



arm

Arm ML Research Project Overview

Mark O'Connor

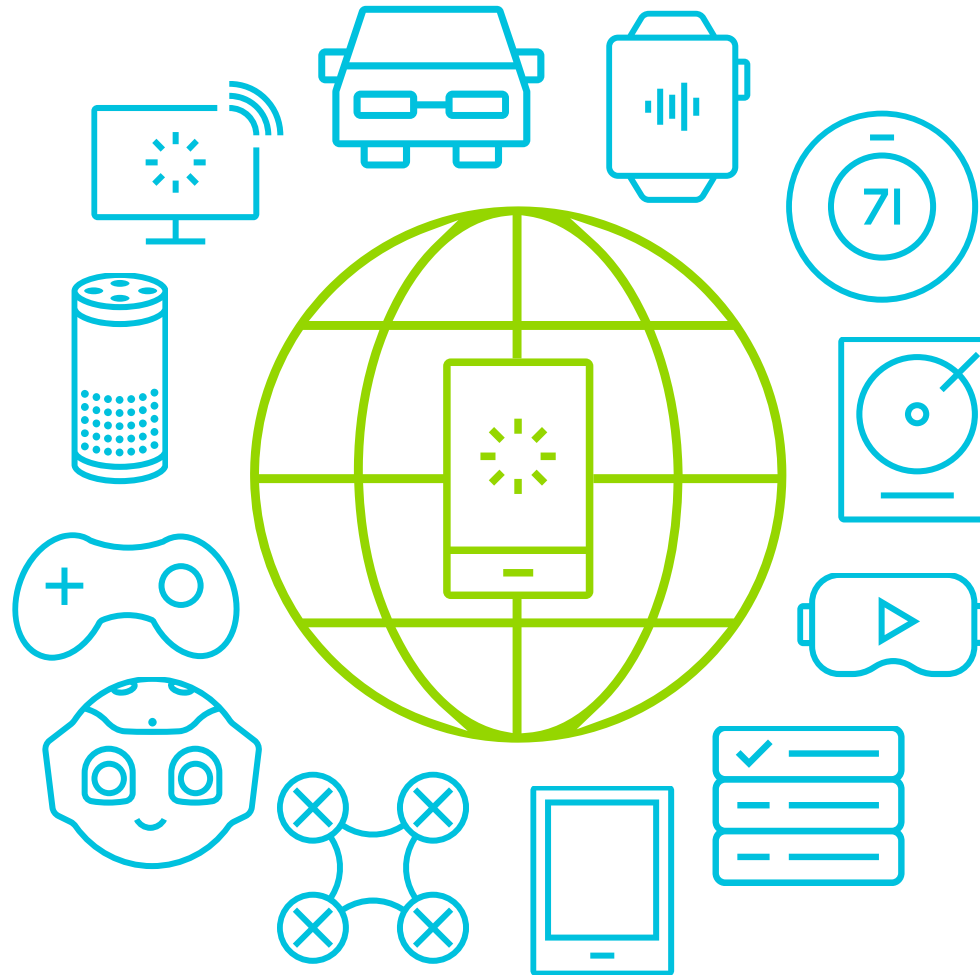
Senior Principal Researcher

Arm Machine Learning Research Lab

Arm Provides Compute

#1

shipping GPU in
the world is
Mali



> 5Bn

people using
Arm-based
mobile phones

23Bn

Arm-based
chips shipped
in 2018

146Bn

Arm-based
chips to date



Neural Networks as Software 2.0

A **fundamental shift** in how we write software

Program is **learned** rather than written

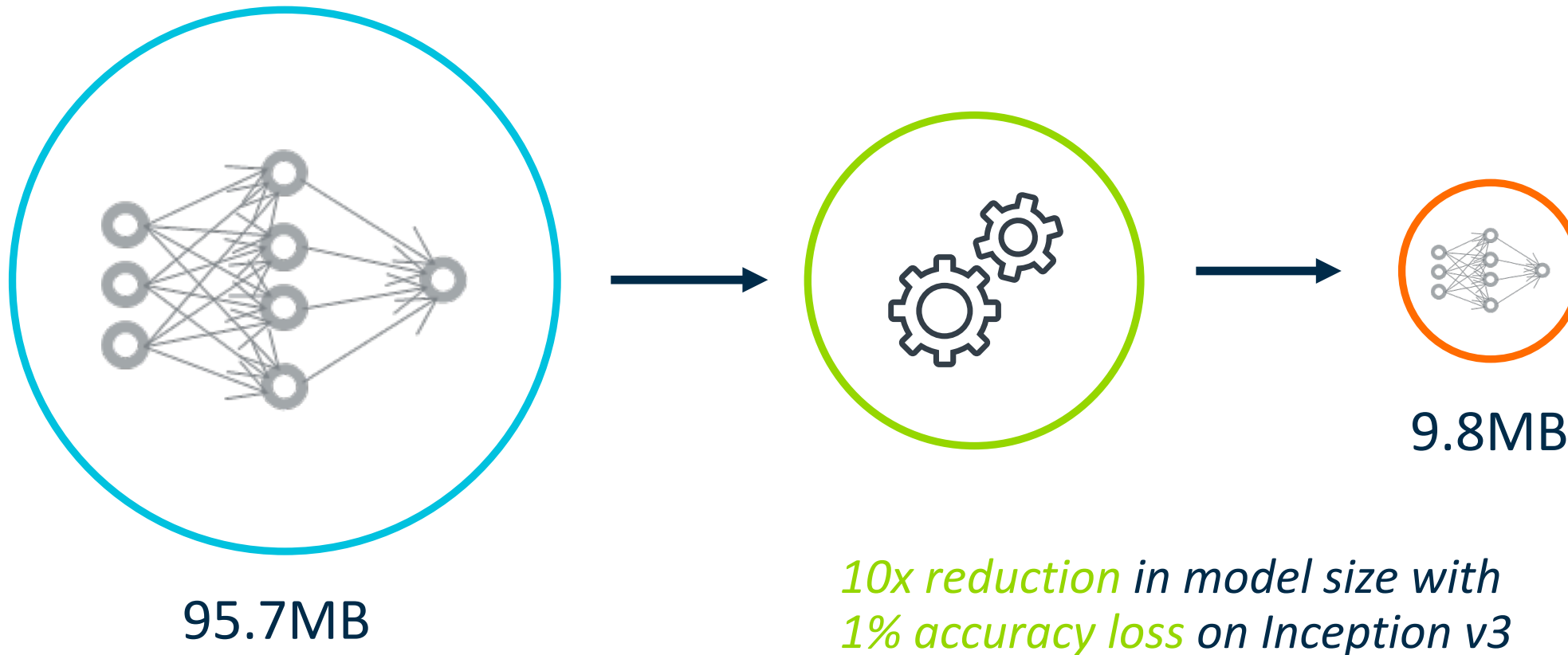
Training compiles **data** directly into **weights**

Weights execute as a **computational graph**

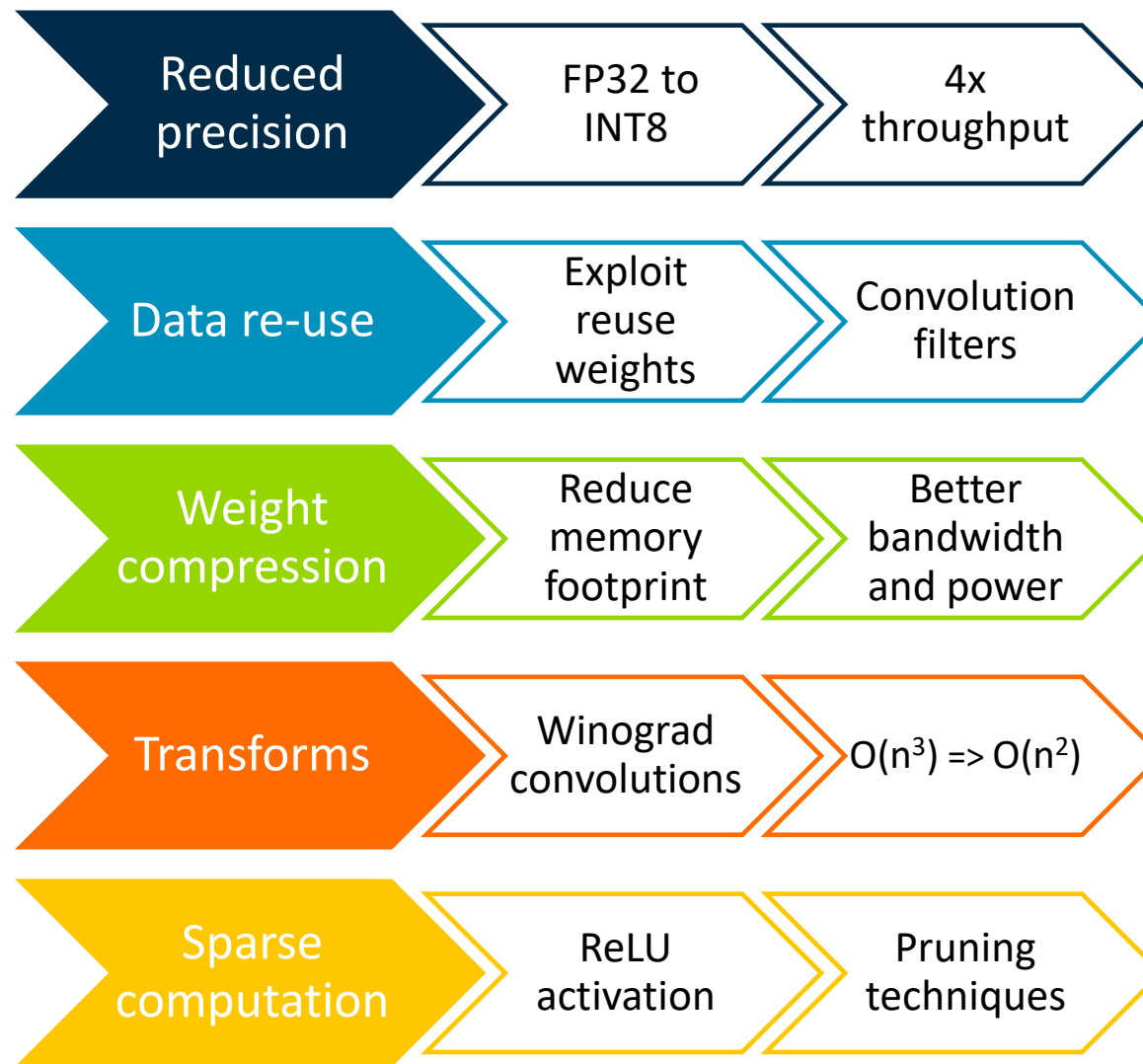
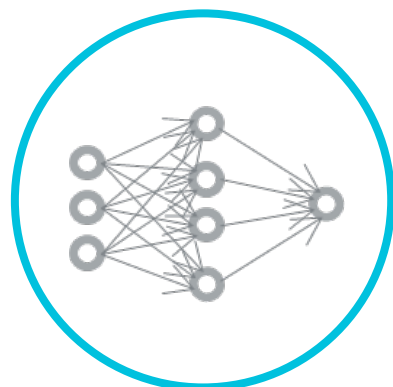
Andrej Karpathy
Director of AI at Tesla

Pete Warden
TensorFlow Engineer at Google

Software 2.0 is Surprisingly Adaptable



Achieving Trillions of Operations-per-Second on a Mobile Platform



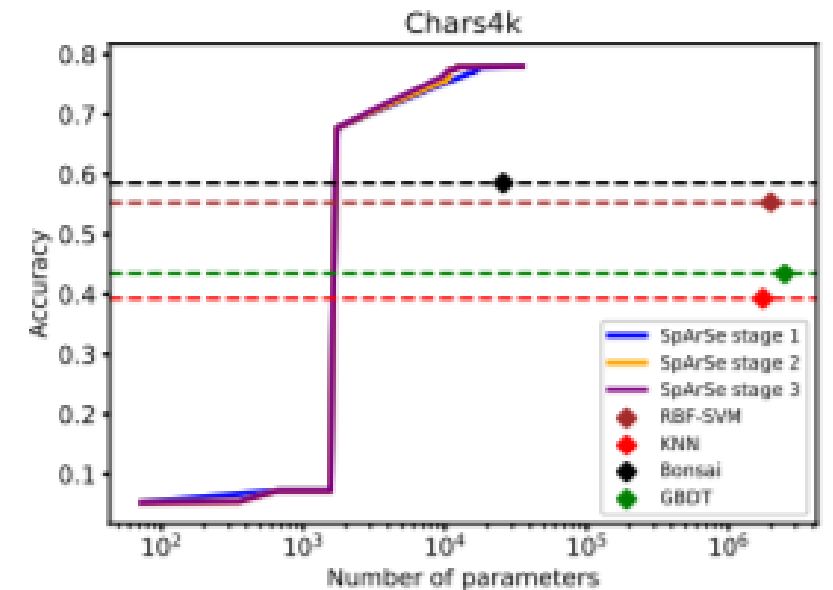
TinyML Hardware Trends

- Low-power general purpose cores and SoCs
 - Wide (vector) registers to support machine learning workloads
 - Dot-product and matrix multiply ops
 - NPU
 - Hardware support for lower precision ($\leq 4b$) operations
 - Hardware support for sparsity/zero-skipping
 - Hardware support for compression of weights and activations
 - Analog & Compute-in-memory
 - First commercial products appearing
- Amortize front-end overhead
- Key operation in all neural networks
- Recent research shows minimal accuracy loss with 4b
- ReLU activation and weight sparsity
- Bandwidth bottleneck, esp. FC
-
- The diagram consists of several blue arrows pointing from explanatory text on the right to specific hardware trends on the left. One arrow points from 'Amortize front-end overhead' to 'Wide (vector) registers to support machine learning workloads'. Another arrow points from 'Key operation in all neural networks' to 'Dot-product and matrix multiply ops'. A third arrow points from 'Recent research shows minimal accuracy loss with 4b' to 'Hardware support for lower precision (<=4b) operations'. A fourth arrow points from 'ReLU activation and weight sparsity' to 'Hardware support for sparsity/zero-skipping'. A fifth arrow points from 'Bandwidth bottleneck, esp. FC' to 'Hardware support for compression of weights and activations'.

SpArSe: Sparse Architecture Search for CNNs on Resource-Constrained Microcontrollers

- Microcontrollers are small, ubiquitous processors:
 - At the heart of the Internet of Things
 - Limited RAM and ROM – as little as 10s of KB
- We want to find CNN designs that:
 - Fit on microcontrollers
 - Deliver state-of-the-art accuracy
- The SpArSe framework:
 - Combines network architecture search and network pruning
 - Multi-objective Bayesian optimizer generates configurations
 - Finds highly accurate neural net models with significantly fewer parameters
 - Presented at NeurIPS 2019 in Vancouver, Canada

Processor	Usecase	Compute	Memory	Power	Cost
Nvidia 1080Ti GPU [3]	Desktop	10 TFLOPs/Sec	11 GB	250 W	\$700
Intel i9-9900K CPU [6, 5]	Desktop	500 GFLOPs/Sec	256 GB	95 W	\$499
Google Pixel 1 (Arm CPU) [10]	Mobile	50 GOPs/Sec	4 GB	~ 5 W	-
Raspberry Pi (Arm CPU) [11]	Hobbyist	50 GOPs/Sec	1 GB	1.5 W	-
Micro:Bit (Arm MCU) [8]	IoT	16 MOPs/Sec	16 KB	~ 1 mW	\$1.75
Arduino Uno (Microchip MCU) [1]	IoT	4 MOPs/Sec	2 KB	~ 1 mW	\$1.14



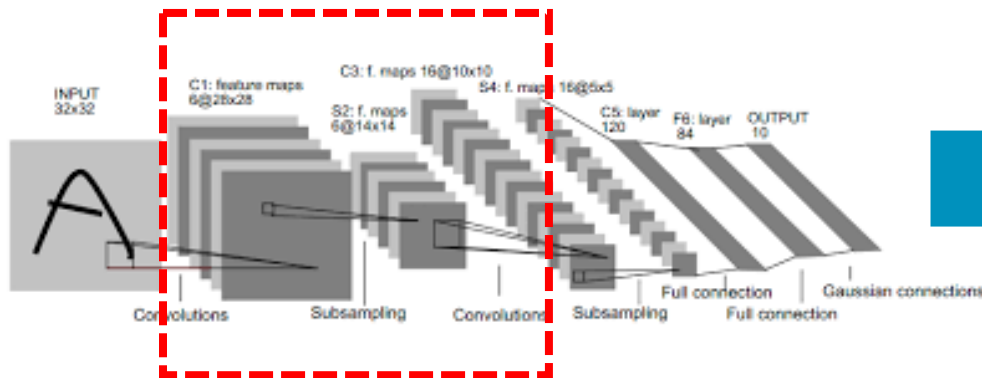
<https://arxiv.org/abs/1905.12107>

How Can We Improve Performance/Power/Area Further?

DeepFreeze: A hardware-generating backend for TensorFlow

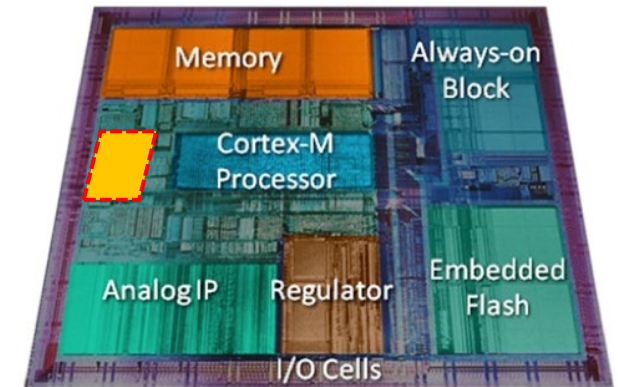
NN model in TensorFlow

Hardware accelerator in Verilog



Specified layers or whole model

DeepFreeze



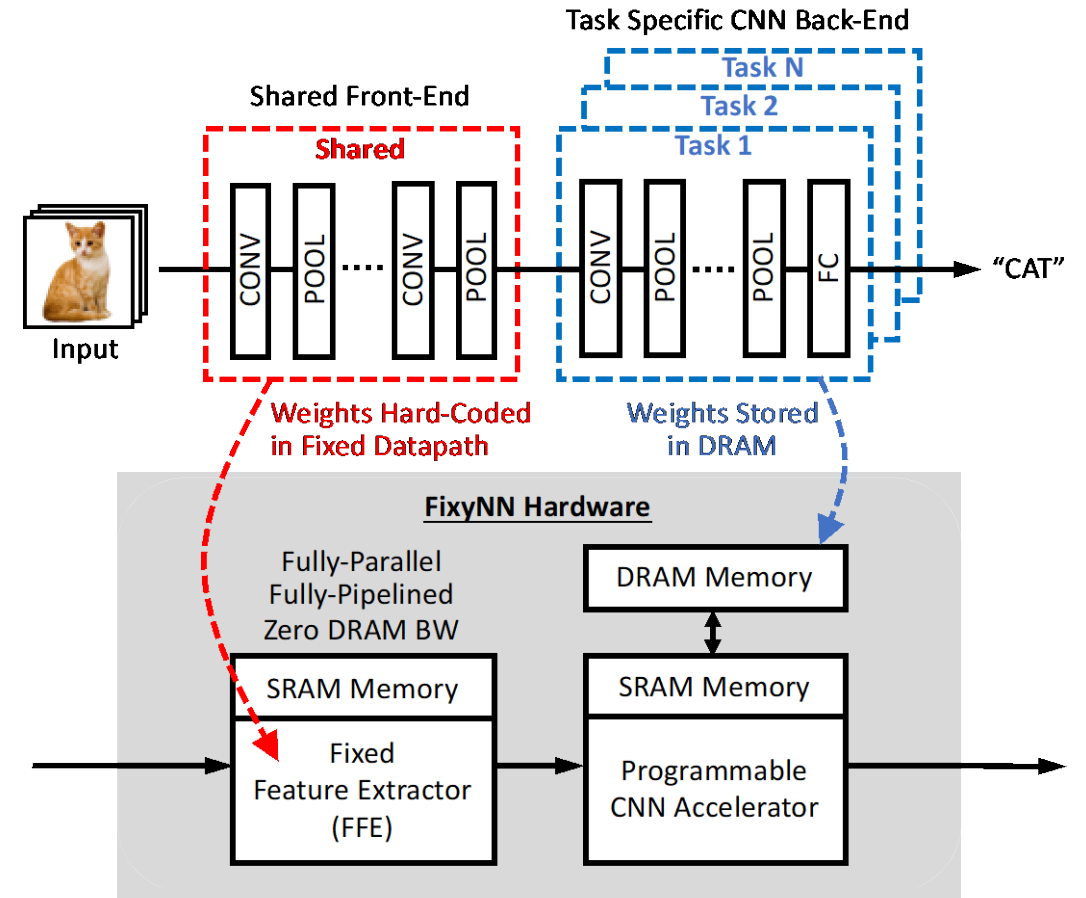
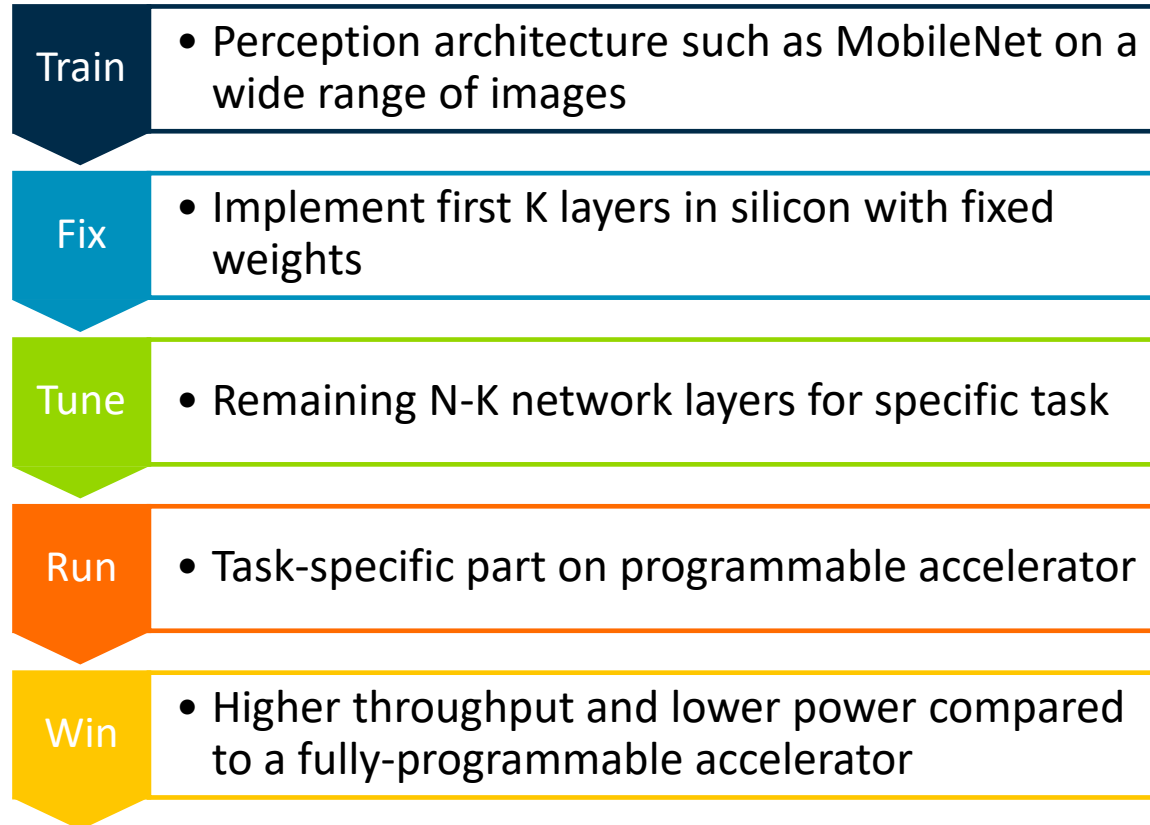
ASIC



FPGA

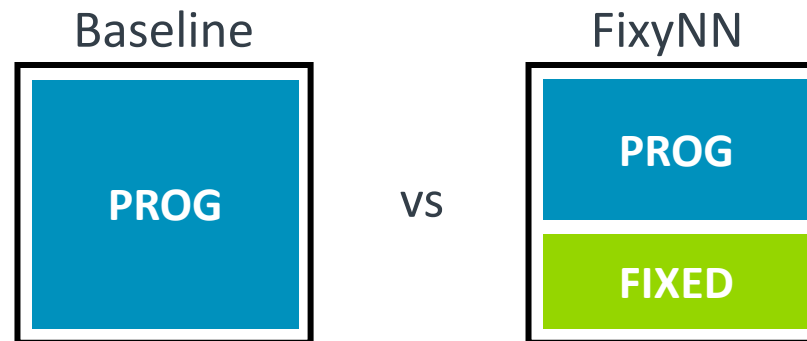
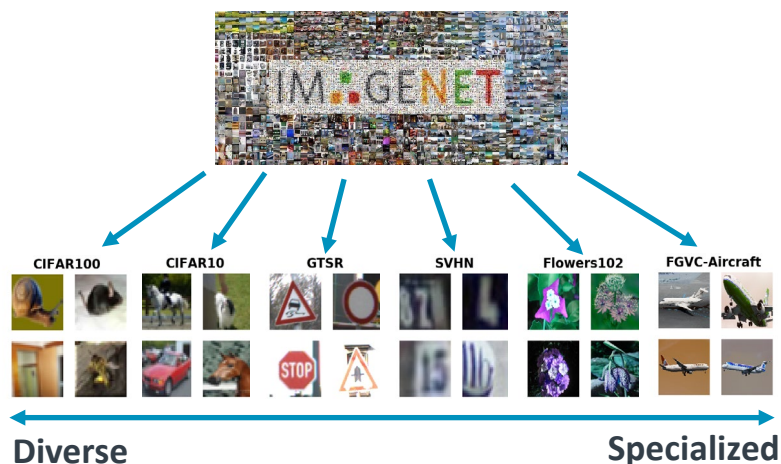
How Can We Improve Performance/Power/Area Further?

FixyNN: Leveraging Transfer Learning for Ultra-Efficient Hardware

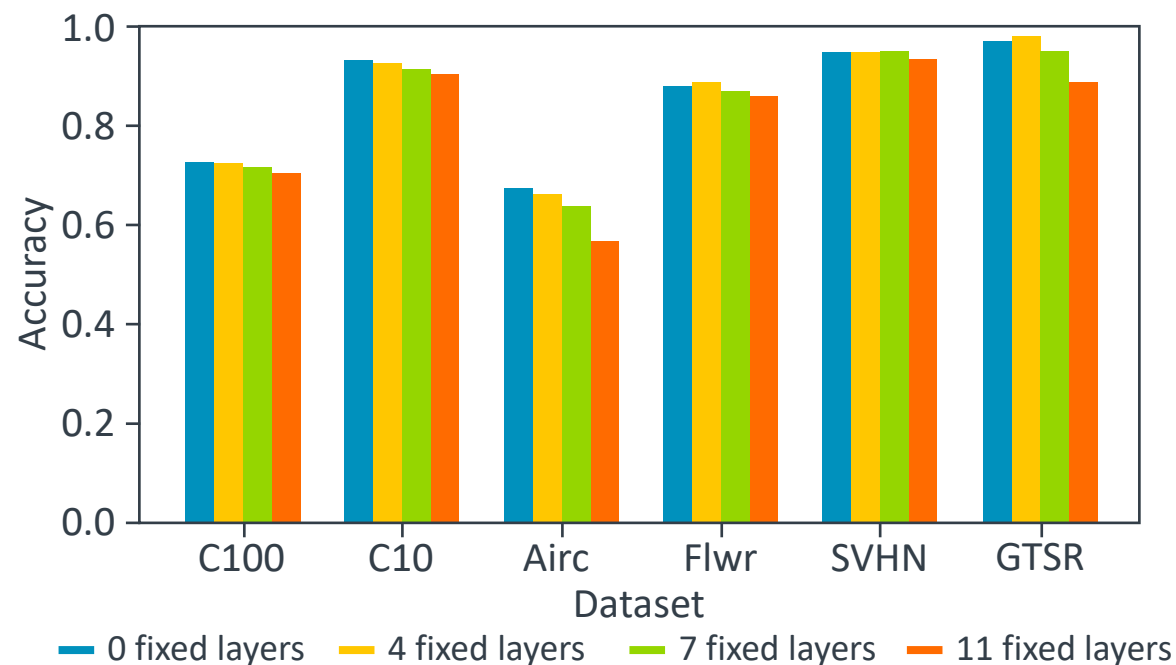


FixyNN Evaluation Results

- Energy efficiency of up to **11.2 TOPS/W** with <1% accuracy loss – nearly **2x more efficient** than NVDLA alone in same area
 - ~**1.5x** TOPS/W by fixing 4 layers
 - ~**2x** TOPS/W by fixing 7 layers
 - Accuracy loss of < 1% over six datasets



3mm² Area



Recent Publications from Arm ML Research Lab

SpArSe: Sparse Ternary Hybrid Neural-Tree Net IoT Applications

Igor Arm | igor.fed.

Dibakar Gope¹ Ganesh Dasika²

Machine learning-based applications are increasingly being deployed on edge devices with limited compute and storage capabilities. These applications often have strict real-time deadlines, so any compression technique that is used must not impact the inference run-time. Hybrid Matrix Decomposition (HMD) is a new compression technique that divides the matrix into two parts - an unconstrained upper half and a lower half composed of rank-1 blocks. This results in output features where the upper sub vector has "richer" features while the lower sub vector has "constrained" features. HMD can compress RNNs by 2x while being 2x faster than pruning and more accurate than a traditional matrix factorization technique, better enabling the deployment of "TinyML" applications.

The vast majority of things (IoT) are processed on tiny, cheap microcontroller units (MCUs), which find their way into everything from smart-home devices to industrial machines. This paper challenges the current state-of-the-art results on IoT datasets. The CNNs we find in previous approaches, while meeting

1 INTRODUCTION

Machine learning algorithms, and neural networks in particular, are increasingly deployed in IoT devices. Popular applications include faces in smart-home devices, predictive maintenance in commercial and industrial machines, wearables, etc. However, due to the compute limitations of highly constrained IoT devices, they are frequently limited to simple requests or to a server. In addition to SRAM, tend to be "always-on" constrained power sources. Reducing the computation and storage for IoT applications is of interest to ensure a longer battery!

To enable this compact models, one particular use of depthwise-separable convolution layers see these layers be applied in keyword-spotting applications in state-of-the-art cases.

While DS convolutions are highly resource constrained, Table 1 compares the broad proliferation of MCUs relative to desktop processors.

RNN Compression

Urmish Thakker, Ganesh Dasika, Machine Learning

Abstract—RNNs can be large and computationally intensive. Yet, many applications that use RNNs run on edge devices with very limited compute and storage capabilities. These applications often have strict real-time deadlines, so any compression technique that is used must not impact the inference run-time. Hybrid Matrix Decomposition (HMD) is a new compression technique that divides the matrix into two parts - an unconstrained upper half and a lower half composed of rank-1 blocks. This results in output features where the upper sub vector has "richer" features while the lower sub vector has "constrained" features. HMD can compress RNNs by 2x while being 2x faster than pruning and more accurate than a traditional matrix factorization technique, better enabling the deployment of "TinyML" applications.

1. HYBRID MATRIX DECOMPOSITION (HMD)

Algorithm 1 Matrix vector product with HMD

Input 1: Matrices A', B, C, D and E

Input 2: Vector I

Output: Matrix O of dimension $M \times 1$

$O_{top_M/2_rows} \leftarrow A' \times I$

$Temp1 \leftarrow B \circ (C \cdot I_{top_M/2_rows})$

$Temp2 \leftarrow D \circ (E \cdot I_{bottom_M/2_rows})$

$O_{bottom_M/2_rows} \leftarrow Temp1 + Temp2$

HMD breaks a matrix into two parts - a fully parameterized upper half and a constrained lower half. Figure 1 gives a visual representation of this decomposition technique for a single matrix along with the parameters required for storage. This creates a dense matrix representation making it more hardware-friendly than pruning. Additionally, it creates a higher rank matrix than low rank matrix factorization (LMF), giving it more expressibility.

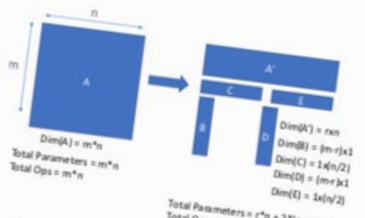


Fig. 1. Representation of a matrix decomposition.

Euphrates: Algorithm-SoC Co-Design for Low-Power Mobile Continuous Vision

Yuhao Zhu¹

Anand Samajdar²

Matthew Mattina³

Paul Whatmough³

¹University of Rochester
²Georgia Institute of Technology
³ARM Research

Abstract

Continuous computer vision (CV) tasks increasingly rely on convolutional neural networks (CNN). However, CNNs have massive compute demands that far exceed the performance and energy constraints of mobile devices. In this paper, we propose and develop an algorithm-architecture co-designed system, Euphrates, that simultaneously improves the energy-efficiency and performance of continuous vision tasks. Our key observation is that changes in pixel data between consecutive frames represents visual motion. We first propose an algorithm that leverages this motion information to relax the number of expensive CNN inferences required by continuous vision applications. We co-design a mobile System-on-a-Chip (SoC) architecture to maximize the efficiency of the new algorithm. The key to our architectural augmentation is to co-optimize different SoC IP blocks in the vision pipeline collectively. Specifically, we propose to expose the motion data that is naturally generated by the CNN engine. Measurement and synthesis results show that Euphrates achieves up to 66% SoC-level energy savings (4x for the vision computations), with only 1% accuracy loss.

Fig. 2. Comparison of Euphrates with other approaches.

HMD will be used in the vision pipeline to reduce the number of matrix multiplications. Its weight matrix decomposition is shown in Figure 1. The decomposition of matrix A into A', B, C, D, and E is shown in Figure 1. The decomposition of matrix A into A', B, C, D, and E is shown in Figure 1.

1. Introduction

Computer vision (CV) is the cornerstone of many emerging application domains, such as advanced driver-assistance systems (ADAS) and augmented reality (AR). Traditionally, CV algorithms were dominated by hand-crafted features (e.g., Haar [109] and HOG [55]), coupled with a classifier such as support vector machine (SVM) [54]. These algorithms have low complexity and are practical in constrained environments, but only achieve moderate accuracy. Recently, convolutional neural networks (CNNs) have rapidly displaced hand-crafted feature extraction, demonstrating significantly higher accuracy on a range of CV tasks including image classification [104], object detection [85, 97, 99], and visual tracking [56, 91]. This paper focuses on continuous vision applications that extract high-level semantic information from real-time video streams. Continuous vision is challenging for mobile architectures due to its enormous compute requirement [117]. Using Euphrates as an example, Fig. 1 shows the compute requirements measured in Tera Operations Per Second (TOPS) as well as accuracies between different detectors under 60 frames per second (FPS). As a reference, we also overlay the 1 TOPS line, which represents the peak compute capability that today's CNN accelerators offer under a typical 1 W mobile power budget [21, 41]. We find that today's CNN-based approaches such as YOLOv2 [98], SSD [85], and Faster R-CNN [99] all have at least one order of magnitude higher compute requirements than accommodated in a mobile device. Reducing the CNN complexity (e.g., Tiny YOLO [97], which is a heavily truncated version of YOLO with 9/22 of its layers) or falling back to traditional hand-crafted features such as Haar [61] and HOG [113] lowers the compute demand, which however, comes at a significant accuracy penalty.

The goal of our work is to improve the compute efficiency of continuous vision with small accuracy loss, thereby enabling new mobile use cases. The key idea is to exploit the motion inherent in real-time videos. Specifically, today's continuous vision algorithms treat each frame as a standalone entity and thus execute an entire CNN inference on every frame. However, pixel changes across consecutive frames are not arbitrary; instead, they represent visual object motion. We propose a new algorithm that leverages the temporal pixel motion to synthesize vision results with little computation while avoiding expensive CNN inferences on many frames. Our main architectural contribution in this paper is to co-

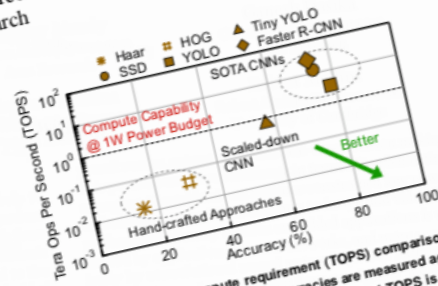


Fig. 1: Accuracy and compute requirement (TOPS) comparison between object detection techniques. Accuracies are measured against the widely-used PASCAL VOC 2007 dataset [32], and TOPS is based on the 480p (640 x 480) resolution common in smartphone cameras.

[1] N. Hammerla, S. Halloran, and T. Ploetz, "recurrent models for human action recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1059-1067.

[2] F. Orban, "Human action recognition as an example, Fig. 1 shows the compute requirements measured in Tera Operations Per Second (TOPS) as well as accuracies between different detectors under 60 frames per second (FPS). As a reference, we also overlay the 1 TOPS line, which represents the peak compute capability that today's CNN accelerators offer under a typical 1 W mobile power budget [21, 41]. We find that today's CNN-based approaches such as YOLOv2 [98], SSD [85], and Faster R-CNN [99] all have at least one order of magnitude higher compute requirements than accommodated in a mobile device. Reducing the CNN complexity (e.g., Tiny YOLO [97], which is a heavily truncated version of YOLO with 9/22 of its layers) or falling back to traditional hand-crafted features such as Haar [61] and HOG [113] lowers the compute demand, which however, comes at a significant accuracy penalty.

arm

Thank You

Danke

Merci

谢谢

ありがとう

Gracias

Kiitos

감사합니다

धन्यवाद

شكرًا

ধন্যবাদ

תודה

arm

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks