Accelerating Quantized Vision Inference on ARM Through Optimized Pointwise and Depthwise Convolutions

Deeksha Kasture[1], Ashish Chopra[1], Masahiro Doteguchi[2], Atsushi Nukariya[1], Priyanka Sharma[1]

[1] Fujitsu Research of India Pvt. Ltd. (FRIPL), [2] Fujitsu Limited

Abstract

- Deep learning models are increasingly deployed on resource constrained devices, making **INT8 quantized CPU inference** critical for performance and energy efficiency.
- Efficient CNNs (MobileNet, EfficientNet, ShuffleNet, MnasNet) are dominated by **pointwise and depthwise convolutions** on CPUs.
- Existing ARM CPU backends lack **efficient INT8 kernels** for both operators.
- We present **JIT generated INT8 SVE kernels** for pointwise and depthwise convolutions.
- **BRGEMM** is used for pointwise convolution; **BRDGEMM** captures the diagonal, channel-wise structure of depthwise convolution.
- **INT8 dot-product** instructions replace FP32 FMLA, significantly improving throughput and vector efficiency.
- Kernels are integrated into **oneDNN opensource library** and evaluated on ARM systems.
- Results show up to **3x speedup over FP32 and 7x over baseline INT8**, improving end to end inference performance.

Methodology

POINTWISE CONVOLUTION

In pointwise (1x1) convolution, inputs and weights are reshaped into 2D matrices for matrix multiplication; since the **kernel is 1x1**, output height and width remain unchanged. We extend the **BRGEMM kernel [1]** with **INT8 support** to efficiently perform multiple batch reduced computation.

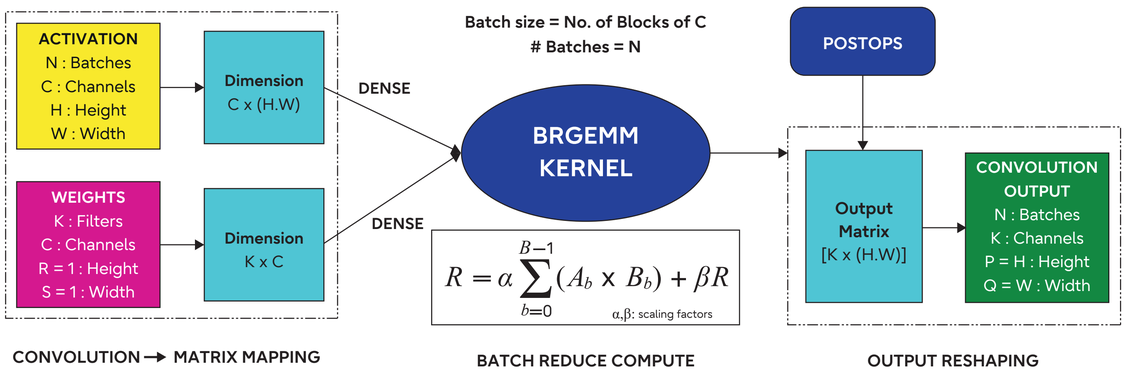

Figure 1: Pointwise convolution flow : leverages standard BRGEMM kernel with added INT8 support

DEPTHWISE CONVOLUTION

For input $NxCxHxW$ depthwise convolution uses **one filter per channel** ($K=C$), so output channels equal input with **no cross-channel mixing**. In matrix form, **weights are effectively diagonal**, reducing computation to channel-wise dot products - hence we use a dedicated **BRDGEMM kernel with INT8 support** instead of standard BRGEMM.

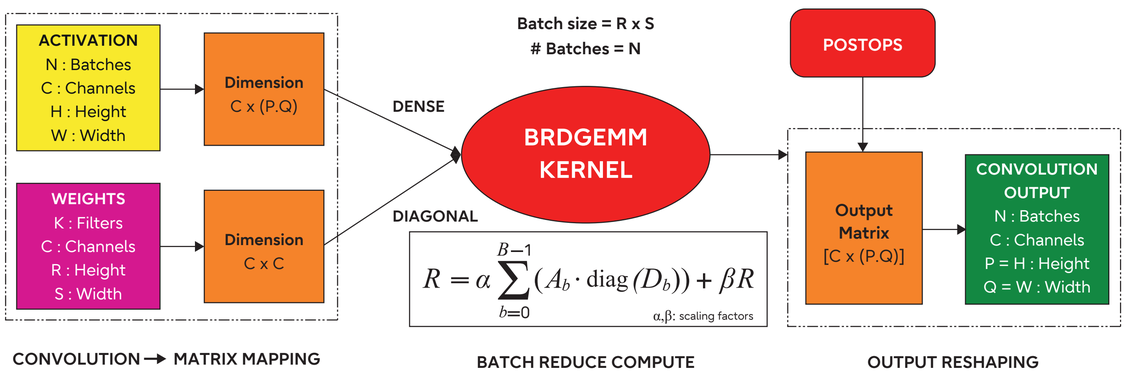

Figure 2: Depthwise convolution flow: channelwise computation using diagonal weights for optimized BRDGEMM with added INT8 support.

For both pointwise and depthwise convolutions, this work applies input blocking and add support for **INT8 inputs and weights** with **FP32, S32, and INT8 outputs**.

ARM SVE OPTIMIZATION

Pointwise and depthwise convolutions are implemented as JITed ARM SVE kernels with an INT8 path, improving throughput via **INT8 dot-product instructions** over FP32 FMLA. Accumulation is in INT32, followed by FP32 conversion for post-ops and final output conversion.

Instruction	INT8 Dot-Product	FP32 FMLA
Multiply accumulate per lane	4	1
Throughput	Higher	Lower
Efficiency	More compute-dense	Less compute -dense

Table 1: Comparison of Instruction used in int8 and fp32 implementation

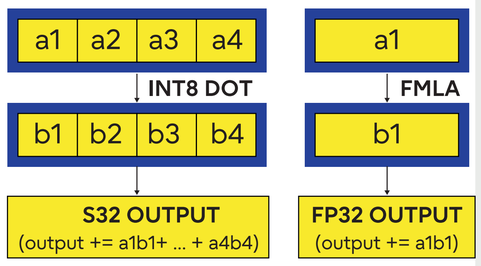

Figure 3: INT8 DOT accumulates 4 element products per lane, whereas FMLA accumulates a single element per lane.

Results

We measured performance at the model level using **PyTorch 2.7** and **TorchVision 0.24.0**, with our kernels integrated into **oneDNN** and invoked by upper level framework pytorch on a **Graviton 3E** machine with 64 cores.

For the evaluated models, pointwise convolutions account for ~60–70% of compute, with the remainder being depthwise convolutions. We measured **individual performance gains** for pointwise and depthwise layers, as well as **overall model level gains**. These gains are reported **relative to the default INT8 implementation**, which uses reference kernels without prior ARM optimizations as shown in **figure 4.**

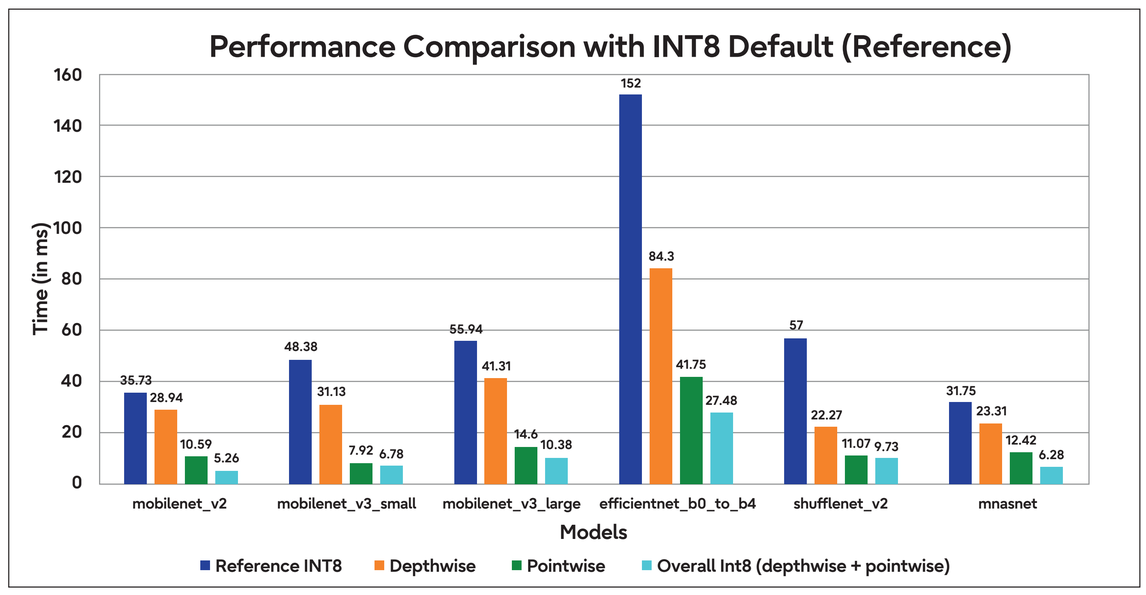

Figure 4 : Inference Speed-up over existing INT8 implementation.

We also compared overall performance against **FP32 optimized kernels** to quantify the benefits of INT8 low precision computation as shown in **figure 5.**

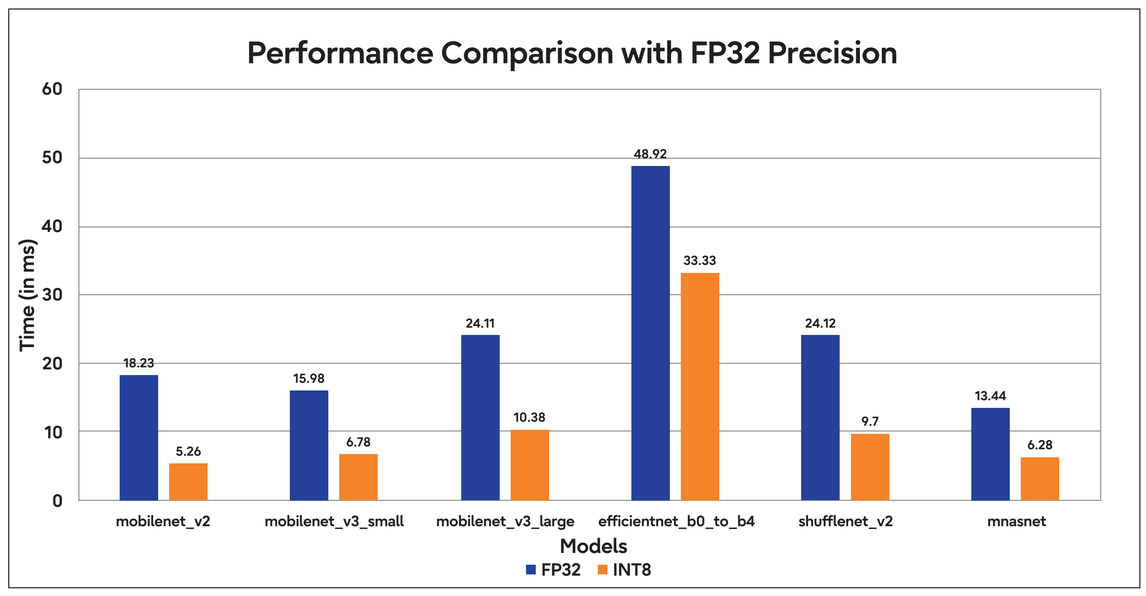

Figure 5 : Inference Speed-up over FP32 Implementation

Thus, the results demonstrate up to **7x speedup over the reference INT8 implementation** as shown in figure 4, and **up to 3x speedup over FP32** as shown in figure 5 highlighting the efficiency of our quantized convolutions.

Conclusion and Future Work

- Our work implements **quantized vision inference** for models heavily dominated by pointwise and depthwise convolutions, such as **MobileNet, ShuffleNet, MnasNet and EfficientNet.** The observed **speedup** both **FP32** and **baseline INT8 implementations** demonstrate the effectiveness of this approach for **resource-constrained systems.**
- We aim to **further optimize these convolutions** and **expand support to additional datatypes** to broaden applicability.

References

[1] Georganas, Evangelos & Banerjee, Kunal & Kalamkar, Dhiraj & Avancha, Sasikanth & Venkat, Anand & Anderson, Michael & Henry, Greg & Pabst, Hans & Heinecke, Alexander. (2019). High-Performance Deep Learning via a Single Building Block. [https://arxiv.org/abs/1906.06440]

[2] ARM A64 Instruction Set Architecture. [https://developer.arm.com/documentation/ddi0596/2021-06/SVE-Instructions]

[3] The QR codes below link to the corresponding oneDNN pull requests for this work.

Brgemm 1x1 Convolution FP32 | Brdgemm Convolution FP32 | BRGEMM INT8 | Brgemm 1x1 Convolution INT8 | Brdgemm Convolution INT8 | Brdgemm int8 destination

This poster is based on results obtained from a project, JPNP21029 subsidized by the New energy and Industrial Technology Development Organization (NEDO).