MODEL DEPLOYMENT FOR EDGE AI

—— CVPR 2024 Tutorial ——

The IEEE/CVF Conference on Computer Vision and Pattern Recognition 2024

Seattle, WA, USA

TUTORIAL AGENDA

1 Model Compression

2 Understanding Key Metrics

3 Model Compression Techniques

4 Case Studies

5 Summary

MODEL DEPLOYMENT FOR EDGE AI

Introduction

Model deployment is a critical phase in Edge AI, where optimized AI models are strategically placed into operation on edge devices. Effective model deployment enables smarter, localized decision-making, minimizes latency, and leverages the full potential of Edge AI.

Objective 01

Understanding model compression techniques

Objective 02

Comprehending the deployment strategies

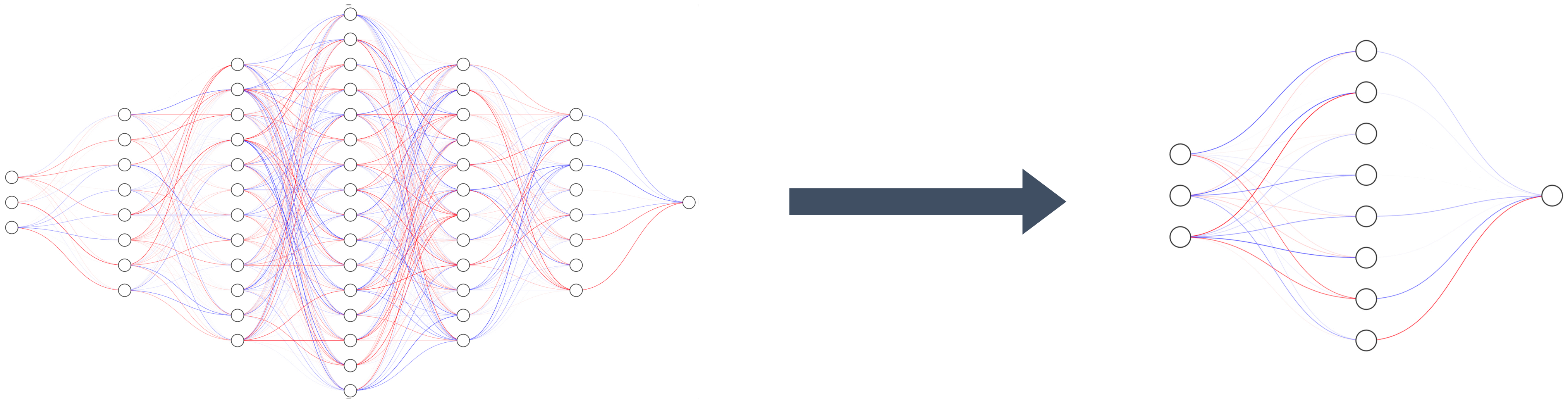
Objective 03

Presenting demos in production and in research

MODEL COMPRESSION

The What & The Why

> "The Art and Science of making an AI model smaller and lighter, without substantially sacrificing its accuracy."



Smaller Model Size

Faster Inference

Reduced Energy Consumption

Deployment on Constrained Devices

UNDERSTANDING KEY METRICS

Model Deployment

Size Reduction

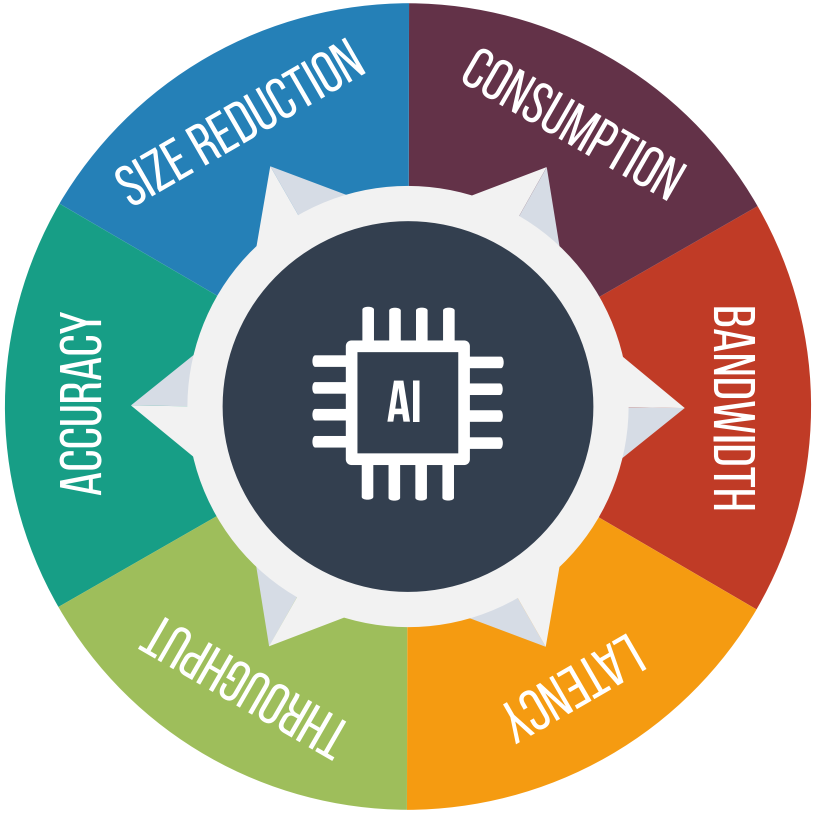
Shrinking model dimensions to fit edge device constraints.

Accuracy

Measuring the model's correctness in predictions against real-world data.

Throughput

Assessing the number of inferences a model can process per unit time.



Power & Energy Consumption

Evaluating the model's energy use and battery impact during operation.

Memory Bandwidth

Gauging the rate at which data is transferred to and from the device's memory.

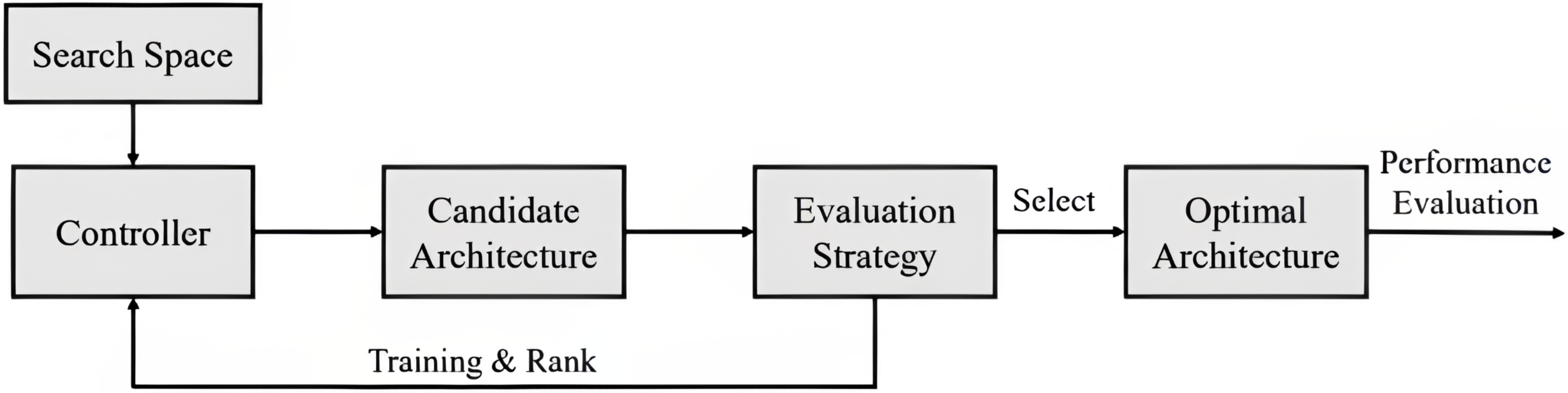
Latency

Timing the delay from input to decision output by the model.

PRIMARY TECHNIQUES FOR MODEL COMPRESSION IN EDGE AI

NEURAL ARCHITECTURE SEARCH

A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions



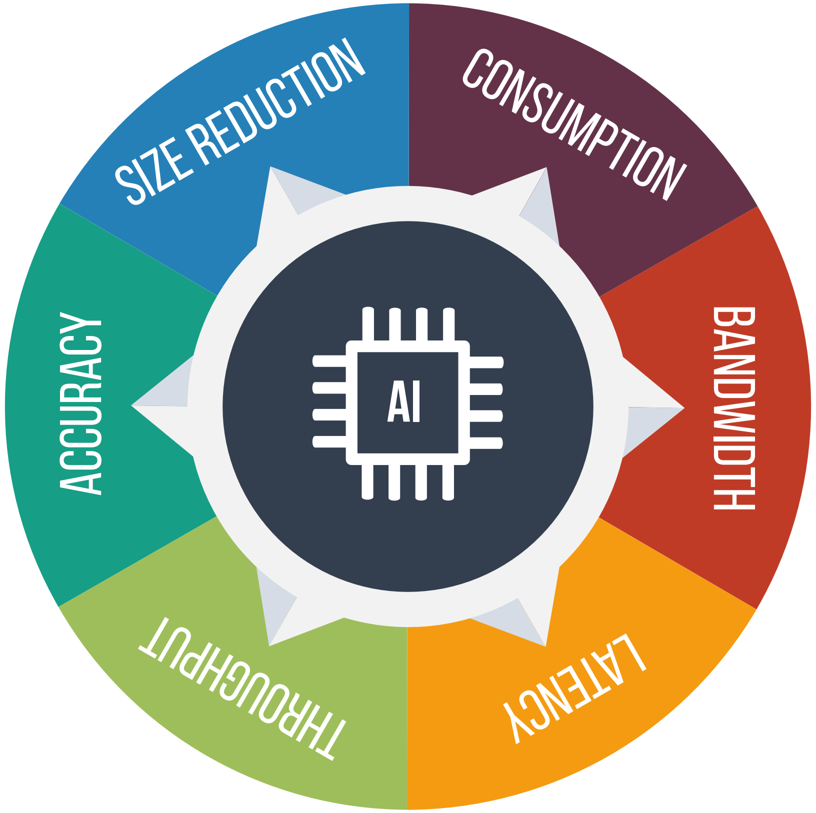
NEURAL ARCHITECTURE SEARCH

Key Metrics

Size Reduction
Up to 10% – 50%

Accuracy
Equal to or often better than handcrafted models.

Throughput
1.5x – 5x higher compared to non-optimized models.



Power & Energy Consumption
2x – 5x lower consumption.

Memory Bandwidth
10% – 50% reduction.

Latency
2x – 5x reduction compared to non-optimized models.

EARLY EXITS

Overview

The *Early Exits* technique in model optimization involves adding intermediate outputs to a deep learning model.

HOW DOES EARLY EXITS TECHNIQUE WORK?

PREDICTIONS

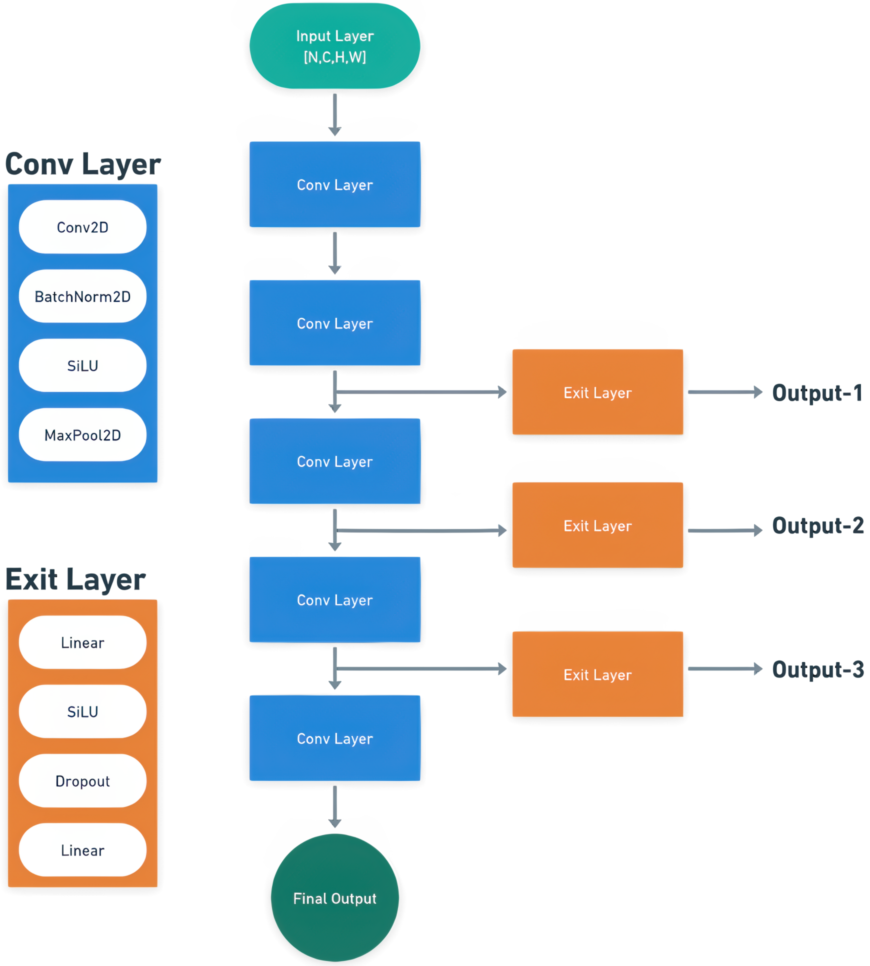
Early exits allow intermediate layers in a deep neural network (DNN) to produce predictions.

EXITS

Uses a confidence threshold to decide when to exit early.

PERFORMANCE

They help reduce the computational costs by exiting the inference once a confident prediction is made.



EARLY EXITS
Overview

The *Early Exits* technique in model optimization involves adding intermediate outputs to a deep learning model.

WHAT ARE THE EARLY EXITS TECHNIQUE ADVANTAGES?

REDUCED LATENCY

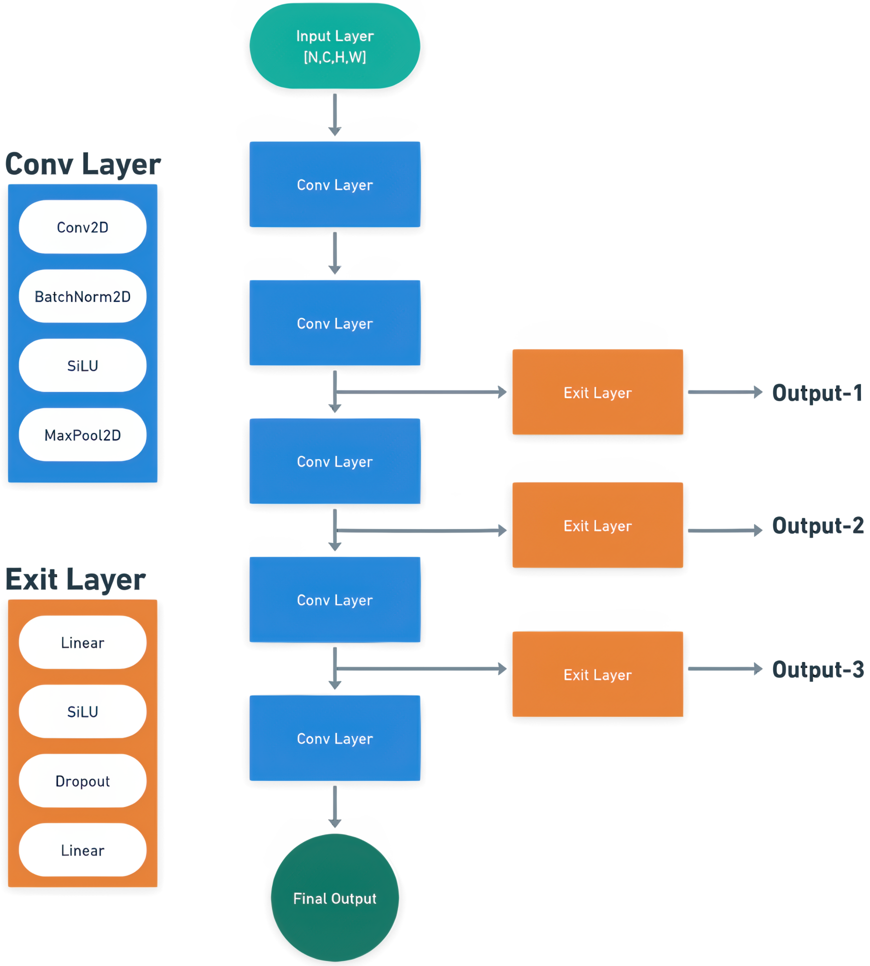
Faster inference as not all layers need to be processed.

LOWER ENERGY CONSUMPTION

Less computation means lower power usage.

ADAPTIVE COMPUTATION

Flexibility to balance accuracy and efficiency dynamically.



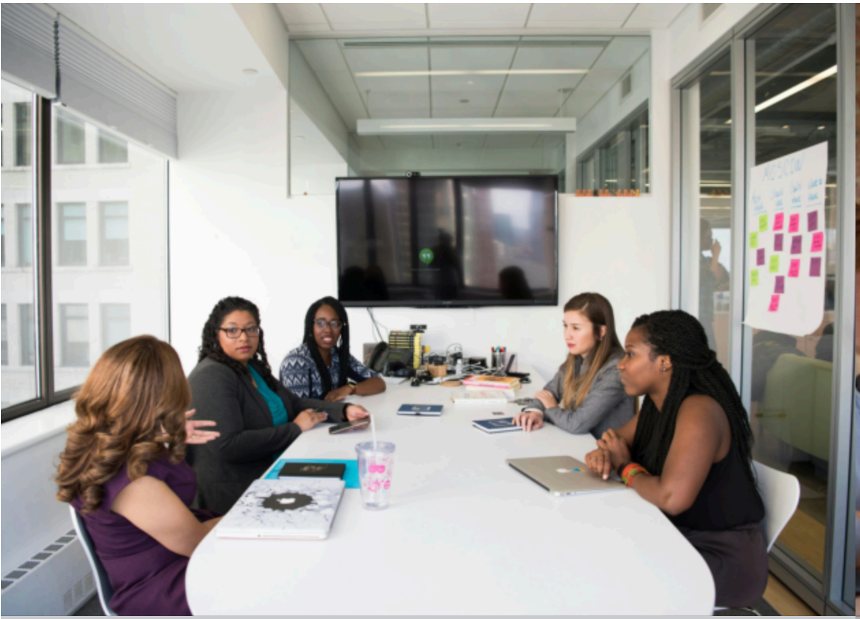
EARLY EXITS

Human Detector

```
Predicted Label: human, Confidence: 1.000, Exit: 3
```

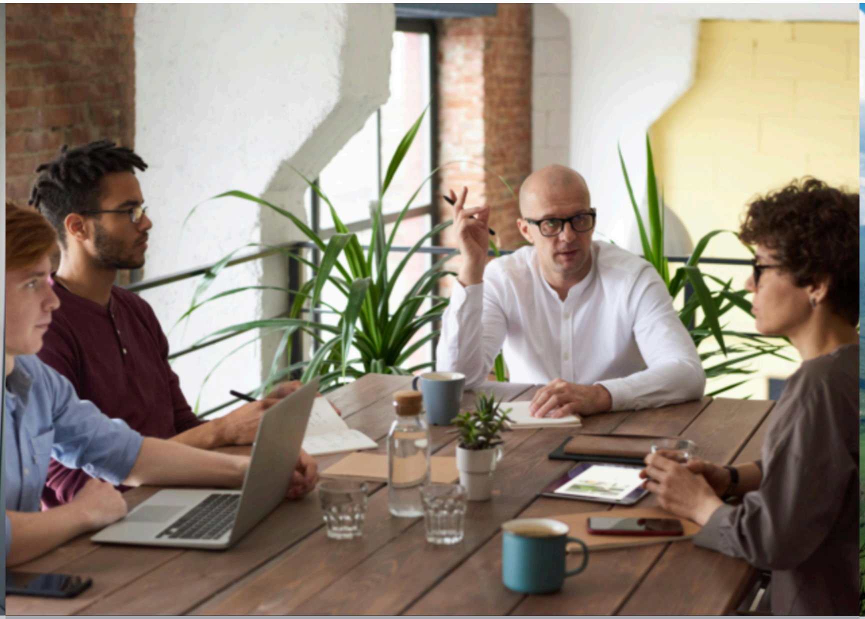
EARLY EXITS

Results



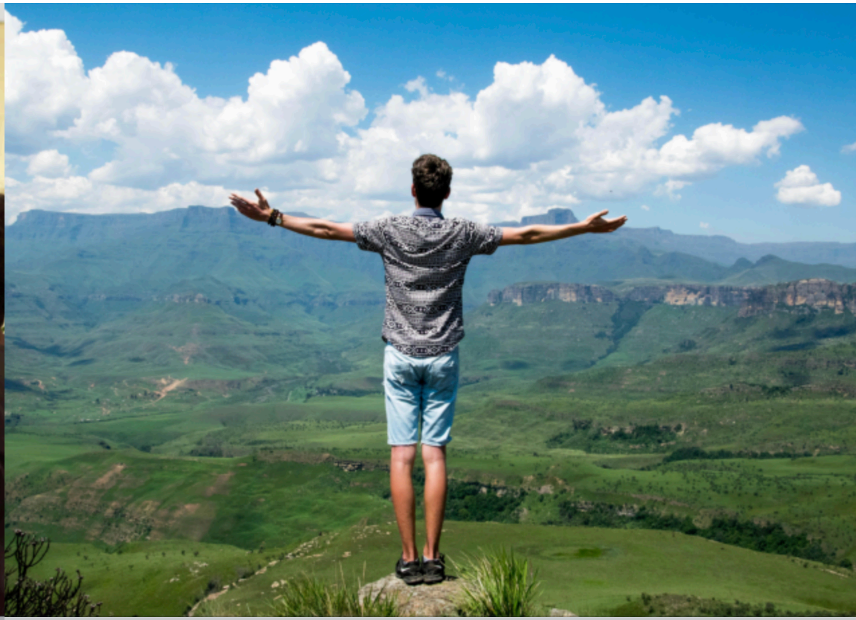
PREDICTED LABEL: HUMAN

CONFIDENCE: 0.920, EXIT: 2



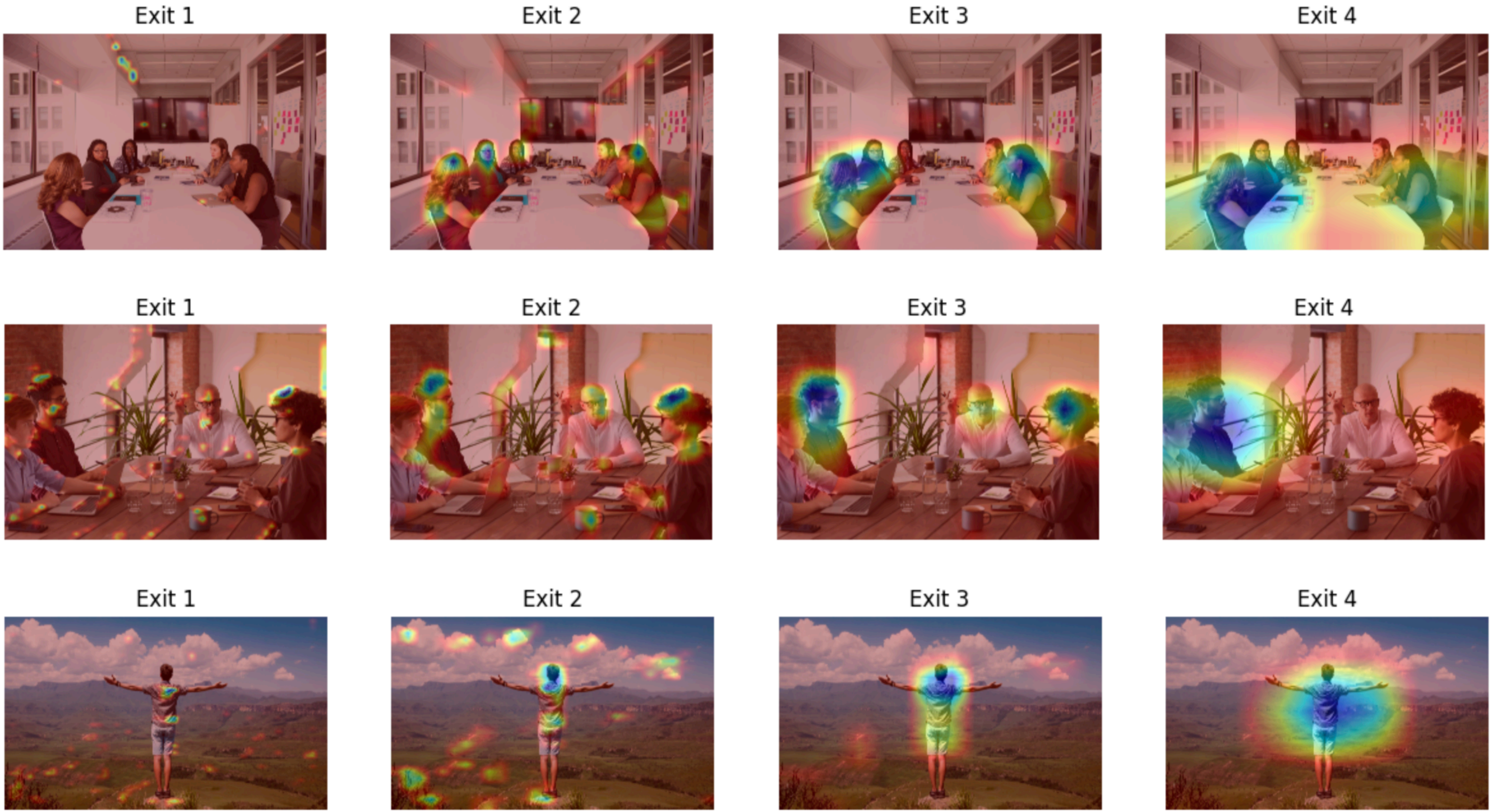
PREDICTED LABEL: HUMAN

CONFIDENCE: 0.937, EXIT: 2



PREDICTED LABEL: HUMAN

CONFIDENCE: 0.959, EXIT: 4


Exit 1
Exit 2
Exit 3
Exit 4
Exit 1
Exit 2
Exit 3
Exit 4
Exit 1
Exit 2
Exit 3
Exit 4

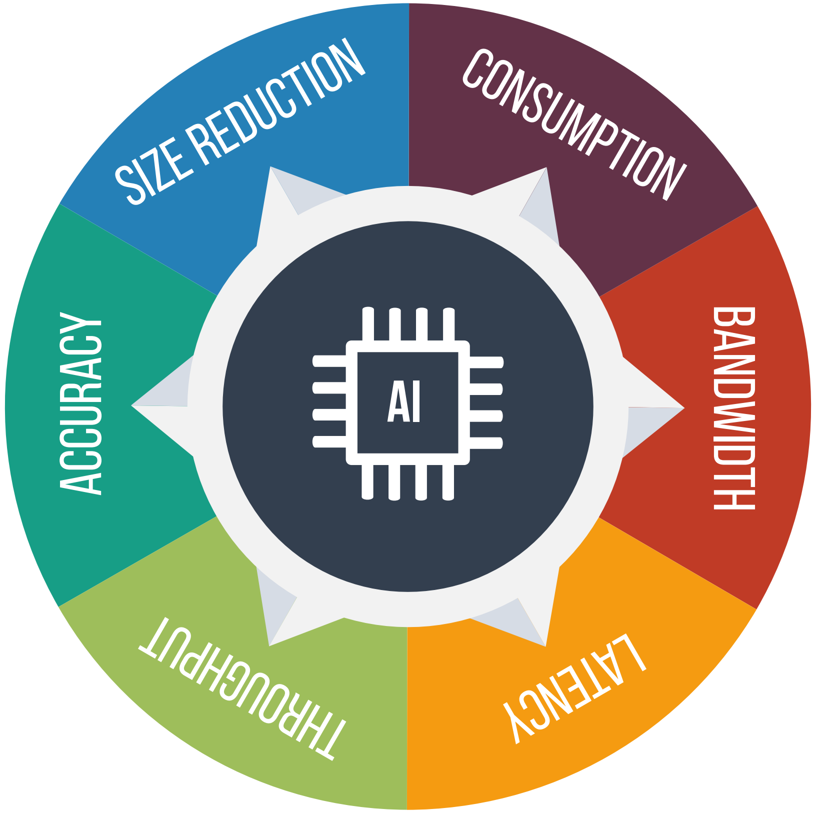
EARLY EXITS

Key Metrics

Size Reduction
Up to 20% – 40%

Accuracy
Equal to or often better than handcrafted models.

Throughput
2x – 4x higher compared to non-optimized models.



Power & Energy Consumption
2x – 4x lower consumption.

Memory Bandwidth
20% – 40% reduction.

Latency
3x – 5x reduction compared to non-optimized models.

MIXTURE OF DEPTHS

Overview

The Mixture of Depths combines predictions from different depths of a DL model to improve accuracy and robustness.

Dynamic Compute Allocation

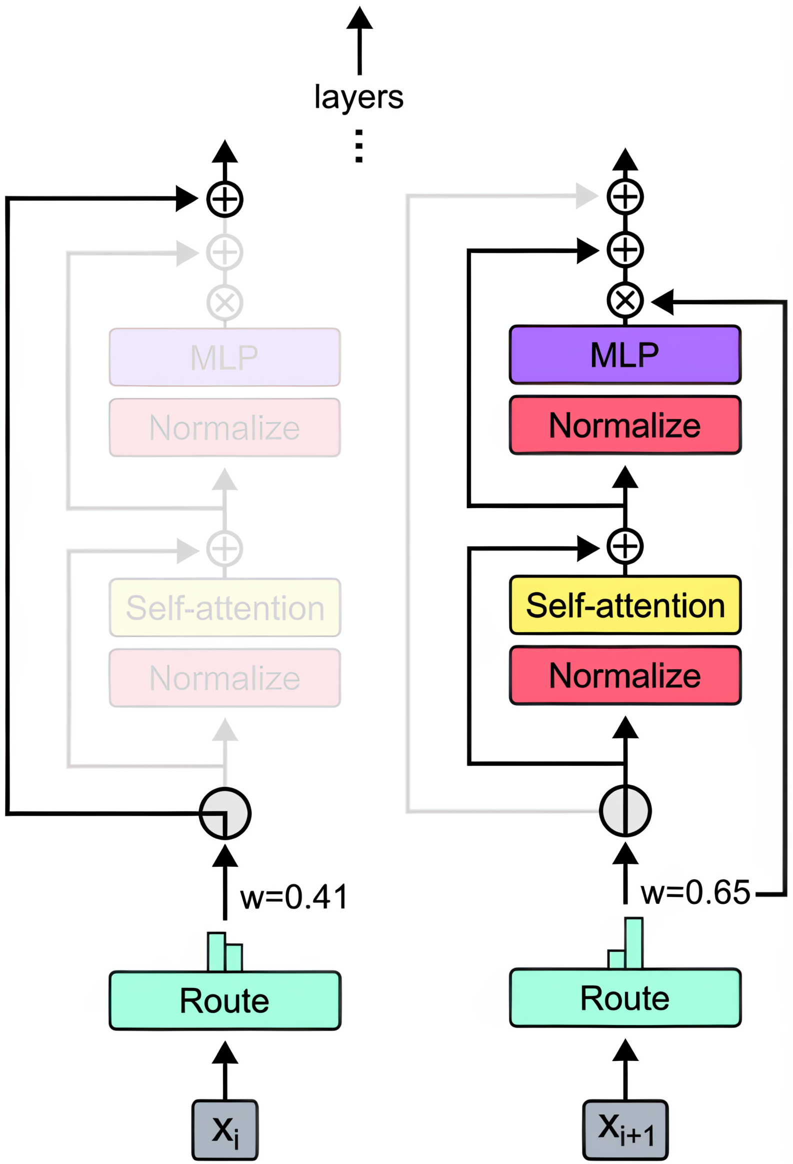
Selectively processes tokens through different layers based on importance.

Skips unnecessary computations to reduce FLOPs and improve efficiency.

Routing Mechanism

Uses a router to decide which tokens pass through expensive layers.

Bypasses less critical tokens via residual connections.



MIXTURE OF DEPTHS

Overview

The Mixture of Depths combines predictions from different depths of a DL model to improve accuracy and robustness.

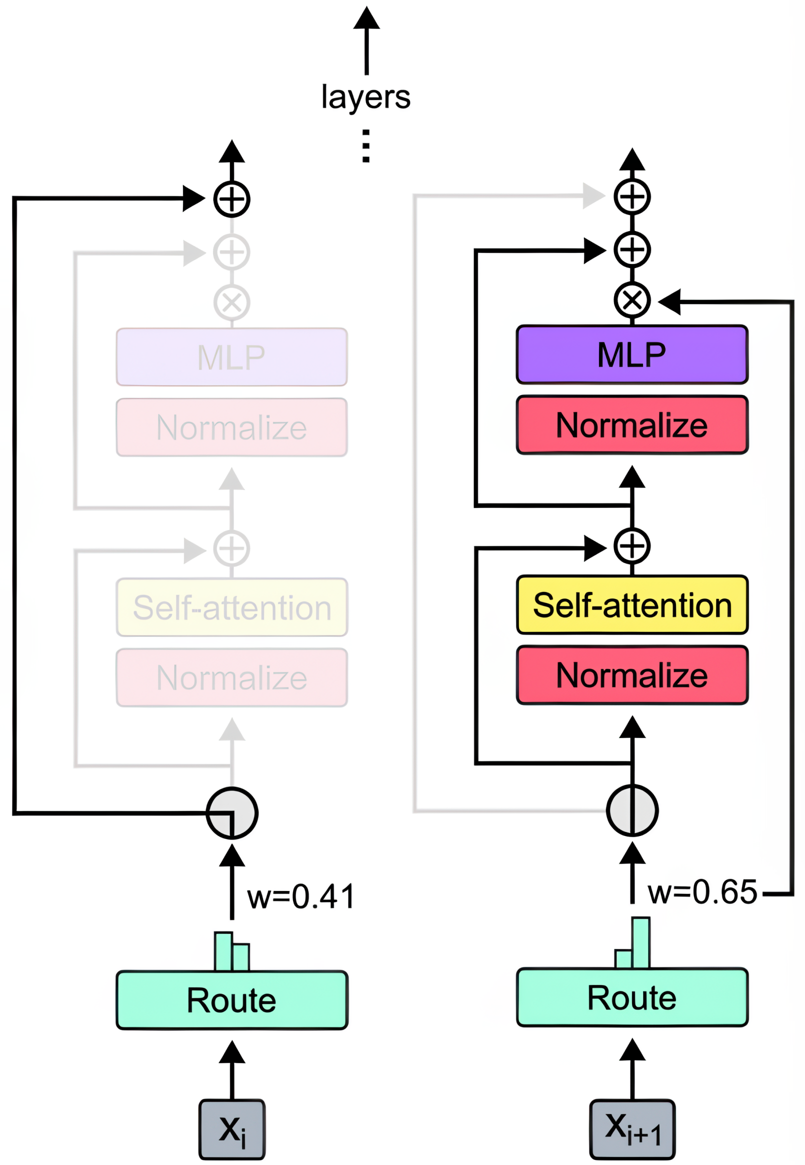
Compute Savings

Significant reduction in computing by routing only essential tokens through costly operations.

Maintains performance while lowering the computational load.

Static Computation Graph

Ensures predictable compute expenditure with dynamic token participation.



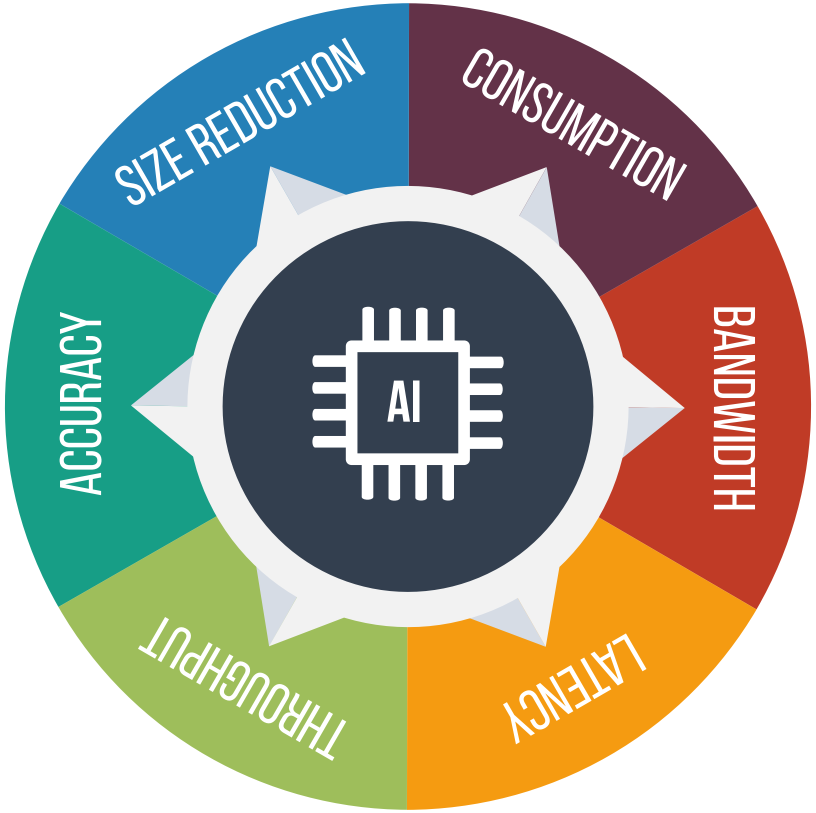
MIXTURE OF DEPTHS

Key Metrics

Size Reduction
Up to 40%

Accuracy
Equal to or often better than handcrafted models.

Throughput
2x – 4x higher compared to non-optimized models.



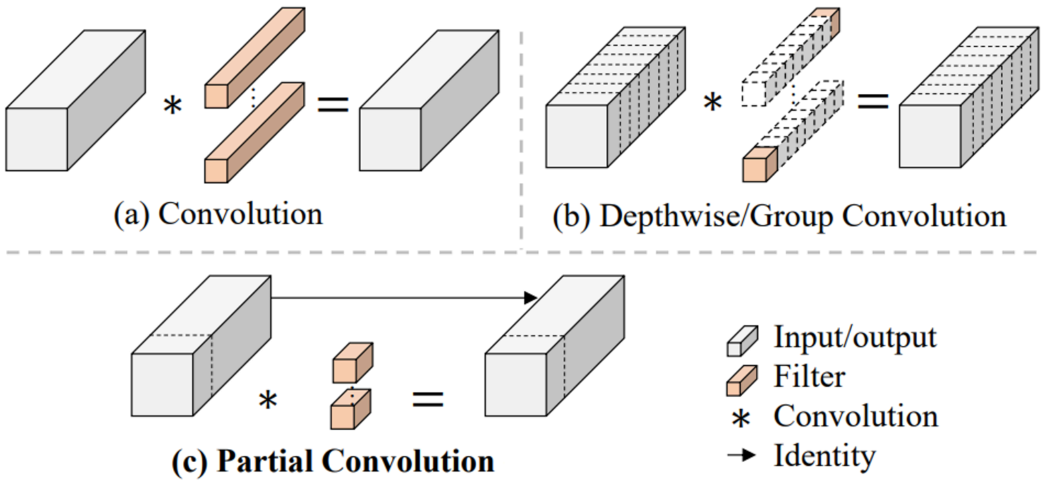
Power & Energy Consumption
2x – 5x lower consumption.

Memory Bandwidth
25% – 50% reduction.

Latency
3x – 6x reduction compared to non-optimized models.

HARDWARE AWARE DESIGN

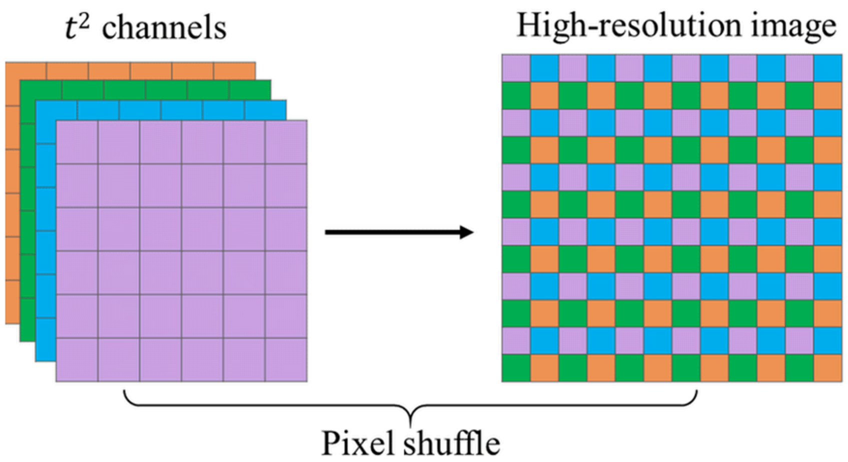
Real-Time Single Image and Video Super-Resolution using an Efficient Sub-Pixel CNN



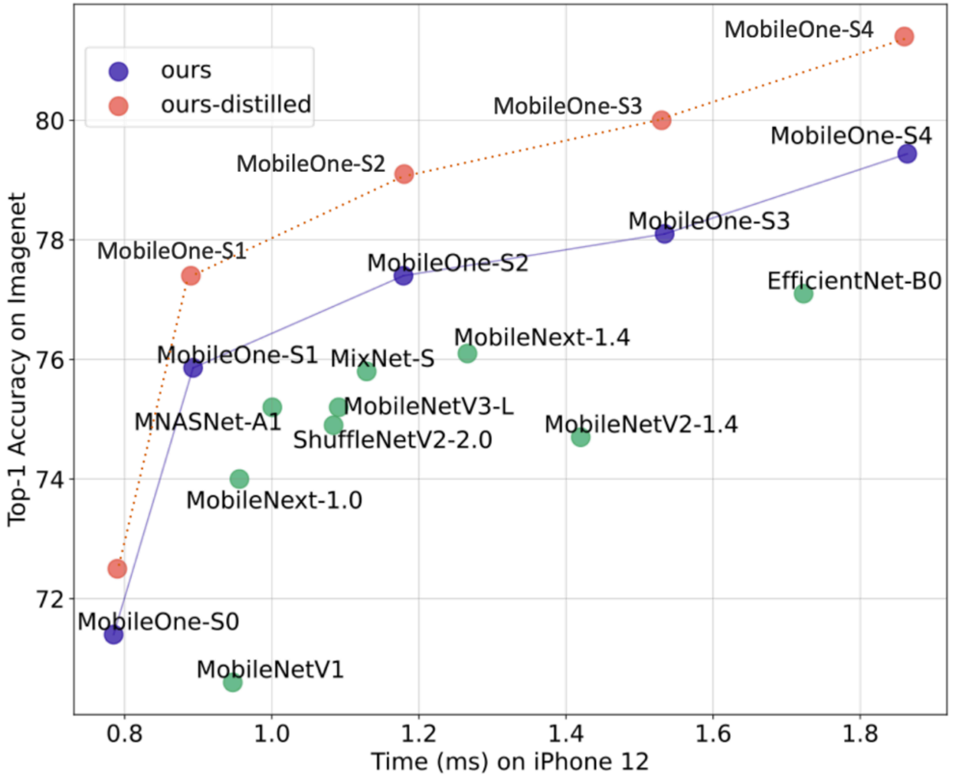
(a) Convolution

(b) Depthwise/Group Convolution

(c) Partial Convolution



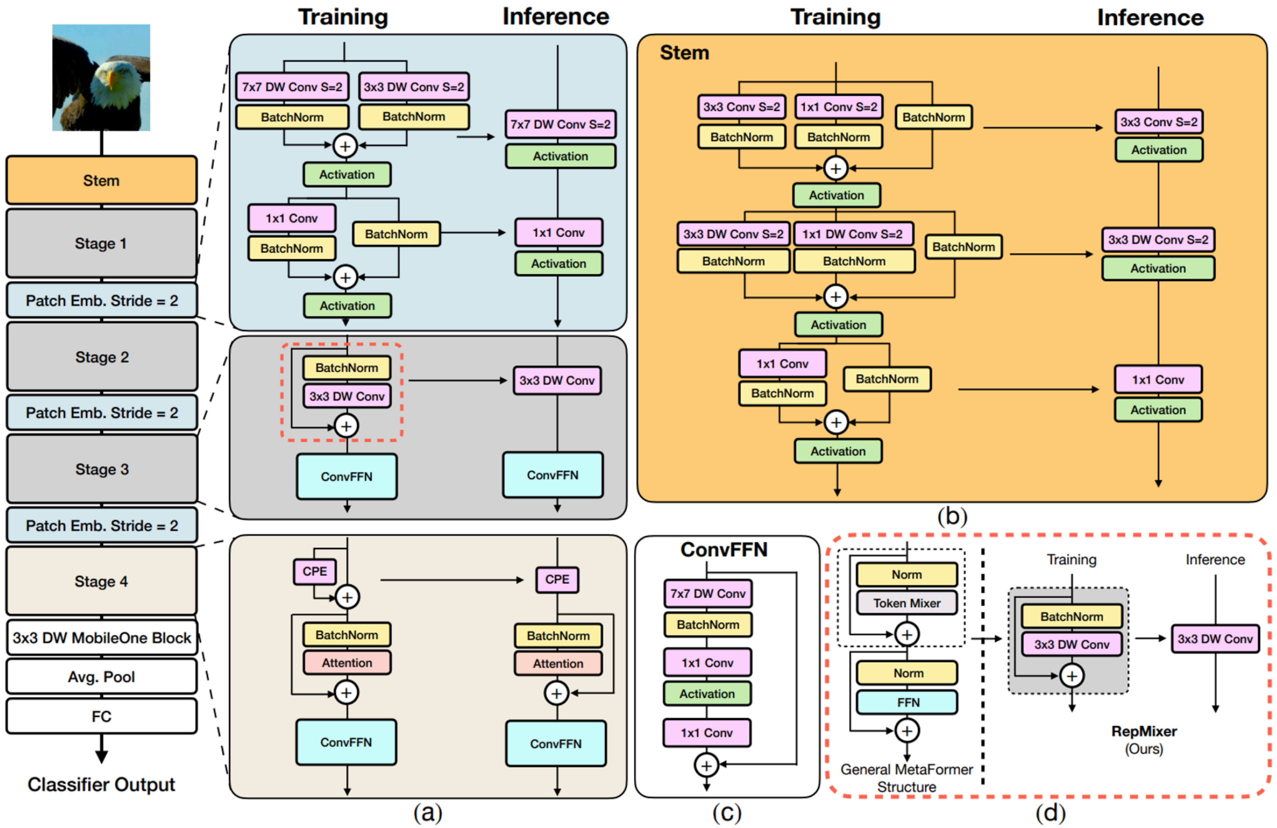
Pixel shuffle

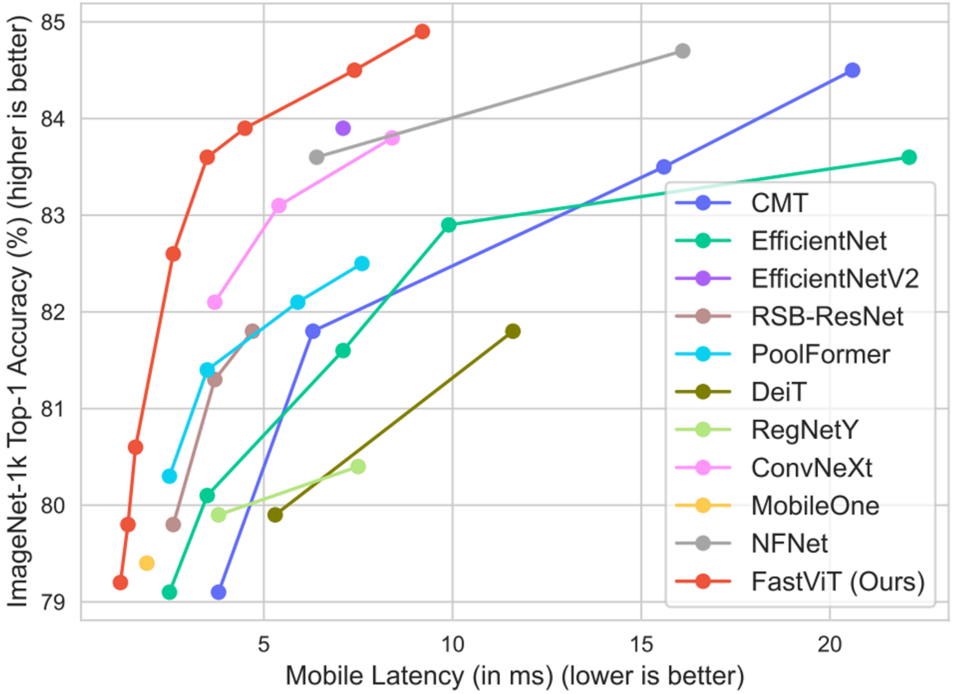


MobileOne: An Improved One millisecond Mobile Backbone

HARDWARE AWARE DESIGN

FastViT: A Fast Hybrid Vision Transformer using Structural Reparameterization





HARDWARE AWARE DESIGN

Key Metrics

Size Reduction

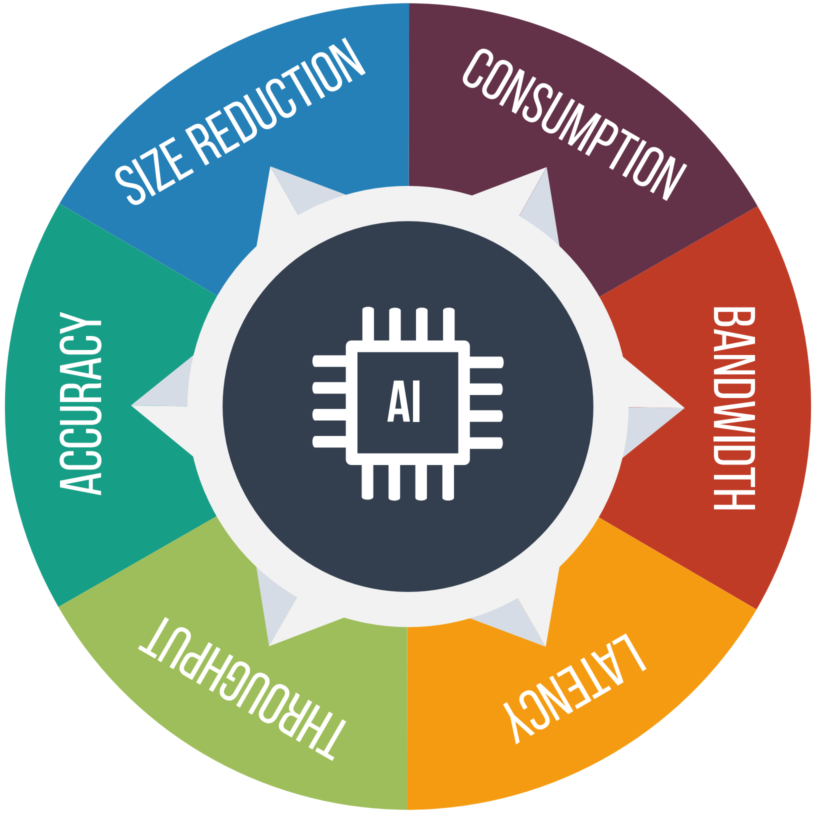
Up to 30% – 50% w.r.t teacher model.

Accuracy

Equal to or often better than handcrafted models.

Throughput

2x – 4x higher compared to non-optimized models.



Power & Energy Consumption

2x – 4x lower consumption.

Memory Bandwidth

20% – 40% reduction, depending on the student model design.

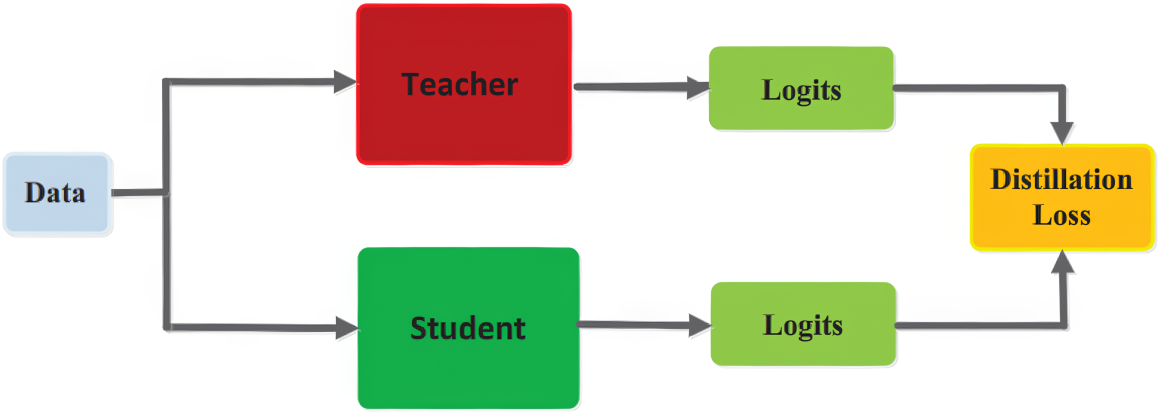
Latency

3x – 5x reduction compared to non-optimized models.

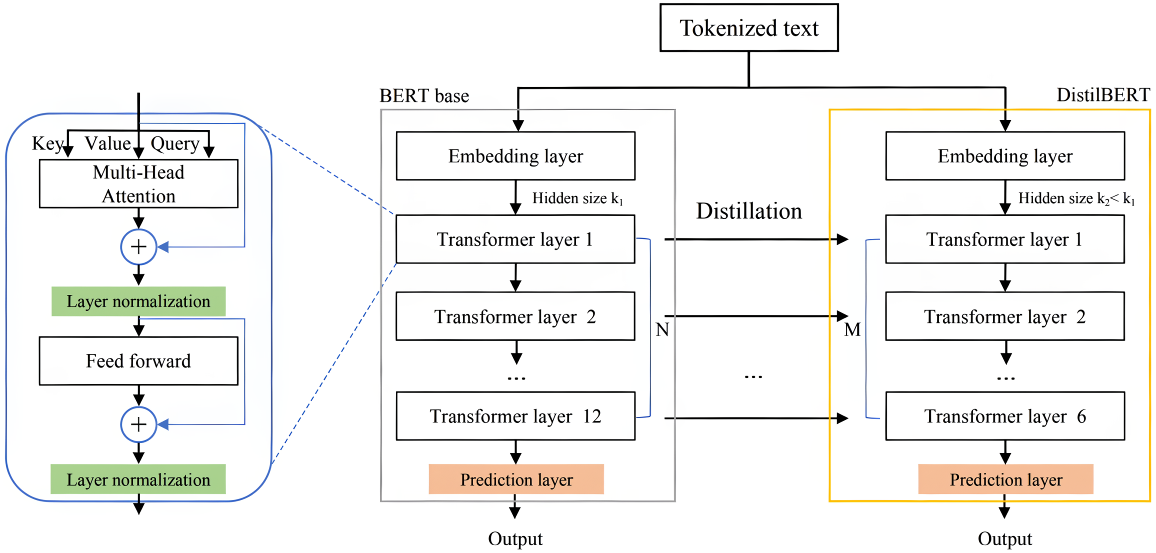
KNOWLEDGE DISTILLATION

Overview

> “A technique where a smaller model (student) is trained to reproduce the behavior of a larger model (teacher) or an ensemble of models, often leading to a compact model with comparable performance.”



Knowledge Distillation: A Survey



Improving Crisis Events Detection Using DistilBERT with Hunger Games Search Algorithm

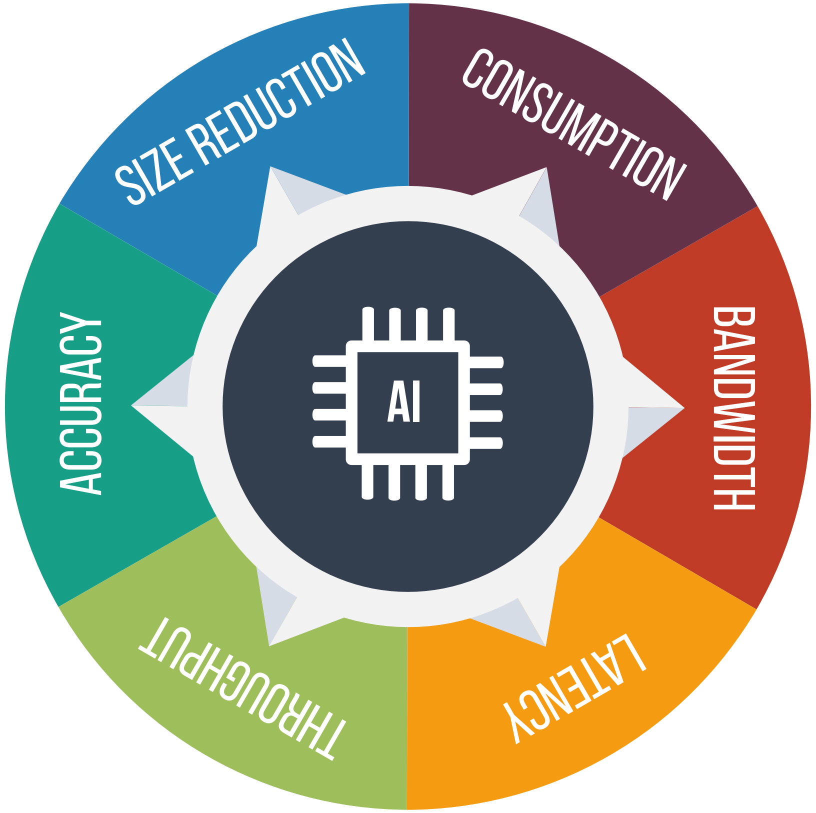
KNOWLEDGE DISTILLATION

Key Metrics

Size Reduction
Up to 10% – 50%
w.r.t teacher model.

Accuracy
1% - 5% drop,
w.r.t teacher model.

Throughput
2x – 10x higher
w.r.t teacher model.



Power & Energy Consumption
2x – 10x lower
consumption.

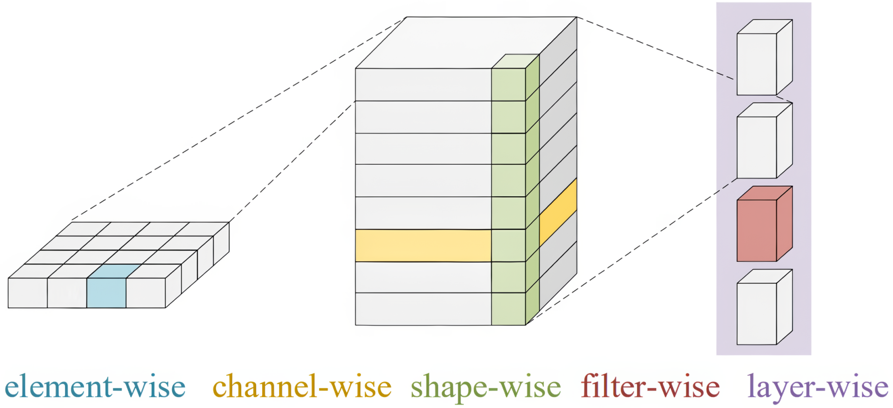
Memory Bandwidth
50% – 75% reduction,
depending on the
student model design.

Latency
2x – 10x reduction.

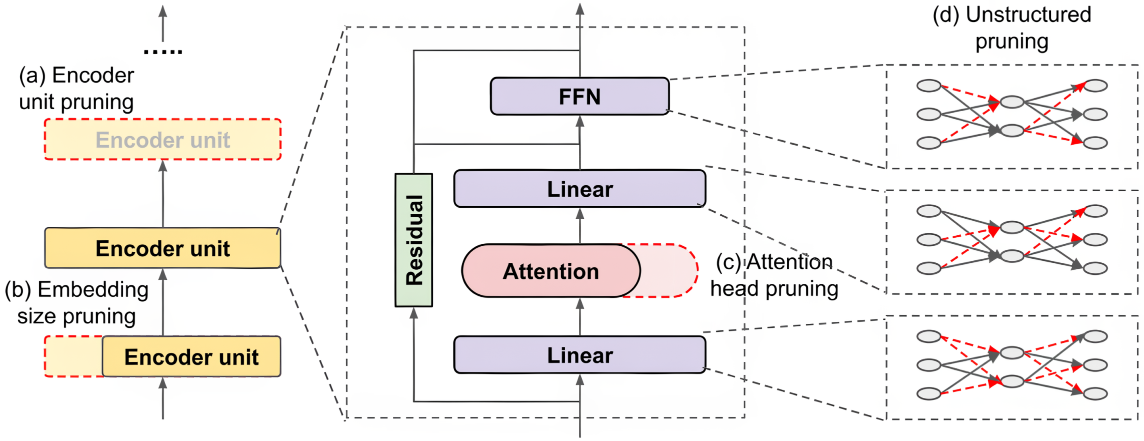
PRUNING

Overview

> “ The process of eliminating unnecessary parameters or connections in a neural network to streamline it, improving efficiency without significantly compromising performance. ”



Pruning and Quantization for Deep Neural Network Acceleration: A Survey



Compressing Large-Scale Transformer-Based Models: A Case Study on BERT

PRUNING

Types of Pruning

Pruning in Edge AI involves strategically removing *redundant* or *non-critical components* from AI models.

THESE ARE THE TYPES OF PRUNING WE WILL DISCUSS TODAY.

MAGNITUDE PRUNING

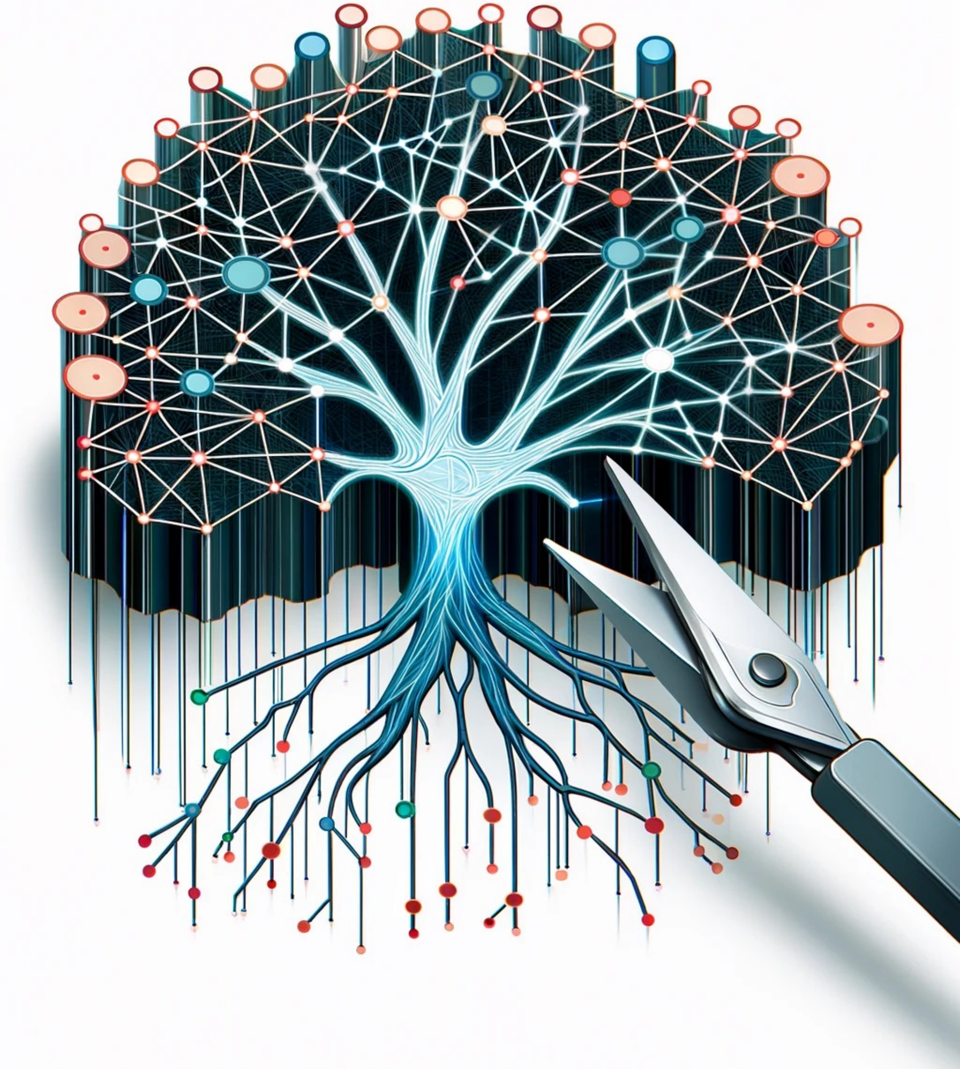
Targeting parameters based on their absolute values.

STRUCTURED VS. UNSTRUCTURED PRUNING

Do we remove entire channels or just sporadic connections?

LOCAL VS. GLOBAL PRUNING

Focusing on individual layers or the entire network?



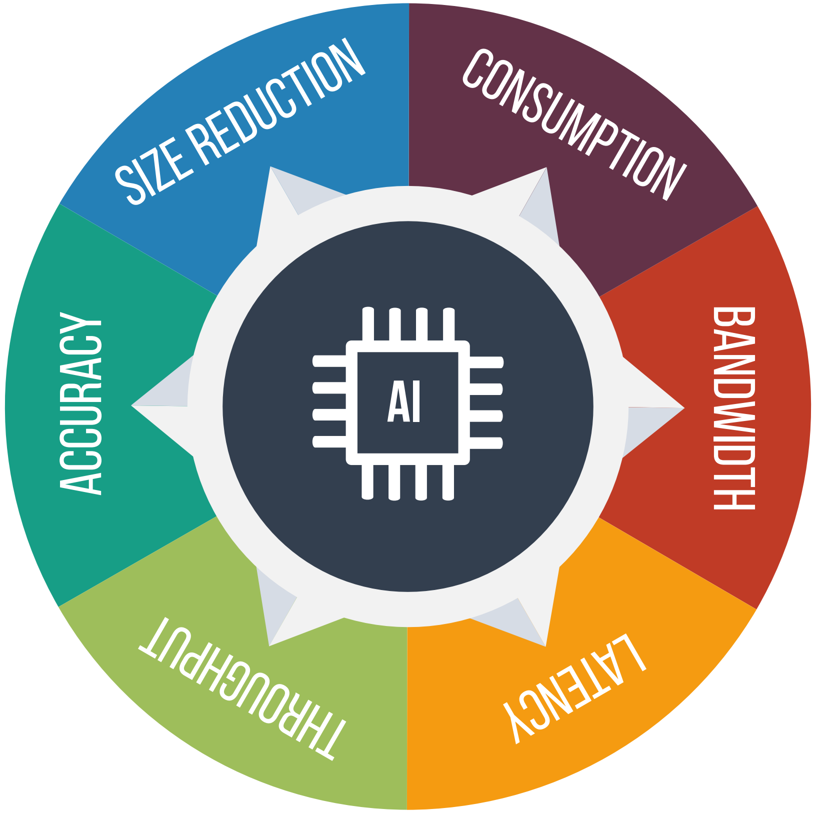
PRUNING

Key Metrics

Size Reduction
Up to 90x smaller models.

Accuracy
1% - 10% drop, recovered by model fine-tuning.

Throughput
10% – 50% higher.



Power & Energy Consumption
10% – 50% lower consumption.

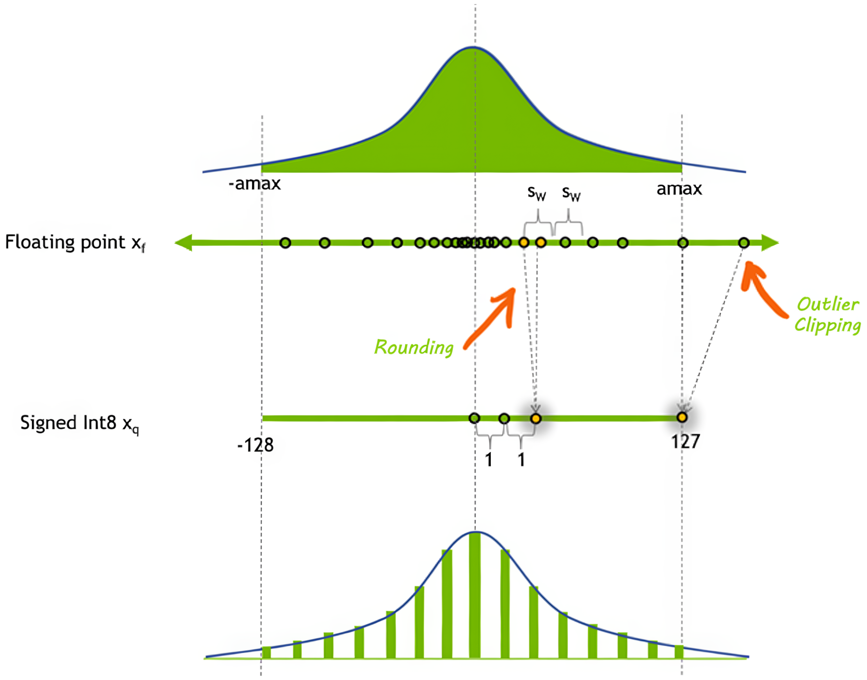
Memory Bandwidth
10% – 50% reduction.

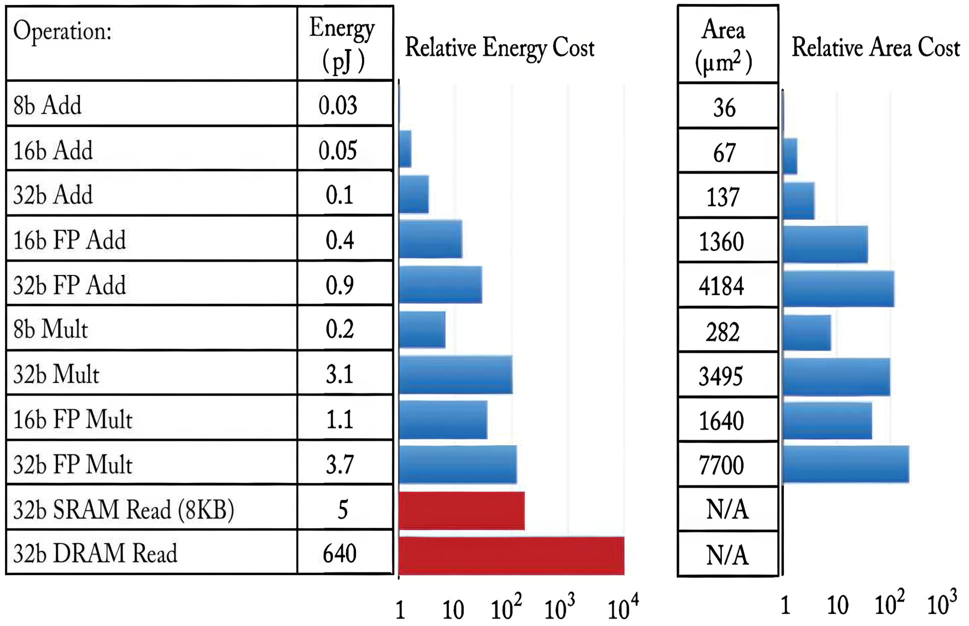
Latency
10% - 40% reduction.

QUANTIZATION

Overview

> “ The process of reducing the numerical precision of model parameters by mapping it from a large number of possible values to a reduced set of values. ”

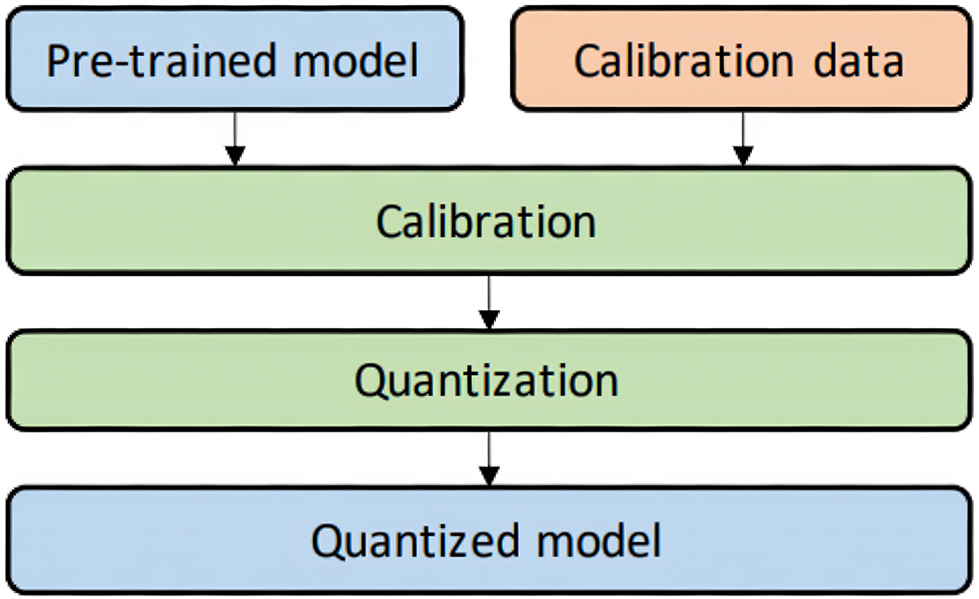




Operation:	Energy (pJ)	Area (μm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A

QUANTIZATION

A Survey of Quantization Methods for Efficient Neural Network Inference



Pre-trained model

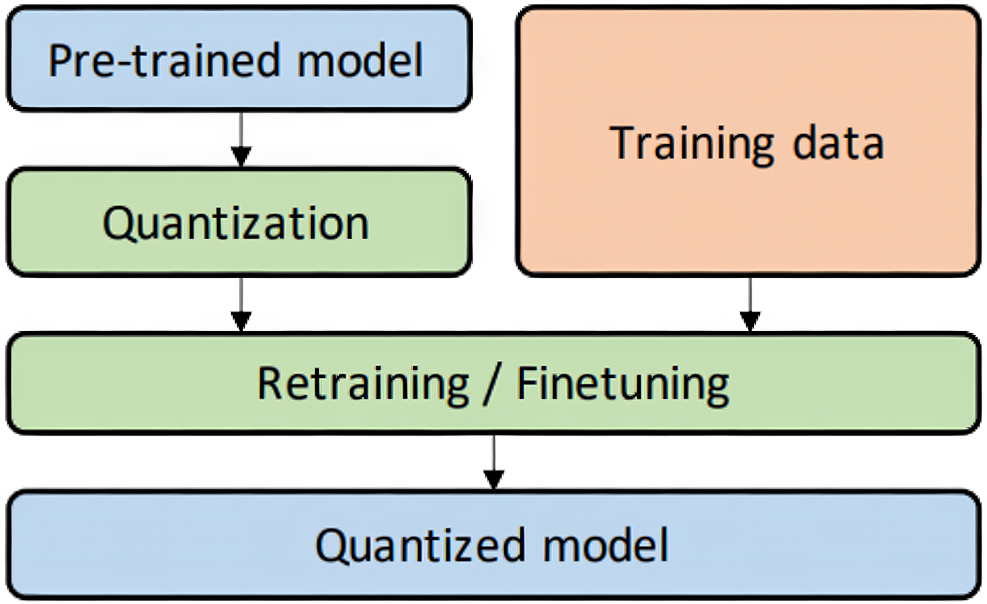
Calibration data

Calibration

Quantization

Quantized model

Post Training Quantization



Pre-trained model

Training data

Quantization

Retraining / Finetuning

Quantized model

Quantization Aware Training

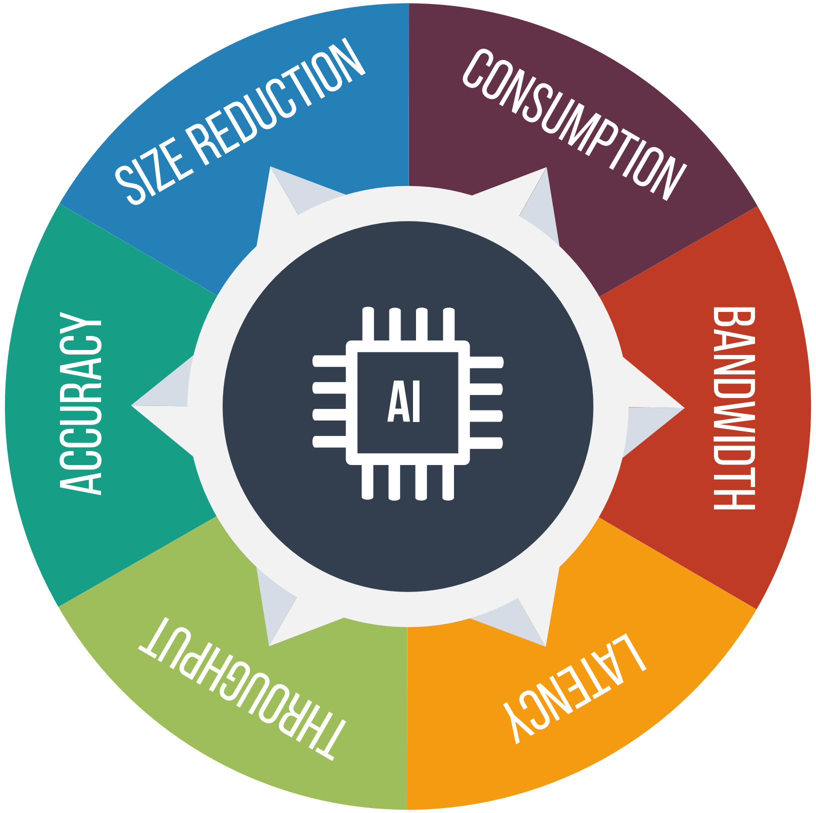
QUANTIZATION

Key Metrics

Size Reduction
Up to 50% – 75% w.r.t FP32 model.

Accuracy
1% - 5% drop, depending on bit-width and quantization technique.

Throughput
2x – 4x higher.

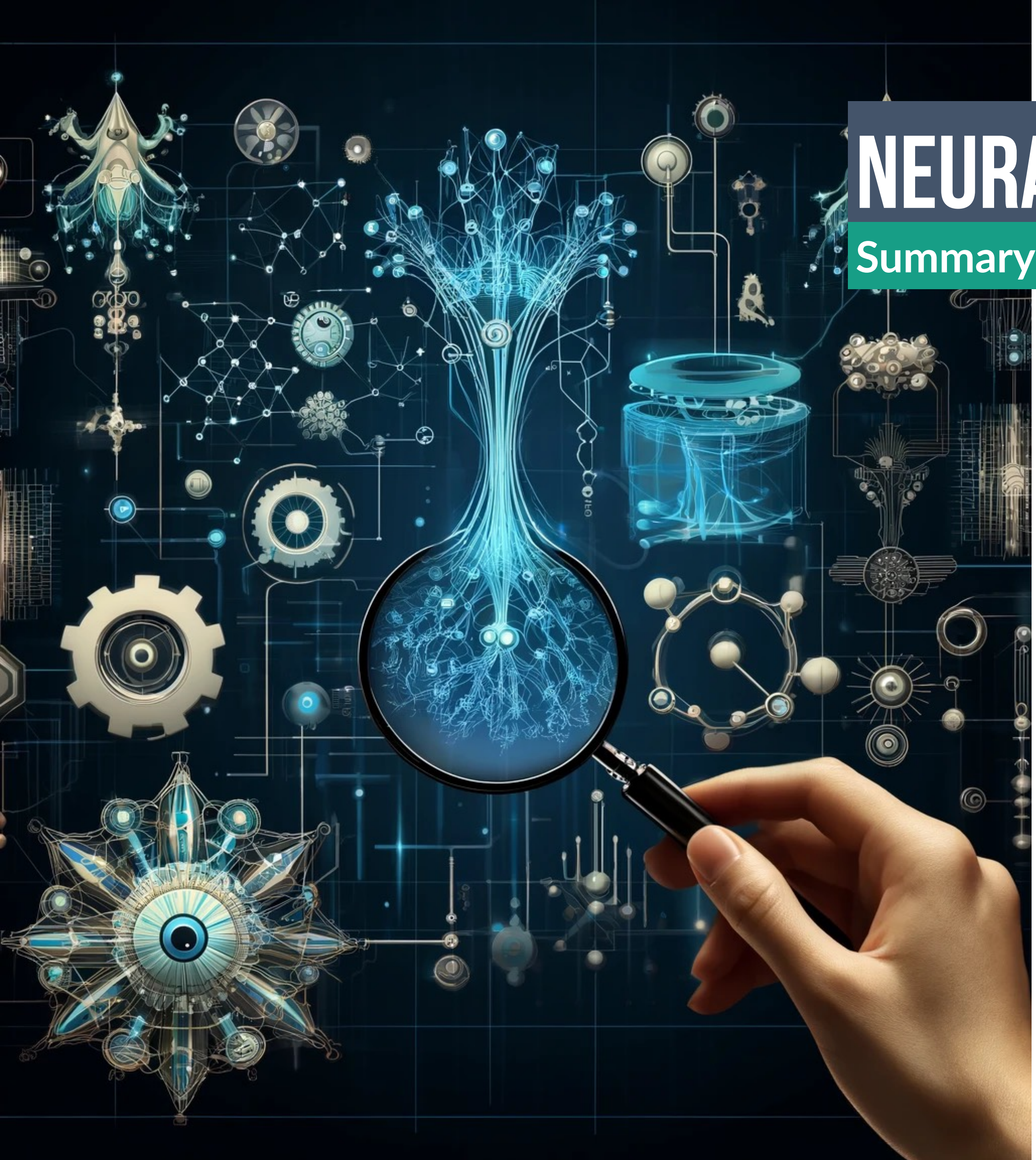


Power & Energy Consumption
2x – 3x lower consumption.

Memory Bandwidth
50% – 75% reduction, depending on bit-width.

Latency
2x – 3x reduction.

SUMMARIES OF MODEL COMPRESSION TECHNIQUES



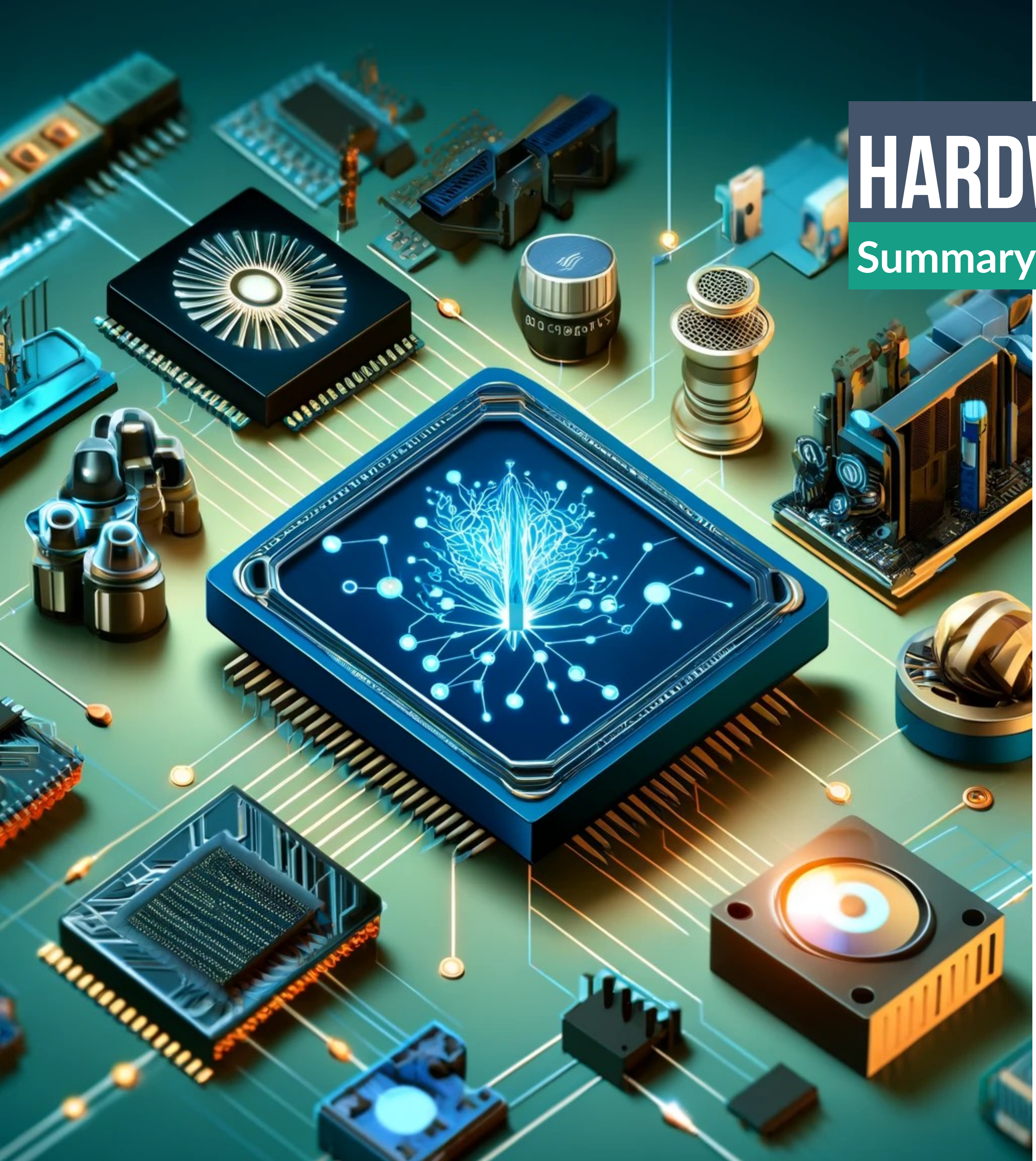
NEURAL ARCHITECTURE SEARCH

Summary

01 **Automation**
Automates the design of machine learning models.

02 **Optimization**
Searches for the most efficient architecture for a given task.

03 **Efficacy**
Useful when performance is crucial and manual tuning isn't yielding desired results.



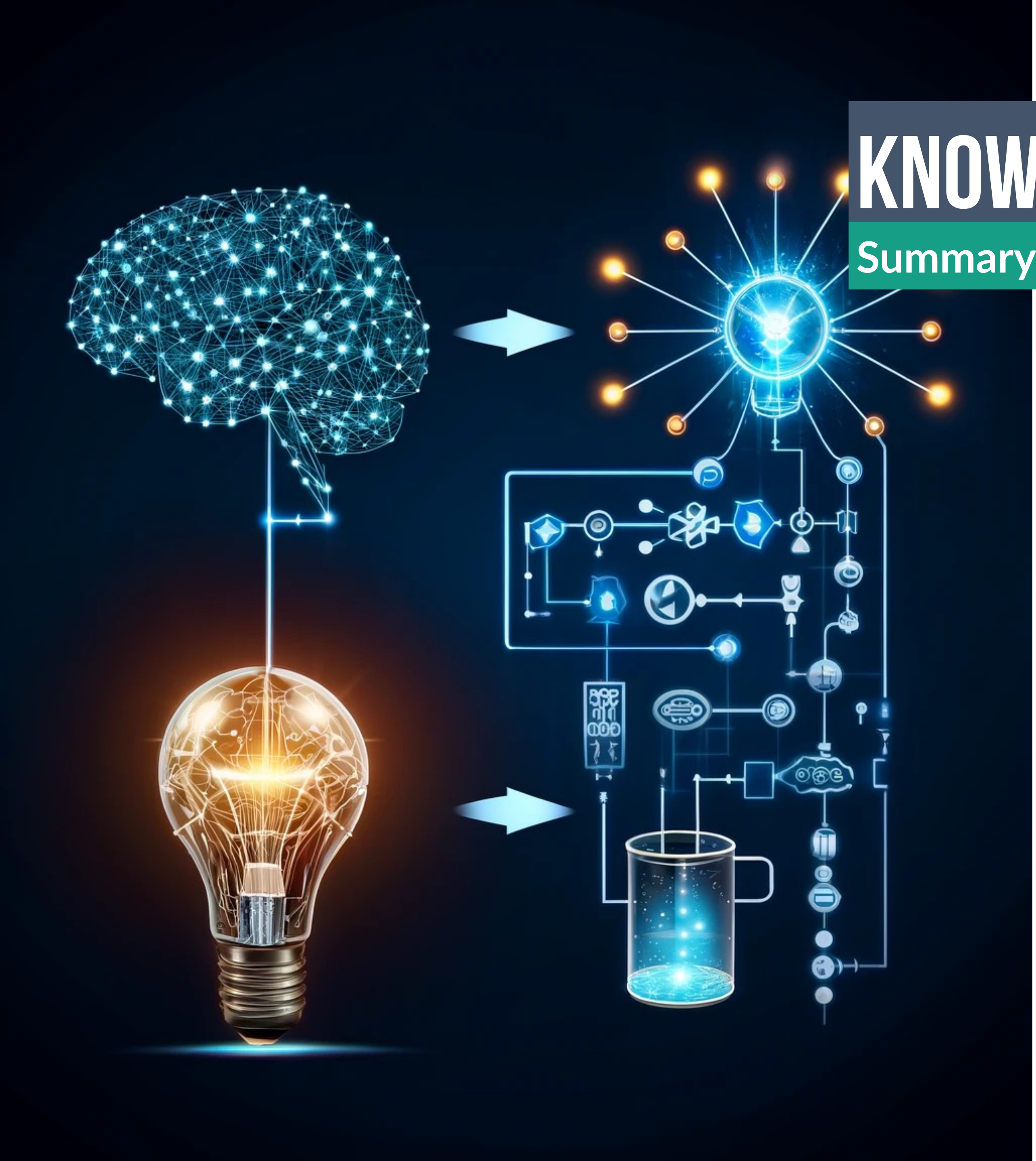
HARDWARE AWARE DESIGN

Summary

01 **Customization**
Tailor models to suit specific hardware constraints.

02 **Maximization**
Maximizes efficiency and performance for EdgeAI deployments.

03 **Adaptability**
Useful when deploying on specific edge devices with unique hardware constraints.



KNOWLEDGE DISTILLATION

Summary

01 **Transfer**

Train smaller student models with the knowledge of larger teacher models.

02 **Efficiency**

Achieve comparable accuracy with significantly reduced model size.

03 **Practicality**

The best when computational resources are limited, but access to pre-trained larger models is available.

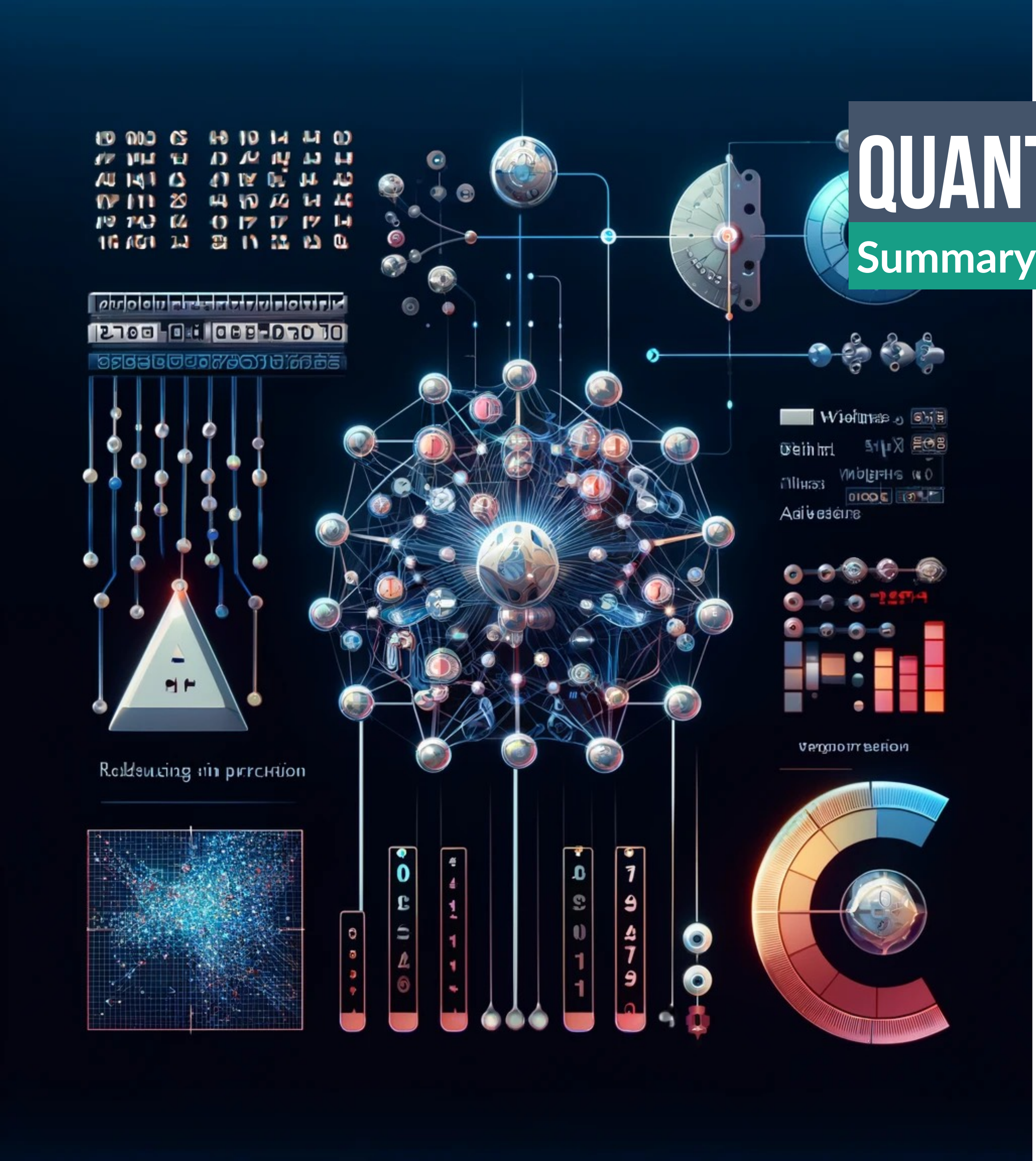
PRUNING

Summary

01 **Simplification**
Removes unnecessary neurons or connections.

02 **Reduction**
Reduces the number of parameters and computational load.

03 **Streamlining**
Ideal for models with a large number of parameters or apparent redundancies.



QUANTIZATION

Summary

01 **Compression**
Reduces the bit-width of weights and activations.

02 **Acceleration**
Enables smaller model size and faster execution with little to no loss in accuracy.

03 **Responsiveness**
Useful for real-time deployments needing faster execution times.

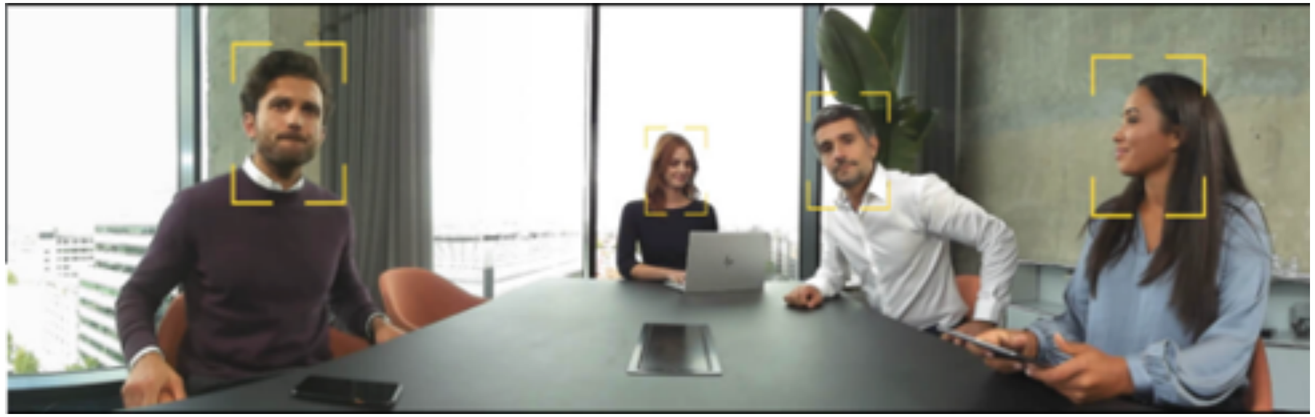
BREAK (10 MINUTES)

HOW TO DEPLOY AN OBJECT DETECTION ON QUALCOMM

OBJECT DETECTION

Jabra PanaCast P20, Jabra PanaCast 50, PanaCast 50 VBS

180-degrees of FoV
4K Video



MYRIADX REQUIREMENTS

Hardware Constraints

Myriad X devices support only FP16 bit widths and have limited memory and compute budget shared across all processes.

Latency

End-to-End acceptable model inference latency - 24 ms to 30 ms.

Range

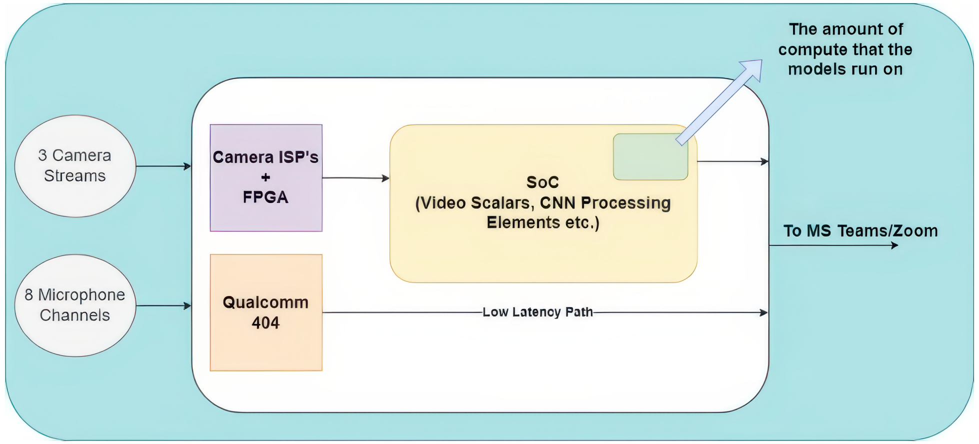
Model Working Distance - 18 ft to 20 ft (small/medium conference rooms).

Detected People

Must be able to detect 1 to 20 people.

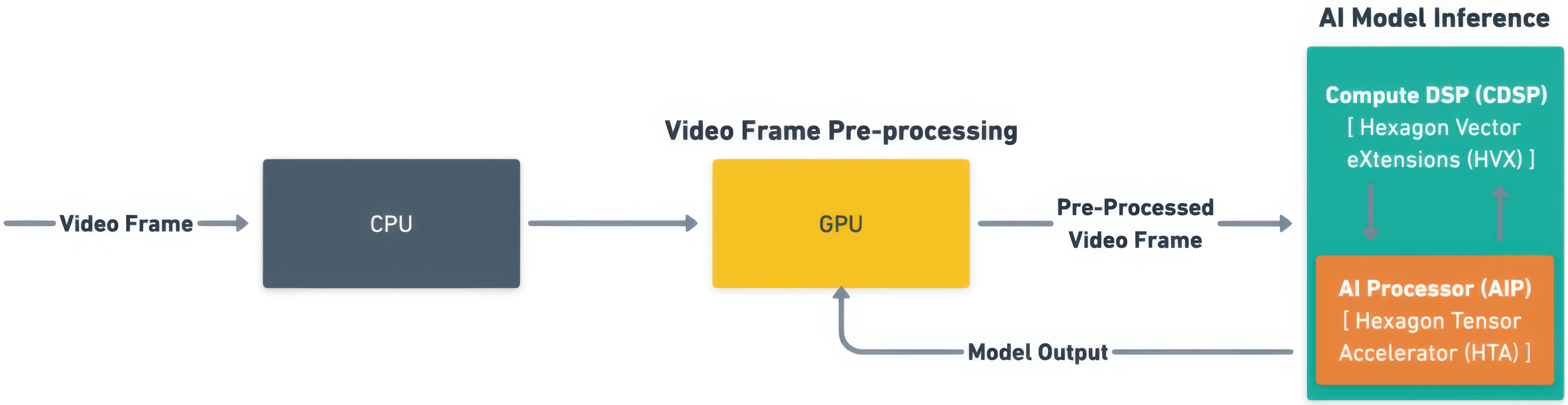
Precision

Low False Positives/False Negatives.



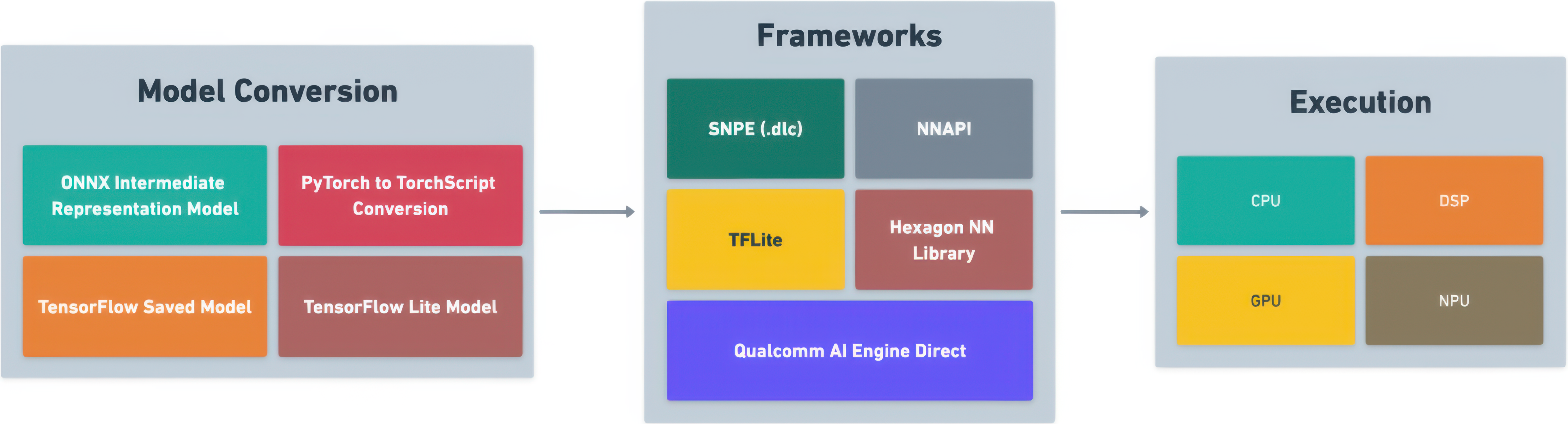
QUALCOMM INFERENCE END-TO-END

Workflow


AI Model Inference
Video Frame Pre-processing
Video Frame
CPU
GPU
Pre-Processed Video Frame
Compute DSP (CDSP) [Hexagon Vector eXtensions (HVX)]
AI Processor (AIP) [Hexagon Tensor Accelerator (HTA)]
Model Output

WORKFLOW FOR MODEL DEPLOYMENT

Deploying Machine Learning Models on Qualcomm Hardware



Model Conversion

ONNX Intermediate Representation Model

PyTorch to TorchScript Conversion

TensorFlow Saved Model

TensorFlow Lite Model

Frameworks

SNPE (.dlc)

NNAPI

TFLite

Hexagon NN Library

Qualcomm AI Engine Direct

Execution

CPU

DSP

GPU

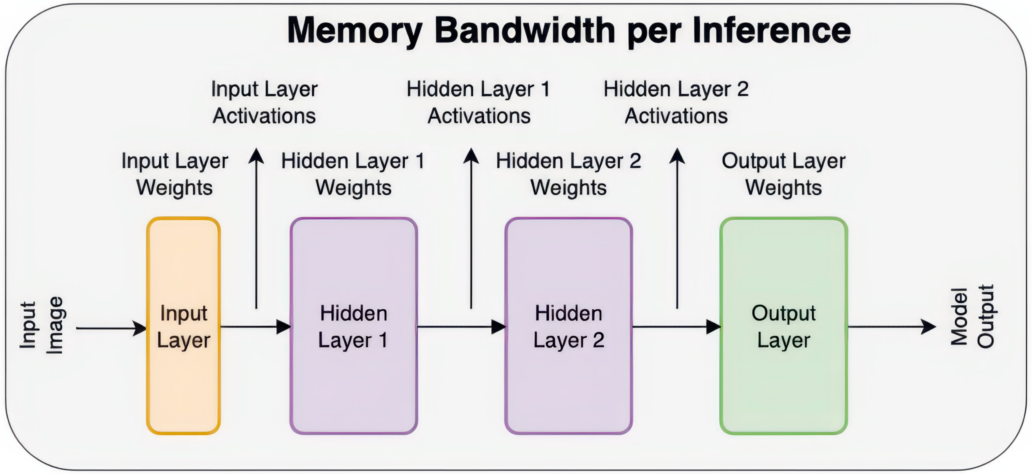
NPU

MEMORY BANDWIDTH

Challenge-1

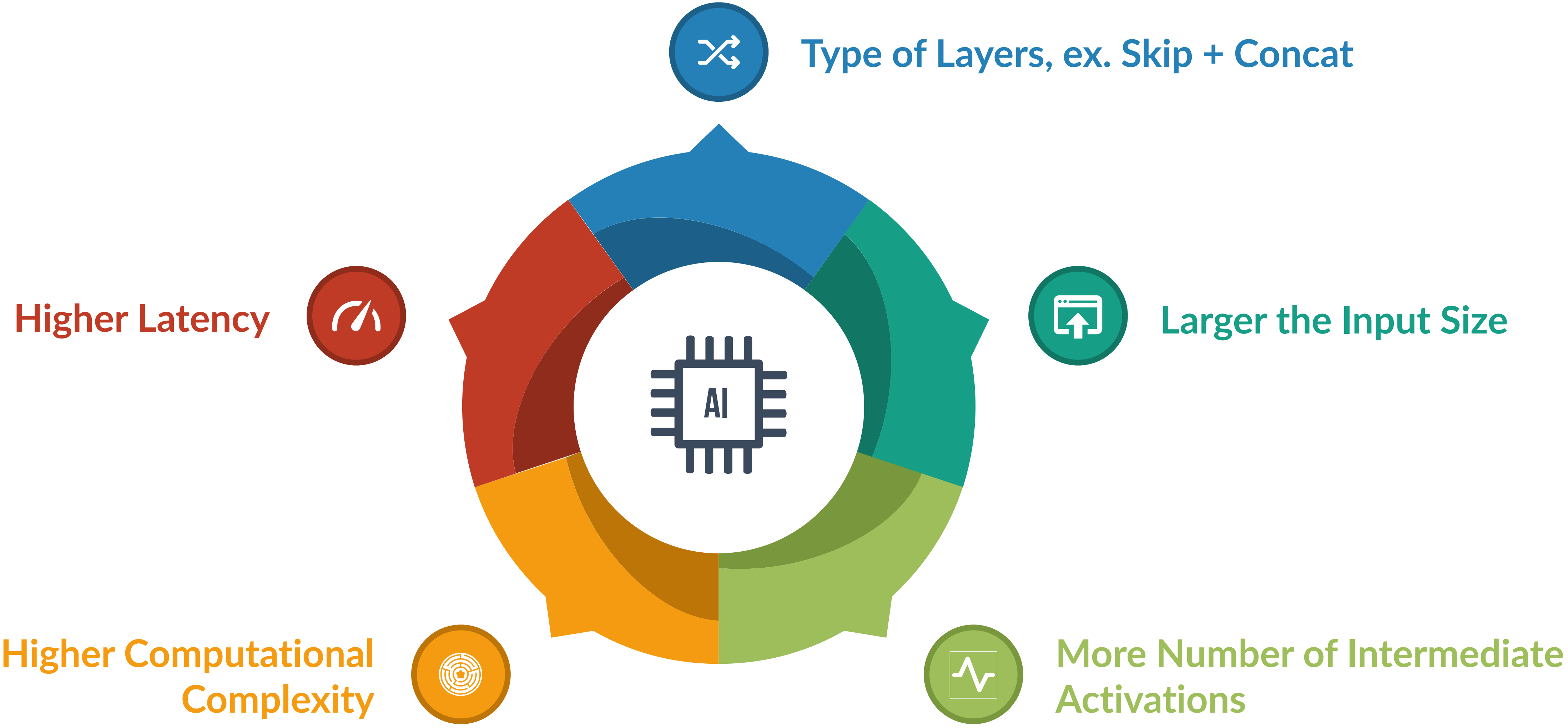
ML models utilize the same memory pool as other system processes. Some factors influencing Memory Bandwidth per Frame:

- INPUT LOAD
- MODEL LAYER WEIGHTS
- MODEL LAYER ACTIVATIONS
- MODEL OUTPUT



MODEL LATENCY

Challenge-2



Type of Layers, ex. Skip + Concat

Larger the Input Size

More Number of Intermediate Activations

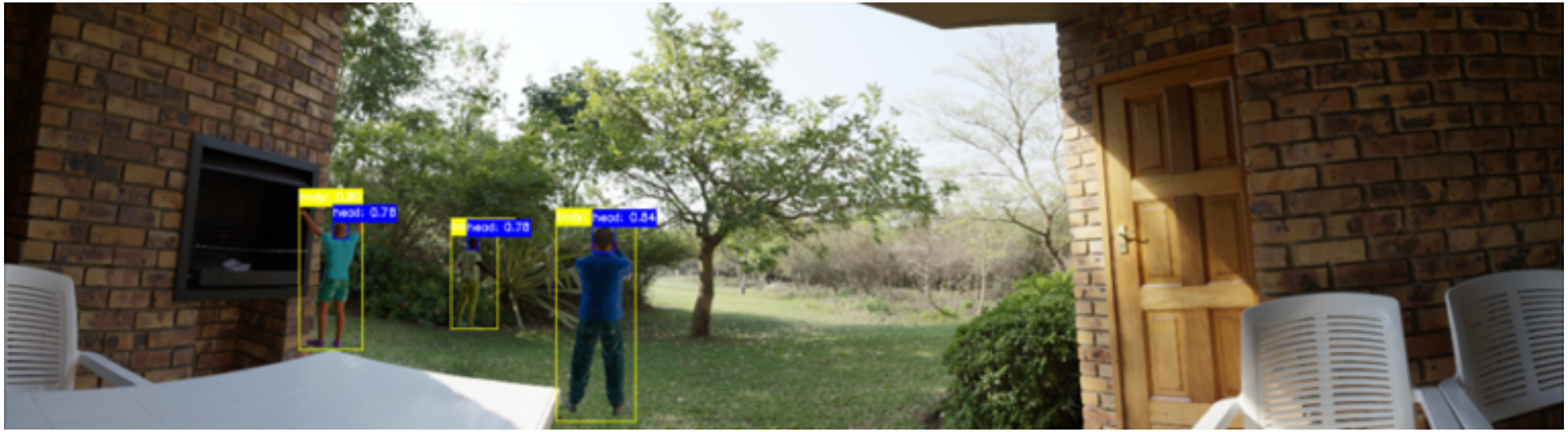
Higher Computational Complexity

Higher Latency

MODEL ACCURACY

Challenge-3

> “ Objects are harder to detection as they move away from the camera. ”



Input Resolution | Lighting Conditions | Occlusions | Scale Variations | Dataset Limitations

OTHER CHALLENGES

Overview

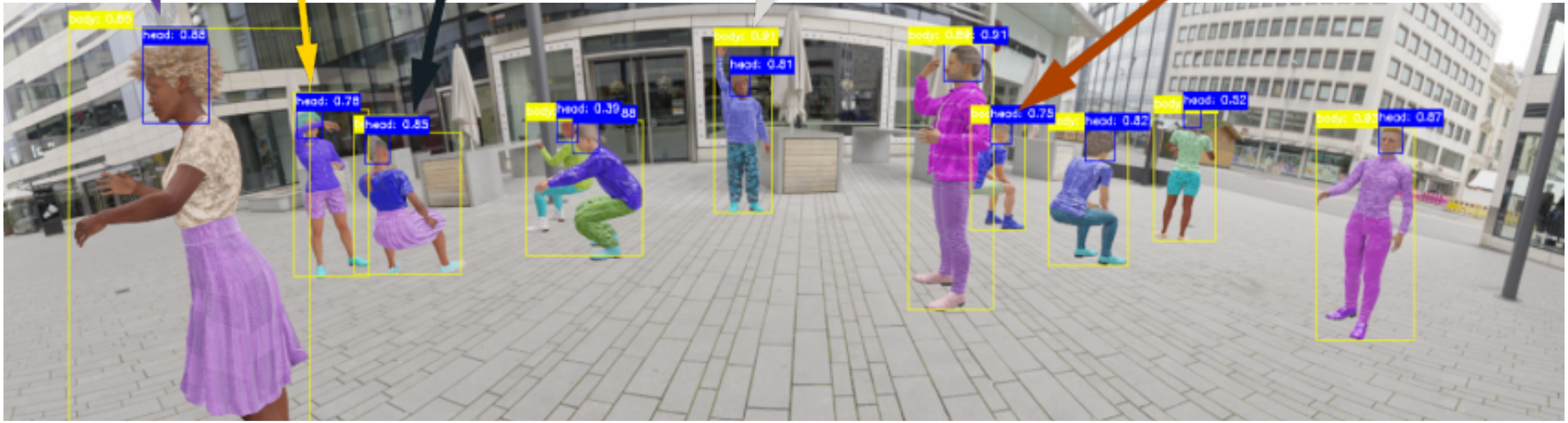
Person Facing
Sideways

Person's Face
Occluded by Hand

Person Facing
Away from the Camera

Person Far Away from
The Camera with Hand Raised

Person Partially Occluded
By Another Person in Front



PROBLEM IMPACT

Discussion

The *problem impact* includes potential memory overflow leading to frame corruption, frame rate reduction, and crash experience. Additionally, model latency may result in a less smooth experience, and the model's performance may be impacted by high false positives and false negatives.

Memory Bandwidth

Model needs to work along other processes utilizing same memory pool.

Latency

The model must work at-least at 27 to 30 FPS to pass Microsoft Teams/ Zoom certification.

Performance

The model must have low false positives and low false negatives.

GOALS

Discussion

Input

Input size is fixed

Reduce feature spatial dimension as soon as possible. This will help decrease latency and memory bandwidth required.

Parameters

Model parameter reduction

Reduce the number of parameters and operations by *Memory Bandwidth Reduction* and/or *Latency Reduction*.

Performance

Precision

Model mAP/mAR should improve, FP/FN should decrease.

MODEL DESIGNING

Understanding Hardware

Architecture

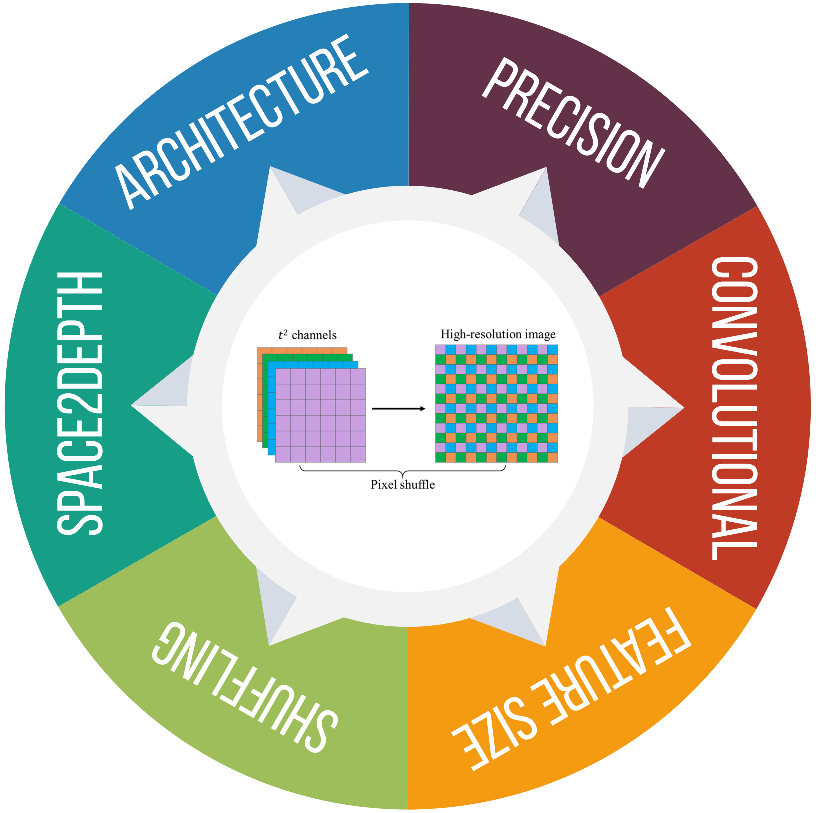
Making sure all the layers are executed using Neural Compute Engine (NCE)

Space2Depth

Use large kernel convolutions with large stride at input.

Pixel Shuffling

Use pixel shuffling at the output instead of TransposeConv2d



Half Precision Training

Train the model with FP16 precision to reduce quantization errors after deployment

Convolutional

Use efficient Conv layers like GhostConv, PartialConv, etc.

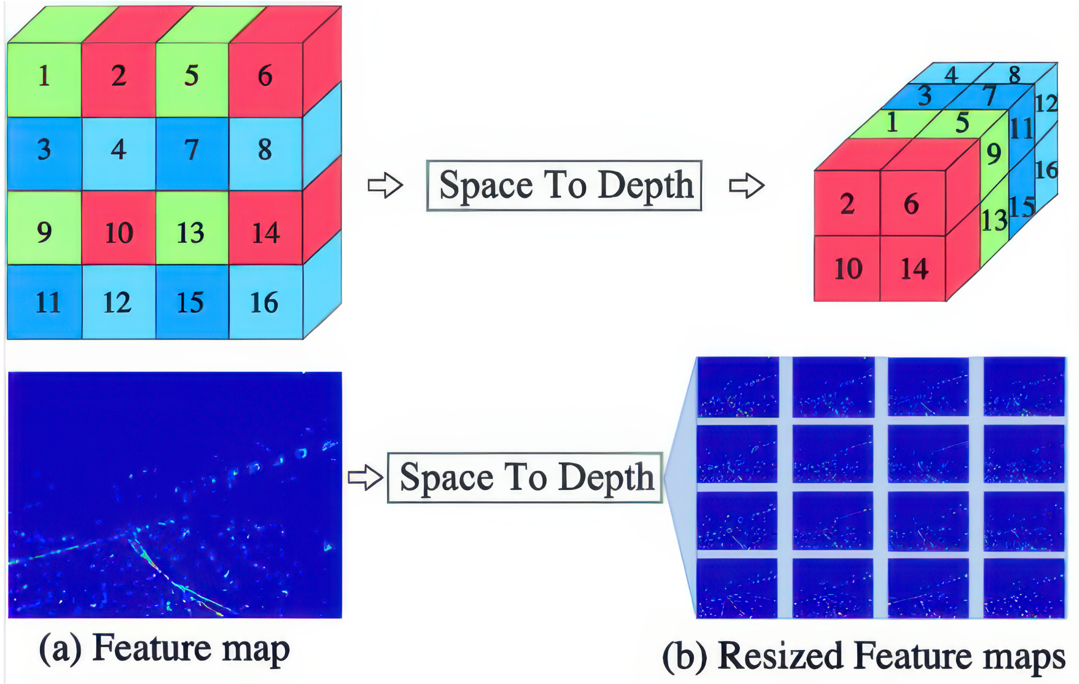
Feature Size

Use small feature size convolution layers to reduce copy-retrieve operations cost.

CHOSEN SOLUTIONS IN DETAIL

INPUT FEATURE SPATIAL

Size Reduction using S2D



COMBINES NEIGHBORING PIXEL VALUES INTO A HIGHER-DIMENSIONAL CHANNEL REPRESENTATION WHILE MAINTAINING THEIR SPATIAL RELATIONSHIP.

Provides a compact, enriched representation for the subsequent convolutional layer.

Prevents immediate loss of spatial correlations, unlike direct downsampling with a Conv2d operation

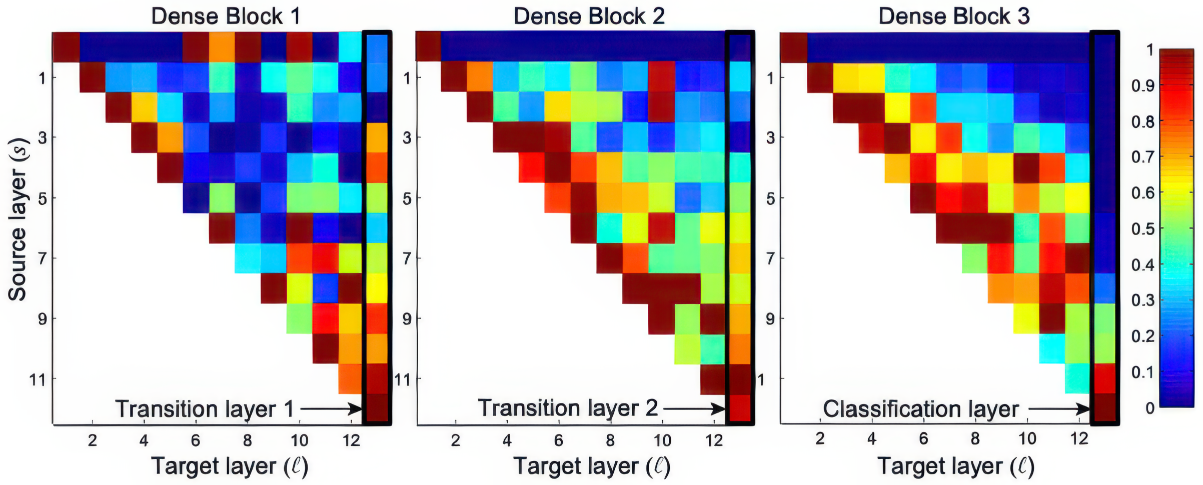
SPACE-TO-DEPTH VS. CONV2D

Results

Layer Type	Input Channels	Output Channels	MAC Operations	Number of Parameters
Conv2D + BN + ReLU	1	32	46.858 M	352 Bytes
Conv2D + BN + ReLU	32	64	2.105 G	18.56 K
Space-to-Depth	1	32	26.04 M	150 Bytes
Space-to-Depth	32	64	1.08 G	5.89 K

OPTIMIZING DOWN SAMPLE CONVOLUTIONS

Model Optimization



- 01 *Dense Connections,* promotes feature reuse across layers, saving on parameters and computations.
- 02 *Unique Concatenation,* combines features from prior layers, enhances feature richness, avoids duplication, and conserves memory bandwidth.
- 03 *Diverse Learning,* dense links foster varied feature learning due to added supervision from loss.
- 04 *Enhanced Propagation,* ensures improved feature spread and minimizes overfitting.
- 05 *Efficiency in Bandwidth,* reduced parameters and redundancy lead to less memory usage, conserving memory bandwidth.

DENSEFEATBLOCK VS. CONV2D

Results

Layer Type	Input Channels	Output Channels	MAC Operations	Number of Parameters
Conv2D + BN + ReLU	32	64	2.105 G	18.56 K
Conv2D + BN + ReLU	64	128	8.362 G	73.98 K
DenseFeatBlock	32	64	1.764 G	15.53 K
DenseFeatBlock	64	128	7 G	61.88 K

GHOST CONVOLUTIONS

Model Optimization

Feature Augmentation

Produces additional 'ghost' feature maps via DepthWiseConv2D.

Performance Boost

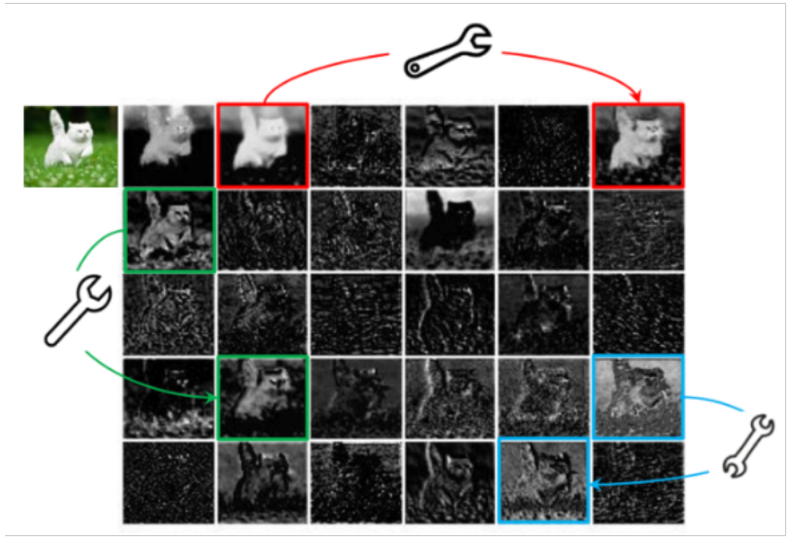
Offers lower FLOPS than Conv2D.

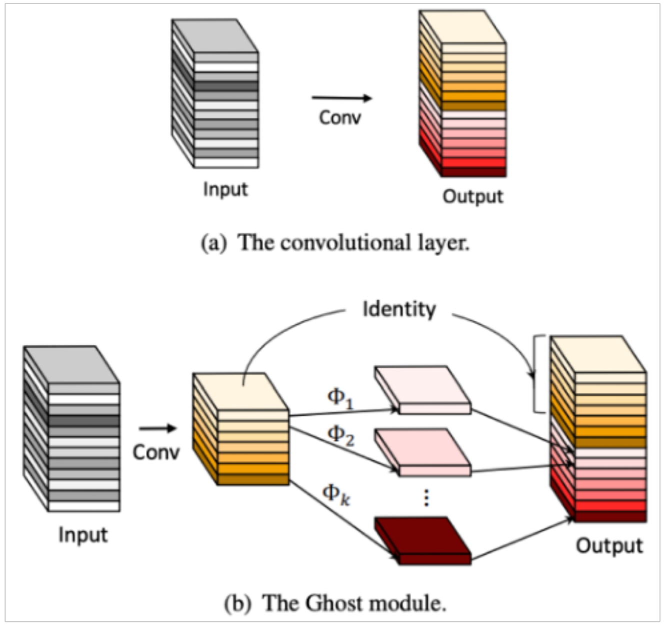
Example 01

Three similar feature map pair examples are annotated with boxes of the same color.

Example 02

One feature map in the pair can be obtained by transforming the other one through cheap operations (denoted by spanners).





(a) The convolutional layer.

(b) The Ghost module.

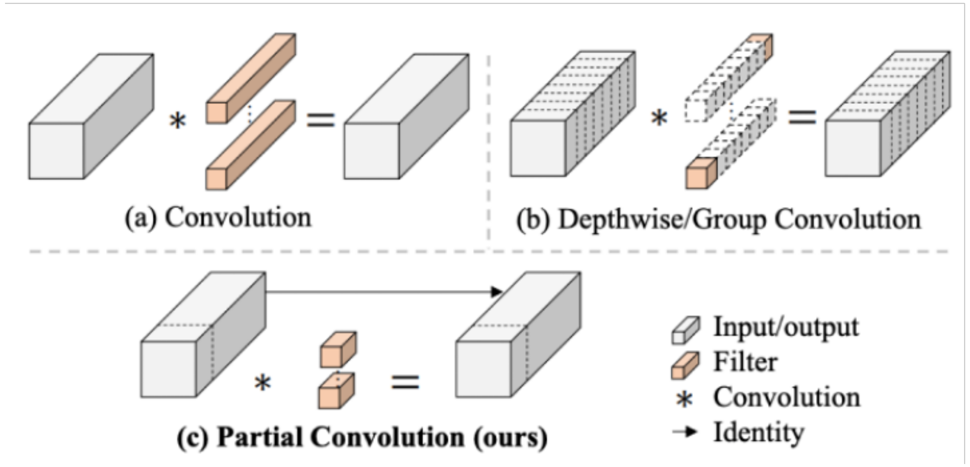
GHOSTCONV2D VS. CONV2D

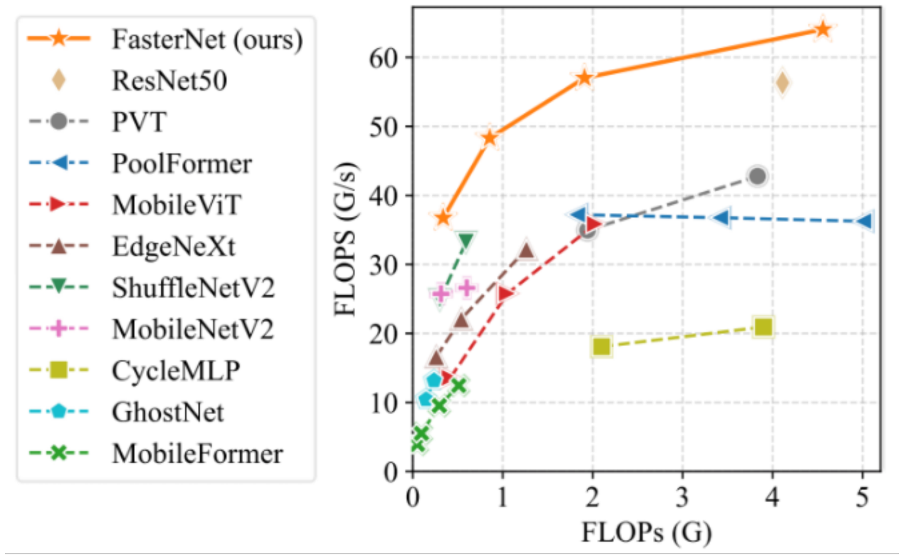
Results

Layer Type	Input Channels	Output Channels	MAC Operations	Number of Parameters
Conv2D + BN + ReLU	32	64	2.105 G	18.560 K
Conv2D + BN + ReLU	64	128	8.362 G	73.984 K
GhostConv2D	32	64	1.157 G	10.144 K
GhostConv2D	64	128	4.390 G	38.720 K

PARTIAL CONVOLUTION

Overview





DWConv2D

Faster then Conv2D but requires frequent memory access.

PConv2D

Cuts down on redundant computations and memory access simultaneously.

Efficiency

Cuts down on unnecessary computation and memory use compared to DepthWiseConv2D.

Optimized Operations

Uses fewer FLOPs than standard convolution but offers more FLOPS compared to DepthWise.

Latency

Higher FLOPS and Lower FLOPs mean Lower Latency.

PARTIALCONV2D VS. CONV2D

Results

Layer Type	Input Channels	Output Channels	MAC Operations	Number of Parameters
Conv2D + BN + ReLU	32	32	4.21 G	9.28 K
Conv2D + BN + ReLU	64	64	16.725 G	36.992 K
PartialConv2D	32	32	320.79 M	742 Bytes
PartialConv2D	64	64	1.157 G	2.630 K

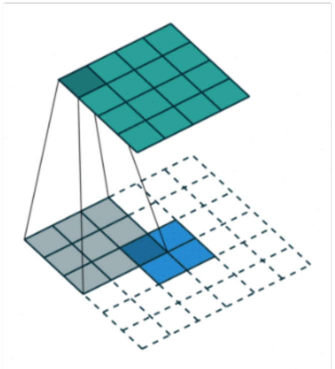
REPLACING TRANSPOSEDCONV2D

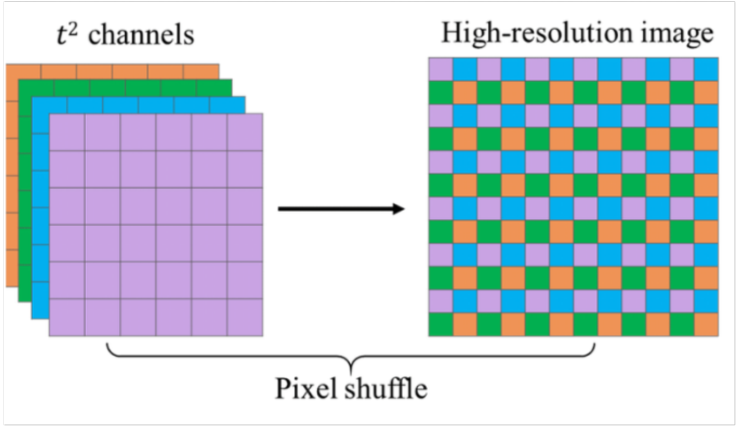
Overview

TransposedConv2d

Upsamples feature maps using learnable parameters.

TransposedConv2D





Pixel Shuffle

Rearranges elements in the feature map for upscaling without introducing new parameters.

START

OVERCOME DIFFICULTIES

SUCCESS

ON-DEVICE EXECUTION TIME ANALYSIS

Results

Layer Type	Width	Height	out_channels	stride	layer_exe_ms
PixelShuffle	32	32	128	2	0,228
PixelShuffle	16	16	128	2	0,127
PixelShuffle	8	8	128	2	0,066
TransposedConv2D	32	32	128	2	2.988
TransposedConv2D	16	16	128	2	0,833
TransposedConv2D	8	8	128	2	0,236

CHOICES IMPACT

Latency Results

~20%

Efficient Bottleneck Block

~20% improvement (due to reduced width).

~30%

PixelShuffle

~30% improvement (no learnable parameters, just rearrangement).

~40%

Partial Conv/GhostConv

~40% improvement (reduced operations, in GhostConv).

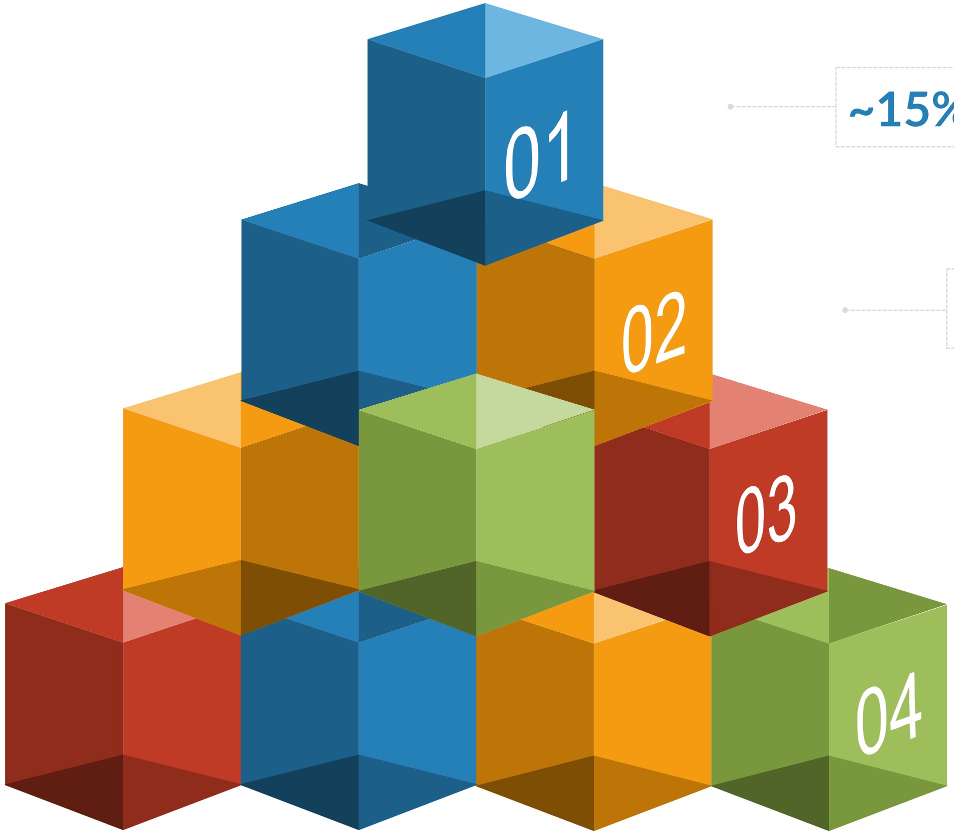
~15%

Skip/Concat

~15% improvement (fewer channels means fewer operations).

CHOICES IMPACT

Memory Bandwidth Results



~15%

Efficient Bottleneck Block

~15% reduction (due to fewer parameters).

~25%

PixelShuffle

~25% reduction (no weights storage).

~35%

Partial Conv/GhostConv

~35% reduction (fewer parameters).

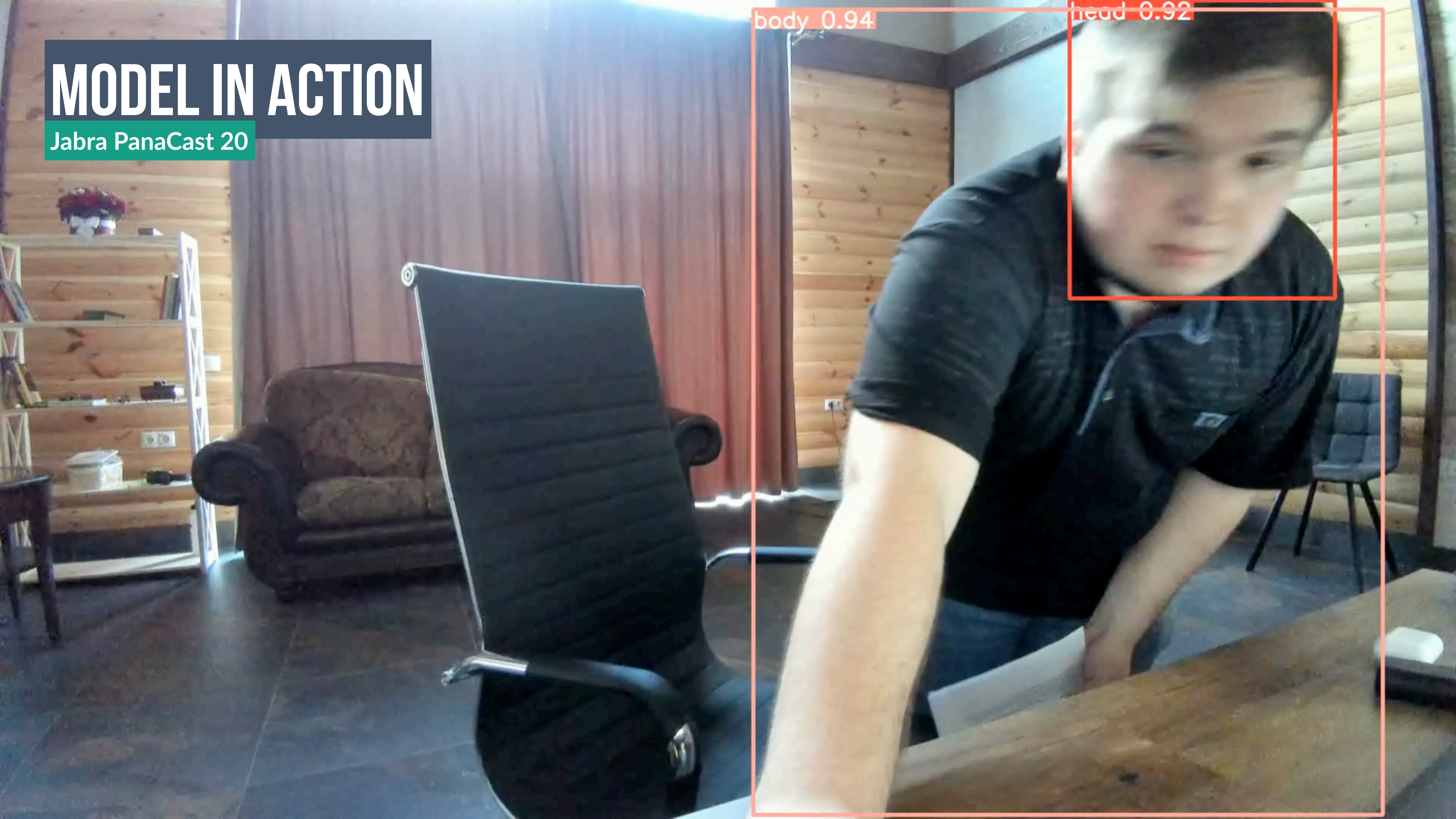
~10%

Skip/Concat

●~10% reduction (fewer channels means fewer weights).

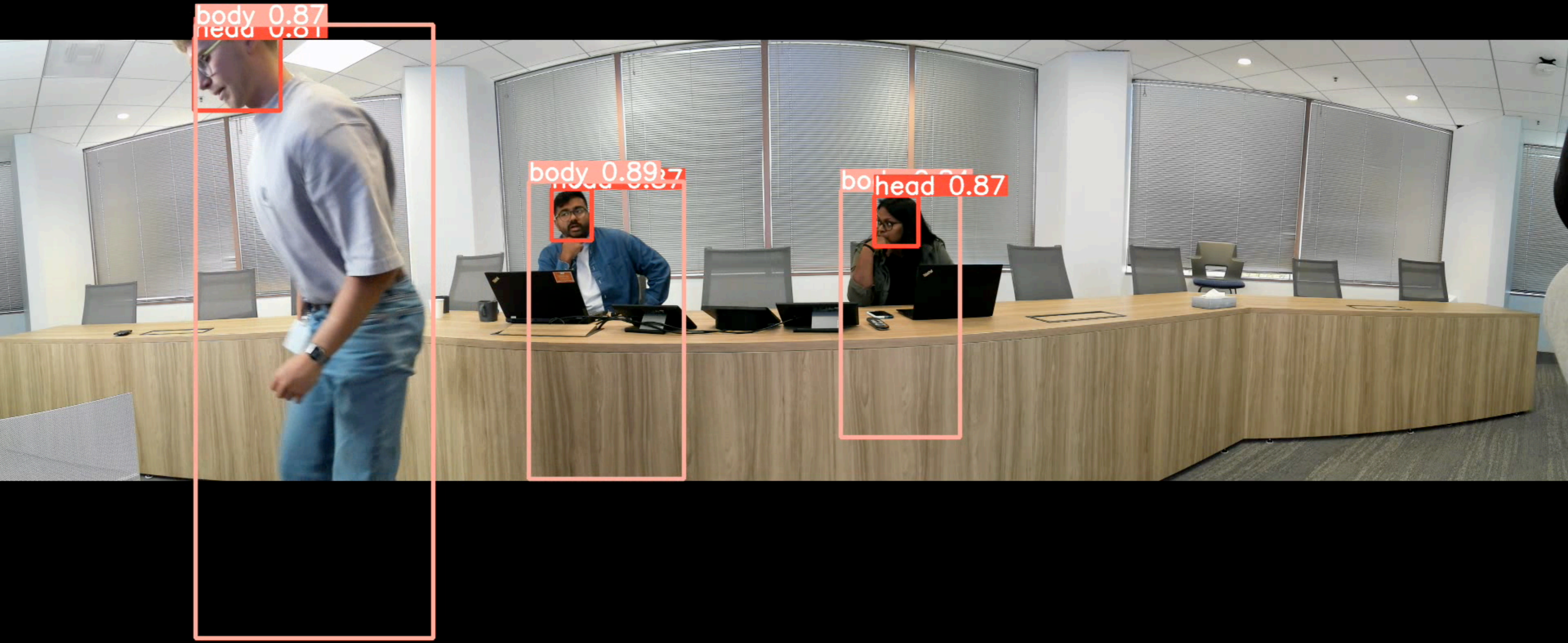
MODEL IN ACTION

Jabra PanaCast 20



MODEL IN ACTION

Jabra PanaCast 50 VBS



INTELLIGENT MEETING SPACES

Jabra PanaCast 50

HOW TO DEPLOY
A GAZE CORRECTION MODEL
ON INTEL MYRIAD X

GAZE CORRECTION

Case Study 2

This case study aims to deploy a gaze correction model on a resource-constrained device. The Luxonis OAK-1 MAX camera will feed its video stream with the user's eye contact for unified communication platforms.

Solution

Use the Intel OpenVINO Toolkit to optimize and deploy the model into a MyriadX chipset.

ONNX Format

Convert the model trained with TensorFlow or PyTorch to ONNX format.

Model Optimization

Use OpenVINO's Model Optimizer for conversion and optimization.

Model Deployment

Deploy the optimized model on an Intel-based edge device, e.g., Luxonis cameras.

MODEL DEPLOYMENT FOR EDGE AI

Deploy a Custom Model on Intel MyriadX on a MacBook M1

Model Deployment

Deploy the custom-trained (or pre-trained) model to the Edge device.

Model Inference

Use Inference Engine (IE) to evaluate the optimized model.

Blob Conversion

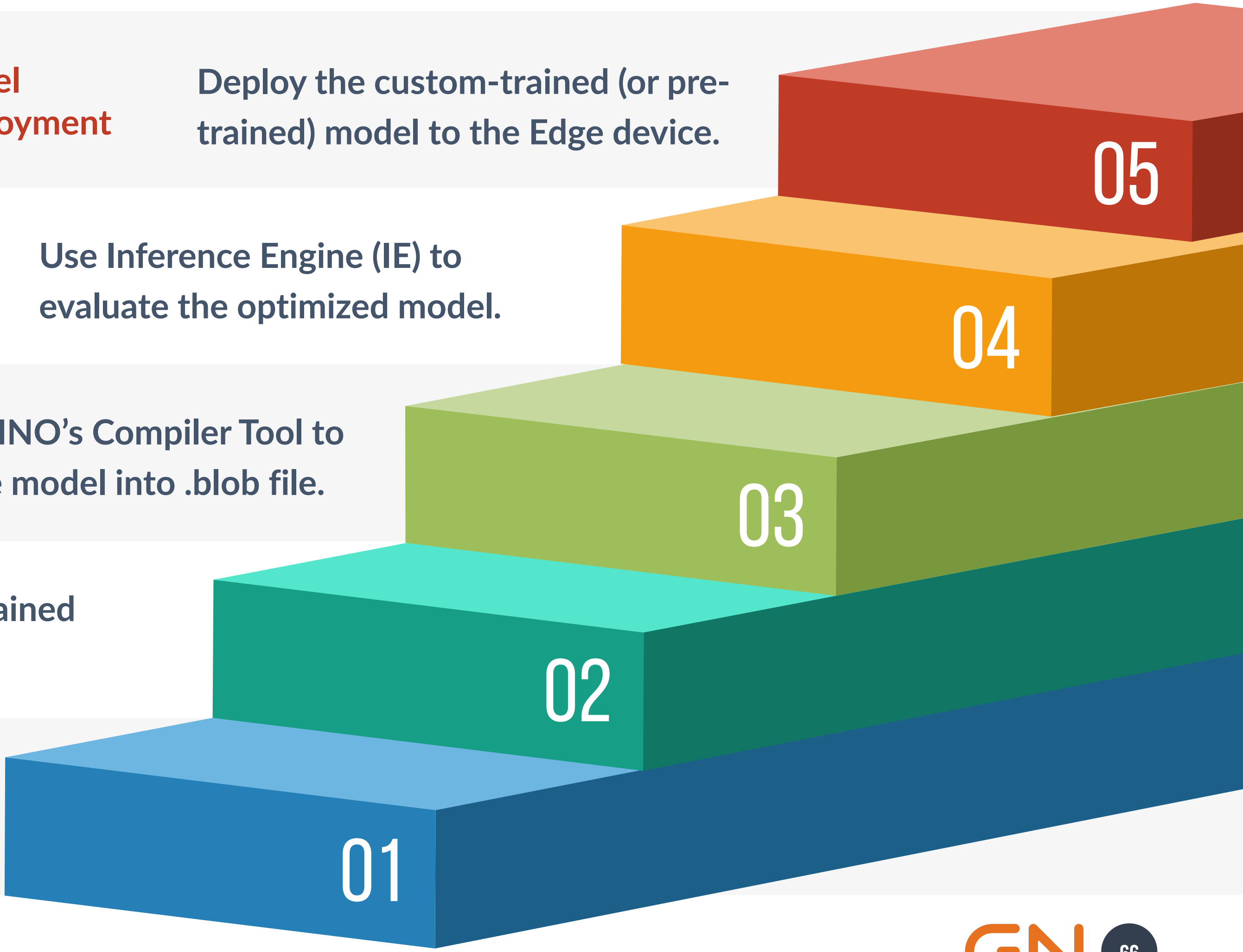
Use OpenVINO’s Compiler Tool to compile the model into .blob file.

ONNX Conversion

You must convert the trained model to ONNX format.

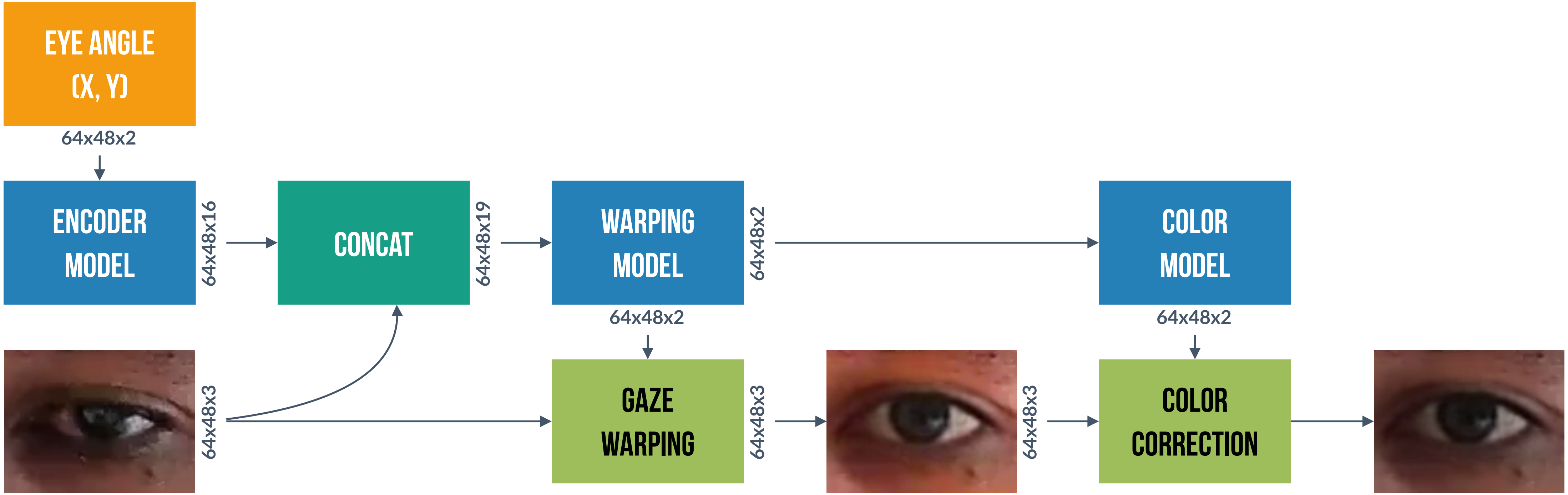
Model Development

You can design and train a model using TensorFlow or PyTorch.



JABRA EYE CORRECTION

Gaze Correction Model Based on Warping Technique



EYE ANGLE (X, Y)

64x48x2

ENCODER MODEL

64x48x16

CONCAT

64x48x19

WARPING MODEL

64x48x2

COLOR MODEL

64x48x2

64x48x2

64x48x3

GAZE WARPING

64x48x3

64x48x3

COLOR CORRECTION

🟦 ML Models
🟩 PyTorch Methods
🟩 CV Algorithm
🟧 Input Data

PYTORCH TO ONNX

Model Conversion

ONNX (*Open Neural Network Exchange*) provides a cross-platform solution to deploy models across different

THESE ARE THE PRIMARY TOOLS TO CONVERT A PYTORCH MODEL INTO AN ONNX FILE:

EXPORT

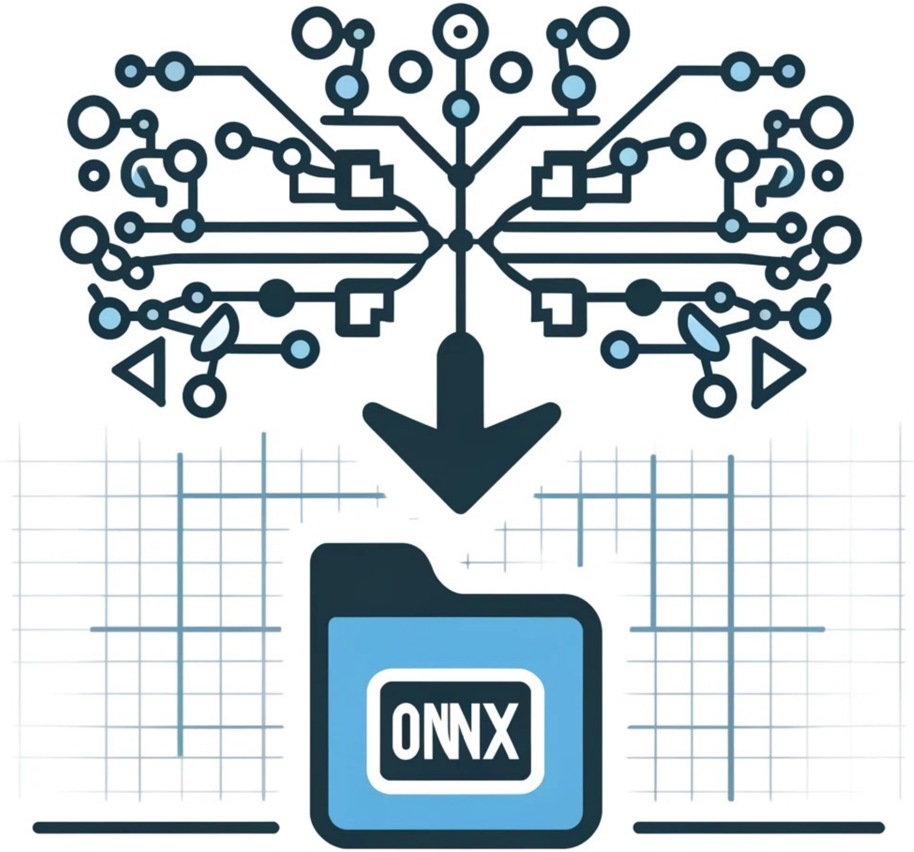
The *export* package is based on TorchScript backend and has been available since PyTorch 1.2.0.

TORCH DYNAMO

The *dynamo_export* package is the newest exporter based on the TorchDynamo technology.

ONNX RUNTIME

The exported model can be executed with ONNX Runtime for inferences across multiple platforms.



PYTORCH TO ONNX

Conversion Steps

Step 01

INSTALL PIP PACKAGES

```
$ pip install onnx
$ pip install onnxscript
```

Step 02

EXPORT THE MODEL TO ONNX FORMAT

```
model = ColorModel()
tensor = torch.randn(1, 2, 48, 64)
onnx_model  = torch.onnx.dynamo_export(model, tensor)
```

Step 03

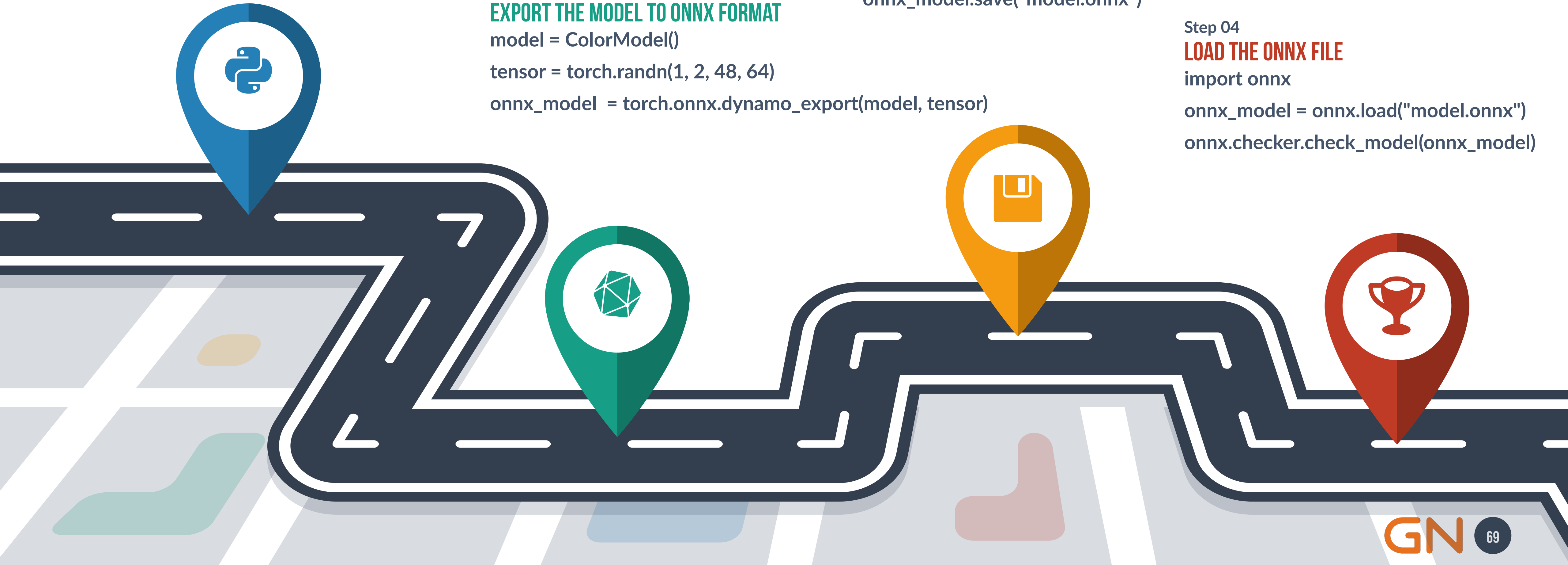
SAVE THE ONNX MODEL

```
onnx_model.save("model.onnx")
```

Step 04

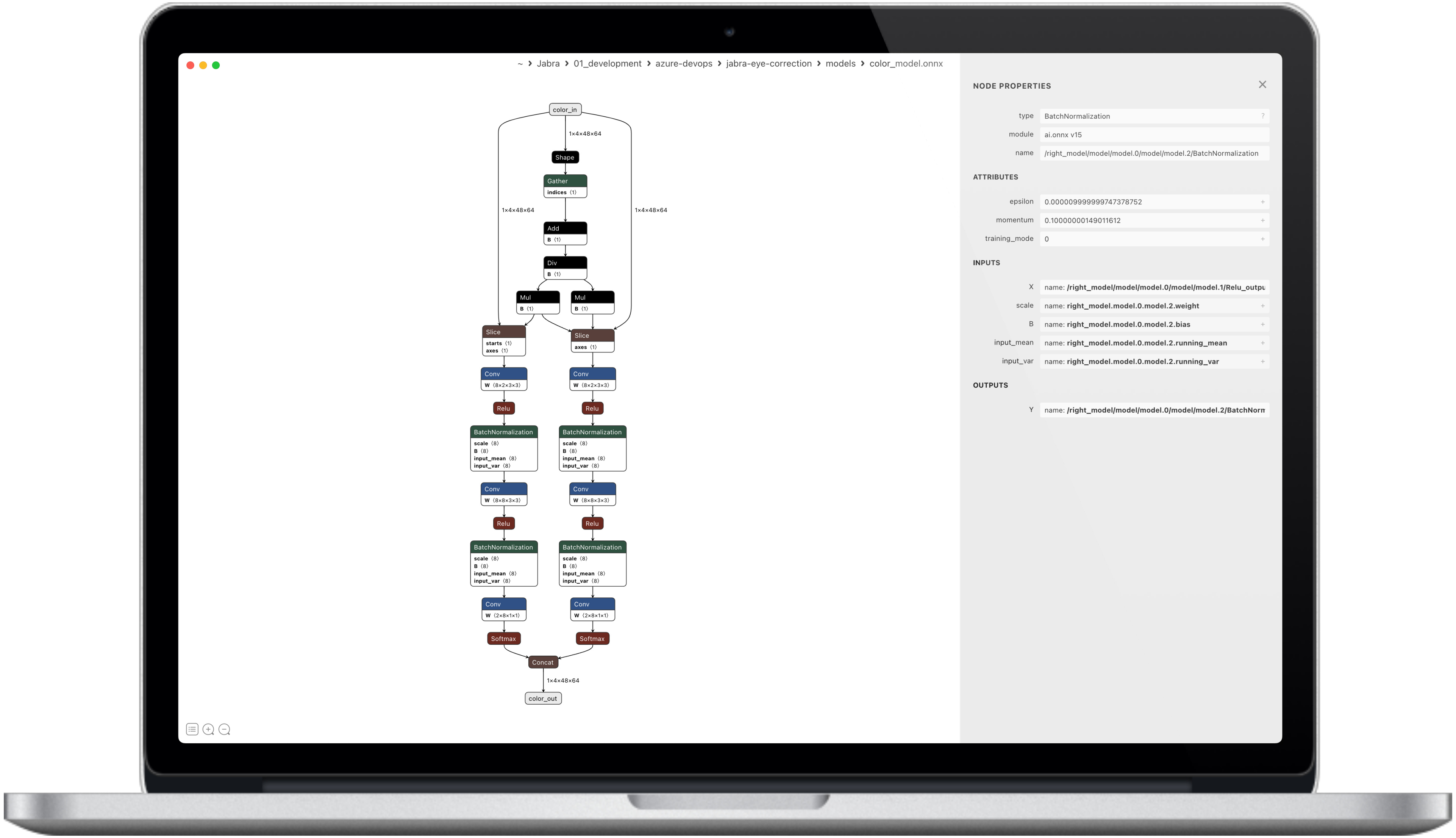
LOAD THE ONNX FILE

```
import onnx
onnx_model = onnx.load("model.onnx")
onnx.checker.check_model(onnx_model)
```



PYTORCH TO ONNX

Visualize the ONNX model graph using Netron app



NODE PROPERTIES

type: BatchNormalization

module: ai.onnx v15

name: /right_model/model/model.0/model/model.2/BatchNormalization

ATTRIBUTES

epsilon: 0.000009999999747378752

momentum: 0.10000000149011612

training_mode: 0

INPUTS

X: name: **/right_model/model/model.0/model/model.1/Relu_outpu**

scale: name: **right_model.model.0.model.2.weight**

B: name: **right_model.model.0.model.2.bias**

input_mean: name: **right_model.model.0.model.2.running_mean**

input_var: name: **right_model.model.0.model.2.running_var**

OUTPUTS

Y: name: **/right_model/model/model.0/model/model.2/BatchNorm**

MYRIADX BLOB CONVERSION

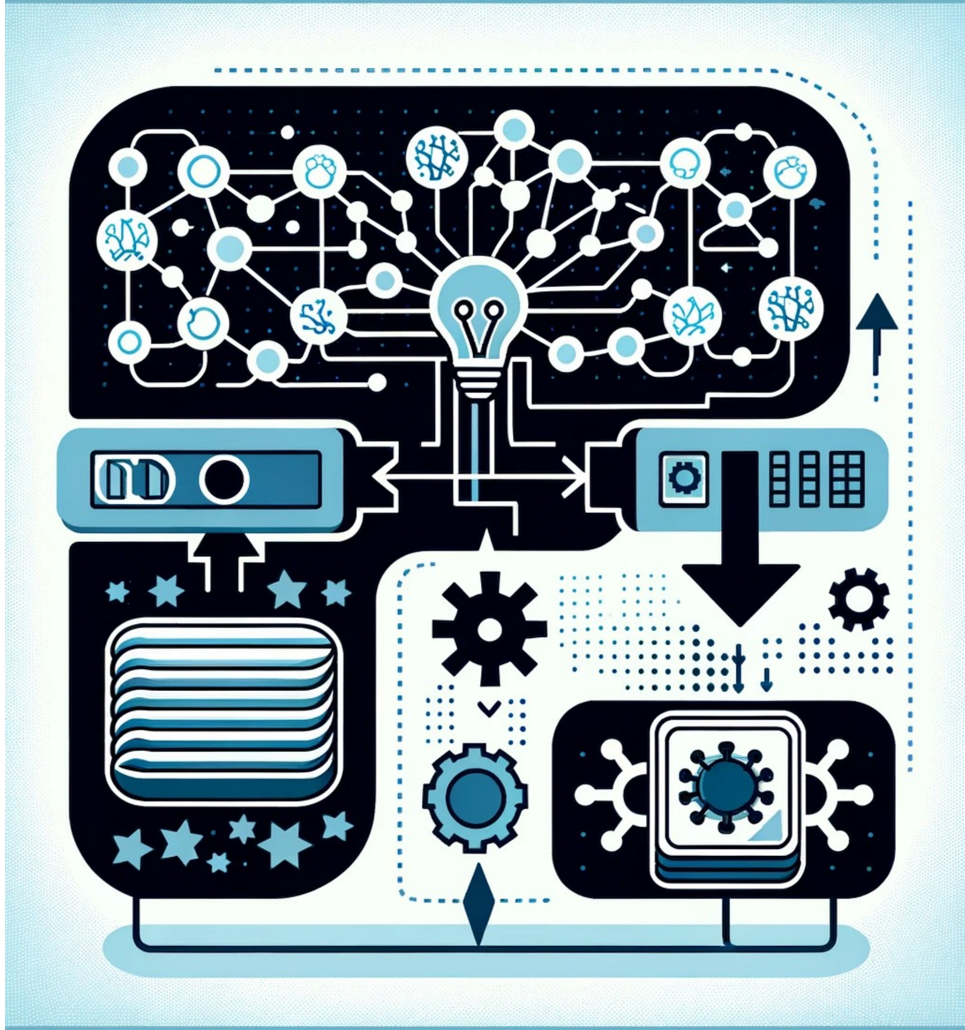
Conversion Tools

MODEL OPTIMIZER

The Model optimizer of OpenVINO converts the model from its original framework format into the Intermediate Representation (IR) standard format of OpenVINO (.bin and .xml).

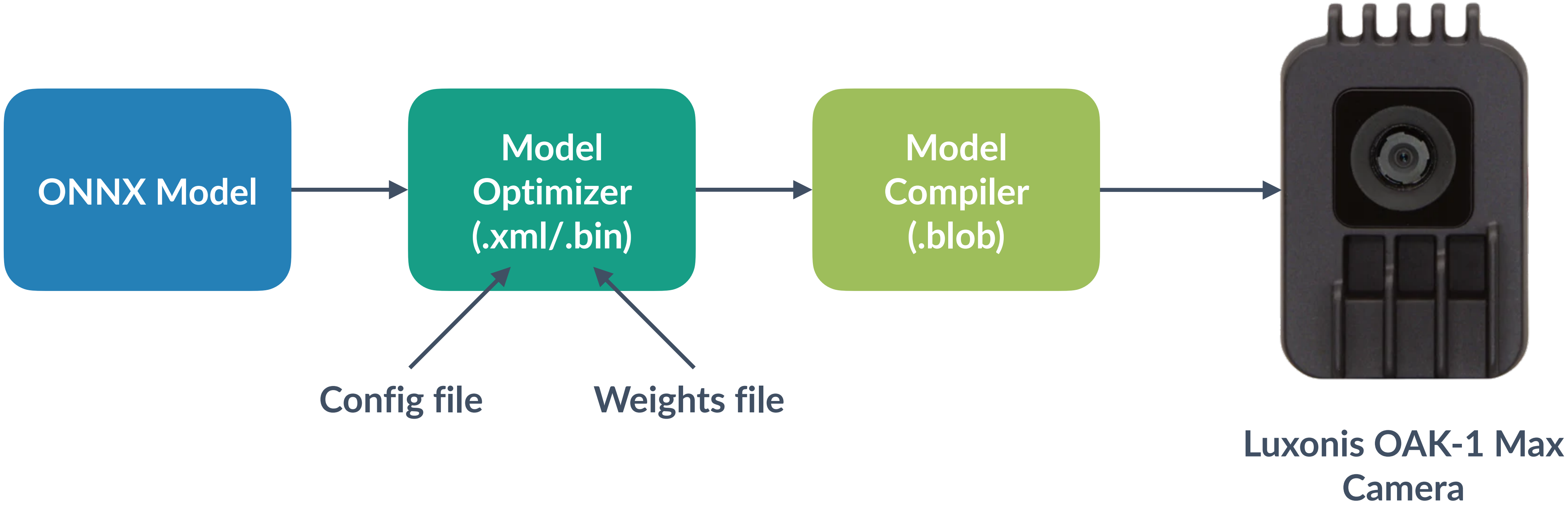
COMPILE TOOL

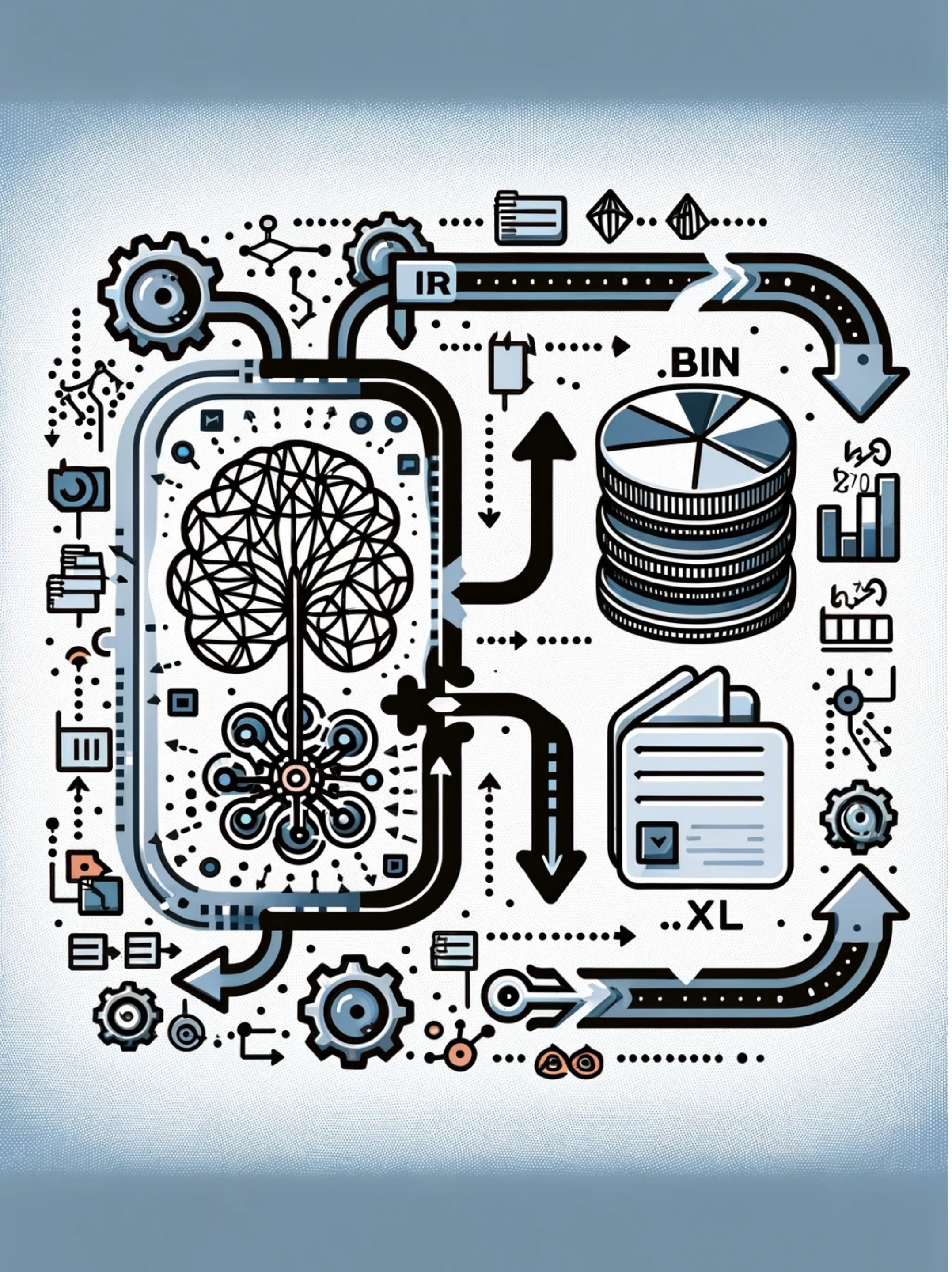
After converting the model to OpenVINO's IR format (.bin/.xml), you must use Compile Tool to compile the model in IR format into a .blob file, which can then be deployed to the device.



MYRIADX BLOB CONVERSION

Conversion Steps





OPENVINO'S MODEL OPTIMIZER

Overview

The initial step is to utilize the Model Optimizer to generate the OpenVINO IR representation (where IR stands for Intermediate Representation).

FP16 Data Type

When converting the model for VPU (OpenVINO MyriadX), the generated IR must be compressed to FP16.

Mean and Scale

You must normalize the mean and scale parameters before running the optimized model in the MyriadX device.

Model Layout

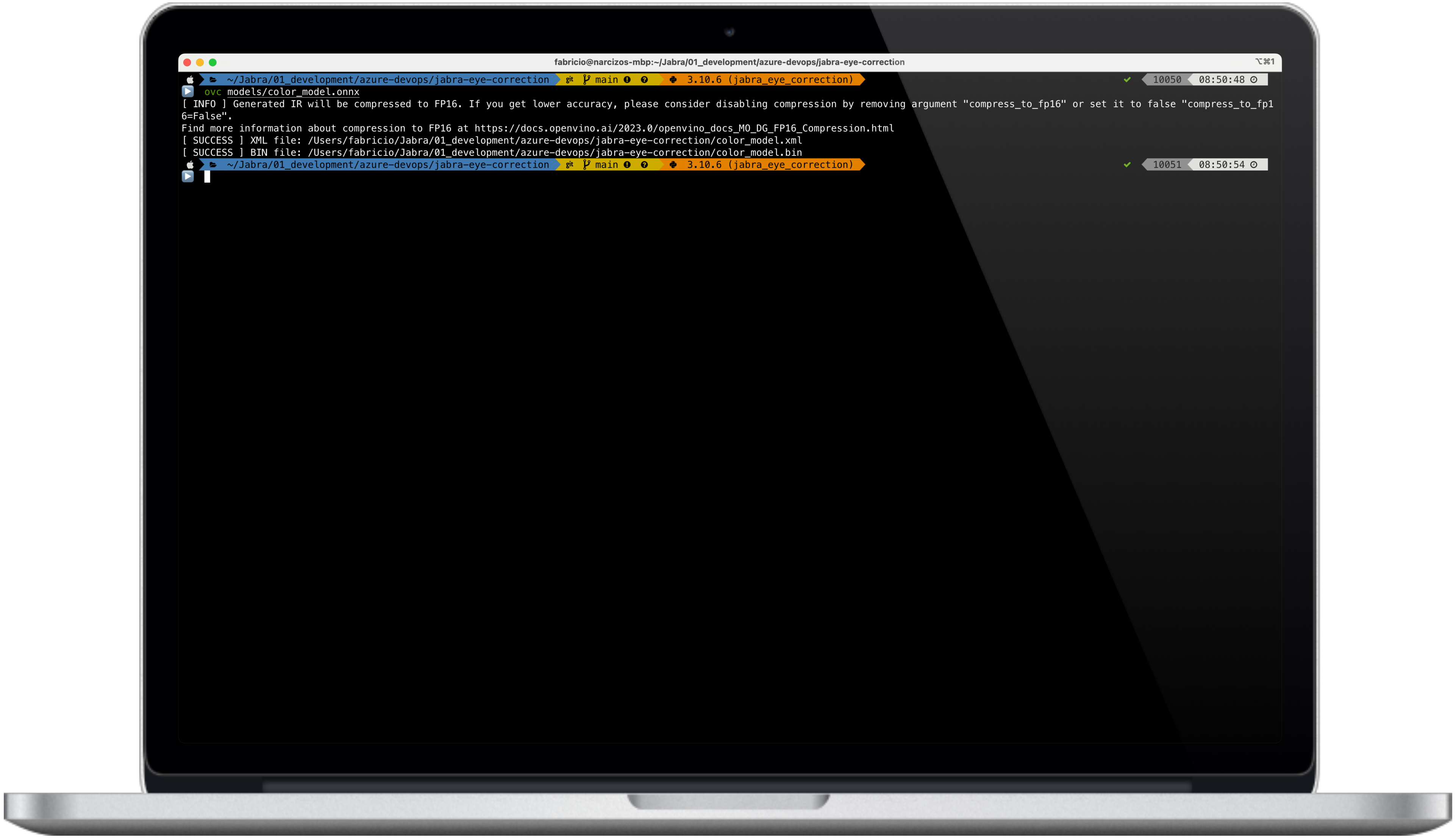
It defines the input/output tensor shape and whether it uses a *Planar Layout* (CHW) or an *Interleaved Layout* (HWC).

Color Order

For standard, OpenVINO uses the BGR color system. However, NN models can be trained on either RGB or BGR color order.

CONVERT ONNX TO OPENVINO

ovc models/color_model.onnx


fabricio@narcizos-mbp:~/Jabra/01_development/azure-devops/jabra-eye-correction
⌥⌘1

```
 ~/Jabra/01_development/azure-devops/jabra-eye-correction  git  main      3.10.6 (jabra_eye_correction)                ✓  10050  08:50:48
▶  ovc models/color_model.onnx
[ INFO ] Generated IR will be compressed to FP16. If you get lower accuracy, please consider disabling compression by removing argument "compress_to_fp16" or set it to false "compress_to_fp1
6=False".
Find more information about compression to FP16 at https://docs.openvino.ai/2023.0/openvino_docs_MO_DG_FP16_Compression.html
[ SUCCESS ] XML file: /Users/fabricio/Jabra/01_development/azure-devops/jabra-eye-correction/color_model.xml
[ SUCCESS ] BIN file: /Users/fabricio/Jabra/01_development/azure-devops/jabra-eye-correction/color_model.bin
 ~/Jabra/01_development/azure-devops/jabra-eye-correction  git  main      3.10.6 (jabra_eye_correction)                ✓  10051  08:50:54
▶  
```

OPENVINO'S COMPILE TOOL

Overview

The second step is to use OpenVINO's Compile Tool to compile the model in Intermediate Representation (IR) format into a .blob file.

Input Layer Precision

RVC2 only supports FP16, so using the parameter -ip U8 will add a conversion layer *U8->FP16* on all input layers.

FP16 Data Type

In some cases, such as when not dealing with frames, you can use the parameter -ip FP16 to use FP16 precision directly.

MyriadX Shaves

The RVC2 has 16 SHAVE cores. Compiling for more SHAVEs can improve the model's performance.

Default Shaves

By default, each model will run on 2 threads. The firmware will alert you about the potentially optimal number of shave cores.

OPENVINO'S COMPILE TOOLS

There are a few options to compile models to Edge AI

Online Blob Converter App

You can access the online Blob Converter app, which converts and compiles the NN model.

Local Compilation

You can utilize the OpenVINO's Toolkit to perform model conversion and compilation locally.

Blob Converter Library

The Blob Converter PyPi package enables the conversion and compilation of models from both the command line and Python script.

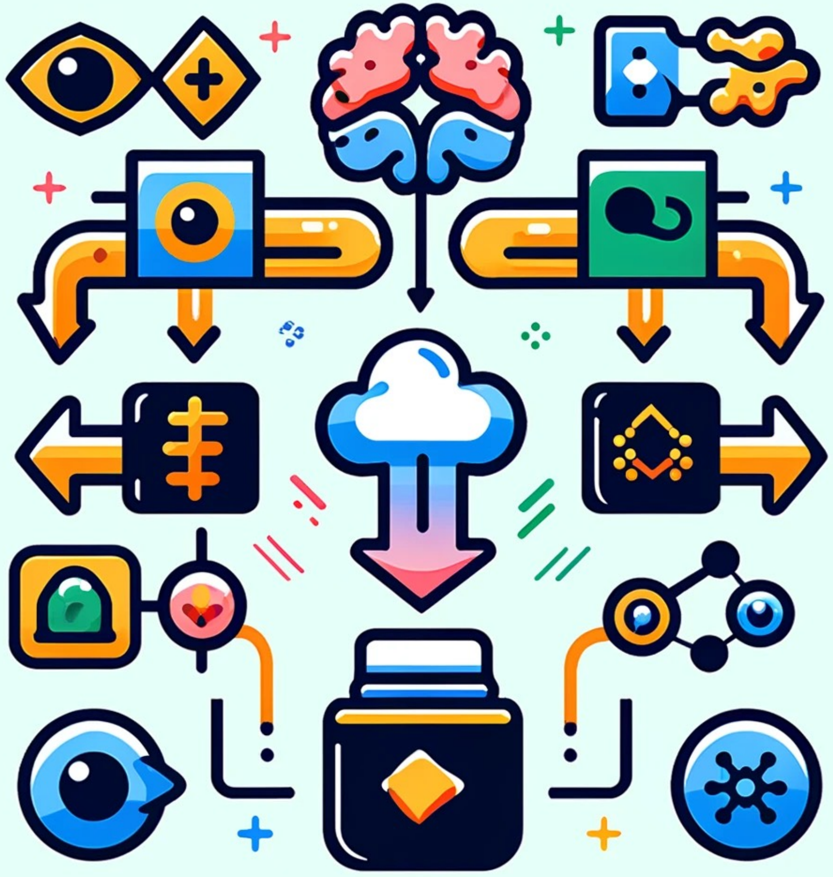
BLOB CONVERTER LIBRARY

pip install blobconverter

This Python library converts neural network files from various sources, such as TensorFlow, PyTorch, Caffe, or OpenVINO, into MyriadX blob files.

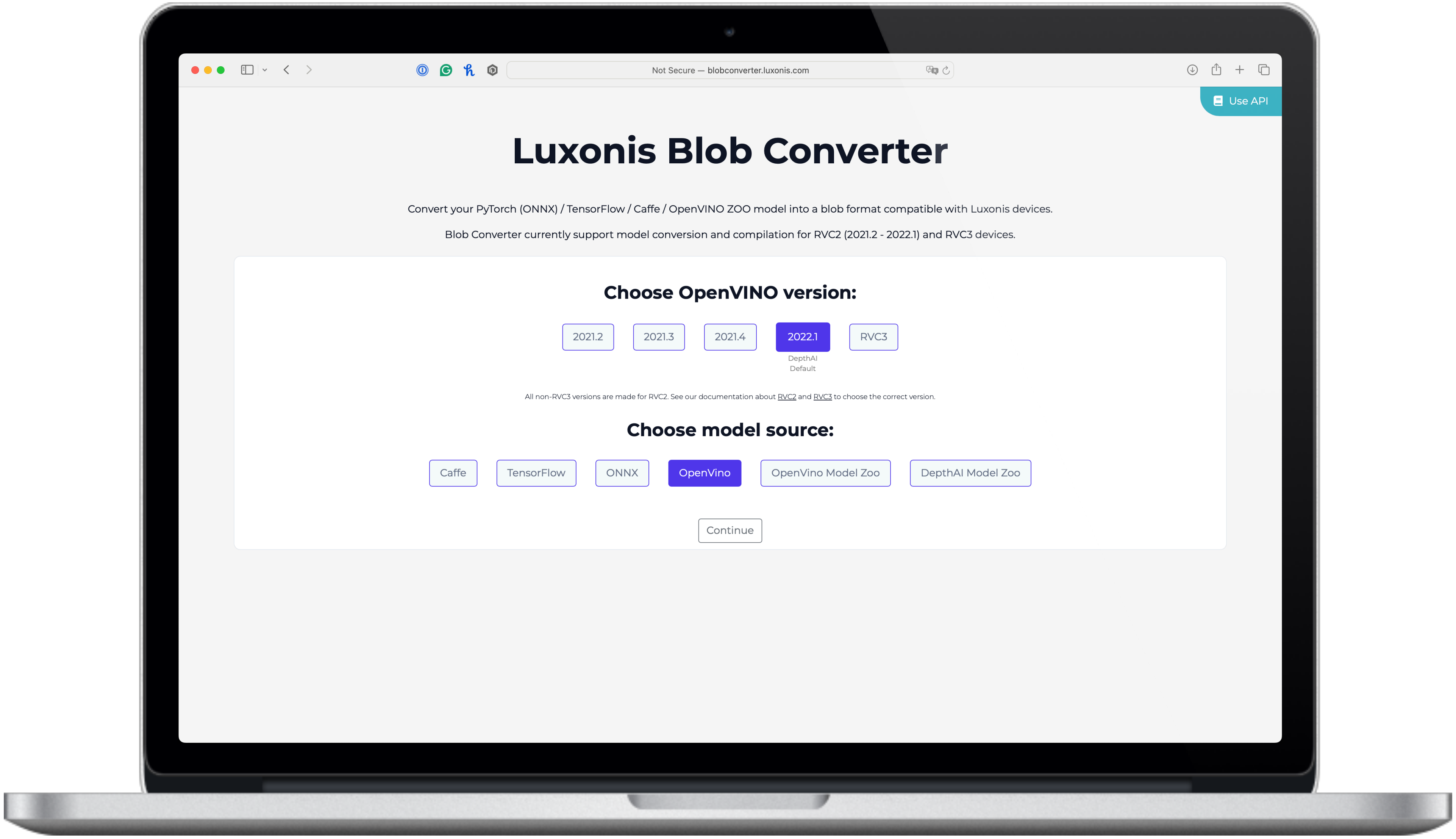
```
import blobconverter

blobconverter.from_onnx(
    model="models/color_model.onnx",
    data_type="FP16",
    shaves=5,
    use_cache=False,
    output_dir="models",
    optimizer_params=[],
    compile_params=[]
)
```



OPENVINO'S COMPILE TOOLS

Online Blob Converter App



Not Secure — blobconverter.luxonis.com

Use API

Luxonis Blob Converter

Convert your PyTorch (ONNX) / TensorFlow / Caffe / OpenVINO ZOO model into a blob format compatible with Luxonis devices.

Blob Converter currently support model conversion and compilation for RVC2 (2021.2 - 2022.1) and RVC3 devices.

Choose OpenVINO version:

2021.2 | 2021.3 | 2021.4 | **2022.1** (DepthAI Default) | RVC3

All non-RVC3 versions are made for RVC2. See our documentation about RVC2 and RVC3 to choose the correct version.

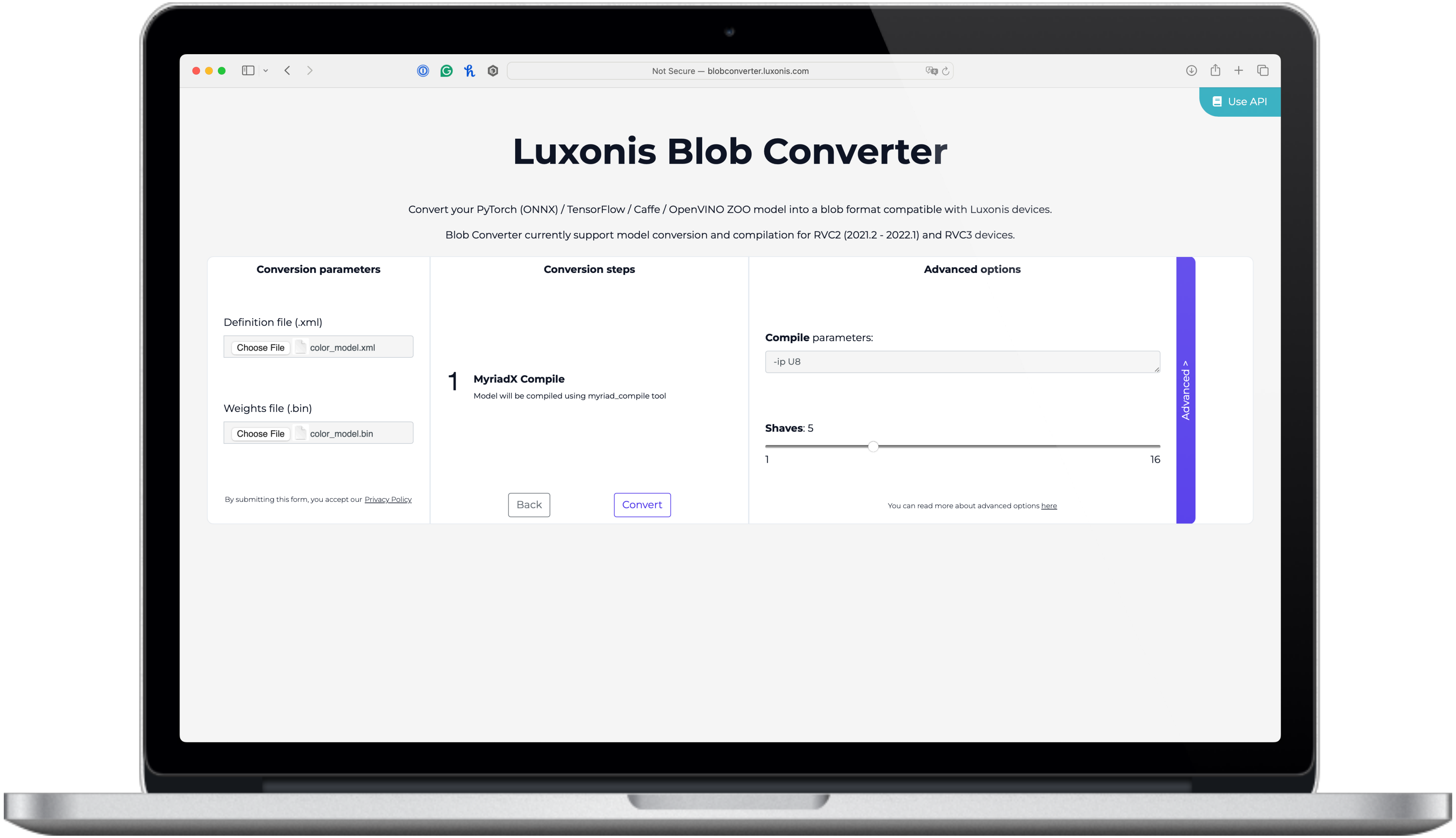
Choose model source:

Caffe | TensorFlow | ONNX | **OpenVino** | OpenVino Model Zoo | DepthAI Model Zoo

Continue

OPENVINO'S COMPILE TOOLS

Online Blob Converter App



Luxonis Blob Converter

Convert your PyTorch (ONNX) / TensorFlow / Caffe / OpenVINO ZOO model into a blob format compatible with Luxonis devices.

Blob Converter currently support model conversion and compilation for RVC2 (2021.2 - 2022.1) and RVC3 devices.

Conversion parameters

Definition file (.xml)
Choose File color_model.xml

Weights file (.bin)
Choose File color_model.bin

By submitting this form, you accept our Privacy Policy

Conversion steps

1 **MyriadX Compile**
Model will be compiled using myriad_compile tool

Advanced options

Compile parameters:
-ip U8

Shaves: 5
1 16

You can read more about advanced options here

LOCAL COMPILATION MODEL

OpenVINO Toolkit

You can use the following Python script to compile a model for inference on a specific device, as the Compile Tool is now deprecated.

```python
import openvino.runtime as ov

core = ov.Core()

model = core.read_model(model="color_model.xml")
compiled_model = core.compile_model(
    model=model, device_name="MYRIAD")
output_stream = compiled_model.export_model()

with open("color_model.blob", "wb") as f:
    f.write(output_stream)
```

DEPLOYING CUSTOM MODELS

Luxonis OAK-1 Max

NOW THAT YOU HAVE THE .BLOB FILE, YOU CAN BEGIN DESIGNING THE DEPTHAI PIPELINE. THESE ARE THE PRIMARY COMPONENTS:

Pipeline

It is a collection of nodes that defines the processing flow.

NeuralNetwork

This node runs neural network inference on input data.

XLinkIn

This node sends data from the host to the device via XLink.

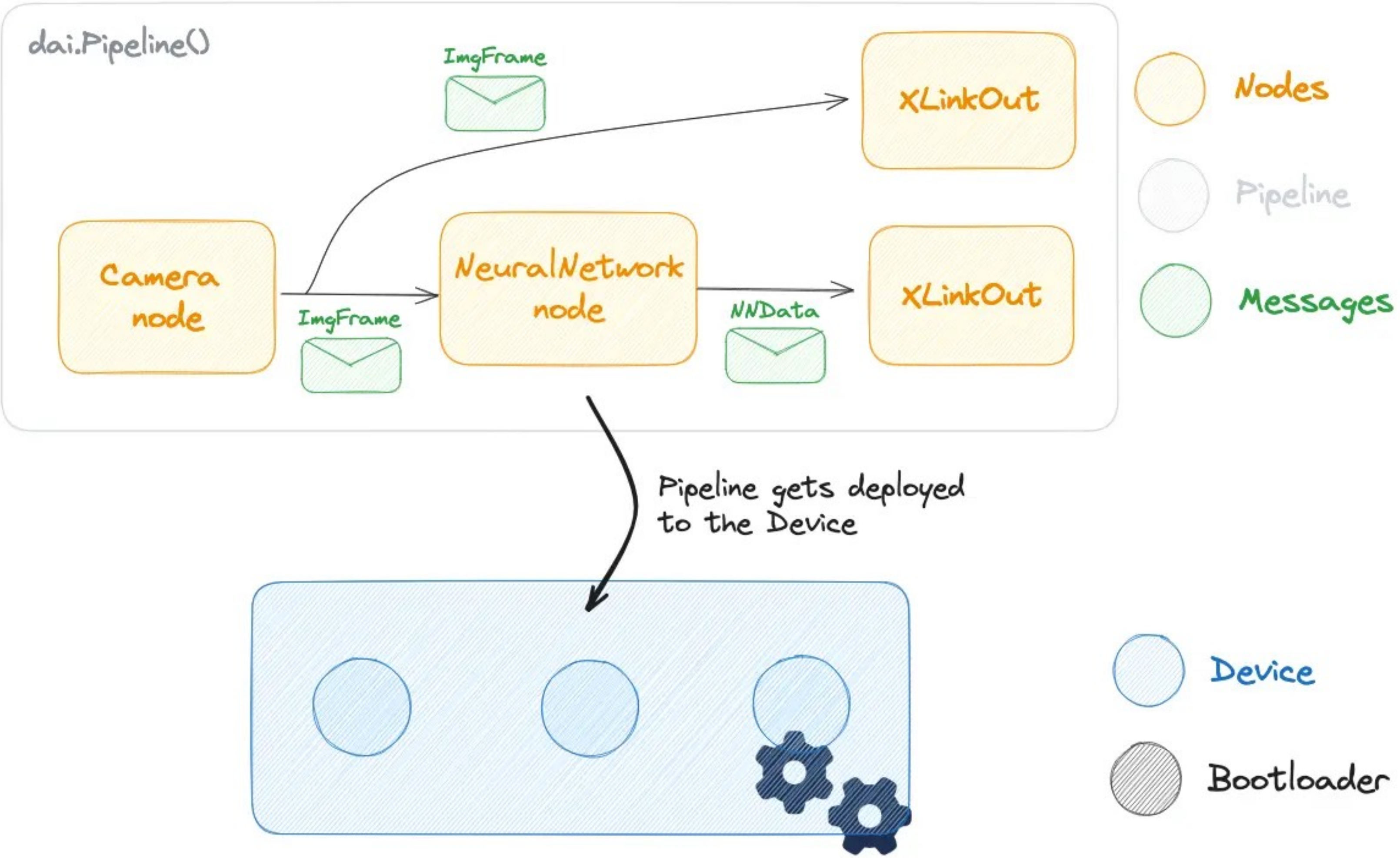
XLinkOut

This node sends data from the device to the host via XLink.

HTTPS://WWW.LUXONIS.COM

DEPLOYING CUSTOM MODELS

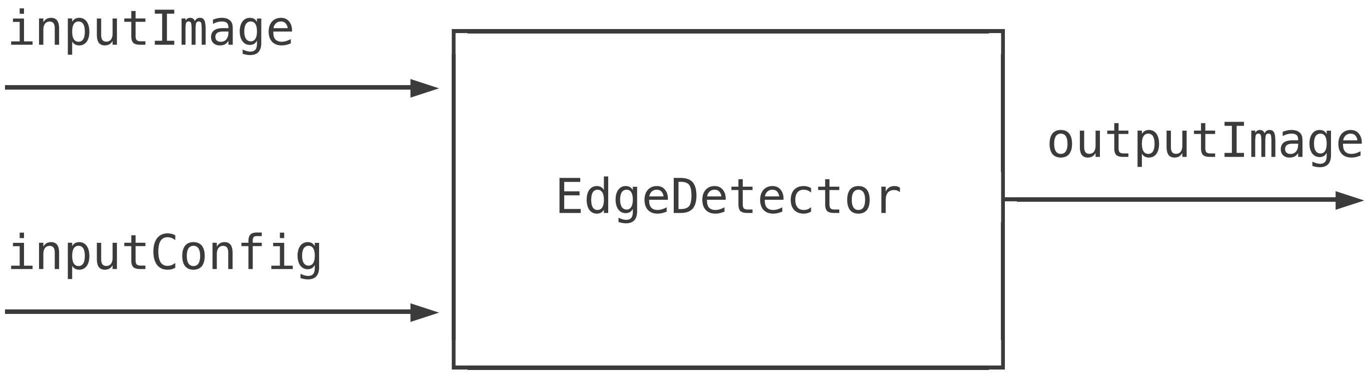
What is DepthAI SDK



DEPLOYING CUSTOM MODELS

Nodes

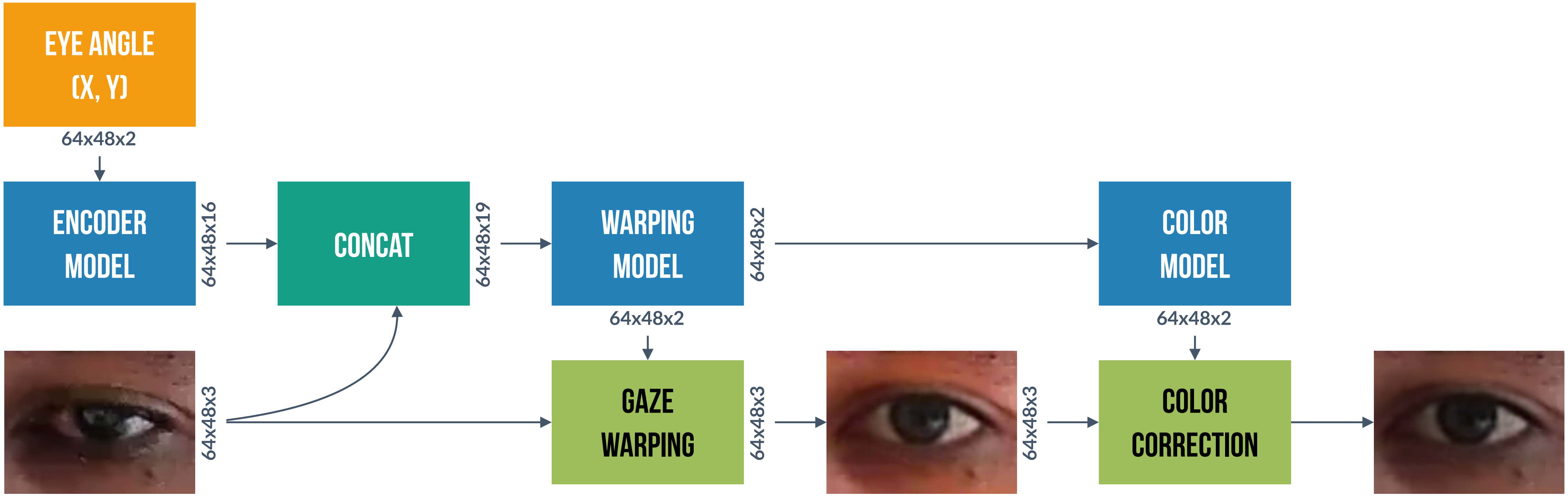
Nodes serve as a building block when populating the Pipeline. They offer specific functionality on the DepthAI, along with a set of configurable properties and inputs/outputs.



EdgeDetector node has 2 inputs and 1 output

DEPLOYING CUSTOM MODELS

I must implement the Luxonis OAK-1 Max’s pipeline similar to the JECModel architecture



- 🟦 ML Models
- 🟩 PyTorch Methods
- 🟩 CV Algorithm
- 🟧 Input Data

QUESTIONS & ANSWERS

THANKYOU!