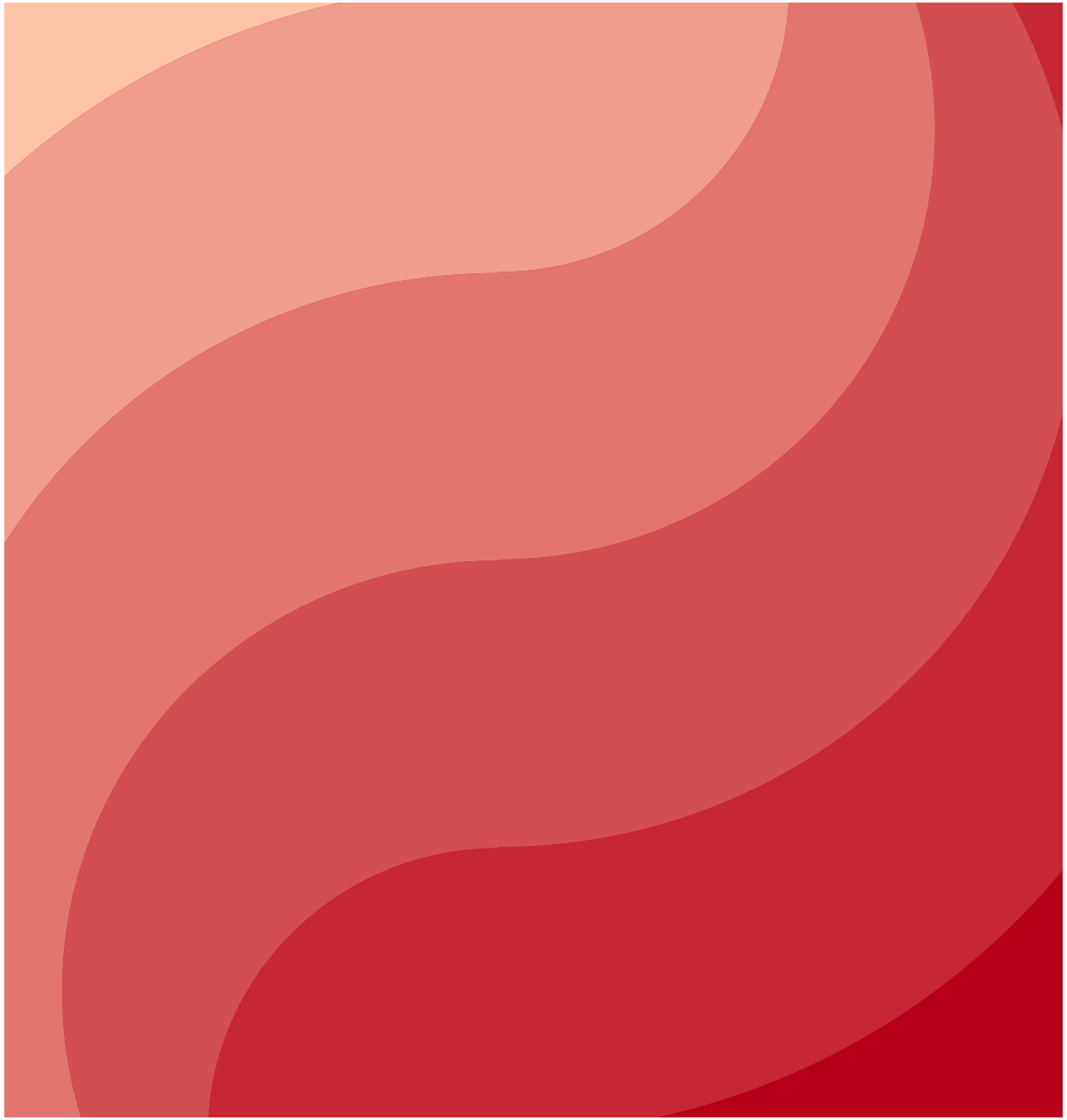


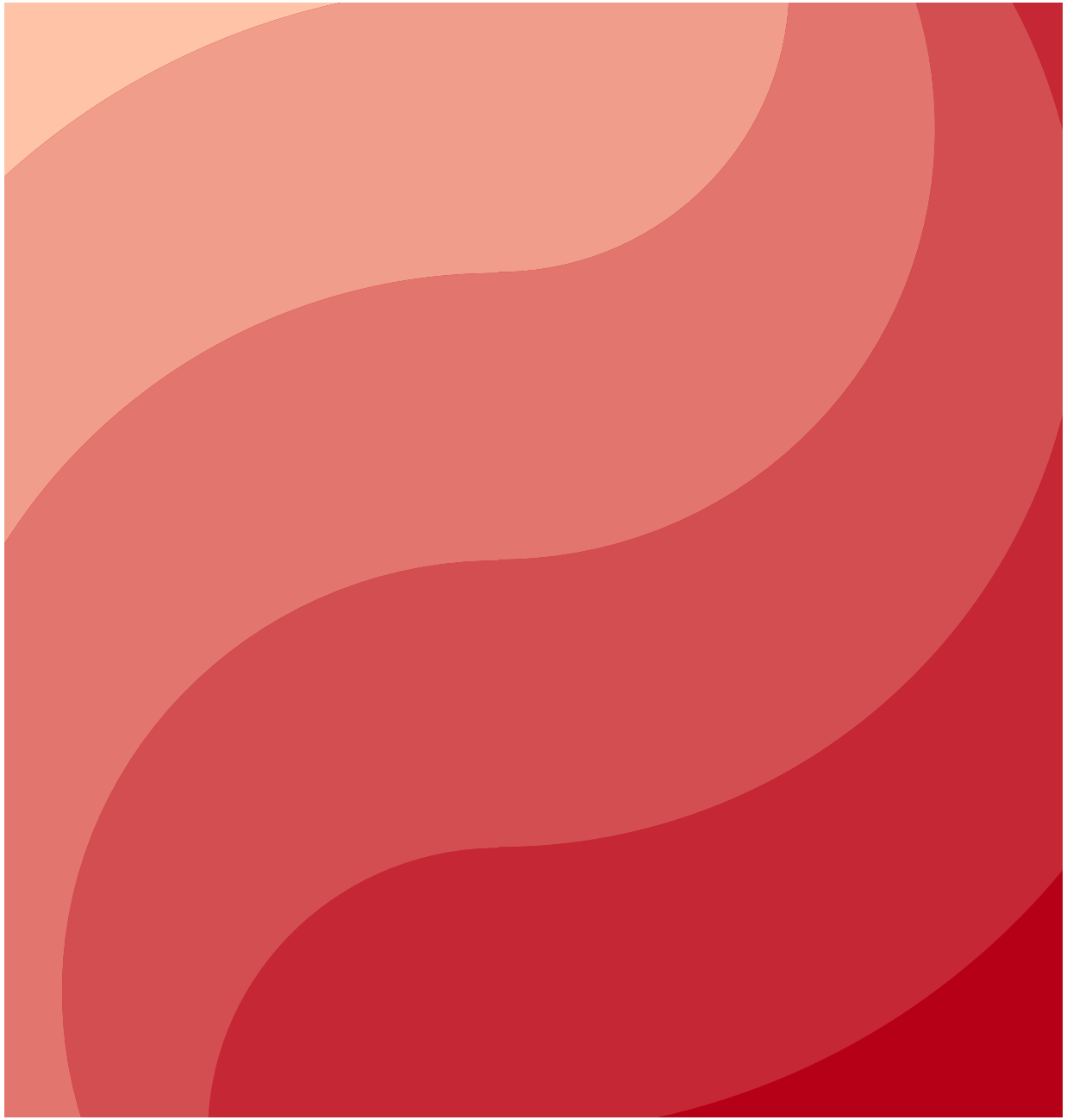
Adapt the Neural Net For the Hardware

Unleash the Full Potential of AI Hardware

Lucas Liebenwein SM’18 PhD’21
Principal Research Scientist, OmniML

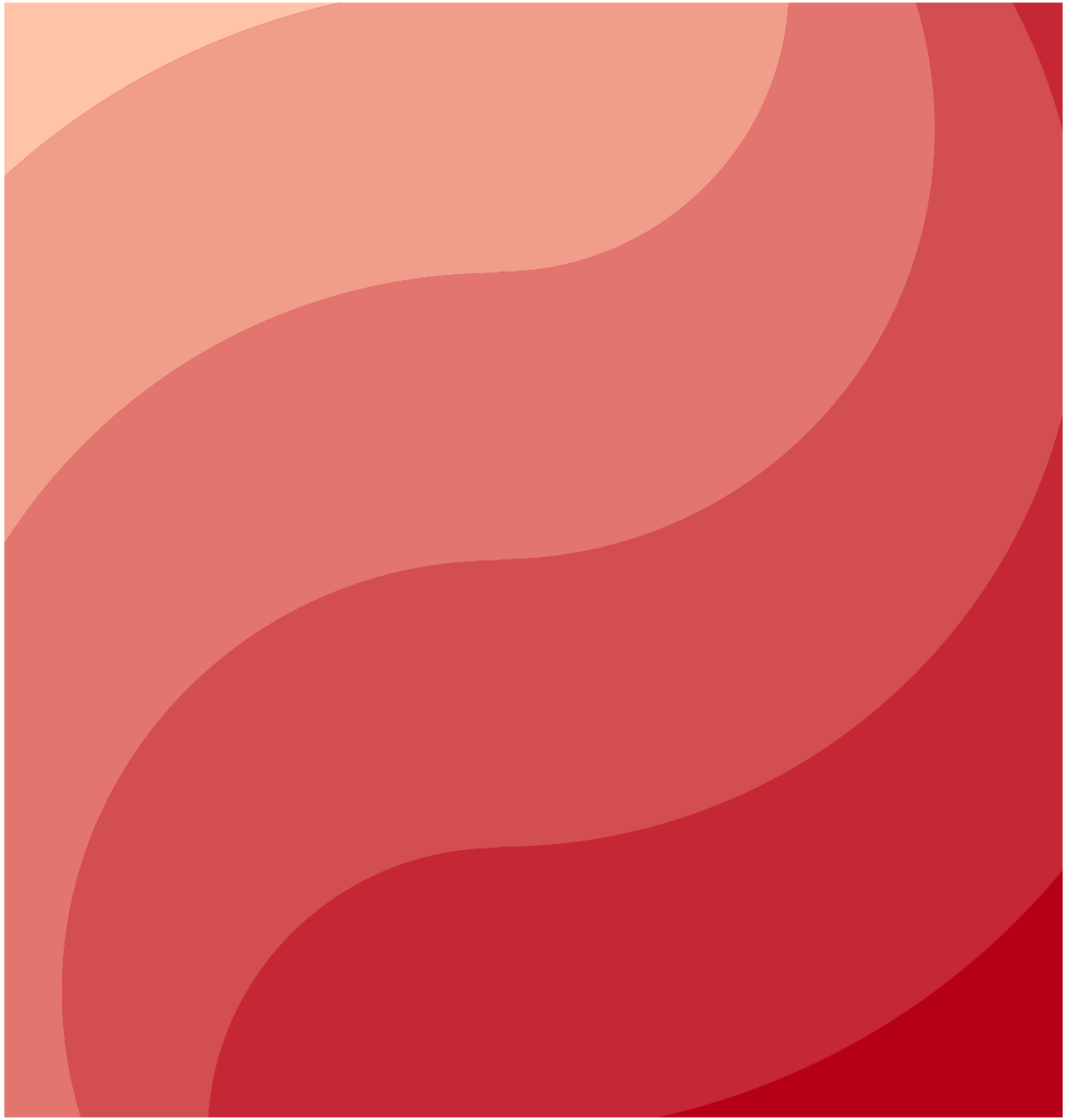
Agenda

- Introduction of OmniML
- Omnimizer: an MLOps platform for edge AI
- Case studies
- Live Demo



Agenda

- Introduction of OmniML
- Omnimizer: an MLOps platform for edge AI
- Case studies
- Live Demo



About OmniML

Adapt the algorithm for the hardware.

OmniML accelerates the design and deployment of high performance, hardware-aware AI across all devices.

Faster to production, easier development

Self-service tool, giving developers full control

Faster model inference for any model

Unleash the full power of existing hardware

AI is coming to the edge …

ADAS

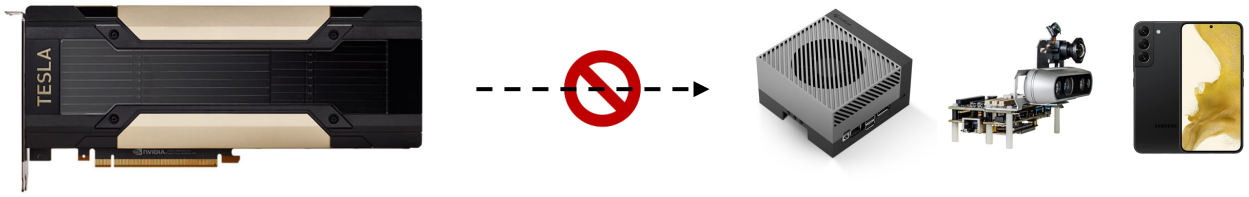
Robotics

IoT

AR/VR

… but **deployment** remains a blocking factor

Big gaps to deploy AI on diverse hardware platforms



Training

Inference

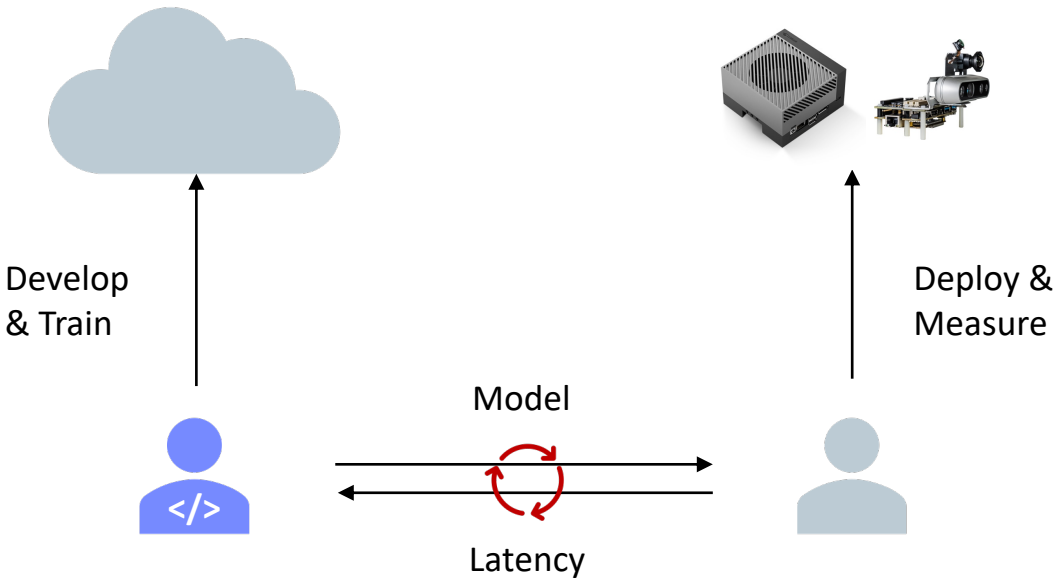
Long time-to-market for deploying existing AI models to hardware

Performance gaps and high hardware cost

Heavy R&D investment and low ROI

Issues in Existing Workflows

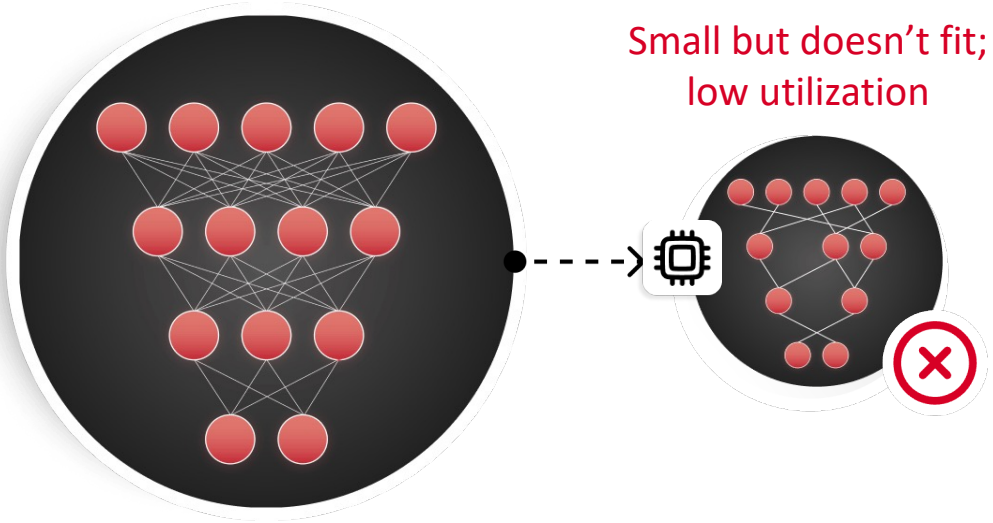
Slow time-to-market with a fundamental problem



Slow Feedback

Slow To Production

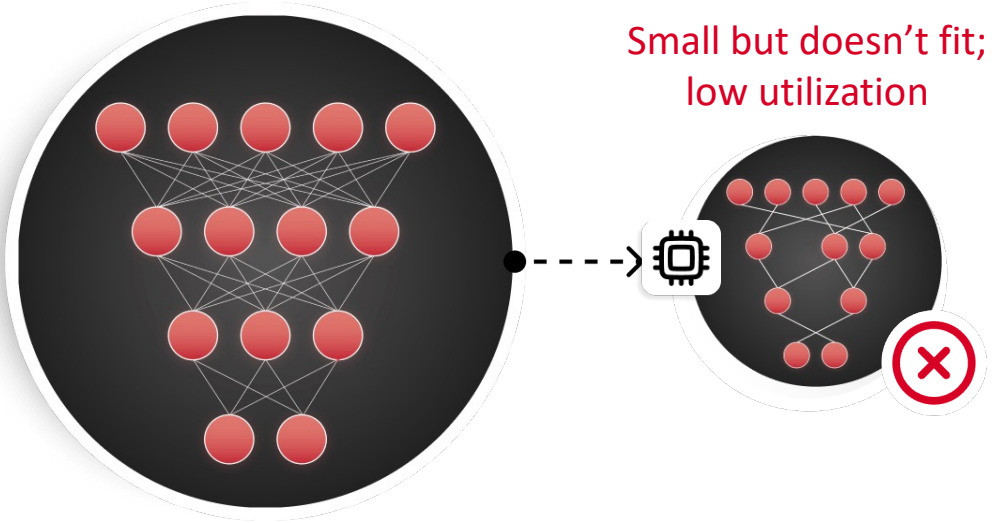
Reason for the big gaps in AI deployment



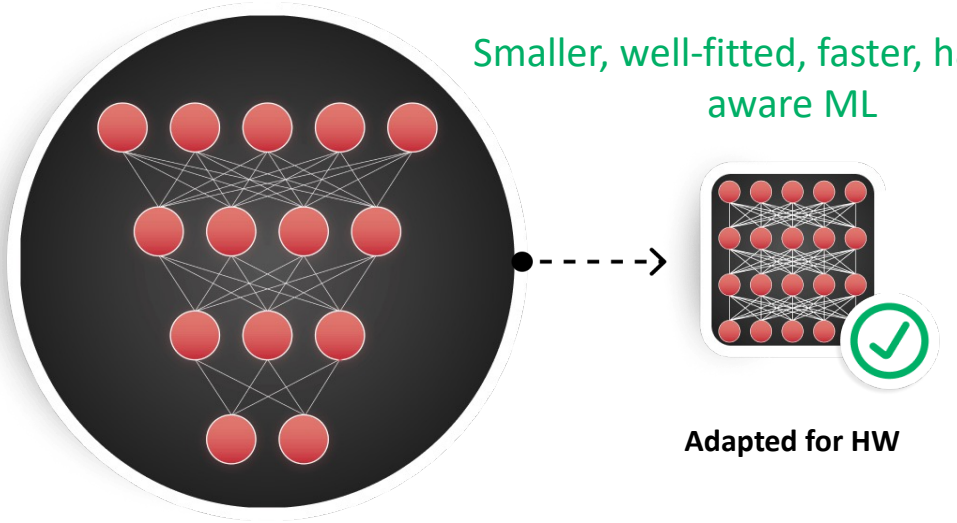
Issues in Existing Workflows

Slow time-to-market with a fundamental problem

Reason for the big gaps in AI deployment



Omnimizer

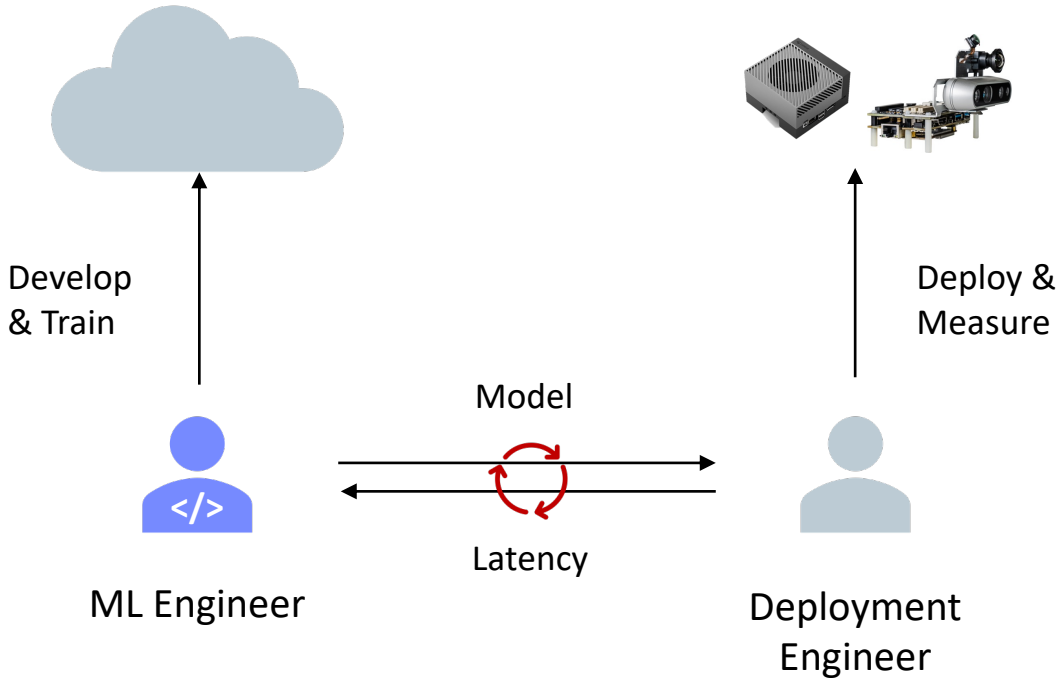


Omnimizer Workflow

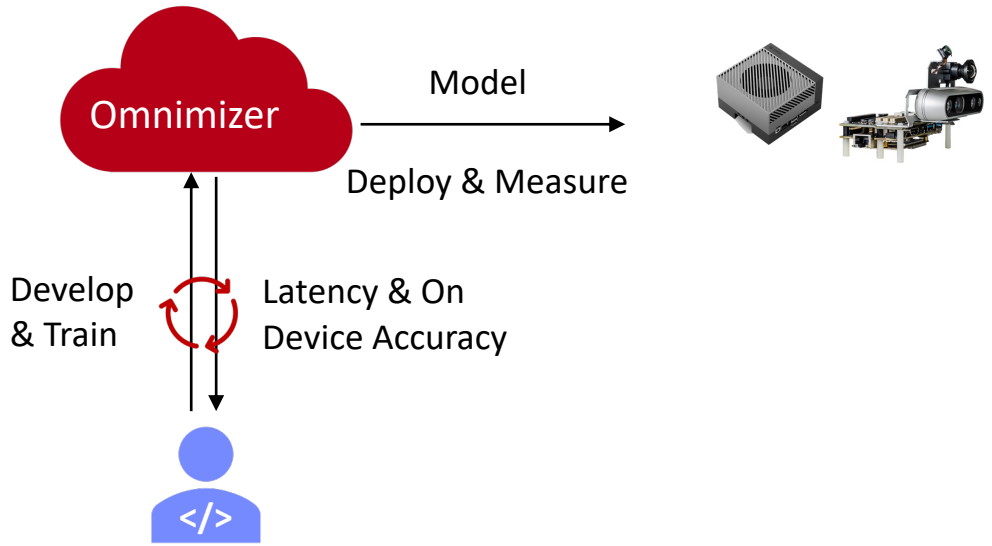
Omnimizer enables faster time to production for edge AI

Existing Workflow

Slow Feedback Slow To Production

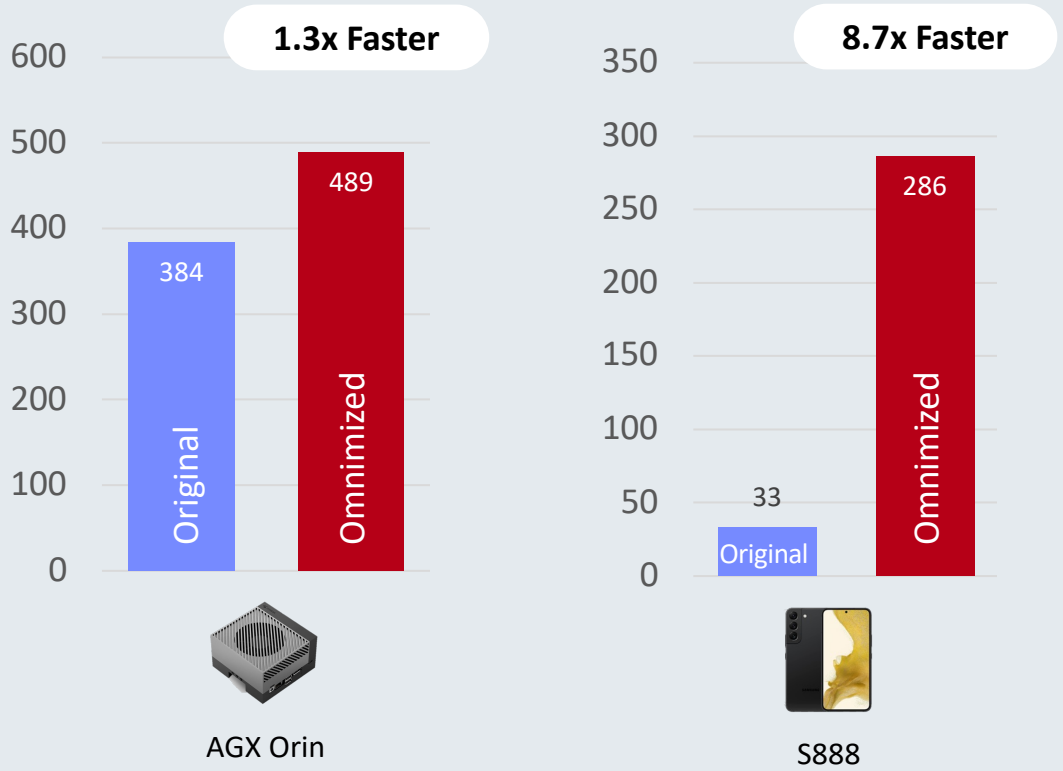


Omnimizer Workflow



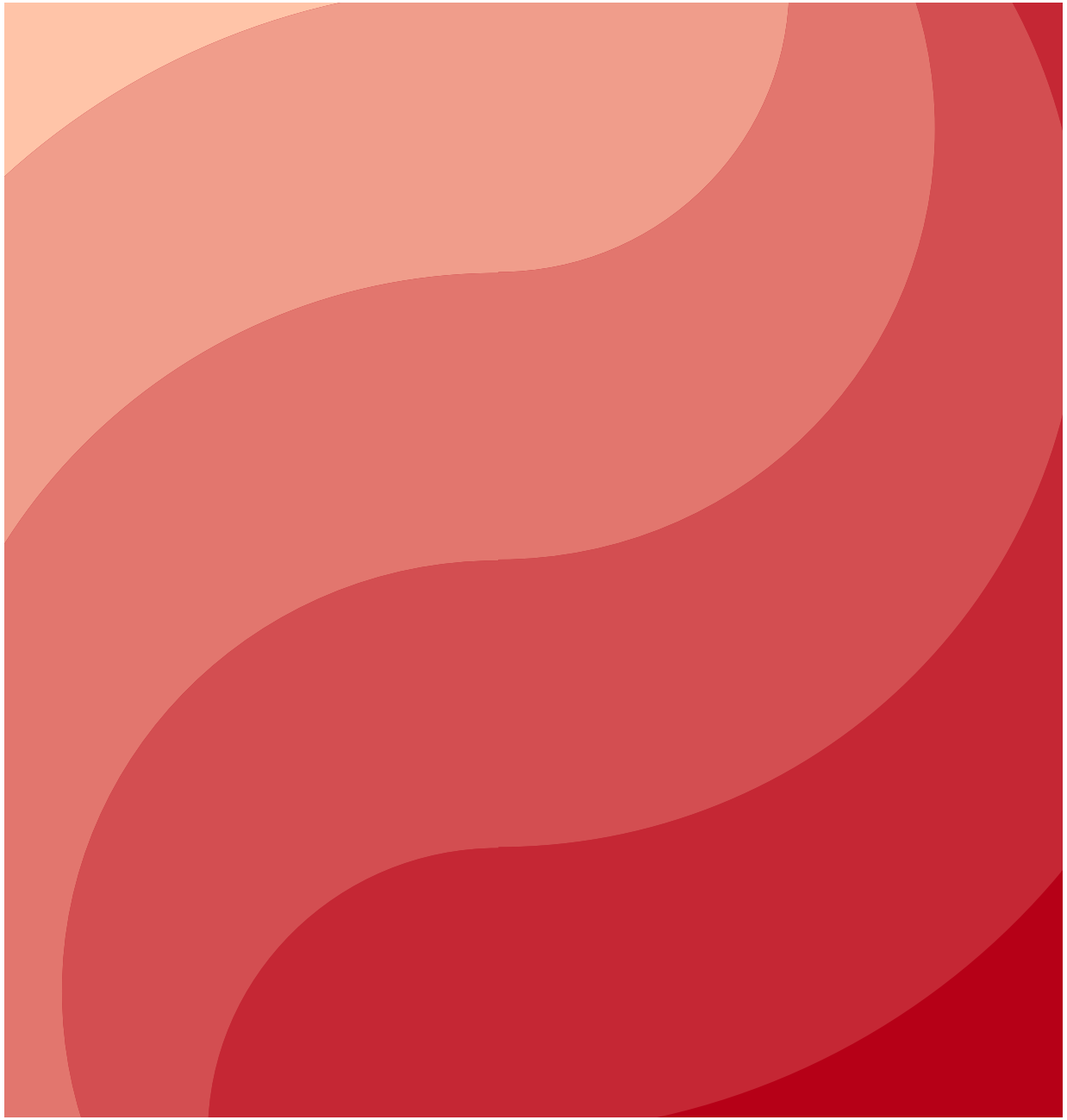
Use Case: Segmentation

Using DDRNet39 with one robotics customer



Agenda

- Introduction of OmniML
- Omnimizer: an MLOps platform for edge AI
- Case studies
- Live Demo



Omnimizer

Bridging the gap between AI development and deployment

Omnimizer

Bridging the gap between AI development and deployment

Omnimizer

Bridging the gap between AI development and deployment

ML Engineer

PyTorch +

Omnimizer Core

Omnimizer Engine

NVIDIA Qualcomm Ambarella™

Faster to production, easier development

Self-service tool, giving developers full control

Faster model inference for any model

Unleash the full power of existing hardware

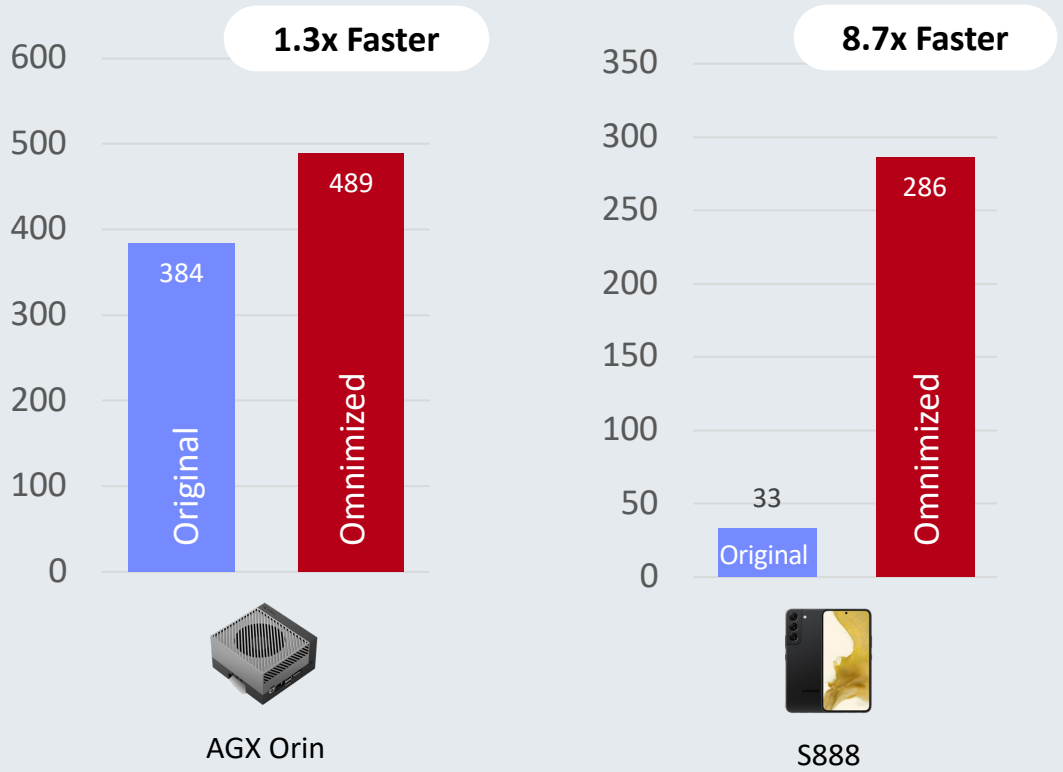
Current approaches in industry

Short-comings of current approaches

Neural Network Intelligence	Vertex AI	Omnimizer
Large collection of research algorithms	Black-box optimizer within GCP	Automatic model optimization and adaption
Manual search space design	Requires difficult proxy task	Native integration into any training environment
No deployment, no hardware-in-the-loop	No integration, GCP only	Instant deployment and hardware-in-the-loop
https://github.com/microsoft/nni/	https://cloud.google.com/vertex-ai	https://www.omniml.ai

Use Case: Segmentation

Using DDRNet39 with one robotics customer



Omnimizer Workflow

Effortlessly Adapt DDRNet for S888

Step 1: Setup

```python
from omnimizer import engine

deployment = {
    "device": "S888",
    "precision": "int8",
}


engine.get_latency(model, deployment)
```

Fast on-device latency testing within a cloud-native environment

Baseline Model

30.5 ms

Latency

Train & eval on Cityscape dataset, scale to 512x512, no pretraining

Omnimizer Workflow

Effortlessly Adapt DDRNet for S888

Step 2: Diagnosis & Adaptation

```python
from omnimizer import engine, nas

deployment = {
    "device": "S888",
    "precision": "int8",
}

# coming soon!
adapted_model = nas.adapt(model, deployment)


engine.profile(adapted_model, deployment)
```

Omnimizer automatically adapts the model to more efficiently use the hardware

	Baseline Model	Adapted Model
Layer-wise Latency (ms)		
Total Latency (ms)	30.5 ms	7.4 ms
Speedup	1.0x	**4.1x**

Accuracy difference less than 0.2%

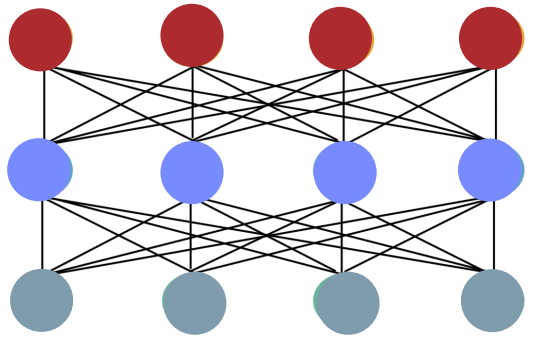
Omnimizer Workflow

Effortlessly Adapt DDRNet for S888

Step 3: Train & Optimize

```
from omnimizer import nas

omni_model = nas.omnimize(model)

train(omni_model, dataloader)
```

"Omnimize" any given model to be elastic with out-of-the-box support for custom models



AutoNAS

FastNAS

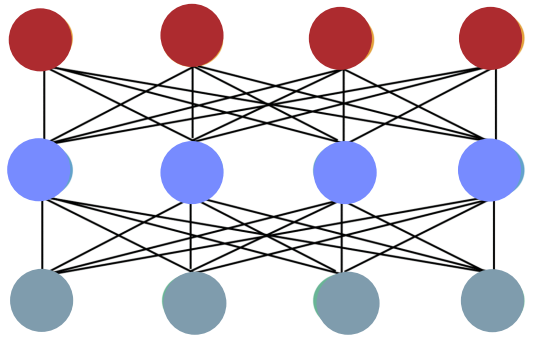
Omnimizer Workflow

Effortlessly Adapt DDRNet for S888

Step 3: Train & Optimize

```
from omnimizer import nas

omni_model = nas.omnimize(model, "autonas")

train(omni_model, dataloader)
```

"Omnimize" any given model to be elastic with out-of-the-box support for custom models



AutoNAS

3x training time

Large search space

FastNAS

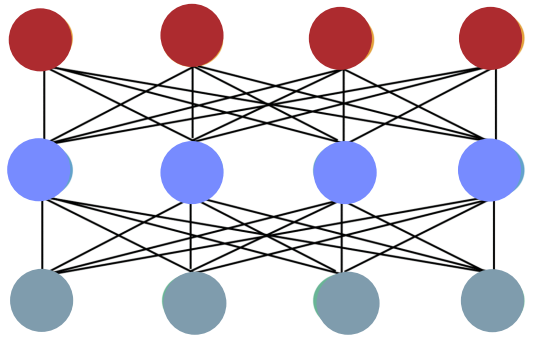
Omnimizer Workflow

Effortlessly Adapt DDRNet for S888

Step 3: Train & Optimize

```
from omnimizer import nas

omni_model = nas.omnimize(model, "fastnas")

train(omni_model, dataloader)
```

"Omnimize" any given model to be elastic with out-of-the-box support for custom models



AutoNAS

3x training time

Large search space

FastNAS

1x training time

Reduced search space

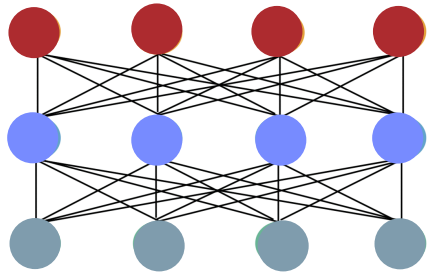
Omnimizer Workflow

Effortlessly Adapt DDRNet for S888

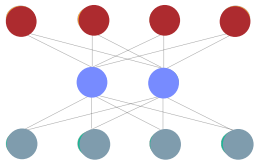
Step 4: Search

```python
from omnimizer import nas

constraints = {
    "latency"  3.0,  # ms
}

# adaptive to "autonas" and "fastnas”
sub_model = nas.search(
    model
    constraints,
    deployment,
)
```

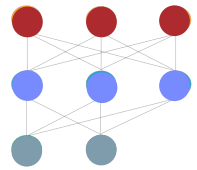
Search for the subnet with optimal accuracy-latency trade-off



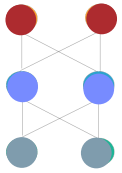
Selected Submodel



3.5 ms



3 ms



2.5 ms

Omnimizer Workflow

Effortlessly Adapt DDRNet for S888

Step 5: Deploy

```
from omnimizer import engine

device_model = engine.compile(sub_model)

device_model.get_latency()

out = device_model(sample_input)
```

Effortlessly deploy and test the search model from a cloud-native environment

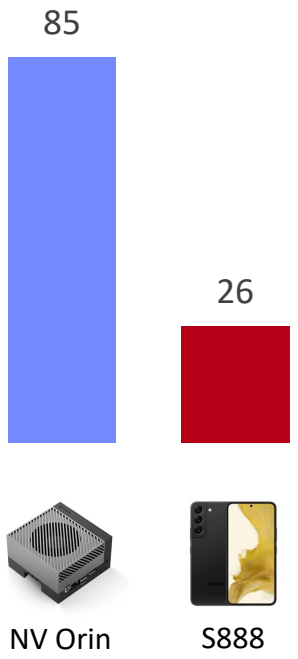
	Baseline Model	Searched Model
Total Latency (ms)	30.5 ms	3.5 ms
Speedup	1.0x	**8.7x**

Accuracy difference less than 0.2%

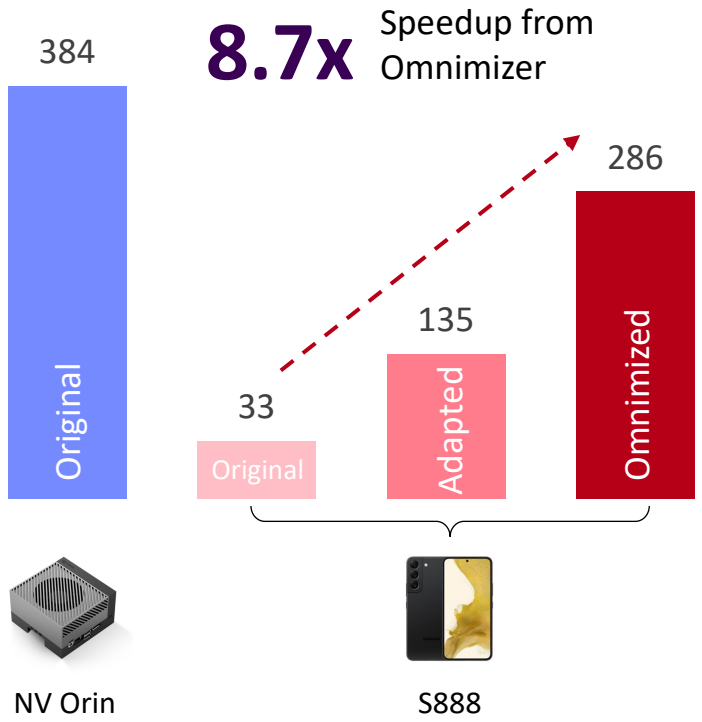
Takeaways: DDRNet

How to get the most out of your hardware device

Peak TOPs
(int8)



FPS



For the original DDRNet, the FPS/peak TOPS of S888 is

0.3x of Orin

But with Omnimizer, the FPS/peak TOPS becomes

2.4x of Orin

Takeaways: Omnimizer

Bridging the gap between AI development and deployment

Omnimizer NAS

PyTorch-native model optimization and adaption.

Integrated training for AutoNAS and FastNAS.

One-stop solution for searching optimal subnet.

Omnimizer Engine

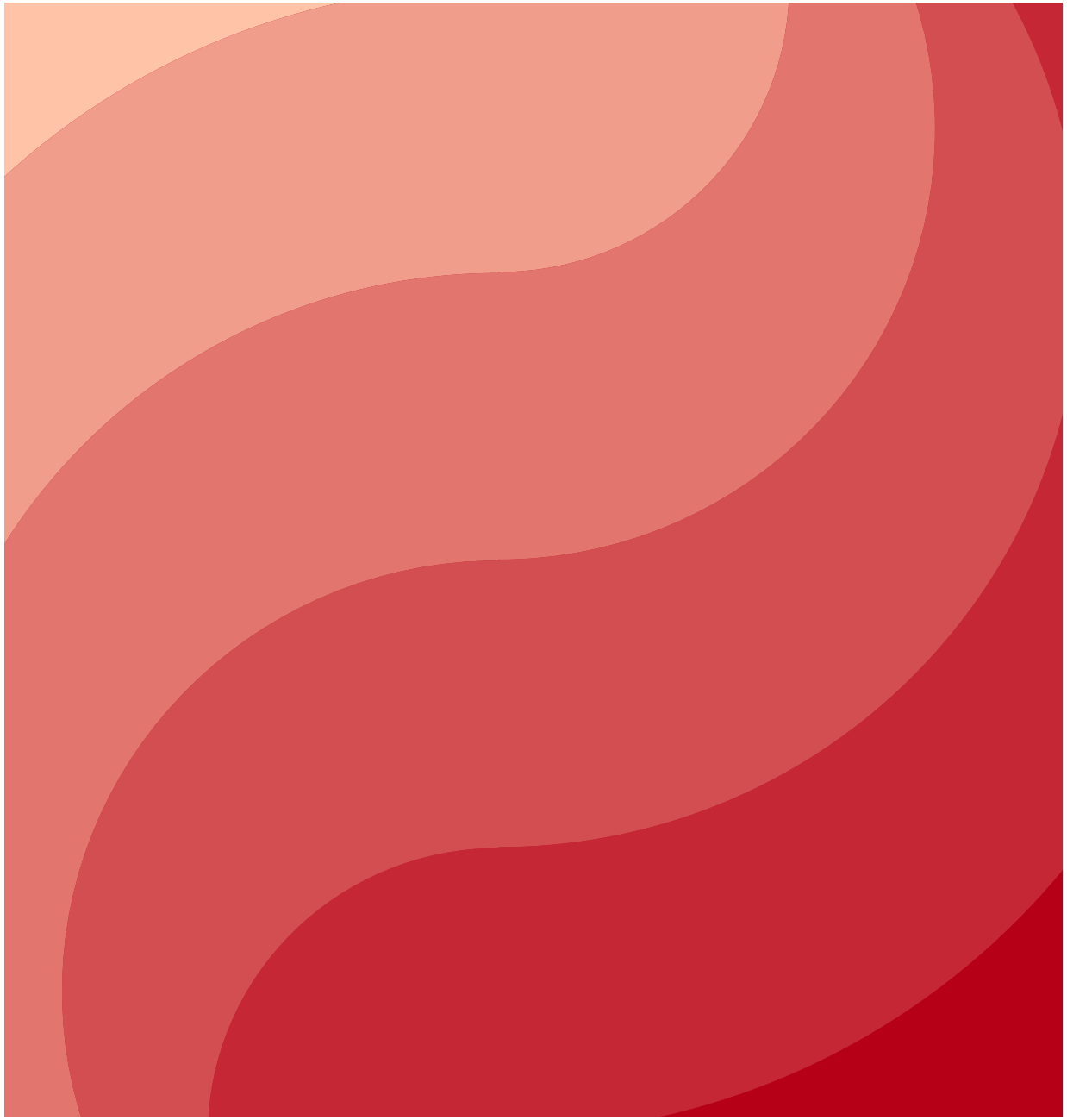
Cloud-native interface for instant deployment.

Accurately & simple model profiling from Python.

Pythonic on-device inference.

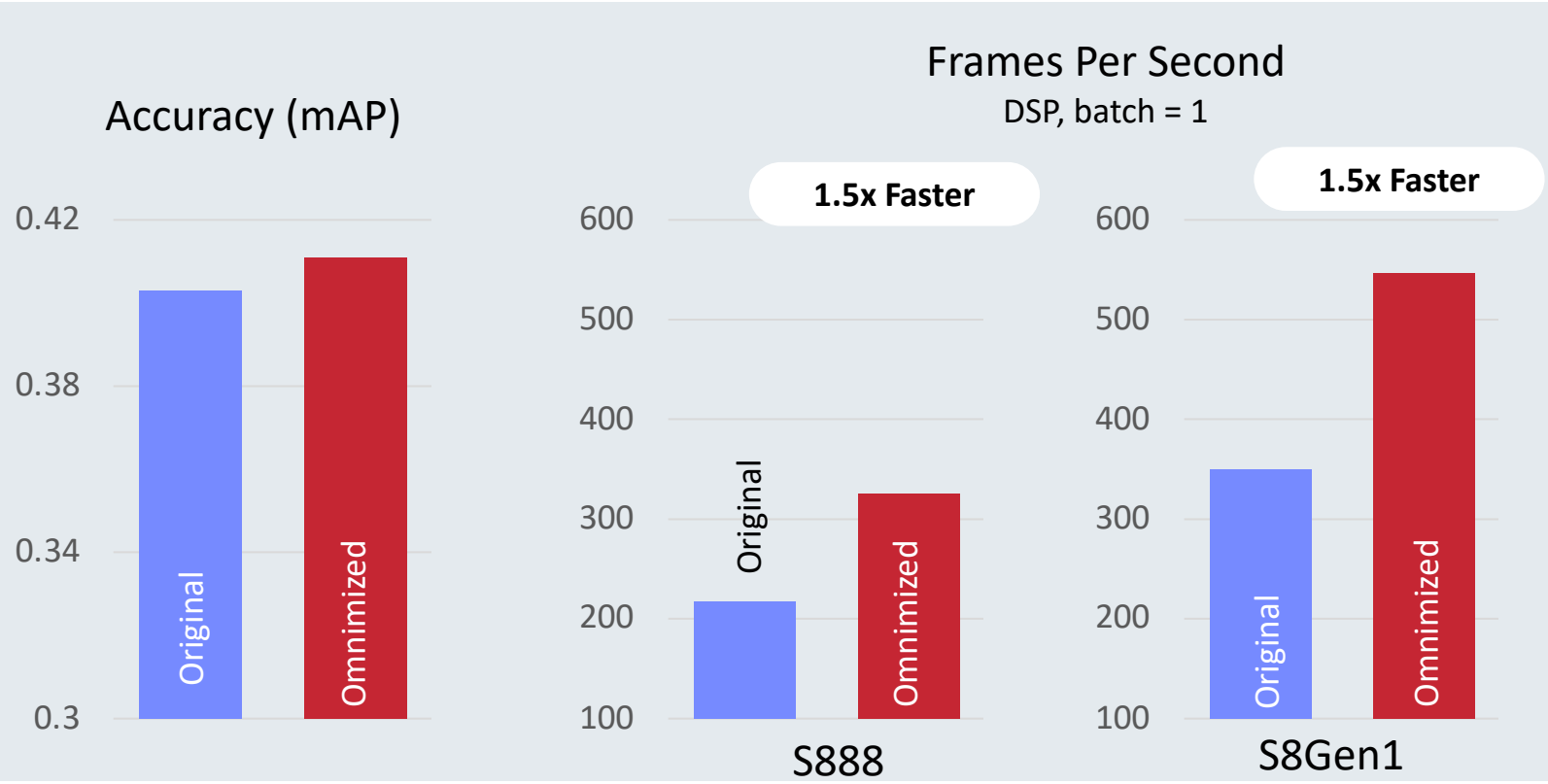
Agenda

- Introduction of OmniML
- Omnimizer: an MLOps platform for edge AI
- Case studies
- Live Demo



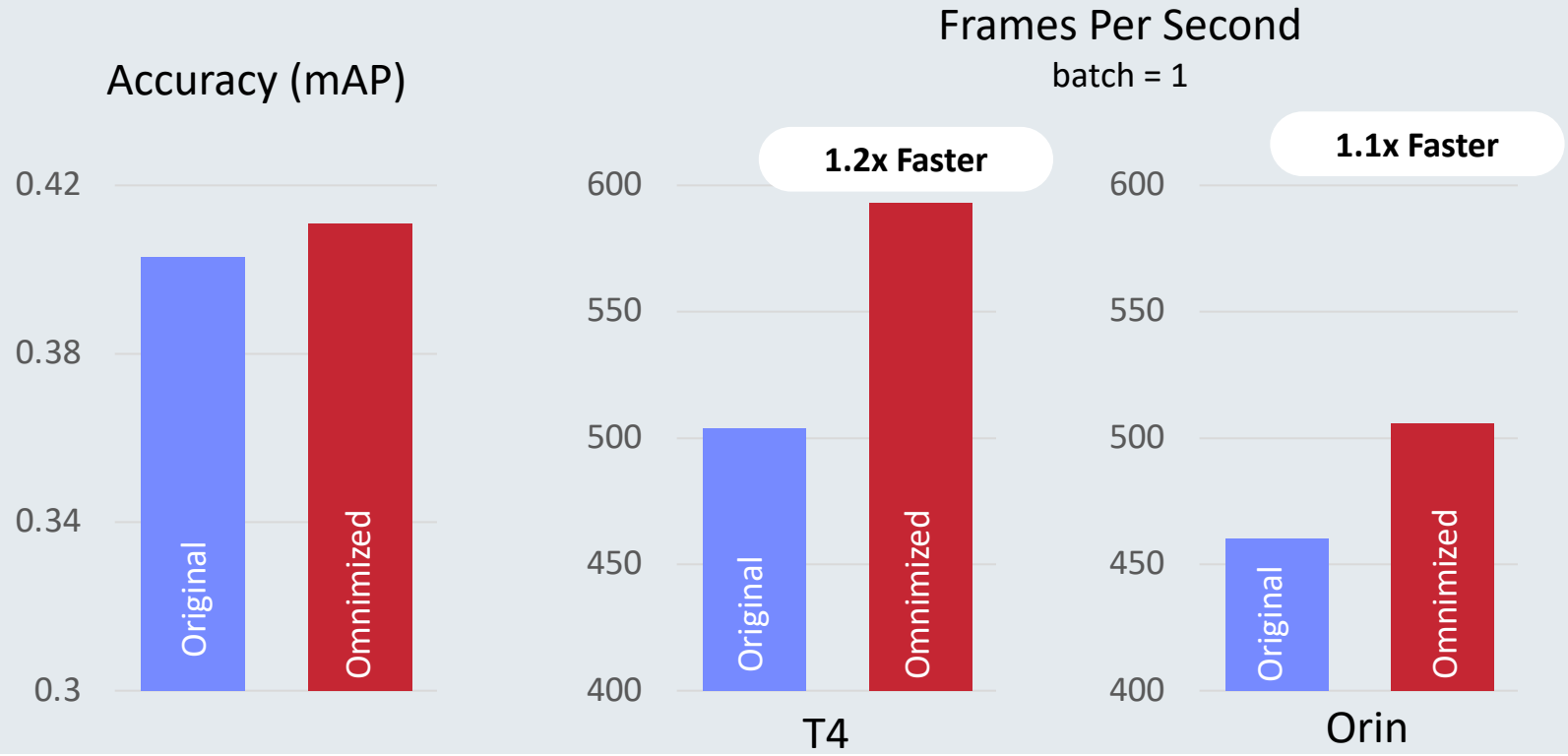
Omnimizer for YoloX

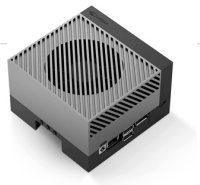
Adapt 2D Detection Model for IoT customers



Omnimizer for YoloX

Adapt 2D Detection Model for IoT customers





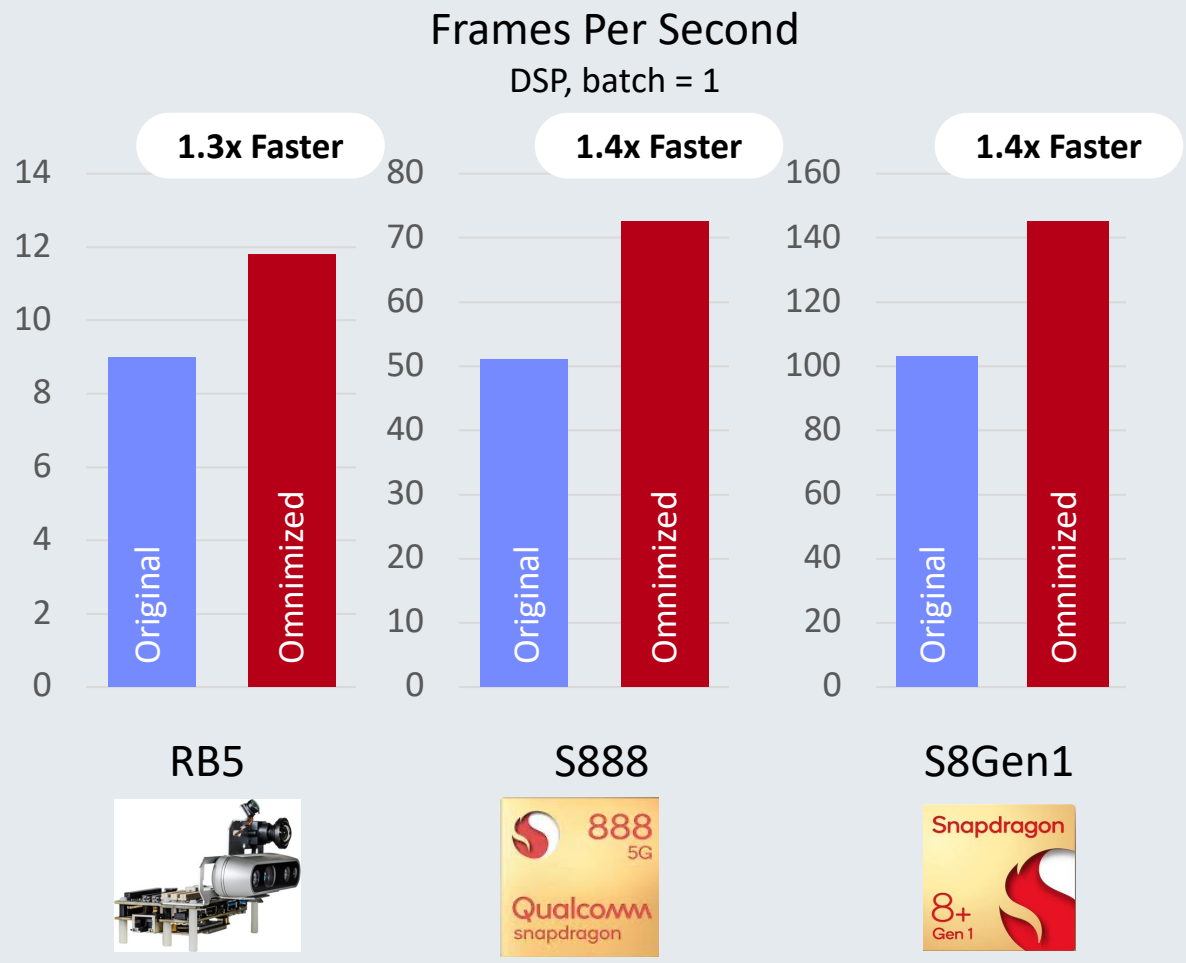
Omnimizer for YoloX

Adapt 2D Detection Model for IoT customers

Latency (int8)	Original YOLOX-S	Optimized YOLOX-S	Speedup	TDP*	Perf / Watt Before Omnimizer	Perf / Watt After Omnimizer
RB5	32.6ms	20.5ms	1.6x	10W	0.59x	0.87x
Xavier	6.37ms	5.95ms	7%	30W	1.0x (baseline)	1.0x (baseline)
S888	4.60ms	3.07ms	1.5x	10W	4.1x	5.8x
S8Gen1	2.86ms	1.83ms	1.5x	10W	6.7x	9.8x

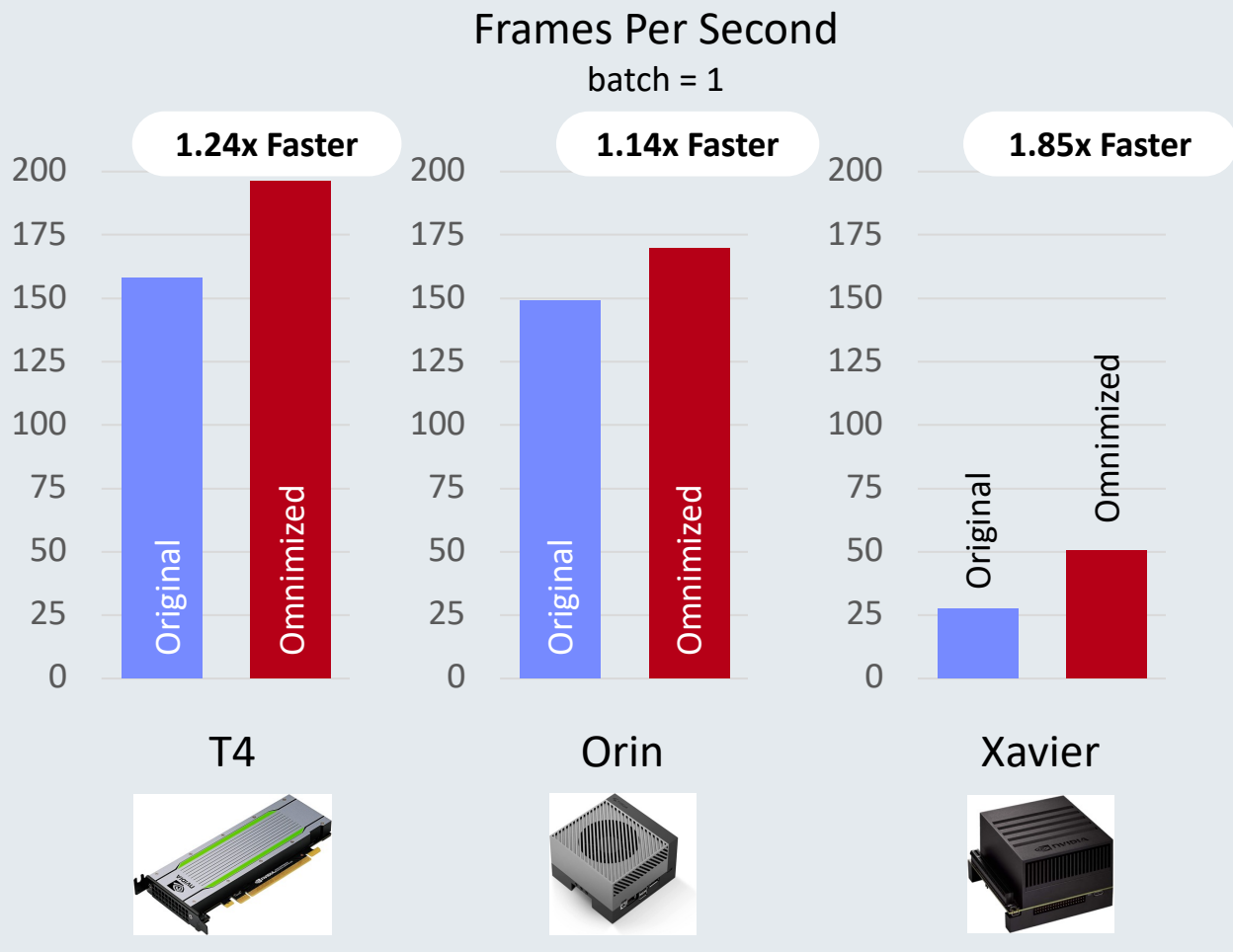
Omnimizer for DD3D

Adapt Monocular 3D detection for an ADAS customer



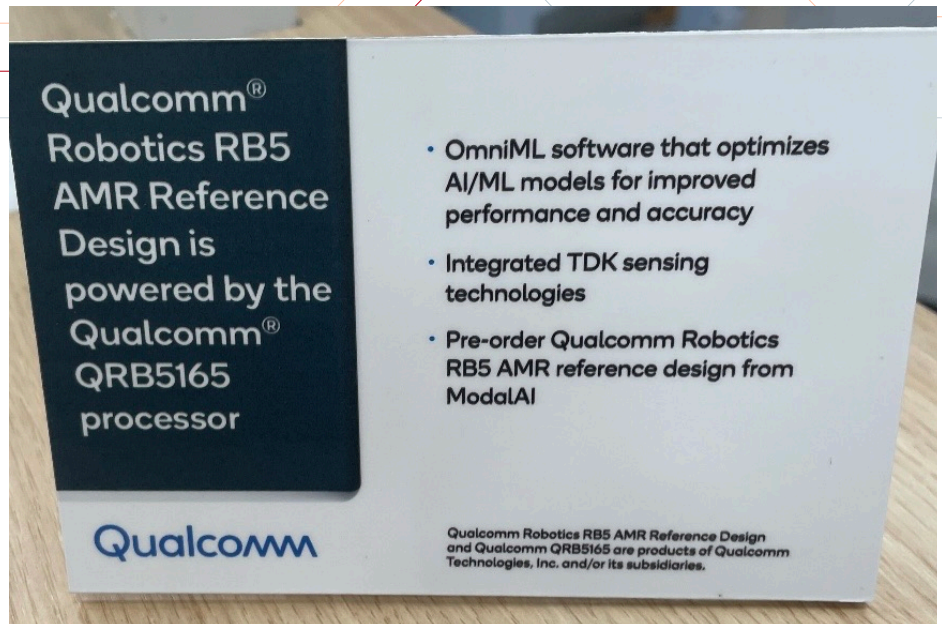
Omnimizer for DD3D

Adapt Monocular 3D detection for an ADAS customer



Collaborating with Qualcomm Robotics

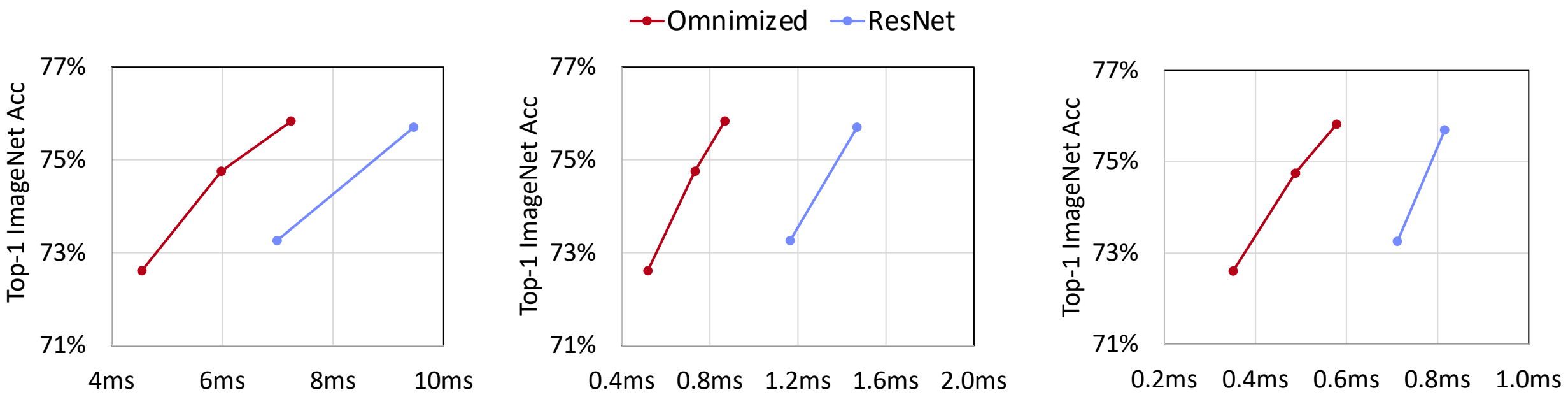
Joint-marketing during Hannover Messe 2022



contact@omniml.ai

Example: ImageNet

Improve core ML efficiency for all hardware



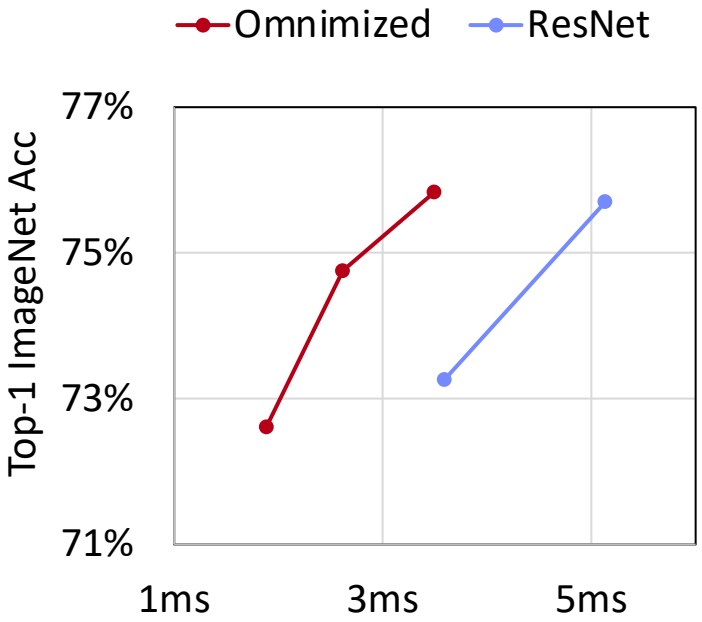
RB5 Latency

S888 Latency

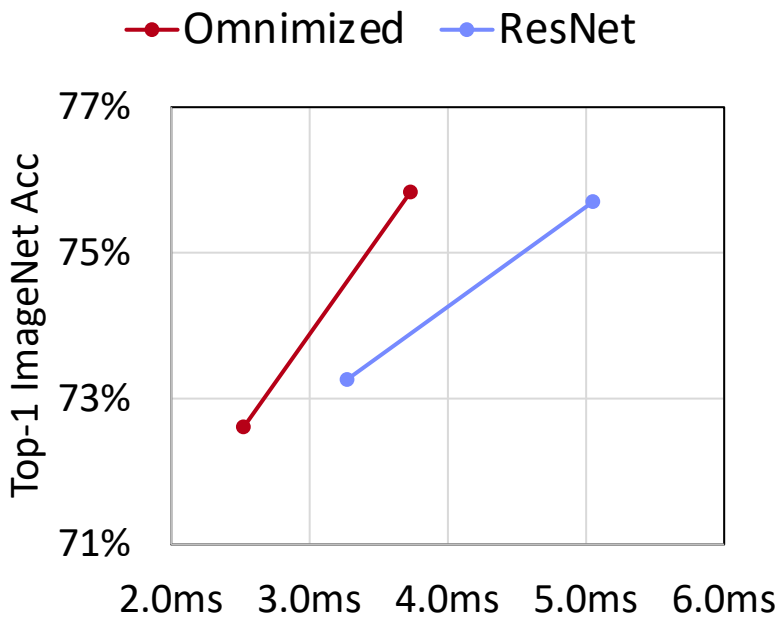
S8Gen1 Latency

Example: ImageNet

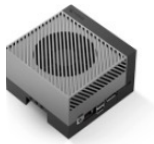
Improve core ML efficiency for all hardware



T4 Latency

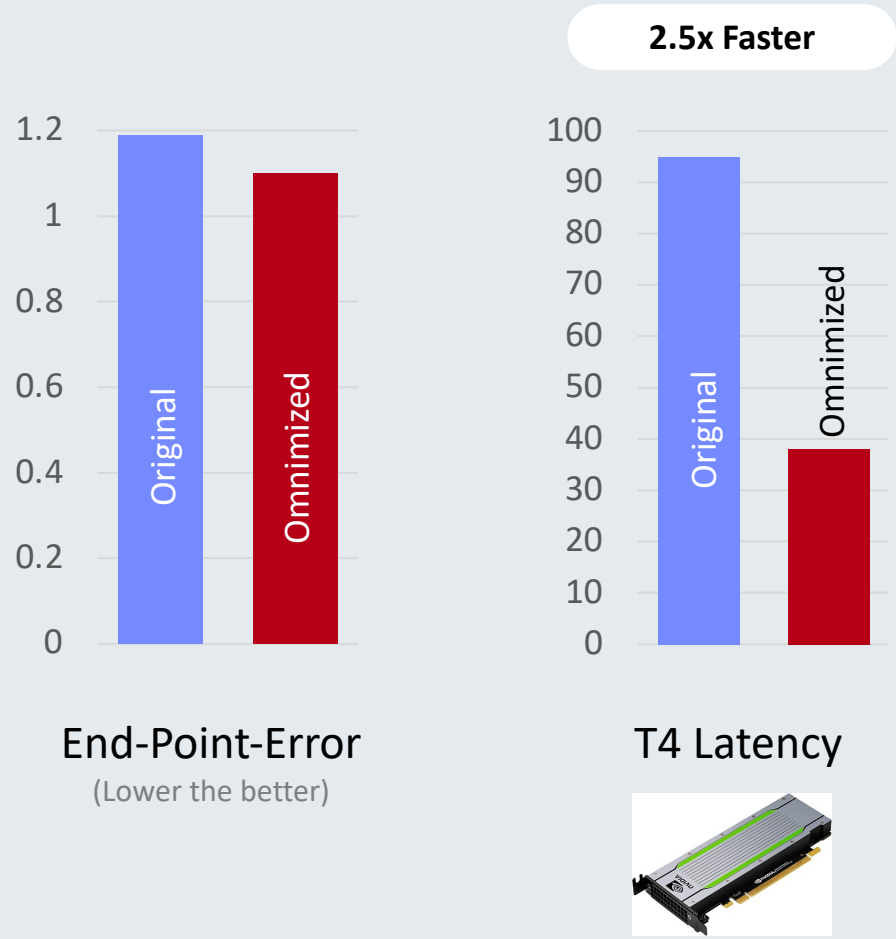


Orin Latency



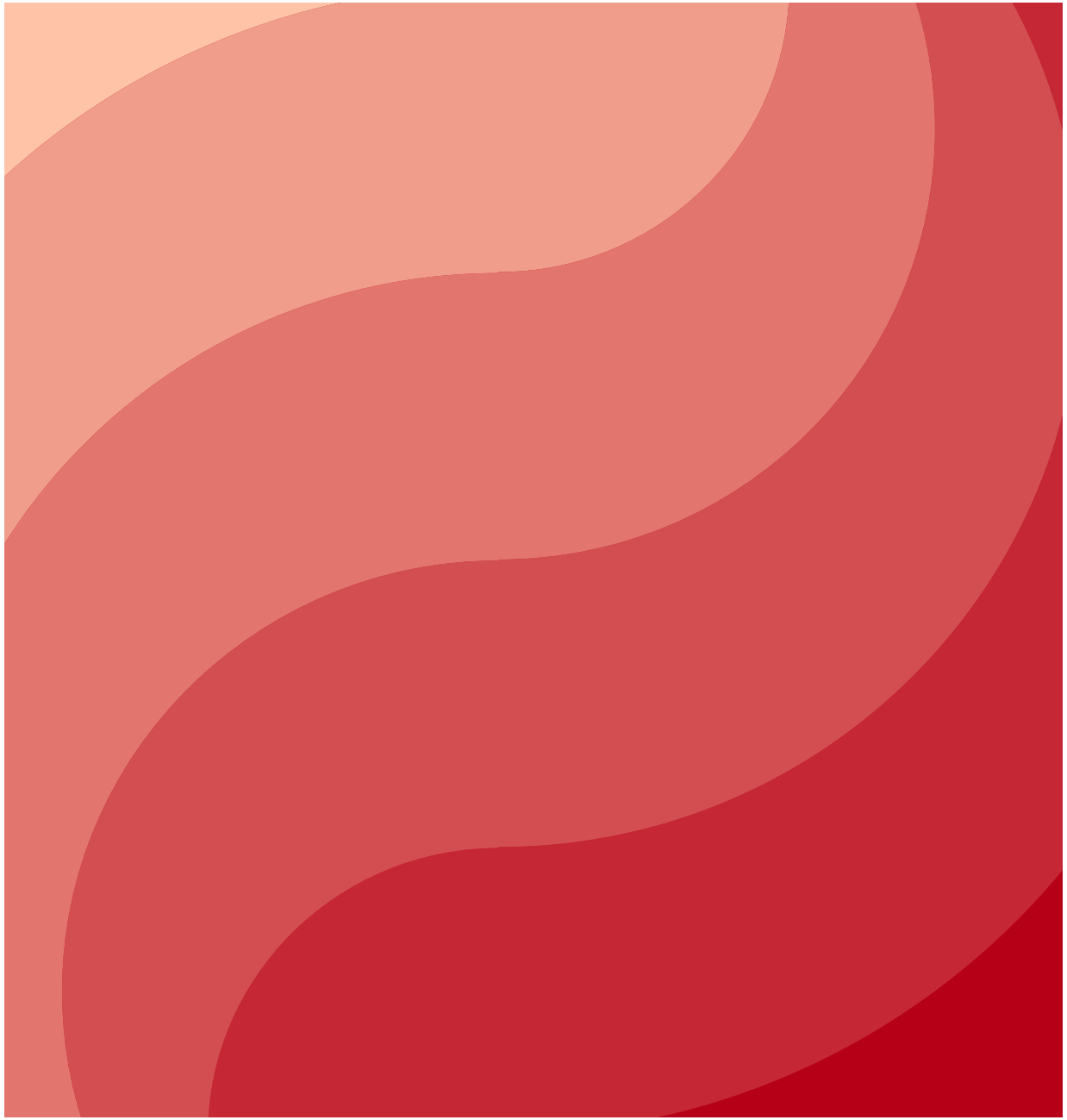
Example: RAFT

Optimizing an optical flow network



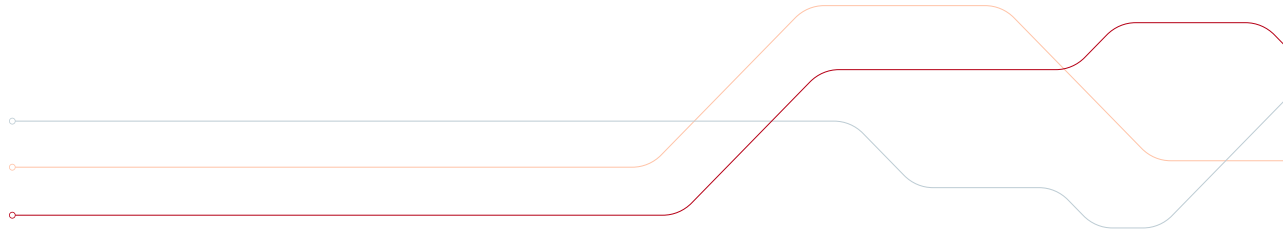
Agenda

- Introduction of OmniML
- Omnimizer: an MLOps platform for edge AI
- Case studies
- Live Demo



Live Demo

ResNet on CIFAR-10



ResNet20 on CIFAR-10

.. tip:: This tutorial is available as a Jupyter Notebook! It would take about 3 hours on Google Colab but can be run as fast as in 1 hour if you use a better GPU. You can expect slightly different accuracies than reported below depending on the system you run this notebook in.

Open in Colab

In this tutorial, we will use the Omnimizer to make the ResNet model faster for our target device constraints without sacrificing much accuracy!

By the end of this tutorial, you will:

- Understand how to use Omnimizer to construct and train an OmniNet from a given model.
- Profile an OmniNet to get the range of possible values for search constraints or check if it can satisfy your constraints.
- Search for the best performing subnet from the OmniNet for your target device constraints.
- Save and restore your searched model for downstream tasks like fine-tuning and inference.

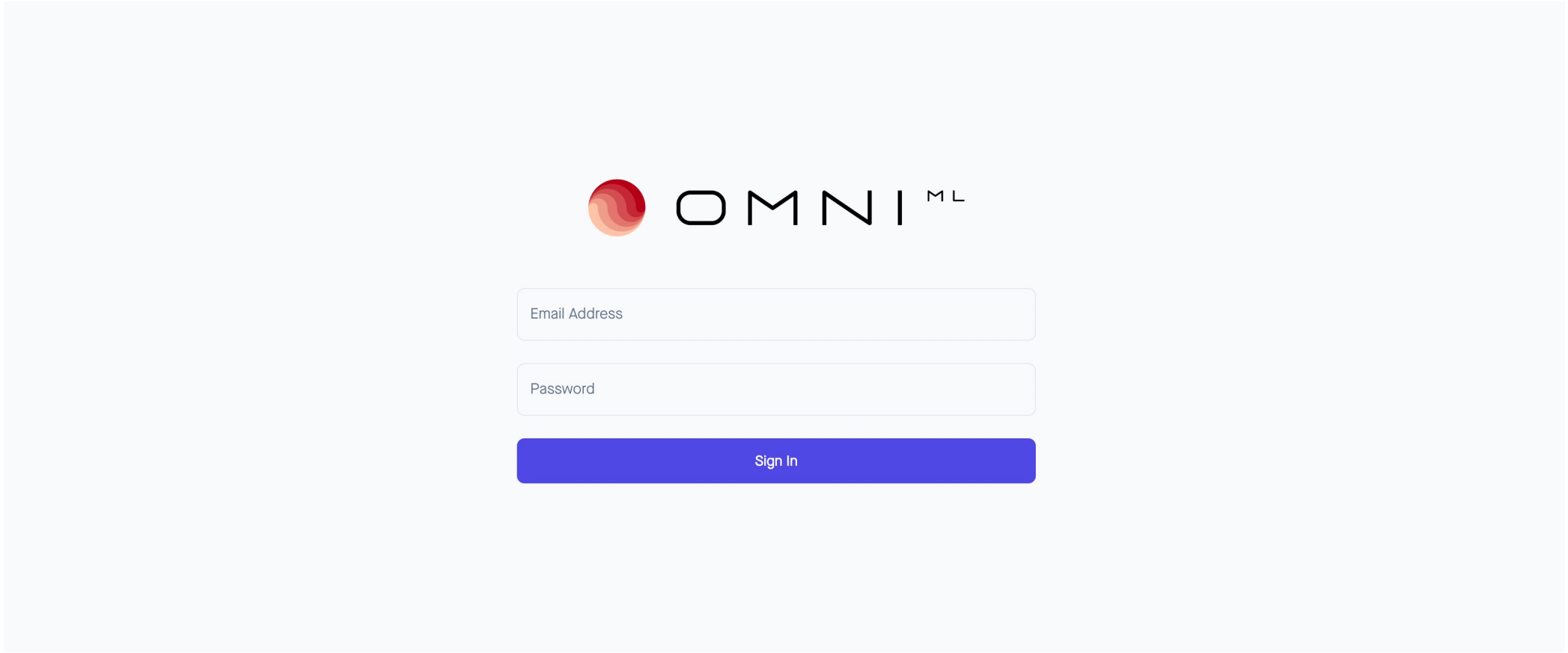
All of this with just a few lines of code! Yes, it's that simple!!

Let's first install `omnimizer` following the Installation. You would be asked to enter your username and password to install.

```
[ ]  ! pip install --extra-index-url https://pypi.omniml.ai/simple/ omnimizer
```

Live Demo

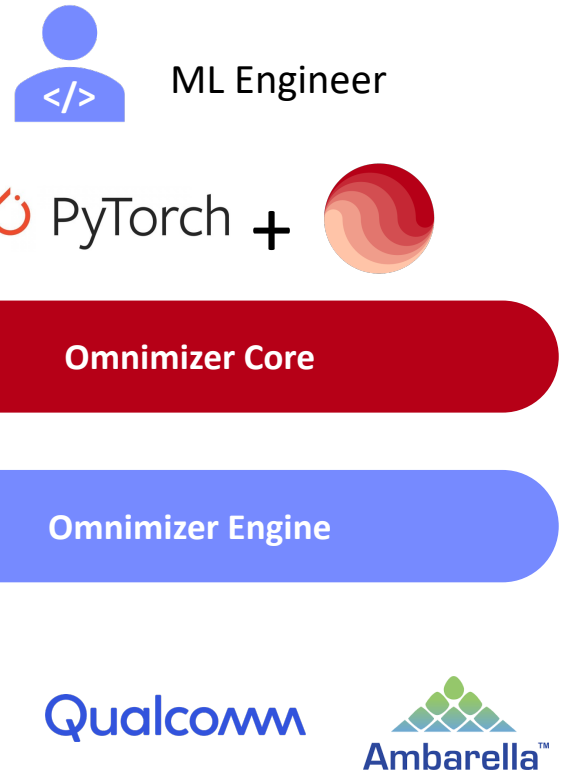
OmniML Portal



Conclusions

Re-envisioning how ML is being deployed on the edge

- **Omnimizer adapts the neural nets for the hardware**
- **Flexible, scalable, automated**
- **Optimize & deploy across ADAS, IoT, Robotics, Mobile**
- **Unleash the full power of existing hardware**



Get in touch for early access to Omnimizer and more…

We want to hear from you!

Contact us at: contact@omniml.ai
Sign-up: https://omniml.ai/sign-up

Sign up with your MIT email
Mention your 6.S965 project

We are actively hiring:
https://jobs.lever.co/OmniML.ai/