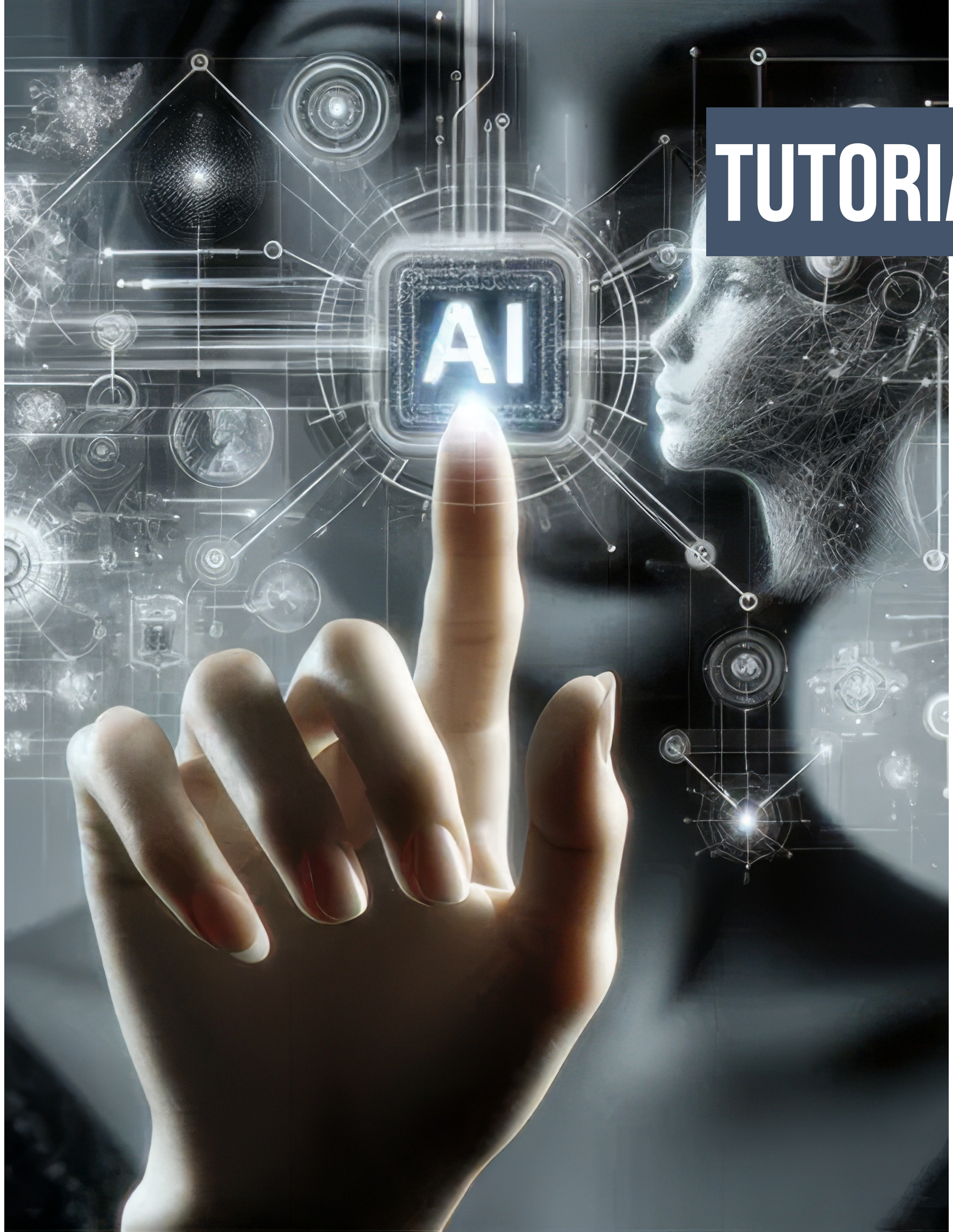
EDGE AI IN ACTION: PRACTICAL APPROACHES TO DEVELOPING AND DEPLOYING OPTIMIZED MODELS

CVPR 2024 Tutorial

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

Seattle, WA, USA



TUTORIAL AGENDA

1 Introduction to Edge AI

2 Model Development for Edge AI

3 Model Deployment for Edge AI

4 Multi-Modal AI for Edge AI

5 Closing Remarks and Joint Q&A

WHO ARE WE?

Organizers and Speakers

FABRICIO BATISTA NARCIZO
Scientist Researcher

fbnarcizo@jabra.com
Jabra / ITU

ELIZABETE MUNZLINGER
Industrial Ph.D. Student

munzlinger@itu.dk
Jabra / ITU

ANUJ DUTT
ML Engineer

adutt@adobe.com
Adobe

HTTP://FABRICIONARCIZO.GITHUB.IO/CVPR2024-EDGE-AI-IN-ACTION

WHO ARE WE?
Organizers and Speakers

SHAN AHMED SHAFFI
AI/Cloud Engineer

shaahmed@gnhearing.com
GN Hearing A/S

SAI NARSI REDDY DONTHI REDDY
AI/ML Researcher

sdreddy@jabra.com
Jabra

HTTP://FABRICIONARCIZO.GITHUB.IO/CVPR2024-EDGE-AI-IN-ACTION

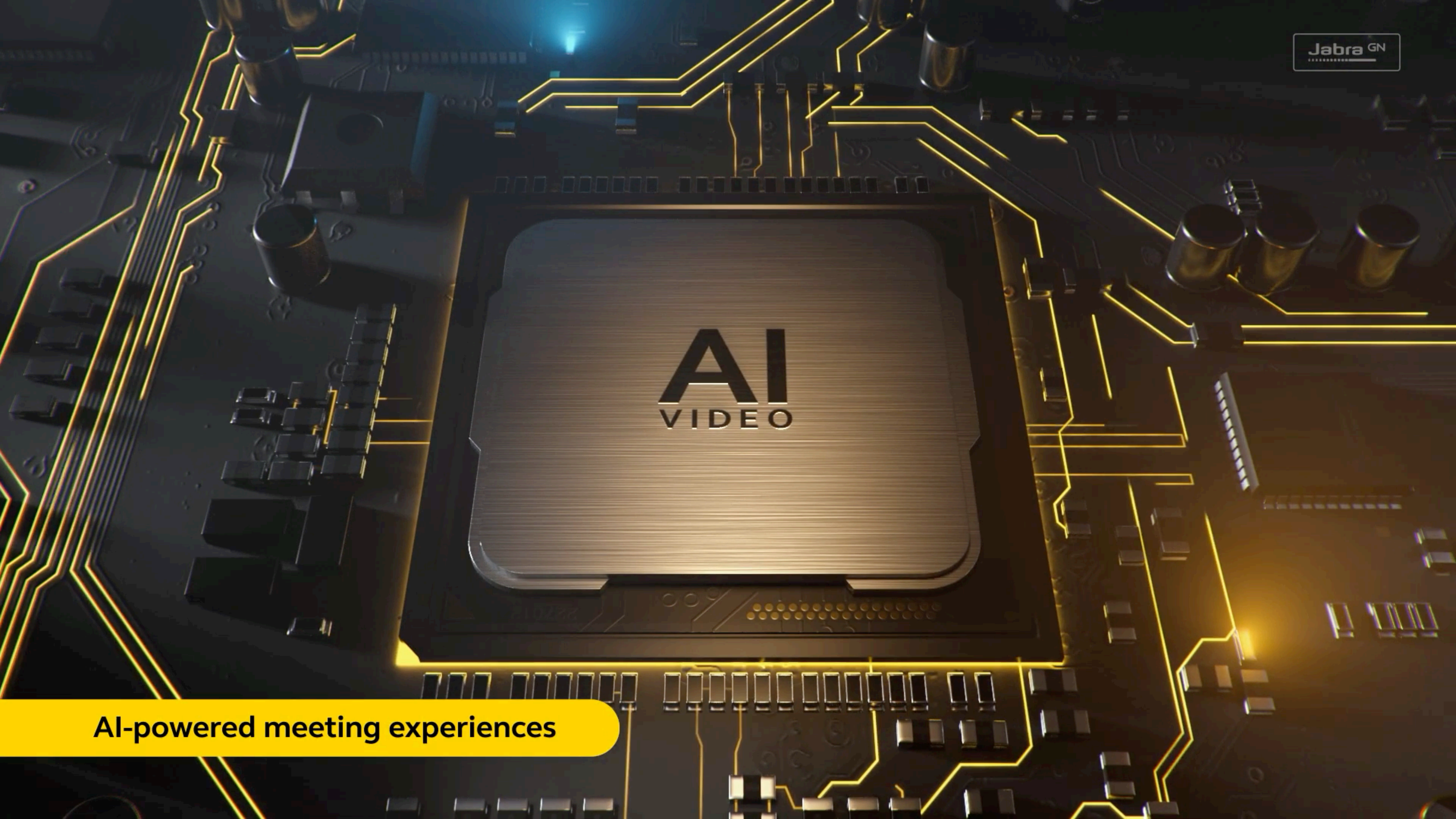
LEARNING OUTCOMES FOR THE TUTORIAL

Step 01
UNDERSTAND THE FUNDAMENTALS OF EDGE AI
Gain a solid understanding of Edge AI, including its motivations, benefits, and key challenges.

Step 02
ACQUIRE KNOWLEDGE IN MODEL DEVELOPMENT
Master techniques and tools for developing efficient AI models suitable for edge devices

Step 03
LEARN MODEL DEPLOYMENT STRATEGIES
Learn the processes involved in converting and optimizing AI models for deployment

Step 04
EXPERIENCE CASE STUDIES
See real-world applications and case studies of the use and benefits of Edge AI


Jabra GN
AI
VIDEO
AI-powered meeting experiences

THANK YOU!

INTRODUCTION TO EDGE AI

CVPR 2024 Tutorial

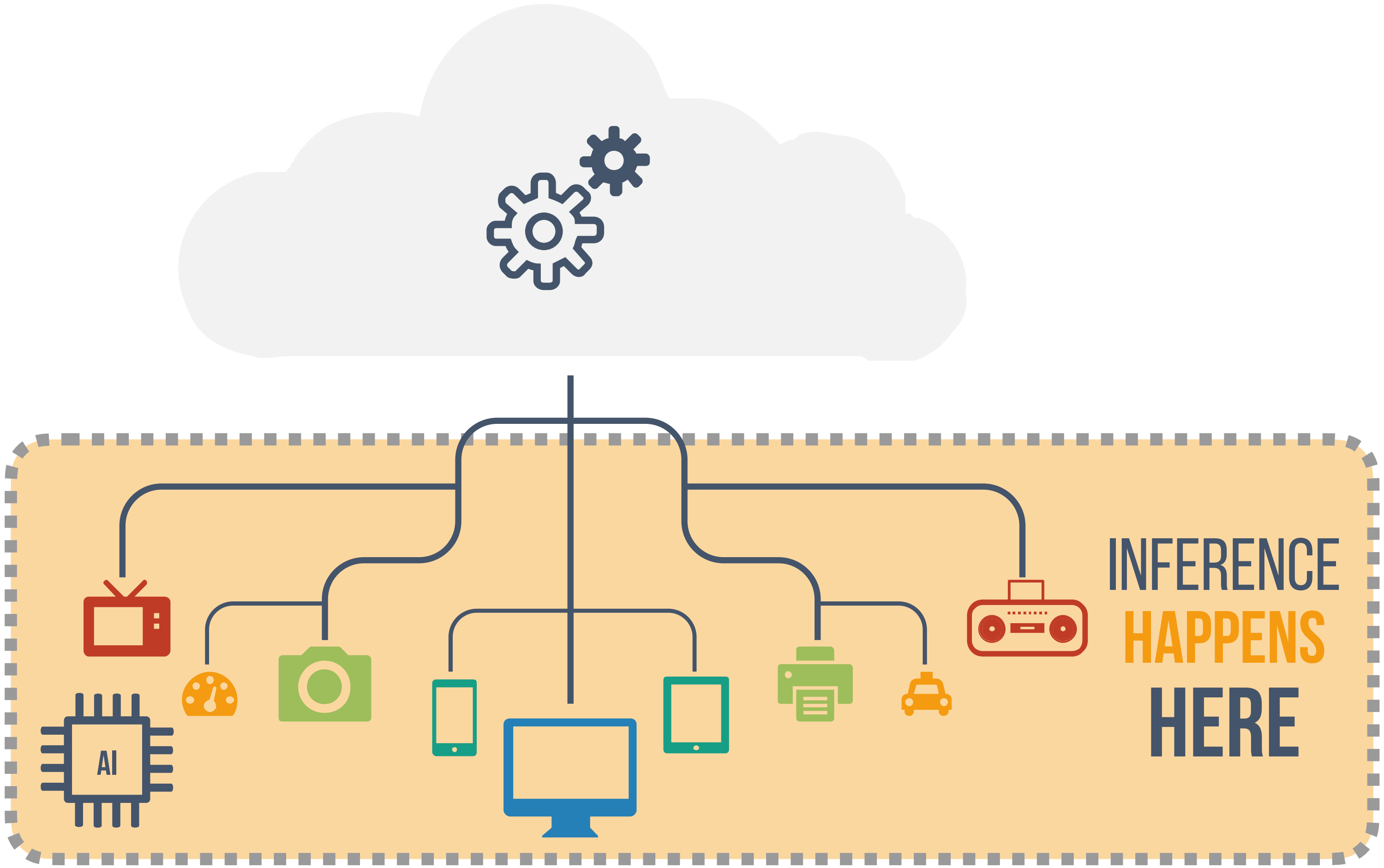
The IEEE/CVF Conference on Computer Vision and Pattern Recognition 2024

Seattle, WA, USA

INTRODUCTION:
WHAT
IS EDGE AI?

WHAT IS EDGE AI?

Introduction



1 **Low Latency**
Local processing significantly reduces response times and improves the performance of real-time applications.

2 **Reduced Bandwidth**
By processing data on the device itself, Edge AI decreases the volume of data transmitted over the network.

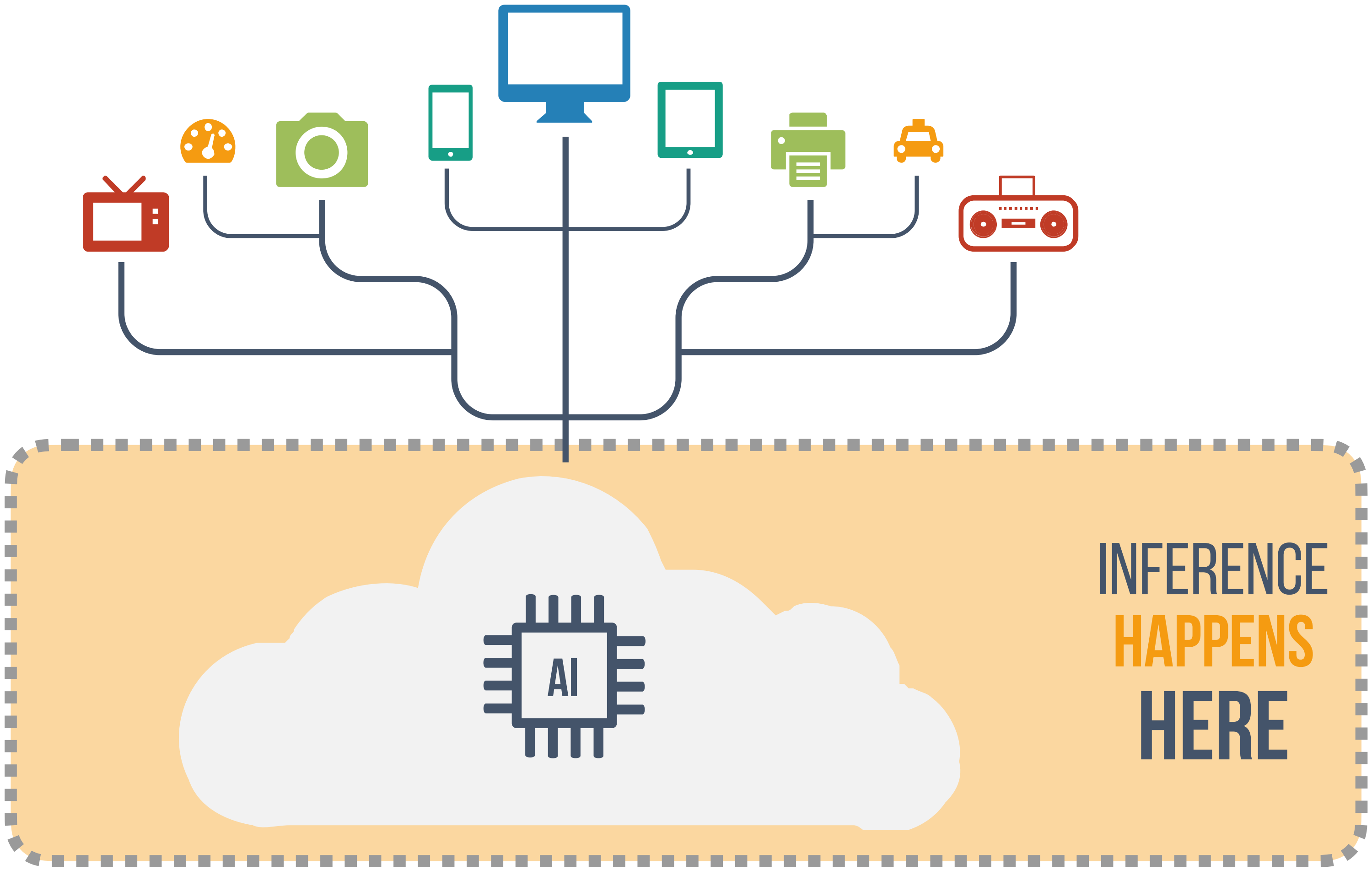
3 **Enhanced Privacy and Security**
Local data processing means sensitive information does not have to leave the device, enhancing data privacy.

4 **Operational Reliability**
Edge AI allows devices to operate uninterrupted, independently of the cloud or central servers.

5 **Energy Efficiency**
Processing data locally can be more energy-efficient than sending data to a cloud for analysis.

WHAT IS CLOUD AI?

Introduction



1. **Scalability**
   Cloud AI systems are highly scalable, allowing for adjustments based on the workload and user demand.

2. **Accessibility**
   Users can access these technologies from anywhere in the world, requiring only an internet connection.

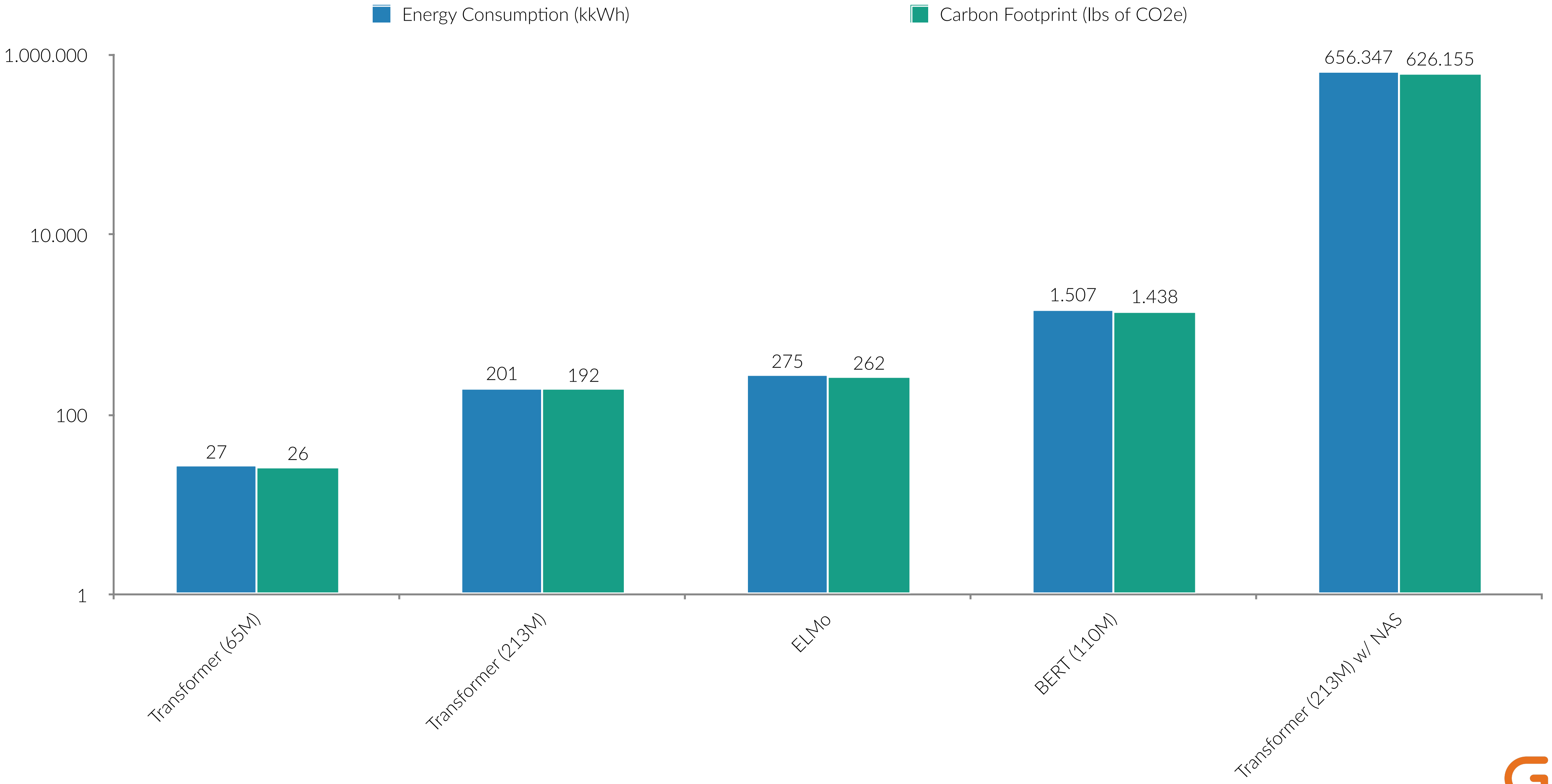
3. **Cost-Effectiveness**
   You can utilize AI tools and computing power on a pay-as-you-go basis, which helps manage costs effectively.

4. **Integration and Collaboration**
   The integration enables seamless data flow and collaboration across different platforms and teams.

5. **Continuous Improvements**
   Cloud AI services are maintained by providers who ensure that the AI models are continuously updated.

WHAT IS THE TRAINING CONSUMPTION?

Training a single AI model can emit as much carbon as five cars in their lifetimes

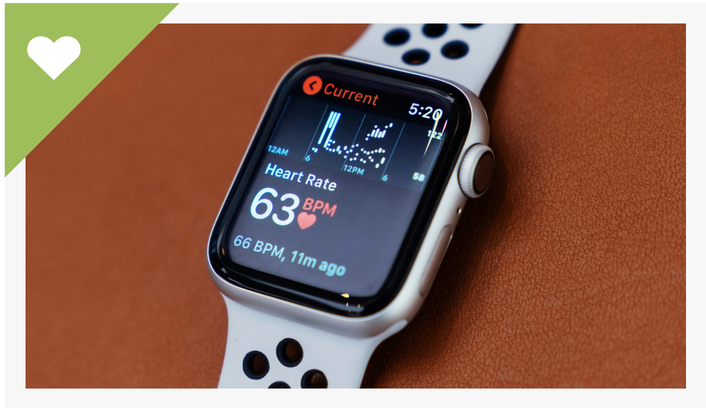


EDGE AI EXAMPLES

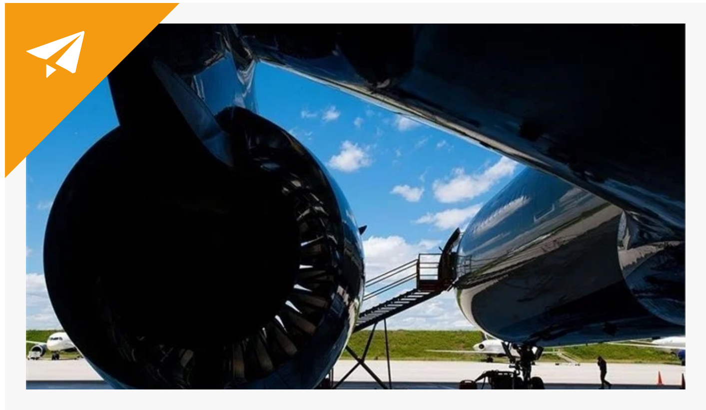
Example in different industries

Tesla Full Self Driving
By Tesla

See and Spray
By John Deere

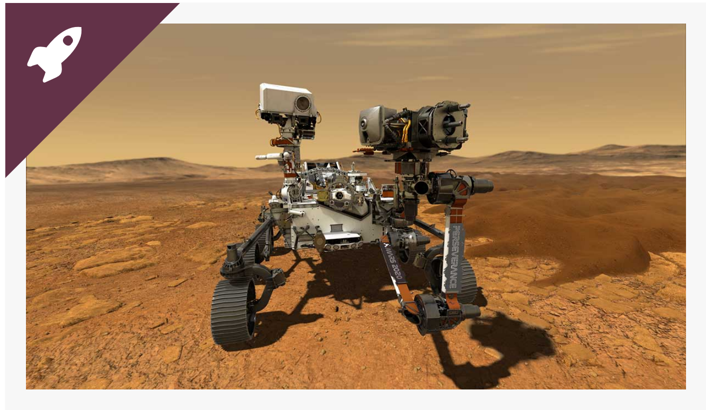


Apple Watch
By Apple



Delta Airlines Predictive Maintenance
By Delta Airlines

Jabra PanaCast 50
By Jabra



Perseverance Mars Rover
By NASA

SECURITY AND PRIVACY

Security & Privacy in Edge AI

As we integrate AI into devices at the edge of our networks, we must adopt robust measures to protect sensitive information and maintain user trust.

Data Encryption

Ensuring data remains encrypted during processing and storage.

Firmware Updates

Protecting devices from unauthorized access and ensuring they run trusted software.

Data Anonymization

Processing data in ways that prevent identification of individuals

Regulatory Compliance

Meeting standards such as GDPR by keeping data processing local

UNVEILING THE PILLARS OF EDGE AI

COMPONENTS OF EDGE AI

Specialized Hardware and Software

AI Models

Lightweight, optimized models for local inference.

Edge Devices

Incorporate many IoT gadgets, smartphones, cameras, and more.

Software Framework

TensorFlow Lite, PyTorch Mobile, ONNX Runtime.

Hardware Accelerators

Specialized chips (e.g., NVIDIA Jetson, Google Edge TPU).

EDGE AI HARDWARE

Examples of Hardware for Edge AI

Microcontrollers and Microprocessors
Basic computing units for simple AI tasks.

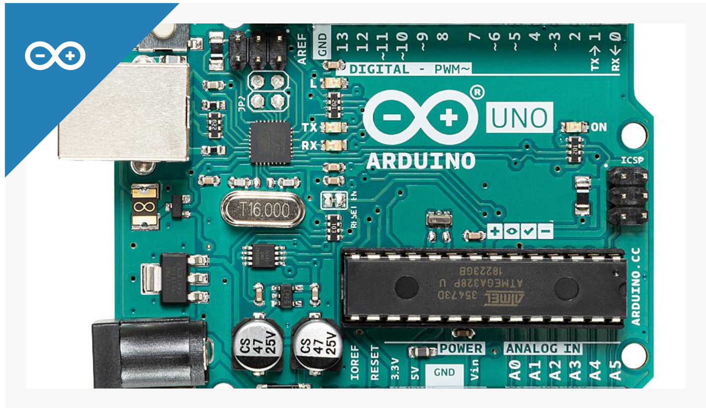
Edge Accelerators
Specialized hardware like NVIDIA Jetson, Google Edge TPU, and Intel Movidius.

Smart Sensors
Integrated sensors with built-in AI capabilities for real-time data processing.

Mobile Devices
Smartphones and tablets equipped with AI chips (e.g., Apple's A-series, Qualcomm's Snapdragon).

EDGE AI HARDWARE

Examples of Hardware for Edge AI



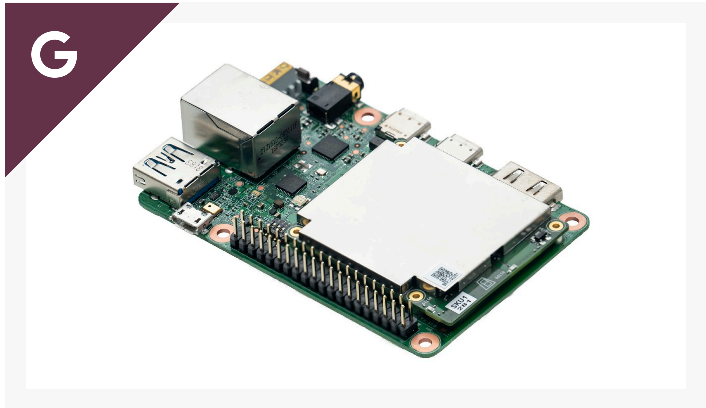
Arduino Microcontroller
By arduino.cc

Intel Neural Compute Stick 2
By Intel

BrainChip Akida
By BrainChip

Qualcomm QCS8250
By Qualcomm

NVIDIA Jetson
By NVIDIA



Google EdgeTPU
By Google

EDGE AI SOFTWARE

Frameworks for Edge AI

1 TensorFlow Lite
Lightweight version of TensorFlow optimized for mobile and edge devices.

2 TFLite Micro
Runs ML models on tiny, low-power devices like microcontrollers.

3 PyTorch Mobile
Enables deployment of PyTorch models on mobile platforms.

4 PyTorch ExecuTorch
Enables on-device inference capabilities across mobile and edge devices.

5 Intel OpenVINO
A toolkit designed to optimize ML and DL models for Intel hardware.

6 Qualcomm SNPE
It allows run DL models on Qualcomm Snapdragon mobile platforms.

7 ONNX Runtime
Cross-platform, high-performance scoring engine for ONNX models.

8 Edge Impulse
A platform for developing, optimizing, and deploying AI models to edge devices.

BRING IT ALL TOGETHER

Workflow for Edge AI Model Deployment

Step 01
01 MODEL DEVELOPMENT AND TRAINING (PYTORCH, TF, KERAS, JAX, ETC)

Trained model weights & model computational graph

Step 02
02 MODEL OPTIMIZATION (PRUNING, QUANTIZATION, KD, ETC)

Optimized model weight & model computational graph

Step 03
03 EXPORT TO IR (ONNX)

IR model weight & computational graph with optimizations.

Step 04
04 MODEL COMPILATION & BINARY GENERATION (.BIN, .DLC, .XML, .BLOB, ETC)

Compiled binary file ready for deployment on edge devices.

Step 05
05 MODEL INTEGRATION & INFERENCE (.C, .H, .CPP, .HPP)

NAVIGATING THE CHALLENGES AND PRIVACY LANDSCAPE OF EDGE AI

CHALLENGES IN EDGE AI DEPLOYMENT

Diagram

Latency	Efficiency	Availability	Privacy
ADAS Drones	Smart Glasses AR/VR	Smart Camera Sensors	Home Assistants
CPU	GPU	DSP (VPU)	DLA

Flexibility

High → Low

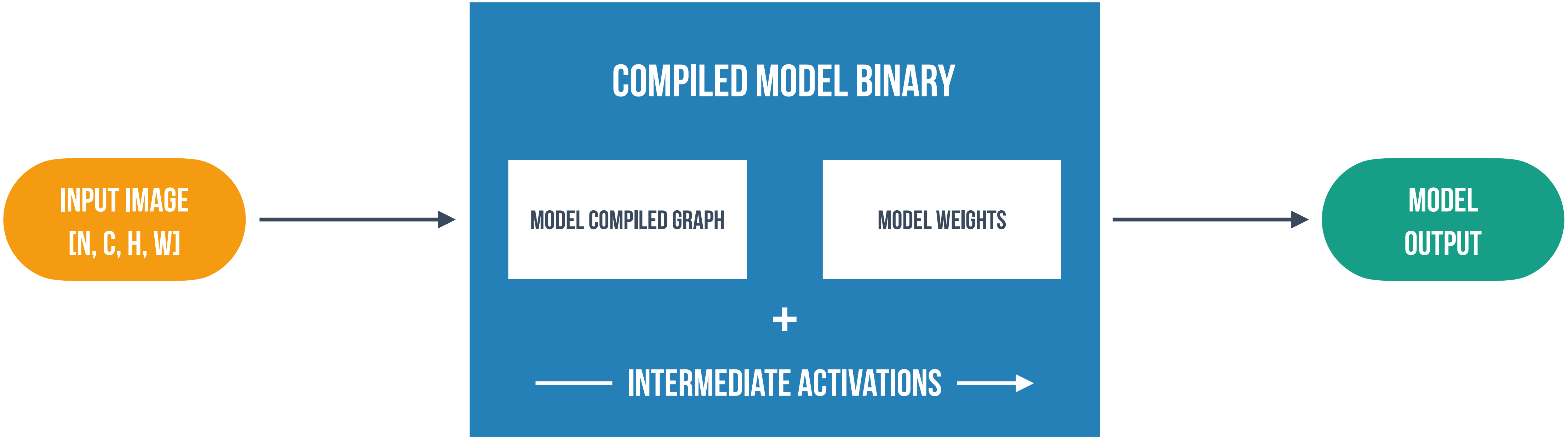
Efficiency

Low ← High

INTRODUCTION TO MODEL DEPLOYMENT FOR EDGE AI

MODEL INFERENCE MEMORY BANDWIDTH

Per Frame Model Inference Memory Bandwidth Components



THE ROOFLINE MODEL

Operational Intensity (ops/byte)

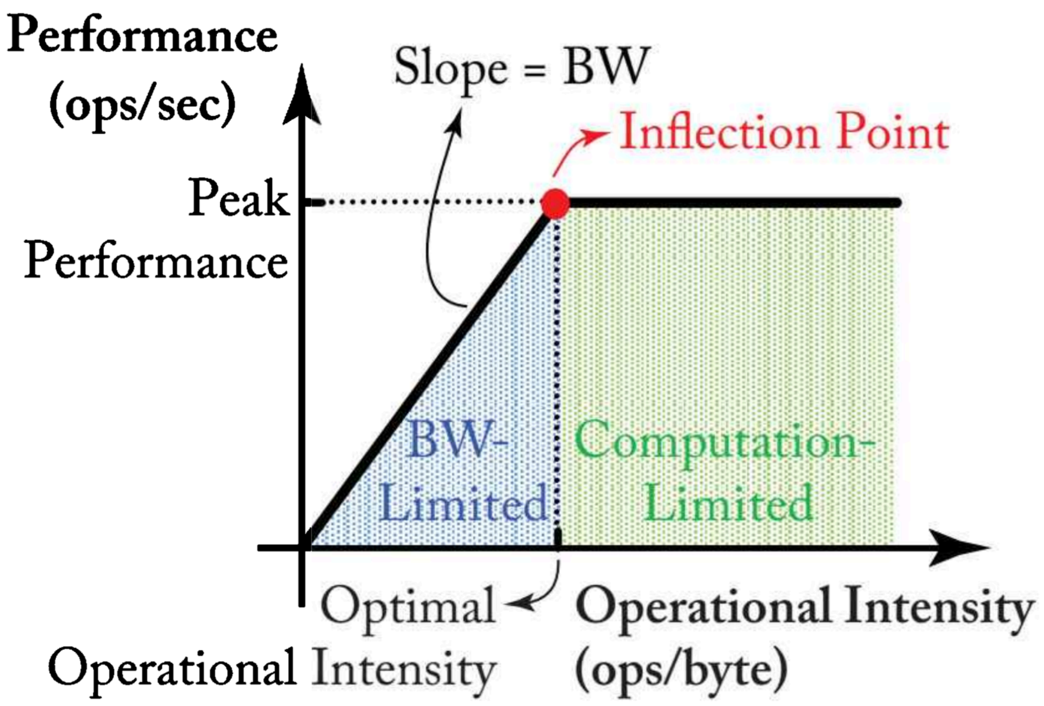
THE ROOFLINE MODEL IS A GRAPHICAL REPRESENTATION TO ILLUSTRATE AN ARCHITECTURE'S PERFORMANCE ACROSS DIFFERENT LEVELS OF OPERATIONAL INTENSITY.

Operational Intensity

How computation-heavy an operation is relative to data movement.

Higher Operational Intensity

More computations are performed for every byte fetched from memory.



THE ROOFLINE MODEL

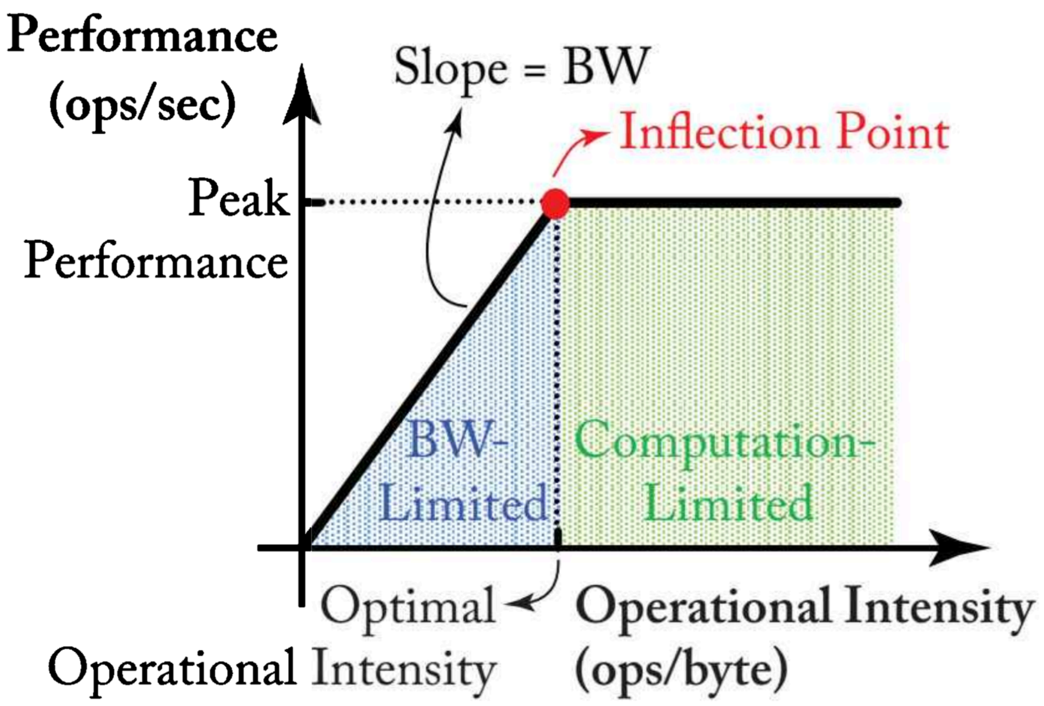
THE ROOFLINE MODEL

Performance (ops/sec)

THE ROOFLINE MODEL IS A GRAPHICAL REPRESENTATION TO ILLUSTRATE AN ARCHITECTURE'S PERFORMANCE ACROSS DIFFERENT LEVELS OF OPERATIONAL INTENSITY.

Performance
Represents peak performance of the hardware.

Peak
Maximum number of operations your hardware can handle per second.



THE ROOFLINE MODEL

QUESTIONS & ANSWERS

THANK YOU!

MODEL DEVELOPMENT FOR EDGE AI

CVPR 2024 Tutorial

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

Seattle, WA, USA

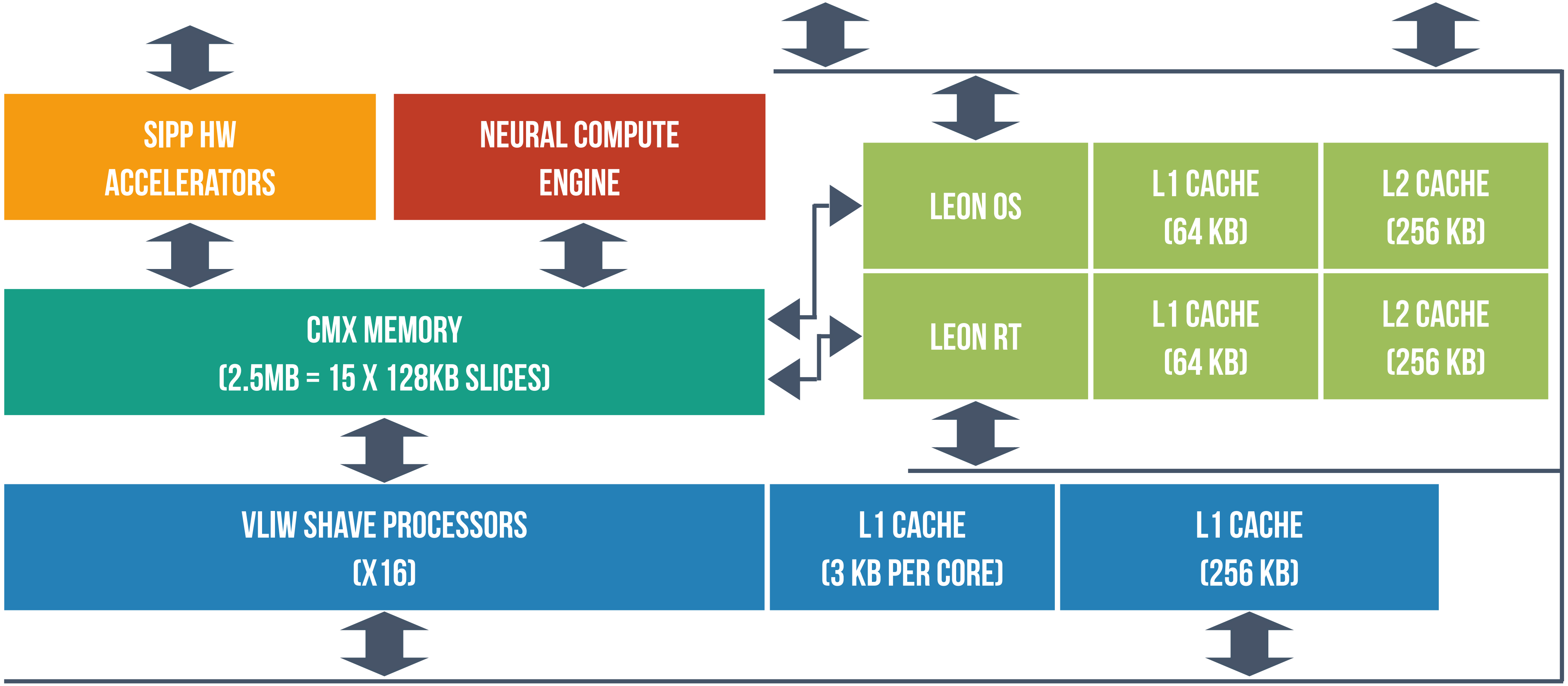
DESIGN A SEGMENTATION MODEL

Overview

1. *Hardware*, the model will be executed in a camera with Intel Movidius Myriad X VPU.
2. *Operating specifications*, the basic requirements to execute the model in an edge device in realtime.
3. *Model design*, the model architecture design considering the Edge device limitations.
4. *Dataset*, the synthetic and real data used to train the machine learning model.
5. *Training*, the process of teaching a machine learning model to make predictions or decisions.
6. *Results*, comparing our results with the models available in unified communication platforms.

INTEL MOVIDIUS MYRIAD X VPU HARDWARE

Petrongonas et al. (2021), ParalOS: A Scheduling & Memory Management Framework for Heterogeneous VPUs



OPERATING SPECIFICATIONS

Requirements

Coexist WITH

Video and audio processing pipeline and other DL models.

SHAVES Number

The body segmentation model must run on 1-2 SHAVES.

Body Detector

The model will segment only a single individual in the frame.

Region of Interest

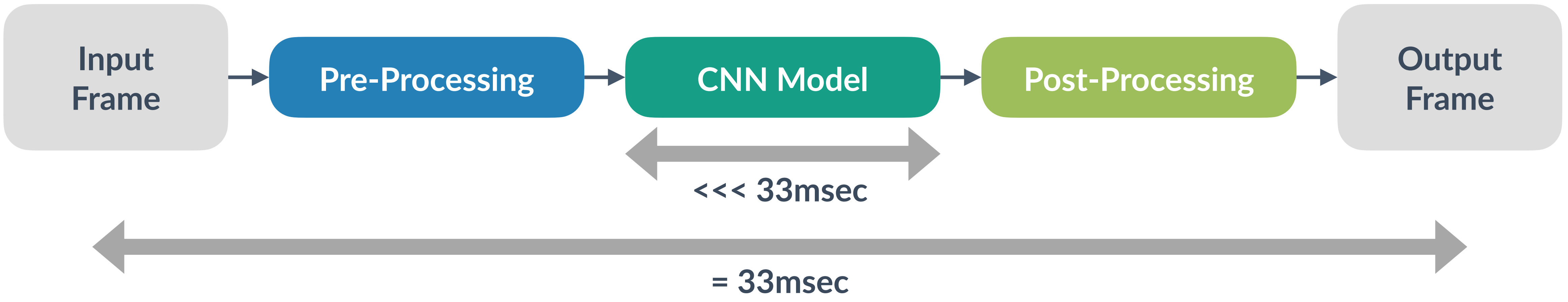
Model receives ROI crop from using the head-body detection model.

Execution Time

Whole segmentation pipeline must run in real-time (33msec).

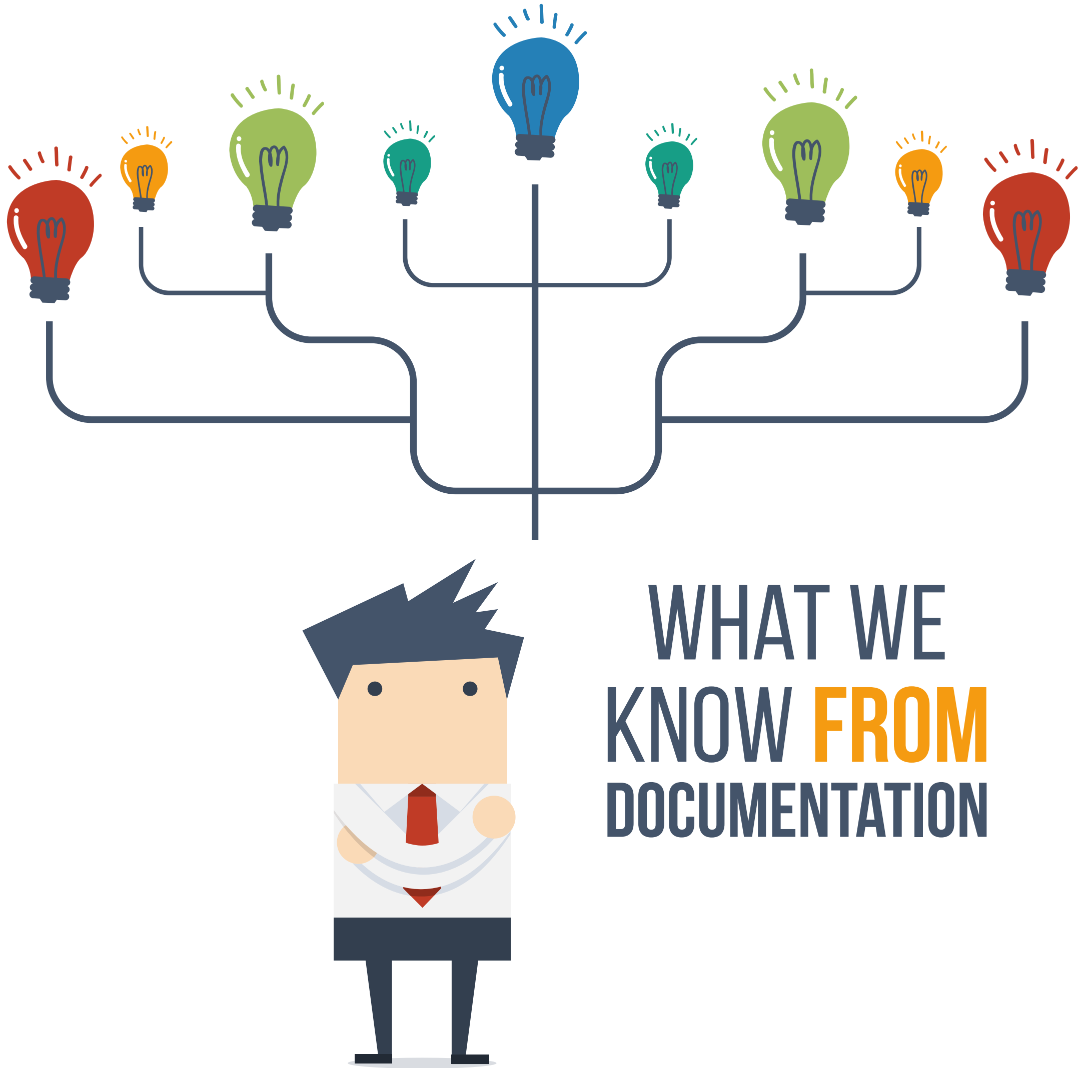
OPERATING SPECIFICATIONS

Requirements



MODEL DESIGNING

Understanding Hardware



1 **Data Type**
Only support 16bit Floating Point.

2 **NCE**
Limited number of operations directly supported by Neural Compute Engine.

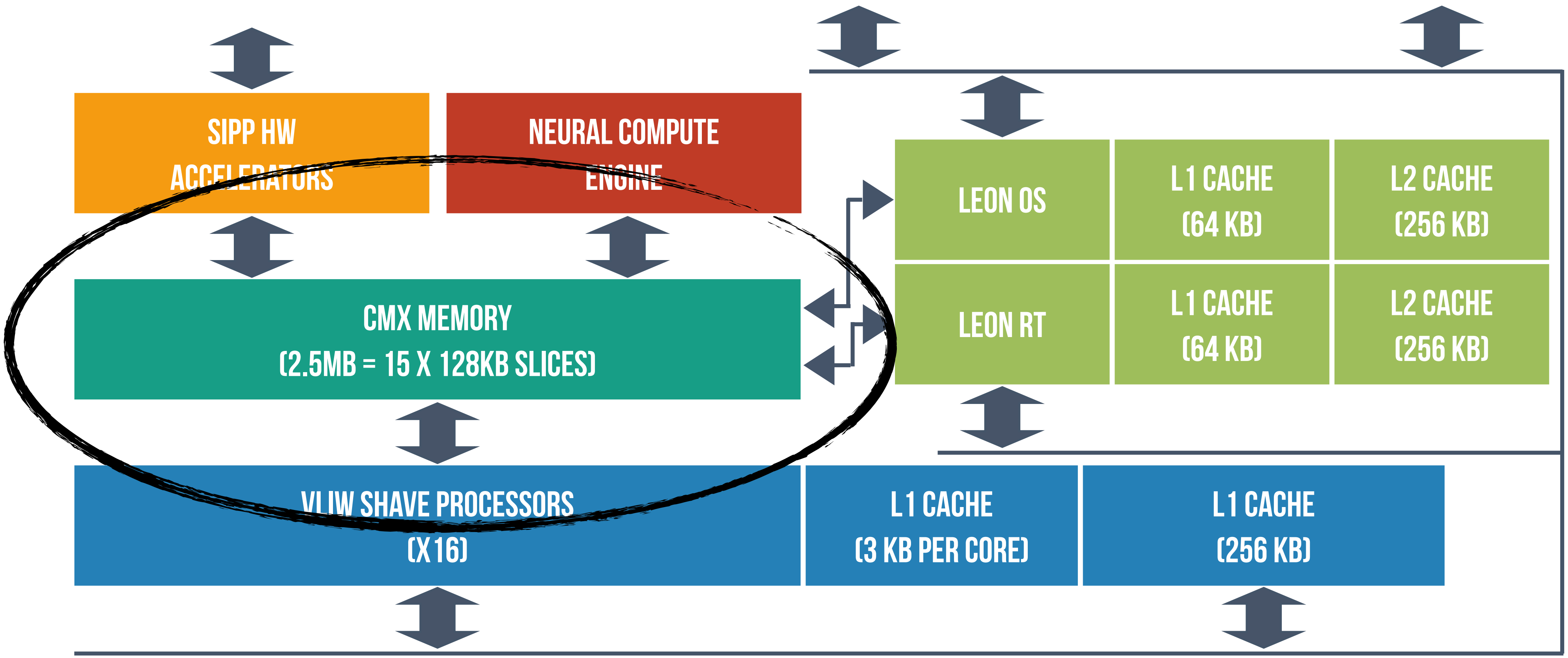
3 **Model Optimization**
Difficult to perform model pruning.

4 **Matrix Format**
Sparse matrix is not supported.

5 **Pruning Type**
Structured pruning is only available.

INTEL MOVIDIUS MYRIAD X VPU HARDWARE

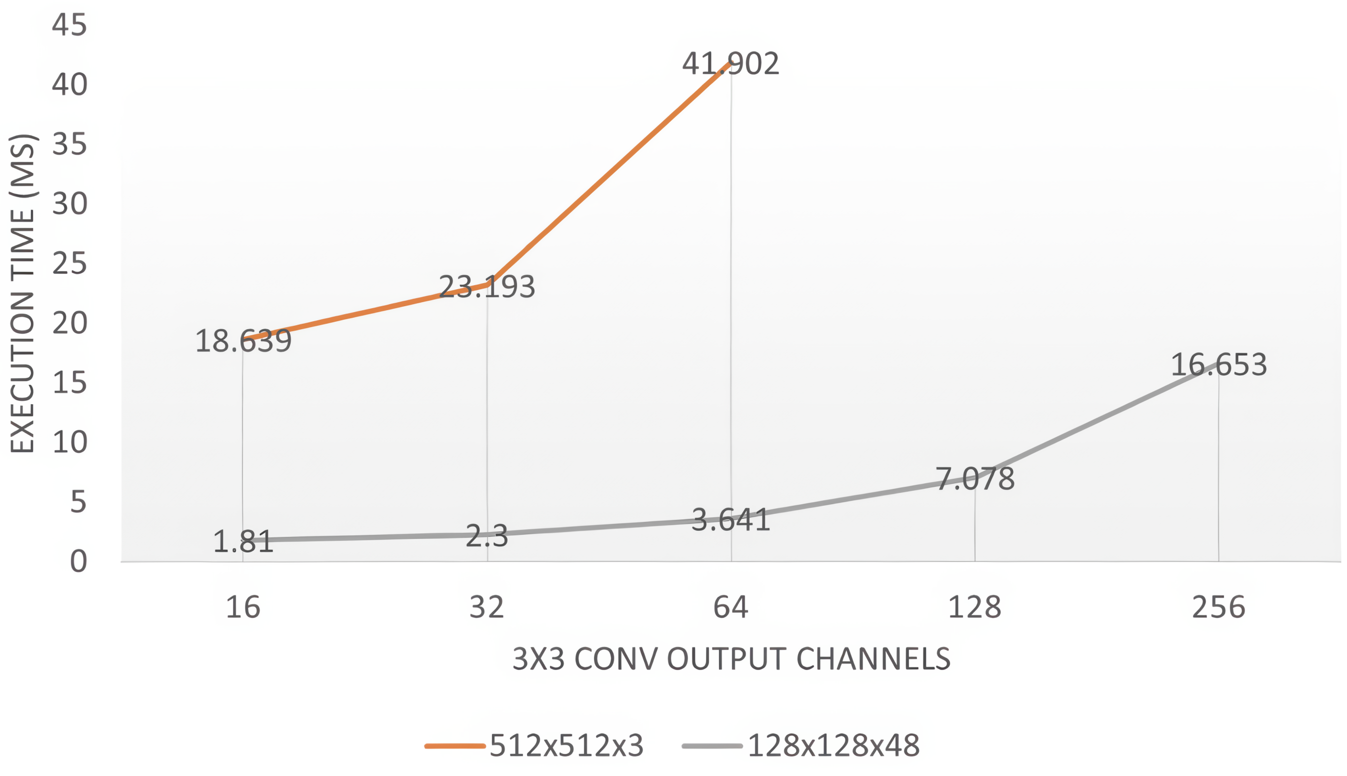
Petrongonas et al. (2021), ParalOS: A Scheduling & Memory Management Framework for Heterogeneous VPUs



MODEL DESIGNING
Understanding Hardware

"Problem: Perform computation at higher spatial resolutions."

WHAT WE FOUND FROM EXPERIMENTS



SOLUTION

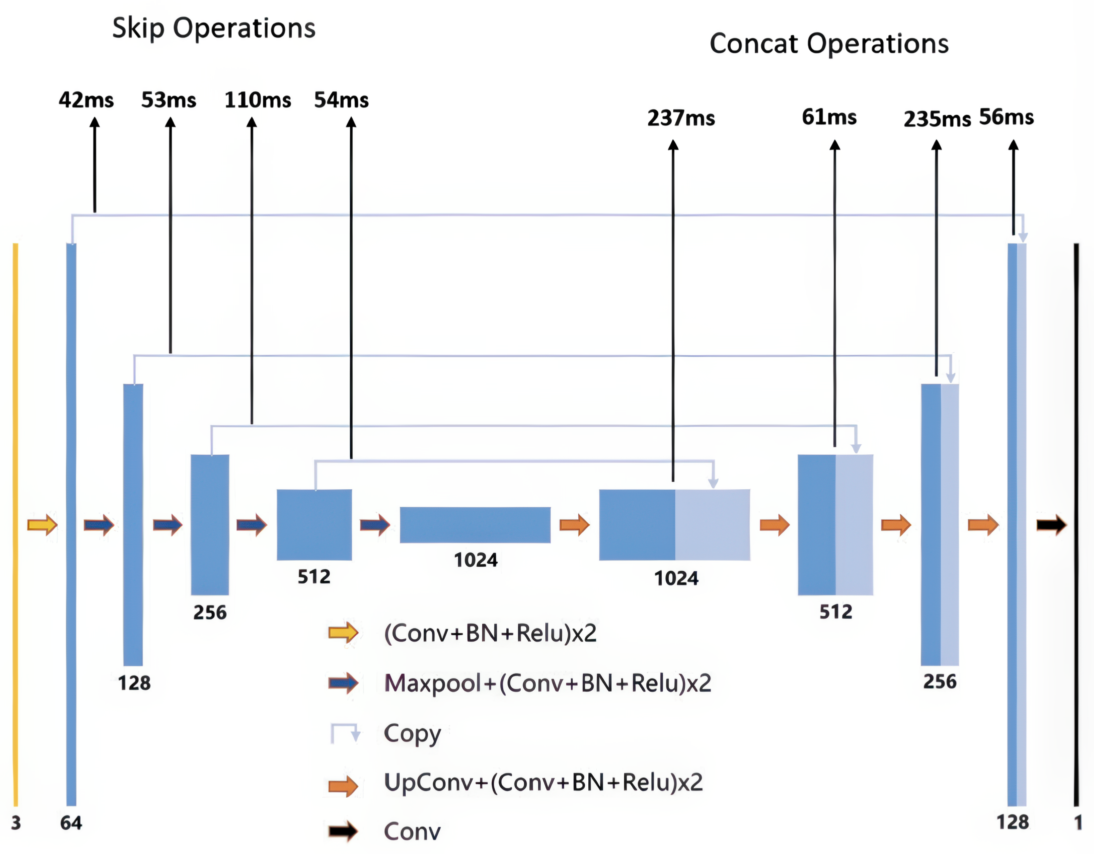
Use techniques such as pixel shuffling and large kernel convolution.

MODEL DESIGNING

Understanding Hardware

"Problem: Skip connection operations are costly if the feature size is large."

WHAT WE FOUND FROM EXPERIMENTS



SOLUTION

When a skip connection is used, reduce feature size.

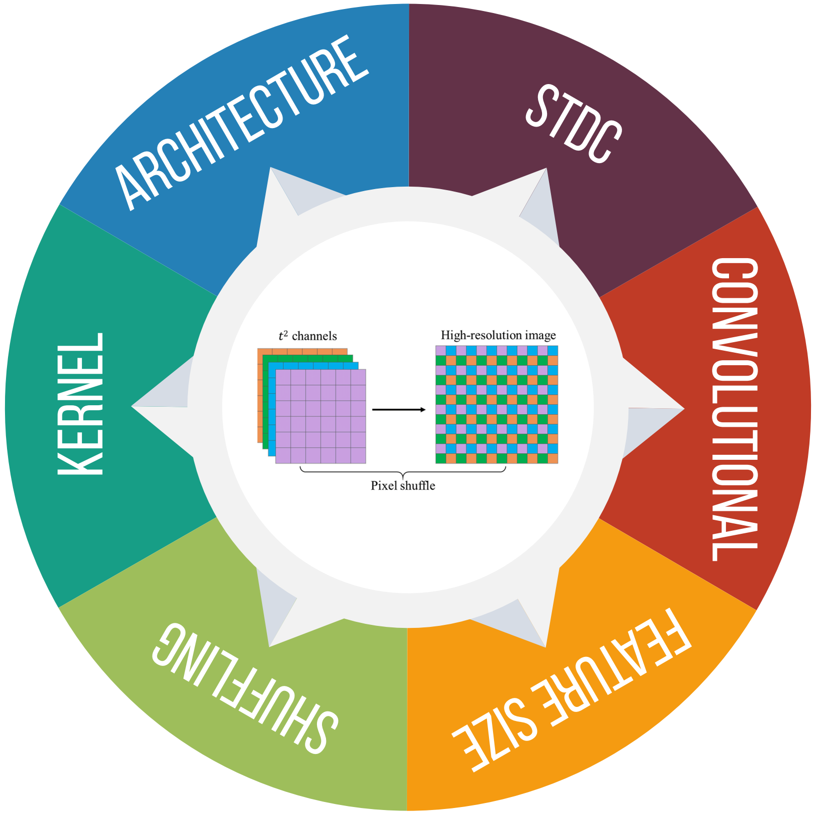
MODEL DESIGNING

Understanding Hardware

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

Kernel
Use large kernel convolutions with large stride at input.

Pixel Shuffling
Use pixel shuffling at the output.



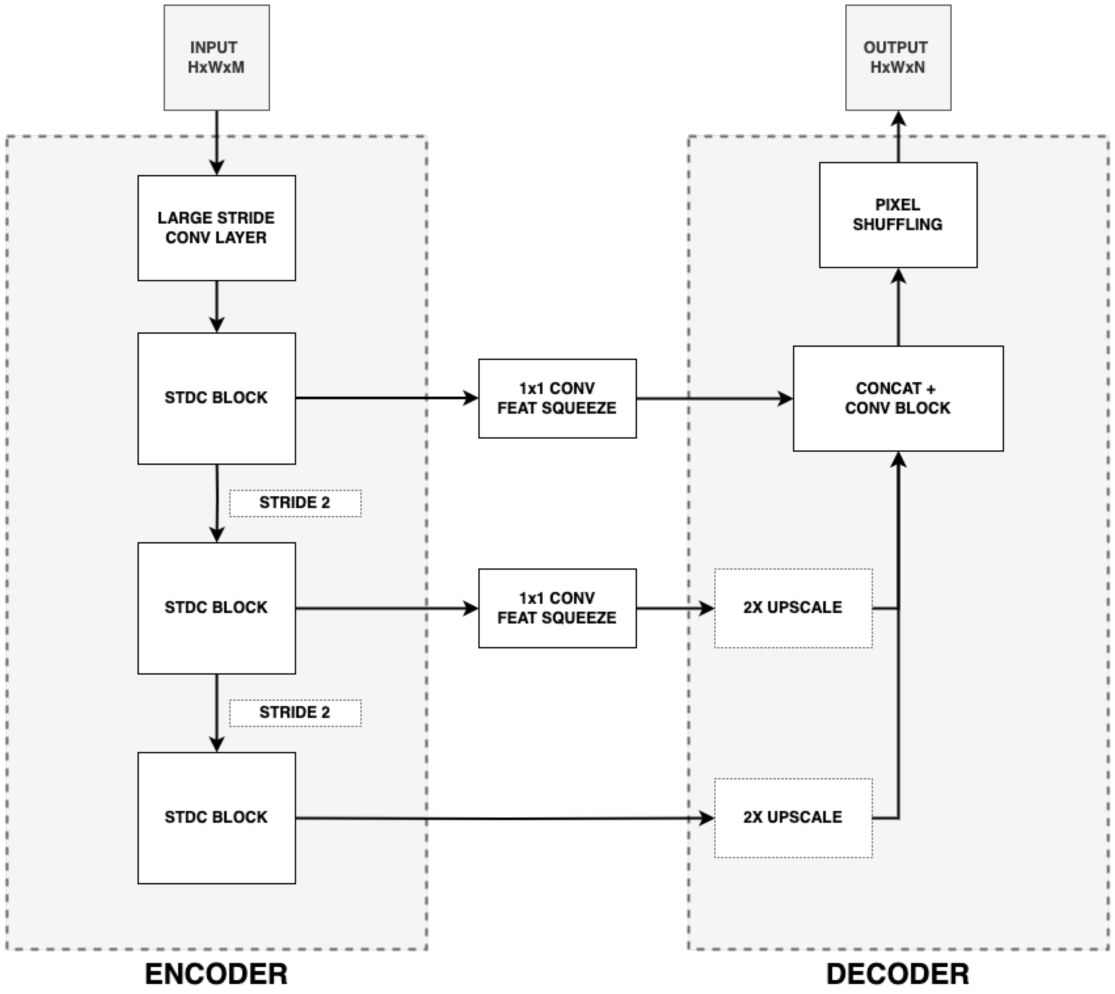
STDC (Convolution Block)
Use a modified Short-Term Dense Concatenate module (STDC).

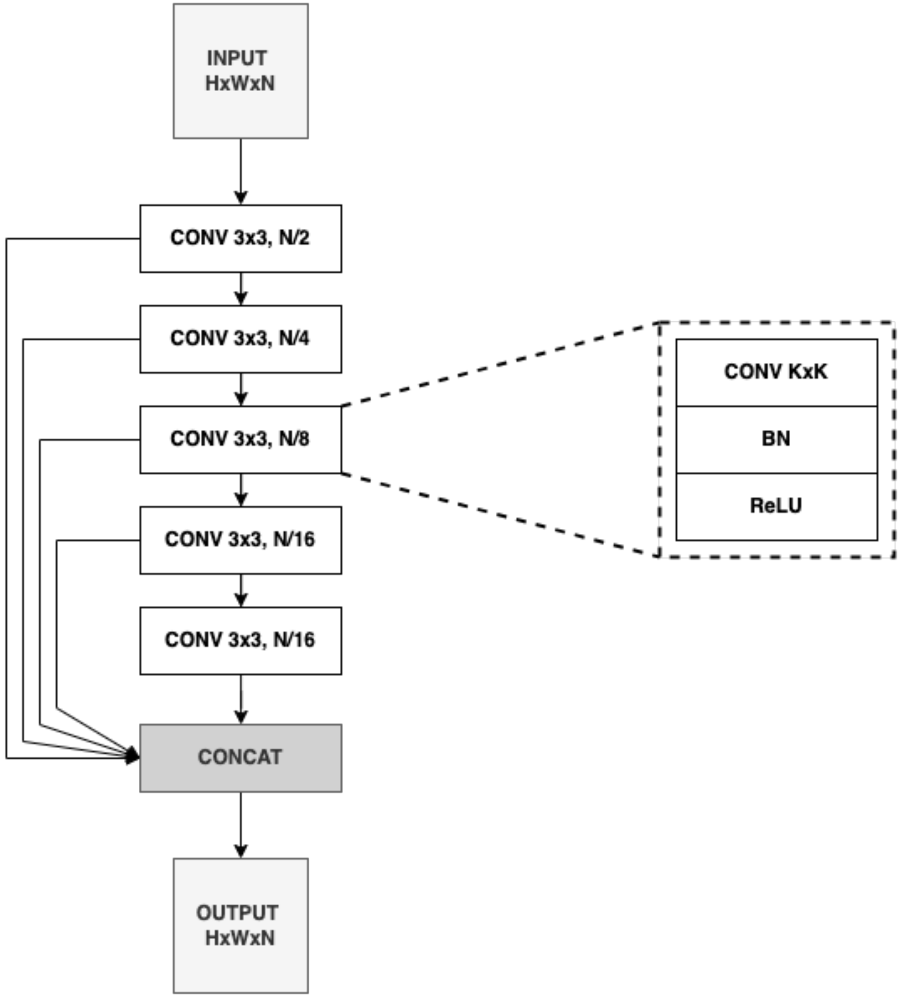
Convolutional
Use 1x1 CONV operations to reduce feature size for longer skip connections.

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

PROPOSED MODEL DESIGN

Mingyuan et al. (2021), Rethinking Bisenet for Real-Time Semantic Segmentation



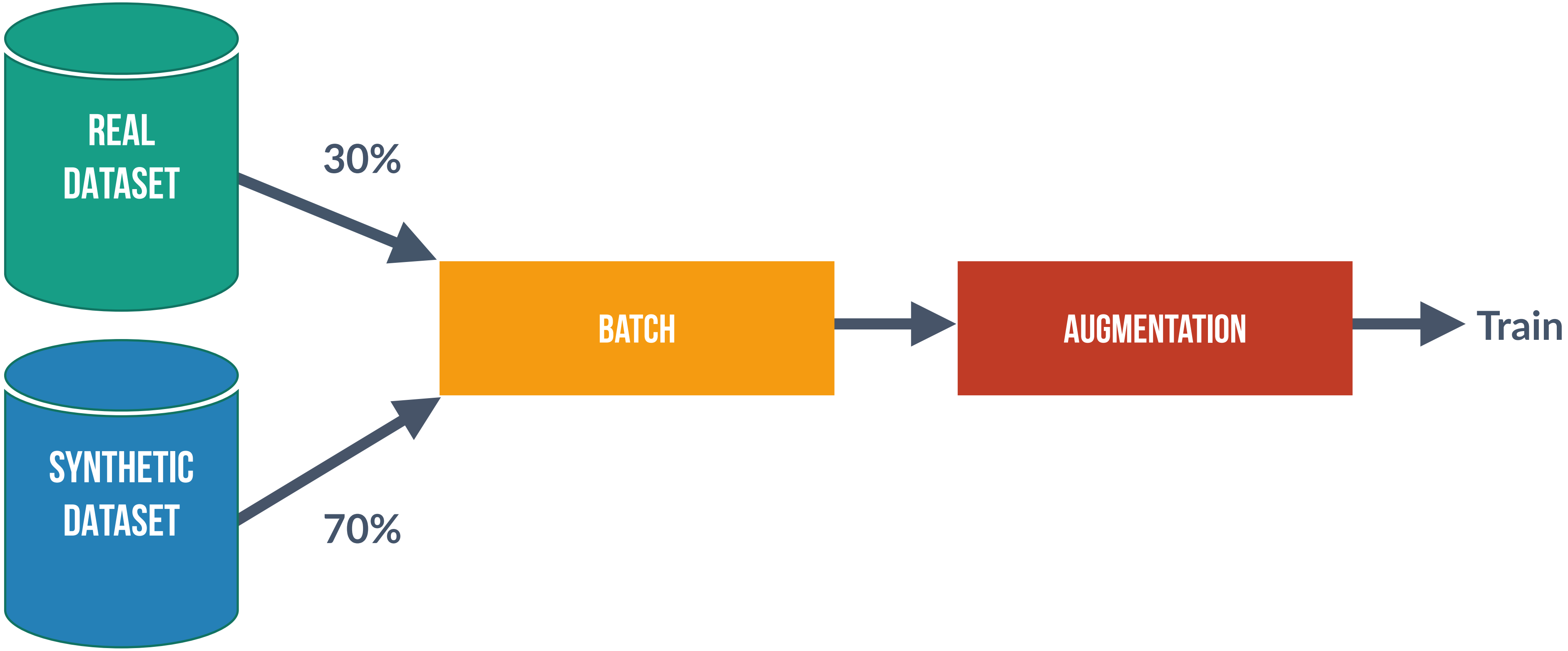



REAL IMAGES DATASET
Body Segmentation Datasets


SYNTHETIC IMAGES DATASET
Body Segmentation Datasets

DATASETS PIPELINE

Sample distribution for each bath



DATASETS AND TRAINING

Pipeline

DINO Pre-Training

It refers to a method of pre-training deep learning models using self-supervised learning techniques.

Training batch

For each batch, we feed 30% real data and 70% synthetic data.

Adam Optimizer

We used the 1e-4 learning rate in the Adam optimizer to update the model parameters.

Number of Batch

We set the number of Batch to 10K to train our segmentation model.

BODY SEGMENTATION RESULTS

The model runs at 18ms on hardware

BODY SEGMENTATION RESULTS

Comparison

Jabra PanaCast 20
On-Device Background Segmentation

Unified Communication
Platform 01

Unified Communication
Platform 02

QUESTIONS & ANSWERS

BREAK (30 MINUTES)

THANK YOU!

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

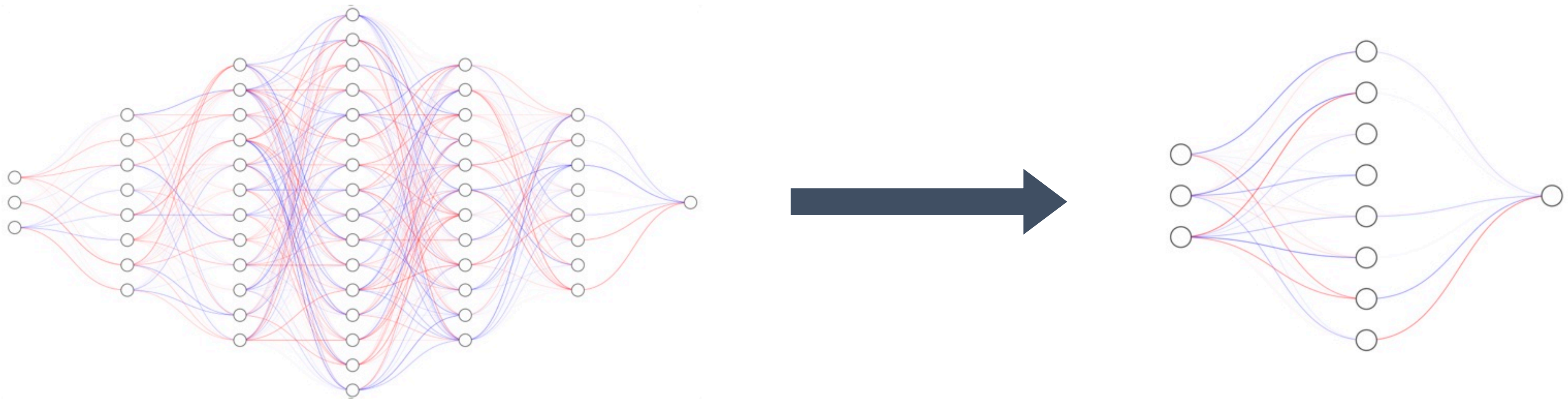
Objective 03

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

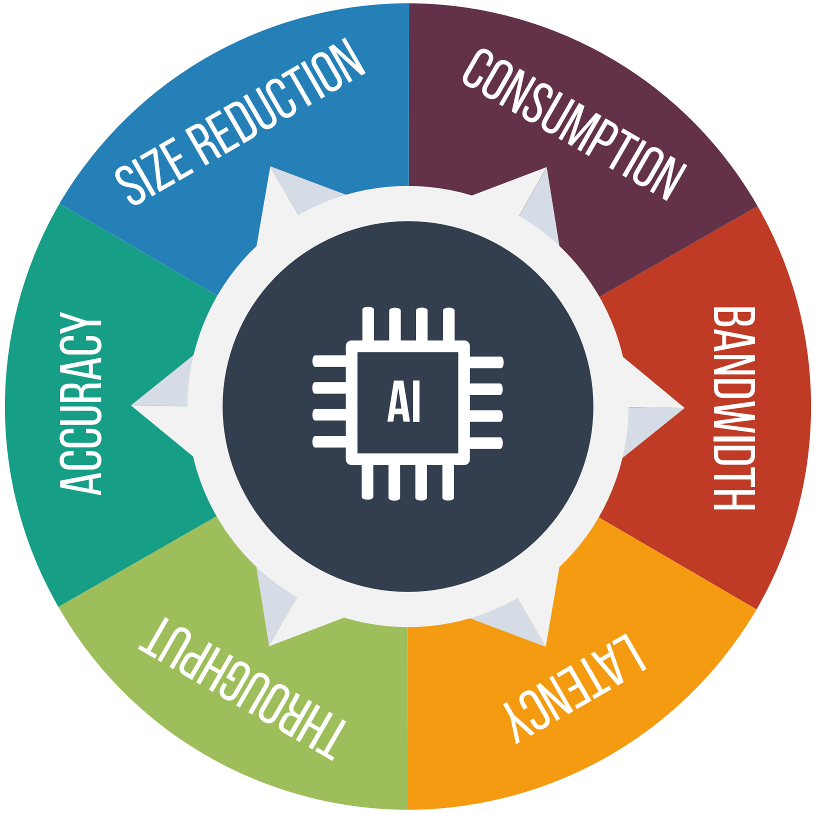
Deployment on Constrained Devices

UNDERSTANDING KEY METRICS

Model Deployment

Size Reduction
Shrinking model dimensions to fit edge device constraints.

Power & Energy Consumption
Evaluating the model's energy use and battery impact during operation.



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

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

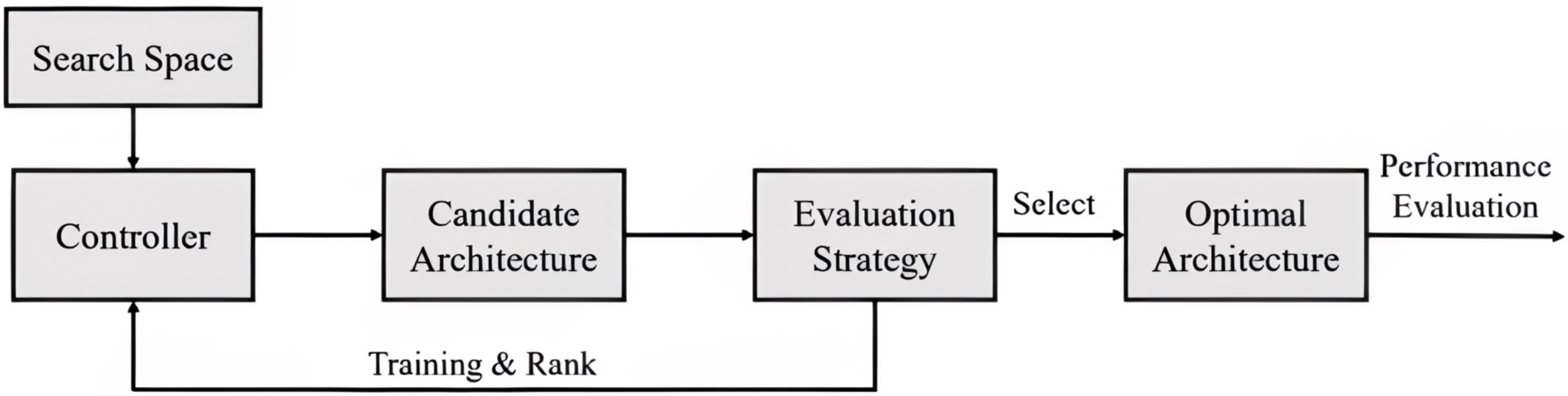
Throughput
Assessing the number of inferences a model can process per unit time.

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



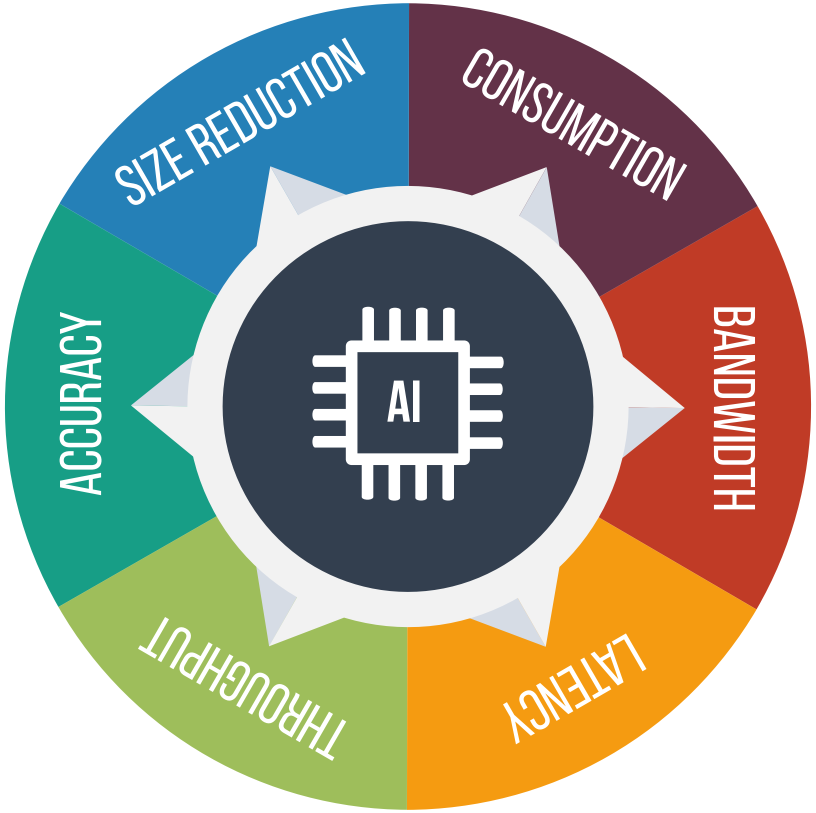
NEURAL ARCHITECTURE SEARCH

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

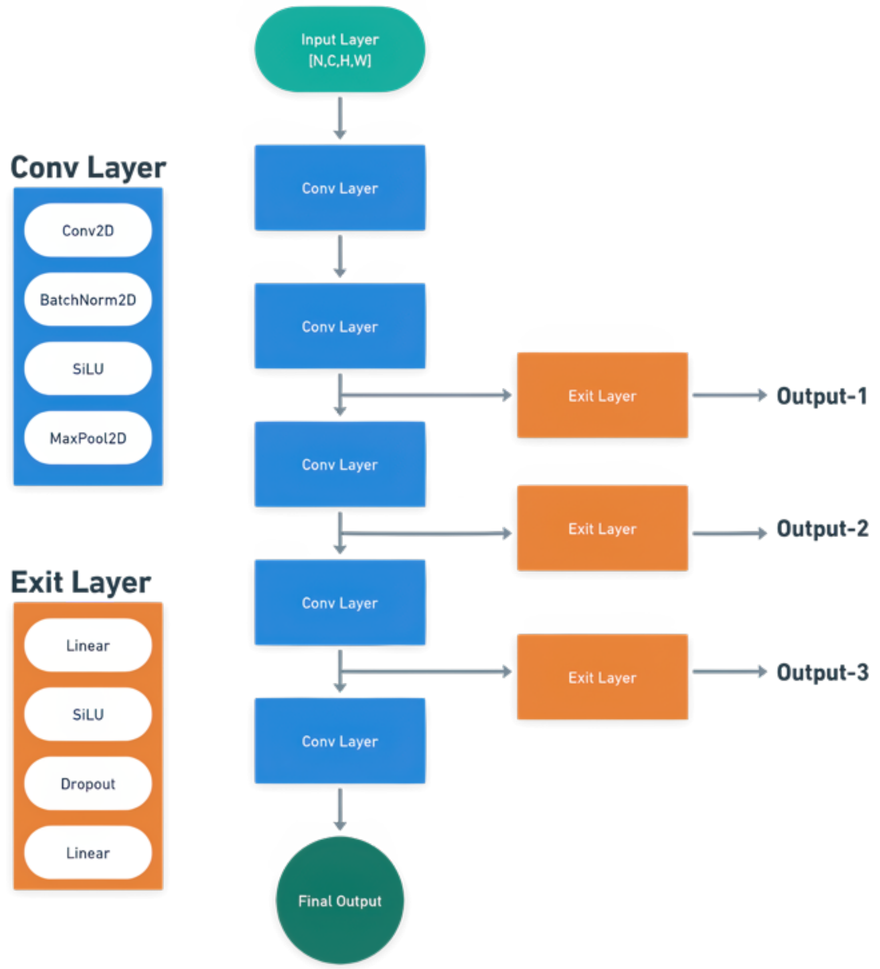
Early exits allow intermediate layers in a deep neural network (DNN) to produce predictions.

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

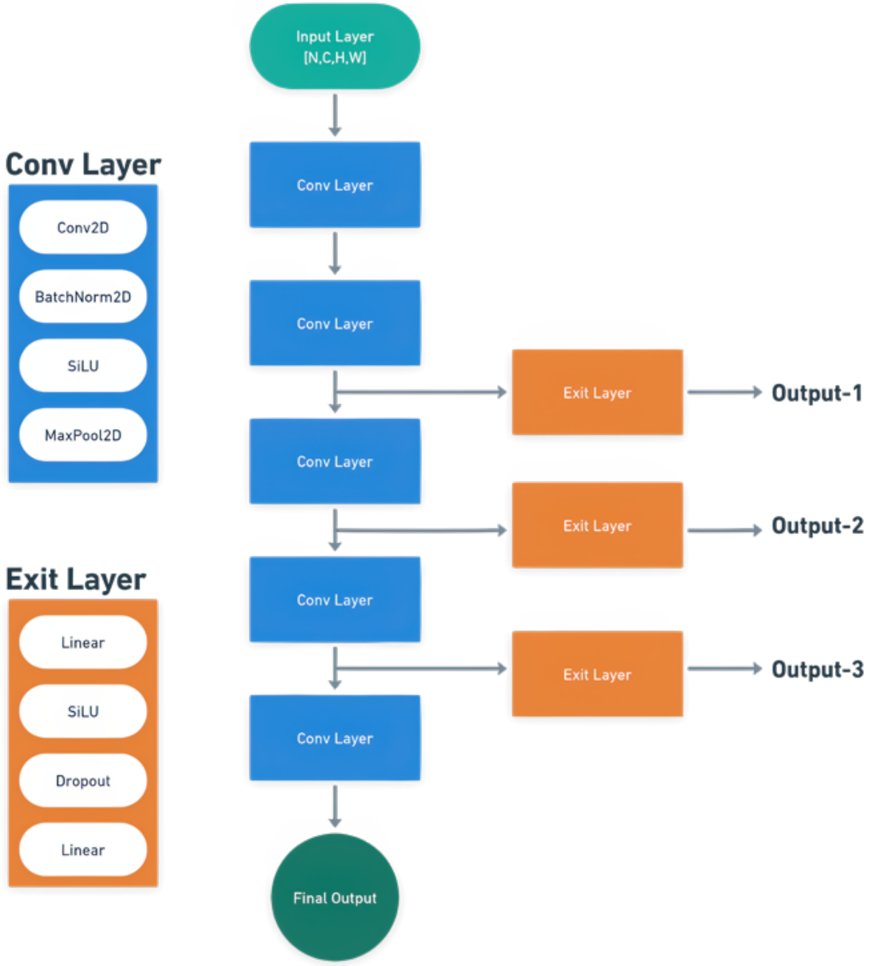
The *Early Exits* technique in model optimization involves adding intermediate outputs to a deep learning model.

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.



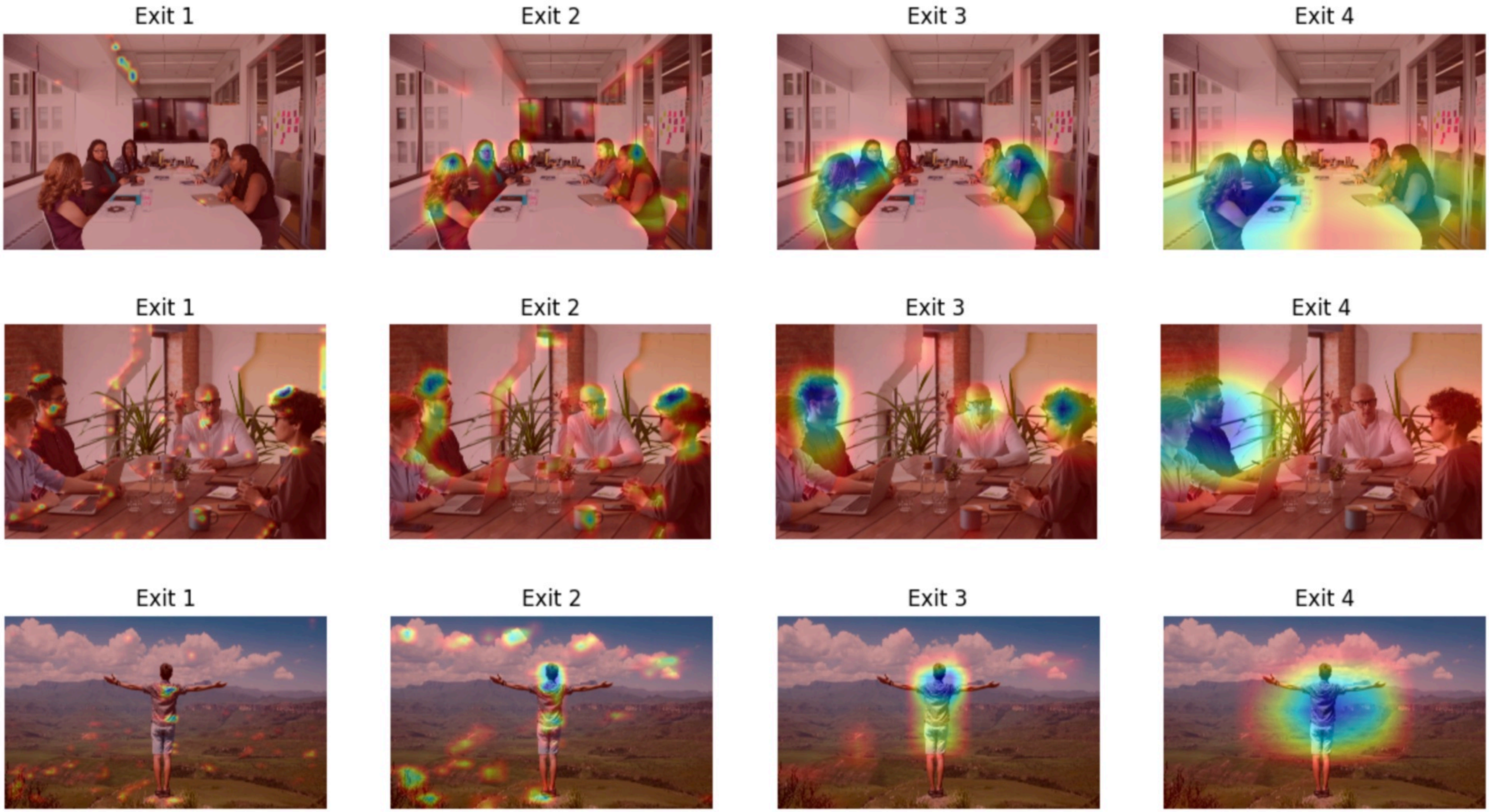

Predicted Label: human, Confidence: 1.000, Exit: 3
EARLY EXITS
Human Detector

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

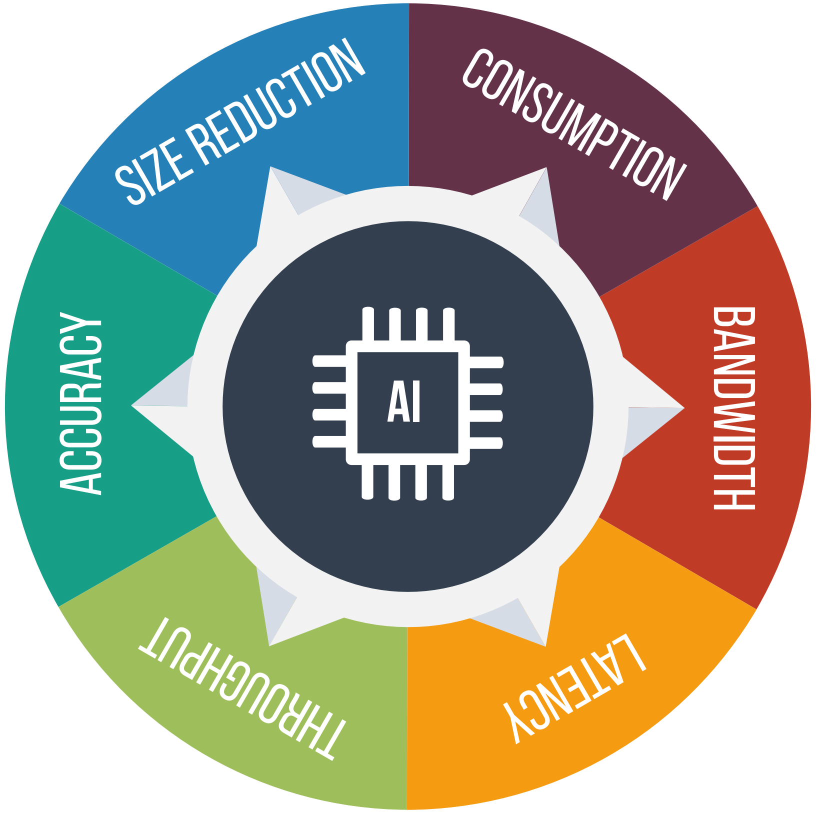

Exit 1
Exit 2
Exit 3
Exit 4
Exit 1
Exit 2
Exit 3
Exit 4
Exit 1
Exit 2
Exit 3
Exit 4

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

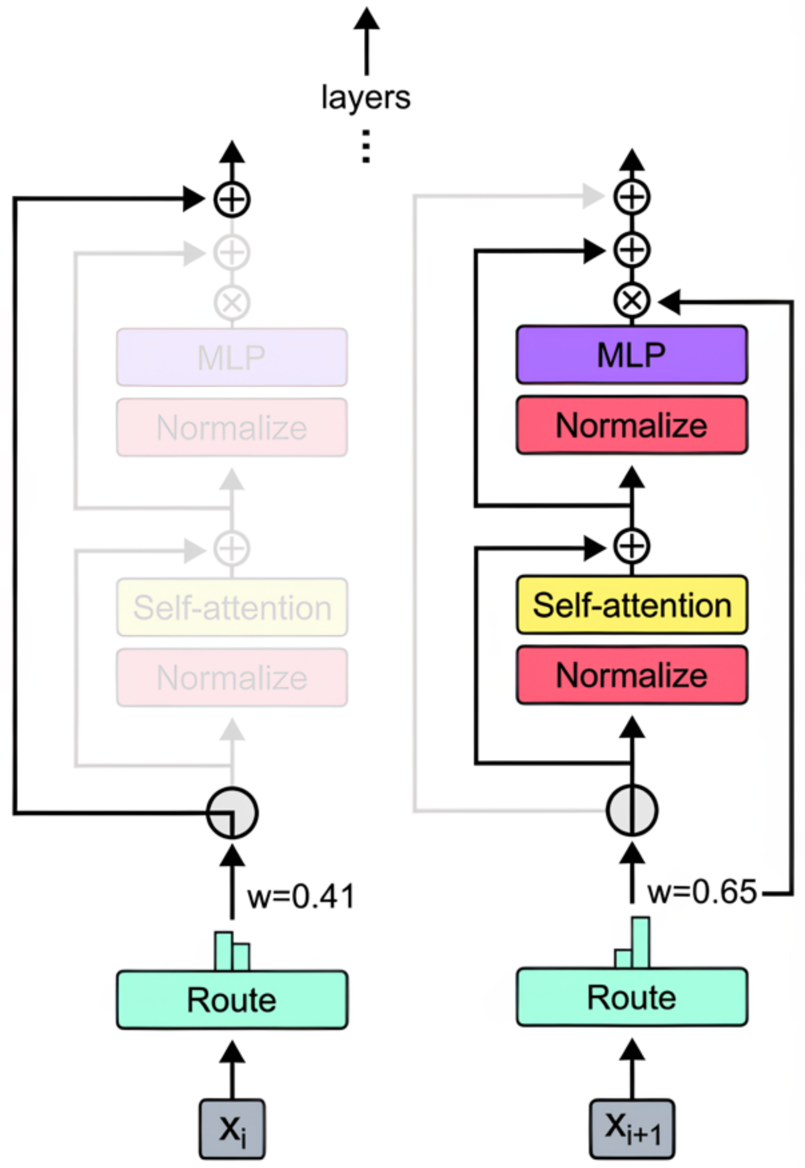
Selectively processes tokens through different layers based on importance.

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.

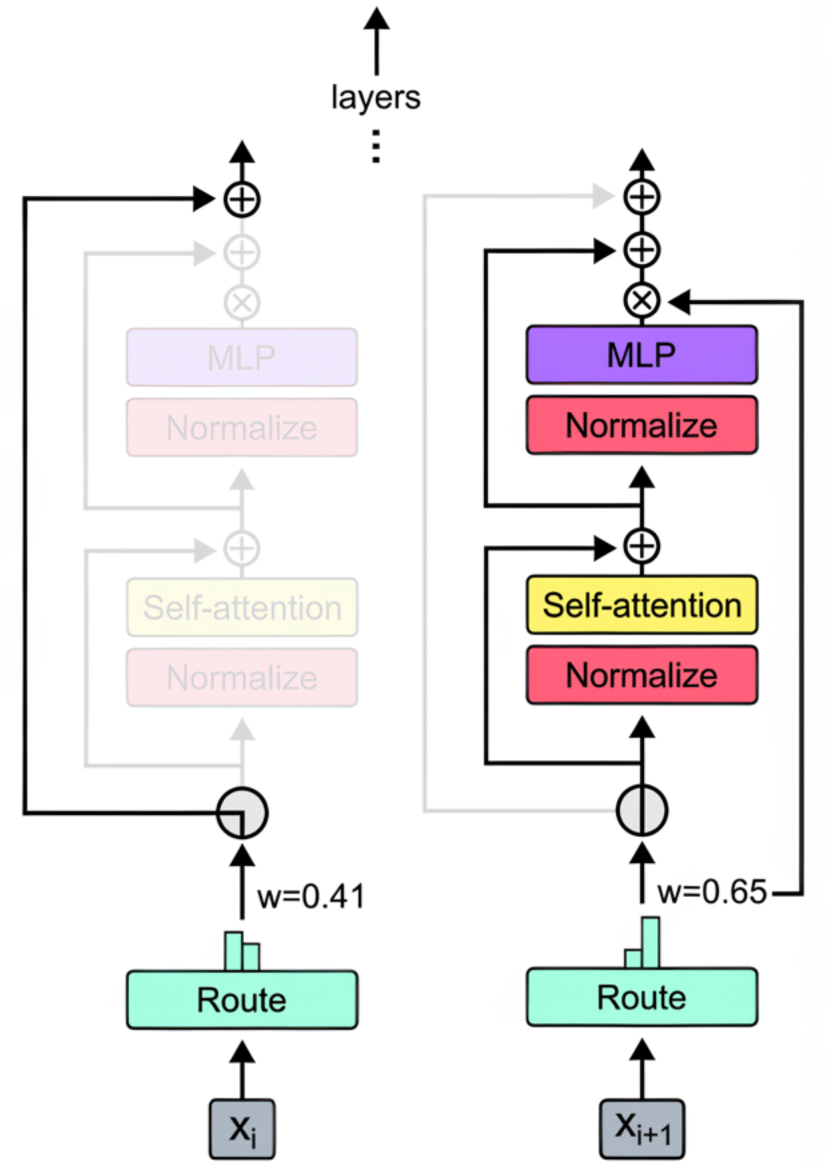
Compute Savings

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.

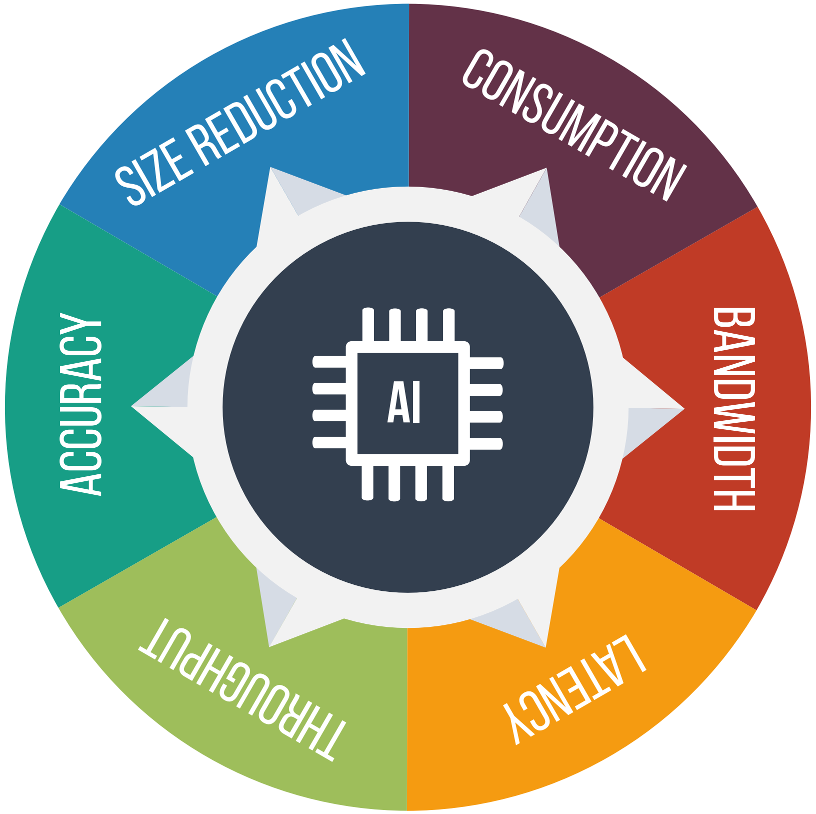


MIXTURE OF DEPTHS

Key Metrics

Size Reduction
Up to 40%

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



Accuracy
Equal to or often better than handcrafted models.

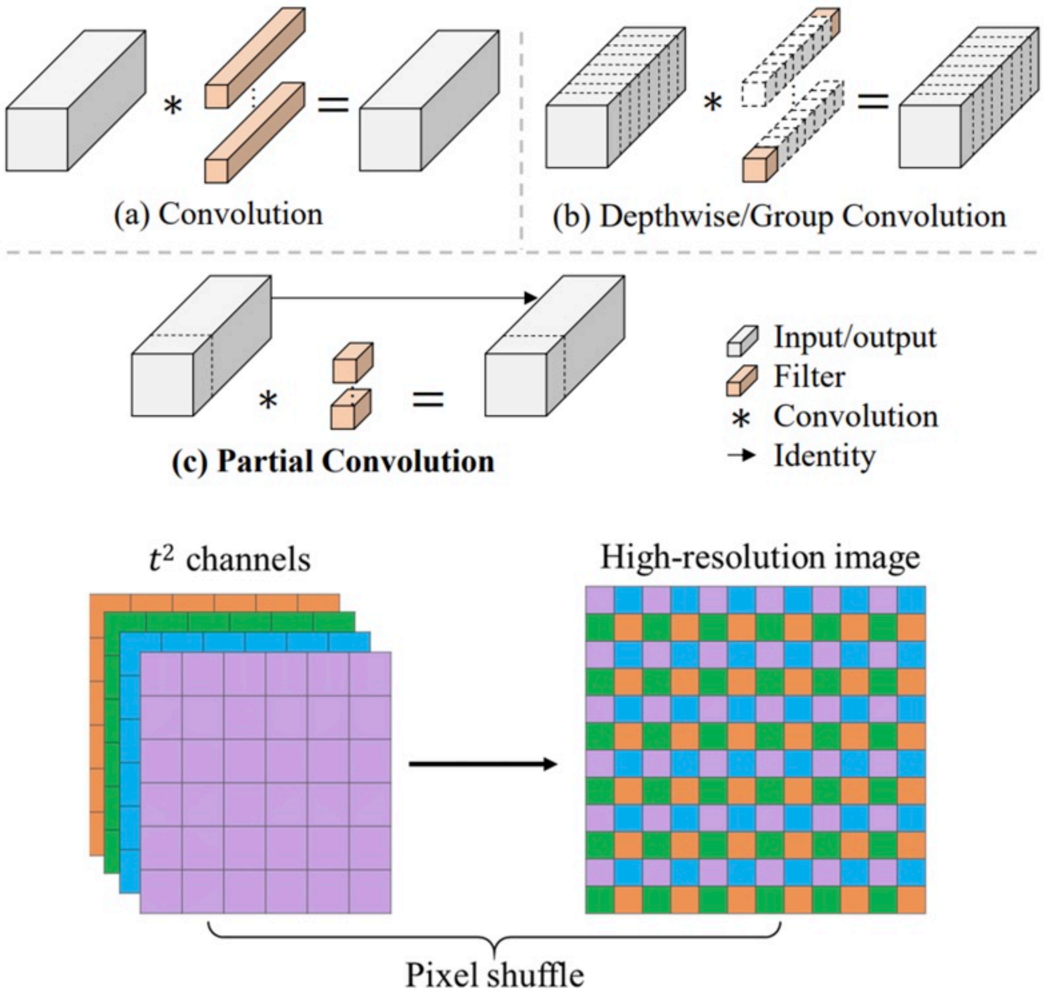
Memory Bandwidth
25% – 50% reduction.

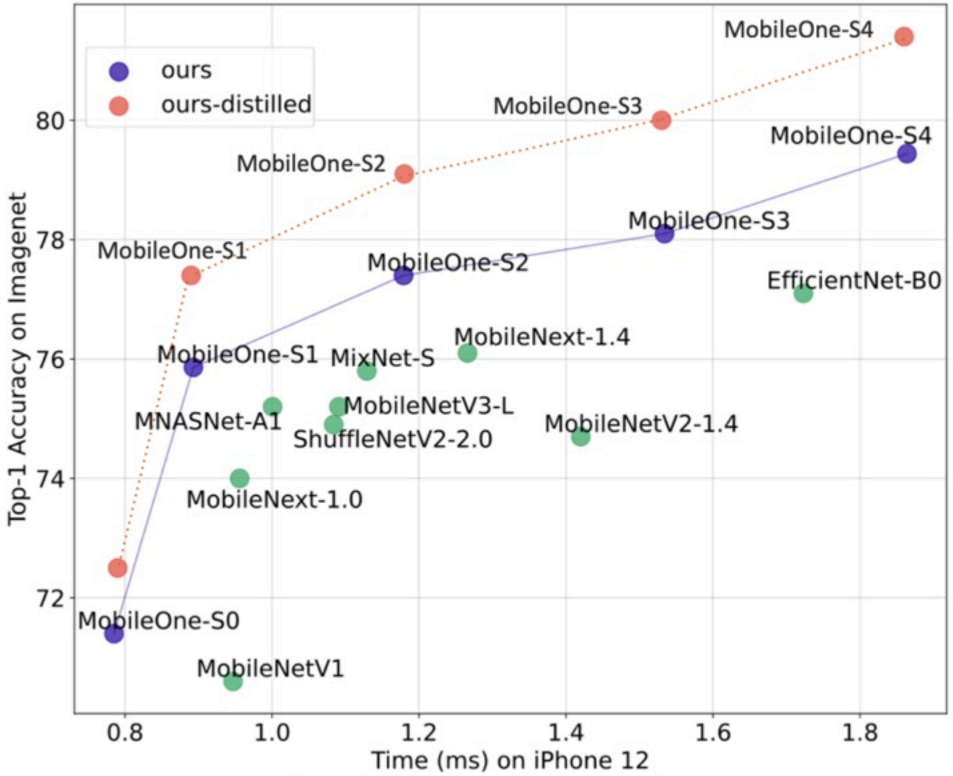
Throughput
2x – 4x higher compared to non-optimized models.

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

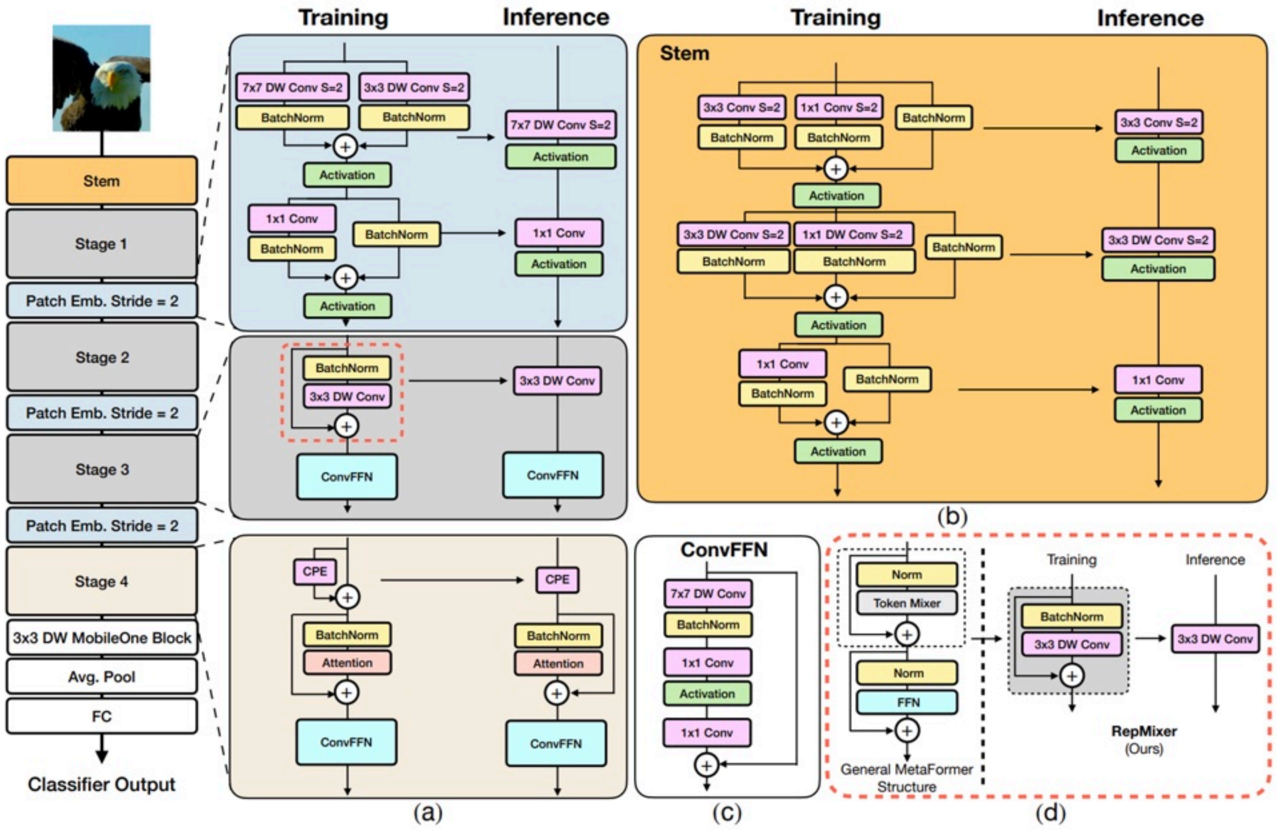


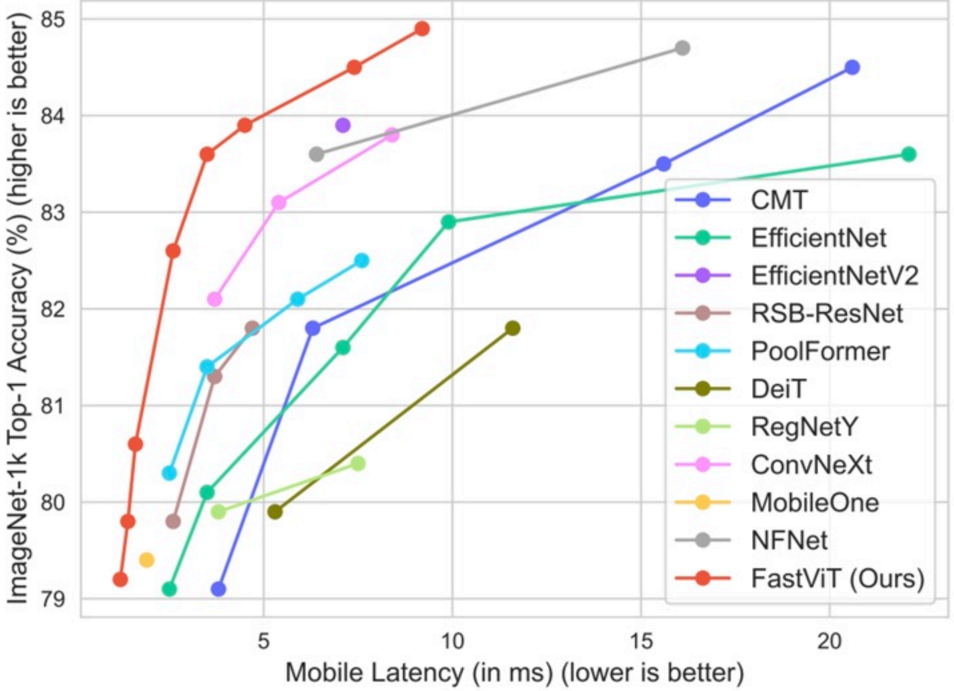


MobileOne: An Improved One millisecond Mobile Backbone

HARDWARE AWARE DESIGN

FastViT: A Fast Hybrid Vision Transformer using Structural Reparameterization



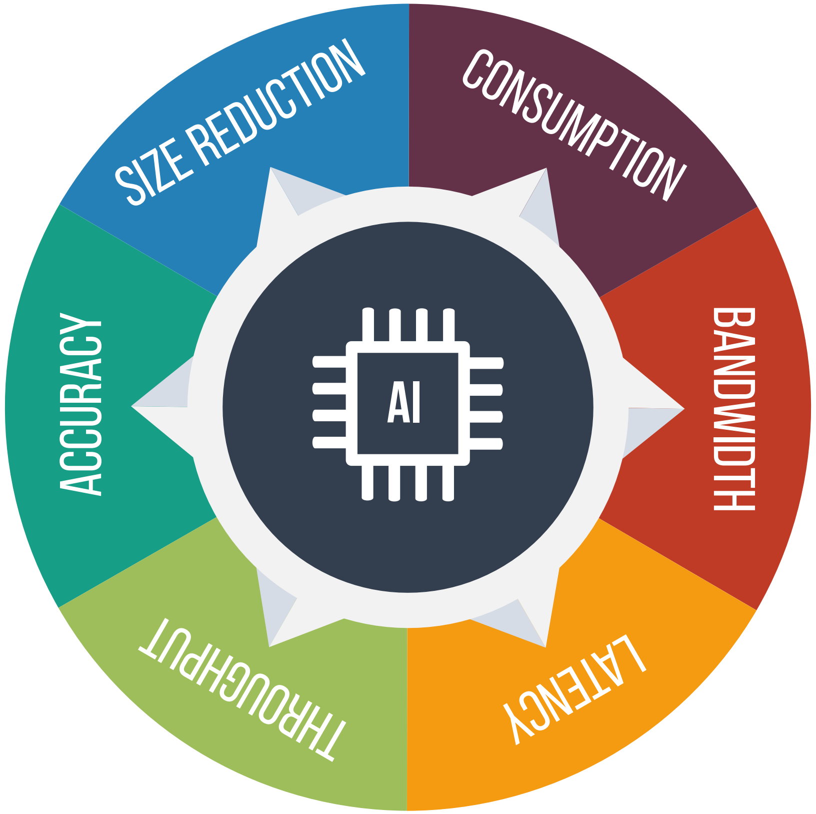


HARDWARE AWARE DESIGN

Key Metrics

Size Reduction
Up to 30% – 50%
w.r.t teacher model.

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



Accuracy
Equal to or often
better than
handcrafted models.

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

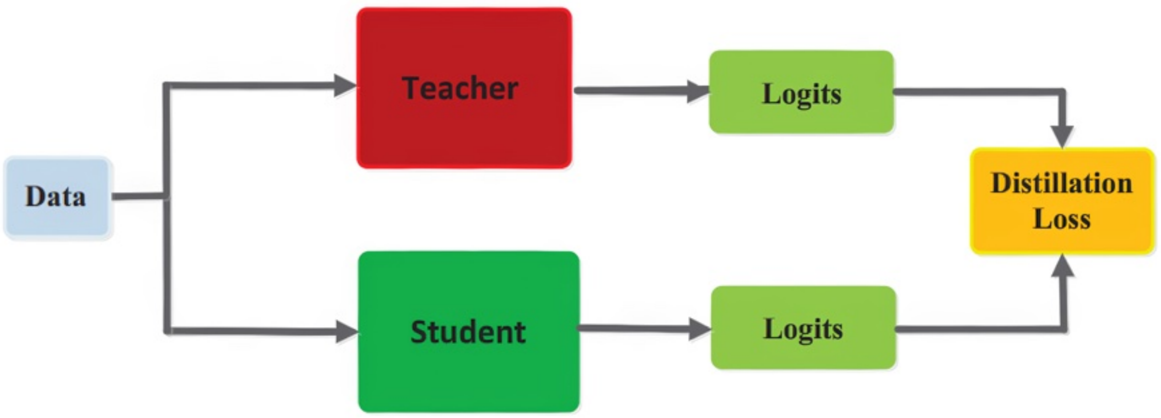
Throughput
2x – 4x higher
compared to non-
optimized models.

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

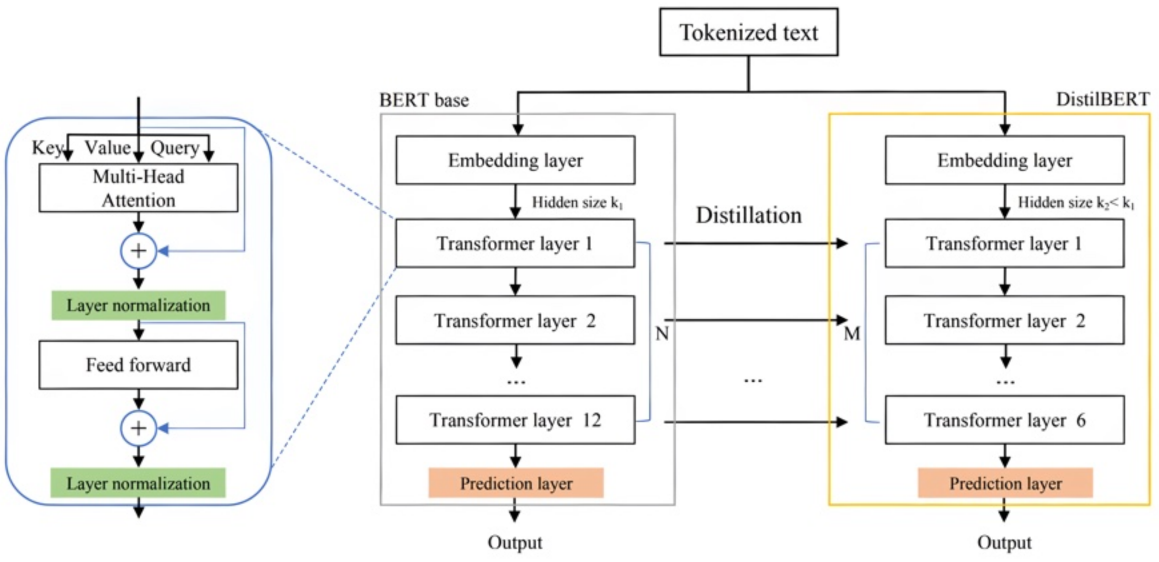
KNOWLEDGE DISTILLATION

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



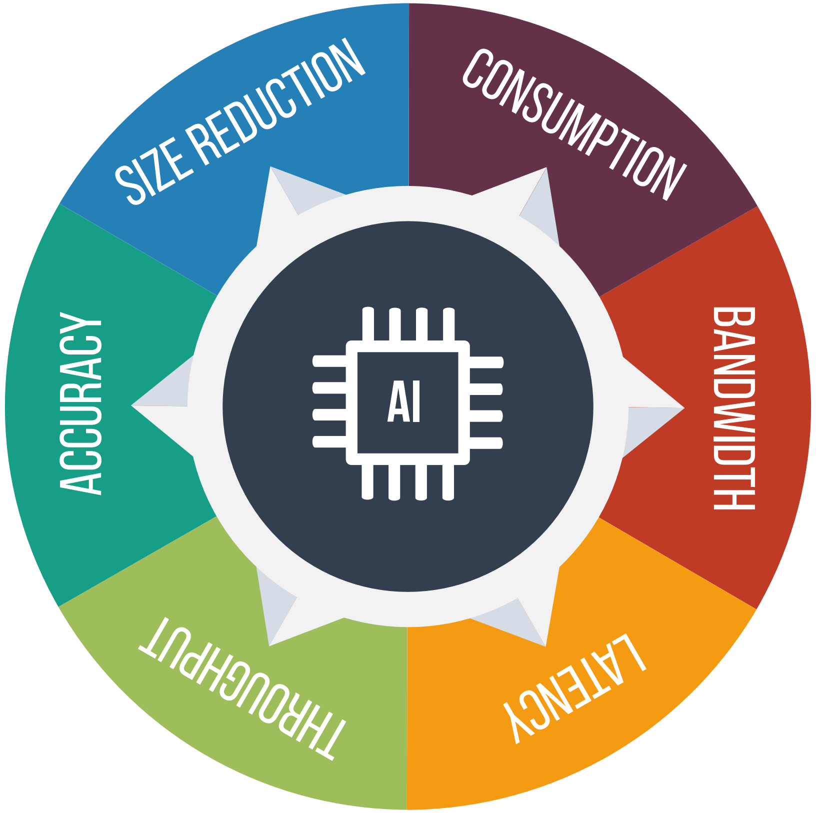
Improving Crisis Events Detection Using DistilBERT with Hunger Games Search Algorithm

KNOWLEDGE DISTILLATION

Key Metrics

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

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



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

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

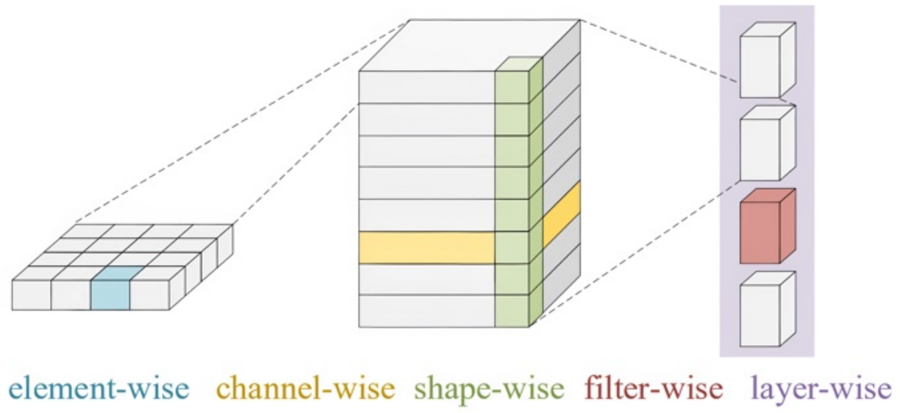
Throughput
2x – 10x higher
w.r.t teacher model.

Latency
2x – 10x reduction.

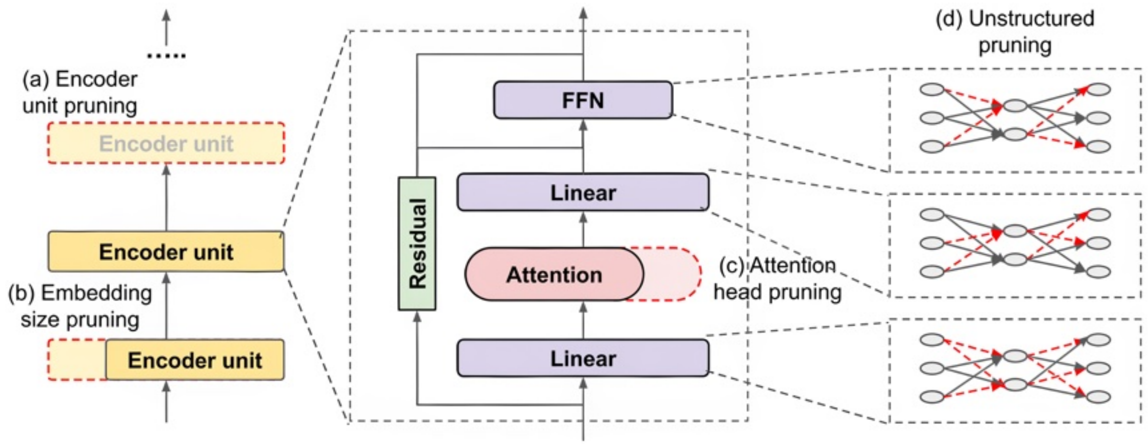
PRUNING

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

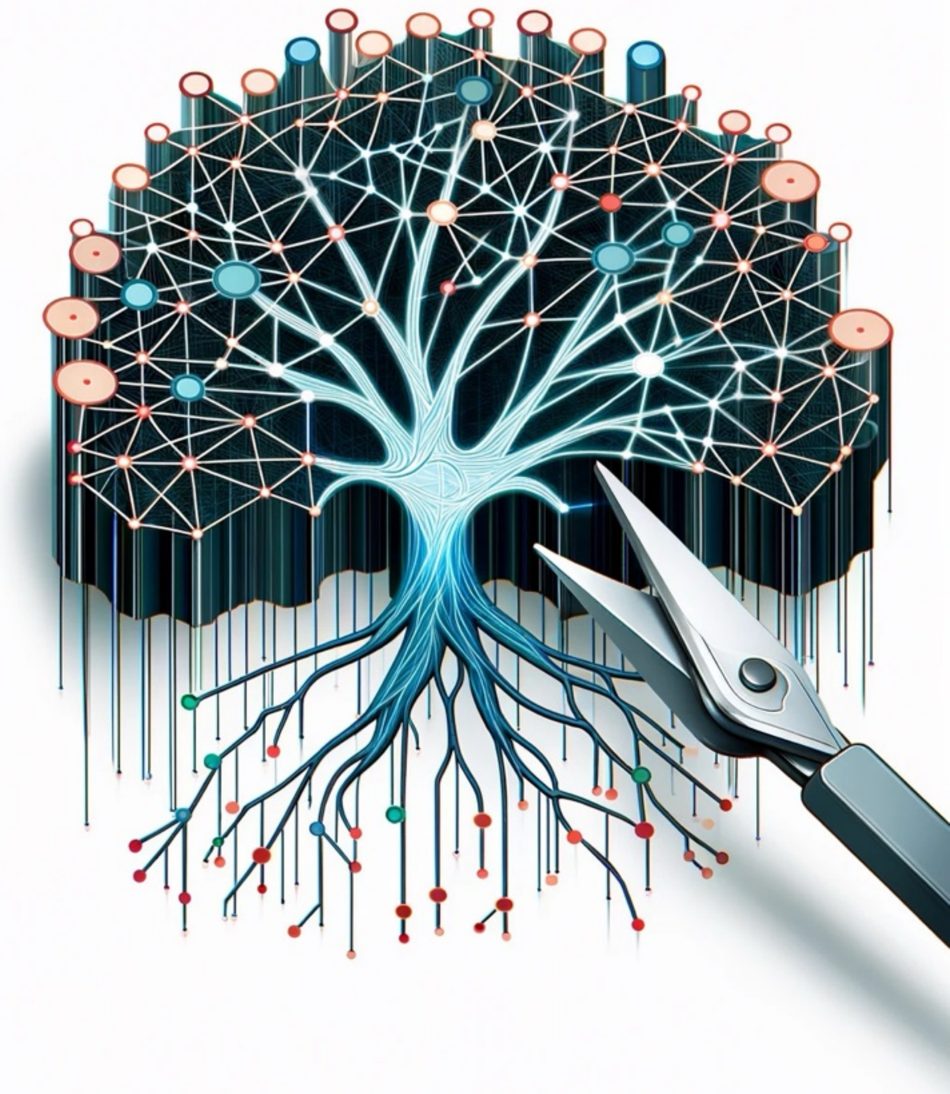
Targeting parameters based on their absolute values.

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?



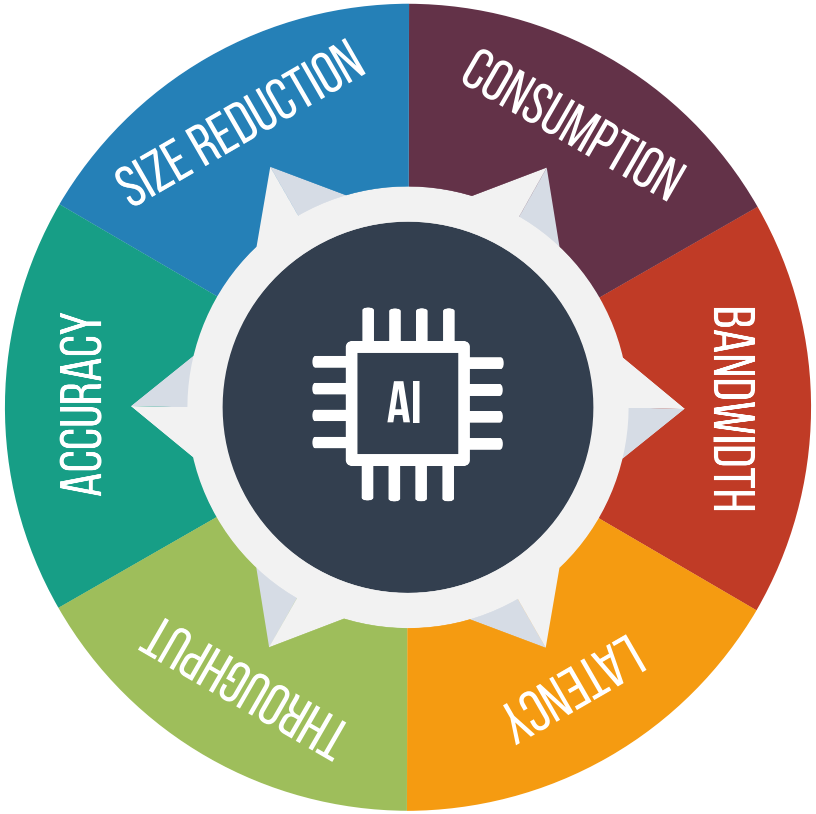
PRUNING

Key Metrics

Size Reduction
Up to 90x smaller models.

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

Throughput
10% – 50% higher.



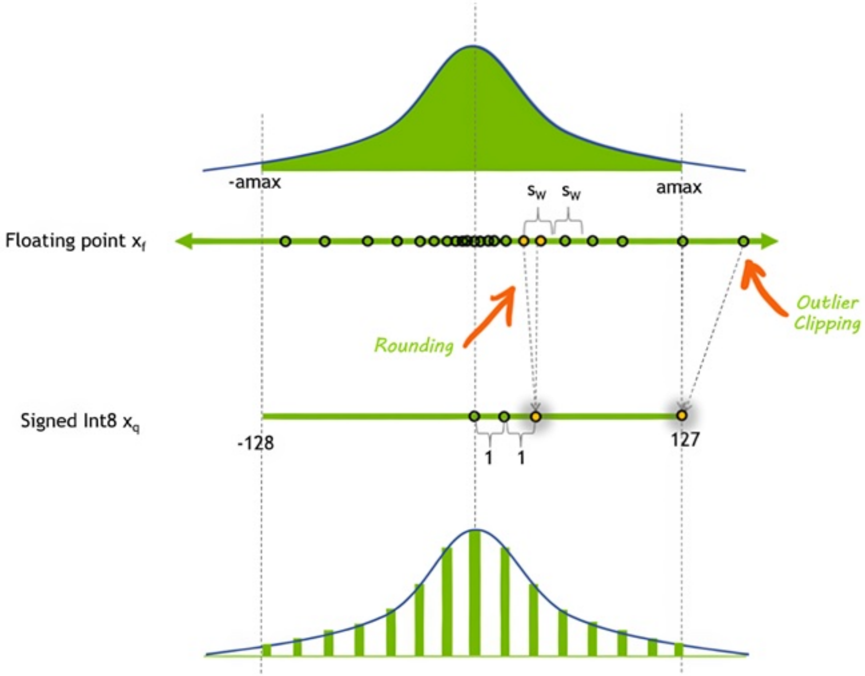
Power & Energy Consumption
10% – 50% lower consumption.

Memory Bandwidth
10% – 50% reduction.

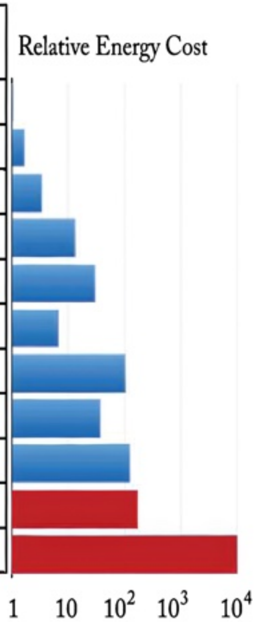
Latency
10% - 40% reduction.

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. ”

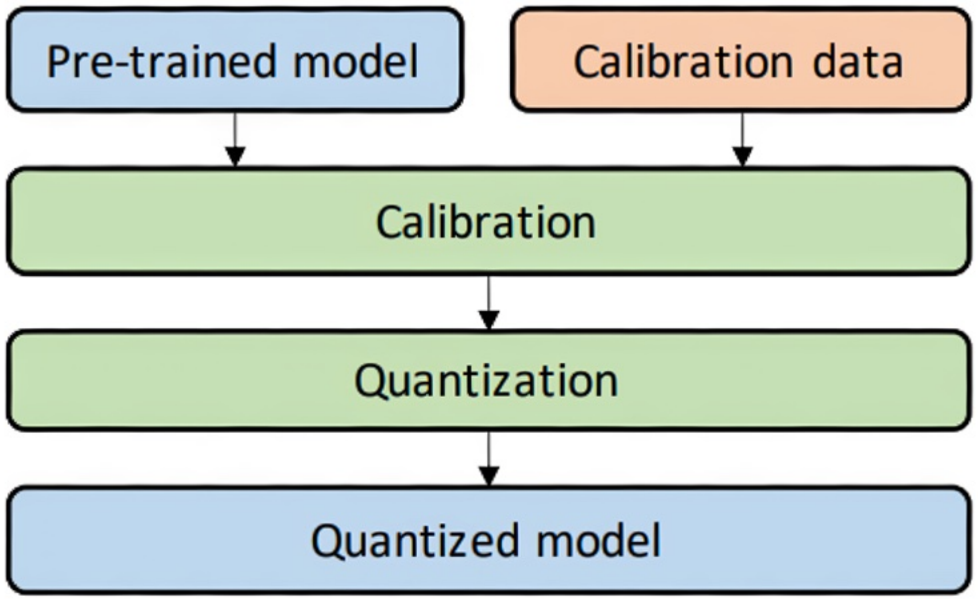


Operation:	Energy (pJ)	Area (μm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A

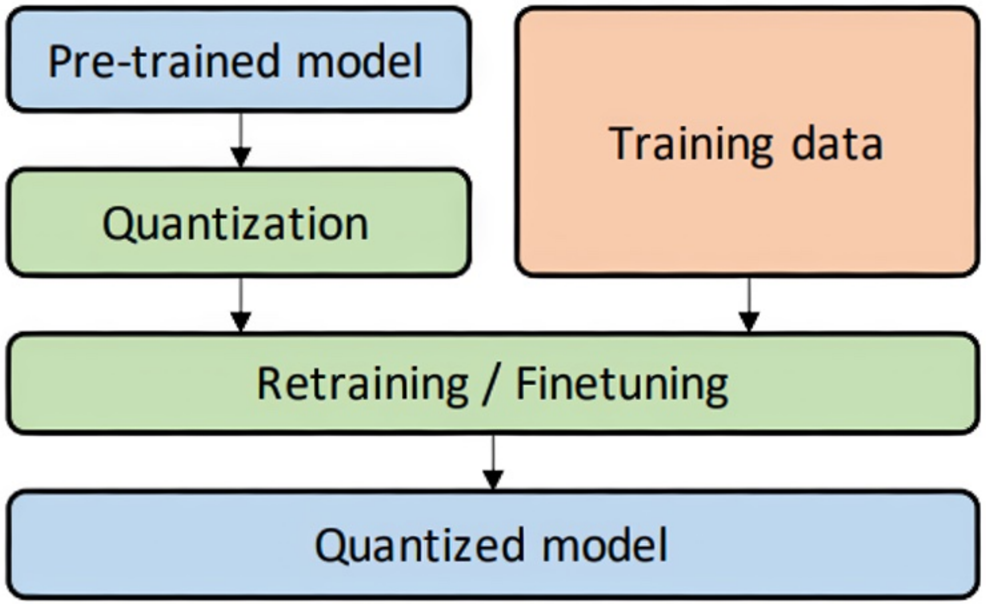


QUANTIZATION

A Survey of Quantization Methods for Efficient Neural Network Inference



Post Training Quantization



Quantization Aware Training

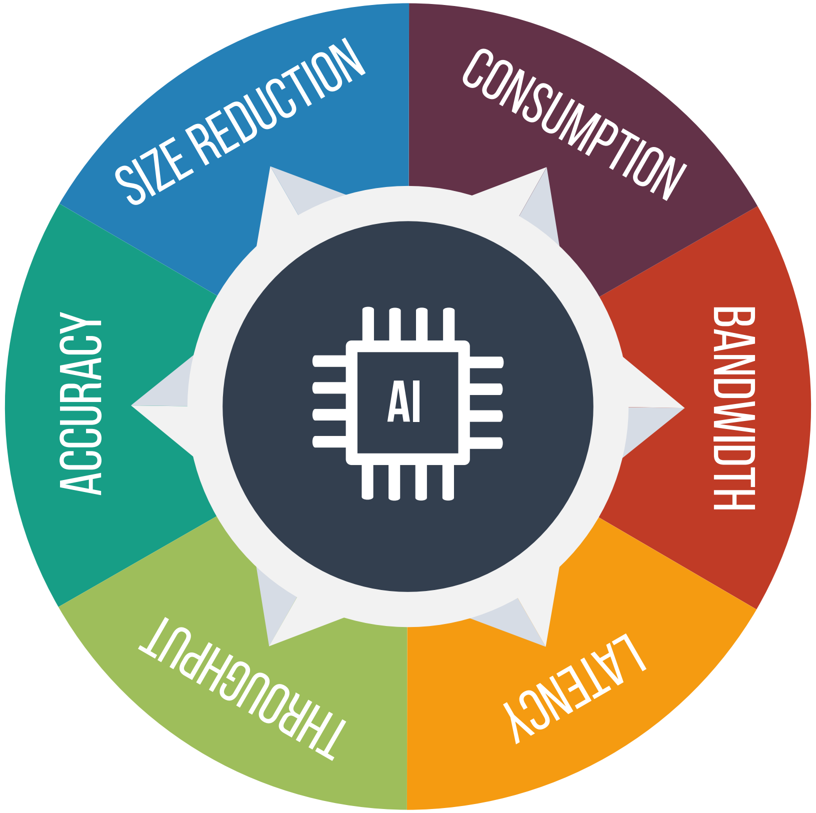
QUANTIZATION

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.

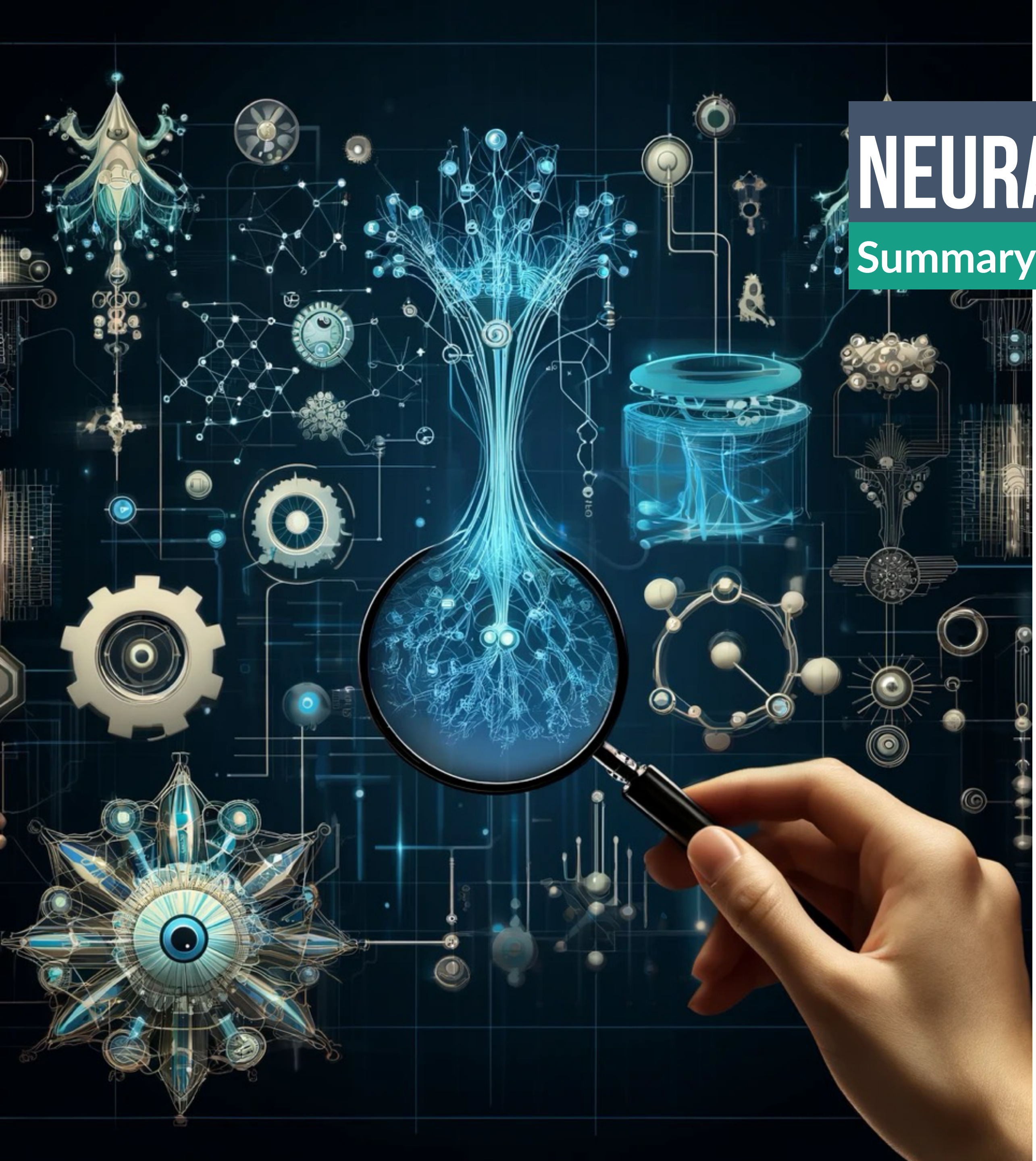


Power & Energy Consumption
2x – 3x lower consumption.

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

Latency
2x – 3x reduction.

SUMMARIES OF MODEL COMPRESSION TECHNIQUES



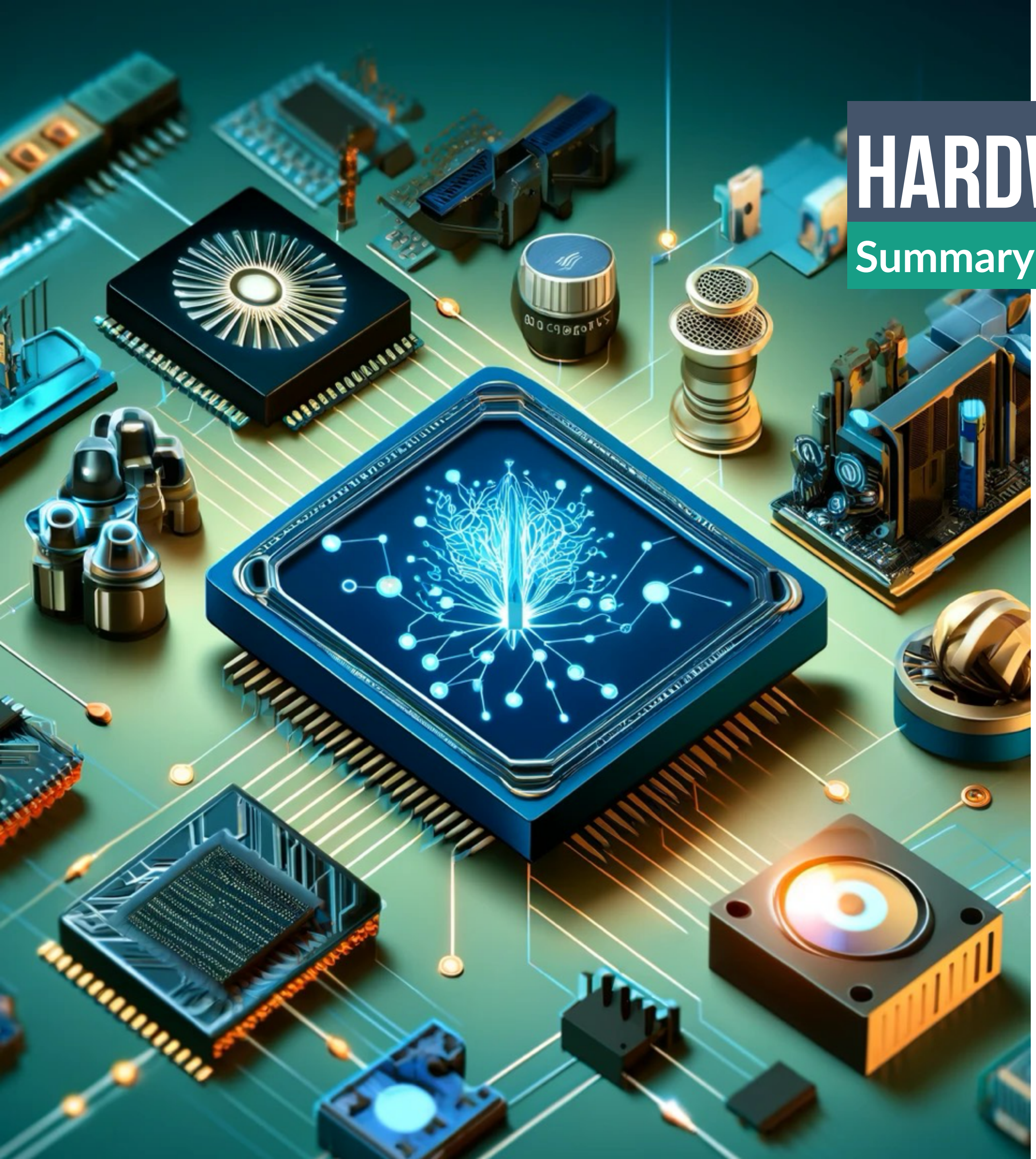
NEURAL ARCHITECTURE SEARCH

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.



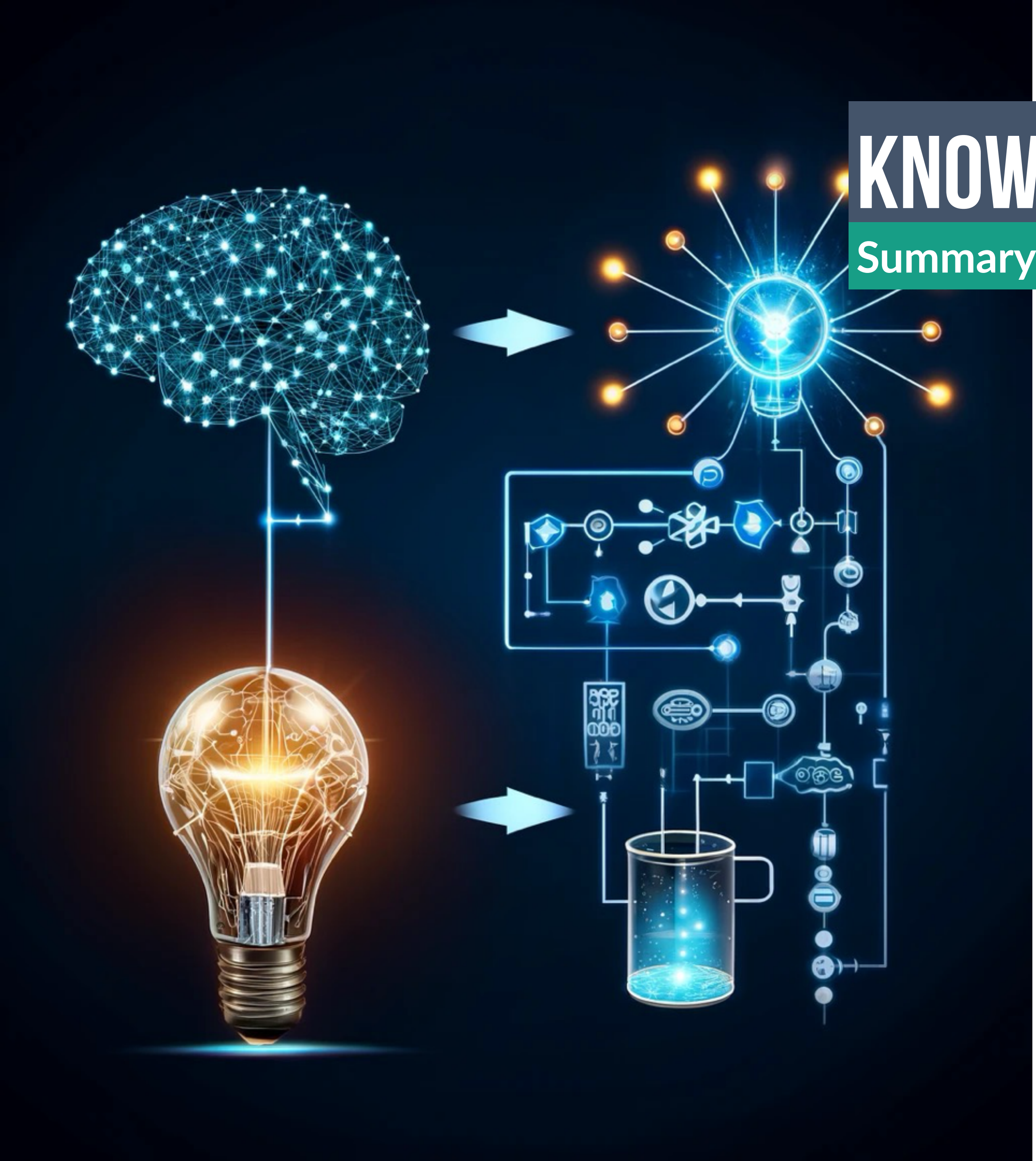
HARDWARE AWARE DESIGN

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.

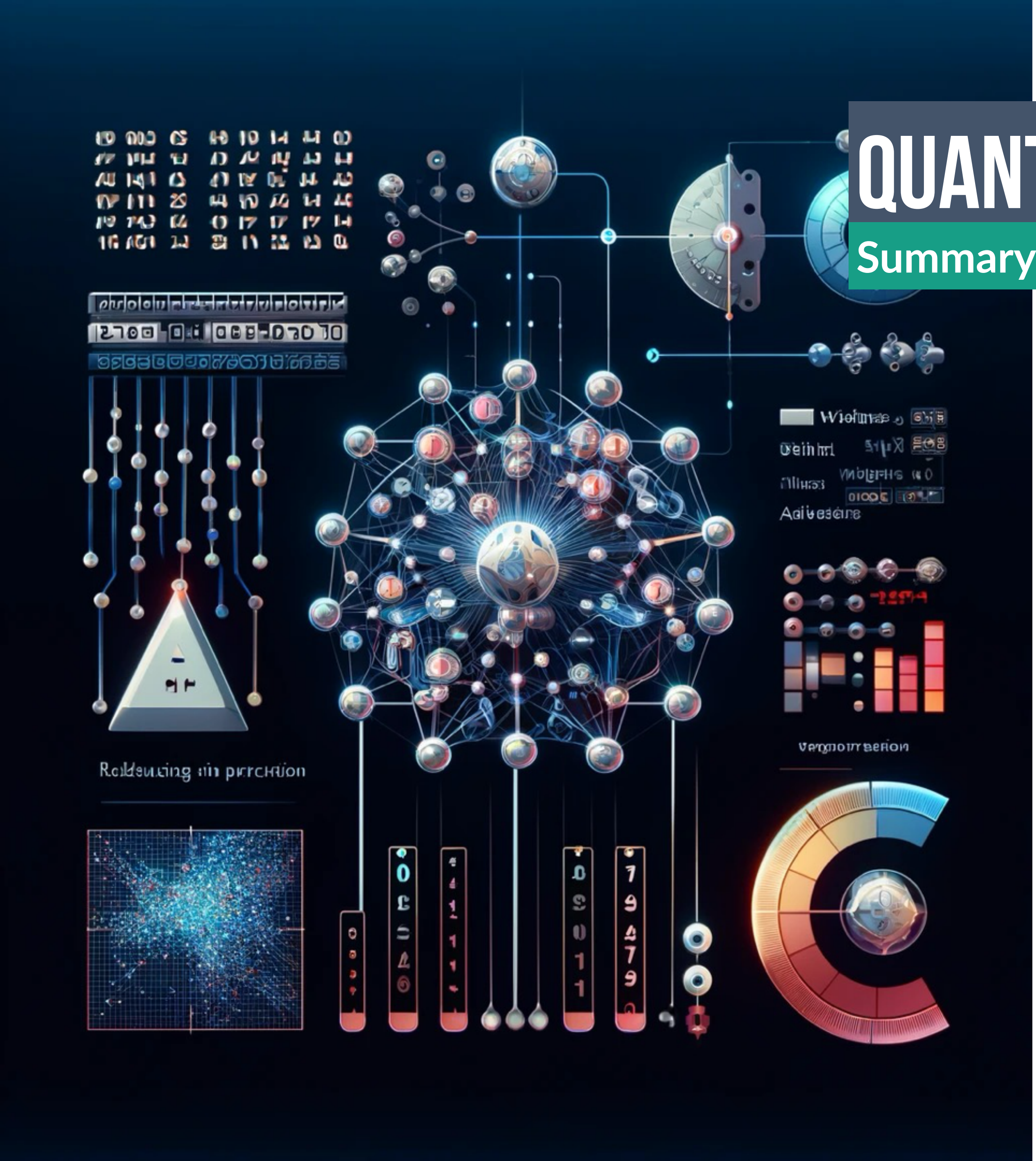
PRUNING

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

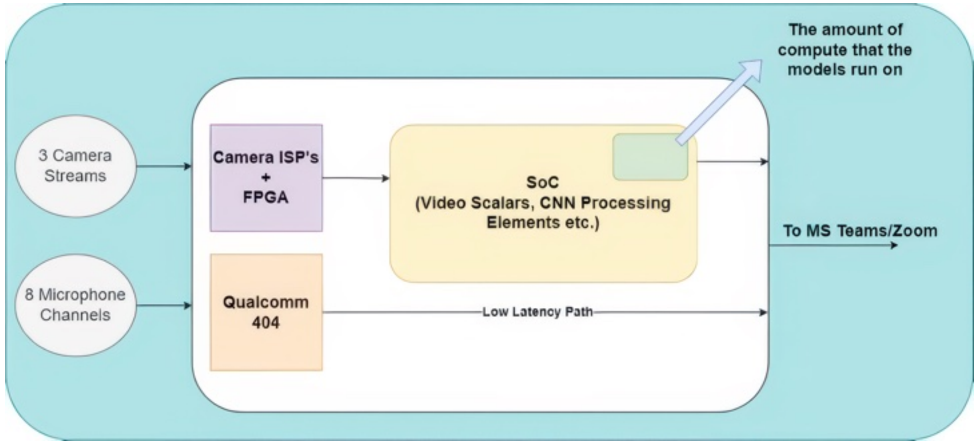
Myriad X devices support only FP16 bit widths and have limited memory and compute budget shared across all processes.

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

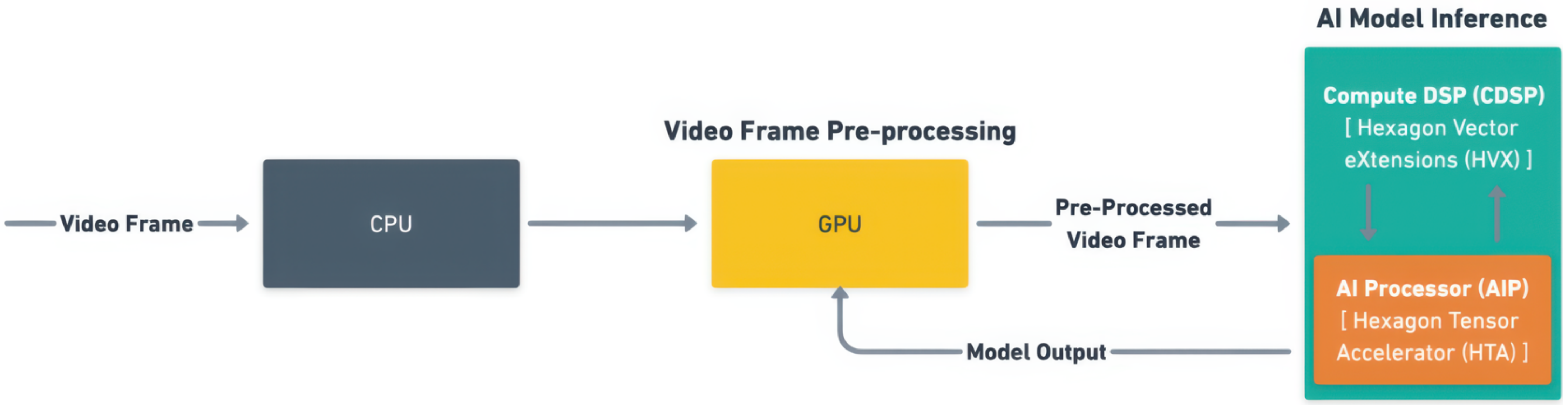
Range
Model Working Distance - 18 ft to 20 ft (small/medium conference rooms).

Detected People
Must be able to detect 1 to 20 people.

Precision
Low False Positives/False Negatives.

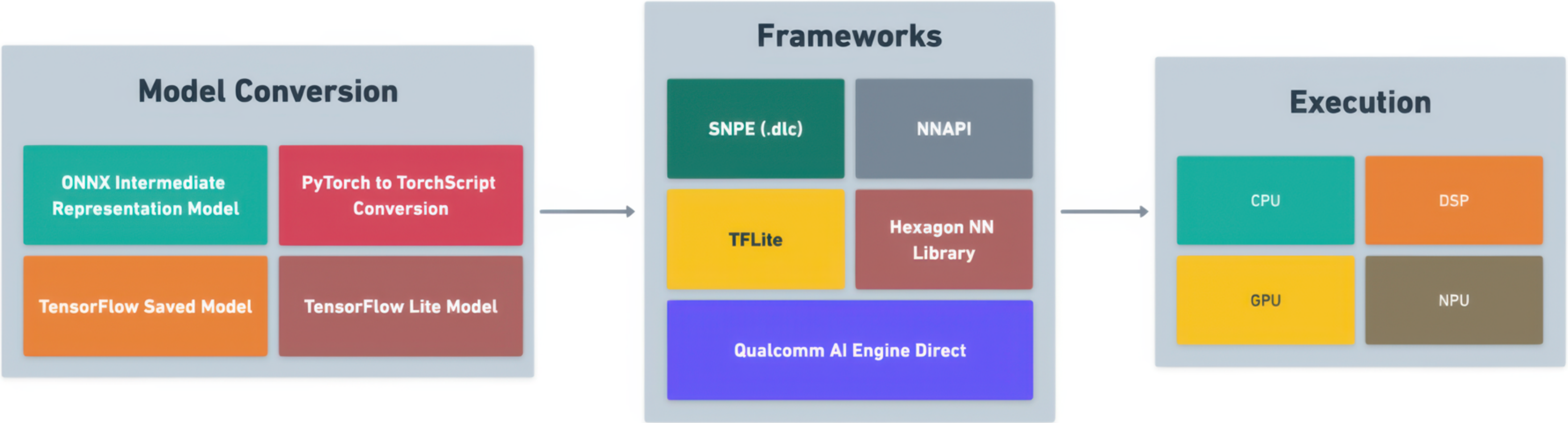


QUALCOMM INFERENCE END-TO-END
Workflow



WORKFLOW FOR MODEL DEPLOYMENT

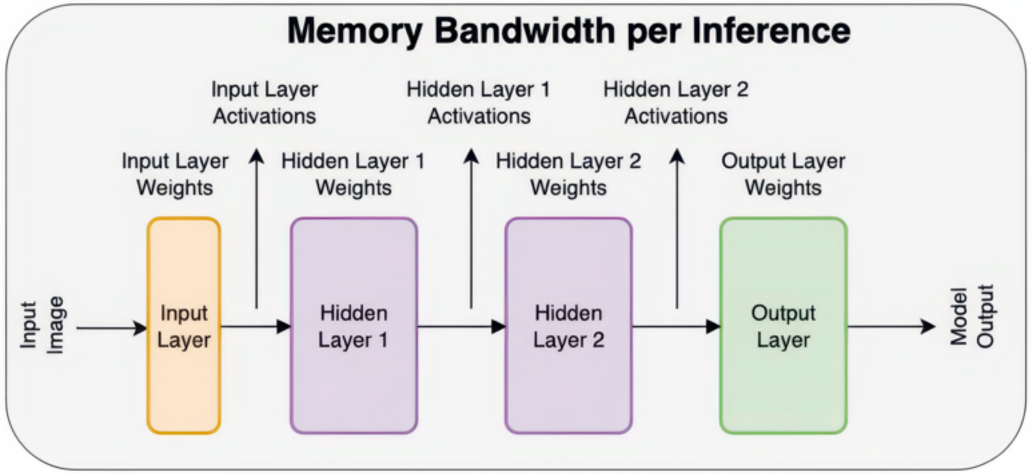
Deploying Machine Learning Models on Qualcomm Hardware



MEMORY BANDWIDTH
Challenge-1

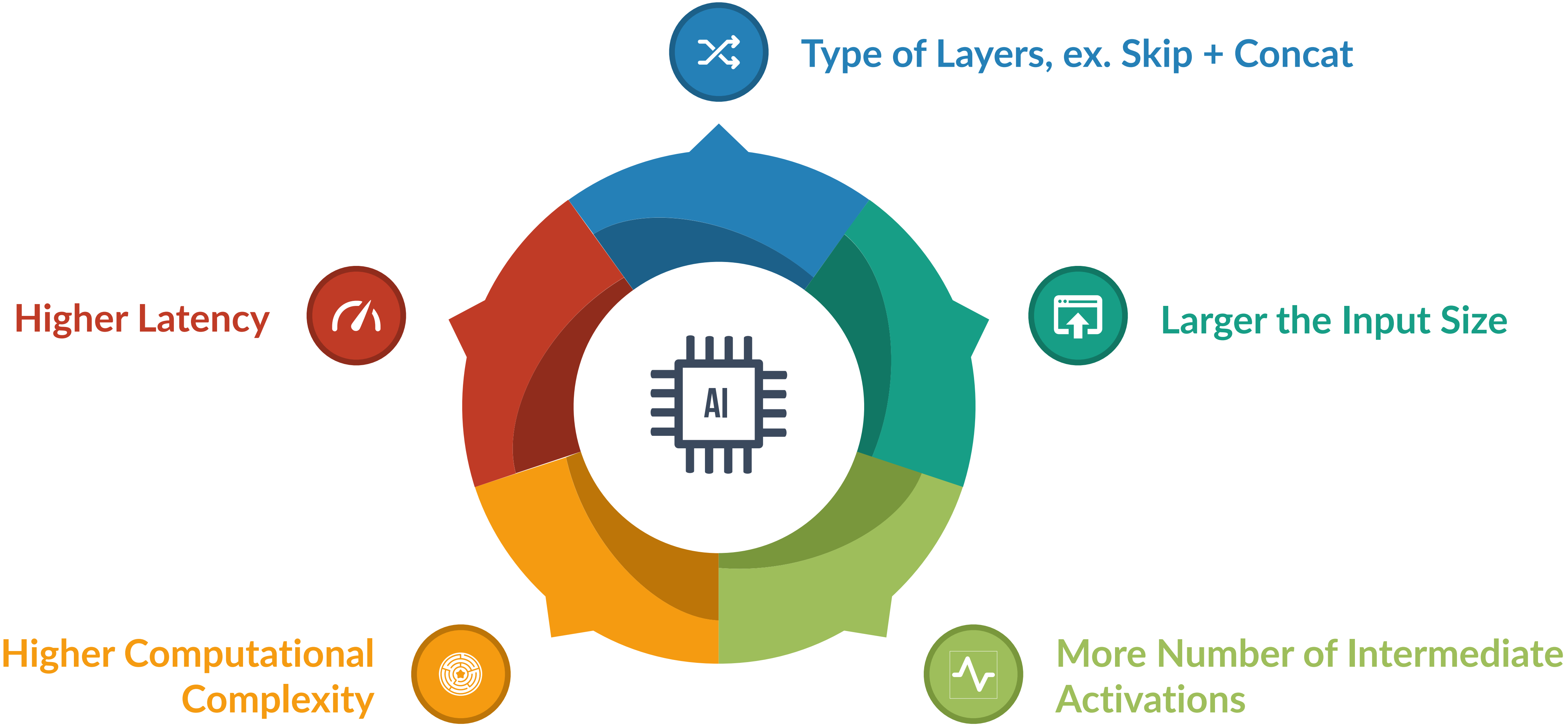
ML models utilize the same memory pool as other system processes. Some factors influencing Memory Bandwidth per Frame:

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



MODEL LATENCY

Challenge-2



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

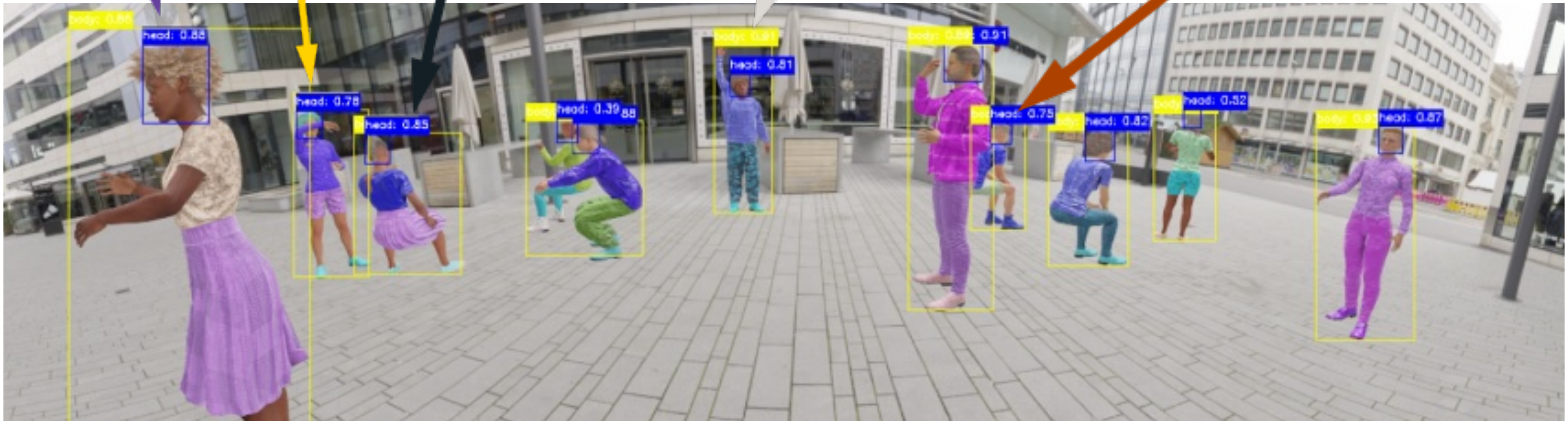
Person Facing Sideways

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

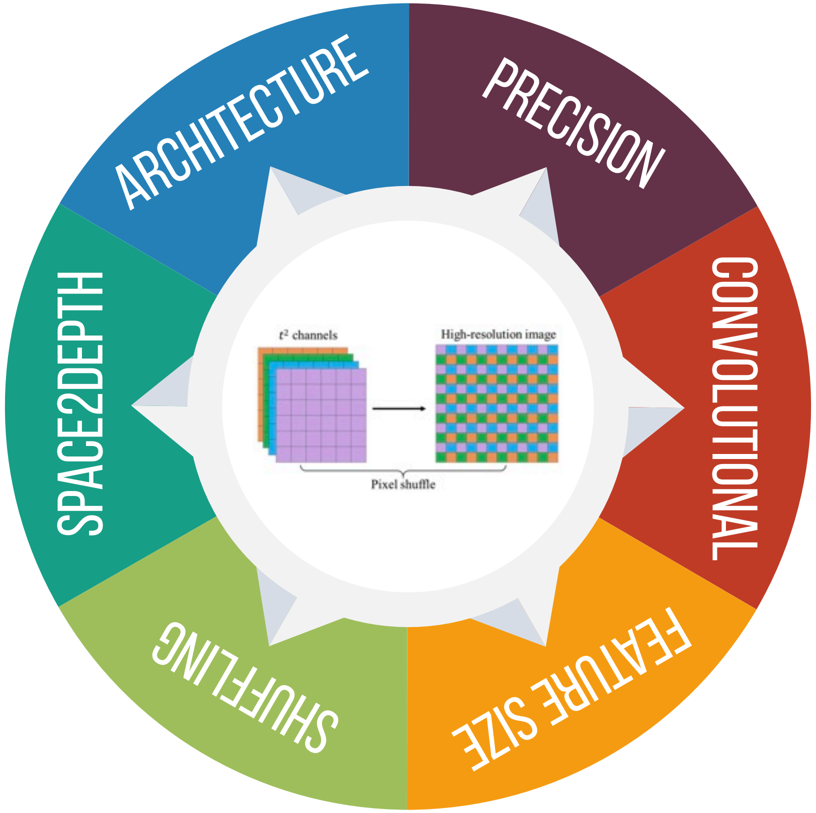
Understanding Hardware

Architecture

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

Half Precision Training

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



Space2Depth

Use large kernel convolutions with large stride at input.

Convolutional

Use efficient Conv layers like GhostConv, PartialConv, etc.

Pixel Shuffling

Use pixel shuffling at the output instead of TransposeConv2d

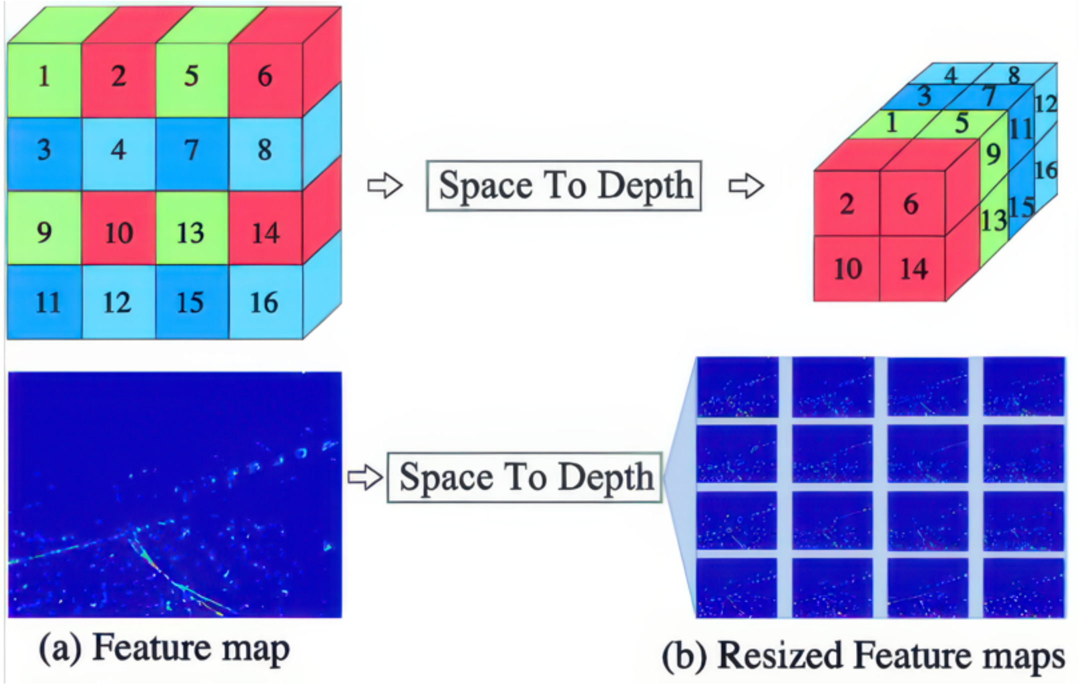
Feature Size

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

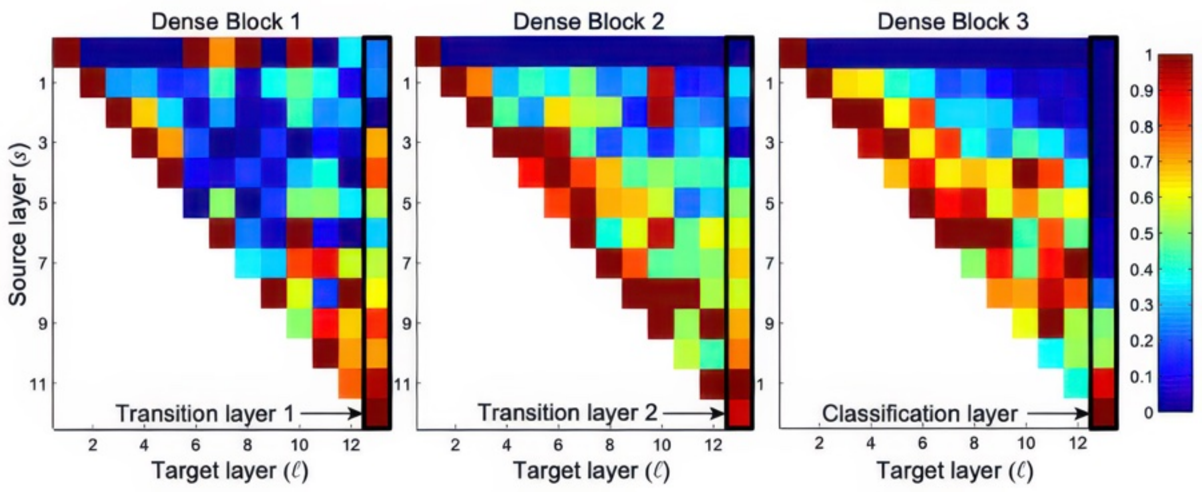
SPACE-TO-DEPTH VS. CONV2D

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

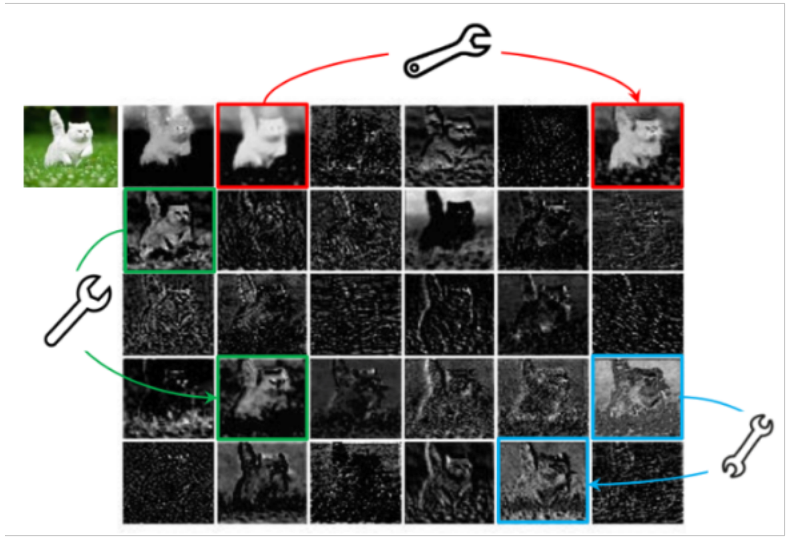
GHOST CONVOLUTIONS
Model Optimization

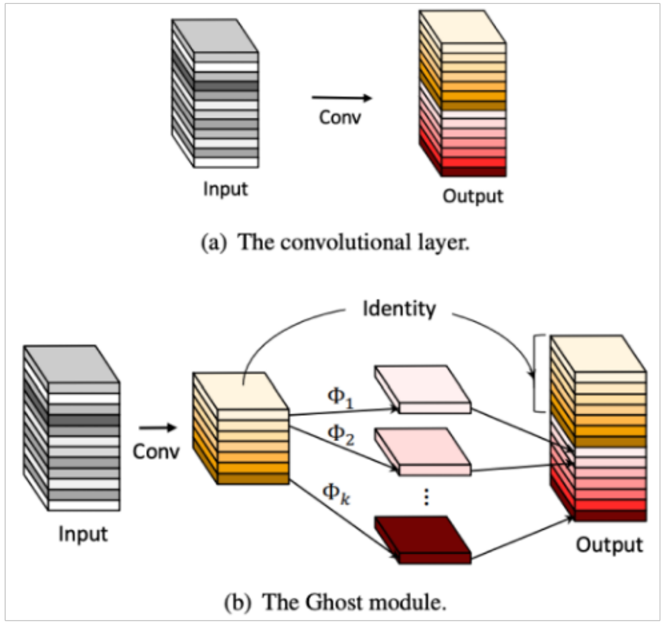
Feature Augmentation
Produces additional 'ghost' feature maps via DepthWiseConv2D.

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.

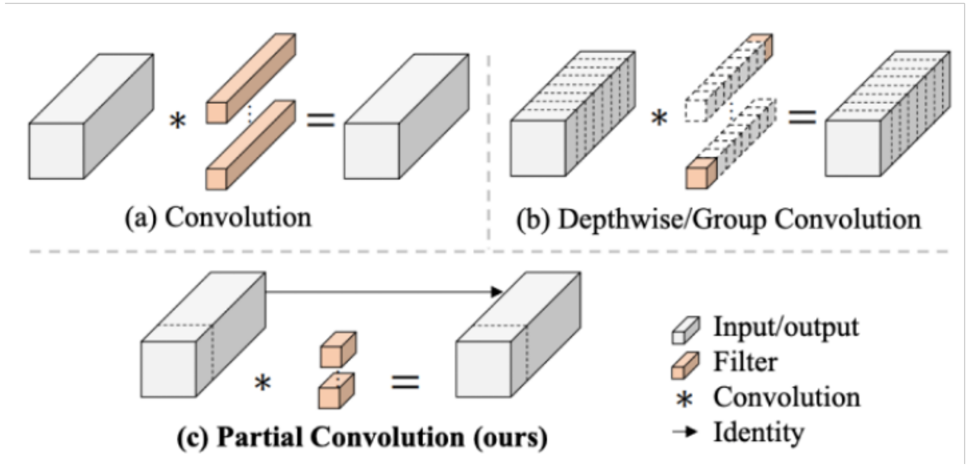
GHOSTCONV2D VS. CONV2D

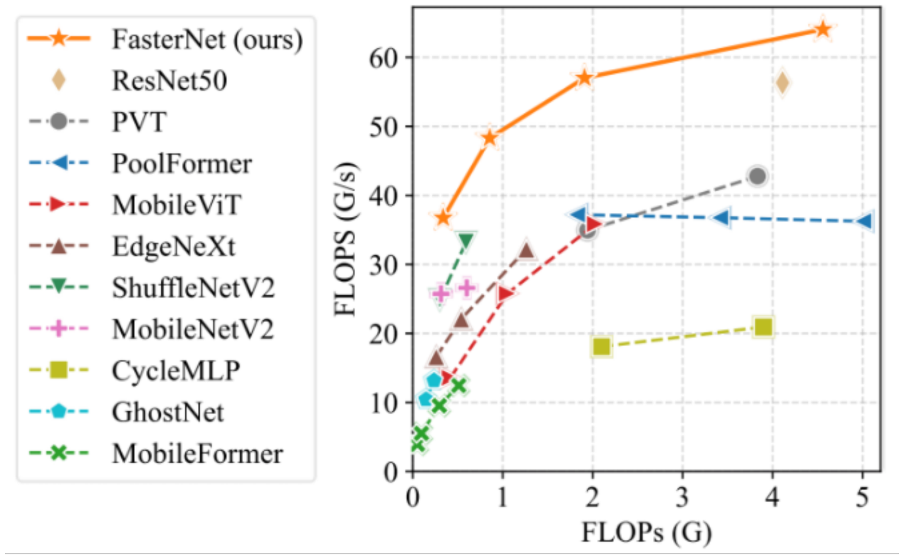
Results

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

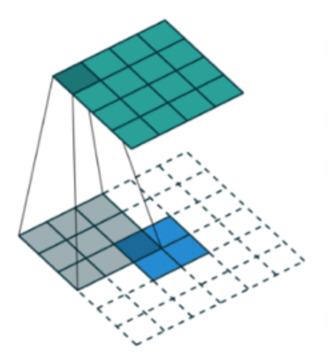
Results

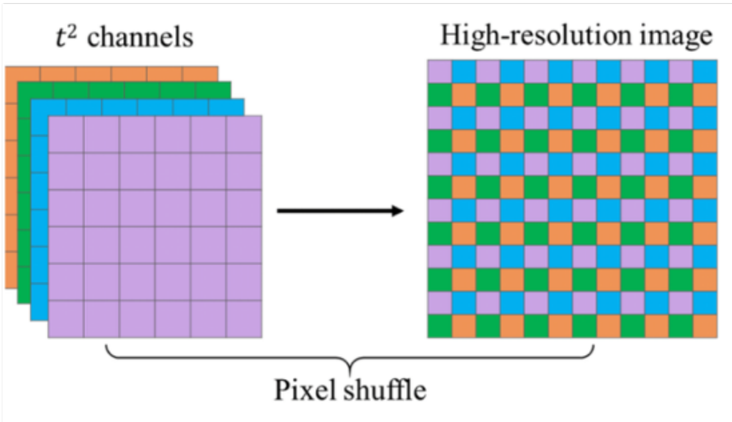
Layer Type	Input Channels	Output Channels	MAC Operations	Number of Parameters
Conv2D + BN + ReLU	32	32	4.21 G	9.28 K
Conv2D + BN + ReLU	64	64	16.725 G	36.992 K
PartialConv2D	32	32	320.79 M	742 Bytes
PartialConv2D	64	64	1.157 G	2.630 K

REPLACING TRANSPOSEDCONV2D

Overview

TransposedConv2D





TransposedConv2d

Upsamples feature maps using learnable parameters.

Pixel Shuffle

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

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

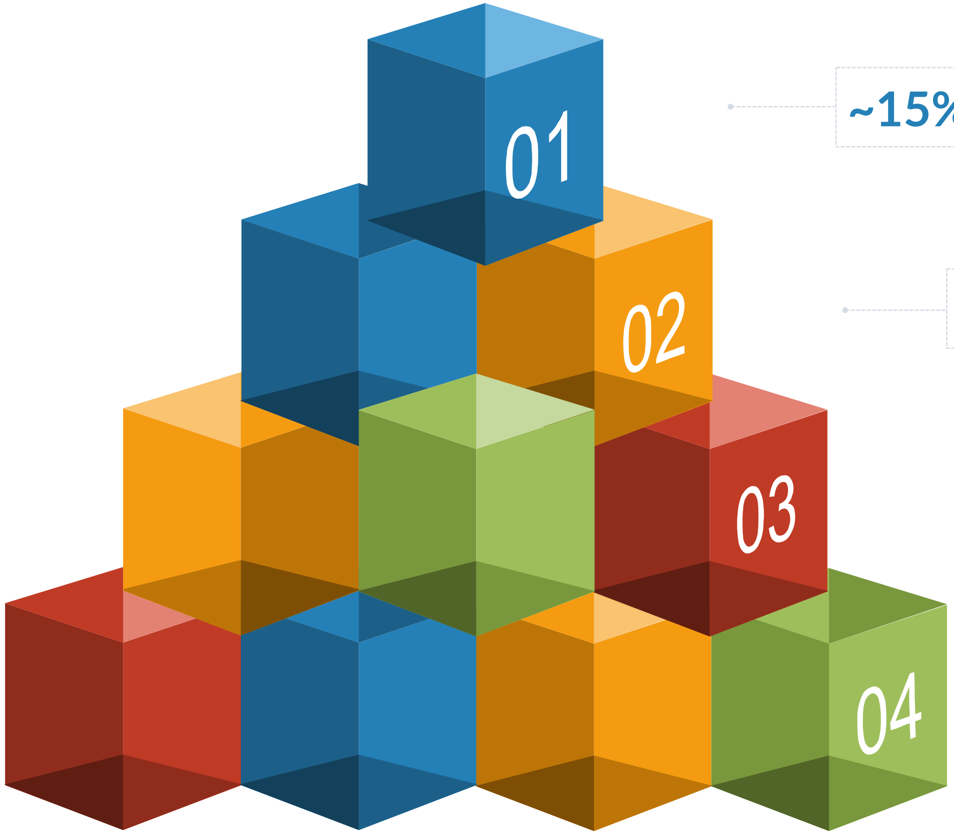
~40% improvement (reduced operations, in GhostConv).

~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%

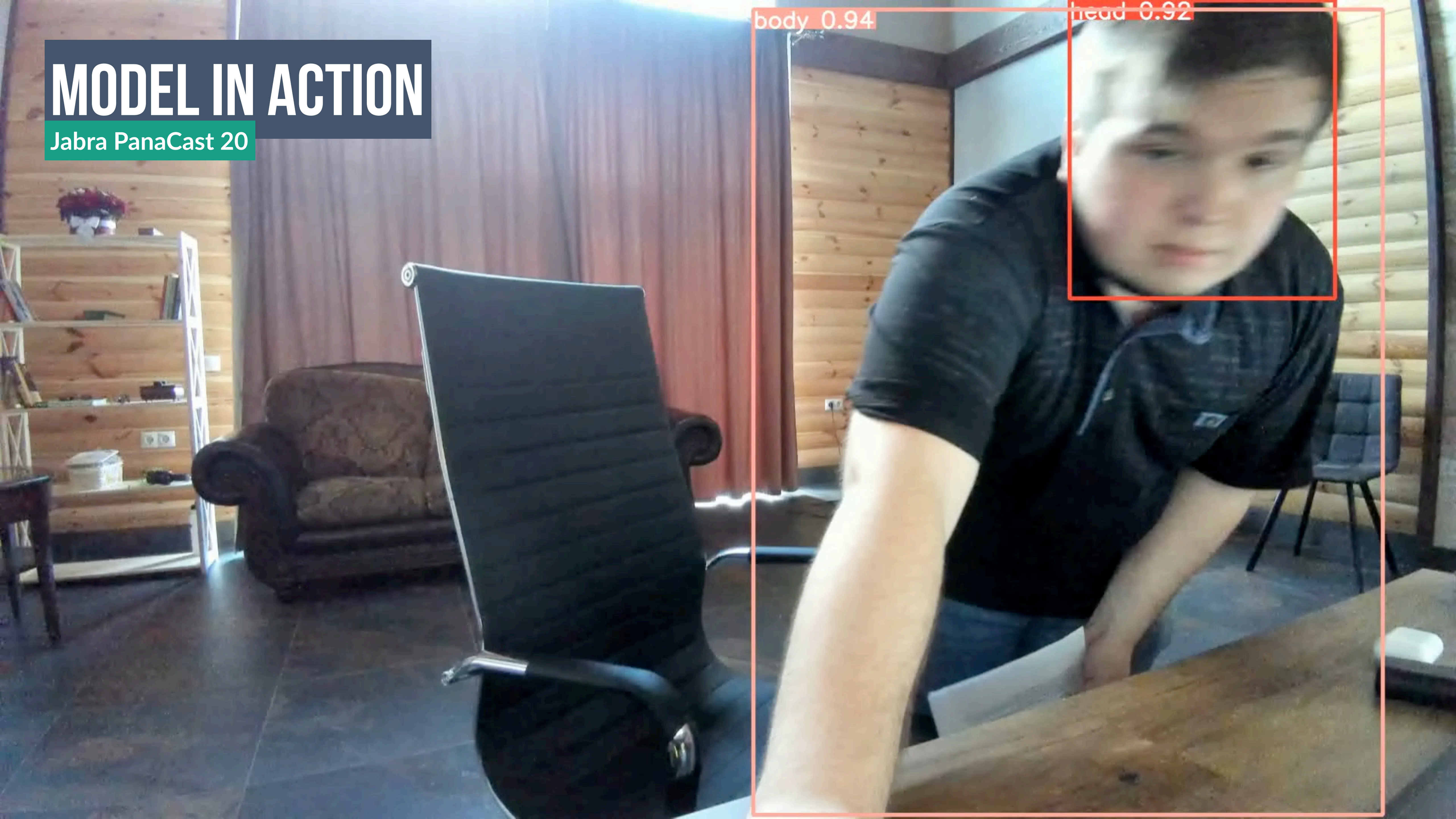
Partial Conv/GhostConv
~35% reduction (fewer parameters).

~10%

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

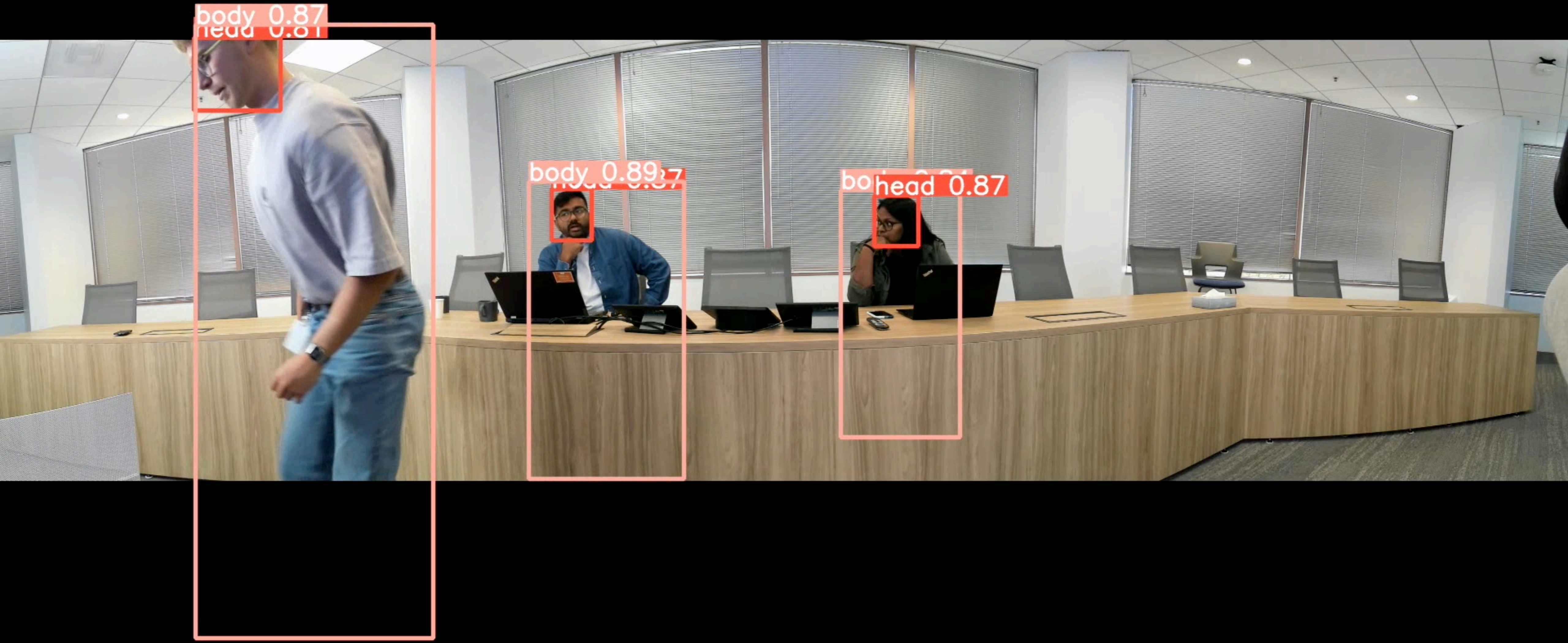
MODEL IN ACTION

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

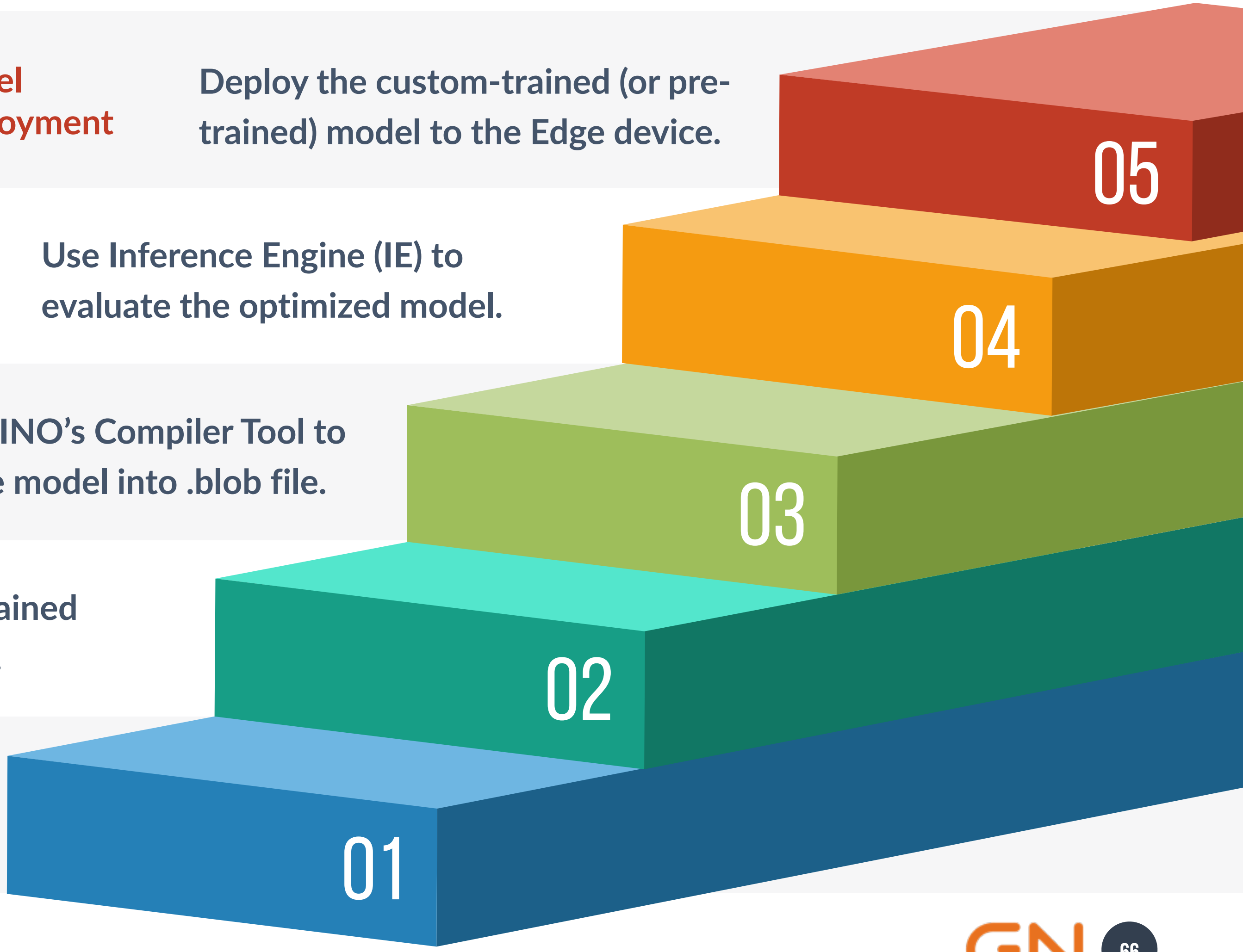
Model Deployment — Deploy the custom-trained (or pre-trained) model to the Edge device.

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.

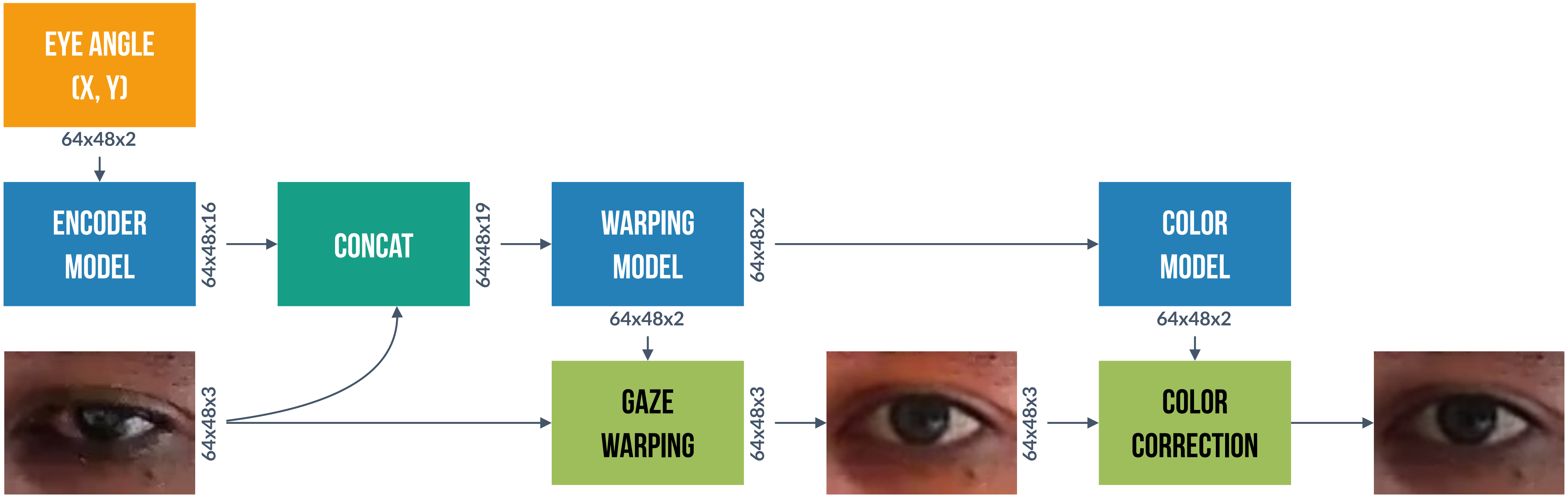
ONNX Conversion — You must convert the trained model to ONNX format.

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



JABRA EYE CORRECTION

Gaze Correction Model Based on Warping Technique



- 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

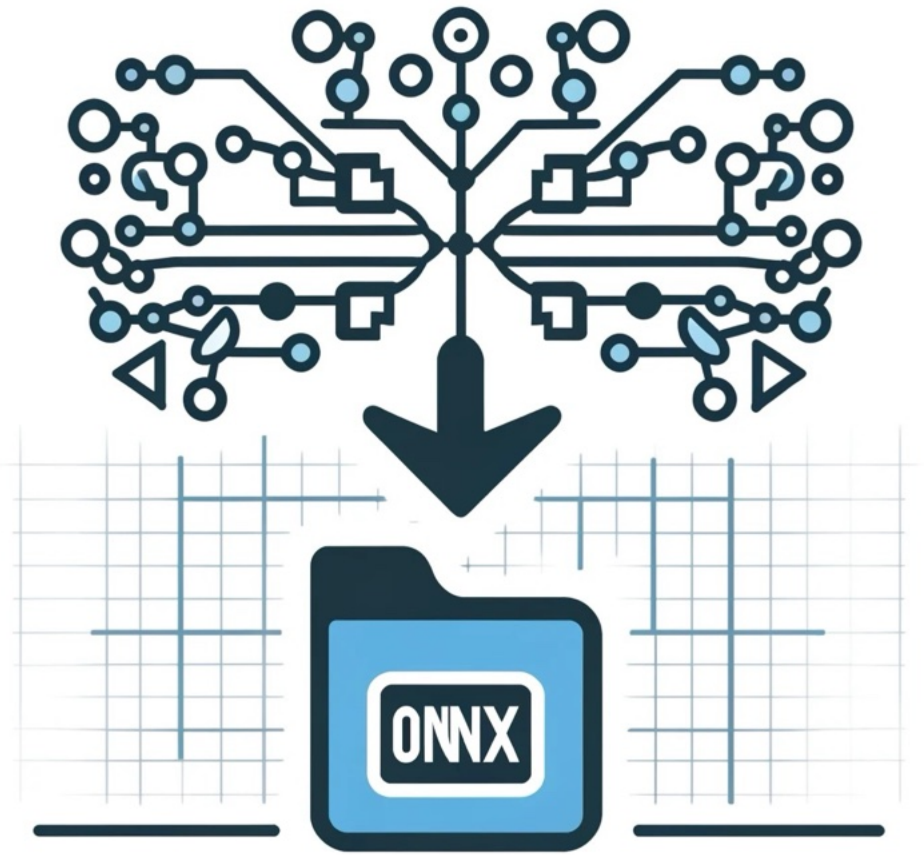
The *export* package is based on TorchScript backend and has been available since PyTorch 1.2.0.

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

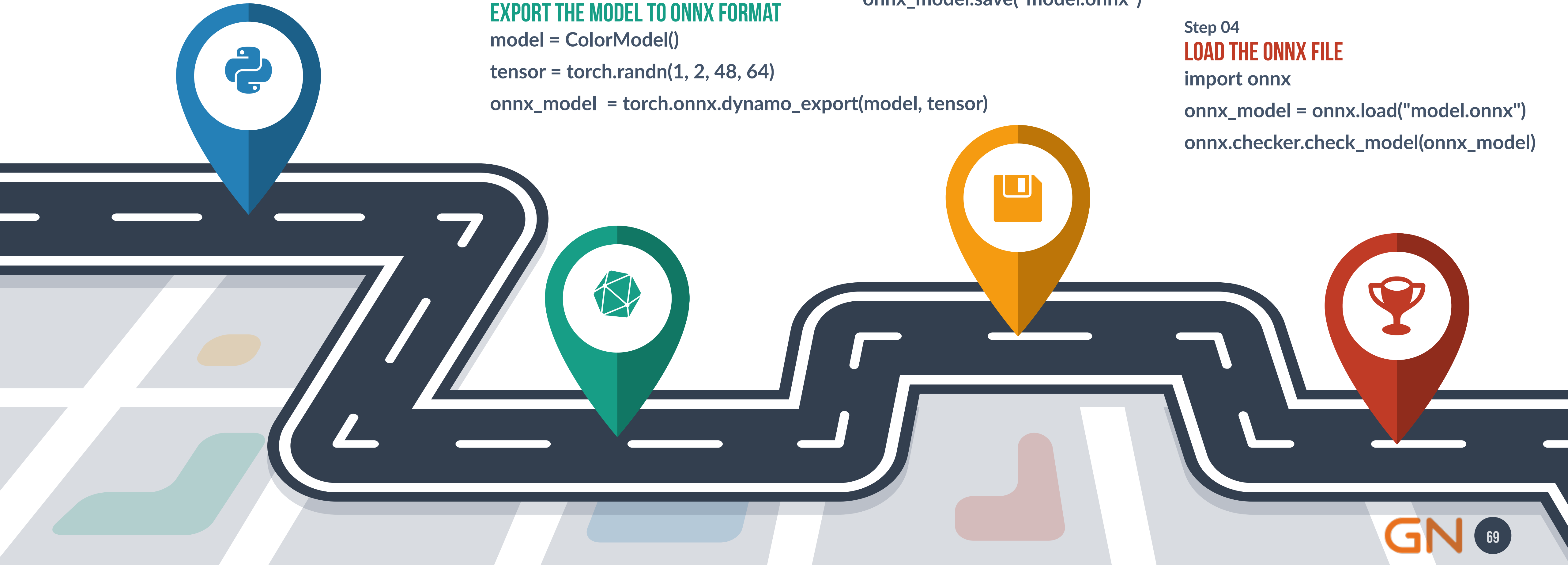
```
model = ColorModel()
tensor = torch.randn(1, 2, 48, 64)
onnx_model  = torch.onnx.dynamo_export(model, tensor)
```

Step 03
SAVE THE ONNX MODEL

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

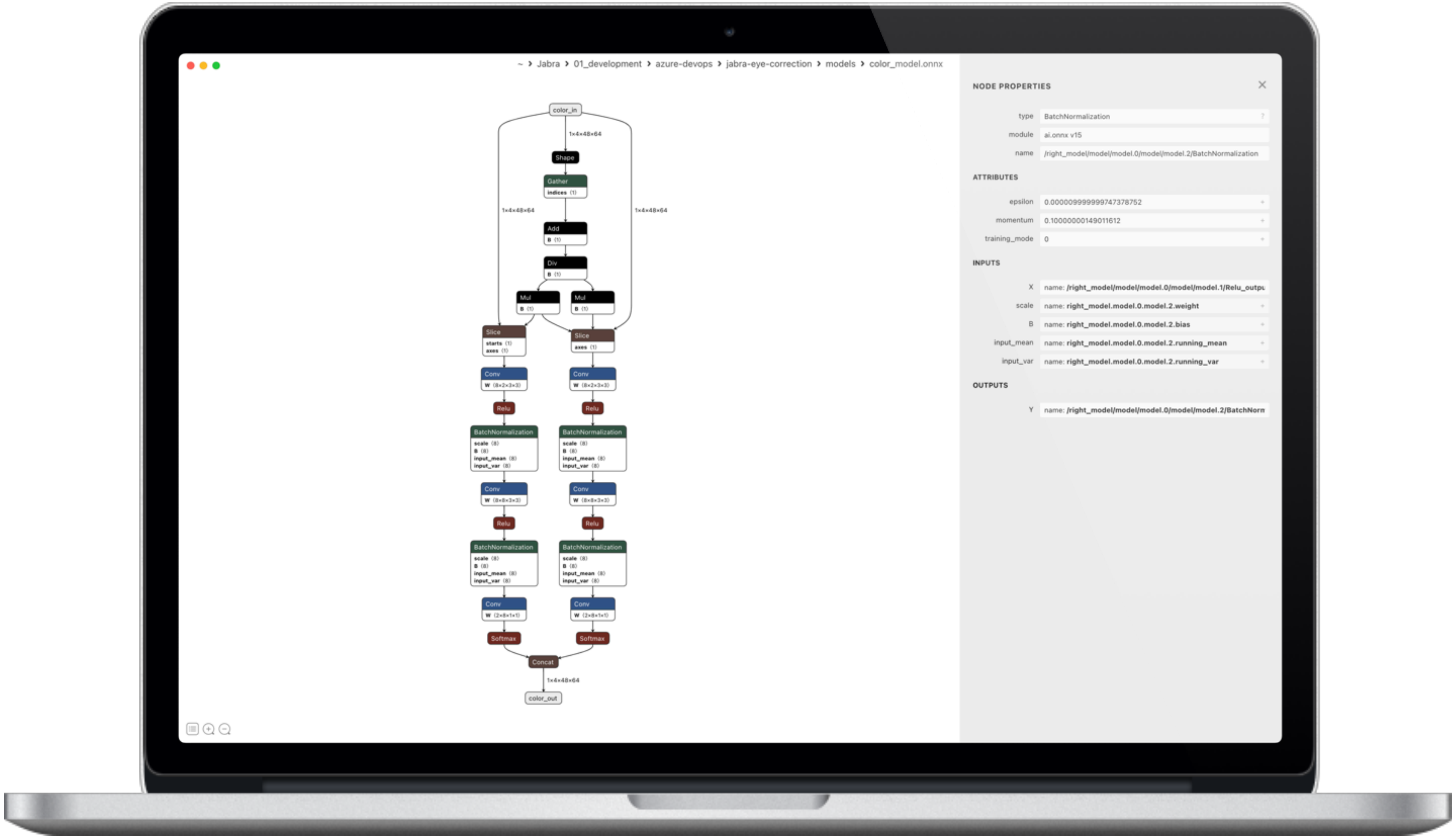
Step 04
LOAD THE ONNX FILE

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



PYTORCH TO ONNX

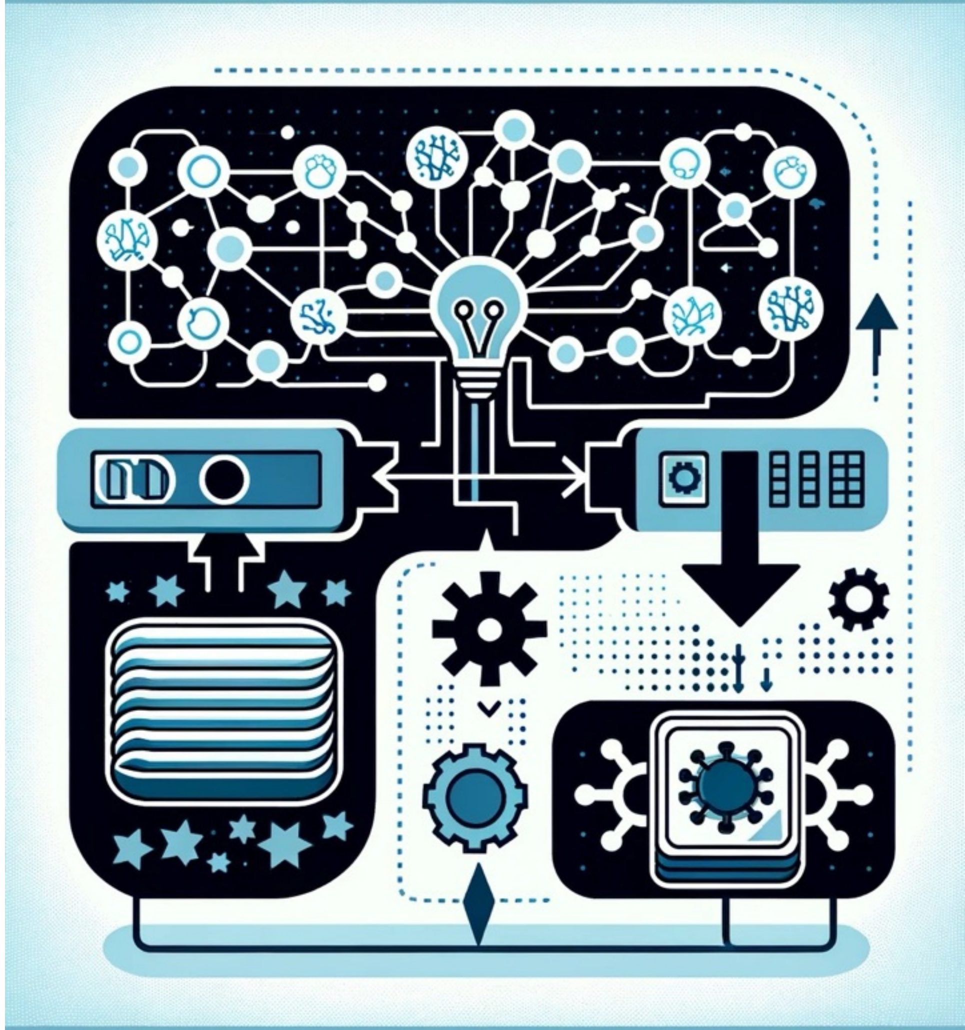
Visualize the ONNX model graph using Netron app



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

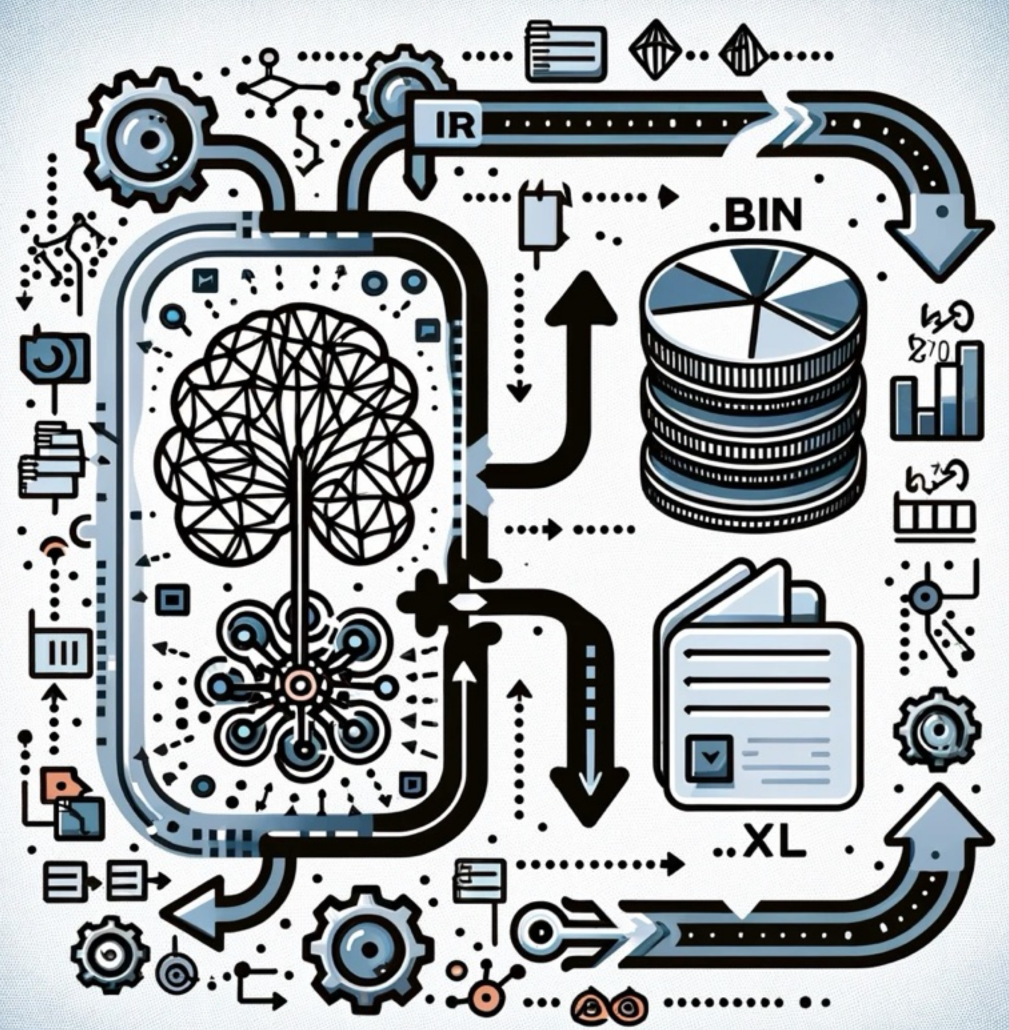
Conversion Steps

ONNX Model → Model Optimizer (.xml/.bin) → Model Compiler (.blob) →

Config file → Model Optimizer

Weights file → Model Optimizer

Luxonis OAK-1 Max Camera



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

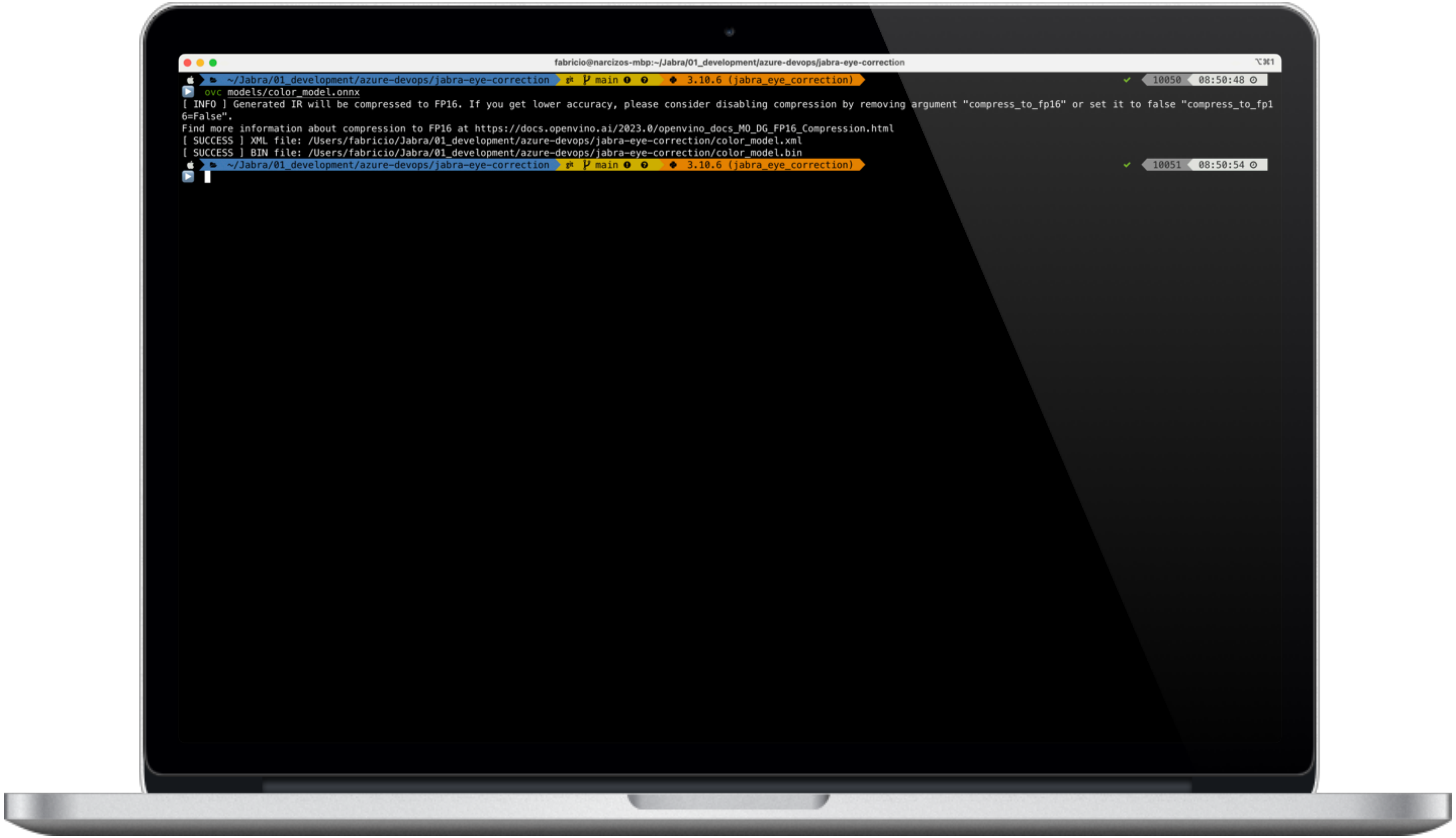
It defines the input/output tensor shape and whether it uses a *Planar Layout* (CHW) or an *Interleaved Layout* (HWC).

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



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

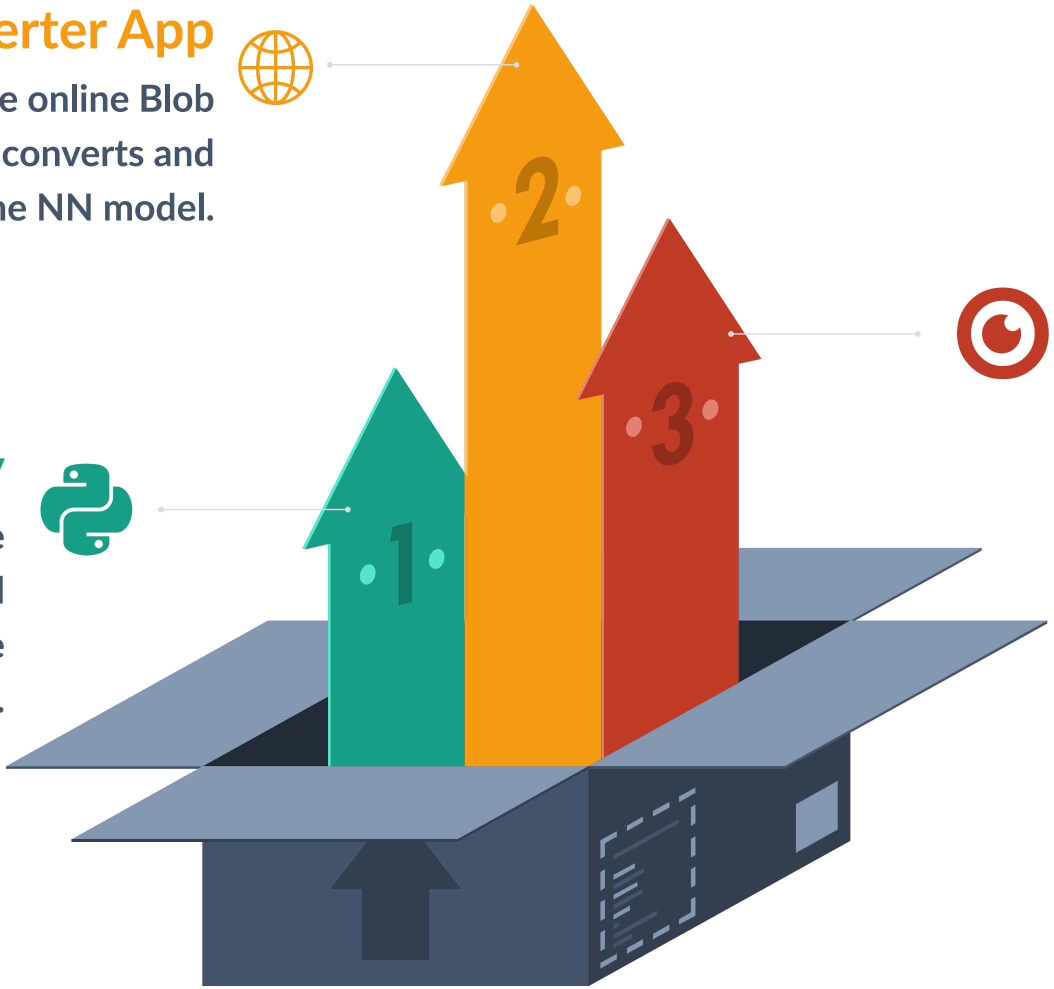
There are a few options to compile models to Edge AI

Online Blob Converter App

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

Blob Converter Library

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



Local Compilation

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

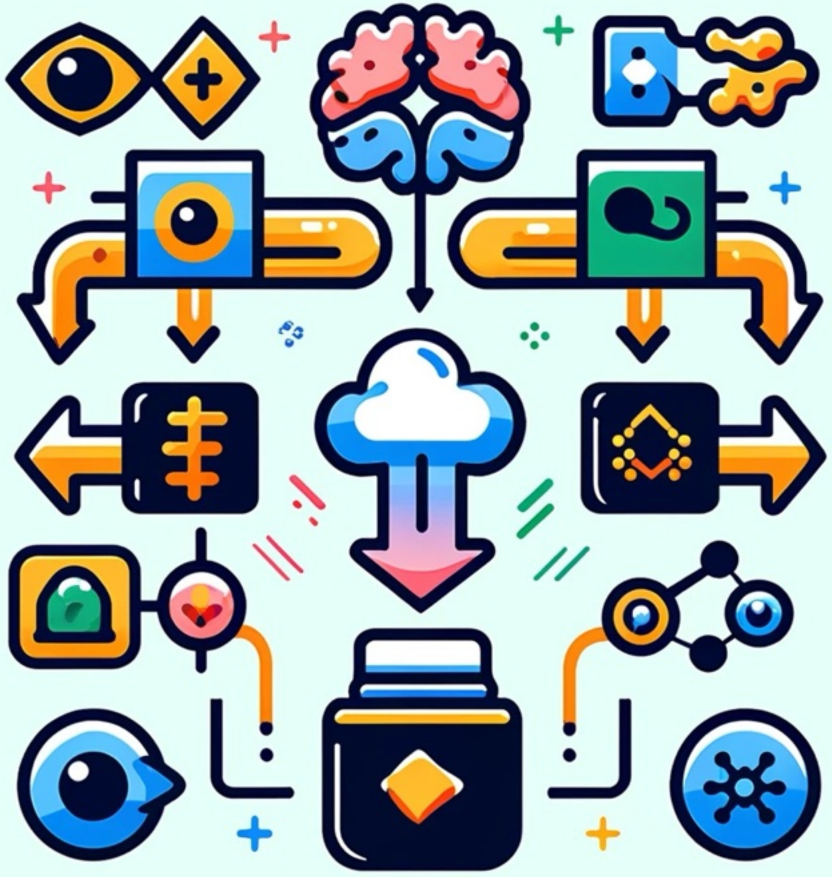
BLOB CONVERTER LIBRARY

pip install blobconverter

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

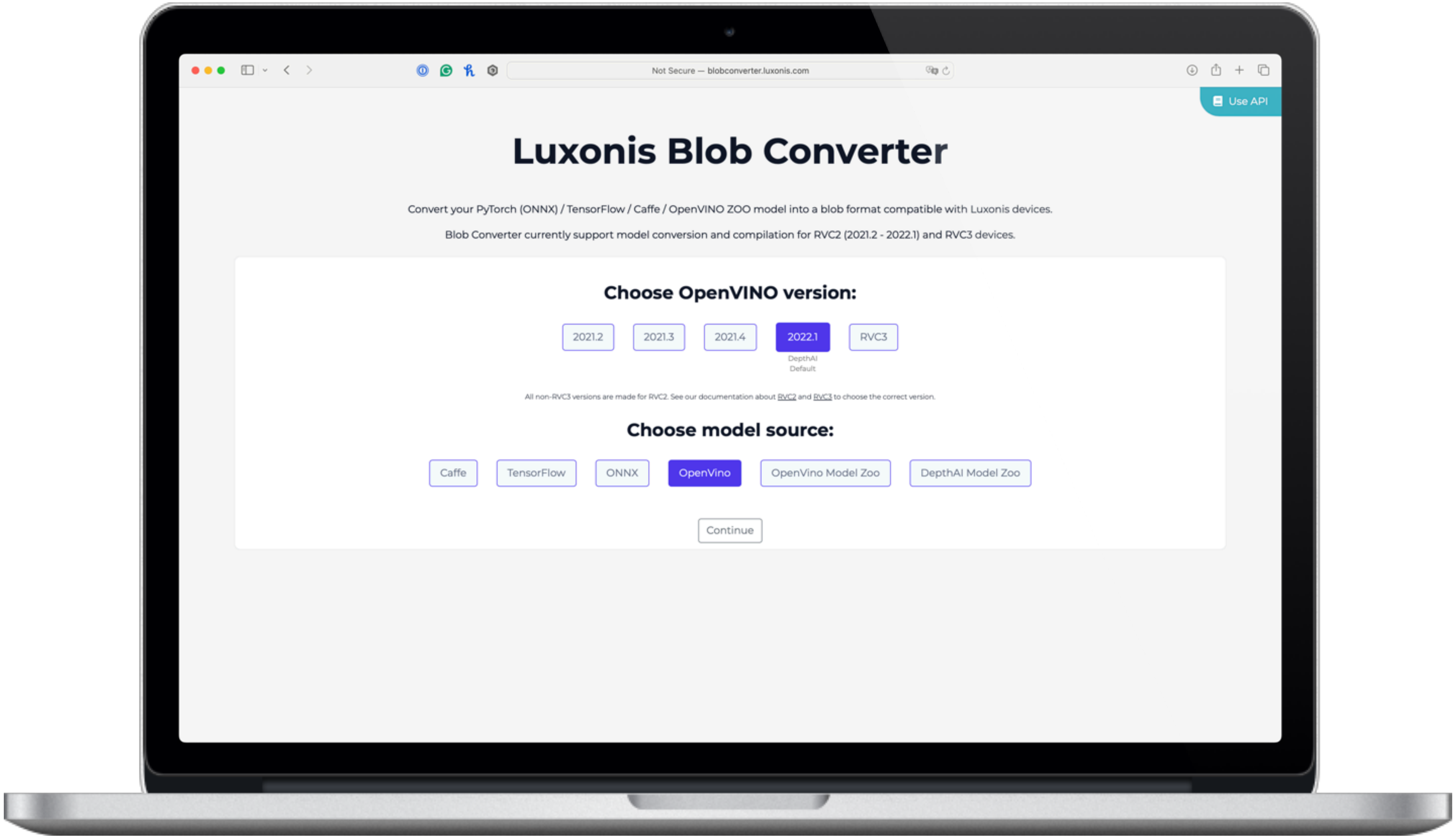
```python
import blobconverter

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



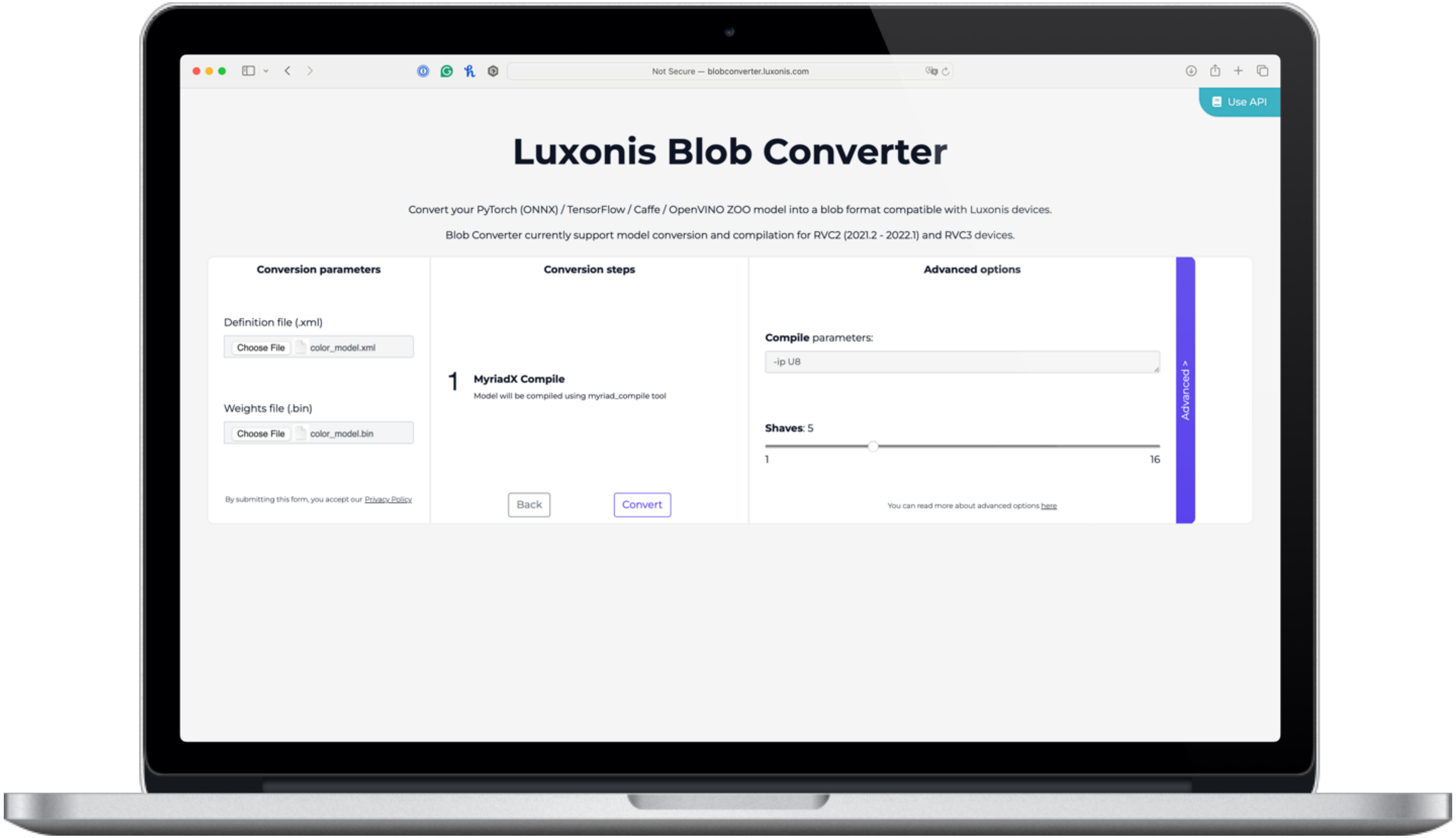
OPENVINO'S COMPILE TOOLS

Online Blob Converter App



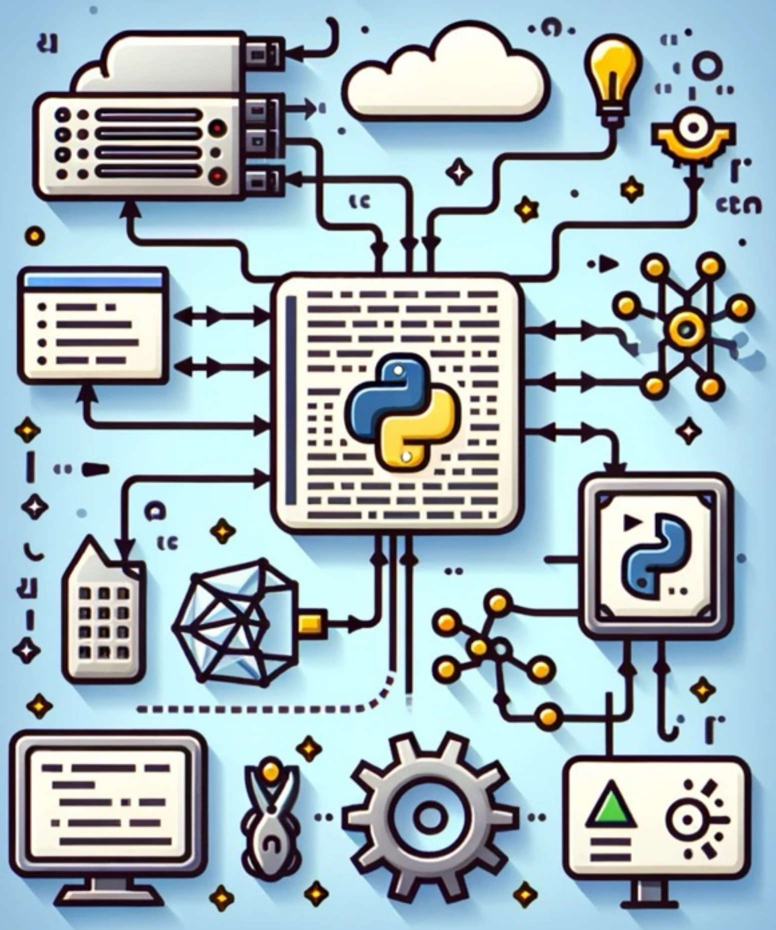
OPENVINO'S COMPILE TOOLS

Online Blob Converter App



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.

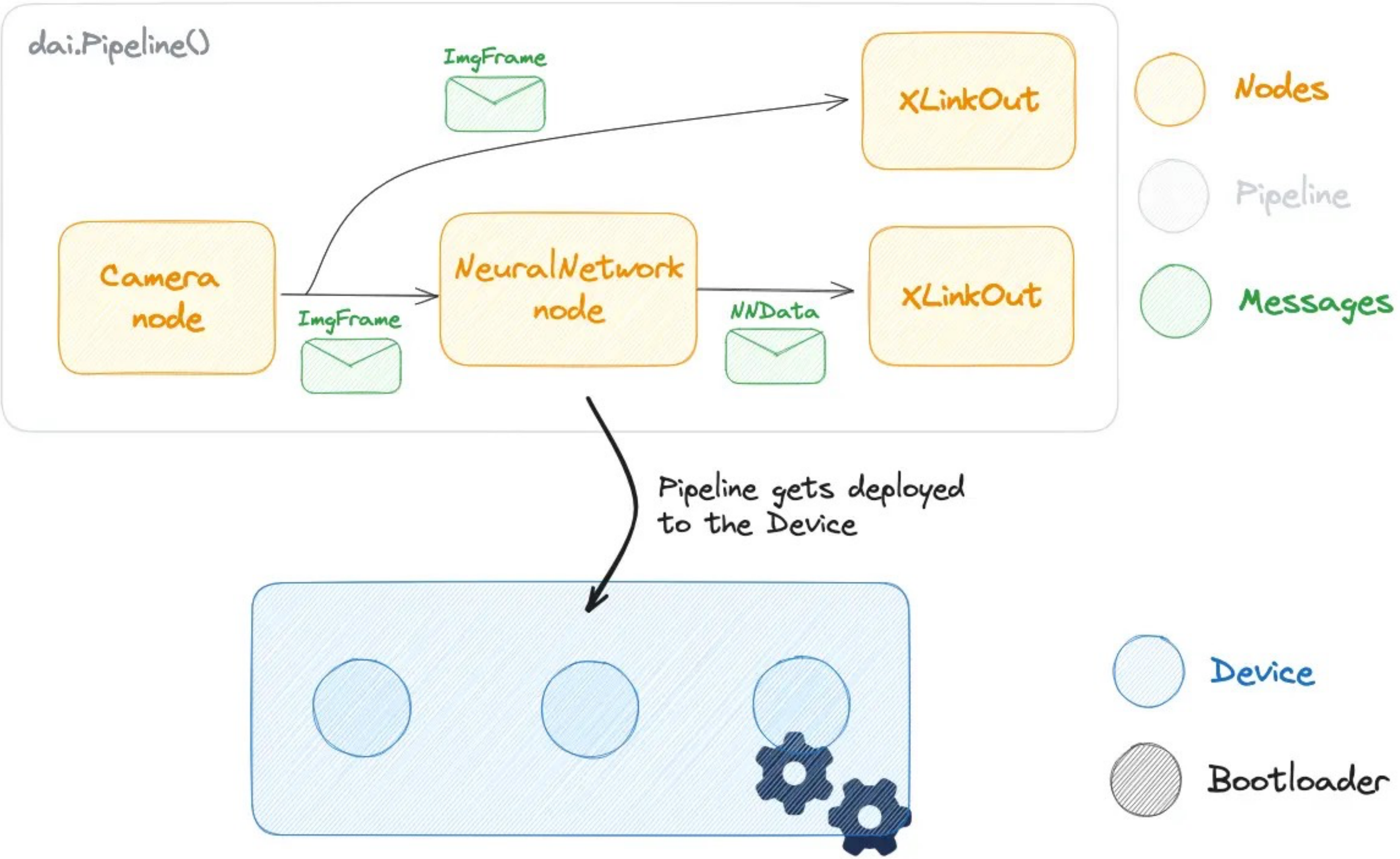
XLinkOut

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

HTTPS://WWW.LUXONIS.COM

DEPLOYING CUSTOM MODELS

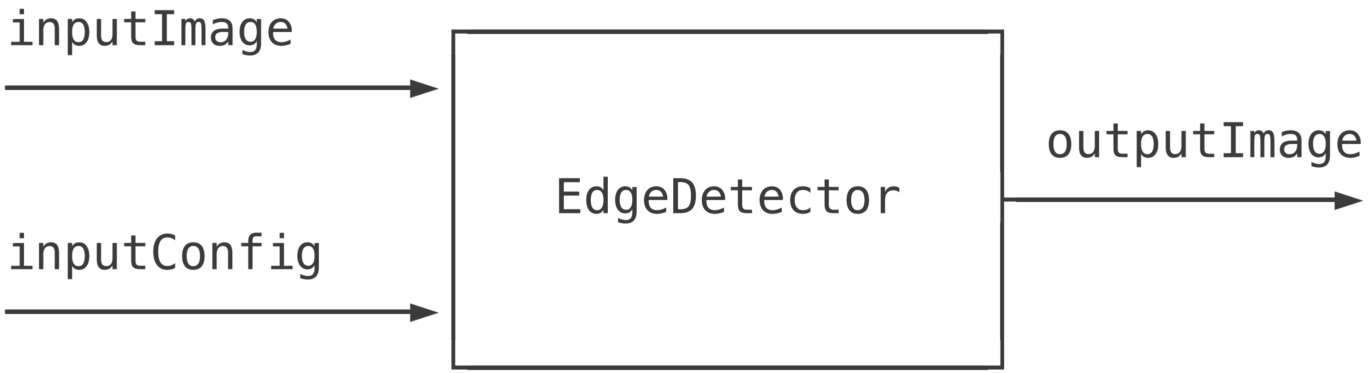
What is DepthAI SDK



DEPLOYING CUSTOM MODELS

Nodes

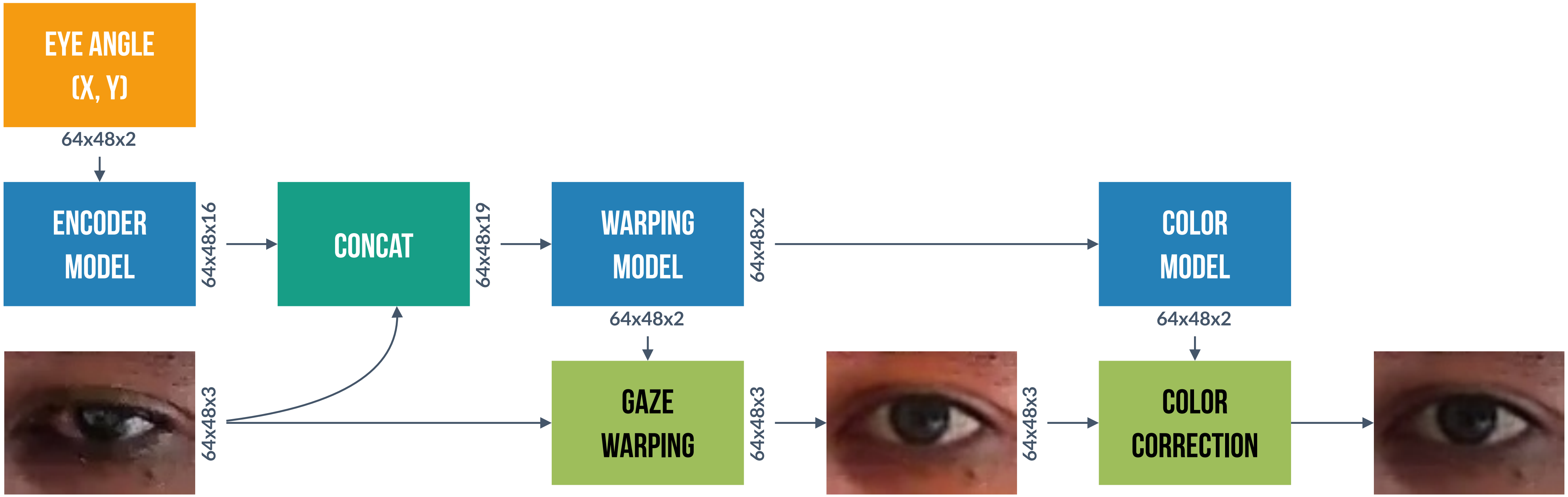
Nodes serve as a building block when populating the Pipeline. They offer specific functionality on the DepthAI, along with a set of configurable properties and inputs/outputs.



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

THANK YOU!

MULTIMODAL AI FOR EDGE AI

CVPR 2024 Tutorial

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

Seattle, WA, USA

TUTORIAL AGENDA

1 Multimodal Perception

2 Gaze Correction

3 Hand Gestures Recognition

4 Sound Localization

5 Demos

GAZE CORRECTION

INTRODUCTION AND CHALLENGES OF GAZE CORRECTION

GAZE CORRECTION

Overview

Gaze correction refers to using computer vision or artificial intelligence techniques to adjust the apparent direction of a person's gaze in digital video communication.

Enhanced Engagement

It provides more natural and engaging interaction by simulating real eye contact

Increased Attention

It can help maintain attention and focus during hybrid and virtual meetings

Professional Appearance

It ensures that speakers appear to be addressing their audience directly in virtual presentations

Improved Comprehension

Speakers may enhance the audience's ability to understand the discussed content

VIRTUAL MEETING EXAMPLE

Original Video

VIRTUAL MEETING EXAMPLE

Gaze Correction

VIRTUAL MEETING EXAMPLE

Original Video vs. Gaze Correction

CHALLENGES IN EDGE AI

Implementing Gaze Correction in Edge AI

Running complex AI models for gaze correction requires efficiently using the limited AI components resources without compromising performance

Real-Time

It must process video in real-time to ensure the gaze is corrected without noticeable lag

Model Optimization

It must be compressed and optimized without significant loss of accuracy

User Diversity

The model must be trained on diverse datasets to work accurately across different ethnicities, genders and ages

Integration

The gaze correction application must integrate seamlessly with existing video conferencing platforms and camera hardware

CHALLENGE OF GAZE CORRECTION

Technical Challenges

CHALLENGE OF GAZE CORRECTION

Technical Challenges

HOW GAZE CORRECTION WORKS

Deep Learning for Gaze Correction

GAN

Generative Adversarial Networks

GANs are used in gaze correction to generate realistic eye images that match the desired gaze direction.

WARPING

Warping Neural Networks

Warping techniques adjust the eye region in an eye region to redirect the gaze, creating the impression of eye contact.

SYSTEMATIC REVIEW ON GAZE CORRECTION MODELS

Recent Published Models

- *Isikdogan et al. (2020),* Eye Contact Correction using Deep Neural Networks.
- *Hsu et al. (2019),* Look at Me! Correcting Eye Gaze in Live Video Communication.
- *He et al. (2019),* Photo-Realistic Monocular Gaze Redirection using Generative Adversarial Networks.
- *Kononenko et al. (2018),* Photorealistic Monocular Gaze Redirection using Machine Learning.
- *Kaur and Manduchi (2021),* Subject Guided Eye Images Synthesis with Application to Gaze Redirection.

JABRA GAZE CORRECTION MODEL FOR EDGE AI

JABRA EYE CORRECTION

Jabra PanaCast 20

Jabra Eye Correction model simulates direct eye contact between participants of online meetings, enhancing the sense of engagement and personal connection during remote interactions. The model can be deployed directly in our Jabra Business Collaboration products.

Speed

The model runs at 30 frames per second with eye images of 64x48 resolution

Eye Shape

The model generates good eye feature shape, especially iris contour and eyelid shape

Color Composition

The model reconstructs the eye colors and skin color in the processed eye region

HOW TO BUILD A GAZE CORRECTION APPLICATION?

Edge AI Deployment

05 — Integration with Edge Devices
Currently, the warping-based model is deployed into Luxonis OAK-1 Pro cameras and Jabra ExPy Environment

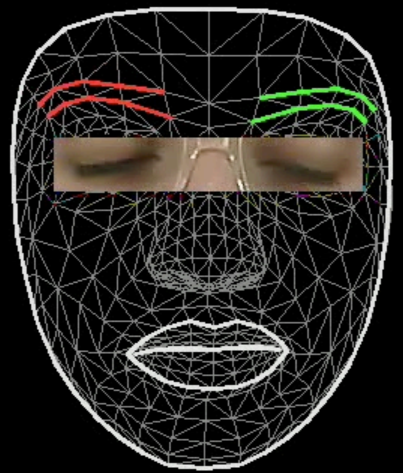
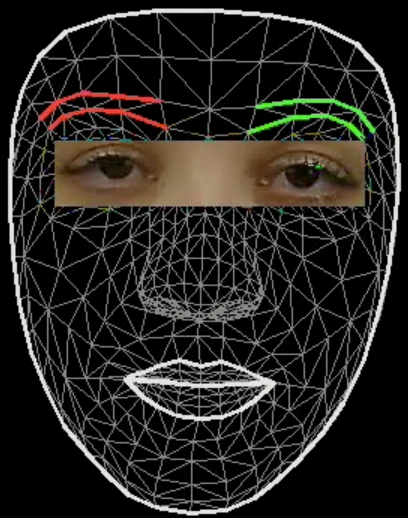
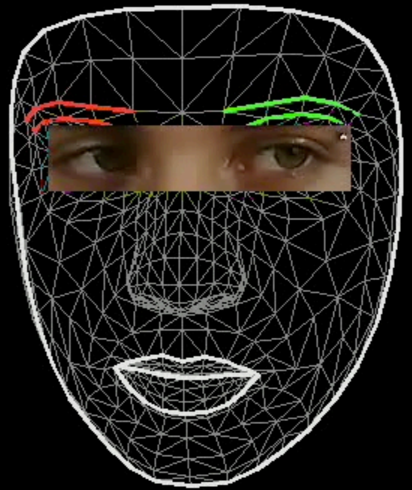
04 — Model Optimization
We have optimized the trained models using OpenVINO Toolkit

03 — Model Training
The models were implemented using PyTorch Lightning

02 — Model Selection
We have implemented two versions of gaze correction models, i.e., GAN-based and warping-based models

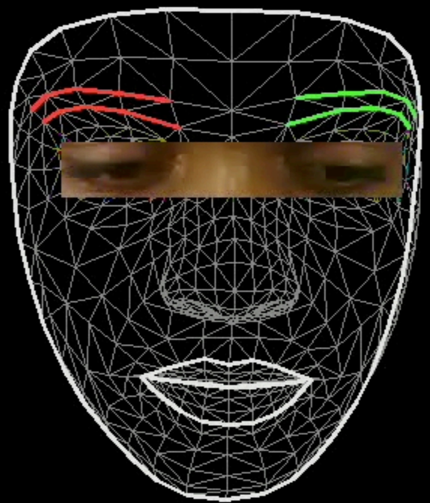
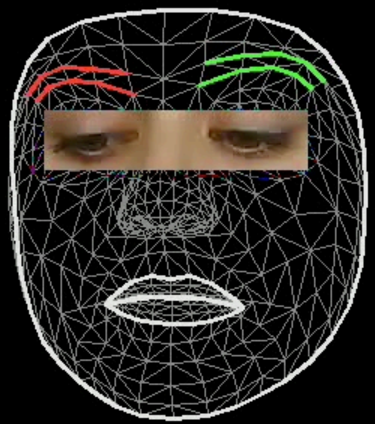
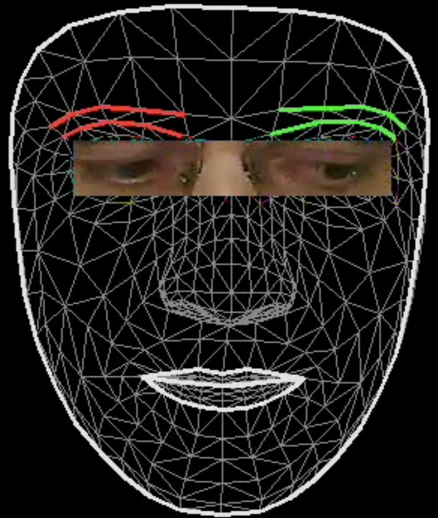
01 — Data Collection and Annotation
A dataset from various angles, lighting conditions, and with different eye shapes and colors


UNITYEYES-JABRA DATASET
Eye Contact Dataset with Synthetic Data

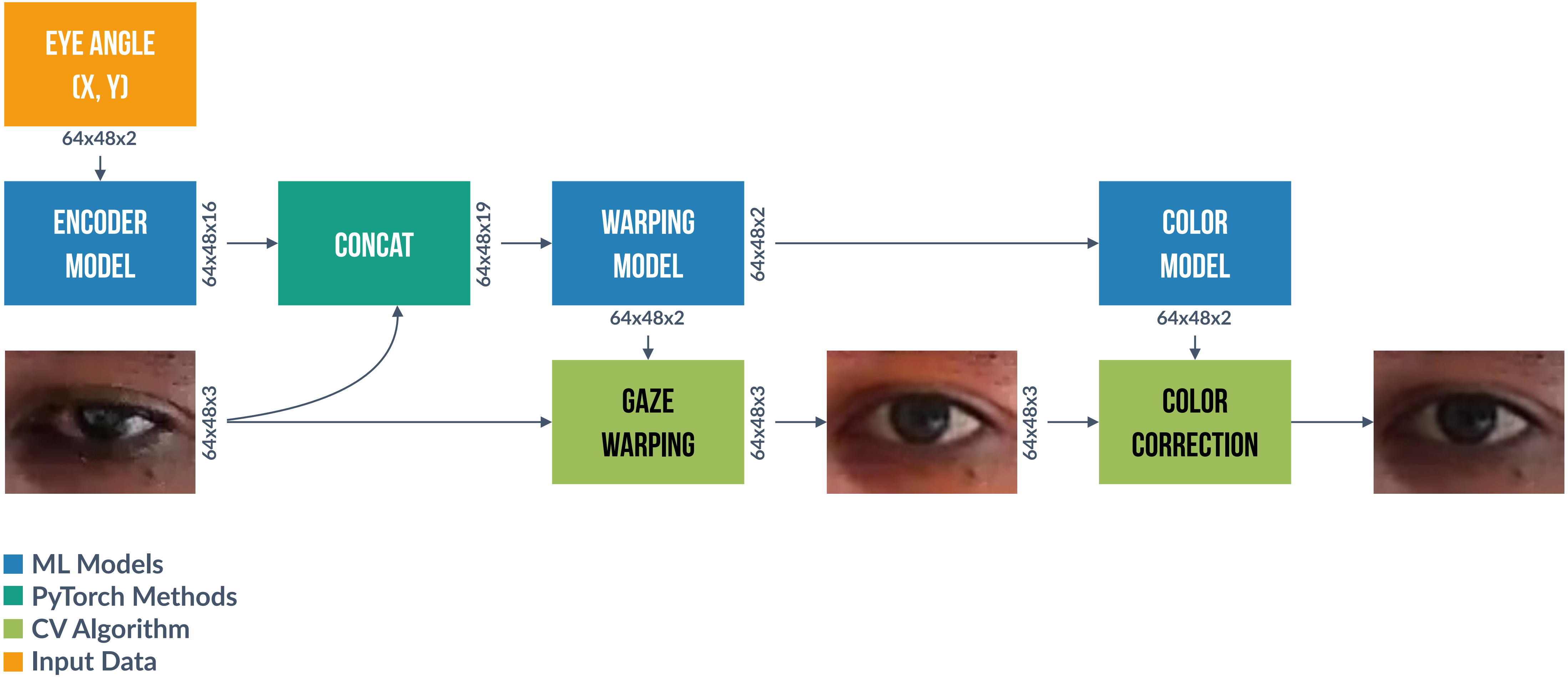
IFSP-JABRA DATASET

Eye Contact Dataset with Real Data

JABRA GAZE CORRECTION MODEL ARCHITECTURE

JECModel



JABRA GAZE CORRECTION MODEL

Video Example (Gaze Correction)

JABRA GAZE CORRECTION MODEL

Video Example (Gaze Correction + Beautification)

HAND GESTURES RECOGNITION

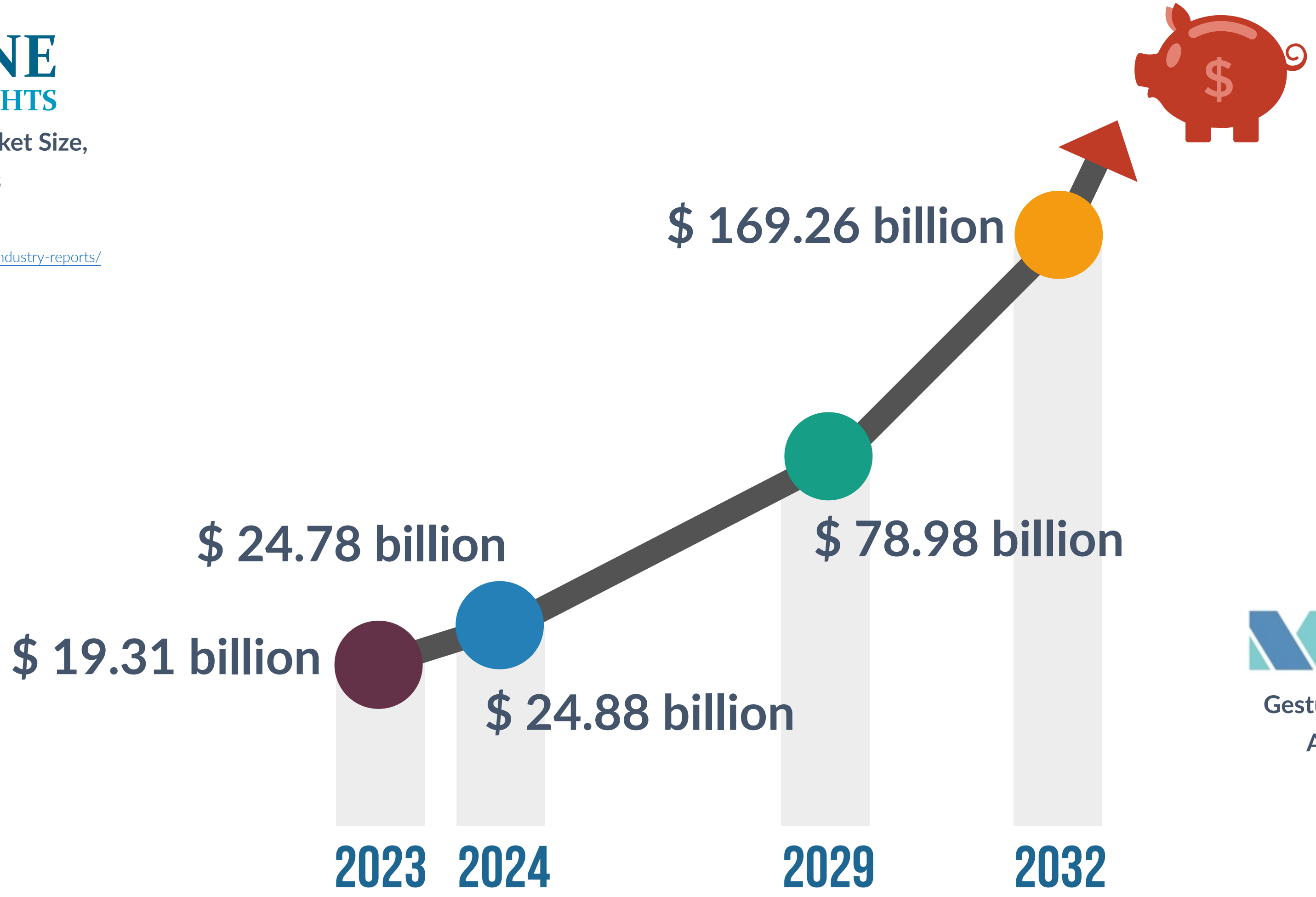
WHY HAND GESTURES

Is hand gesture recognition still a thing?

FORTUNE BUSINESS INSIGHTS

Gesture Recognition Market Size, Share & Industry Analysis (2024 – 2032)

https://www.fortunebusinessinsights.com/industry-reports/gesture-recognition-market-100235



https://www.mordorintelligence.com/industry-reports/gesture-recognition-market

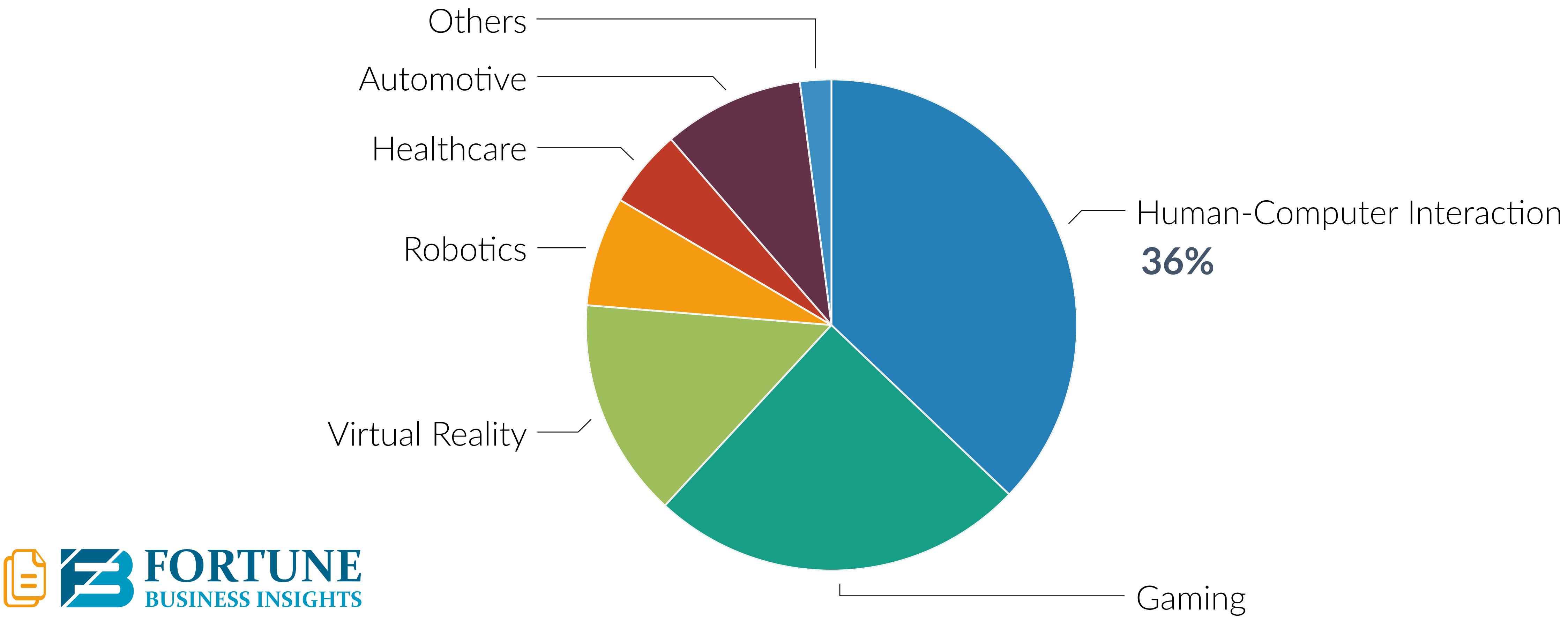
Mordor Intelligence

Gesture Recognition Market Size & Share Analysis - Growth Trends & Forecasts (2024 - 2029)

Global gesture recognition market size estimation and projection

WHY HAND GESTURES

Is hand gesture recognition still a thing?



Global gesture recognition market share, by application segment, 2023

WHY HAND GESTURES

Hand gestures are everywhere

AGRICULTURE AND INDUSTRY

Equipment Operation
On-Site Inspections

HEALTHCARE

Surgical Assistance
Rehabilitation
Assistive Devices

SMART HOME AND ENTERTAINMENT

Interactive Media Control
Home Automation and Control
Security System
Access Control

ROBOTICS AND AUTOMATION

Service Robots
Collaborative Robots
Machine Control
Quality Inspection
Training Simulations

HUMAN-COMPUTER INTERACTION

Assistive Technology
Interactive Learning
Virtual Classrooms
Special Education

FITNESS AND SPORTS

Exercise Monitoring
Sports Training

PUBLIC INTERACTION

Interactive Displays
Kiosks and Information Terminals
Exhibits and Museums
Transportations Hubs

AUTOMOTIVE

In-Car Infotainment Systems
Driver Assistance

VIRTUAL REALITY AND GAMING

Virtual Reality (VR)
Augmented Reality (AR)

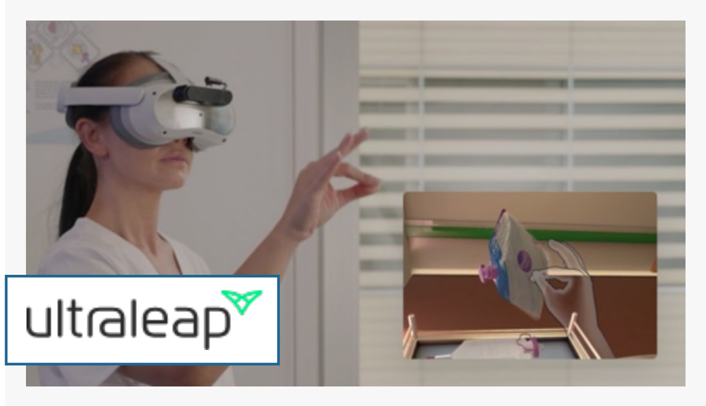
CONSUMER

Smartphones and Tablets
Virtual Assistants
Selfie cameras

Several Fields of Applications

HAND GESTURE PRODUCTS

Example in different industries



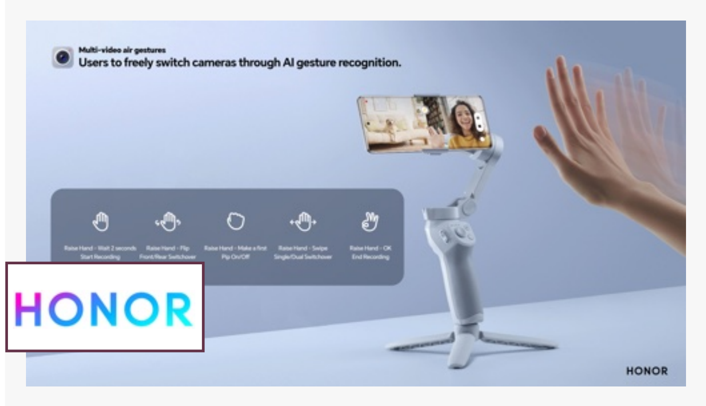
Leap Motion Controller 2

HoloLens 2

Echo Show

Gesture Control Armband

AIR Neo Selfie Pocket Drone



HONOR Cellphone Camera

BENEFITS OF HAND GESTURES

Overview

HG supports immersive experiences of entertainment and control by providing more natural and engaging ways to interact with digital environments, systems and devices.

Enhances User Experience

Provides multimodal interaction methods, making systems more user-friendly and versatile.

Promotes Accessibility

Offers alternative communication methods for individuals with disabilities, enhancing inclusivity and usability.

Enables Touchless Control

Enables hygienic interaction by eliminating the need for physical contact, ideal for public and shared environments.

Increases Efficiency

Allows for quick and efficient execution of commands through simple gestures, reducing reliance on traditional input devices.

HAND-BASED TECHNOLOGY

General view

Hand-based technology uses cameras or other sensors to capture the users' hand gestures and movements.
Algorithms or Machine Learning models then analyze and interpret the hand poses or performances from the captured data.

General View for Hand Gesture Technology

HAND GESTURE RECOGNITION

Looking into the pipeline process

HAND DETECTION

HAND TRACKING

GESTURE RECOGNITION

DATA AQCUISITION

Capturing data using one or more sensors

PREPROCESSING DATA

Processing the raw data to focus on the relevant part

FEATURE EXTRACTION

Identifying and isolating key characteristics

GESTURE CLASSISIFCATION

Classifying the extracted features into predefined gesture categories

POST PROCESSING

Refining the classification results and mapping the gestures to actions

INTERFACE FEEDBACK

Providing system feedback to the user

HAND GESTURE RECOGNITION

General pipeline process

- Cameras
- Optical Sensors
- Ultrasonic Sensors
- Motion Sensors
- Electromyography Sensors
- Infrared Sensors

TEMPORAL HAND GESTURE CLASSIFICATION (IMAGE BASED)

STATIC

Postures/Poses

Hand Shapes

DYNAMIC

Performed Gestures

Hand Movements

DATA AQCUISITION

Sensor types

PREPROCESSING DATA

FEATURE EXTRACTION

GESTURE CLASSISIFCATION

POST PROCESSING

INTERFACE FEEDBACK

HAND GESTURE RECOGNITION

General pipeline process

Data Cleaning
Normalization
Segmentation
Data Augmentation
Color Space Conversion
Cropping and Resizing

DATA ACQUISITION

PREPROCESSING DATA

Preparing data

FEATURE EXTRACTION

GESTURE CLASSISIFCATION

POST PROCESSING

INTERFACE FEEDBACK

EXAMPLE PROCESS FOR PROCESSING DATA (IMAGE BASED)

SUGGESTED STEPS

Noise Reduction
Background Subtraction
Thresholding
Contour Detection
ROI Extraction
Resizing
Grayscale Conversion

HAND GESTURE RECOGNITION

General pipeline process

Shape-Based
Texture-Based
Keypoint Detection
Geometric
Motion-Based
Spatial and temporal

DATA AQCUISITION

PRE PROCESSING DATA

FEATURE EXTRACTION

Feature types/Techniques

GESTURE CLASSISIFCATION

POST PROCESSING

INTERFACE FEEDBACK

COMMON LIBRARIES / FRAMEWORKS

FEATURE EXTRACTION

OpenCV
scikit-image
Keras
Tensor-Flow
MediaPipe
Kornia (for Pytorch)
dlib
MathLab (Image Processing Toolbox)

HAND GESTURE RECOGNITION

General pipeline process

- Deep Learning Ensembles
- Artificial Neural Networks (ANN)
- 3D Convolutional Neural Networks (3D-CNN)
- Support Vector Machines (SVM)
- Convolutional Neural Networks (CNN)
- Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM)

COMMON AVAILABLE MODELS

FRAMEWORKS / LIBRARIES

Keras

Google Tensor-Flow

PyTorch

scikit-learn

MediaPipe

DATA AQCUISITION

PRE PROCESSING

FEATURE EXTRACTION

GESTURE CLASSISIFCATION

Inference models

POST PROCESSING

INTERFACE FEEDBACK

HAND GESTURE RECOGNITION

General pipeline process

ENHANCE ACCURACY AND RELIABILITY OF THE RECOGNIZED GESTURES

THECNIQUES / TOOLS

Statistical Methods

Confidence Interval Calculation

Minimizing the mean of the squared error

Outlier Detection

Reinforcement Learning

Filtering and Smoothing

Error Correction

Gesture Segmentation

Gesture Mapping

Event Detection Gesture

Adaptation and Learning

DATA AQCUISITION

PRE PROCESSING

FEATURE EXTRACTION

GESTURE CLASSISIFCATION

POST PROCESSING

Recognized gestures

INTERFACE FEEDBACK

HAND GESTURE RECOGNITION

General pipeline process

PERCEPTIVE RESPONSE TO THE USER

MAPPED TO SPECIFIC ACTIONS

Interacting with the UI

Controlling hardware

Sending commands to other applications or devices

Confirm that the gesture has been recognized

Triggering Actions

Saving Data

Feedback Generation

DATA AQCUISITION

PRE PROCESSING

FEATURE EXTRACTION

GESTURE CLASSISIFCATION

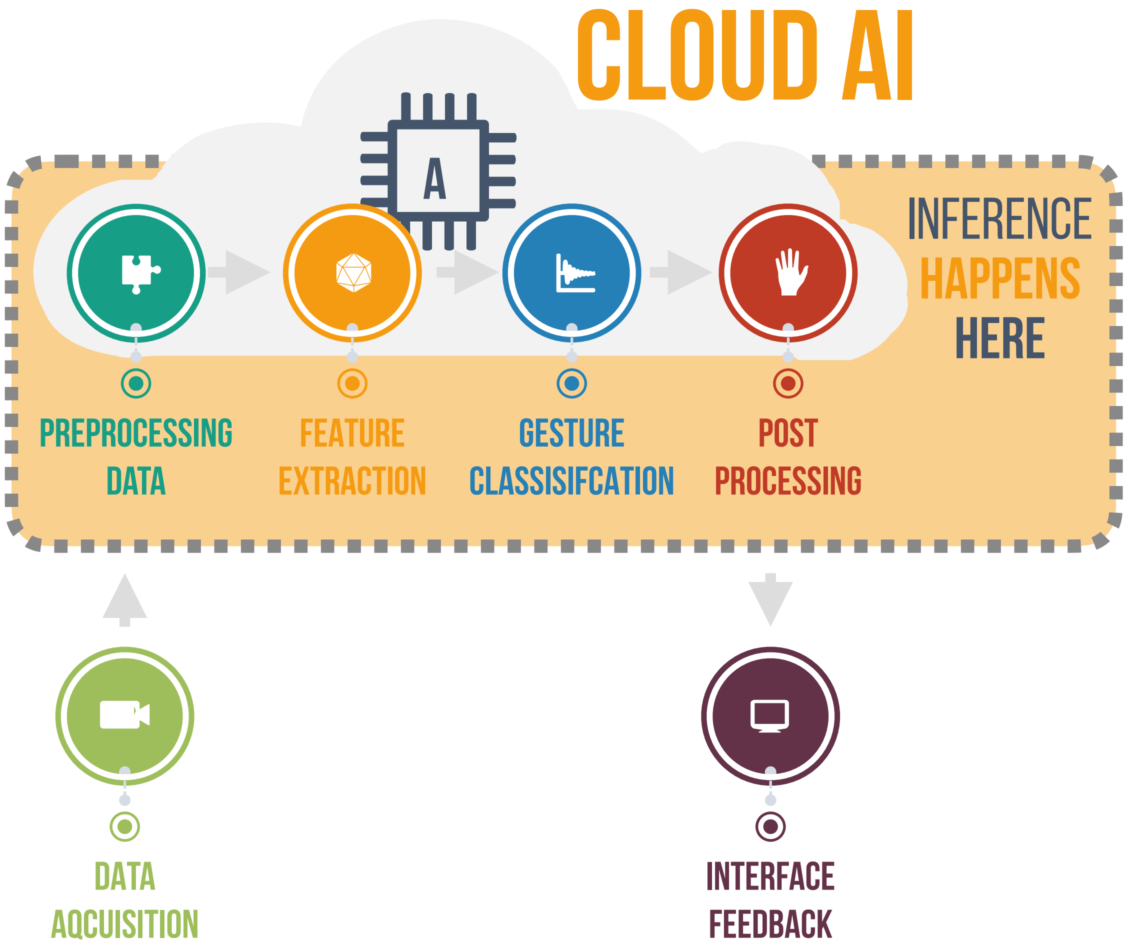
POST PROCESSING

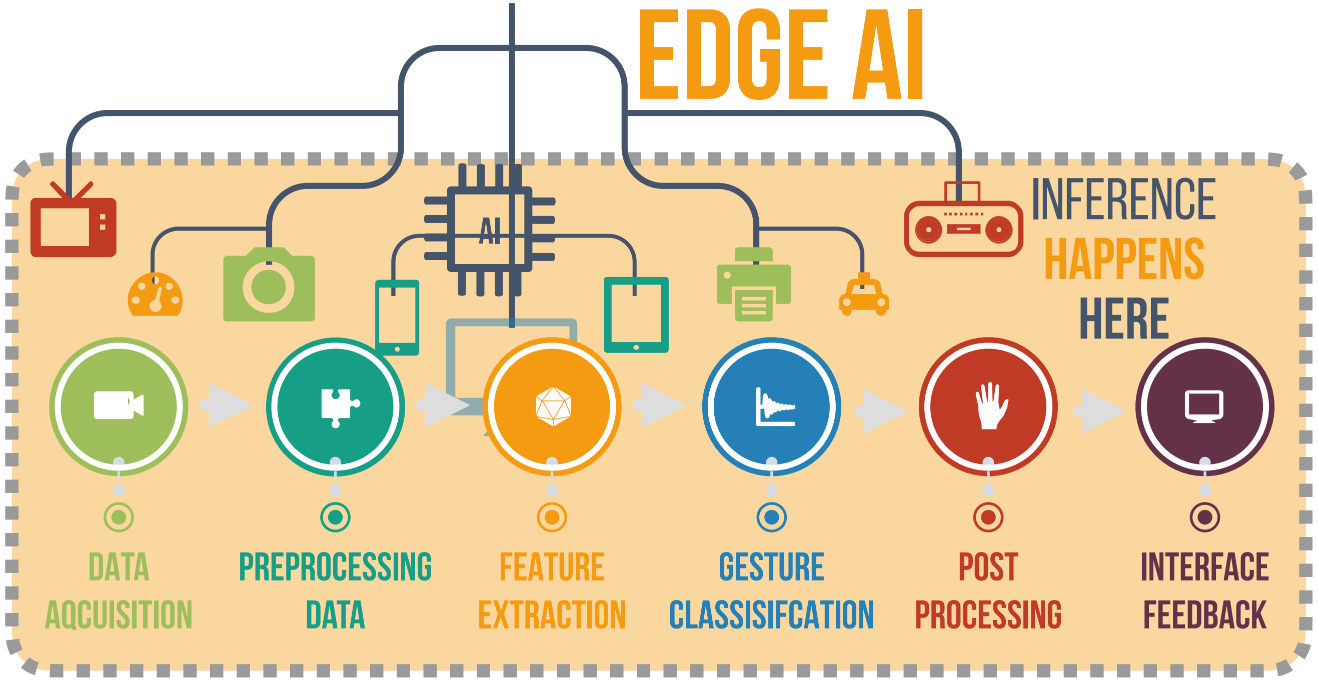
INTERFACE FEEDBACK

Interaction / Response

HAND GESTURE RECOGNITION

Cloud versus Edge AI





CHALLENGES IN HAND GESTURES

Technical problems

Improving performance in these areas is essential for making hand gesture recognition systems more practical, reliable, and widely applicable in real-world scenarios.

Datasets x Data Privacy

Ensuring datasets used for training gesture recognition models are diverse and representativity

Model Size

It must be compressed and optimized without significant loss of accuracy

Real-Time Processing

Low-latency processing to provide immediate feedback and smooth interaction in real-time applications

Gesture Vocabulary

Common shared hand gestures vocabulary for contexts or systems actions

CHALLENGES IN HAND GESTURES

Cross-cutting problems

The most critical challenges in hand gesture recognition today include

HG Education

Is it enough to rely on users' experience and intuitiveness?

Fluidity

Depends on the perfect integration between the user and the system

Cultural Prism

Hand gesture recognition must account for the cultural prism, as the meaning and interpretation of gestures can vary significantly across different cultures.

Shared Vocabulary

A lack of shared vocabulary in hand gesture recognition can lead to inconsistencies and misunderstandings, as different systems and users may interpret gestures differently.

HAND GESTURE EDGE AI DEMO
Volume Control

CAMERA
Camera Luxonis OAK-1 MAX

AI MODEL FORMAT
MyriadX blob format

PALM DETECTOR / HAND LANDMARK TRACKING
Google MediaPipe (Blob)

SOUND LOCALIZATION

SOUND LOCALIZATION
Motivation

Sound location models involve identifying the spatial position of sound-emitting objects within an image or a video to localize auditory cues.

Current Experience
A sound location model that incorporates direction of arrival and head/body detection.

ML Models
Developing a separate machine learning model tailored explicitly for audio and video.

Multimodal Interaction
Find a multimodal solution or application with one neural network that inputs both audio and video components.

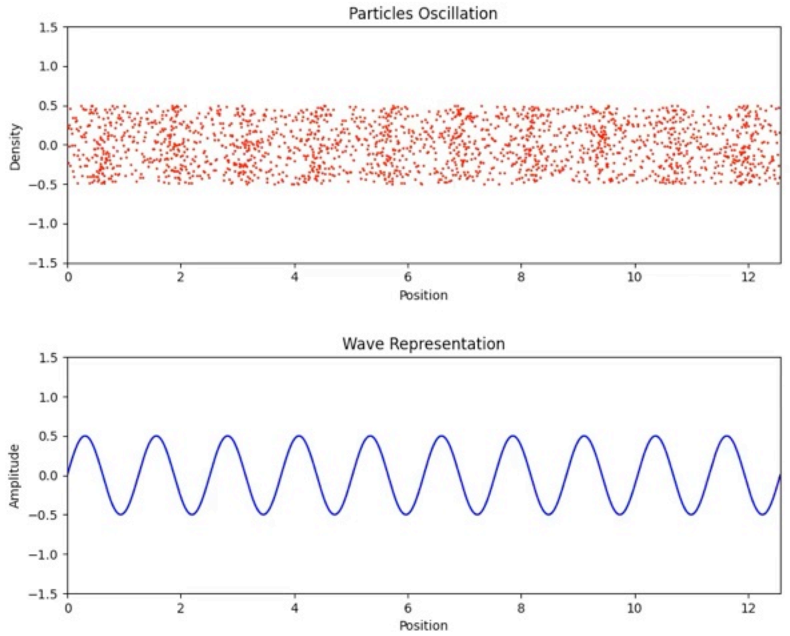
Edge AI
Optimizing the sound location model to be specifically tailored for efficient and effective use on edge devices

AUDIO PROCESSING
Modality

01 Sound is produced by vibration of an object.

02 This causes air molecules to oscillate.

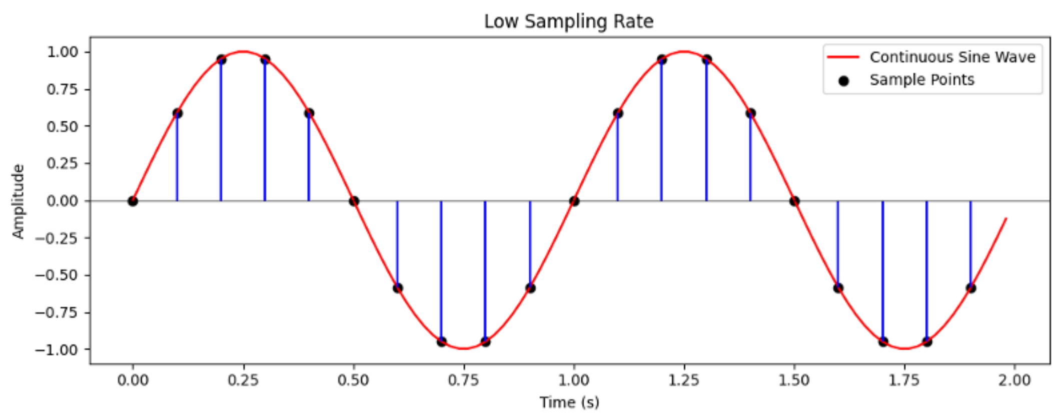
03 Change in air pressure into the wave.

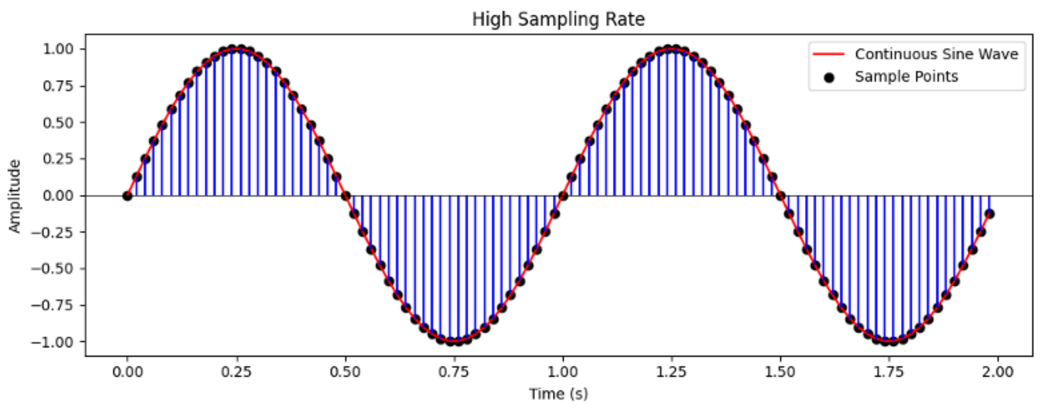


PARTICLES AND WAVE OSCILLATION

AUDIO PROCESSING
Sampling

“Sampling in audio processing involves capturing and converting continuous audio signals into discrete digital data points at regular intervals.”





Represented with sample points

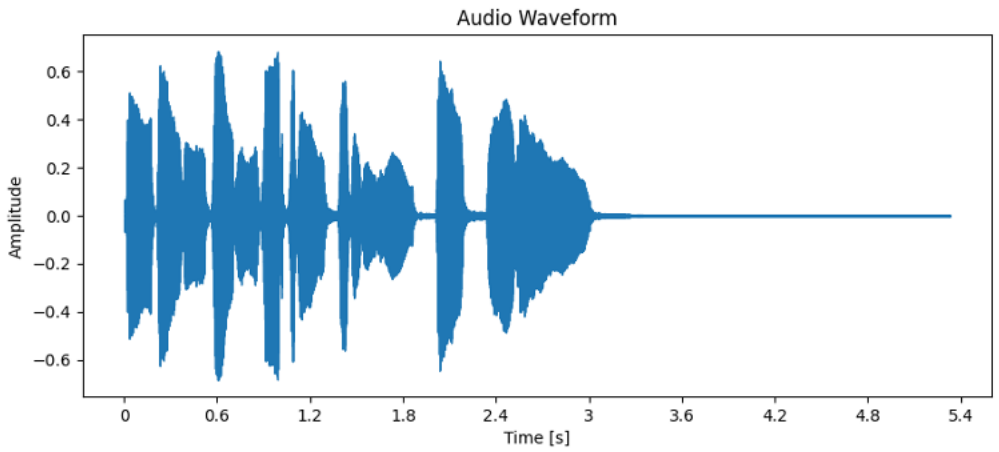
Better quality with higher sampling rate

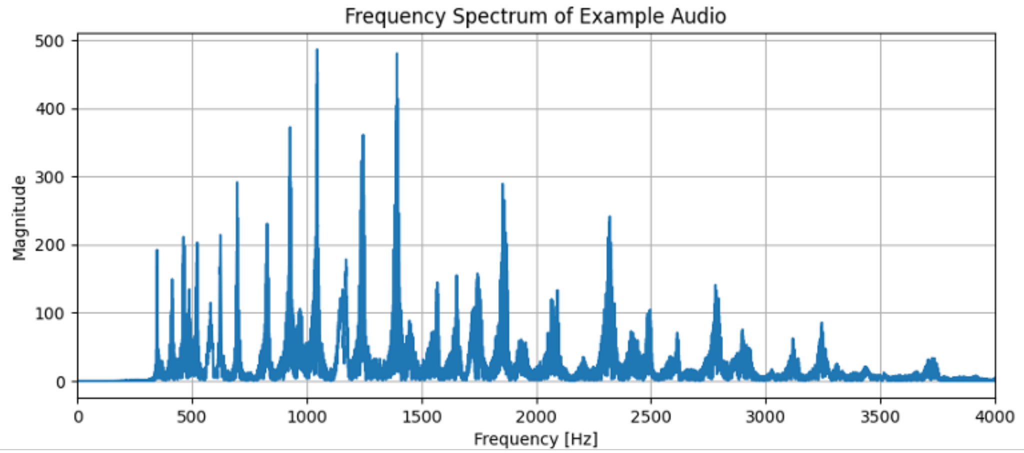
Similar to pixel in images

Quantization

AUDIO REPRESENTATION IN DIFFERENT DOMAINS

Examples



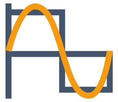


Waveform

Waveform is a graphical representation of the audio signal in the time domain.

Frequency Spectrum

Frequency Spectrum is obtained using the Fast Fourier Transform.

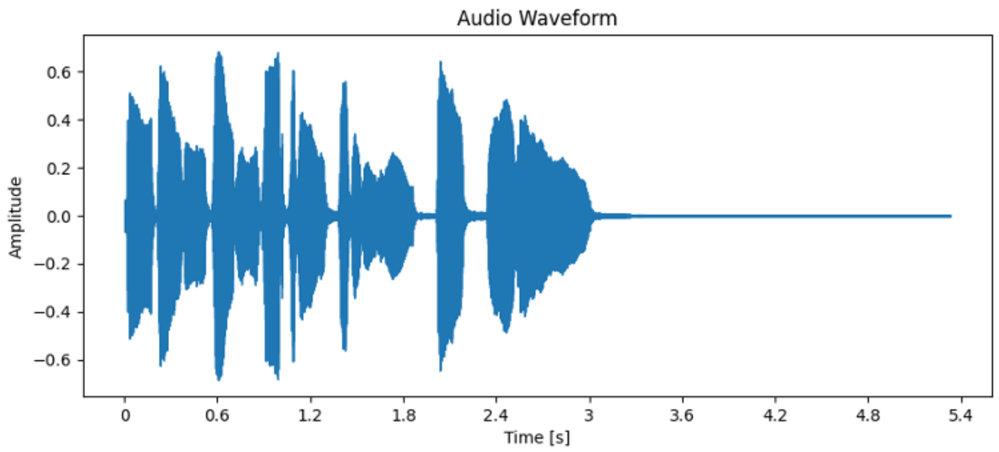


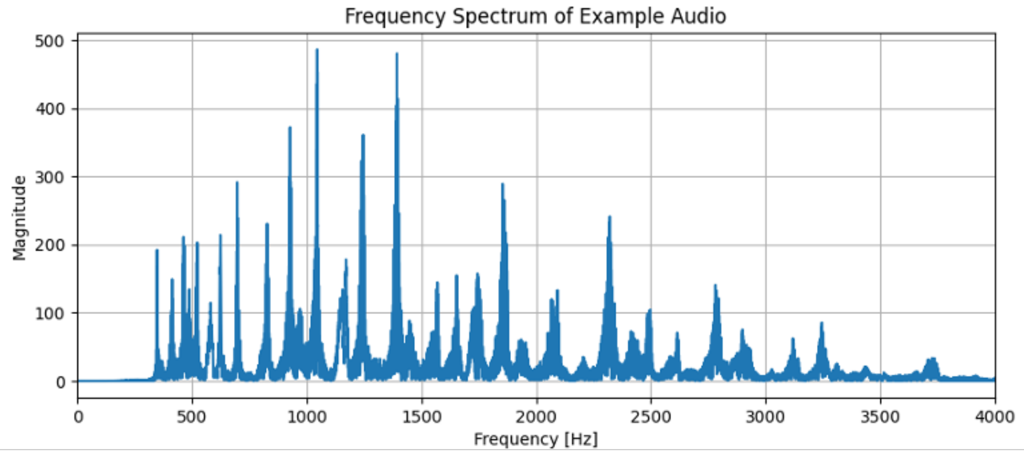
FFT

Fast Fourier Transform is a representation in the Frequency domain.

AUDIO REPRESENTATION IN DIFFERENT DOMAINS

Examples





Waveform

Waveform is a graphical representation of the audio signal in the time domain.

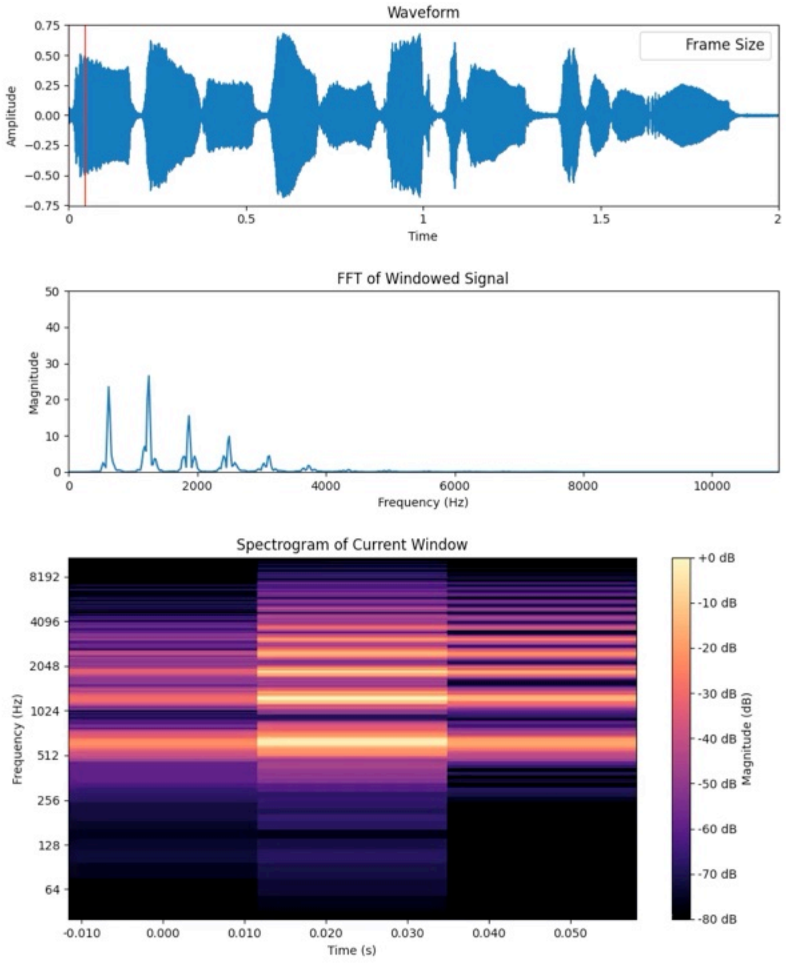
Frequency Spectrum

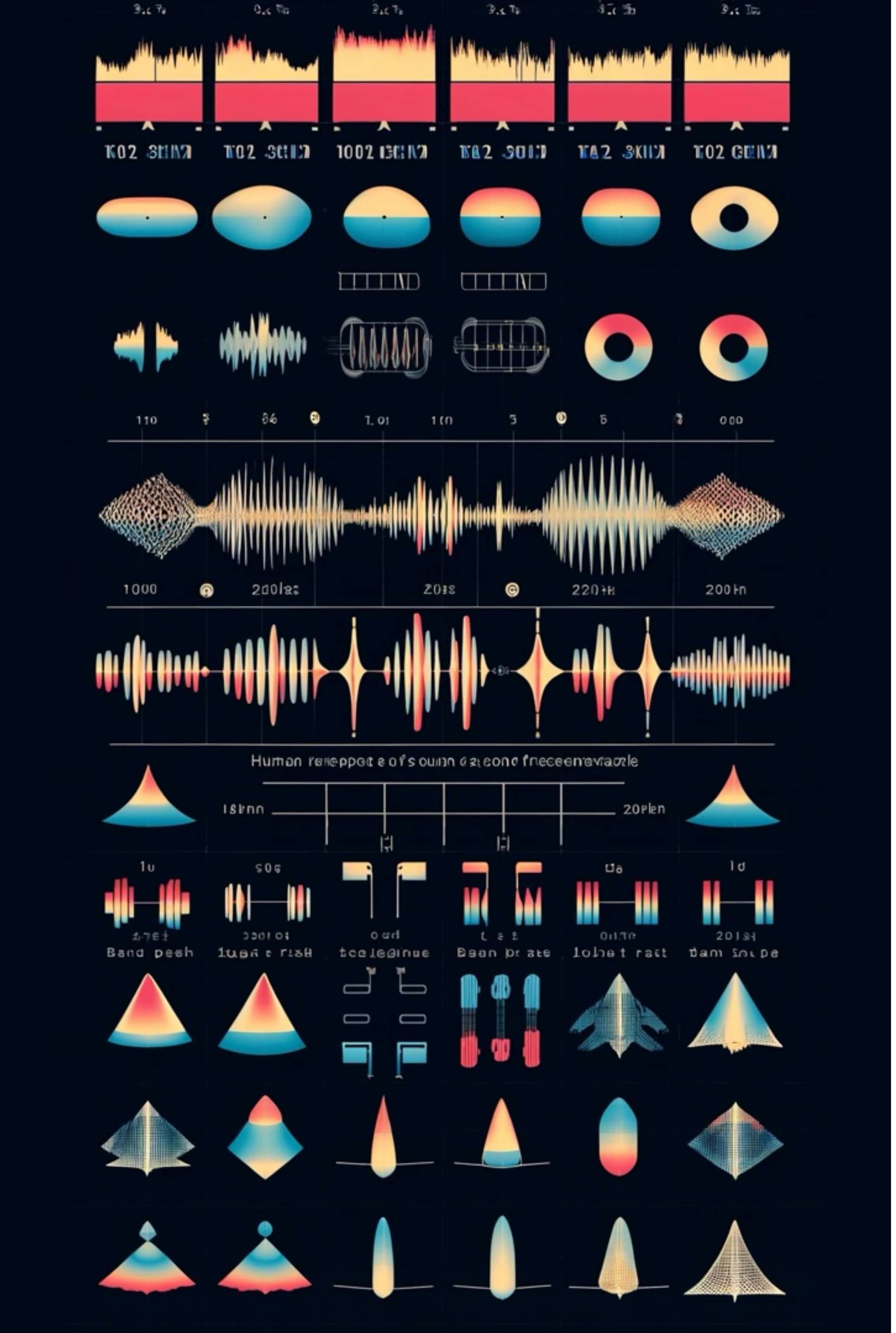
Frequency Spectrum is obtained using the Fast Fourier Transform.

FFT

Fast Fourier Transform is a representation in the Frequency domain.

TIME-FREQUENCY DOMAIN
Windowing to Spectrogram





PERCEPTION OF SOUND

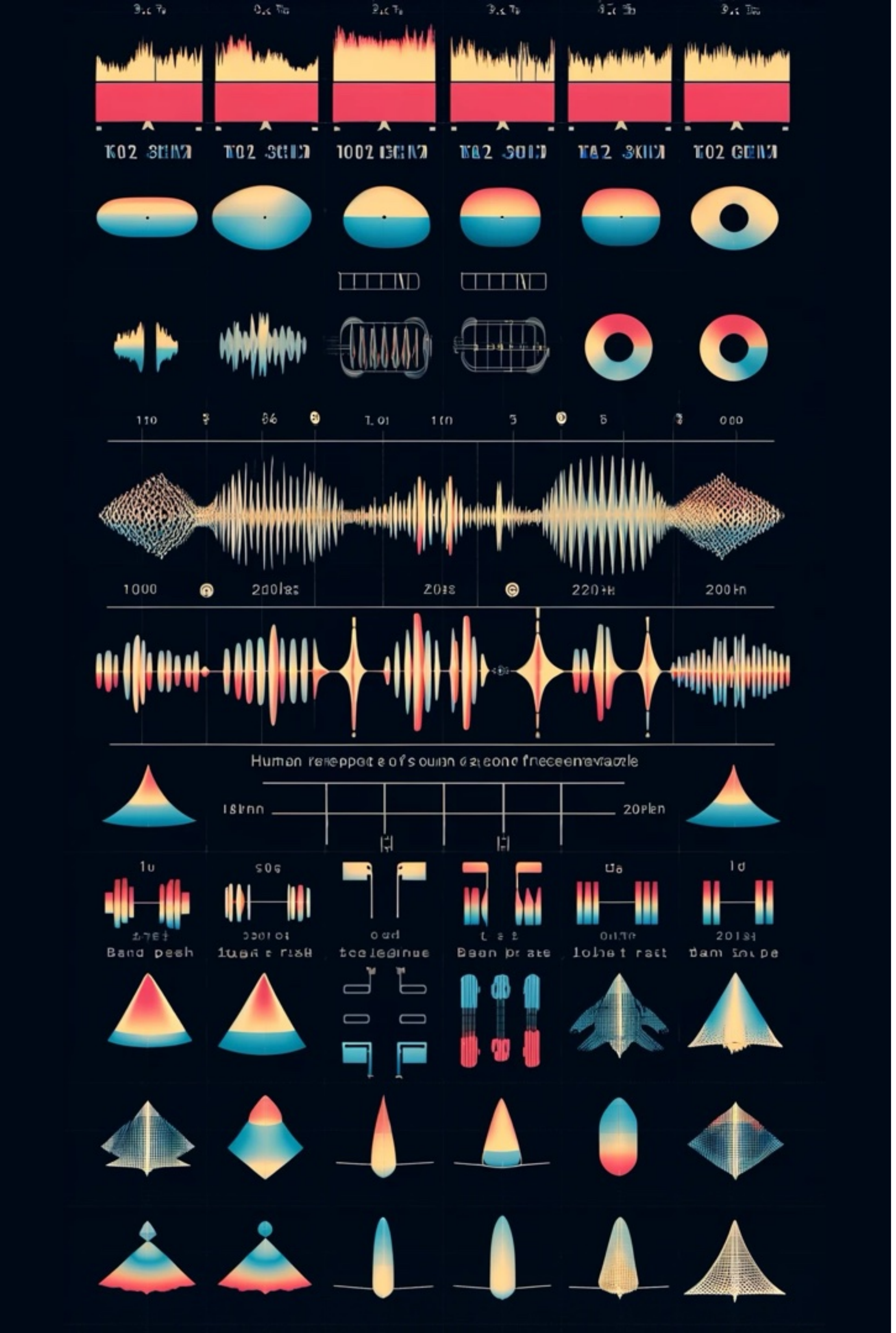
Overview

THE NON-LINEAR SCALE OF THE HUMAN EAR

The human ear does not perceive frequencies on a linear scale. Instead, it perceives them on a logarithmic scale. This means that a change in frequency at lower frequencies is more noticeable than at higher frequencies.

CAPTURING THE NON-LINEAR PERCEPTION OF FREQUENCY

The Mel scale is a perceptual scale of pitches listeners judge as equal in distance. It captures this non-linear perception of frequency.



PERCEPTION OF SOUND

Overview

A COMPARISON OF AUDIO DISCRIMINATION

Humans can easily distinguish between 100Hz and 200Hz audio, but it will be tough to tell the difference between 2100Hz and 2000Hz audio.

ACHIEVING SMOOTH MAGNITUDE SPECTRUM

The magnitude frequency response is multiplied by a set of triangular band-pass filters called Mel filter banks to attain a smooth magnitude spectrum.

LOG-MEL SPECTROGRAM

Spectrogram vs. Log-Mel Spectrogram

LOG-MEL IS DESIGNED TO MIMIC HUMAN PERCEPTION, WHICH IS MORE SENSITIVE TO DIFFERENCES IN LOWER FREQUENCIES THAN HIGHER ONES.

Spectrogram

Spectrogram uses a linear/ logarithmic frequency scale.

Linear

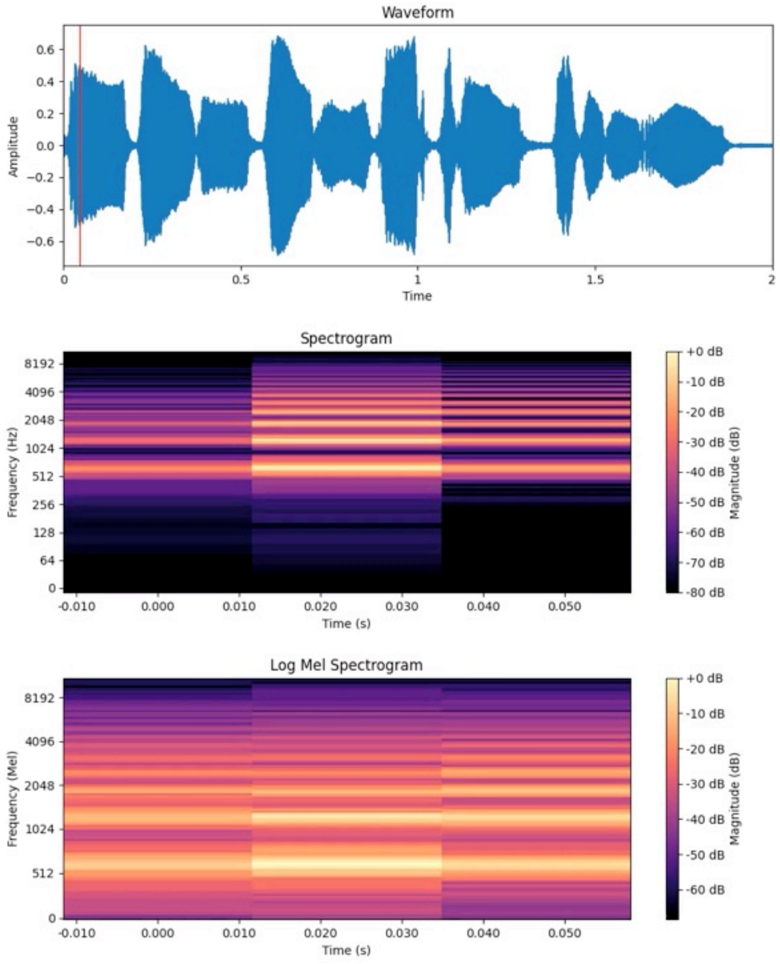
Direct frequency.

Logarithmic

Lower Frequency better seen.

Log-Mel Spectrogram

Uses the Mel scale for the frequency axis.



MEL-FREQUENCY CEPSTRAL COEFFICIENT

Log-Mel Spectrogram to MFCC

The speech signal's time power spectrum envelope represents the vocal tract, and MFCC (which is nothing but the coefficients that make up the *Mel-Frequency Cepstrum*) accurately represents this envelope.

Usage

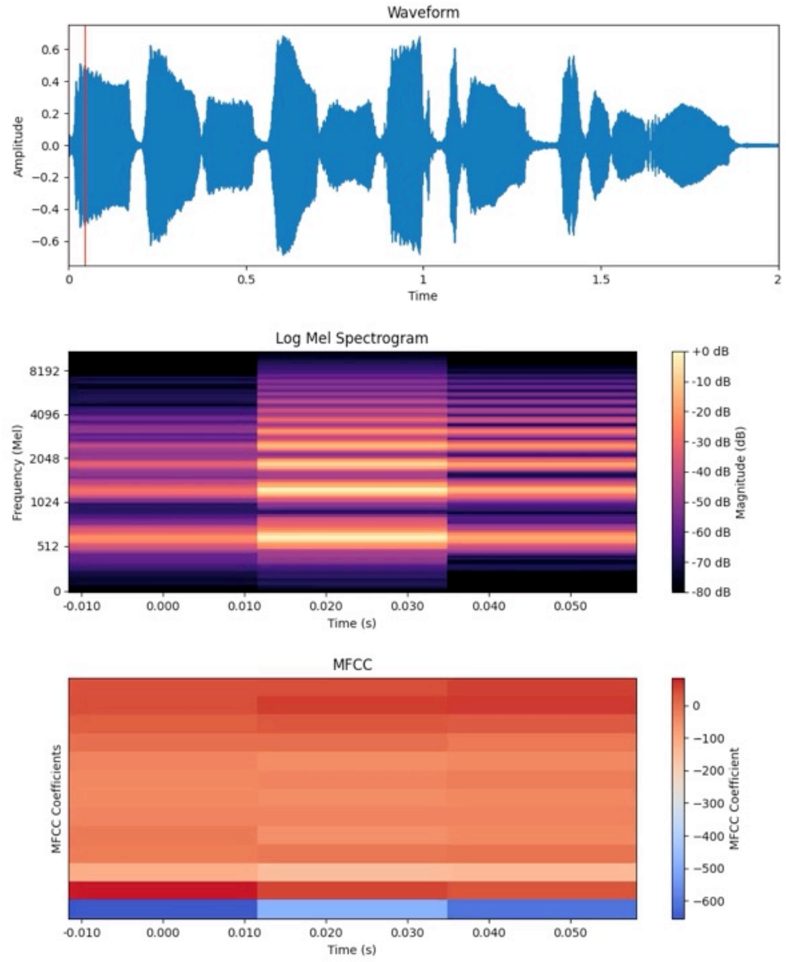
They are widely used in Speech Recognition.

Transformation

Discrete Cosine Transform is done on Log-Mel to create Mel-Scale Coefficients.

Influence

Sound generated by humans is determined by the shape of their vocal tract.



LIBROSA PACKAGE

A Python package for music and audio analysis

01 All the math is simplified using Librosa

02 Similar to OpenCV for Computer Vision

03 The method librosa.load() loads audio like cv2.imread()

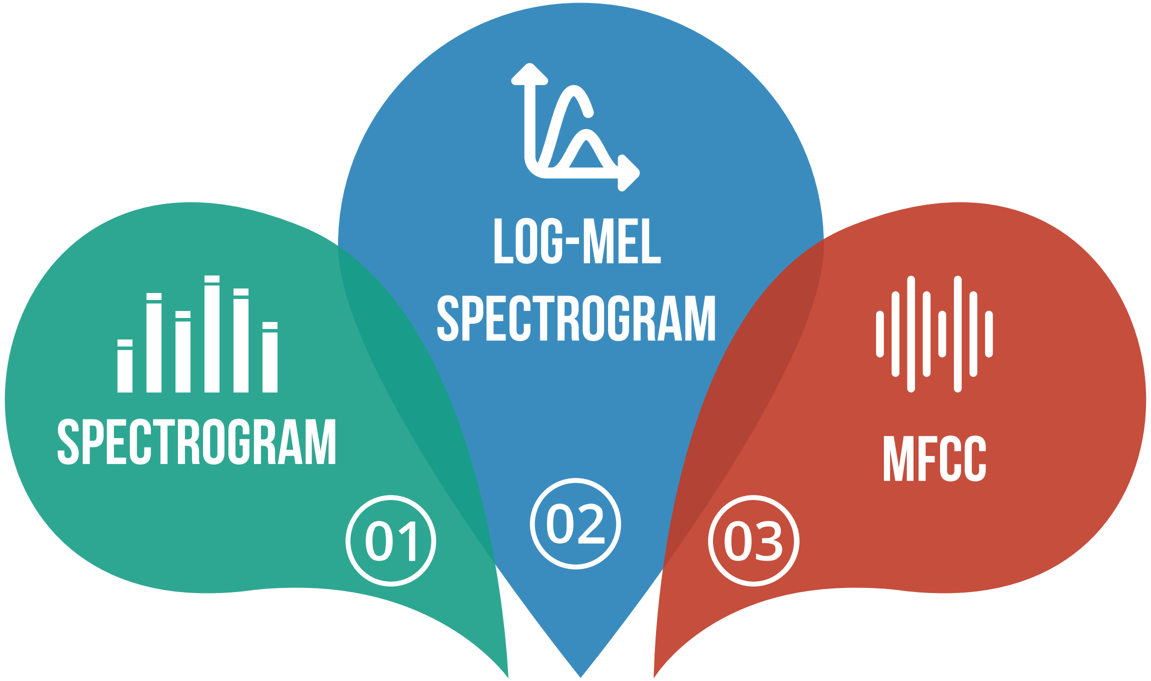
04 To display waveforms, use *librosa.display.waveshow*

05 All that math is a simple line of *librosa.feature.mfcc()*

06 Generate spectrograms using *librosa.stft()*

LIBROSA PACKAGE

Key Differences



Option 01

Spectrogram

Represents the magnitude of frequencies over time.

Option 02

Log-Mel Spectrogram

Represents frequencies on the Mel scale, providing a more perceptually relevant frequency axis and using logarithmic magnitude scaling.

Option 03

MFCC

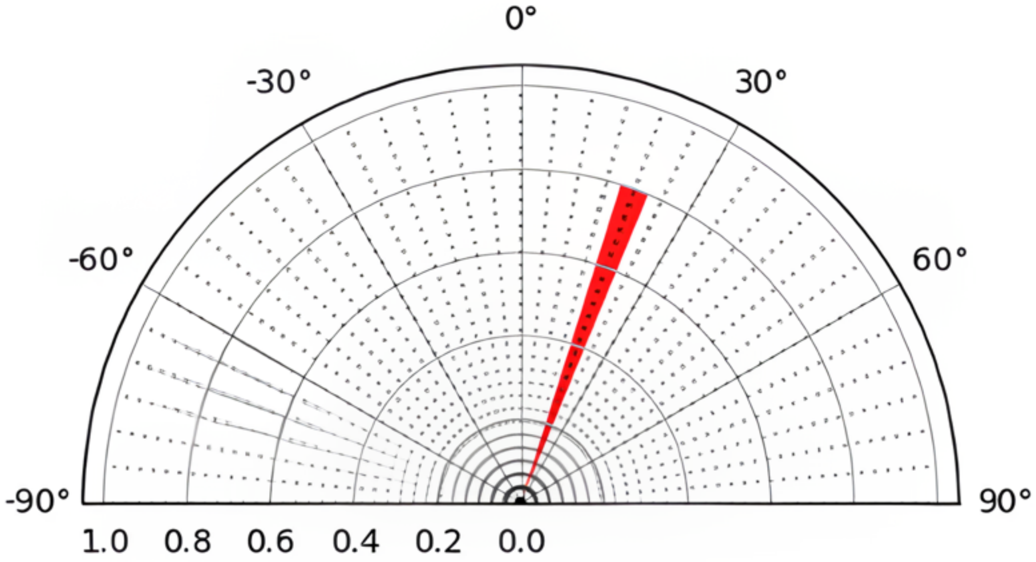
Represents the signal in a compact form, capturing the most important aspects of the power spectrum while reducing dimensionality.

DOA ESTIMATION PLOT

Adavanne et al. (2021), Differentiable Tracking-Based Training of DL Sound Source Localizers

Inputs

Real and imaginary components of spectrograms from 8 microphone channels.



Outputs

Probabilities for 37 angles ranging from -90° to 90° in 5 degrees resolution.

SOUND LOCALIZATION

Owens et al. (2021), How to Listen: Rethinking Visualizing and Localizing Sound

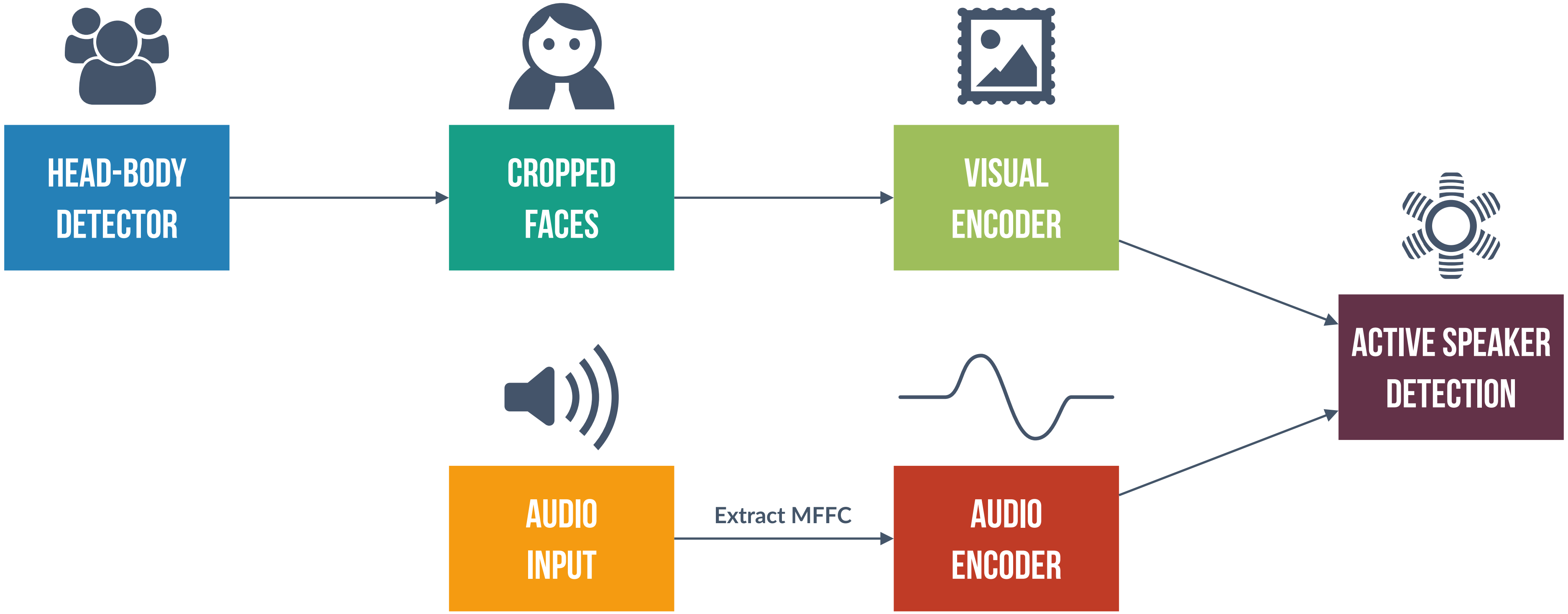
- Image Encoder CLIP
- Audio Encoder Wav2CLIP
- Contrastive Learning
- Uses Raw Audio Waveform

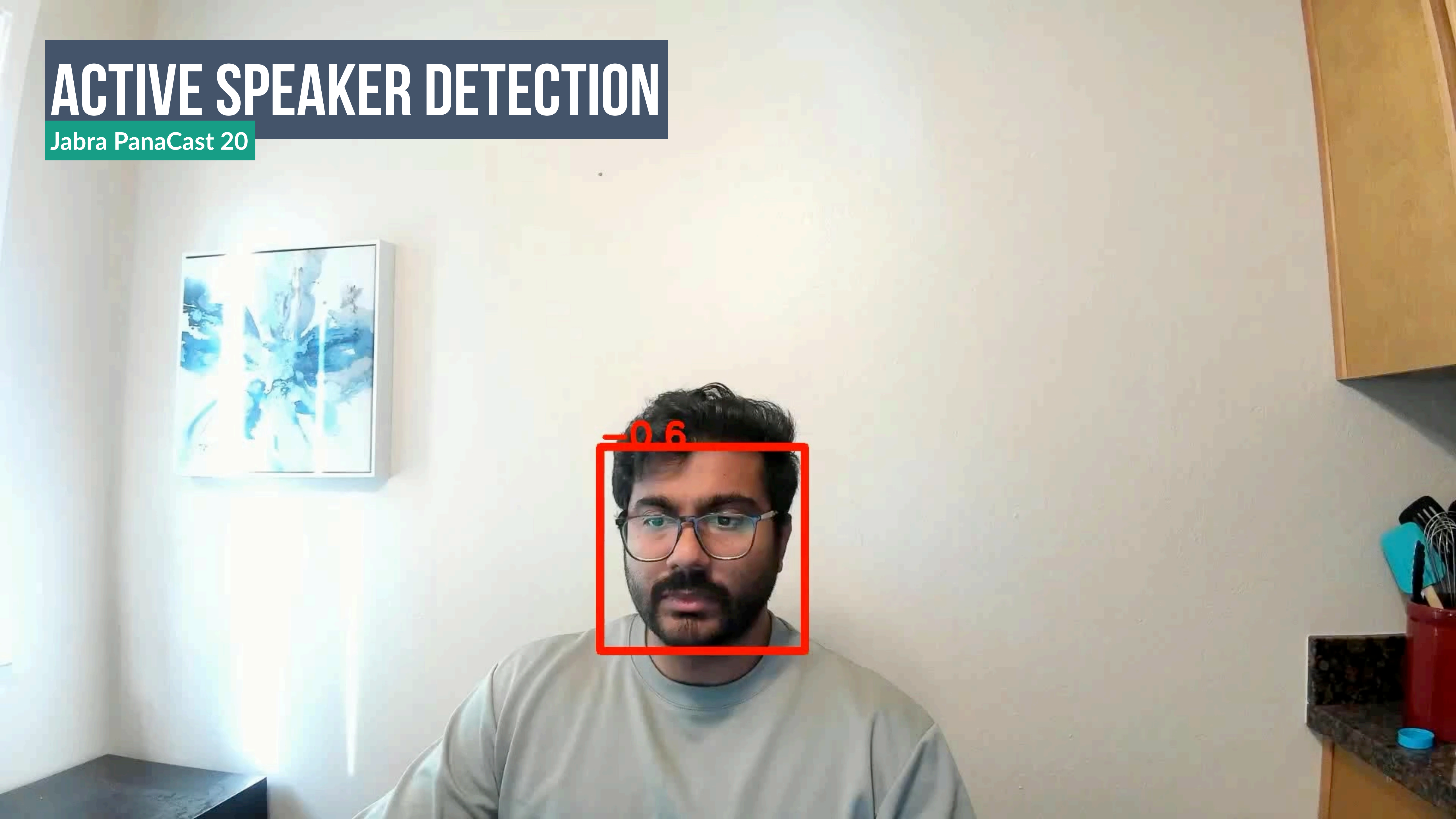
SOUND LOCALIZATION

Owens et al. (2021). How to Listen: Rethinking Visualizing and Localizing Sound.

ACTIVE SPEAKER DETECTION

Ruijie et al. (2021), Is Someone Speaking? Exploring Long-Term Temporal Features for Audio-Visual Active Speaker Detection




ACTIVE SPEAKER DETECTION
Jabra PanaCast 20

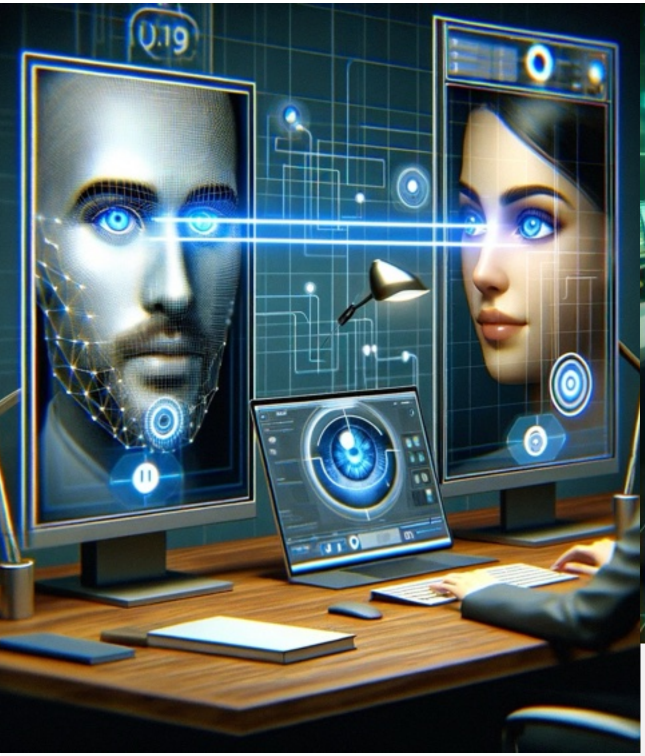
ACTIVE SPEAKER DETECTION

Jabra PanaCast 50

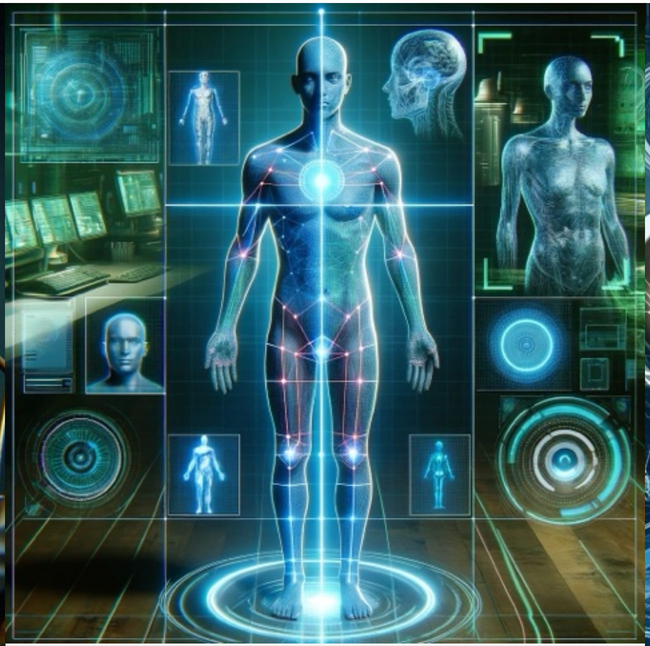
DEMOS

JABRA COLLABORATION BUSINESS

Try Our Multimodal Demos



GAZE CORRECTION
(JABRA EYE CONTACT)



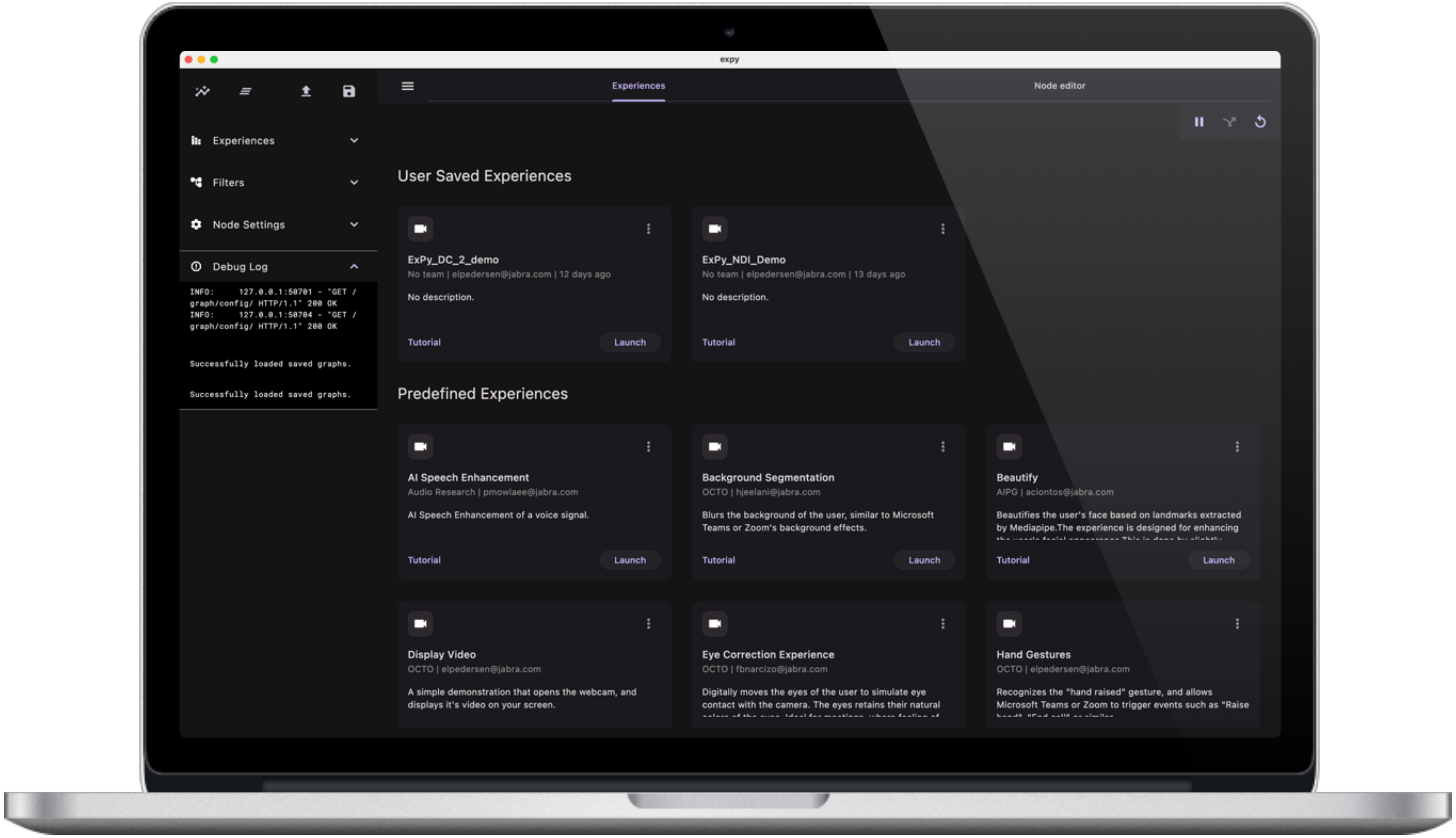
BODY SEGMENTATION
(JABRA PANACAST 20)

HAND GESTURES RECOGNITION
(EXPY)

SOUND LOCALIZATION
(EXPY)

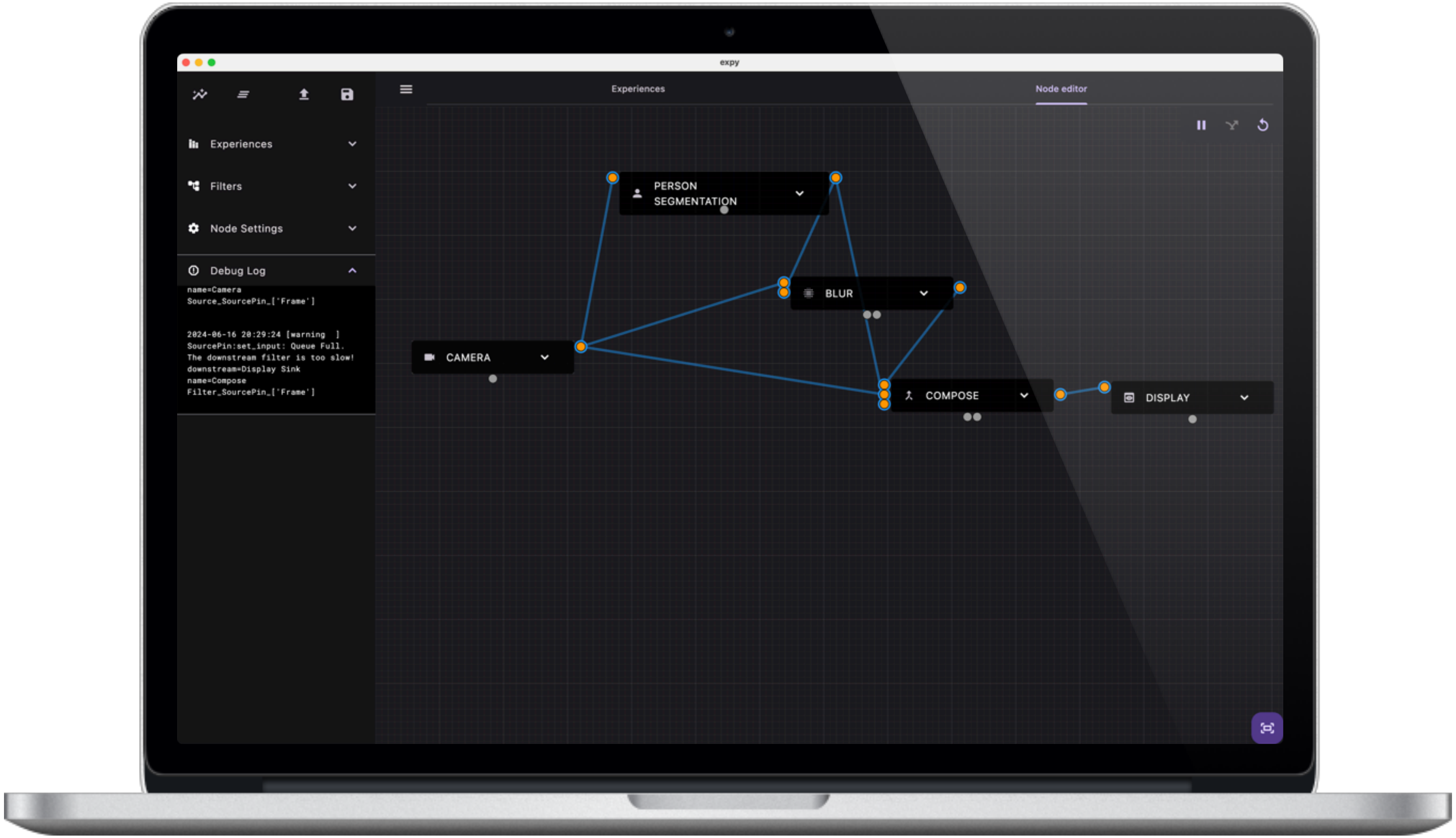
JABRA COLLABORATION BUSINESS

Expy Experience Platform



JABRA COLLABORATION BUSINESS

Expy Experience Platform



QUESTIONS & ANSWERS

THANK YOU!

CLOSING REMARKS AND JOINT Q&A