DeepliteRT: Computer Vision at the Edge

Saad Ashfaq, Alexander Hoffman, Saptarshi Mitra, Sudhakar Sah, MohammadHossein AskariHemmat, Ehsan Saboori

Deeplite Inc., Toronto, Canada

Motivation

Ultra low-bit quantization scales down model weights and activations from 32-bit full-precision to less than 8-bit, presenting an attractive solution for deep learning model deployment on edge devices. Using state-of-the-art techniques for quantization-aware training (QAT) such as LSQ, it is possible to quantize a convolutional neural network to 2 bits with minimal accuracy degradation.

Model	Top-1 Accuracy@32-bit	Top-1 Accuracy@2-bit				
		PACT (2018)	LQ-NET (2018)	QIL (2019)	PACT-SAWB (2019)	LSQ (2020)
ResNet18	70.5%	64.4%	65.2%	65.7%	67.0%	67.9%
ResNet50	76.9%	72.2%	71.5%		74.2%	74.6%

Table 1. 2-bit accuracy on ImageNet dataset with different QAT methods.

However, realizing inference on edge devices with sub-8-bit data remains challenging due to the following reasons.

- Lack of support for sub-8-bit data types on mainstream CPU architectures.
- Fake-quantization during QAT retains weights and activations in full-precision.
- Modifications required in training and inference paths to enable ultra low-bit deployment.

Contributions

We introduce DeepliteRT, an inference solution for the deployment of ultra low-precision sub-8-bit quantized models on ARM-based platforms, that makes the following contributions:

- **High performance bit-serial convolution kernels** defined as `dlrt_bitserial_conv2d` that achieve SOTA performance with up to 4.34×speedup over existing ultra low-bit methods on ARMv7 and ARMv8 CPUs.
- **Compiler passes** that automatically convert 32-bit convolution operators, weights and activations in fake-quantized models produced by QAT into ultra low-bit representations.
- **Mixed precision inference** by allowing the number of bits for weights and activations to be specified on a per-layer basis.

DeepliteRT makes ultra low-bit model deployment on edge devices highly accessible to ML practitioners as no code changes are required in the training, quantization, or inference paths.

Bitpacking

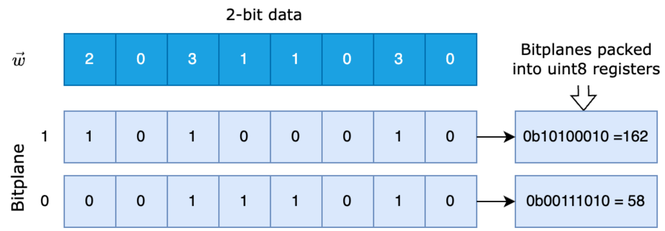


Figure 1. 2-bit data sliced into bitplanes that are then bitpacked into uint8 registers.

Ultra low-bit data can be compactly stored in standard data types such as 8-bit unsigned integers through simple bitwise operations. The data is first sliced into bitplanes based on the number of bits, that are then assigned to standard data types.

Bit-serial Dot Product

Considering binary bitpacked vectors with unipolar (unsigned) encoding where each input value is either 0 or 1, the bit-serial dot product is given by eq. (1a). A bit-wise AND operation gives the element-wise product of the binary inputs and the popcount operation, that counts the number of bits set to 1, performs the accumulation. The binary case can easily be extended to larger bit-widths by slicing the inputs into binary vectors and performing a summation of the bit-serial dot products over all possible bit-sliced combinations as shown in eq. (1b).

$$\vec{w} \cdot \vec{a} = popcount(\vec{w} \ \& \ \vec{a}) \tag{1a}$$

$$\vec{w} \cdot \vec{a} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (popcount(\vec{w_m} \ \& \ \vec{a_n})) << (n+m) \tag{1b}$$

Optimized bit-serial dot product with `dlrt_bitserial_conv2d`

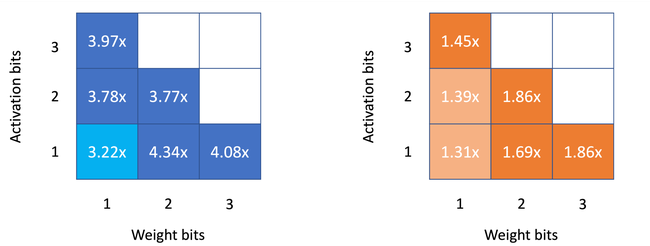
We propose a novel bit-serial computation method in eq. (2) that employs a hybrid unipolar-bipolar scheme with unipolar activations and bipolar (signed) weights.

$$\vec{w} \cdot \vec{a} = \begin{cases} -1 \times \sum_{n=0}^{N-1} (popcount(\vec{w}_{M-1} \ \& \ \vec{a_n})) << (n+m), & \text{if } m = M-1 \\ \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (popcount(\vec{w_m} \ \& \ \vec{a_n})) << (n+m), & \text{otherwise} \end{cases} \tag{2}$$

This bit-serial dot product is the building block of our highly optimized bit-serial convolution operator `dlrt_bitserial_conv2d` that advances the state-of-the-art for ultra low-bit inference on ARM devices.

- **Higher accuracy** due to the hybrid unipolar-bipolar scheme.
- **Similar instruction count** to the unipolar variant from eq. (1b).
- **Significant speedups** of up to 4.34×relative to existing ultra low-bit kernels.
- **Zero mapping for weights** ensuring comptaibility with SOTA uniform quantization techniques such as LSQ.

SOTA Ultra Low-bit Performance



(a) Speedup on second layer of ResNet18 across different bit-widths.

(b) Speedup on ResNet18 model across different bit-widths.

Figure 2. Speedups of `dlrt_bitserial_conv2d` over `nn.bitserial_conv2d` on the Raspberry Pi 4B.

Our bit-serial convolution operator `dlrt_bitserial_conv2d` achieves substantial performance uplifts over the open-source hybrid uniploar-bipolar implementation from TVM's `nn.bitserial_conv2d` operator.

Compiler Passes

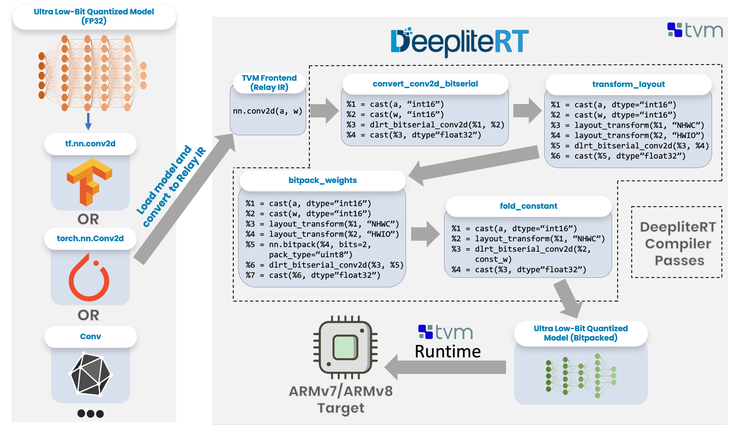


Figure 3. DeepliteRT converts fake-quantized convolution layers from models in different formats to optimized ultra low-bit convolution operators through a series of compiler passes. The passes replace `nn.conv2d` with `dlrt_bitserial_conv2d`, bitpack the weights in ultra low-bit, and cast and transform the layouts of data as required. The resulting compiled model can be deployed on ARMv7 and ARMv8 CPUs via TVM runtime.

End-to-end Performance

Model	Raspberry Pi 4B - 32-bit ARMv7				Raspberry Pi 4B - 64-bit ARMv8			
	FP32	INT8	2A2W	2A2W (Ours)	FP32	INT8	2A2W	2A2W (Ours)
ResNet18	149.29	145.44	130.92	70.32	110.94	91.13	123.28	67.13
ResNet50	433.19	326.49	311.8	196.79	315.03	203.56	295.96	197.91
ResNet101	-	558.47	487.96	325.37	545.01	378.27	471.71	319.09
VGG19	-	1399	1003	654.69	-	922.28	962.65	636.79
InceptionV3	312.82	245.16	357.77	165.05	218.18	151.55	340.82	164.62
DenseNet121	387.98	589.03	296.27	252.65	302.50	261.94	269.91	227.05
VGG16-SSD300	1671	2310	1780	1190	1547	1462	1631	1060
YOLOv5s	219.72	197.27	135.64	100.32	169.93	113.5	130.03	97.49
Average speedup	1.89×	1.91×	1.58×	-	1.54×	1.20×	1.56×	-
Minimum speedup	1.40×	1.49×	1.17×	-	1.32×	0.92×	1.19×	-
Maximum speedup	2.20×	2.33×	2.17×	-	1.71×	1.45×	2.07×	-

Table 2. End-to-end latencies (ms) and speedups of our 2-bit DeepliteRT inference engine over 32-bit TVM (FP32), 8-bit ONNX Runtime (INT8) and 2-bit TVM bit-serial (2A2W) baselines.

DeepliteRT offers leading performance for both ARMv7 and ARMv8 targets. On average, DeepliteRT realizes speedups of 1.89×, 1.91× and 1.58× in 32-bit mode and 1.54×, 1.20× and 1.56× in 64-bit mode over TVM FP32, ONNX Runtime INT8 and TVM 2A2W, respectively.

References

[1] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. *CoRR*, abs/1902.08153, 2019.