

Block Minifloat Arithmetic for Deep Learning Inference and Training

Philip Leong

Director, Computer Engineering Laboratory

<http://phwl.org/talks>



THE UNIVERSITY OF
SYDNEY



Computer Engineering Laboratory

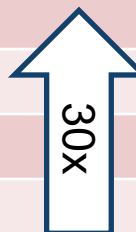
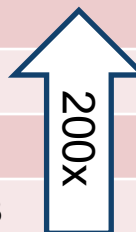
- › Focuses on how to use parallelism to solve demanding problems
 - Novel architectures, applications and design techniques using FPGAs
- › Research: reconfigurable computing, radio frequency machine learning



Tradeoff between performance and precision

- › CPUs/GPUs designed to support datatypes of fixed wordlength
 - Double, float, long, short, char
- › FPGA and ASICs can provide custom datapaths of arbitrary wordlength

Precision	Peak TOPS	On-chip weights
1b	~66	~70 M
8b	~4	~10 M
16b	~1	~5 M
32b	~0.3	~2 M



Slide: Xilinx

- › So how can we utilize low-precision for inference and training?



- › Block Minifloat
- › Time series Prediction
- › Transfer Learning

Block Minifloat

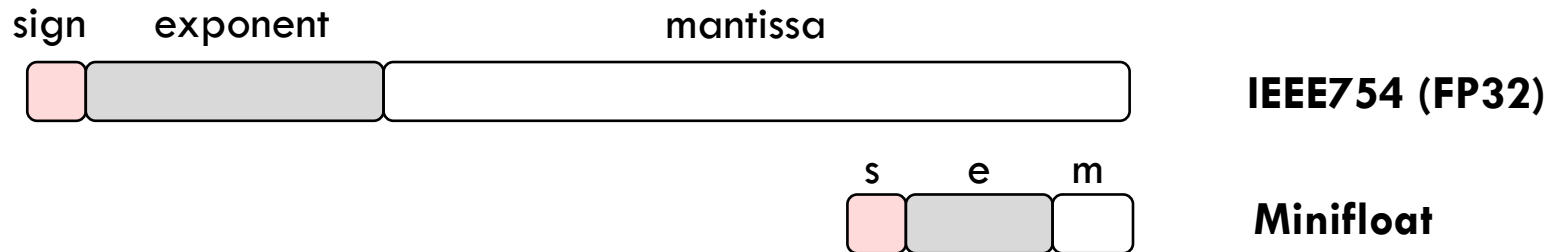
Sean Fox

2



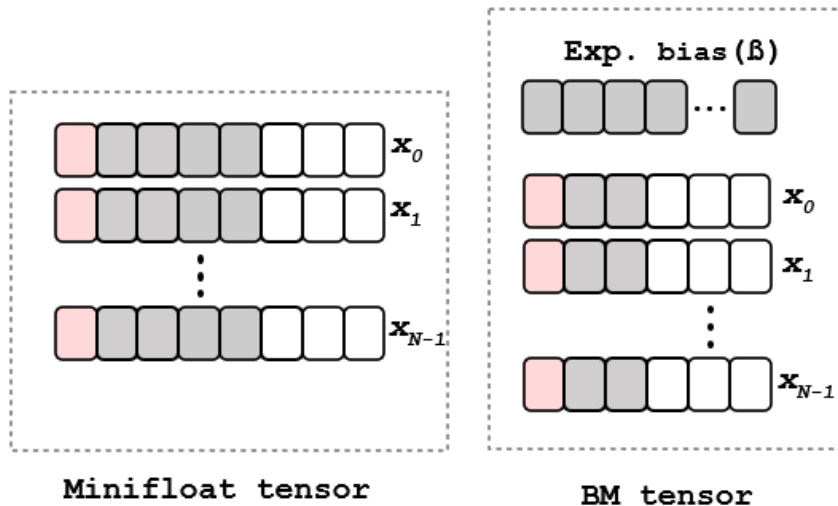
- Training has greater efficiency problem than inference!
 - E.g. 3x more MACs, much higher memory requirements
 - Specialized number representations have been proposed
 - Alternatives to FP32/FP16
 - 4-8 bits for weights, activations and gradients
 - Cheaper and faster training systems
 - Focus on Edge (not sure about the Data Center)
-

- Narrow floating-point representation
 - Our range between 4-8 bits
 - NaN/Infinity NOT supported



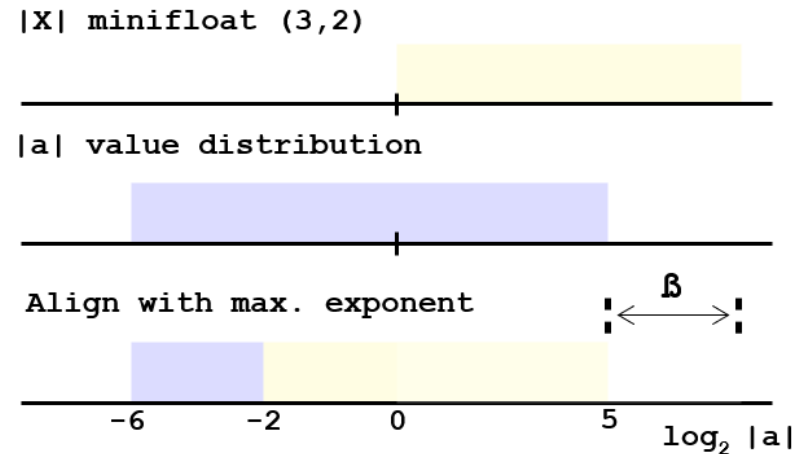
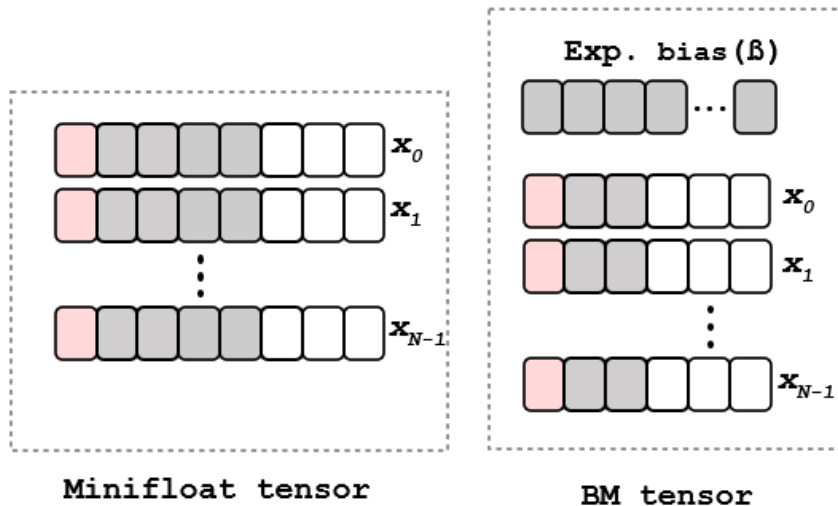
- Pros:
 - Memory (fewer bits)
 - Smaller hardware
- Cons:
 - Dynamic Range (exponent bits)

- Share exponent bias across **blocks** of N minifloat numbers



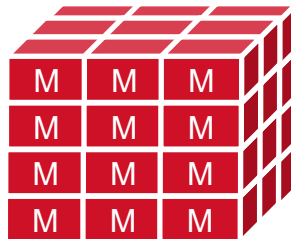
- Dynamic range (with fewer bits)
- Denser dot-products in hardware

- Share exponent bias across **blocks** of N minifloat numbers

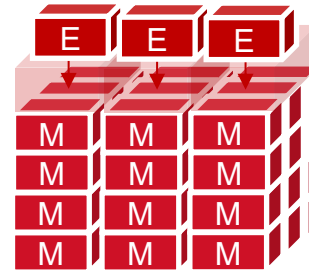


- Dynamic range (with fewer bits)
- Denser dot-products in hardware

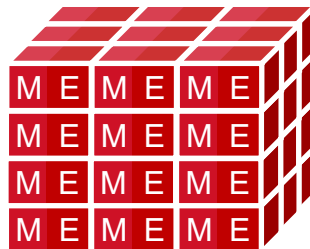
- Align with **max** exponent
- Underflow is tolerated



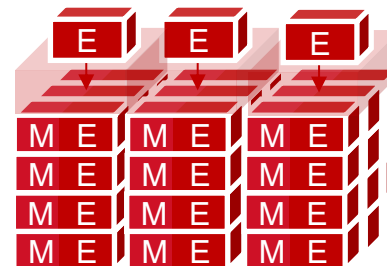
Fixed



BFP



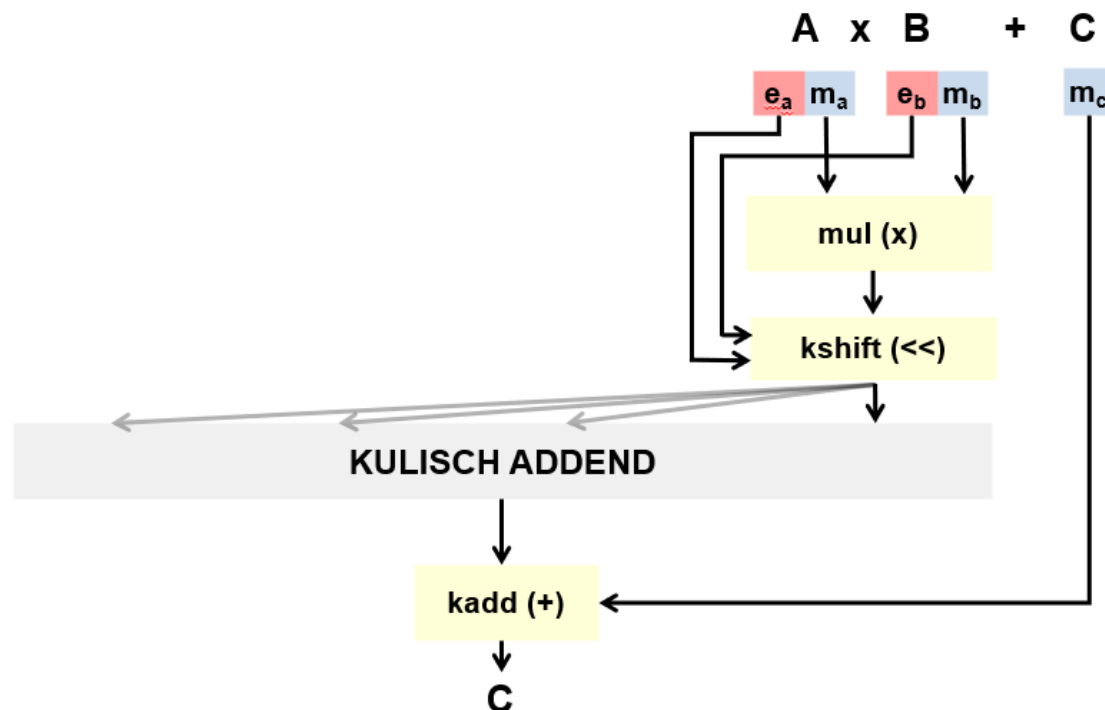
Minifloat



Block Minifloat

Fused Multiply-Add (FMA) with Kulisch Accumulation

- **Kulisch Accumulator:** Fixed point accumulator wide enough to compute error-free sum of floating-point products
- Integer-like hardware complexity for **exponent ≤ 4 bits**

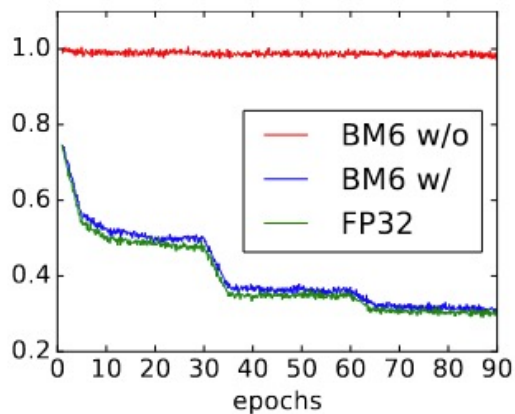


- Three techniques to reduce data loss:
 - Gradual underflow, Block Design, Hybrid Formats
- Simulate specialized BM hardware on GPU (with FP32)
 - Apply Block Minifloat to all weights, acts, grads
- Our Spectrum of Block Minifloats

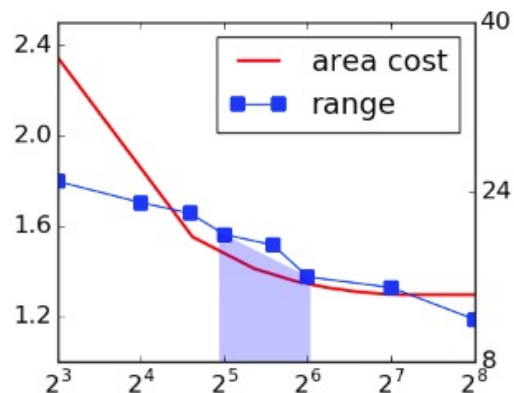
BM8 (ours)	(2,5)/(4,3)
BM7 (ours)	(2,4)/(4,2)
BM6 (ours)	(2,3)/(3,2)
BM5 (ours)	(2,2)/(3,1)
BM5-log (ours)	(4,0)/(4,0)
BM4 (ours)	(2,1)/(3,0)
BM4-log (ours)	(3,0)/(3,0)



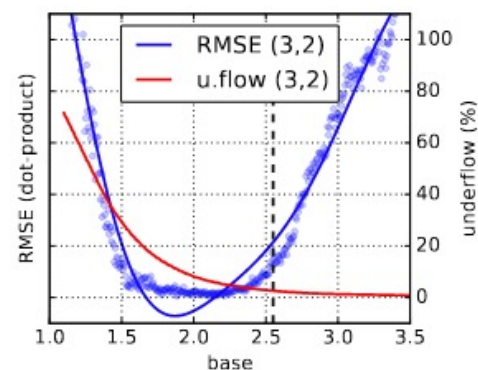
Data Loss Experiments



(a) Validation Accuracy: Training with denormal numbers on ImageNet

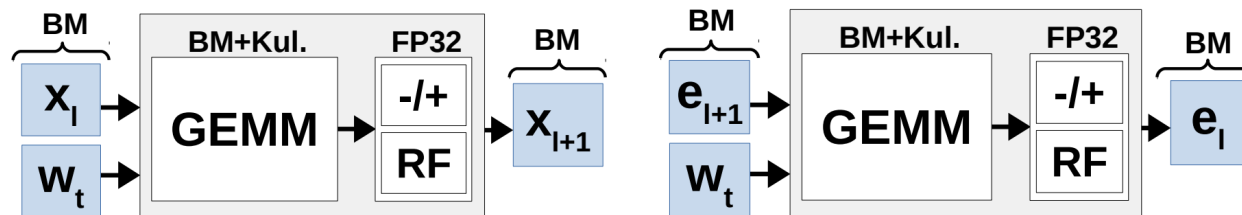


(b) HW (left axis) vs Range (right axis): Selecting the block size



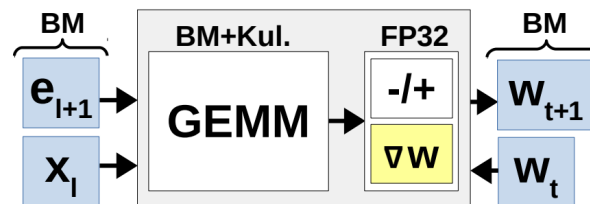
(c) Minifloat scaling by varying the exponent base

End-to-end GPU Training with BM



(a) Fwd activation

(b) Bwd activation grad.

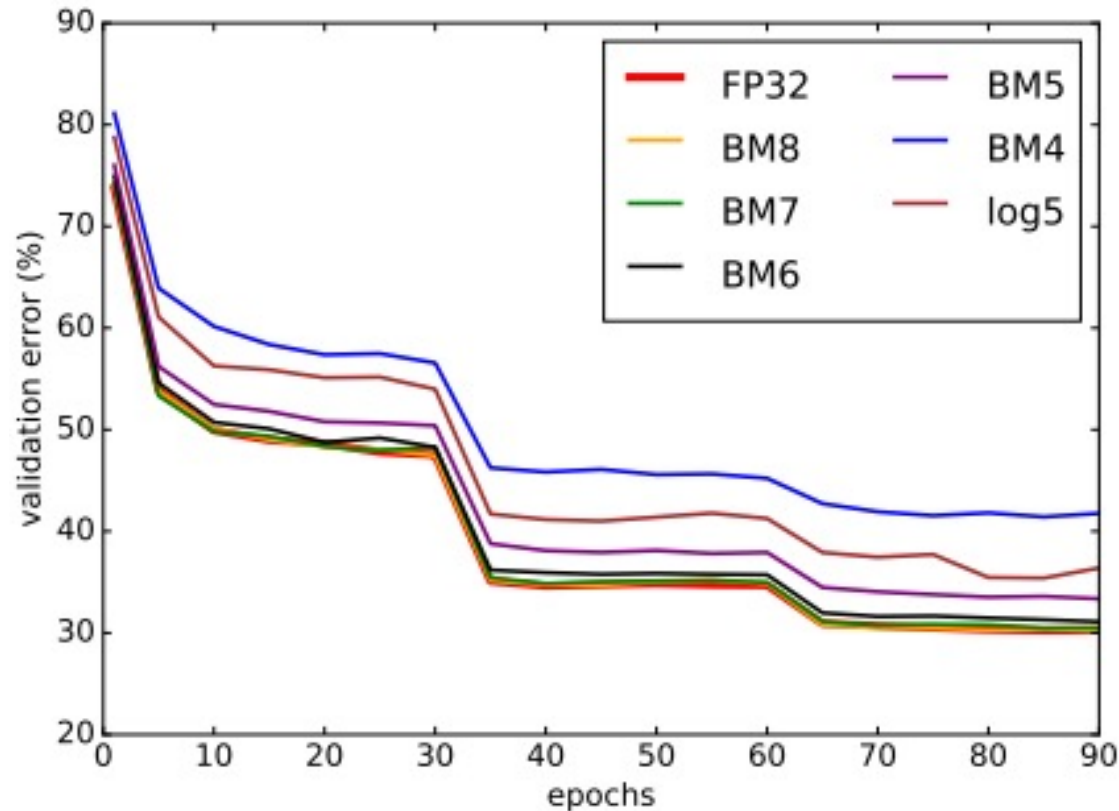


(c) Bwd weight grad. and update

- Weight, activation and gradient tensors quantized to BM with stochastic rounding
- Kulisch accumulator ensures our dot products are exact (can use FP CUDA lib directly)
- FP32 used for Kulisch to floating-point conversion, block minifloat alignments, quantization etc.
- Approx 1x floating point operation every N MACs, 5x slowdown



Training Experiments (1)

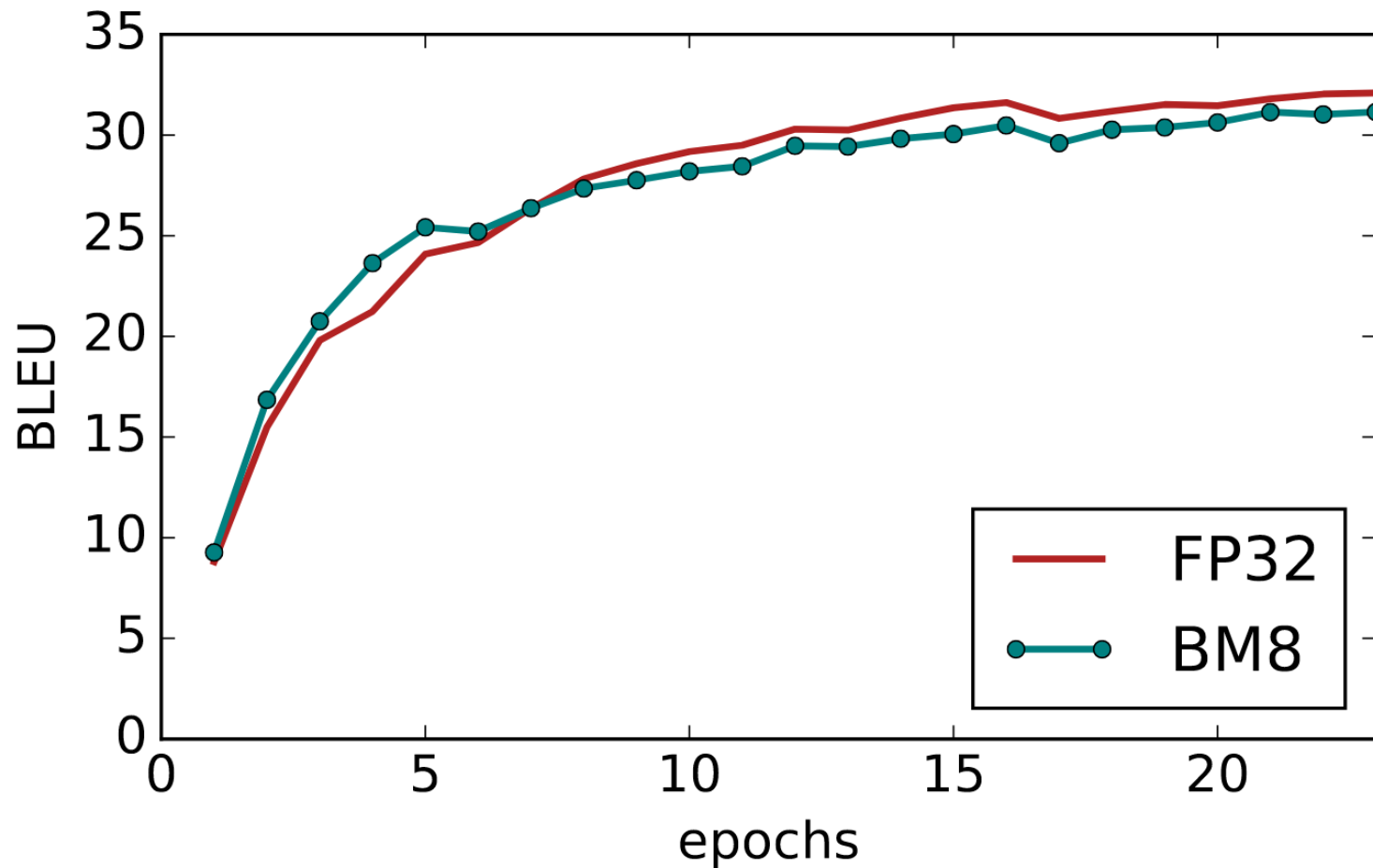


Scheme	BFP (ours)	BM (ours)	∇
6-bit	67.0	69.0	+2.0
8-bit	69.2	69.8	+0.6

ResNet18 Validation



Training Experiments (2)



Transformer on IWSLT'14 DE-En dataset

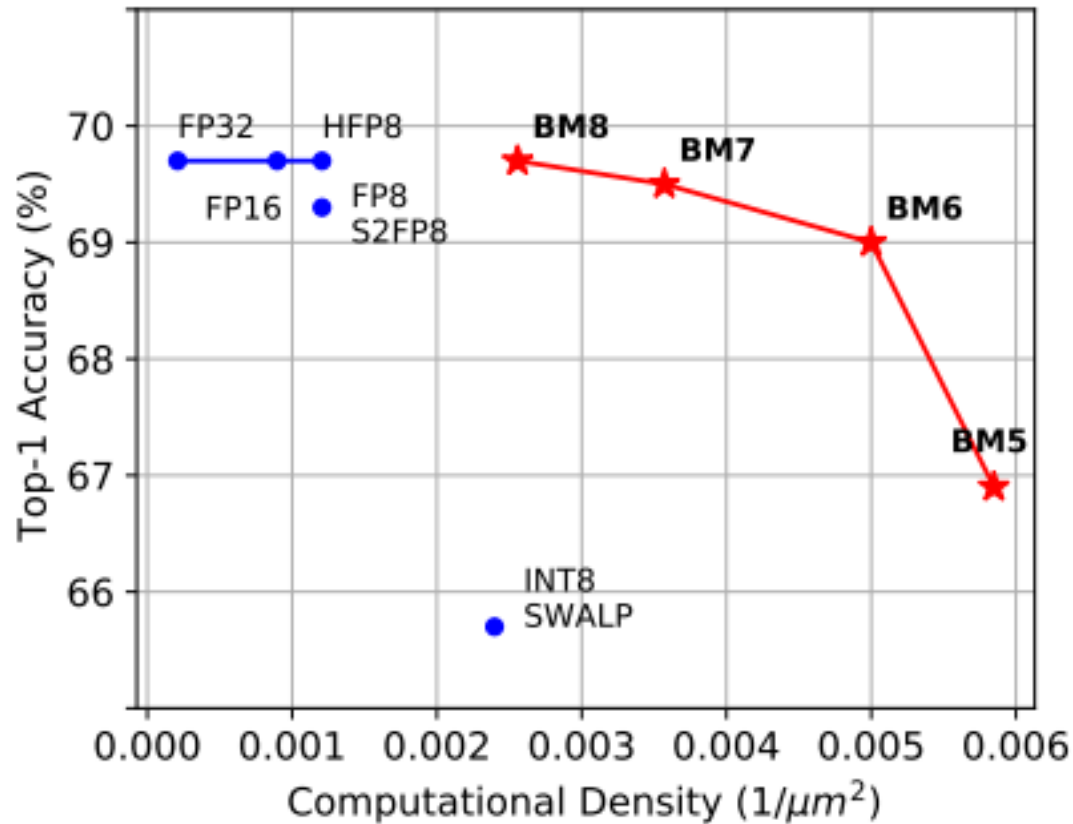
Model (Dataset) [Metric]	FP32	BM8
AlexNet (ImageNet)	56.0	56.2
EfficientNet-b0 (small ImageNet)	62.6	61.8
LSTM (PTB)[Val ppl.]	84.7	87.33
Transformer-base (IWSLT)[BLEU]	32.3	31.8
SSD-Lite (MbNetV2) (VOC)[mAP]	68.6	68.0

**Training Accuracy
with BM \approx FP32**

- Designs synthesized at 750MHz with Cadence RTL Compiler and 28nm cell library
 - Fused multiply-add (FMA)
 - 4x4 systolic matrix multipliers

Component	Area (μm^2)	Power (μW)
FP32	4782	10051
FP8 (w/ FP16 add)	829	1429
INT8 (w/ INT32 add)	417	1269
BM8	391	1141
BM6	200	624
INT8 (4x4 systolic)	7005	20253
FP8 (4x4 systolic)	18201	56202
BM8 (4x4 systolic)	6976	18765

**BM8 area and
power comparable
to INT8**



BM units are:

- **Smaller**
- **Consume less Power**

Model: ResNet-18
Dataset: ImageNet

Time Series Prediction

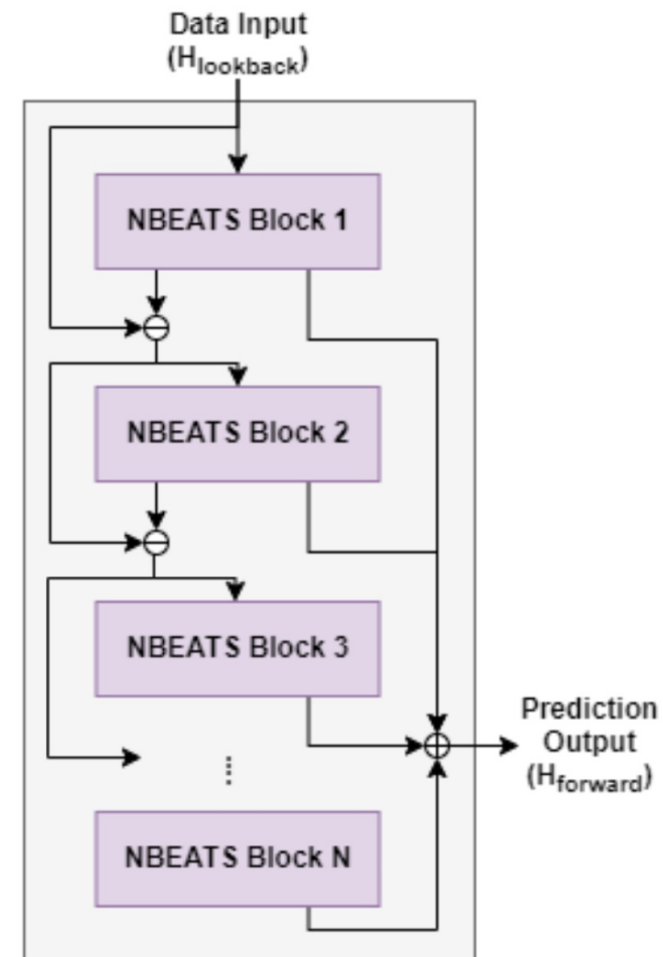
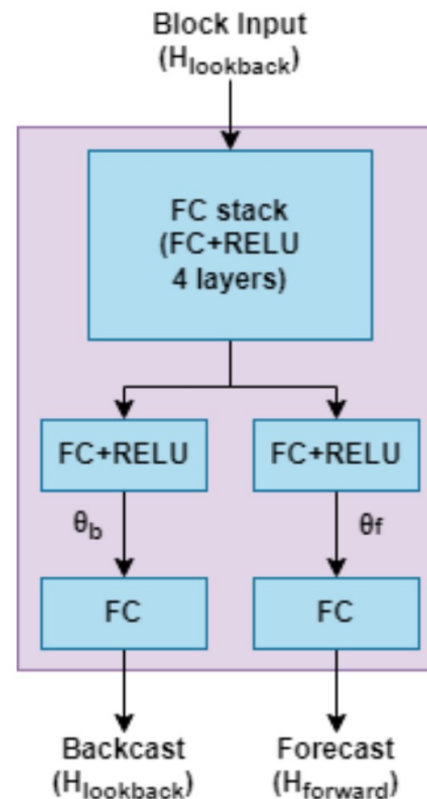
Wenjie Zhou

2



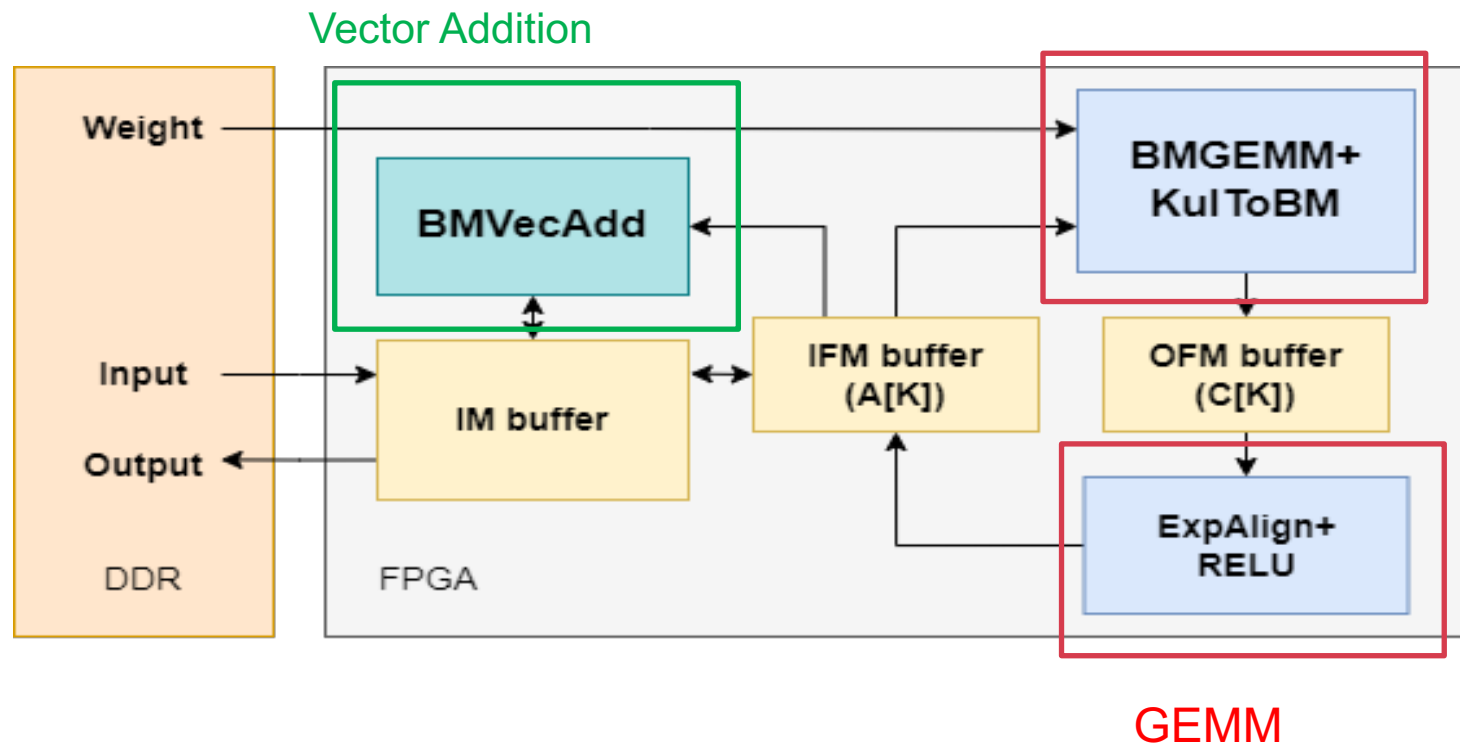
- › Previous work used GPU implementations with 28nm ASIC study
- › Here we explore FPGA implementation
 - NBEATS Inference and Training implementation using 4-bit mixed-precision BM
 - BM GEMM array and Training accelerator architecture for NBEATS

- › N-beats: Neural basis expansion analysis for interpretable time series forecasting. ICLR, 2019
- › Achieves state of the art time series prediction results
- › NN comprises mainly FC layers with shortcut connections



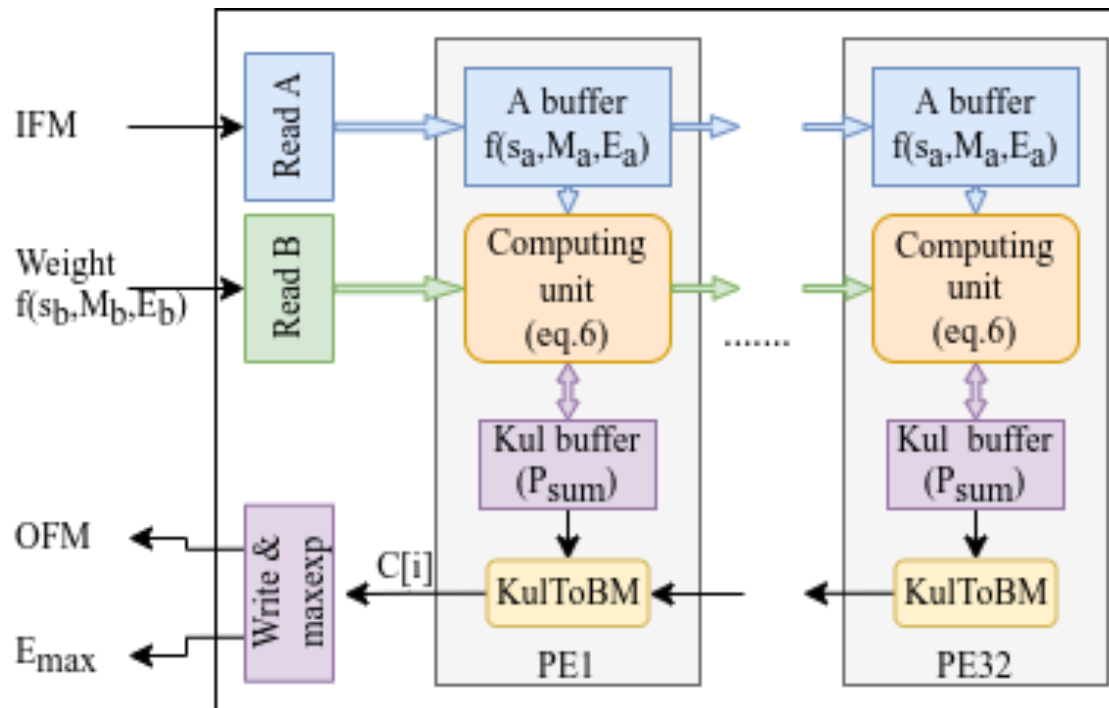


Inference Accelerator Architecture



GEMM Systolic Architecture

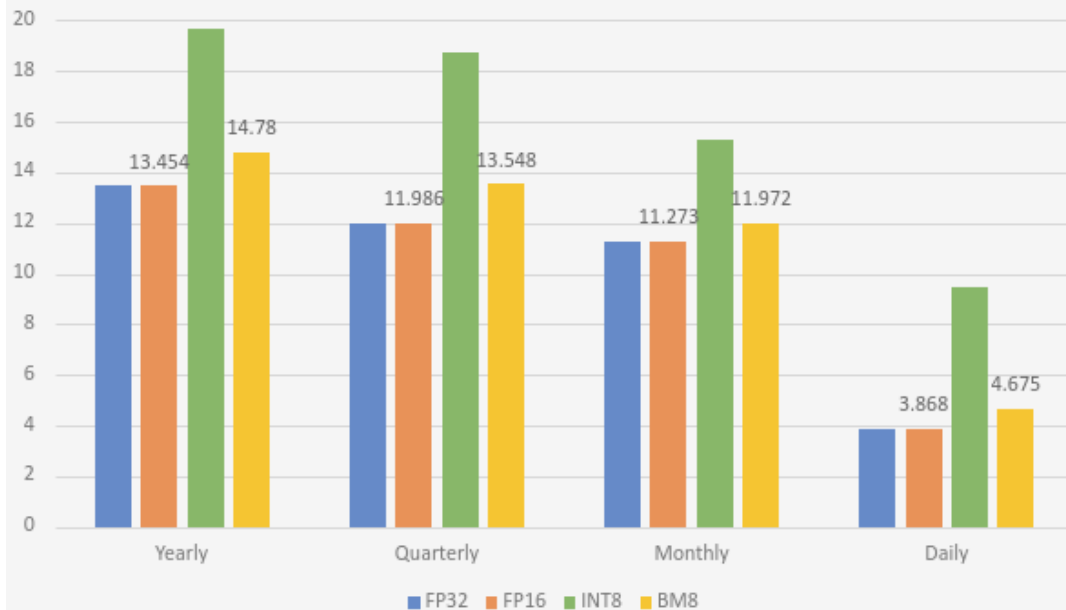
- › Each PE performs multiplication and Kulisch accumulation
- › Intermediate results are stored in the Kul buffer
- › Result transformed to a BM format





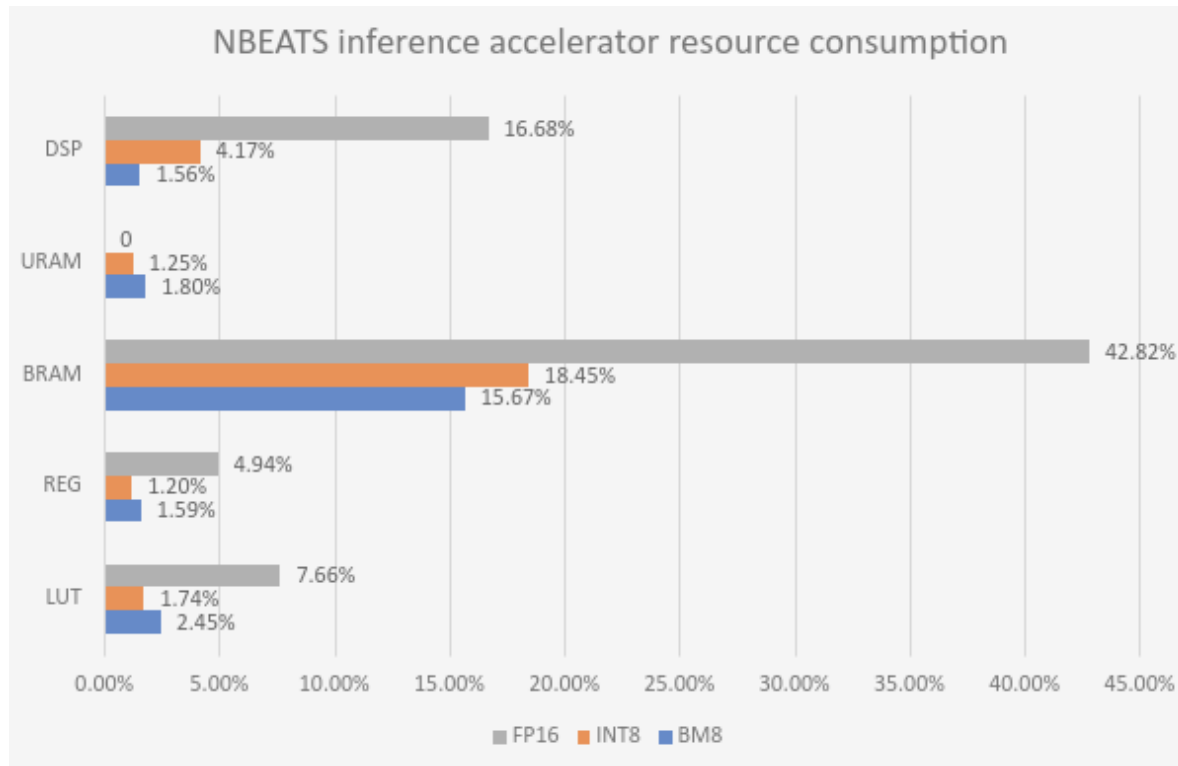
Accuracy (Since Improved)

NEBATS accuracy(SMAPE Loss) comparison



Benchmark	M4 dataset
Dataset	Yearly, Quarterly, Monthly, Daily
Loss	symmetric mean absolute percentage error (sMAPE)

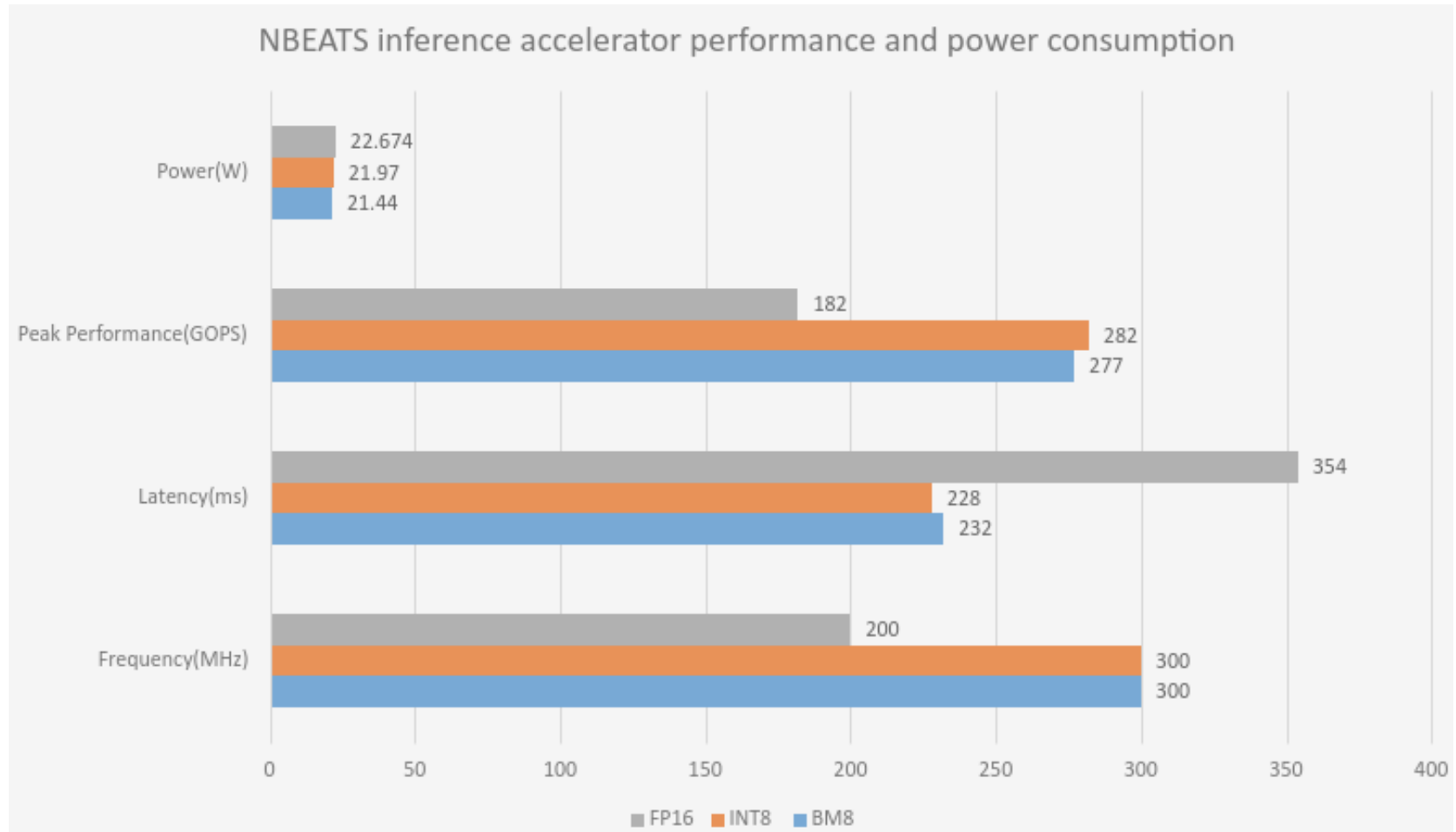
Accuracy of BM8 is similar to FP32



Area of BM8 is similar to INT8 but smaller than FP16



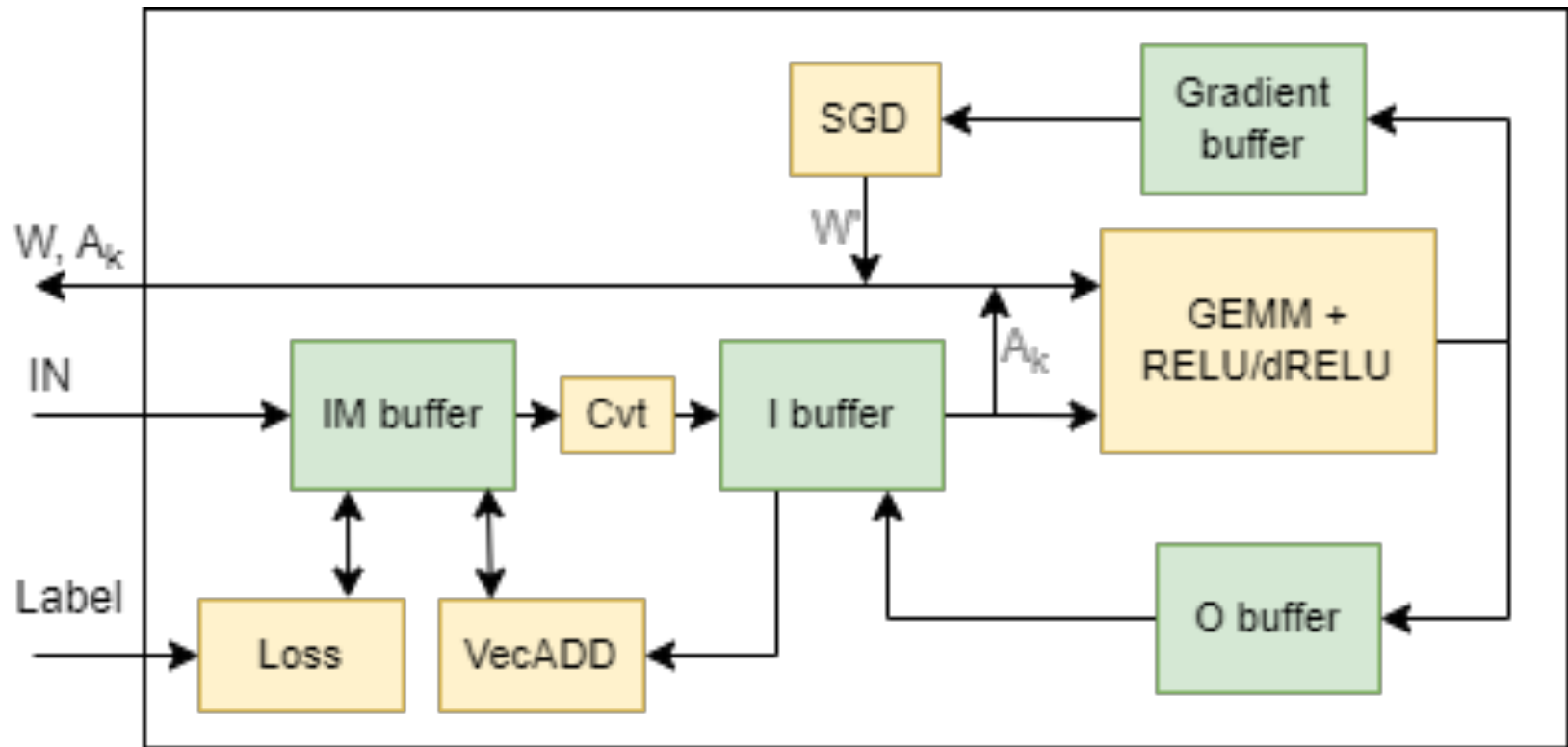
Inference Performance



BM8 performance and power is close to INT8

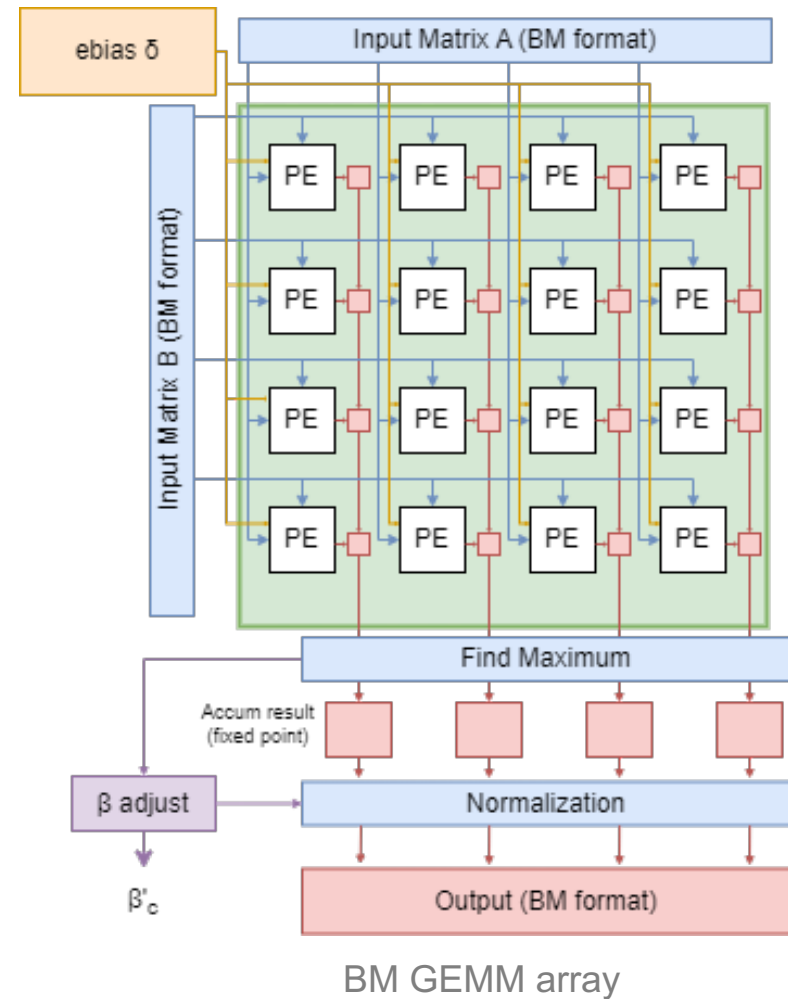
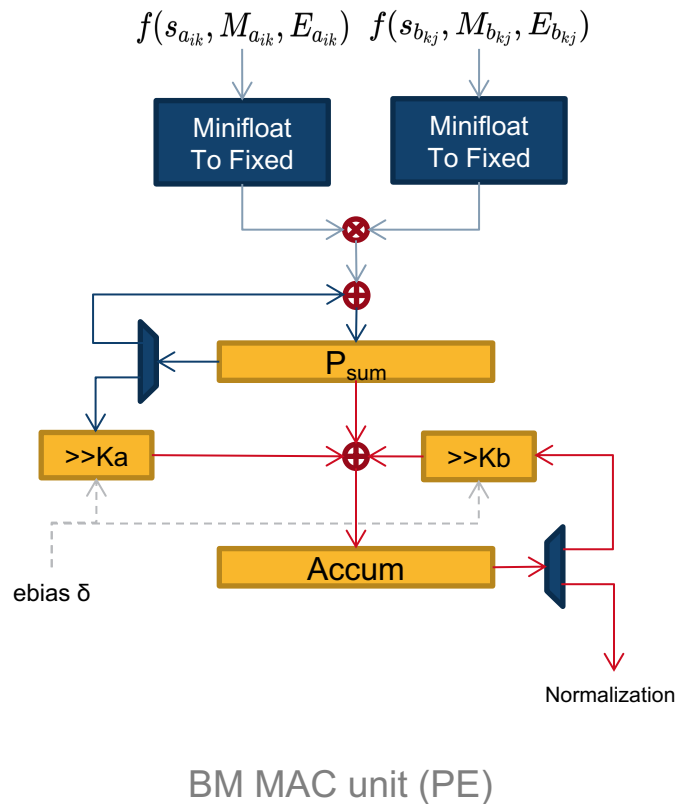


NBEATS Training Accelerator Architecture





Mixed-precision Block Minifloat Training



- › Dataset: M4-Yearly, validation loss: SMAPE loss, block size: 64

	Loss	Configuration			
		weight	activation	error	gradient
BM4(1)	14.471649	BM<2,1>	unsigned BM<0,4>	BM<0,3>	BM<0,3>
BM4(2)	14.463654	BM<2,1>	unsigned BM<0,4>	BM<0,3>	FP32
BFP8	12.914178	BM<0,7>	BM<0,7>	BM<0,7>	BM<0,7>
BM8	12.939716	BM<2,5>	BM<2,5>	BM<0,7>	BM<0,7>
FP32	12.924581				

Transfer Learning

Chuliang Guo



- › Re-training light-weight CNNs at edge for various applications
 - Private and secure
 - No personal information uploaded to clouds
 - Low-latency
 - Locally training at edge without sending to cloud servers
 - Reconfigurable and customised logic resources tailored for low-precision
 - Energy efficient
 - Faster than CPUs- e.g., 10X throughput/latency performance
 - More energy efficient than GPUs- e.g., 1/10 power consumption
-

› Back-propagation using SGD

– 3X workload of inference

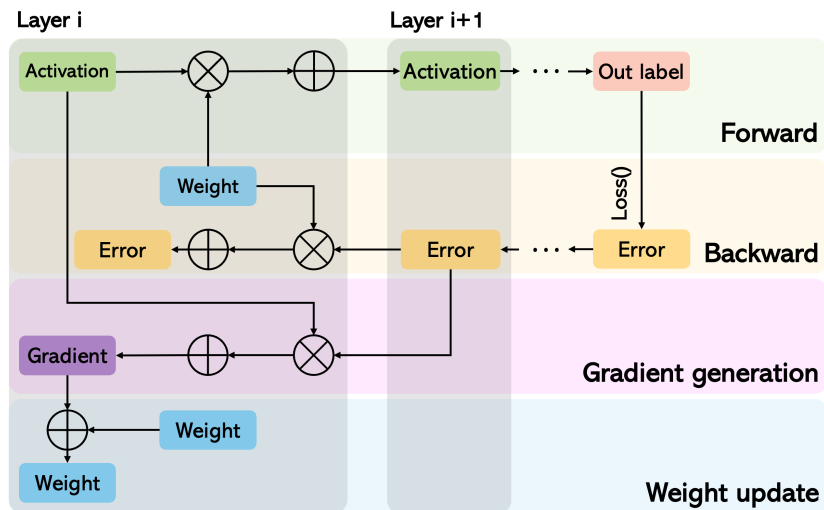
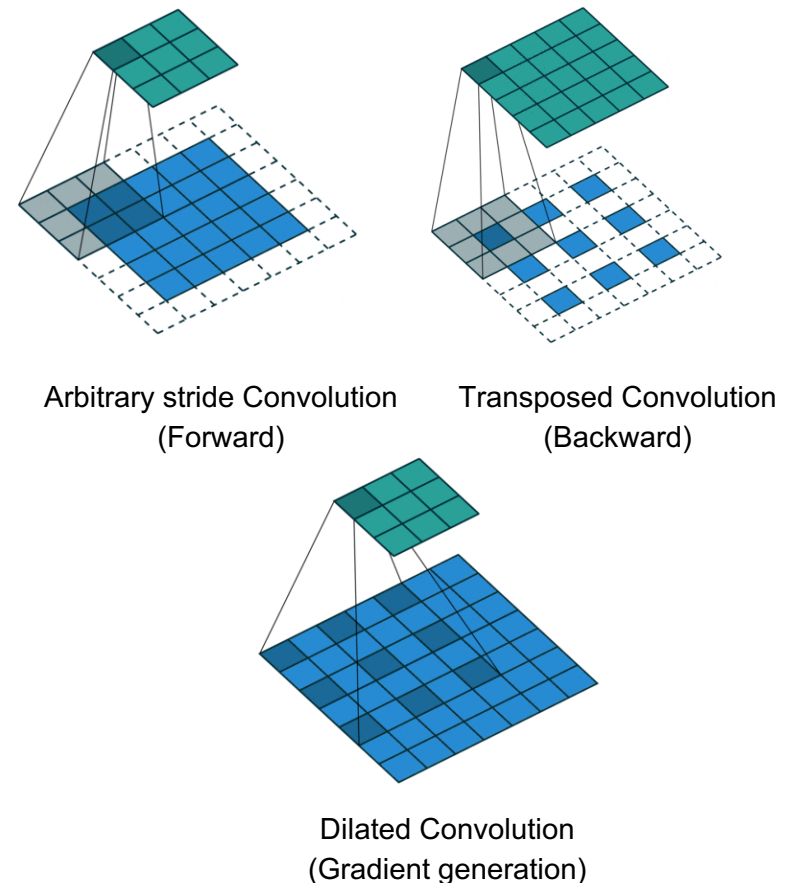


Fig. 1 CNN training workflow: (1) Conv in forward path, (2) transposed Conv in backward path, (3) dilated Conv in gradient generation, and (4) weight update.



ResNet20/VGG-like accelerator

› Layer-wise CNN blocks

- Unified $\text{bm}(2,5)$ representation
- Non-unit stride Conv support
- Simplified mult/add/MAC
- Fused BN&ReLU

› Main blocks

- Unified Conv
- Conv & transposed Conv
- Dilated Conv
- Weight kernel partition

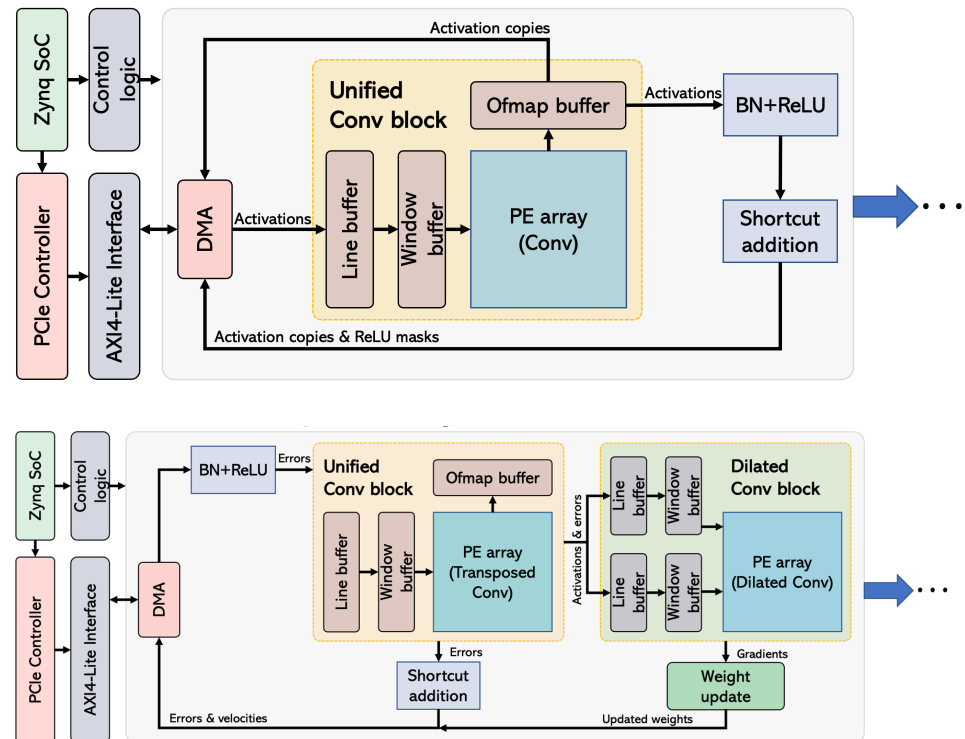


Fig. 3 Overall architecture of the generic training accelerator for layer-by-layer processing. BN and ReLU are fused.

CIFAR10 Training from Scratch

- › Shortcut addition after BN and ReLU functions (enabling fusing)
- › Unified $\text{bm}(2,5)$ for activations, weights, errors, and gradients (simpler HW)
- › Full precision accuracy with these changes

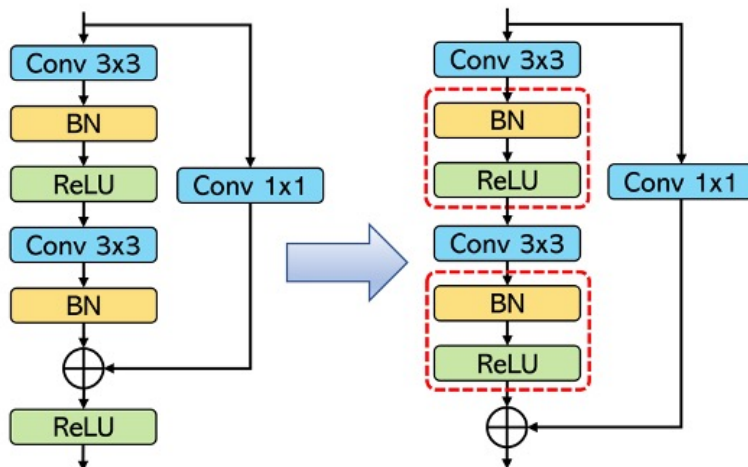


Fig.3. ResNet20 building block.

Tab. 1. Top-1 accuracy on CIFAR-10.

Model	Precision (FP/BP)	CIFAR-10 Acc %
ResNet20	FP32	91.11%
	BFP8 (i.e., $\text{bm}(0,7)$)	89.72%
	$\text{bm}(2,5)/\text{bm}(4,3)$	90.54%
	$\text{bm}(2,5)$	90.44%
Modified ResNet20	FP32	91.04%
	BFP8 (i.e., $\text{bm}(0,7)$)	89.95%
	$\text{bm}(2,5)/\text{bm}(4,3)$	90.72%
	$\text{bm}(2,5)$	90.61%

Transfer learning application

– Channel tiling accelerator

– Updating last several Conv & FC

- Shortened back-propagation
- Reduced BRAM for activations
- Faster convergence

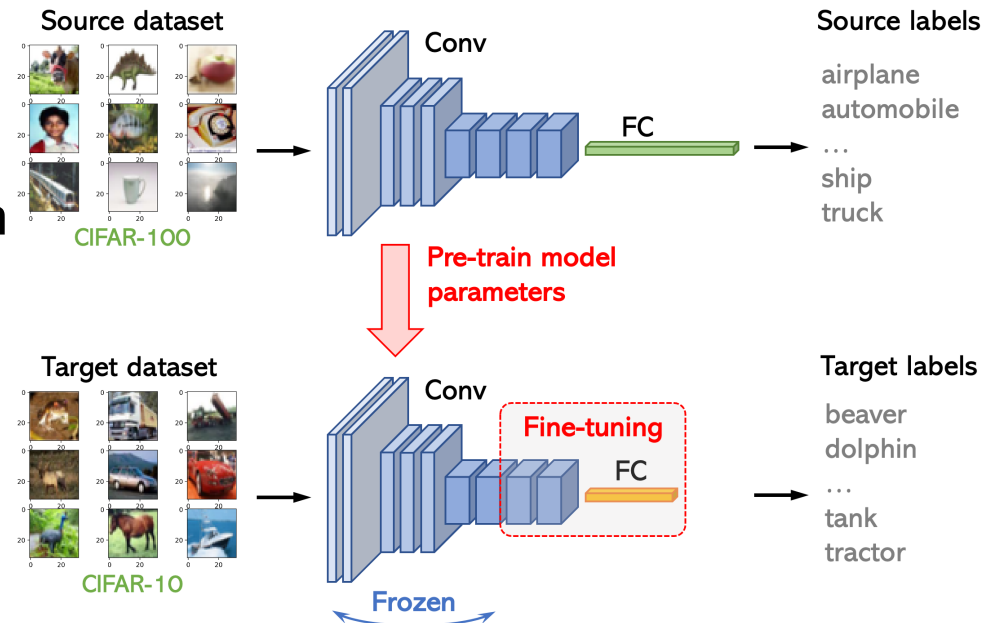
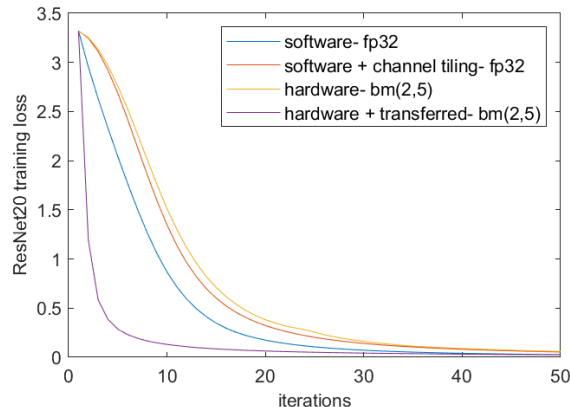


Fig. 8 Transfer learning example from CIFAR-100 to CIFAR-10.



TABLE III
RESOURCE UTILISATION OF AND POWER THE RESNET20 ACCELERATOR
(WITH THE STATIC POWER OF 30W).

	CLB	LUT	DSP	BRAM	Vivado(W)	PPS(W)
Full update	28824	166502	686	1171	8.714	35
6 Conv+FC	25589	161129	685	671	7.725	34
2 Conv+FC	21340	129453	621	571	6.779	34

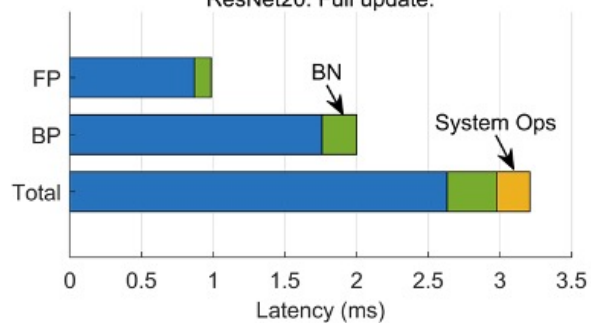
TABLE IV
RESOURCE UTILISATION AND POWER OF THE VGG-LIKE ACCELERATOR
(WITH THE STATIC POWER OF 30W).

	CLB	LUT	DSP	BRAM	Vivado(W)	PPS(W)
Full update	20688	119086	614	505	6.824	34
3 Conv+FC	20489	119740	613	325	6.499	34

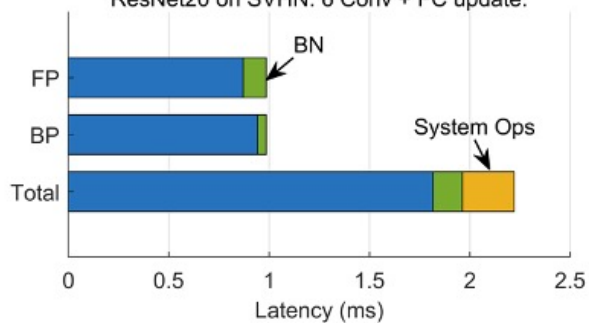


Latency Breakdown

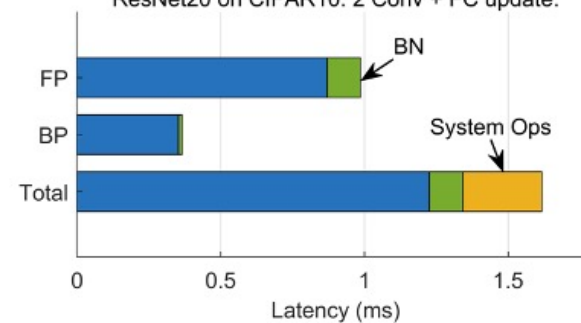
ResNet20: Full update.



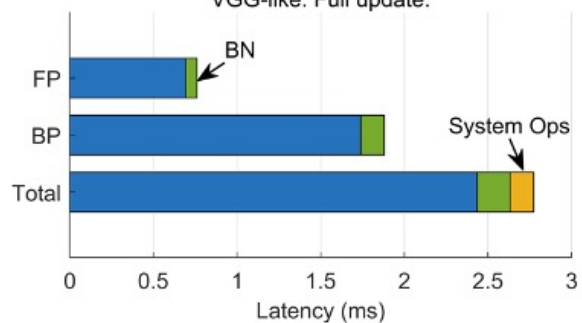
ResNet20 on SVHN: 6 Conv + FC update.



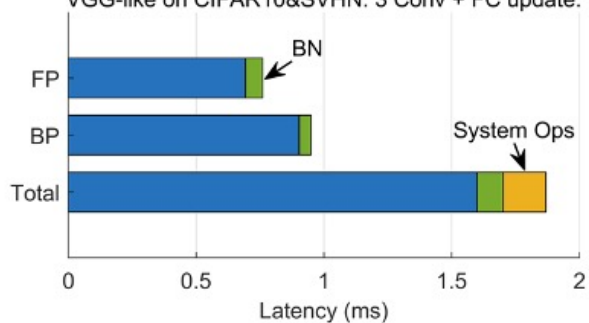
ResNet20 on CIFAR10: 2 Conv + FC update.



VGG-like: Full update.



VGG-like on CIFAR10&SVHN: 3 Conv + FC update.



Conclusion

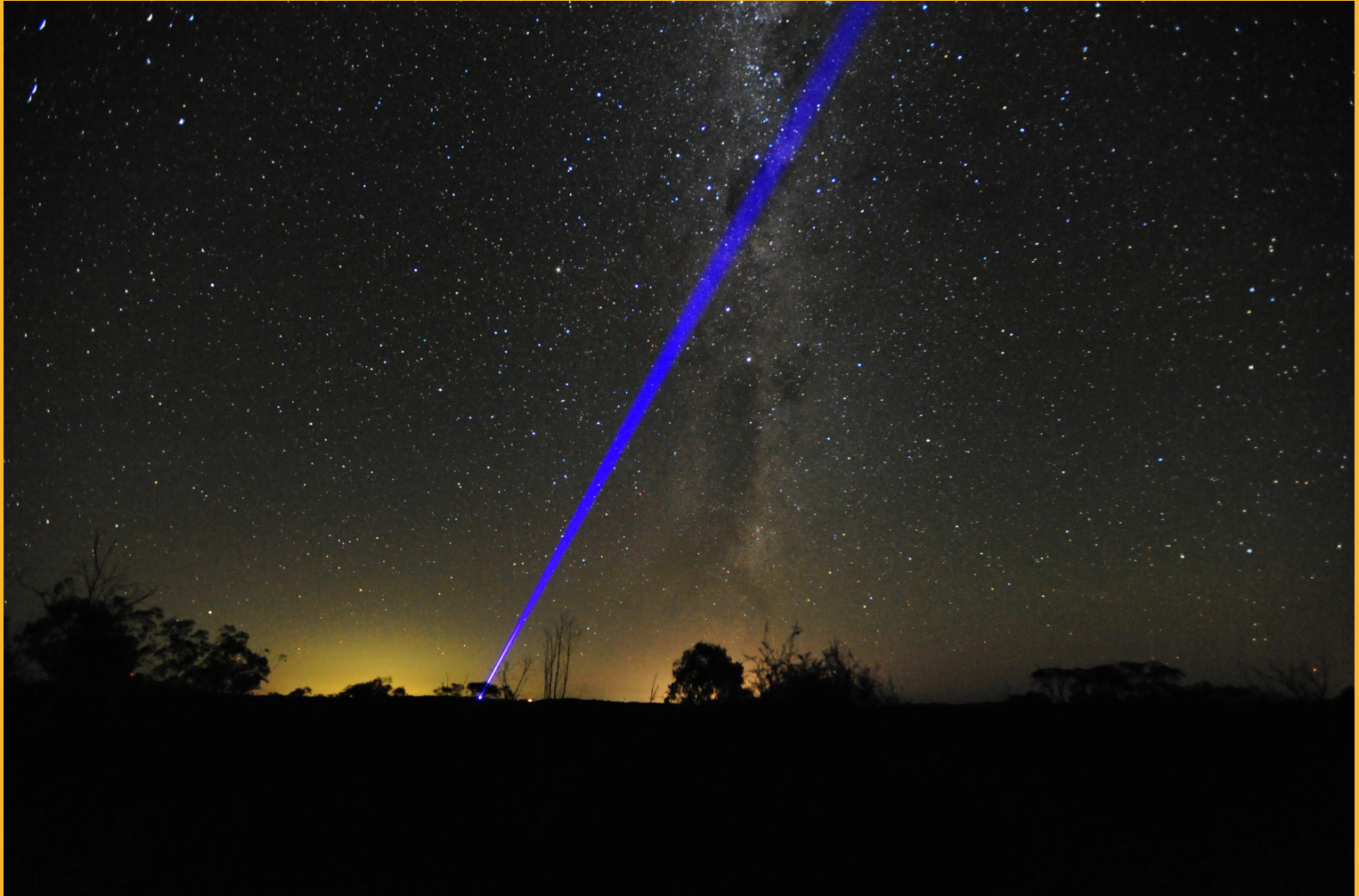
2



- Low-precision formats have wide applicability for inference and training in Edge applications
 - Doesn't necessitate accuracy reduction
- Faster Training is possible using BM
 - Fewer bits – important for memory-bound
 - Narrow exponents – denser MAC in compute-bound

What are the applications?

- [1] Sean Fox, Seyedramin Rasoulinezhad, Julian Faraone, and David Boland Philip H.W. Leong. A block minifloat representation for training deep neural networks. In *Proc. of The International Conference on Learning Representations (ICLR)*. 2021. URL: [bm_iclr21.pdf](#).
- [2] Wenjie Zhou, Haoyan Qi, David Boland, and Philip H.W. Leong. FPGA implementation of N-BEATS for time series forecasting using block minifloat arithmetic. In *Proc. Asia Pacific Conference on Circuits and Systems (IEEE APCCAS 2022)*. 2022. URL: [nbeats_apccas22.pdf](#).



Philip Leong (philip.leong@sydney.edu.au)
<http://phwl.org/talks>