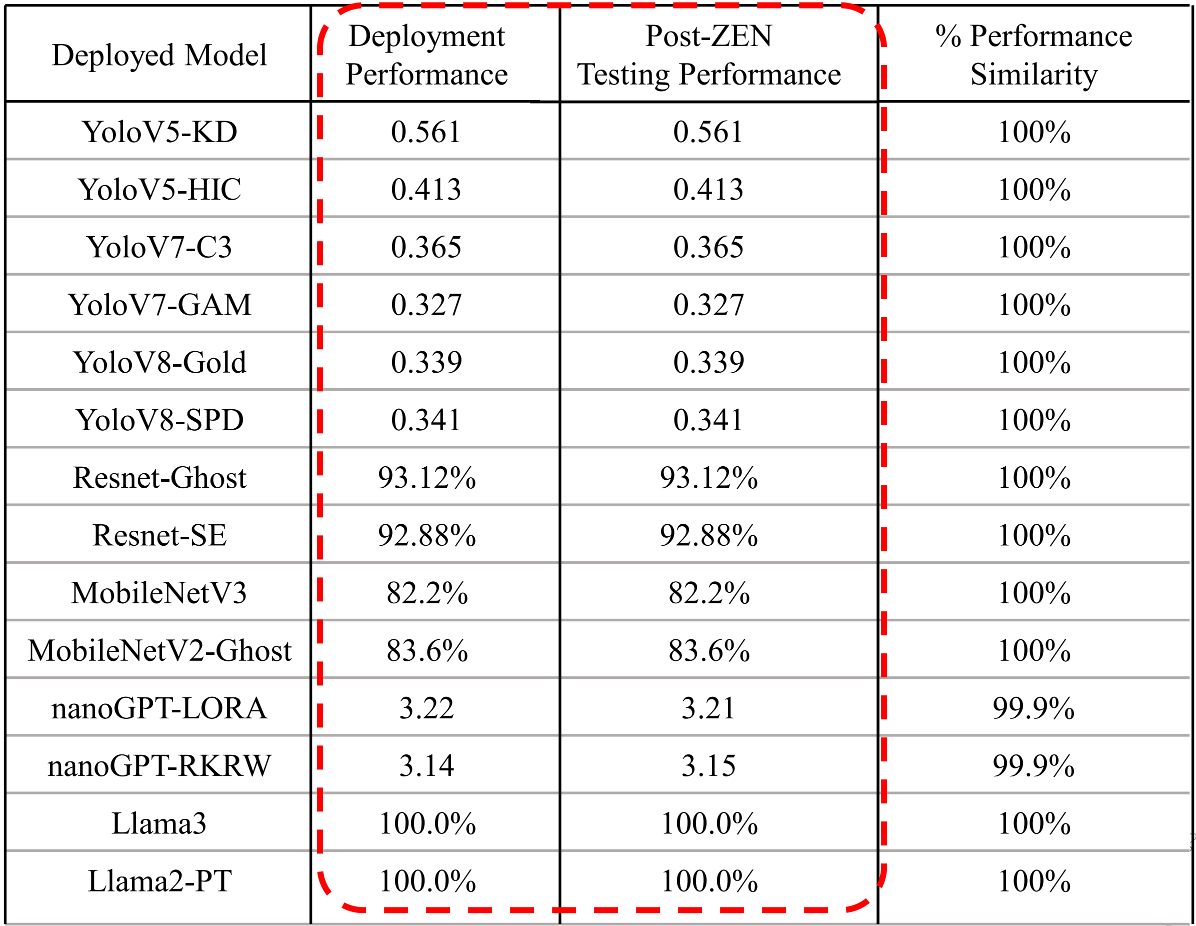


Deployed Model	Deployment Performance	Post-ZEN Testing Performance	% Performance Similarity
YoloV5-KD	0.561	0.561	100%
YoloV5-HIC	0.413	0.413	100%
YoloV7-C3	0.365	0.365	100%
YoloV7-GAM	0.327	0.327	100%
YoloV8-Gold	0.339	0.339	100%
YoloV8-SPD	0.341	0.341	100%
Resnet-Ghost	93.12%	93.12%	100%
Resnet-SE	92.88%	92.88%	100%
MobileNetV3	82.2%	82.2%	100%
MobileNetV2-Ghost	83.6%	83.6%	100%
nanoGPT-LORA	3.22	3.21	99.9%
nanoGPT-RKRW	3.14	3.15	99.9%
Llama3	100.0%	100.0%	100%
Llama2-PT	100.0%	100.0%	100%

ZEN Post Patching Performance

To check the success of model rehosting, we once again measure the pre/post deployment performance

Pre-/Post- performance for customized models matches performance in deployment!



Deployed Model	Deployment Performance	Post-ZEN Testing Performance	% Performance Similarity
YoloV5-KD	0.561	0.561	100%
YoloV5-HIC	0.413	0.413	100%
YoloV7-C3	0.365	0.365	100%
YoloV7-GAM	0.327	0.327	100%
YoloV8-Gold	0.339	0.339	100%
YoloV8-SPD	0.341	0.341	100%
Resnet-Ghost	93.12%	93.12%	100%
Resnet-SE	92.88%	92.88%	100%
MobileNetV3	82.2%	82.2%	100%
MobileNetV2-Ghost	83.6%	83.6%	100%
nanoGPT-LORA	3.22	3.21	99.9%
nanoGPT-RKRW	3.14	3.15	99.9%
Llama3	100.0%	100.0%	100%
Llama2-PT	100.0%	100.0%	100%

ZEN Post Patching Performance

To check the success of model rehosting, we once again measure the pre/post deployment performance

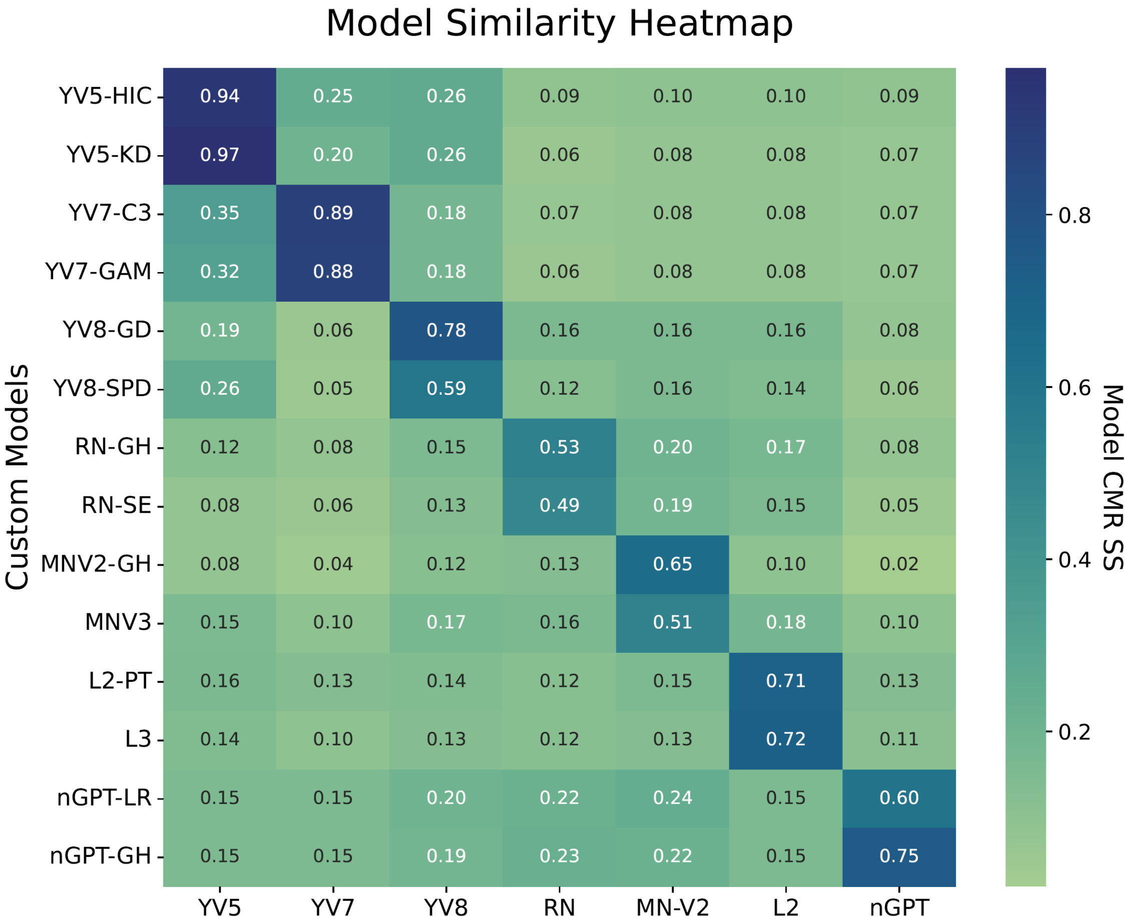
Pre-/Post- performance for customized models matches performance in deployment!

Similarity in performance between Pre/Post ZEN models is at least 99.9% similar (slight difference in loss measurement for nanoGPT)

Deployed Model	Deployment Performance	Post-ZEN Testing Performance	% Performance Similarity
YoloV5-KD	0.561	0.561	100%
YoloV5-HIC	0.413	0.413	100%
YoloV7-C3	0.365	0.365	100%
YoloV7-GAM	0.327	0.327	100%
YoloV8-Gold	0.339	0.339	100%
YoloV8-SPD	0.341	0.341	100%
Resnet-Ghost	93.12%	93.12%	100%
Resnet-SE	92.88%	92.88%	100%
MobileNetV3	82.2%	82.2%	100%
MobileNetV2-Ghost	83.6%	83.6%	100%
nanoGPT-LORA	3.22	3.21	99.9%
nanoGPT-RKRW	3.14	3.15	99.9%
Llama3	100.0%	100.0%	100%
Llama2-PT	100.0%	100.0%	100%

ZEN’s Model Attribution!

Model similarity heatmap between customized/base models



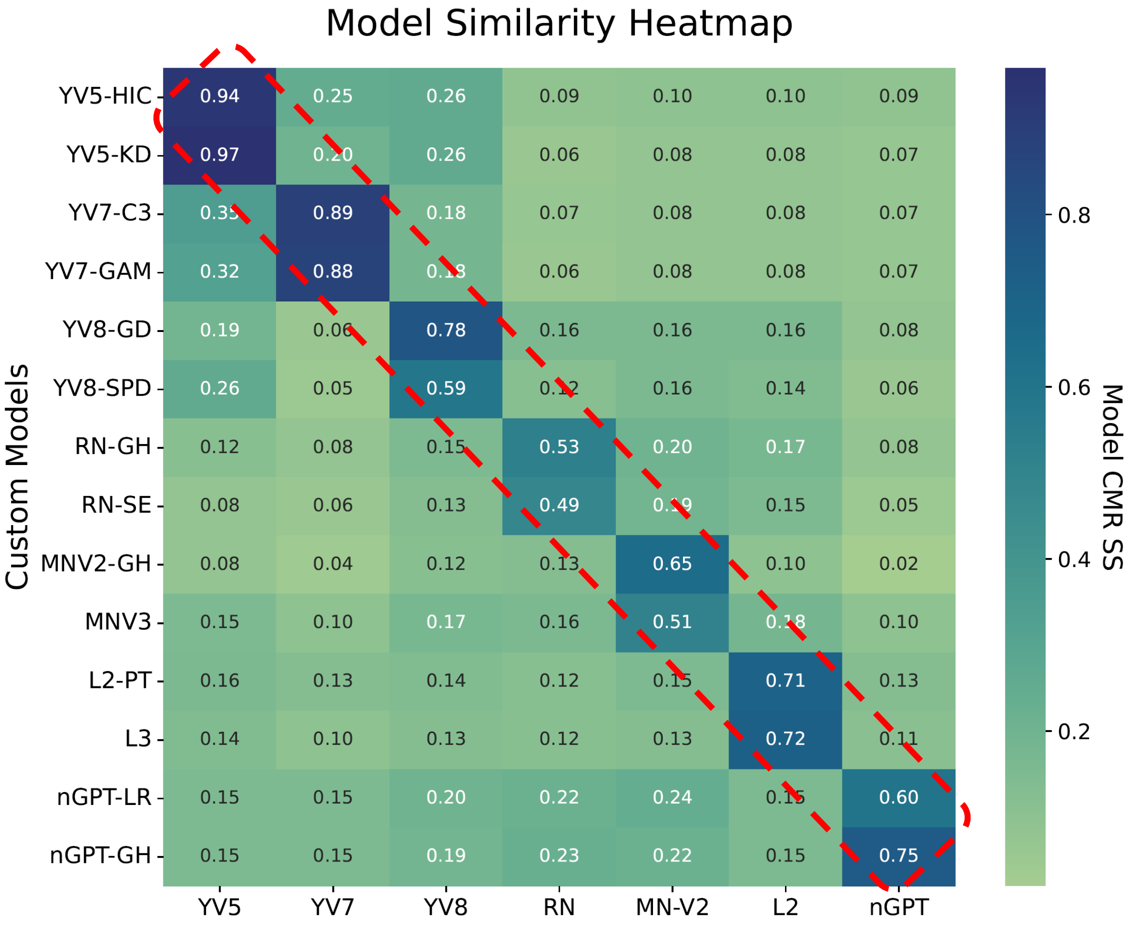
Model Similarity Heatmap

Custom Models	YV5	YV7	YV8	RN	MN-V2	L2	nGPT
YV5-HIC	0.94	0.25	0.26	0.09	0.10	0.10	0.09
YV5-KD	0.97	0.20	0.26	0.06	0.08	0.08	0.07
YV7-C3	0.35	0.89	0.18	0.07	0.08	0.08	0.07
YV7-GAM	0.32	0.88	0.18	0.06	0.08	0.08	0.07
YV8-GD	0.19	0.06	0.78	0.16	0.16	0.16	0.08
YV8-SPD	0.26	0.05	0.59	0.12	0.16	0.14	0.06
RN-GH	0.12	0.08	0.15	0.53	0.20	0.17	0.08
RN-SE	0.08	0.06	0.13	0.49	0.19	0.15	0.05
MNV2-GH	0.08	0.04	0.12	0.13	0.65	0.10	0.02
MNV3	0.15	0.10	0.17	0.16	0.51	0.18	0.10
L2-PT	0.16	0.13	0.14	0.12	0.15	0.71	0.13
L3	0.14	0.10	0.13	0.12	0.13	0.72	0.11
nGPT-LR	0.15	0.15	0.20	0.22	0.24	0.15	0.60
nGPT-GH	0.15	0.15	0.19	0.23	0.22	0.15	0.75

ZEN’s Model Attribution!

Model similarity heatmap between customized/base models

ZEN correctly matches customized to base models with upwards of 97% similarity

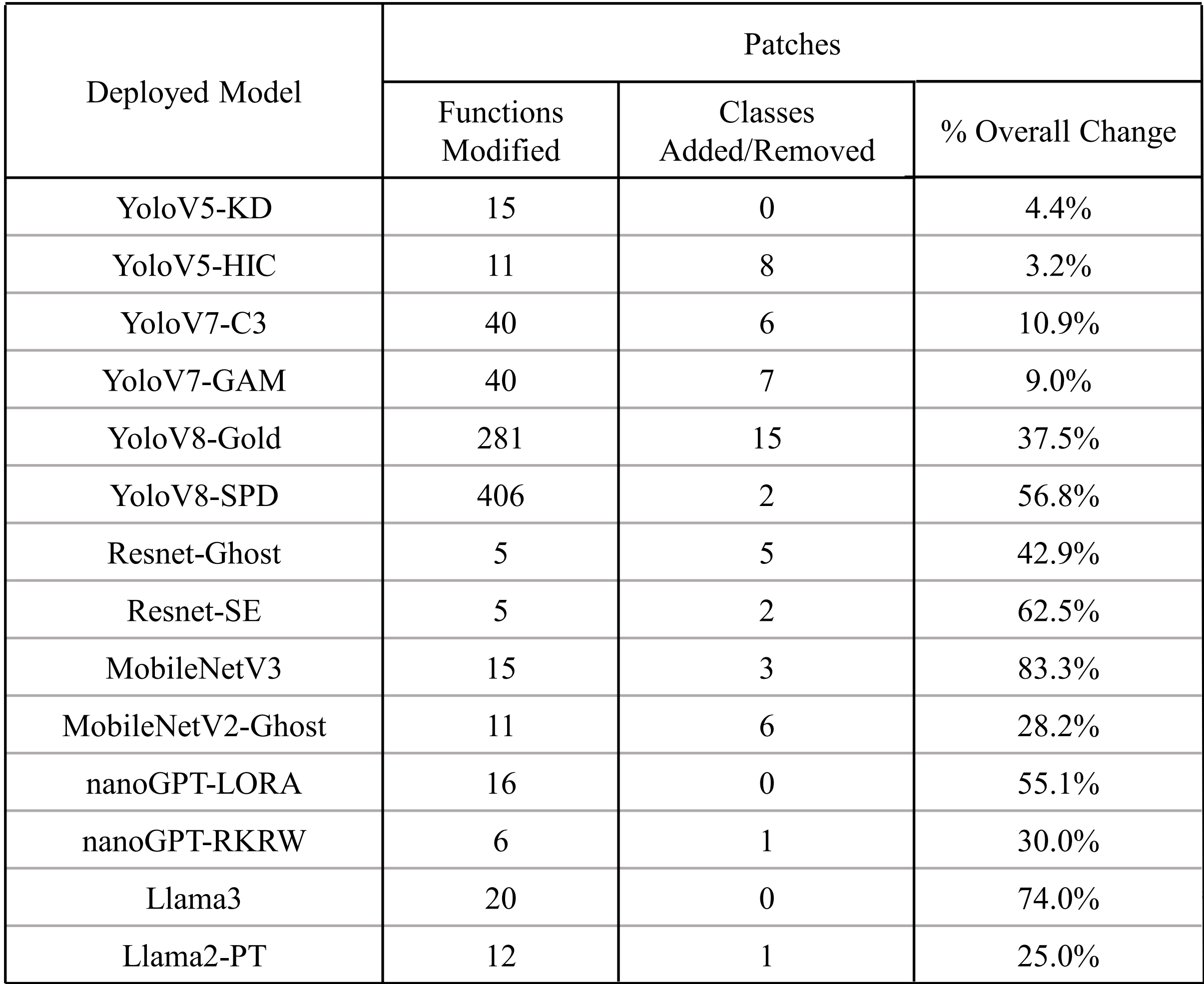


Model Similarity Heatmap

	YV5	YV7	YV8	RN	MN-V2	L2	nGPT
YV5-HIC	0.94	0.25	0.26	0.09	0.10	0.10	0.09
YV5-KD	0.97	0.20	0.26	0.06	0.08	0.08	0.07
YV7-C3	0.35	0.89	0.18	0.07	0.08	0.08	0.07
YV7-GAM	0.32	0.88	0.18	0.06	0.08	0.08	0.07
YV8-GD	0.19	0.06	0.78	0.16	0.16	0.16	0.08
YV8-SPD	0.26	0.05	0.59	0.12	0.16	0.14	0.06
RN-GH	0.12	0.08	0.15	0.53	0.20	0.17	0.08
RN-SE	0.08	0.06	0.13	0.49	0.19	0.15	0.05
MNV2-GH	0.08	0.04	0.12	0.13	0.65	0.10	0.02
MNV3	0.15	0.10	0.17	0.16	0.51	0.18	0.10
L2-PT	0.16	0.13	0.14	0.12	0.15	0.71	0.13
L3	0.14	0.10	0.13	0.12	0.13	0.72	0.11
nGPT-LR	0.15	0.15	0.20	0.22	0.24	0.15	0.60
nGPT-GH	0.15	0.15	0.19	0.23	0.22	0.15	0.75

ZEN Patch Generation

Show patches generated for all of our tested customized models

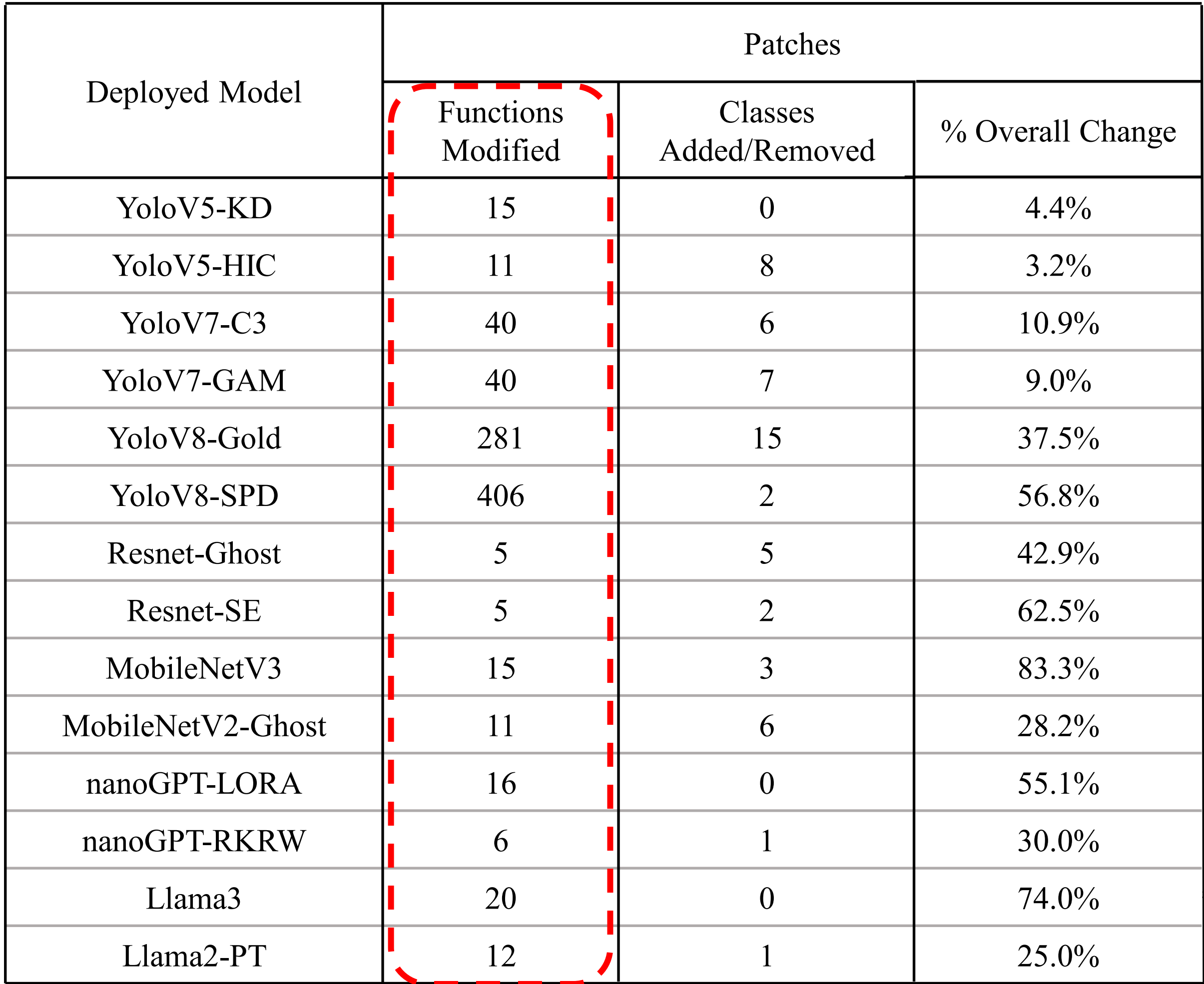


Deployed Model	Patches		
	Functions Modified	Classes Added/Removed	% Overall Change
YoloV5-KD	15	0	4.4%
YoloV5-HIC	11	8	3.2%
YoloV7-C3	40	6	10.9%
YoloV7-GAM	40	7	9.0%
YoloV8-Gold	281	15	37.5%
YoloV8-SPD	406	2	56.8%
Resnet-Ghost	5	5	42.9%
Resnet-SE	5	2	62.5%
MobileNetV3	15	3	83.3%
MobileNetV2-Ghost	11	6	28.2%
nanoGPT-LORA	16	0	55.1%
nanoGPT-RKRW	6	1	30.0%
Llama3	20	0	74.0%
Llama2-PT	12	1	25.0%

ZEN Patch Generation

Show patches generated for all of our tested customized models

Upwards of 406 functions modified, patches successfully generated



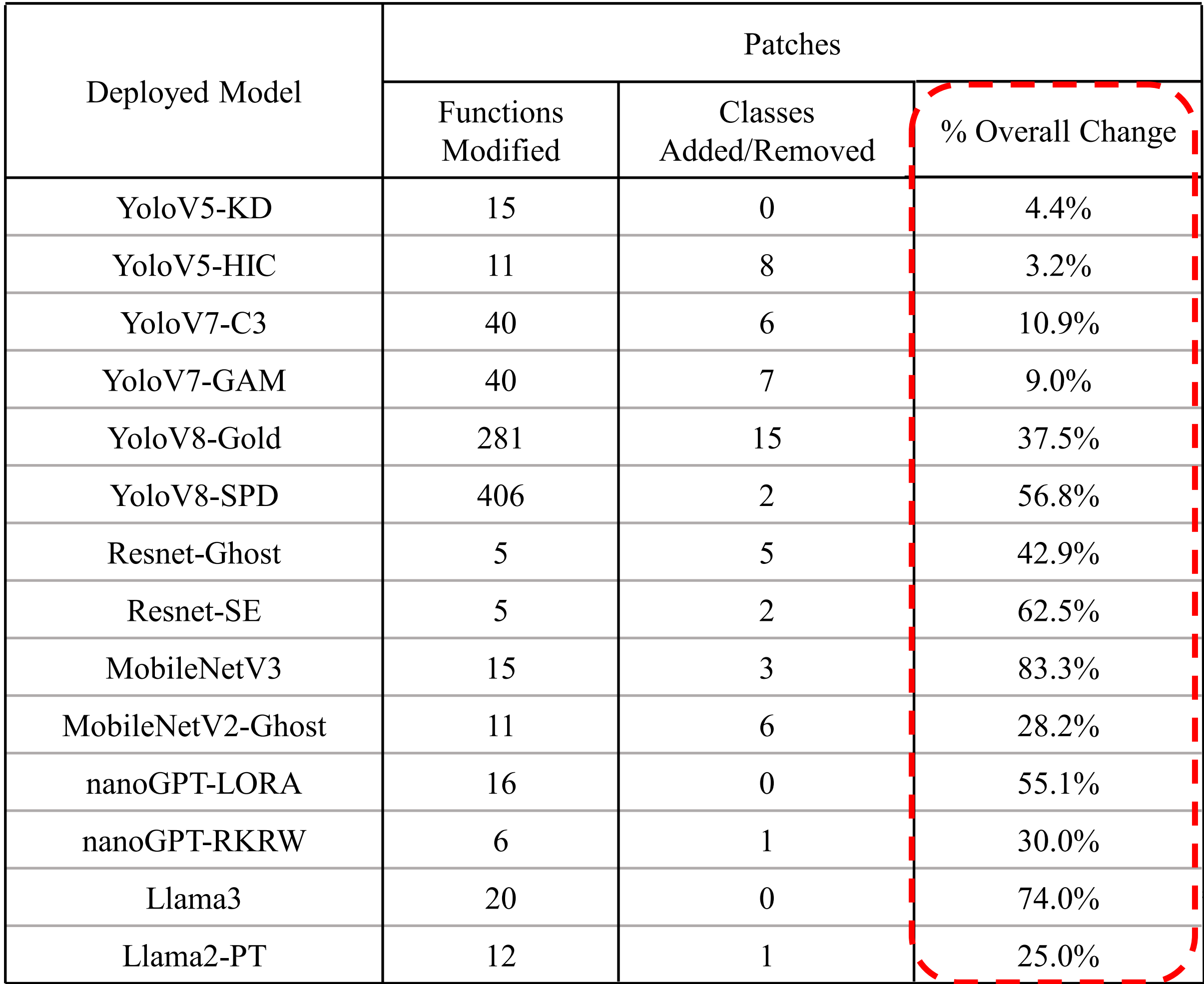
Deployed Model	Patches		
	Functions Modified	Classes Added/Removed	% Overall Change
YoloV5-KD	15	0	4.4%
YoloV5-HIC	11	8	3.2%
YoloV7-C3	40	6	10.9%
YoloV7-GAM	40	7	9.0%
YoloV8-Gold	281	15	37.5%
YoloV8-SPD	406	2	56.8%
Resnet-Ghost	5	5	42.9%
Resnet-SE	5	2	62.5%
MobileNetV3	15	3	83.3%
MobileNetV2-Ghost	11	6	28.2%
nanoGPT-LORA	16	0	55.1%
nanoGPT-RKRW	6	1	30.0%
Llama3	20	0	74.0%
Llama2-PT	12	1	25.0%

ZEN Patch Generation

Show patches generated for all of our tested customized models

Upwards of 406 functions modified, patches successfully generated

Upwards of 83.3% change from the base model!



Deployed Model	Patches		
	Functions Modified	Classes Added/Removed	% Overall Change
YoloV5-KD	15	0	4.4%
YoloV5-HIC	11	8	3.2%
YoloV7-C3	40	6	10.9%
YoloV7-GAM	40	7	9.0%
YoloV8-Gold	281	15	37.5%
YoloV8-SPD	406	2	56.8%
Resnet-Ghost	5	5	42.9%
Resnet-SE	5	2	62.5%
MobileNetV3	15	3	83.3%
MobileNetV2-Ghost	11	6	28.2%
nanoGPT-LORA	16	0	55.1%
nanoGPT-RKRW	6	1	30.0%
Llama3	20	0	74.0%
Llama2-PT	12	1	25.0%

Much More in the Paper!

- Copyright Investigation Case Study
- Potential Evasion and Defense Strategies
- Breakdown of Model Attribution Metrics

Many thanks!

GTRI

Georgia Tech Research Institute

Artifact Evaluated

NDSS SYMPOSIUM

Available

Functional

Reproduced

Achieving Zen: Combining Mathematical and Programmatic Deep Learning Model Representations for Attribution and Reuse

David Oygenblik, Dinko Dermendzhiev, Filippos Sofias, Mingxuan Yao, Haichuan Xu, Runze Zhang, Jeman Park, Amit Kumar Sikder, Brendan Saltaformaggio

NDSS 2026

https://github.com/CyFI-Lab-Public/ZEN.git

Thank you!
Questions?