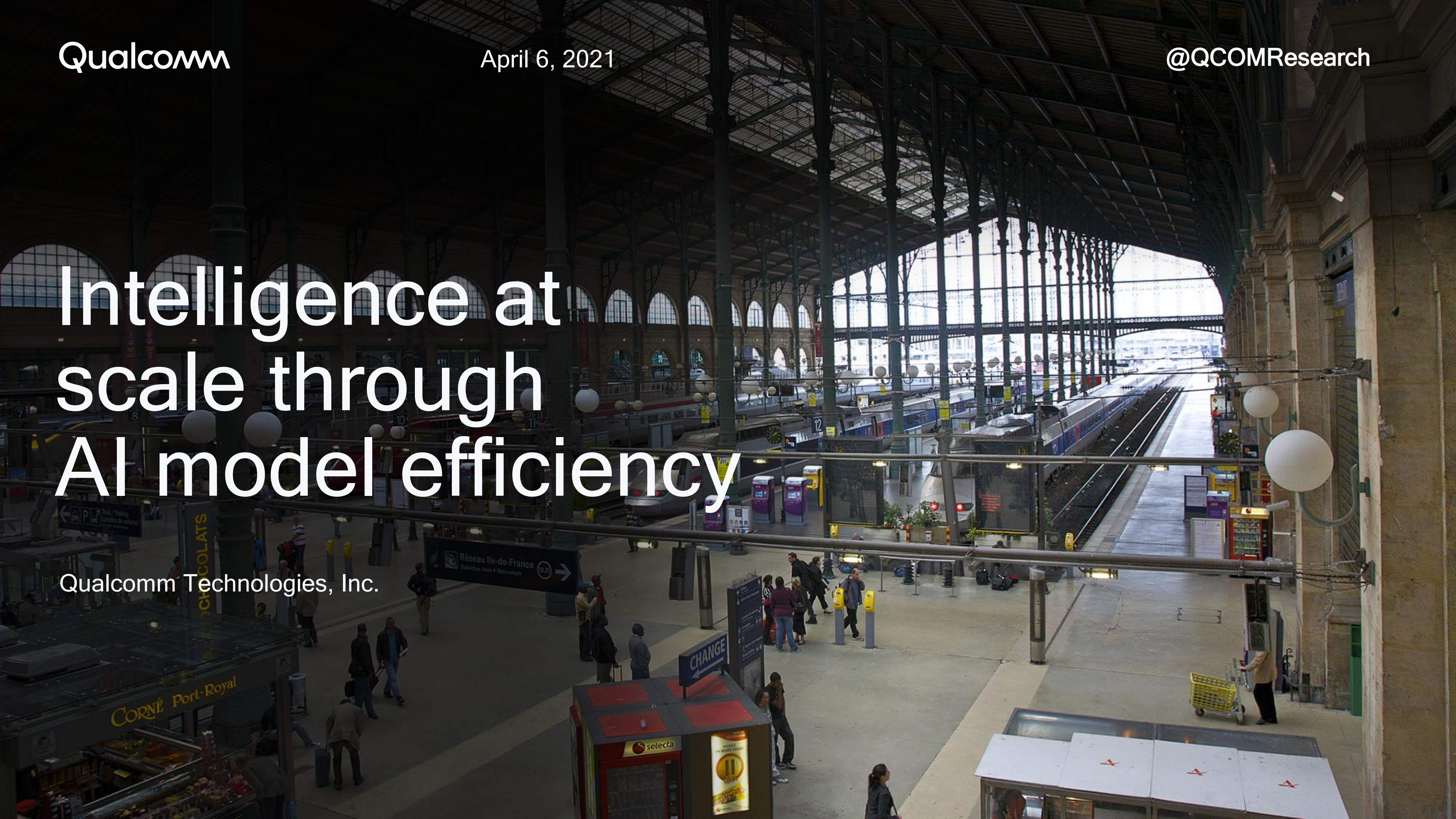


Qualcomm

April 6, 2021

@QCOMResearch

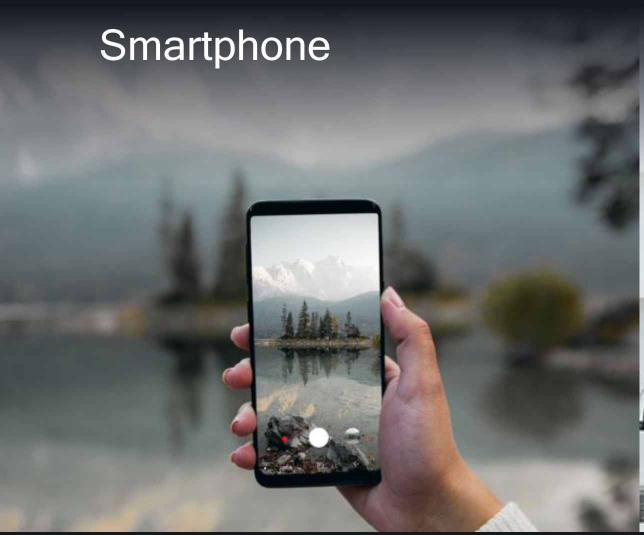
Intelligence at scale through AI model efficiency

Qualcomm Technologies, Inc.

Agenda

- Why efficient machine learning is necessary for AI to proliferate
- Our latest research to make AI models more efficient
- Our open-source projects to scale efficient AI

Smartphone



Smart homes

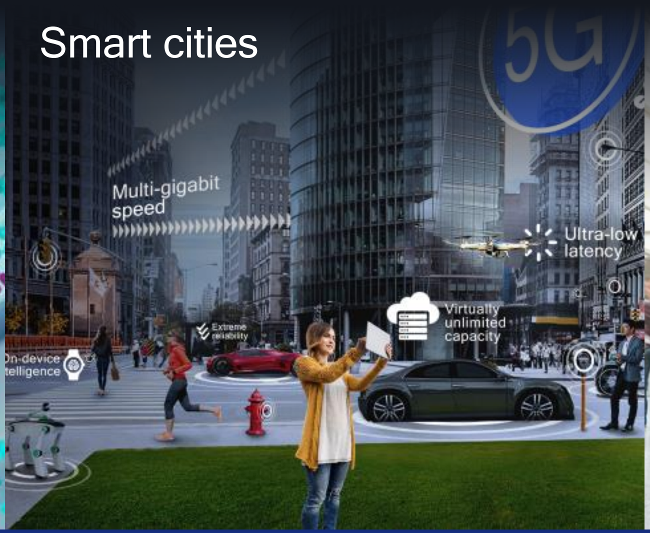
Video conferencing

Autonomous vehicles

Smart factories

Extended reality

Smart cities



Video monitoring

AI is being used all around us

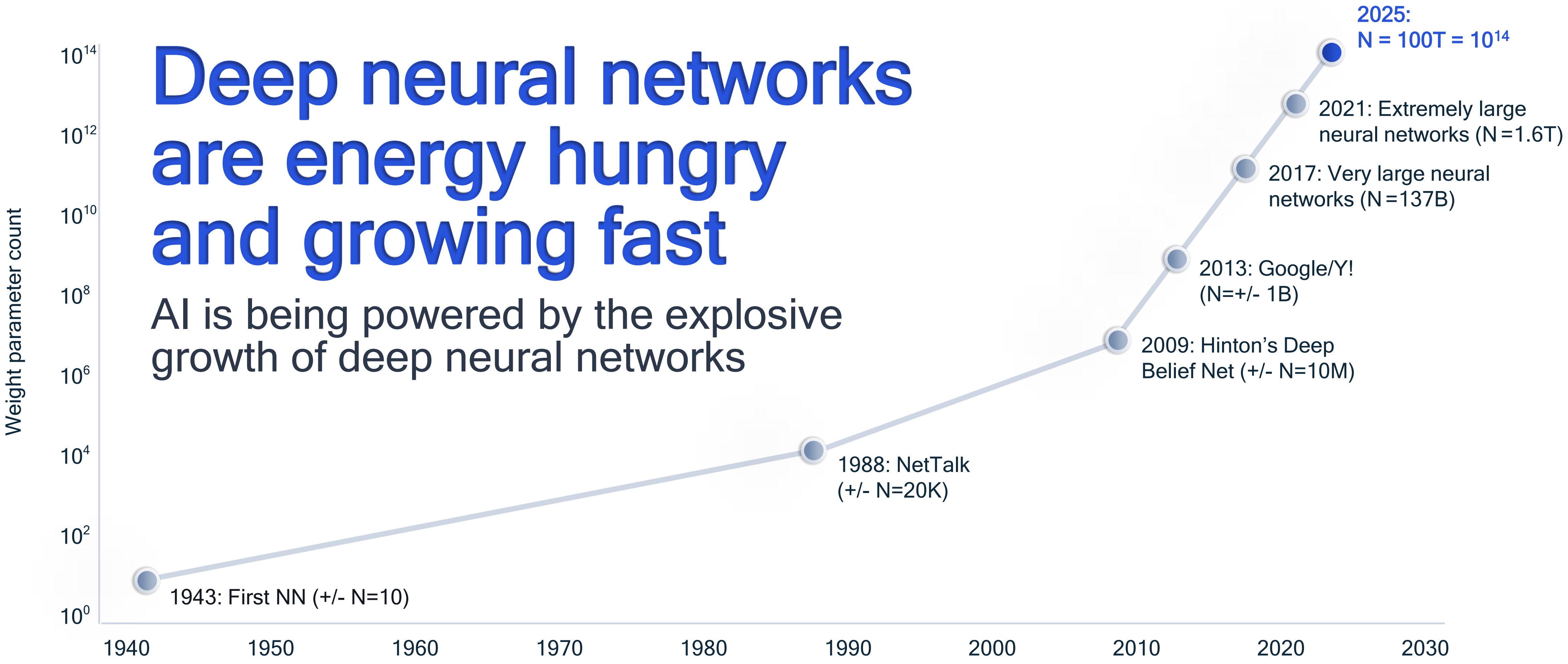
increasing productivity, enhancing collaboration, and transforming industries

AI video analysis is on the rise

Trend toward more cameras, higher resolution, and increased frame rate across devices

Deep neural networks are energy hungry and growing fast

AI is being powered by the explosive growth of deep neural networks



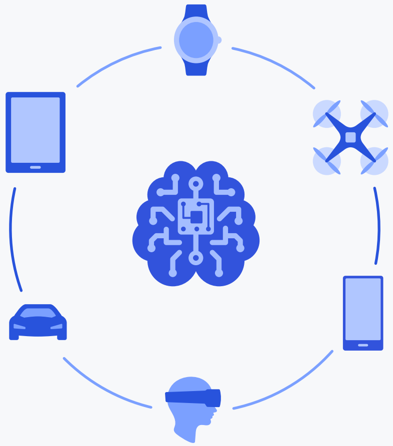
2025

Will we have reached the capacity of the human brain?
Energy efficiency of a brain is 100x better than current hardware

Source: Welling

The challenge of AI workloads

- Very compute intensive
- Large, complicated neural network models
- Complex concurrencies
- Real-time
- Always-on



Power and thermal efficiency are essential for on-device AI

Constrained mobile environment

- Must be thermally efficient for sleek, ultra-light designs
- Requires long battery life for all-day use
- Storage/memory bandwidth limitations

Holistic model efficiency research

Multiple axes to shrink AI models and efficiently run them on hardware

Quantization

Learning to reduce bit-precision while keeping desired accuracy

Compilation

Learning to compile AI models for efficient hardware execution

Compression

Learning to prune model while keeping desired accuracy

Neural architecture search

Learning to design smaller neural networks that are on par or outperform hand-designed architectures on real hardware

Leading research to efficiently quantize AI models

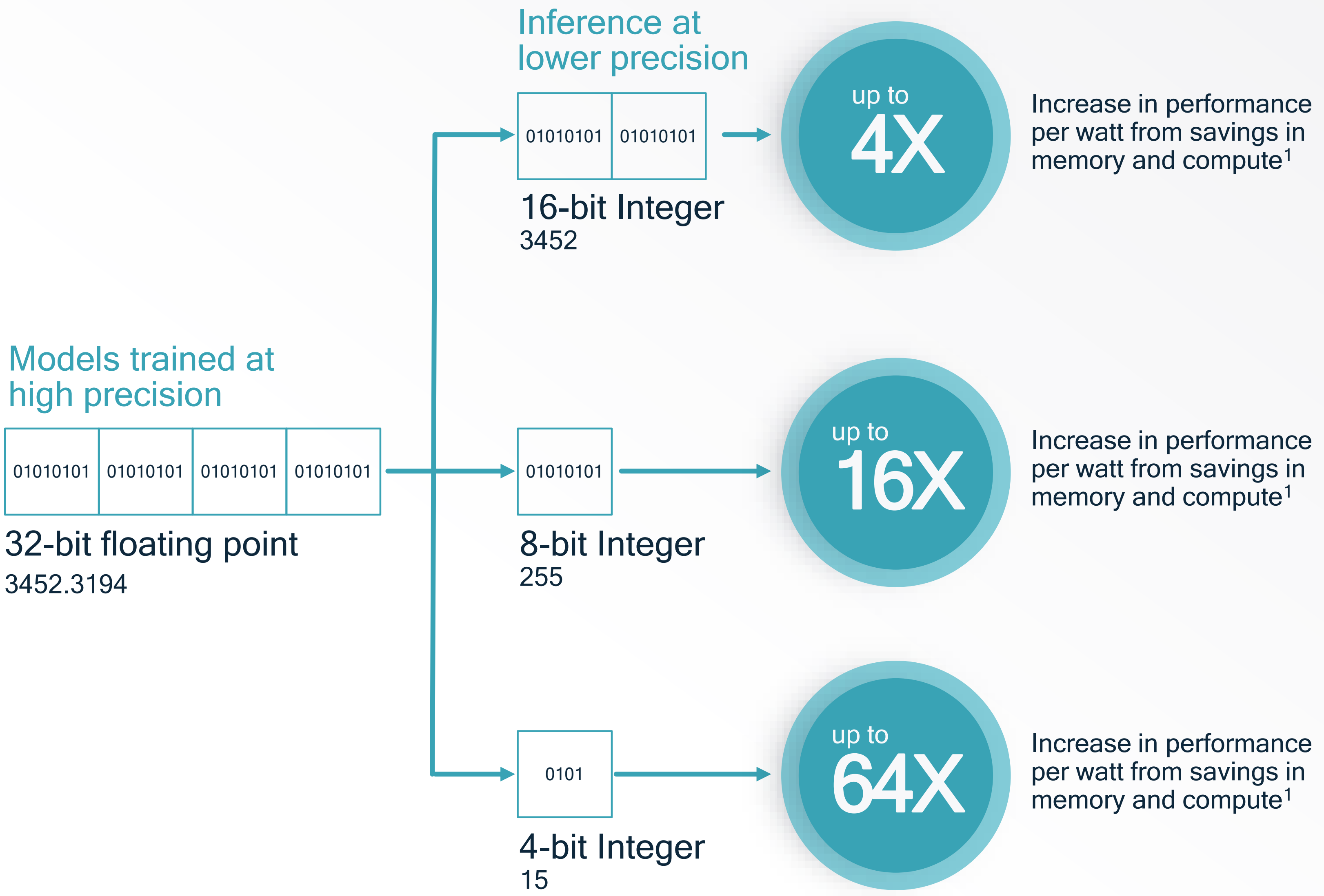
Automated reduction in precision of weights and activations while maintaining accuracy

Promising results show that low-precision integer inference can become widespread

Virtually the same accuracy between a FP32 and quantized AI model through:

- Automated, data free, post-training methods
- Automated training-based mixed-precision method

Significant performance per watt improvements through quantization



1: FP32 model compared to quantized model

Pushing the limits of what's possible with quantization

Data-free quantization

How can we make quantization as simple as possible?

Created an automated method that addresses bias and imbalance in weight ranges:

- ☑ No training
- ☑ Data free

SOTA 8-bit results

Making 8-bit weight quantization ubiquitous

<1% Accuracy drop for MobileNet V2 against FP32 model

Data-Free Quantization Through Weight Equalization and Bias Correction (Nagel, van Baalen, et al., ICCV 2019)

AdaRound

Is rounding to the nearest value the best approach for quantization?

Created an automated method for finding the best rounding choice:

- ☑ No training
- ☑ Minimal unlabeled data

SOTA 4-bit weight results

Making 4-bit weight quantization ubiquitous

<2.5% Accuracy drop for MobileNet V2 against FP32 model

Up or Down? Adaptive Rounding for Post-Training Quantization (Nagel, Amjad, et al., ICML 2020)

Bayesian bits

Can we quantize layers to different bit widths based on precision sensitivity?

Created a novel method to learn mixed-precision quantization:

- ☑ Training required
- ☑ Training data required
- ☑ Jointly learns bit-width precision and pruning

SOTA mixed-precision results

Automating mixed-precision quantization and enabling the tradeoff between accuracy and kernel bit-width

<1% Accuracy drop for MobileNet V2 against FP32 model for mixed precision model with **computational complexity equivalent to a 4-bit weight model**

Bayesian Bits: Unifying Quantization and Pruning van Baalen, Louizos, et al., NeurIPS 2020)

SOTA: State-of-the-art

Neural network complexity

Many state-of-the-art neural network solutions are large, complex, and do not run efficiently on target hardware

Neural network diversity

For different tasks and use case cases, many different neural networks are required

Device diversity

Deploying neural networks to many different devices with different configurations and changing software is required

Cost

Compute and engineering resources for training plus evaluation are too costly and time consuming

Optimizing and deploying state-of-the-art AI models for diverse scenarios at scale is challenging

NAS

Neural Architecture Search

An automated way to learn a network topology that can achieve the best performance on a certain task

Search space

Set of operations and how they can be connected to form valid network architectures

Search algorithm

Method for sampling a population of good network architecture candidates

Evaluation strategy

Method to estimate the performance of sampled network architectures

Existing NAS solutions do not address all the challenges

Lack diverse search

Hard to search in diverse spaces, with different block-types, attention, and activations

Repeated training phase for every new scenario

High cost

Brute force search is expensive

>40,000 epochs per platform

Do not scale

Repeated training phase for every new device

>40,000 epochs per platform

Unreliable hardware models

Requires differentiable cost-functions

Repeated training phase for every new device

Introducing new AI research

DONNA

Distilling Optimal Neural Network Architectures

Efficient NAS with hardware-aware optimization

A scalable method that finds pareto-optimal network architectures in terms of accuracy and latency for any hardware platform at low cost

Starts from an oversized pretrained reference architecture

Oversized pretrained reference architecture

↓

DONNA

↓

Set of pareto-optimal network architectures

Diverse search to find the best models

Supports diverse spaces with different cell-types, attention, and activation functions (ReLU, Swish, etc.)

Low cost

Low start-up cost of 1000-4000 epochs, equivalent to training 2-10 networks from scratch

Scalable

Scales to many hardware devices at minimal cost

Reliable hardware measurements

Uses direct hardware measurements instead of a potentially inaccurate hardware model

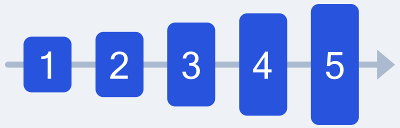
Distilling Optimal Neural Networks: Rapid Search in Diverse Spaces (Moons, Bert, et al., arXiv 2020)

DONNA 4-step process

Objective: Build accuracy model of search space once, then deploy to many scenarios

A Define reference and search space once

Define backbone:
- Fixed channels
- Head and Stem



Varying parameters:
- Kernel Size
- Expansion Factors
- Network depth
- Network width
- Attention/activation
- Different efficient layer types

Define reference architecture and search-space once

A diverse search space is essential for finding optimal architectures with higher accuracy

Select reference architecture

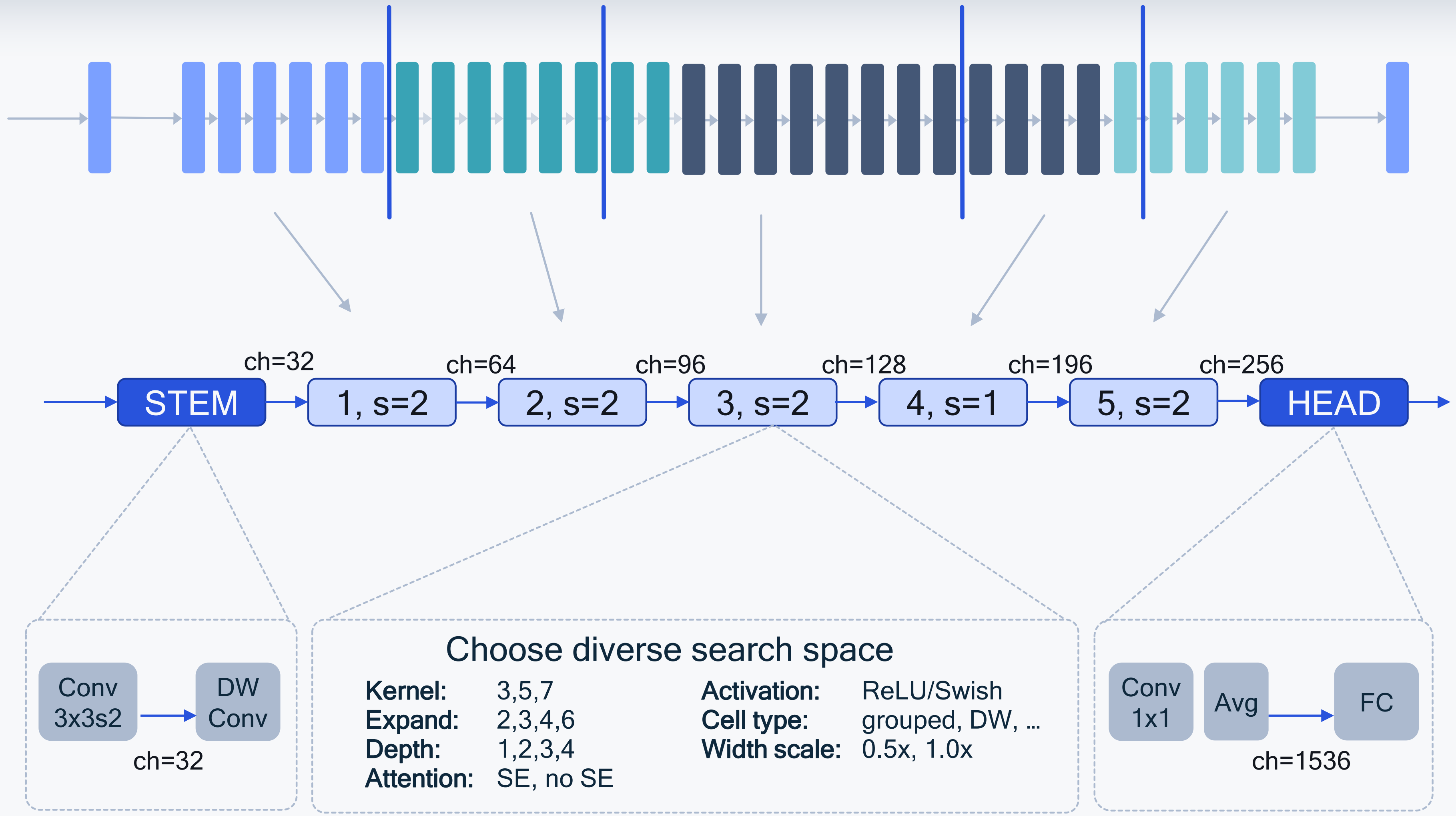
The largest model in the search-space

Chop the NN into blocks

Fix the STEM, HEAD, # blocks, strides, # channels at block-edge

Choose search space

Diverse factorized hierarchical search space, including variable kernel-size, expansion-rate, depth, # channels, cell-type, activation, attention



Ch: channel; SE: Squeeze-and-Excitation

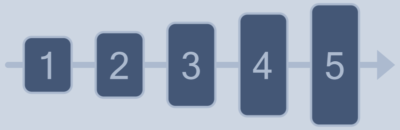
DONNA 4-step process

Objective: Build accuracy model of search space once, then deploy to many scenarios

A Define reference and search space once

Define backbone:

- Fixed channels
- Head and Stem

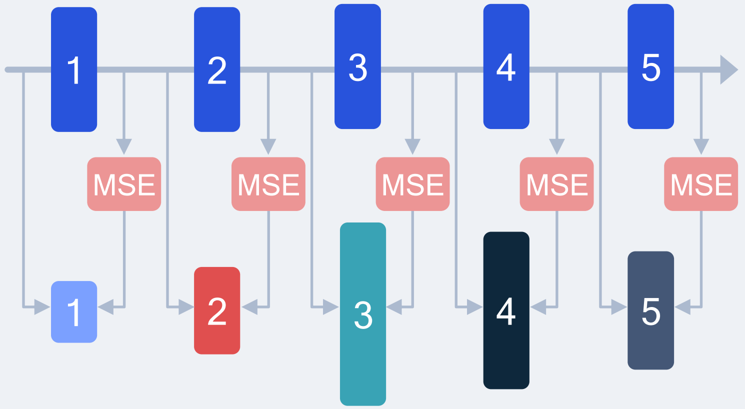


Varying parameters:

- Kernel Size
- Expansion Factors
- Network depth
- Network width
- Attention/activation
- Different efficient layer types

B Build accuracy model via Knowledge Distillation (KD) once

Approximate ideal projections of a reference model through KD



Use quality of blockwise approximations to build accuracy model

Build accuracy predictor via BKD once

Low-cost hardware-agnostic training phase

Block library

Pretrain all blocks in search-space through blockwise knowledge distillation

Block pretrained weights

Block quality metrics

- Fast block training
- Trivial parallelized training
- Broad search space

Architecture library

Quickly finetune a representative set of architectures

Finetuned architectures

- Finetune sampled networks
- Fast network training
- Only 20-30 NN required

Accuracy predictor

Fit linear regression model

- Regularized Ridge Regression
- Accurate predictions

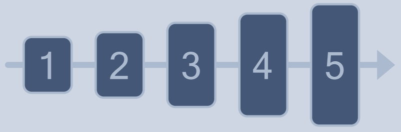
BKD: blockwise knowledge distillation

DONNA 4-step process

Objective: Build accuracy model of search space once, then deploy to many scenarios

A Define reference and search space once

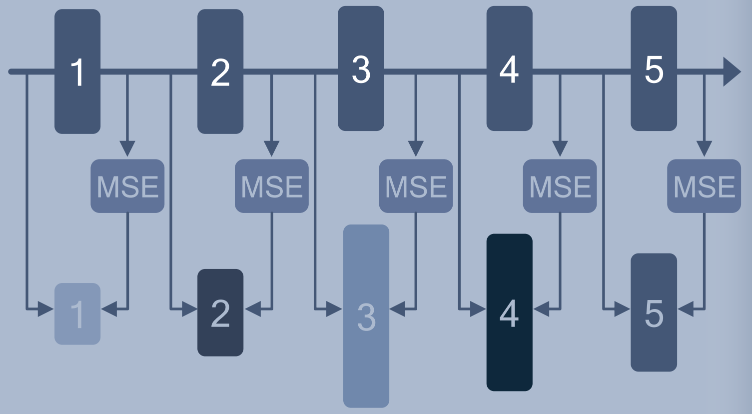
Define backbone:
- Fixed channels
- Head and Stem



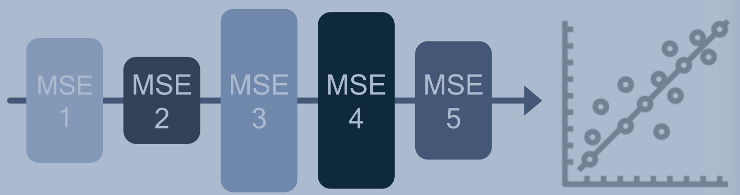
Varying parameters:
- Kernel Size
- Expansion Factors
- Network depth
- Network width
- Attention/activation
- Different efficient layer types

B Build accuracy model via Knowledge Distillation (KD) once

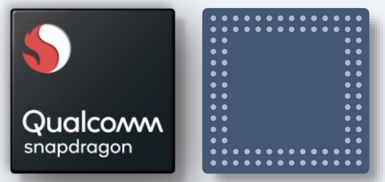
Approximate ideal projections of a reference model through KD



Use quality of blockwise approximations to build accuracy model



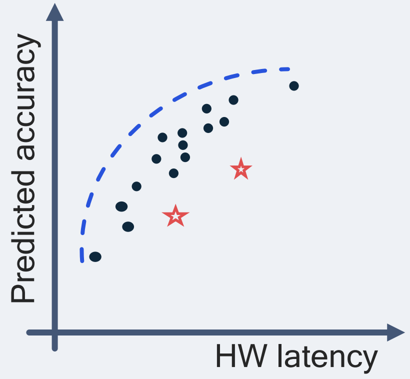
C Evolutionary search in 24h



\+

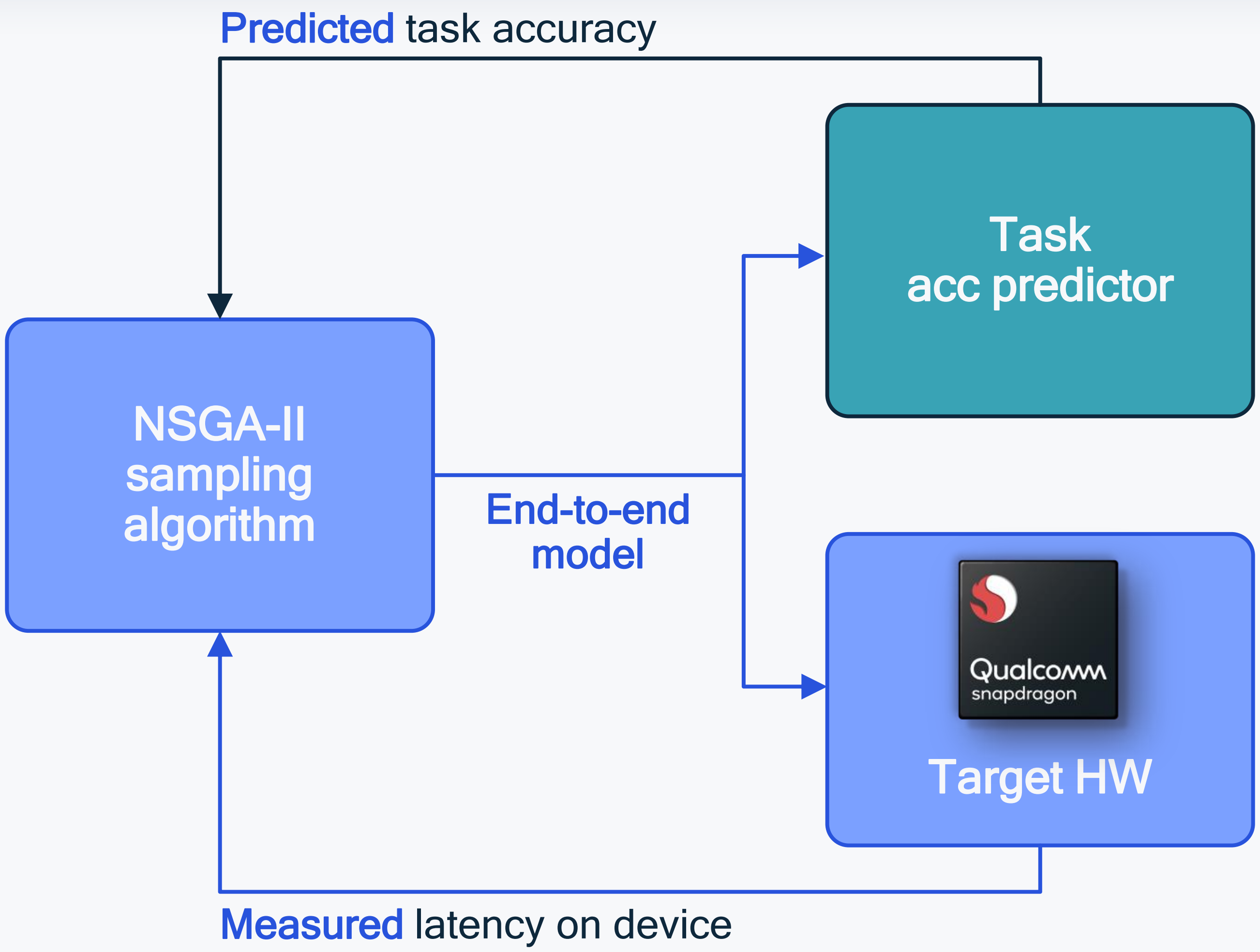
different compiler versions, different image sizes

Scenario-specific search



Evolutionary search with real hardware measurements

Scenario-specific search allows users to select optimal architectures for real-life deployments



Quick turnaround time

Results in +/- 1 day using one measurement device

Accurate scenario-specific search

Captures all intricacies of the hardware platform and software – e.g. run-time version or devices

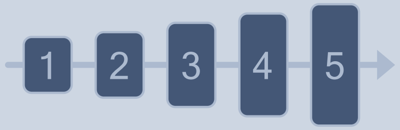
NSGA: Non-dominated Sorting Genetic Algorithm

DONNA 4-step process

Objective: Build accuracy model of search space once, then deploy to many scenarios

A Define reference and search space once

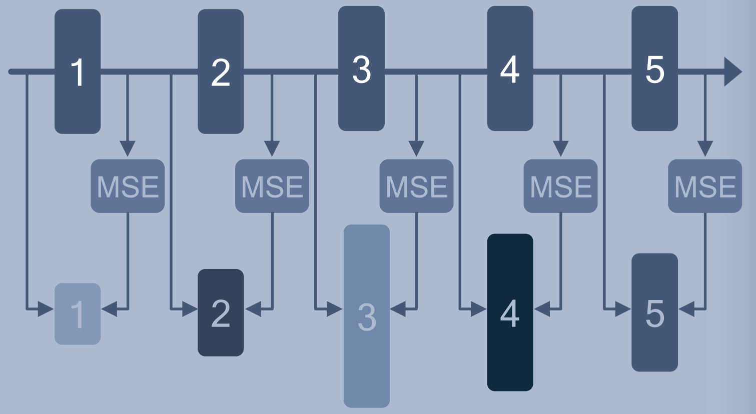
Define backbone:
- Fixed channels
- Head and Stem



Varying parameters:
- Kernel Size
- Expansion Factors
- Network depth
- Network width
- Attention/activation
- Different efficient layer types

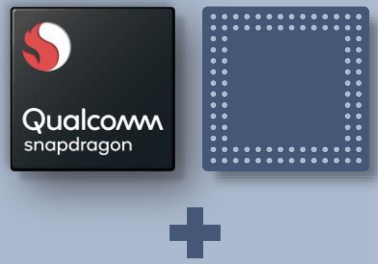
B Build accuracy model via Knowledge Distillation (KD) once

Approximate ideal projections of a reference model through KD



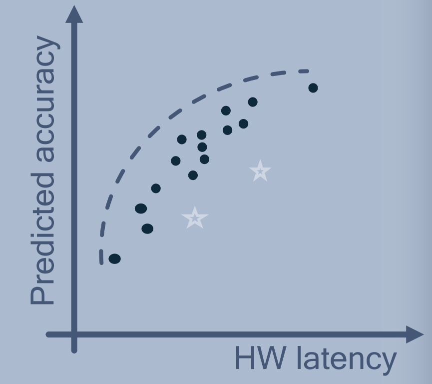
Use quality of blockwise approximations to build accuracy model

C Evolutionary search in 24h

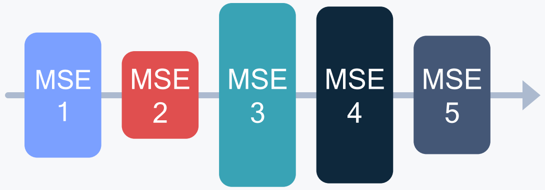


different compiler versions, different image sizes

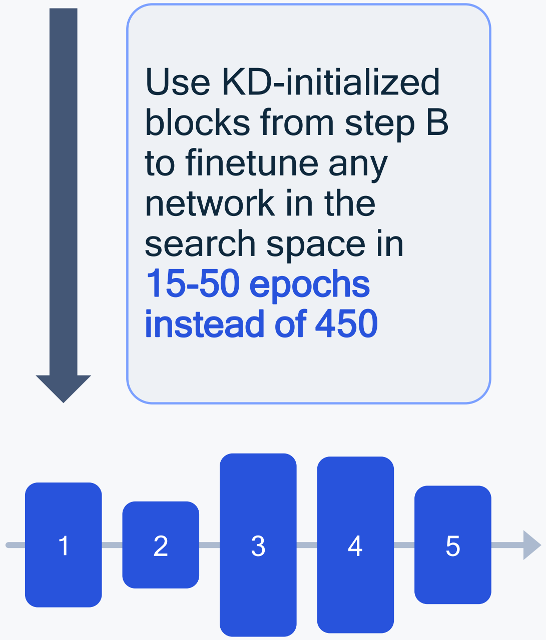
Scenario-specific search



D Sample and finetune



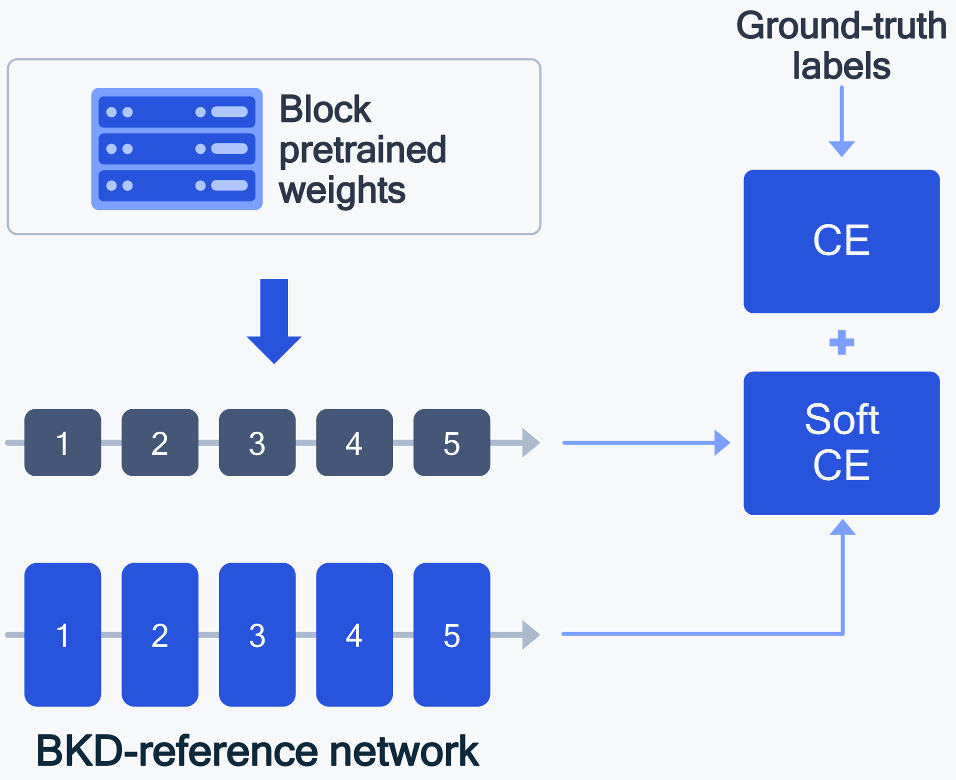
Use KD-initialized blocks from step B to finetune any network in the search space in **15-50 epochs instead of 450**

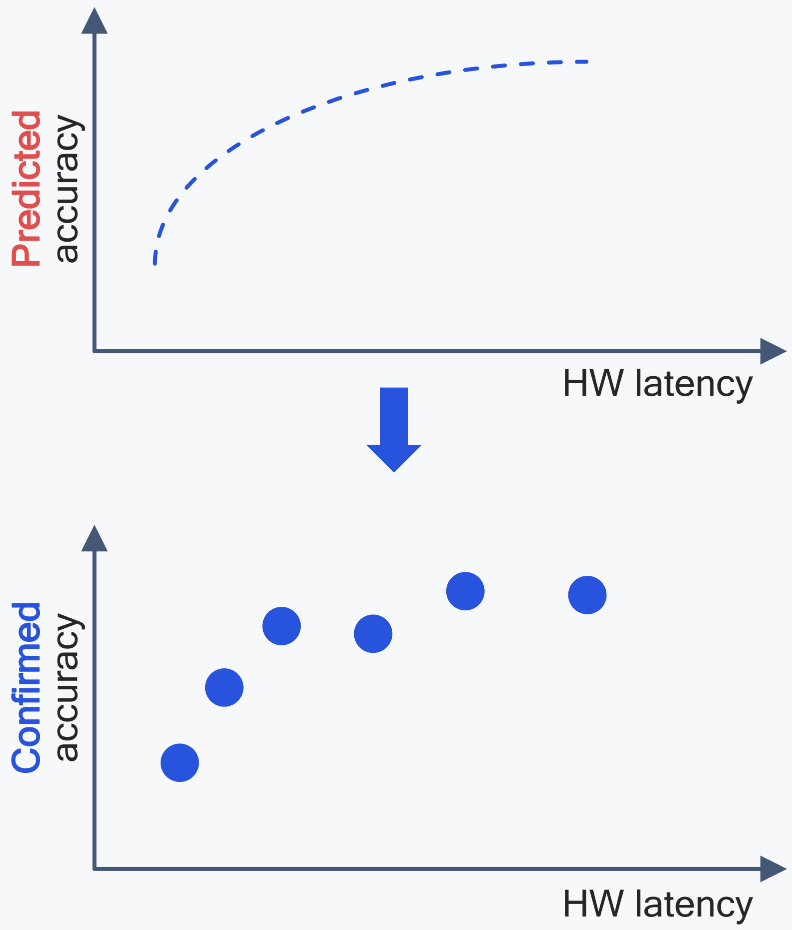


Quickly finetune predicted Pareto-optimal architectures

Finetune to reach full accuracy and complete hardware-aware optimization for on-device AI deployments

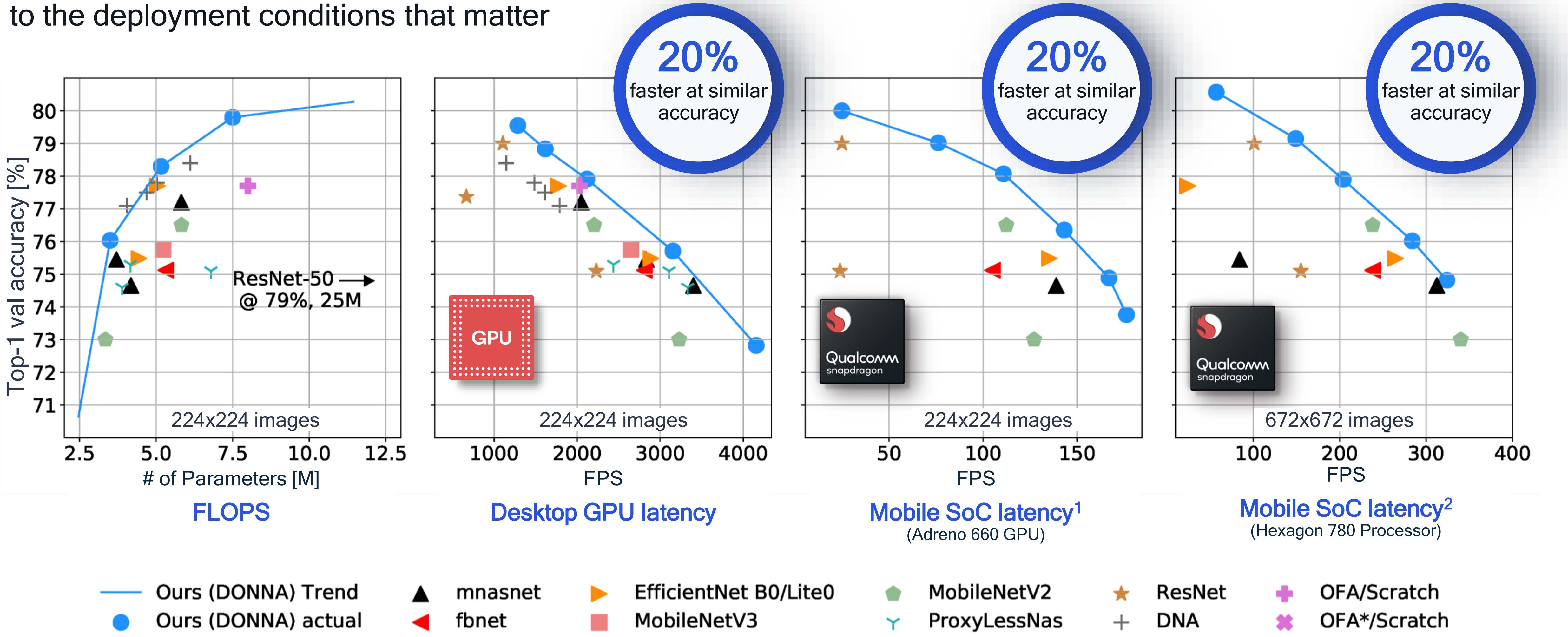
Soft distillation on teacher logits





DONNA finds state-of-the-art networks for on-device scenarios

Quickly optimize and make tradeoffs in model accuracy with respect to the deployment conditions that matter



FLOPS

Desktop GPU latency

Mobile SoC latency[1]
(Adreno 660 GPU)

Mobile SoC latency[2]
(Hexagon 780 Processor)

Qualcomm® Adreno™ 660 GPU in the Snapdragon 888 running on the Samsung Galaxy S21. 2: Qualcomm® Hexagon™ 780 Processor in the Snapdragon 888 running on the Samsung Galaxy S21.

DONNA efficiently finds optimal models over diverse scenarios

Cost of training is a handful of architectures*

Method	Granularity	Macro-diversity	Search-cost 1 scenario [epochs]	Cost / scenario 4 scenarios [epochs]	Cost / scenario ∞ scenarios [epochs]
OFA	Layer-level	Fixed	1200+10×[25 – 75]	550 – 1050	250 – 750
DNA	Layer-level	Fixed	770+10×450	4700	4500
MNasNet	Block-level	Variable	40000+10×450	44500	44500
DONNA	Block-level	Variable	4000+10×50	1500	500

Good OK Not good

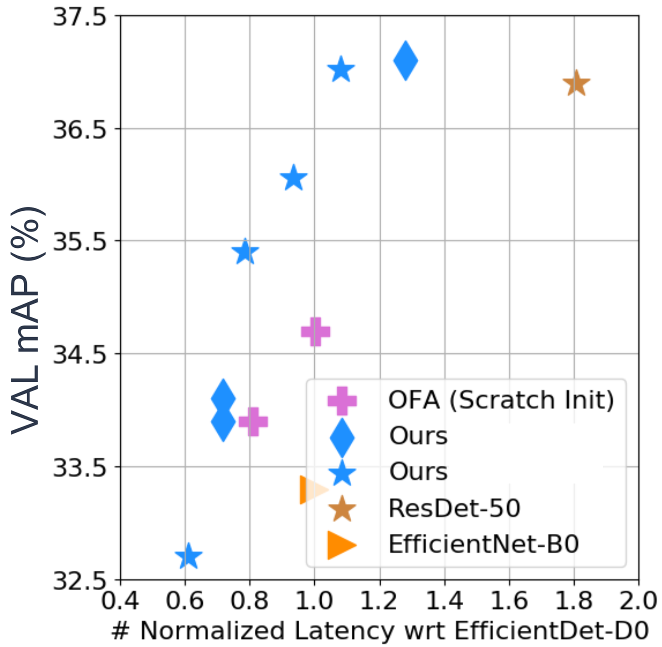
DONNA provides MnasNet-level diversity at 100x lower cost

*Training 1 model from scratch = 450 epochs

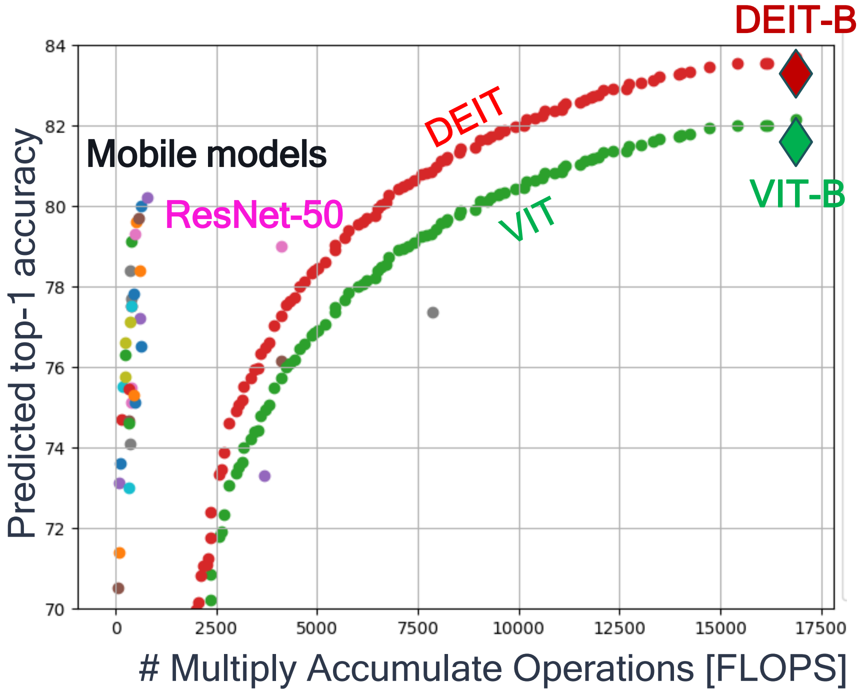
DONNA finds state-of-the-art networks for on-device scenarios

Quickly optimize and make tradeoffs in model accuracy with respect to the deployment conditions that matter

Object detection



Vision transformers

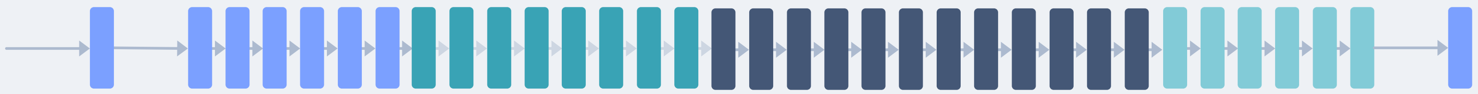


DONNA applies directly to downstream tasks and non-CNN neural architectures without conceptual code changes

User perspective for DONNA

Build accuracy model of search space once, then deploy to many scenarios

IN — Oversized pretrained reference architecture



DONNA

A — Define a **search-space** of smaller, faster network architectures

B — Create an **accuracy predictor** for all network architectures

Roughly equal to training 2-10 nets from scratch

C — Execute predictor-driven **evolutionary search** on-device

1 **day** / use-case

D — **Finetune best searched models**, 9-30x faster vs regular training

50 GPU-hrs / **net**

OUT — Set of Pareto-optimal network architectures

Deploy best models @ 3ms, 6ms, 9ms, … on different chips

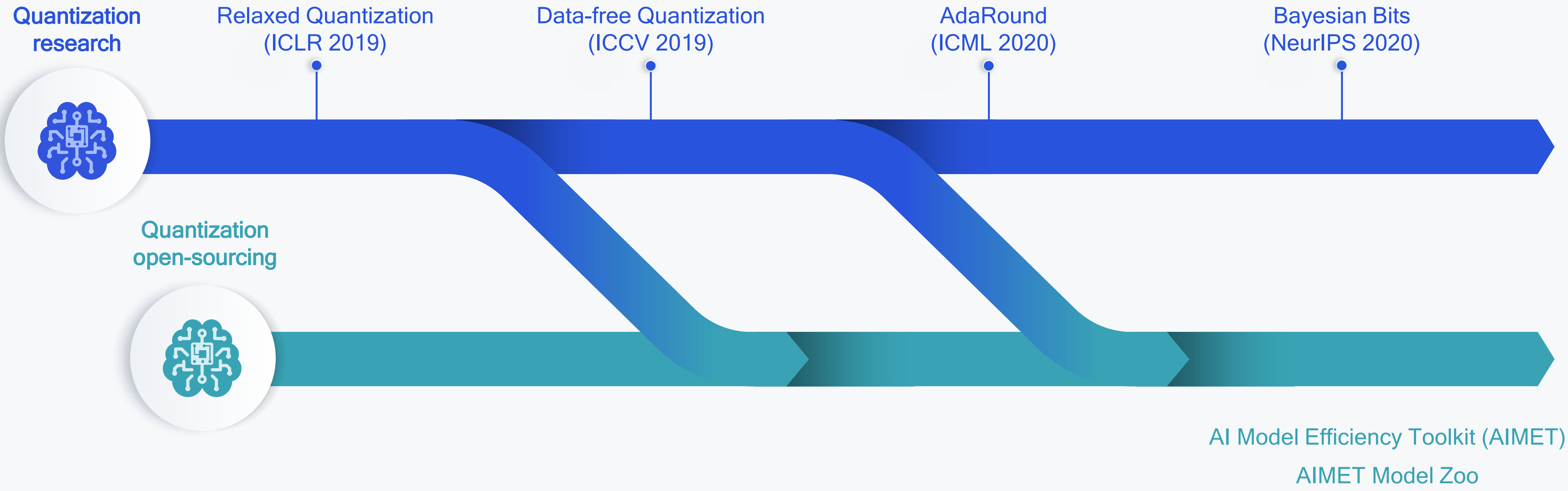
DONNA
Conclusions

DONNA shrinks big networks in a hardware-efficient way

DONNA can be rerun for any new device or setting within a day

DONNA works on many different tasks out of the box

DONNA enables scalability and allows models to be easily updated after small changes rather than starting from scratch



Leading AI research and fast commercialization

Driving the industry towards integer inference and power-efficient AI

AIMET and AIMET Model Zoo are products of Qualcomm Innovation Center, Inc.

AIMET & AIMET Model Zoo

Open-source projects to scale model-efficient AI to the masses

AIMET makes AI models small

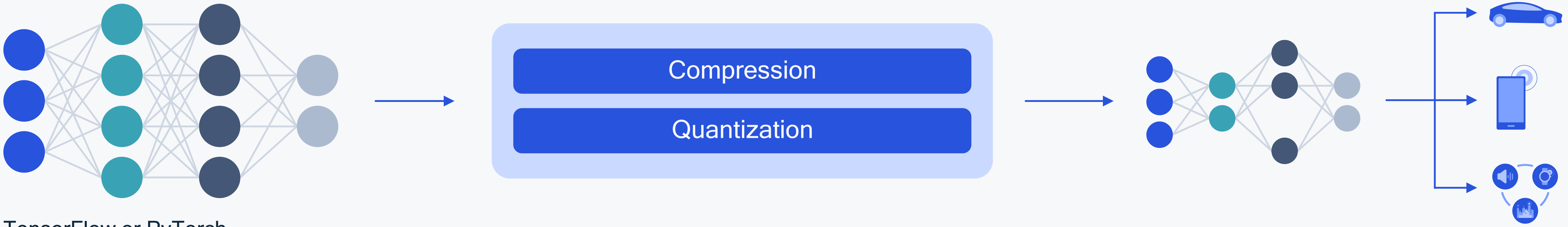
Open-sourced GitHub project that includes state-of-the-art quantization and compression techniques from Qualcomm AI Research

Trained AI model

AI Model Efficiency Toolkit (AIMET)

Optimized AI model

Deployed AI model



TensorFlow or PyTorch

If interested, please join the AIMET GitHub project: https://github.com/quic/aimet

Features:

- State-of-the-art network compression tools
- State-of-the-art quantization tools
- Support for both TensorFlow and PyTorch
- Benchmarks and tests for many models
- Developed by professional software developers

AIMET

Providing advanced model efficiency features and benefits

Benefits

- Lower power
- Lower memory bandwidth
- Maintains model accuracy
- Lower storage
- Higher performance
- Simple ease of use

Features

Quantization

State-of-the-art INT8 and INT4 performance

Post-training quantization methods, including Data-Free Quantization and Adaptive Rounding (AdaRound) – *coming soon*

Quantization-aware training

Quantization simulation

Compression

Efficient tensor decomposition and removal of redundant channels in convolution layers

Spatial singular value decomposition (SVD)

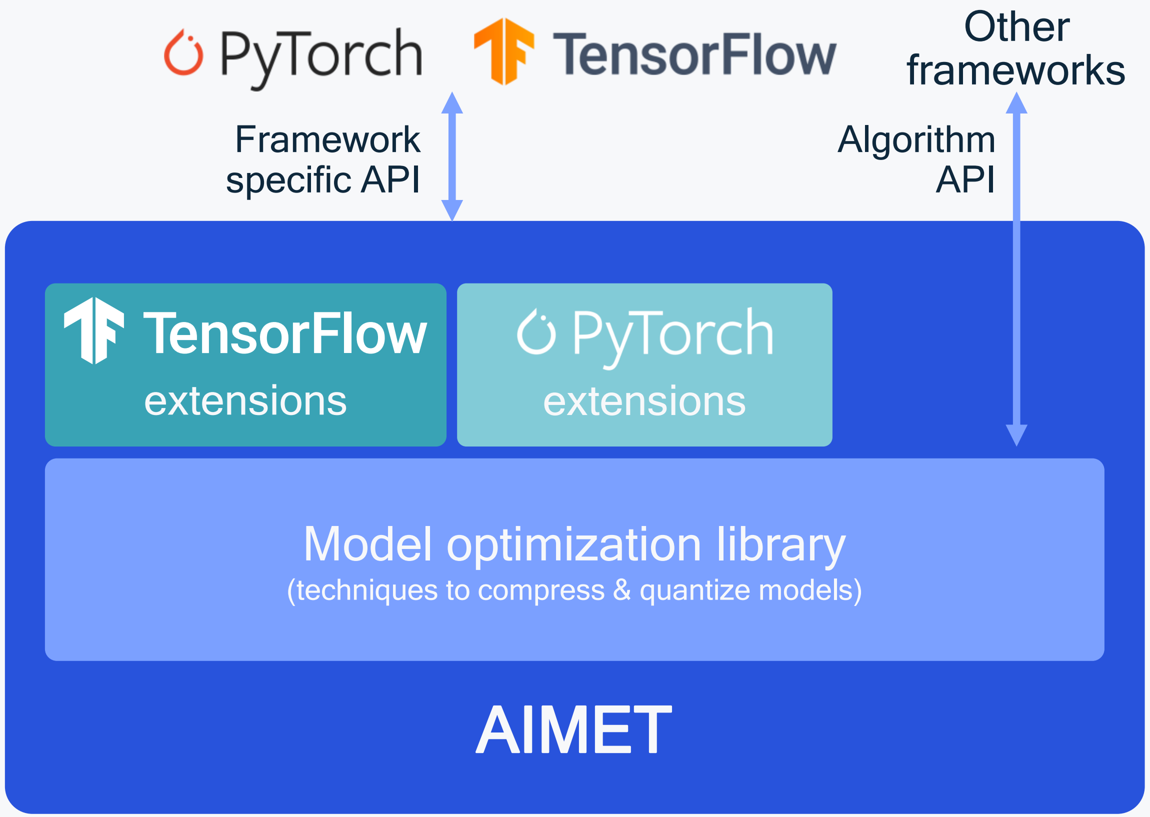
Channel pruning

Visualization

Analysis tools for drawing insights for quantization and compression

Weight ranges

Per-layer compression sensitivity



APIs invoked directly from the pipeline

Supports TensorFlow and PyTorch

Direct algorithm API frameworks

User-friendly APIs

```
compress_model (model,
        eval_callback=obj_det_eval,
        compress_scheme=Scheme.spatial_svd, ... )

equalize_model (model, ...)
```

AIMET features and APIs are easy to use

Designed to fit naturally in the AI model development workflow for researchers, developers, and ISVs

Data Free Quantization results in AIMET

Post-training technique enabling INT8 inference with very minimal loss in accuracy

Cross-layer equalization	Bias absorption	Weight quantization	Bias correction	Activation range estimation
Equalize weight ranges	Avoid high activation ranges		Measure and correct shift in layer outputs	Estimate activation ranges for quantization

DFQ example results

% Reduction in accuracy between FP32 ad INT8

<1%	<1%	<1%
MobileNet-v2 (top-1 accuracy)	ResNet-50 (top-1 accuracy)	DeepLabv3 mean intersection over union)

DFQ: data free quantization

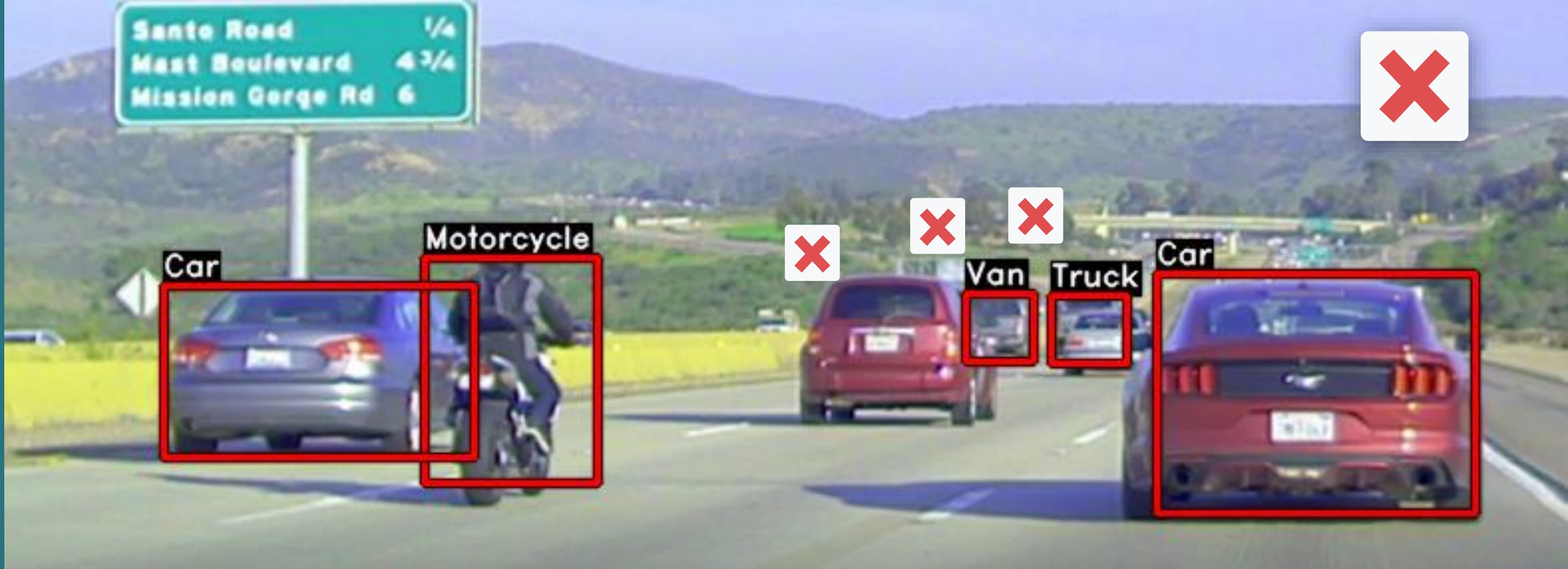
AdaRound is coming soon to AIMET

Post-training technique that makes INT8 quantization more accurate and INT4 quantization possible

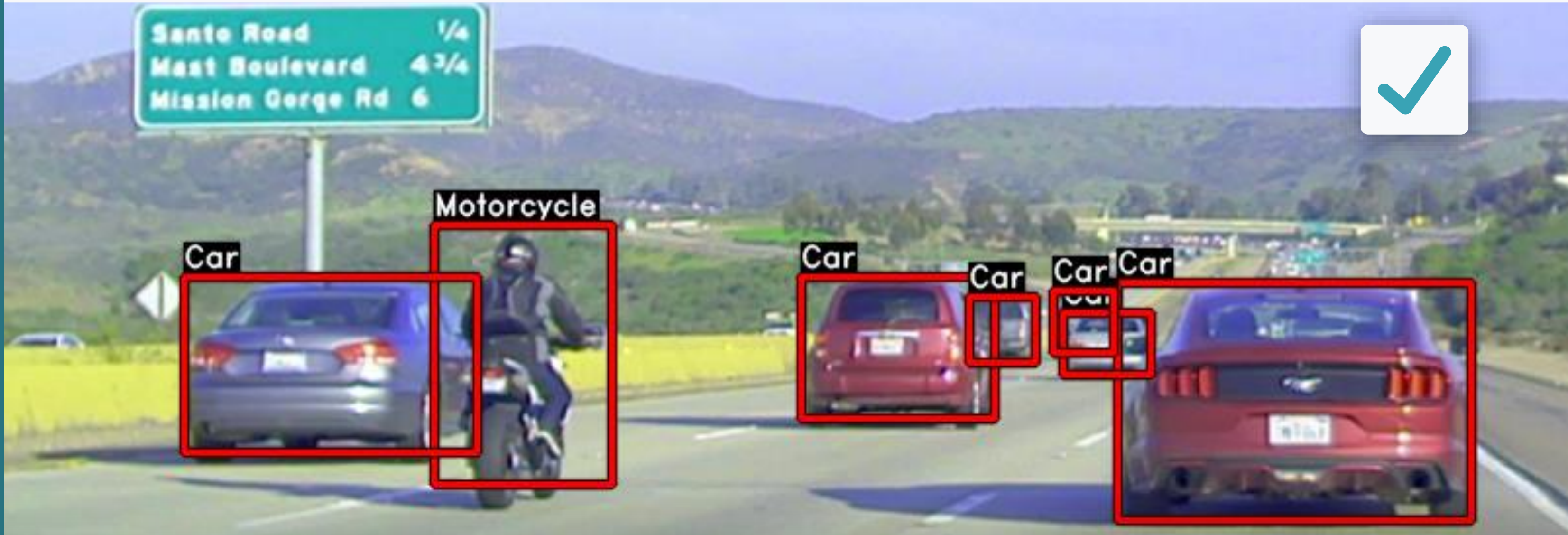
Bitwidth	Mean AP (mAP)
FP32	82.20
INT8 baseline quantization	49.85
INT8 AdaRound quantization	81.21

<1%

Reduction in accuracy between FP32 ad INT8 AdaRound quantization



INT8, baseline quantization



INT8, AdaRound quantization

AP: Average Precision


Camera Roll
Camera Roll
Camera Roll
THOUGHT PROVOKING
MULTILINGUAL
CULTURED
DIVERSE
ASIAN PACIFIC AMERICAN HERITAGE MONTH
Search
Create
Scan
Browse
Explore
Qualcomm
snapdragon

AIMET Model Zoo

Accurate pre-trained 8-bit quantized models

- Image classification
- Semantic segmentation
- Super resolution
- Object detection
- Pose estimation
- Speech recognition

AIMET Model Zoo includes popular quantized AI models

Accuracy is maintained for INT8 models – less than 1% loss*

<1% Loss in accuracy*

TensorFlow

Model	FP32	INT8	Metric
ResNet-50 (v1)	75.21%	74.96%	Top-1 accuracy*
MobileNet-v2-1.4	75%	74.21%	Top-1 accuracy*
EfficientNet Lite	74.93%	74.99%	Top-1 accuracy*
SSD MobileNet-v2	0.2469	0.2456	mAP*
RetinaNet	0.35	0.349	mAP*
Pose estimation	0.383	0.379	mAP*
SRGAN	25.45	24.78	PSNR*

PyTorch

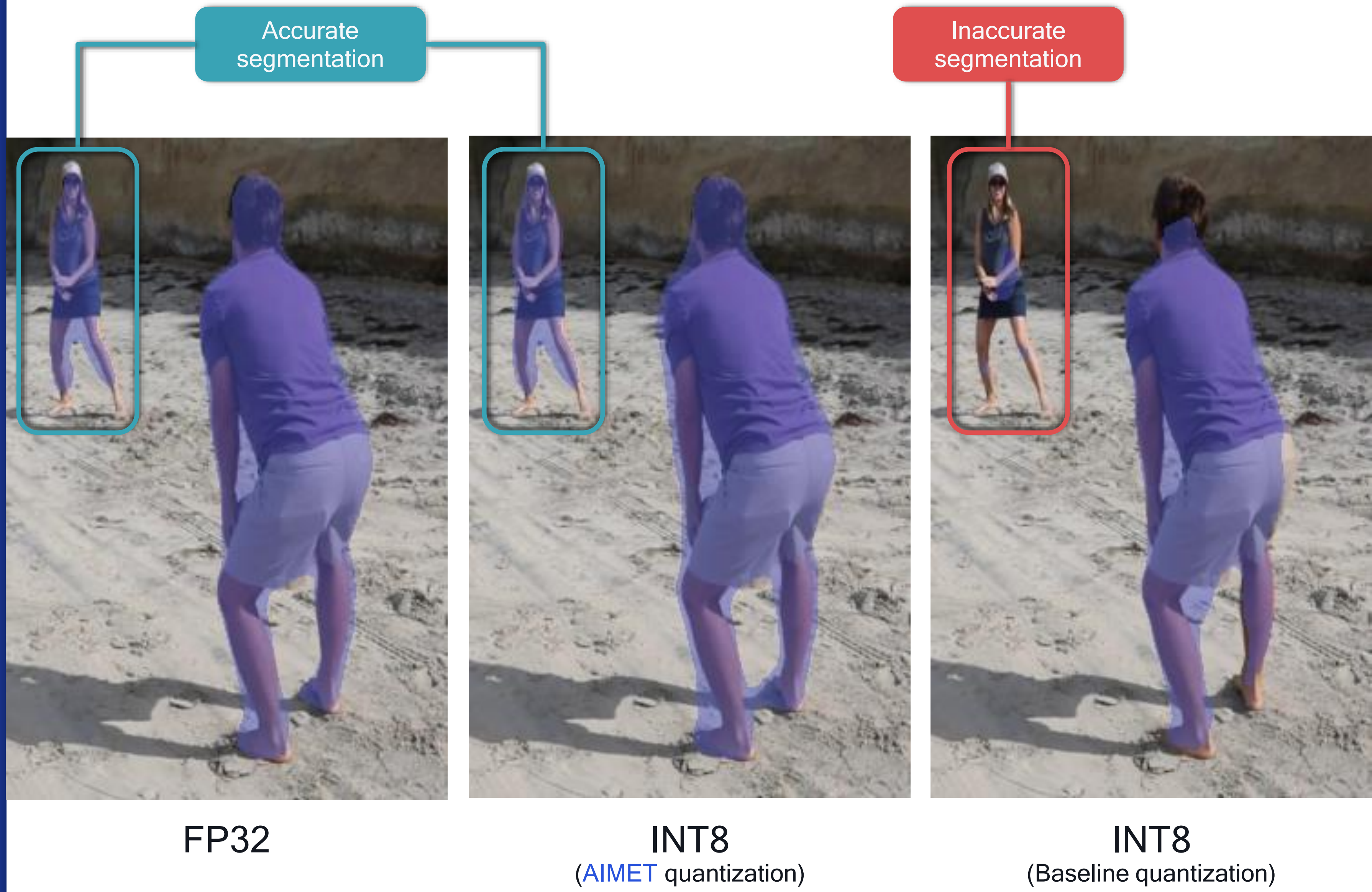
Model	FP32	INT8	Metric
MobileNetV2	7167%	71.14%	Top-1 accuracy*
EfficientNet-lite0	75.42%	74.44%	Top-1 accuracy*
DeepLabV3+	72.62%	72.22%	mIoU*
MobileNetV2-SSD-Lite	68.7%	68.6%	mAP*
Pose estimation	0.364	0.359	mAP*
SRGAN	25.51	25.5	PSNR
DeepSpeech2	9.92%	10.22%	WER*

*: Comparison between FP32 model and INT8 model quantized with AIMET.
For further details, check out: https://github.com/quic/aimet-model-zoo/

AIMET Model Zoo models preserve accuracy

Visual difference in model accuracy is telling between AIMET and baseline quantization methods

For DeepLabv3+ semantic segmentation, AIMET quantization maintains accuracy, while baseline quantization method is inaccurate

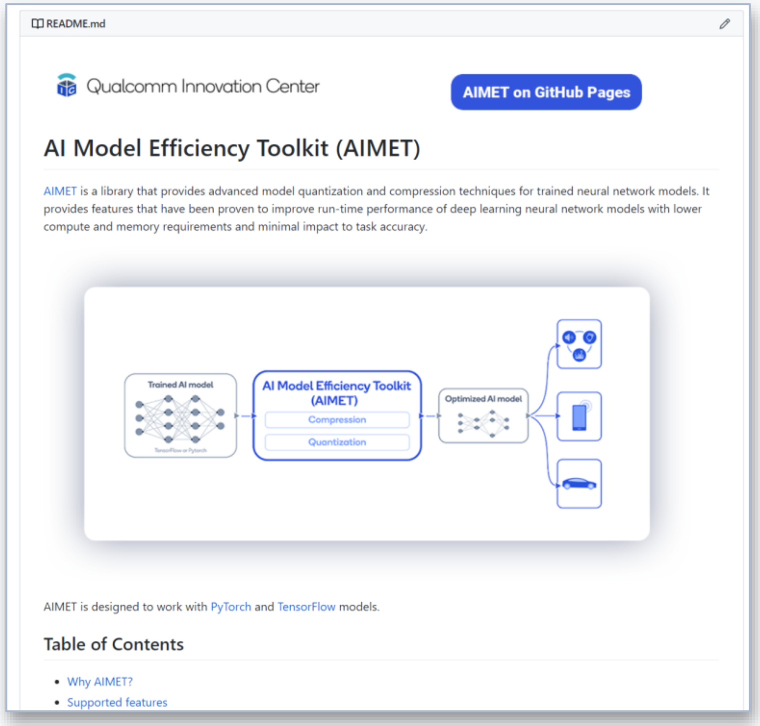


Baseline quantization: Post-training quantization using min-max based quantization grid

AIMET quantization: Model fine-tuned using Quantization Aware Training in AIMET

AIMET

State-of-the-art quantization and compression techniques


README.md

Qualcomm Innovation Center

AIMET on GitHub Pages

AI Model Efficiency Toolkit (AIMET)

AIMET is a library that provides advanced model quantization and compression techniques for trained neural network models. It provides features that have been proven to improve run-time performance of deep learning neural network models with lower compute and memory requirements and minimal impact to task accuracy.

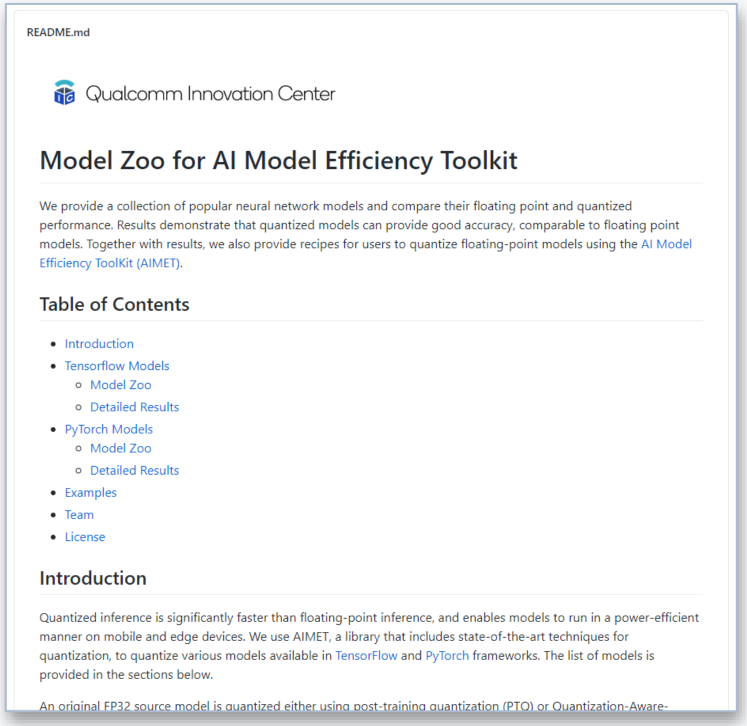
AIMET is designed to work with PyTorch and TensorFlow models.

Table of Contents

github.com/quic/aimet

AIMET Model Zoo

Accurate pre-trained 8-bit quantized models


README.md

Qualcomm Innovation Center

Model Zoo for AI Model Efficiency Toolkit

We provide a collection of popular neural network models and compare their floating point and quantized performance. Results demonstrate that quantized models can provide good accuracy, comparable to floating point models. Together with results, we also provide recipes for users to quantize floating-point models using the AI Model Efficiency ToolKit (AIMET).

Table of Contents

Introduction

Quantized inference is significantly faster than floating-point inference, and enables models to run in a power-efficient manner on mobile and edge devices. We use AIMET, a library that includes state-of-the-art techniques for quantization, to quantize various models available in TensorFlow and PyTorch frameworks. The list of models is provided in the sections below.

An original FP32 source model is quantized either using post-training quantization (PTQ) or Quantization-Aware-

github.com/quic/aimet-model-zoo

Join our open-source projects

Qualcomm

AI model efficiency is crucial for making AI ubiquitous, leading to smarter devices and enhanced lives

We are conducting leading research and development in AI model efficiency while maintaining accuracy

Our open-source projects, based on this leading research, are making it possible for the industry to adopt efficient AI models at scale

Questions?

Connect with Us

www.qualcomm.com/ai

www.qualcomm.com/news/onq

@QCOMResearch

https://www.youtube.com/qualcomm?

http://www.slideshare.net/qualcommwirelessevolution

Qualcomm

Thank you

Follow us on:

For more information, visit us at:

www.qualcomm.com & www.qualcomm.com/blog