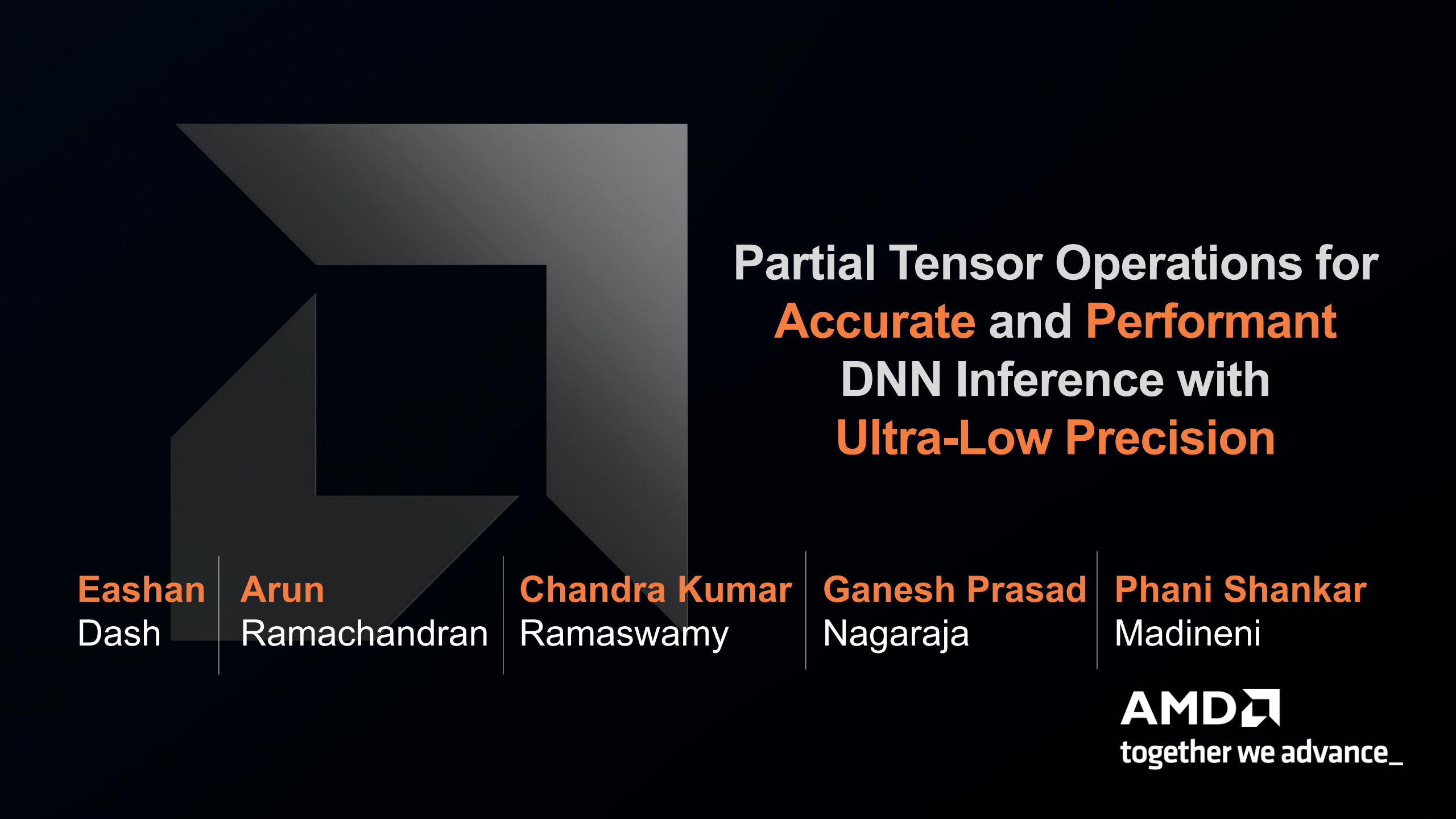


Partial Tensor Operations for Accurate and Performant DNN Inference with Ultra-Low Precision

Eashan Dash | Arun Ramachandran | Chandra Kumar Ramaswamy | Ganesh Prasad Nagaraja | Phani Shankar Madineni

AMD

together we advance_

Short Bio Speakers

- **Eashan Dash – Staff Machine Learning Engineer at AMD**
  - Has 5+ years of professional experience in and around AI, DNNs (Deep Neural Networks), and HPC (High Performance Compute) domains. His primary research areas include DNN acceleration and optimizations at library and framework level, Large language model optimization, Numerical linear algebra algorithms and optimizations, BLAS, Low precision GEMM, SIMD AI kernel algorithms etc.

- **Arun Coimbatore Ramachandran – Principal Staff Machine Learning Engineer at AMD**
  - Has 17+ years of industry experience. He is currently pursuing PhD in IISC Bangalore with interest in intersection of Machine Learning and Systems.

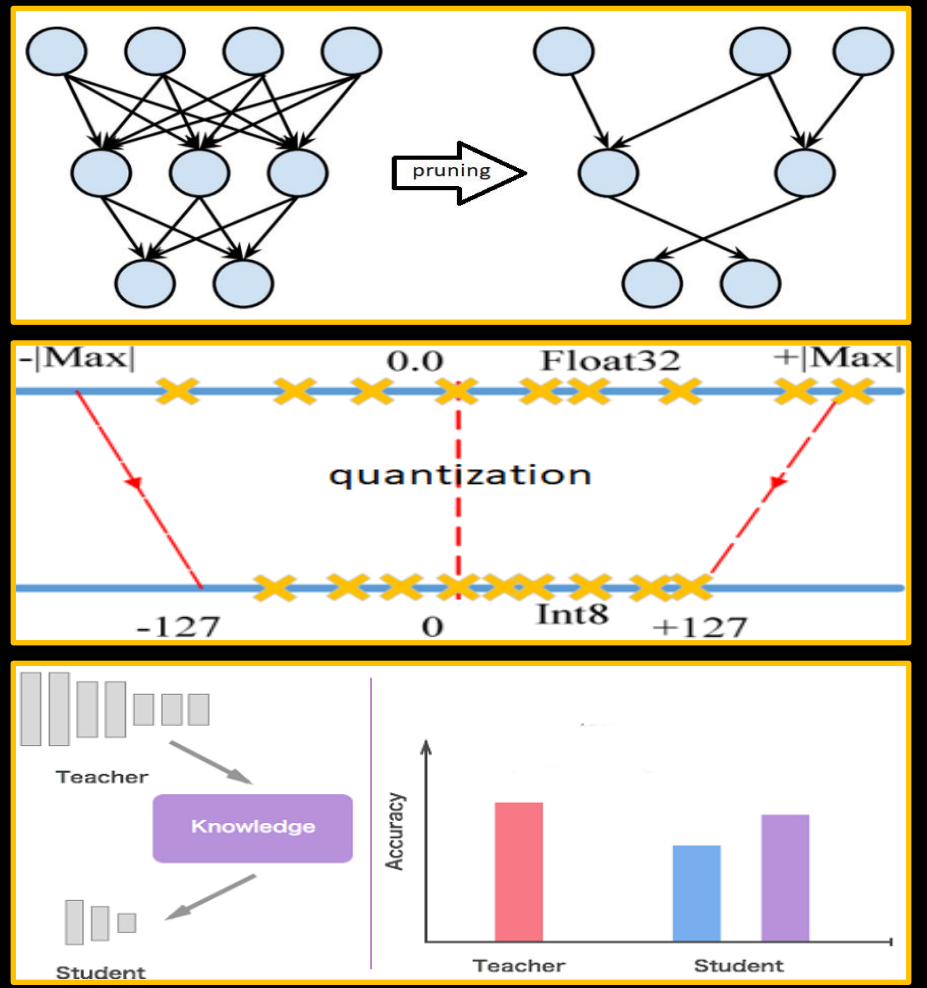
Introduction (1/1)

DNN ACCELERATION – Software Solutions And Processor Trends

- Accelerating Deep Neural Networks (DNNs) for inference/training has become inevitable for real time deployment

- Software solutions focus on neural network compression techniques for network acceleration
  - Reduces model size and increases computational efficiency
  - Pruning, Quantization, Knowledge distillation

- **Future of Computing**: Heterogenous architectures combining the CPU and accelerators with unified memory access
  - For example, AMD MI300 APUs – CPUs + GPUs
  - Future generation platforms – CPUs + DPUs
    - DPU – Deep Learning Processing Unit
  - Accelerators support ultra low precision compute like INT4



Problem Statement (1/1)

Need for Software Solution to utilize AMD's Heterogenous Architectures

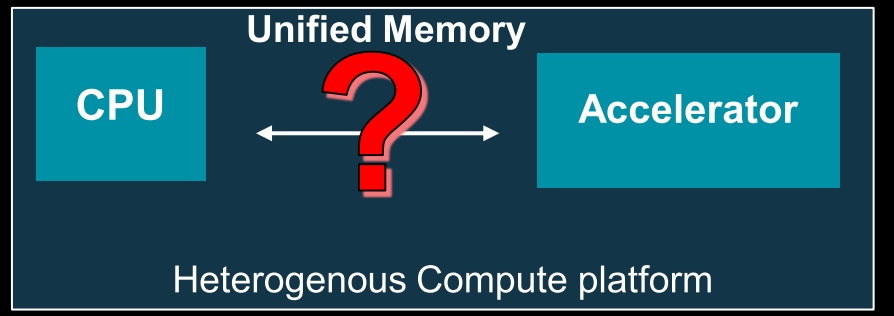
- Several challenges with state-of-the-art solutions implementing neural network compression techniques
  - Major challenge is to retain the accuracy while increasing the speed

- Quantization – Promising technique for DNNs compression
  - Uses lower-precision numerical formats (FP16, BF16, INT8, INT4) for weights and activations
  - **Accuracy gets a hit during quantization due to loss of information**
    - Significant accuracy degradations with INT-4, INT-8
  - **Retraining/Fine-tuning are Data-driven and Iterative**
    - Requires training data, compute resources and time consuming!

- **No state-of-the-art solution addressing these challenges with heterogenous computing -**
  - To utilize future generation heterogenous AI platforms – CPUs + DPUs/GPUs
  - To improve accuracy and end to end performance with heterogenous computations involving low precisions like INT-4, INT-8
  - To improve accuracy with Data-Free and Non-Iterative methods
    - Modern accelerators support INT4 inference

$Y_{fp32} = X_{fp32} \pm Z_{fp32}$

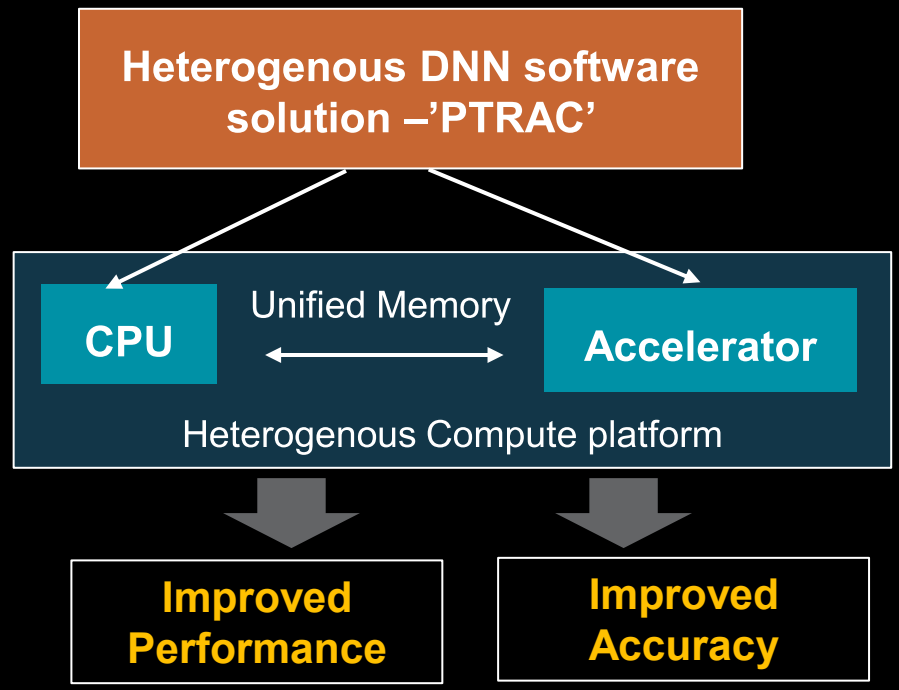
Because of quantization noise



Proposed Solution (1/8)

'PTRAC' – A Novel Heterogenous Solution Targeting Next Gen AI Architectures

- Partial Tensor Retention And Correction
- We propose a heterogenous software solution 'PTRAC' for addressing state-of-the-art challenges with Quantization neural network compression technique
  - Reduces quantization noise
  - Improves accuracy, CPU utilization and end-to-end performance
- PTRAC includes following three novel ideas:
  - **Partial Tensor Correction (PTC)**
    - Novel weight correction scheme, applied to partial weight tensor.
    - Improves accuracy with low precisions like INT-4 and INT-8
  - **Partial Tensor Retention (PTR)**
    - Heterogenous DNN compute method to improve accuracy and performance
    - Part of weight tensor is retained for CPU operations, rest for accelerator
  - **Partial Tensor Retention And Correction (PTRAC): PTC + PTR**.
    - Achieves superior accuracy and performance with low precision heterogenous compute



Proposed Solution (2/8)

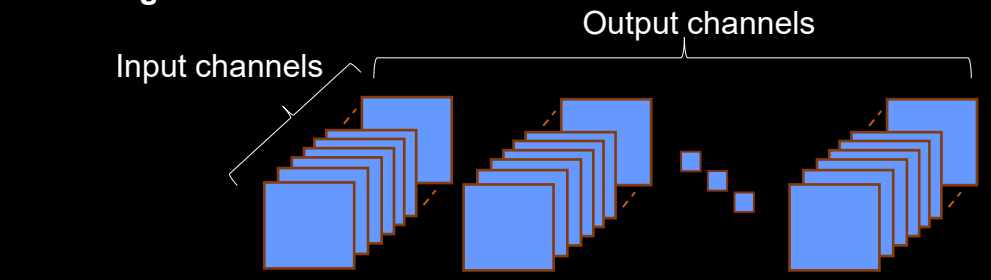
'PTC' – A Novel Weight Correction Method

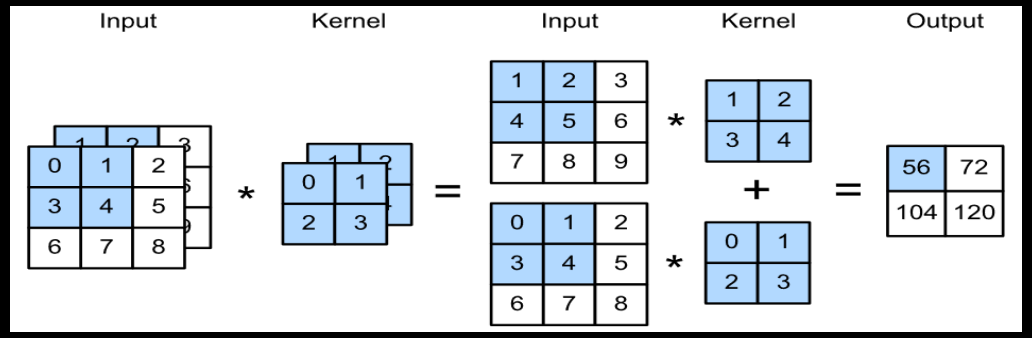
- Quantization of FP32 weights to ultra-low precision (INT-8, INT-4) comes with **heavy accuracy degradation**
- Existing weight correction schemes -
  - **Are Data-driven and Iterative**
    - Retraining procedures to regain accuracy
    - Requires training data, compute resources and time consuming!
  - **Correct Complete Weight Tensors**
    - Complete weight tensor retraining and fine-tuning is time consuming
  - Apply weight corrections at Output channel levels

- Proposed **Partial Tensor Correction** (PTC) Method
  - Novel weight correction scheme, corrects partial weight tensor
  - Per-Input Channel level, Data Free, Non-Iterative
  - PTC method scales across shallower, deeper and wider models

- **Per-Input Channel level**
  - Multiply Accumulate (MAC) operation - input channel levels
  - PTC corrects error induced by MAC operation at input channels
  - PTC minimizes the output error defined in (Eq 1)

CNN Weight Tensor: 4 dimension





MAC computation happens at input channel levels

Traditional correction schemes minimizes: **MSE(W) = E[(w – W)²]**
(W and w denote the dequantized and original weight)

Output Error

Original = $\sum_{i=1}^{N} x_i w_i$

Quantized = $\sum_{i=1}^{N} X_i W_i$

Output Error = $\sum_{i=1}^{N} x_i w_i - \sum_{i=1}^{N} X_i W_i$ *(Eq 1)*

Proposed Solution (3/8)

'PTC' – A Novel Weight Correction Method

- **Data-Free Quantization**
  - Our experiments show that – Quantization rounding errors of activations < Rounding errors of weights
  - Leveraging this we approximate the output error from (Eq 1) as:

    Output Error = $\sum_{i=1}^{N} w_i$ - $\sum_{i=1}^{N} W_i$ (Eq 2)
- **Non-Iterative**
  - Weight Tensor is corrected with error correction factor in one shot
- **Partial Tensor Correction**
  - Traditional weight tensor correction schemes correct the complete weight tensor by certain factor
  - PTC corrects alpha % of the tensor by the same error leading to larger correction factor (0 < alpha < 100)

Partial error correction factor

$$\frac{\text{Output Error}}{\text{Total Input Channels} * \text{alpha \%}}$$

>

Complete error correction factor

$$\frac{\text{Output Error}}{\text{Total Input Channels} * 100\ \%}$$

Partial Weight Tensor

Output error is corrected for alpha % of input channels

Complete Weight Tensor

Output error is corrected for all input channels

P (error reduction) with **Partial error correction**

>

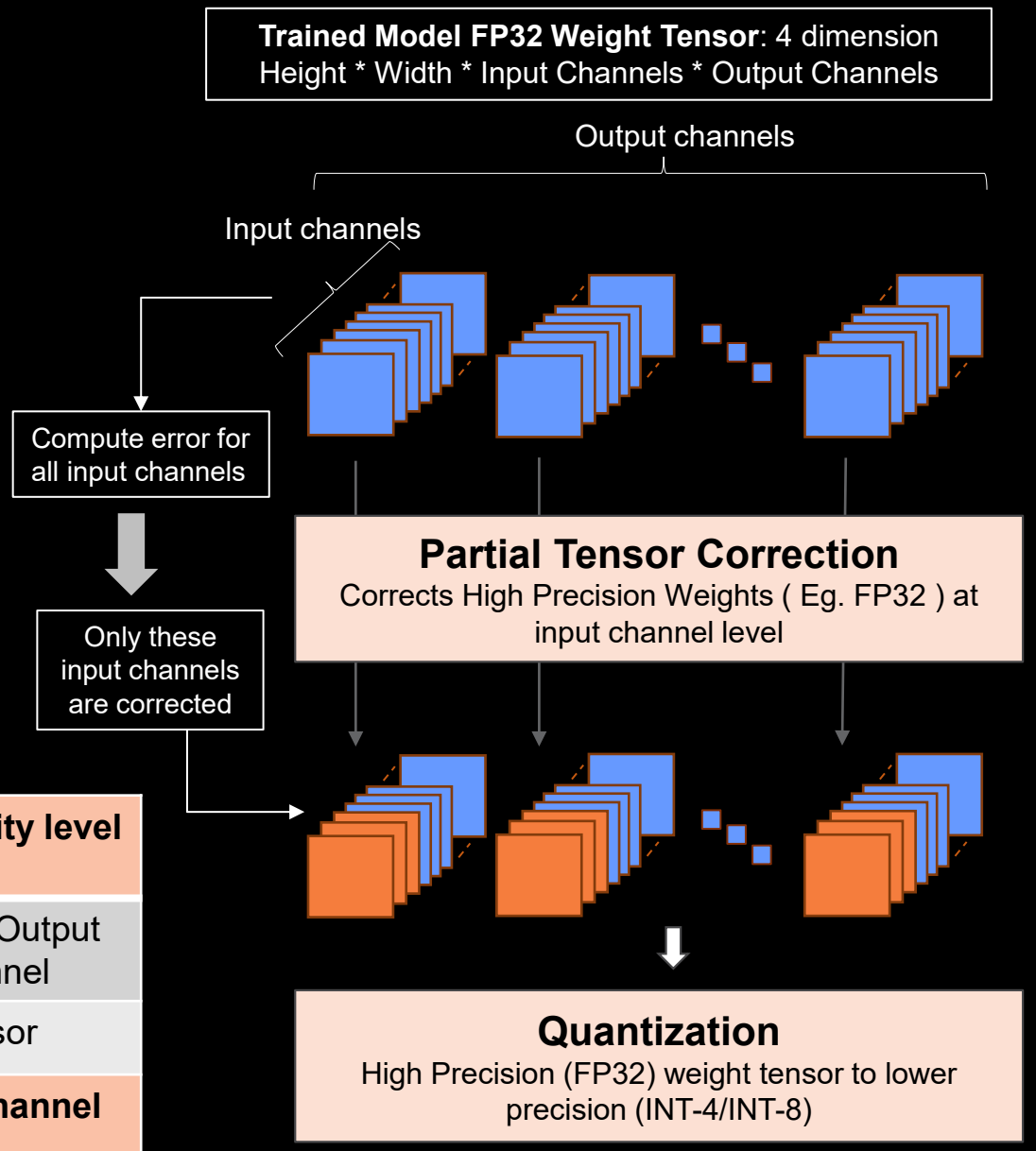
P (error reduction) with **Complete error correction**

Probability of error correction

Proposed Solution (4/8)

'PTC' – A Novel Weight Correction Method

- PTC is efficient in reducing the output errors
  - Smaller corrections on complete tensor – **Under Correction**
  - Higher correction on complete tensor – **Over Correction**
  - **PTC** – Same error is corrected for partial tensor
- PTC can co-exist with SOTA
  - Does not modify quantization scales, rounding schemes or threshold-based clipping
  - Can be extended with Data-Driven Techniques like QAT and PQT for improved accuracy



Techniques	Data-driven	Iterative	Scale correction	Full weight correction	Granularity level
PTQ	Yes/ No	Yes/ No	Yes	Yes	Tensor/ Output Channel
QAT	Yes	Yes	No	Yes	Tensor
PTC	**No**	**No**	**No**	**No**	**Input Channel**

Proposed Solution (5/8)

‘PTR’ – Partial Tensor Retention for Heterogenous Compute

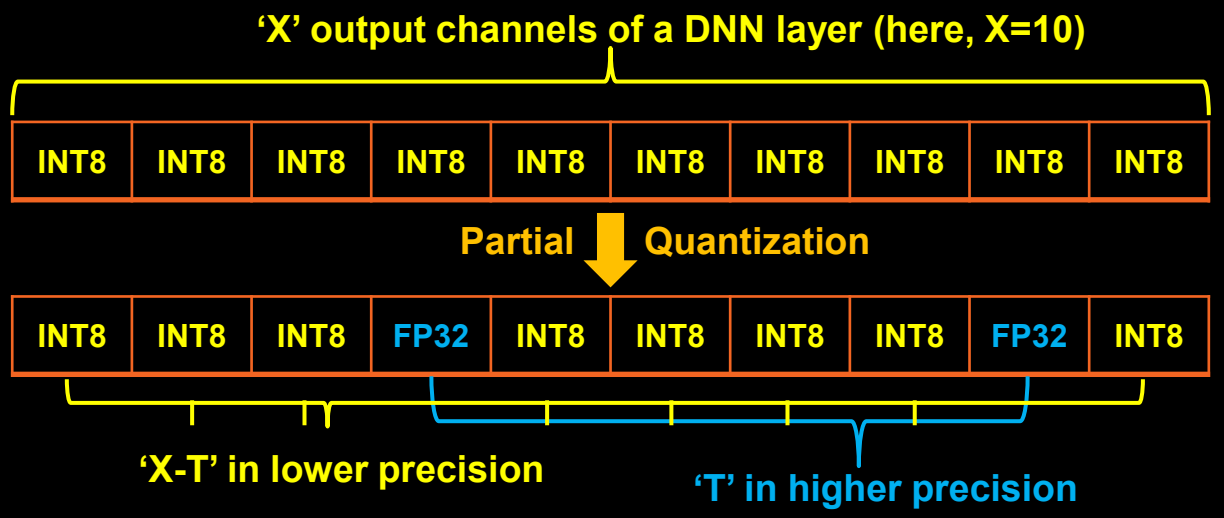
PTR implements the idea of **partial quantization** exploiting the **heterogenous compute**.

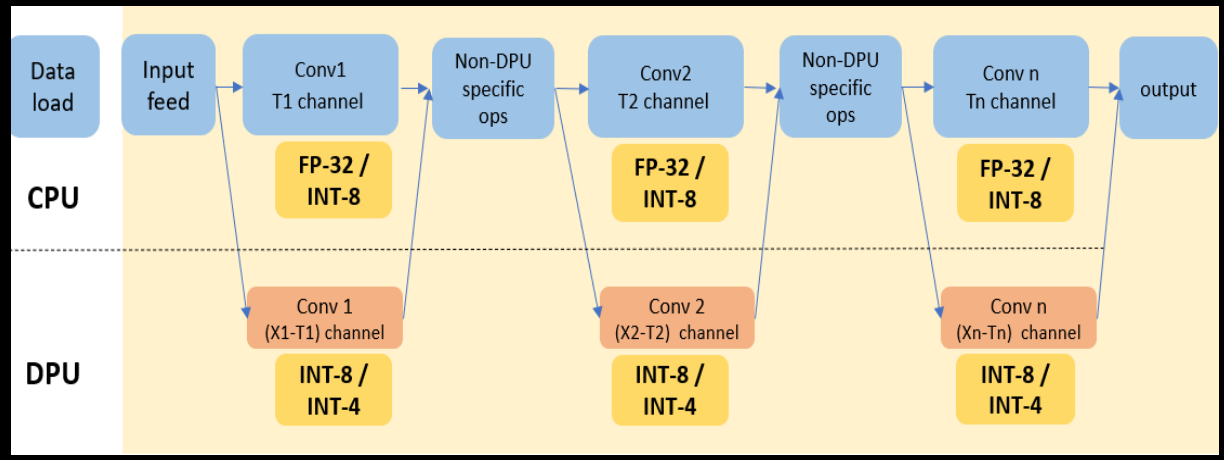
- Achieves superior accuracy along with improved performance.

- **PTR’s partial quantization technique**
  - In each DNN layer, PTR identifies channels sensitive to quantization.
    - ‘T’ sensitive channels from total channels ‘X’.
  - Retains ‘T’ channels in higher precision.
  - X-T channels will be in lower precision.

PTR Quantization noise (Partially quantized) **<** Original Quantization noise (All quantized)

- **Partial quantization for heterogenous compute**
  - Identified sensitive channels ‘T’ will be deployed on CPUs with relatively higher precision.
  - Rest of the channels ‘X-T’ will be deployed on accelerators with lower precision.

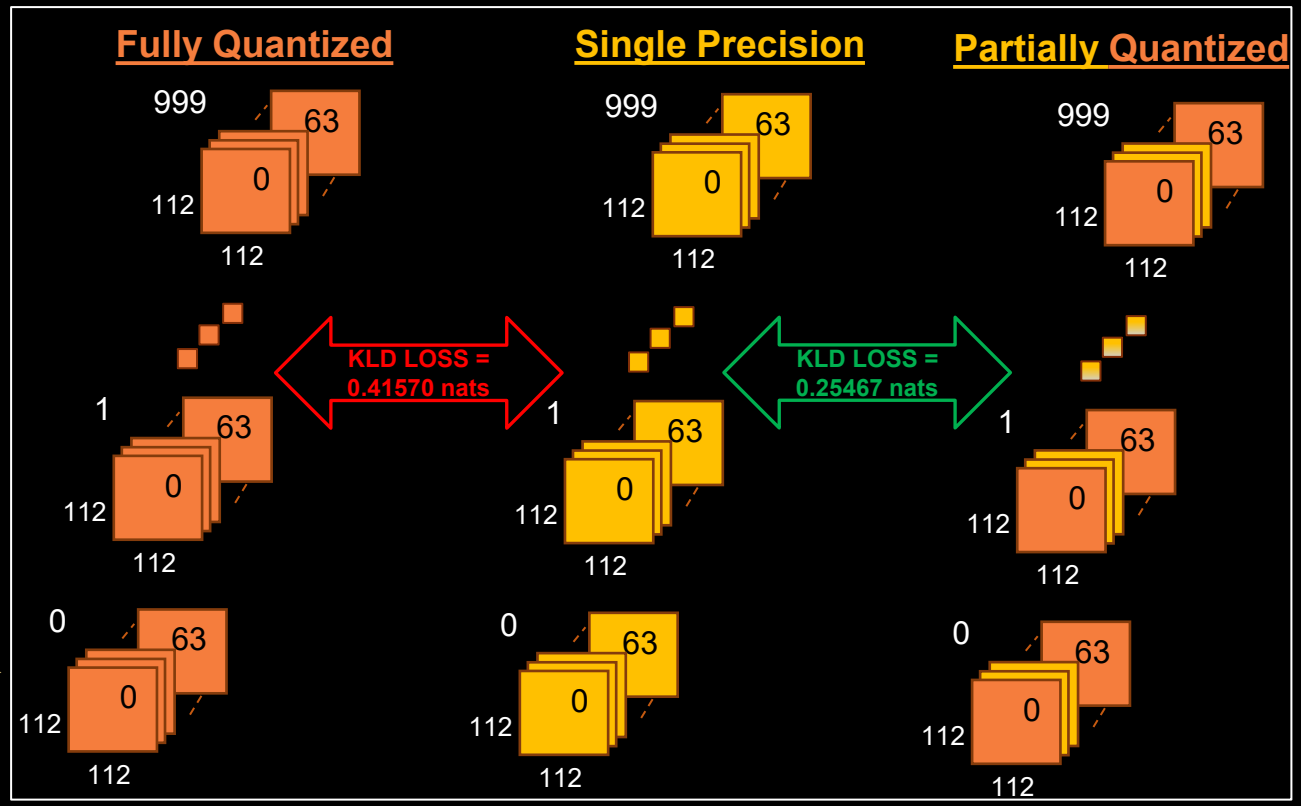




Proposed Solution (6/8)

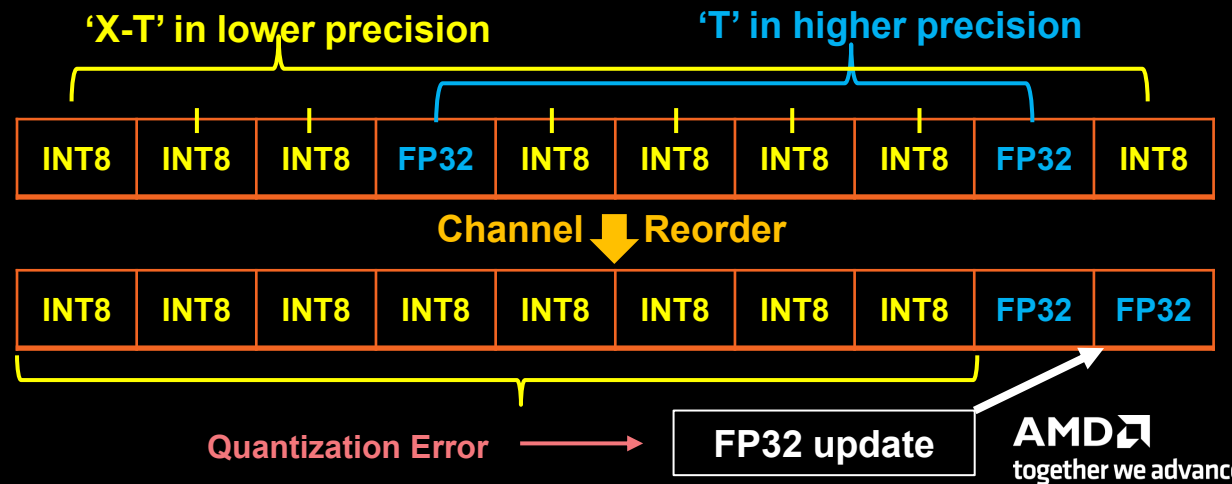
‘PTR’ – Sensitive Channels And Channel Reorder

- **Sensitive Channels (Filters)**
  - Some channels are more sensitive to quantization
    - Due to difference in weight distribution range
  - Sensitive channels are identified
    - Data-Free and Data-driven techniques using MSE and KLD loss
- Fully Quantized vs Partially Quantized
  - Our experiments show KLD Loss between
    - FP32 and fully quantized (INT-8) is 0.41570 nats
    - FP32 and partially quantized versions is reduced to 0.25467 nats



- **Sensitive Channel Reordering**
  - Reordering of sensitive channels such that all the quantization sensitive channels are contiguous
    - To improve memory access latencies

- **Partial Tensor Fine-Tuning**
  - FP32 channels of a DNN layer are only fine-tuned while the quantized values are frozen to improve accuracy



AMD
together we advance_

Proposed Solution (7/8)

'PTR' – Rationale Behind Improved Accuracy And Performance

Heterogenous Quantization with CPUs, DPUs and GPUs reduce Quantization Noise

- Executing few channels in a CPU supported format (like FP32, BF16, INT-8) on CPU
- Others, in lower precision format (like INT4, INT-8, FP-16, other Blocked FP formats) on a GPU and/or DPU
- Improved performance and/or accuracy

Heterogenous Quantization Rounding Schemes simulates Stochastic Rounding

- Rounding to nearest not optimal always
- Every weight value in layer l, can take a value from the set{ $w_i^{l.\ floor}$, $w_i^{l.\ ciel}$ }
- One device follows round up and other follows round down
- Improved accuracy and performance

Heterogenous Quantization Scales simulates Ensembles

- Our work shows that optimal scale factor
  - CPU – Non-power of scale
  - DPU – Power of 2 scale
- Leverage different scales format for different device type computation
- Reduction in quantization noise for a given node
- Find optimal scale factors for each device

CPU	FP-32, BF16	Round up (Ceil)	E.g., Round (281.66) => 282	Non-power of 2 Qnt scales	E.g., Scale = 252.4
Accelerator	INT-4, INT-8	Round down (Floor)	E.g., Round (281.66)= 281	Power of 2 Qnt scales	E.g., Scale = 128

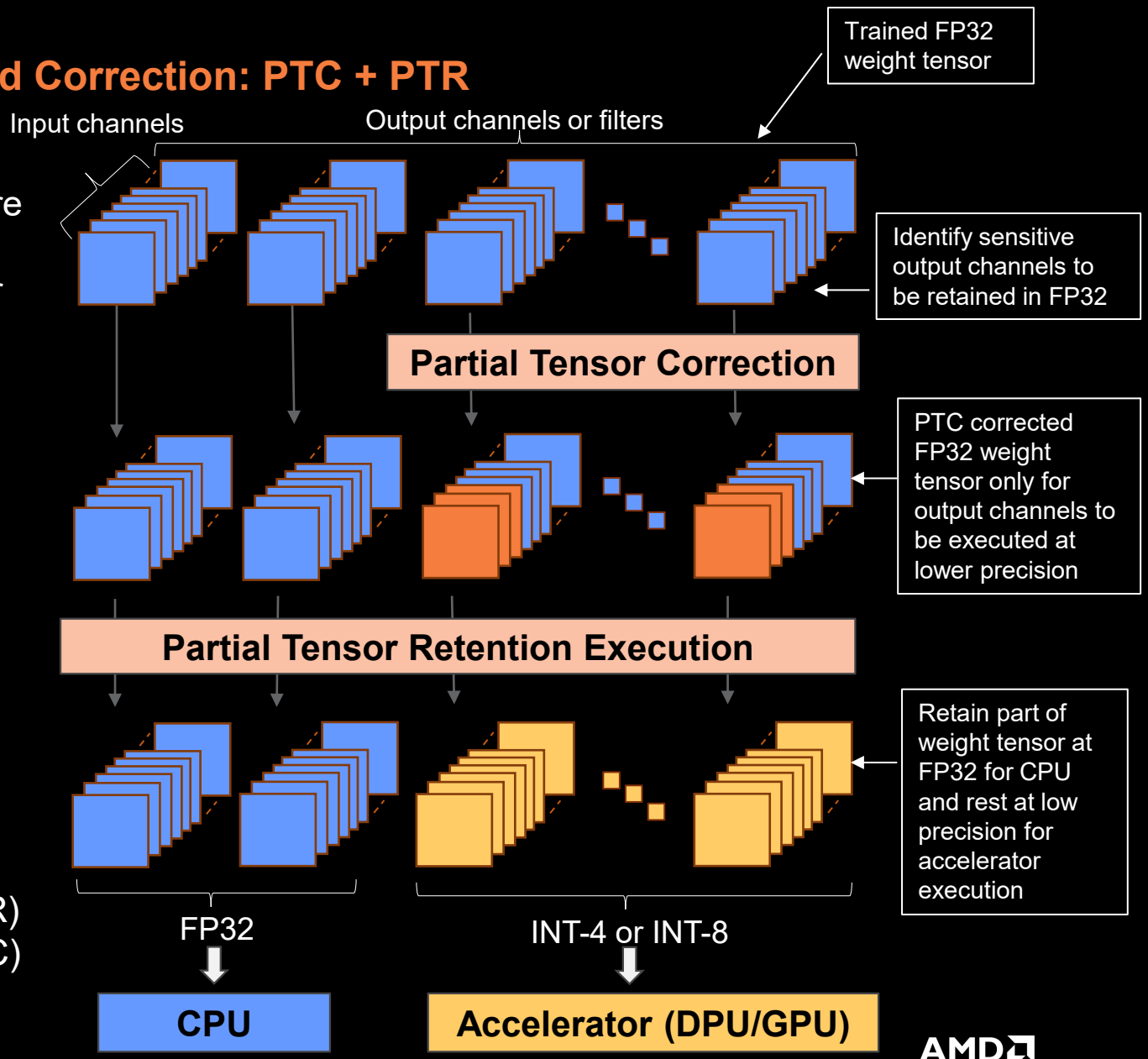
Proposed Solution (8/8)

‘PTRAC’ – Combined Partial Tensor Retention and Correction: PTC + PTR

- **PTR:**
  - Around top 25% of quantization sensitive filters are executed in higher precision on CPU
  - Around 75% of filter compute executed with lower precisions (INT-4 or INT-8) on accelerators
  - INT-4, INT-8 suffer heavy accuracy degradation

- **Combine PTR with PTC:**
  - PTC improves accuracy with low precision (INT4, INT8)
  - Correct the rest 75% filters with PTC
  - 75% PTC corrected filters are executed on accelerators for low precision compute
  - 25% sensitive filters are executed on CPU for higher precision

- **PTRAC improves end to end accuracies with**
  - Retaining sensitive filters at higher precision (PTR)
  - Correcting rest of the filters for low precision (PTC)



Results (1/3) – Quantization Noise Reduction with PTR

By retaining the top 25% of quantization sensitive filters of a DNN node at original format of FP-32, the noise is reduced by

- **Data-Free MSE: 89%**
- **Data-Driven KLD: 55%**

Data-Free Quantization noise reduction computed with MSE

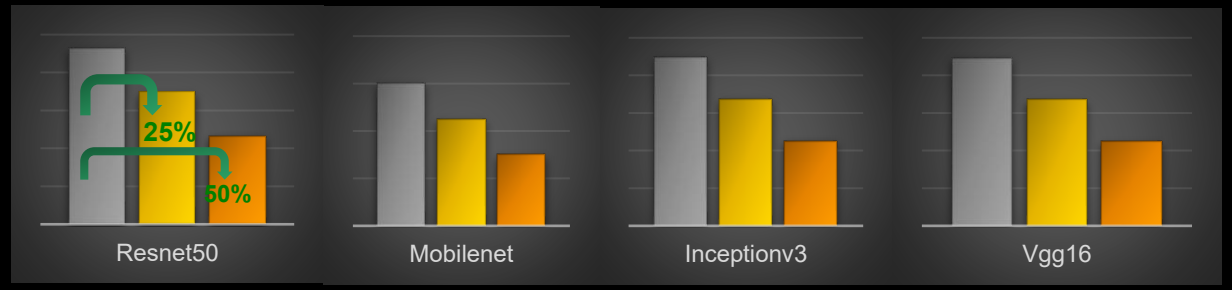
	Quantization Noise		
Models	**All INT8 Quantized**	**25% Partial Retention in FP32**	**% loss reduced**
Resent50	328.6969	**31.5844**	**90.39**
Vgg16	96.8256	**3.0276**	**96.87**
GoogleNet	784.5601	**137.1241**	**82.52**
Densenet121	365.9569	**59.5984**	**83.71**
Squeezenet	153.0169	**10.452289**	**93.16**
Alexnet	350.8129	**71.7409**	**79.55**
Densenet169	228.3121	**38.3161**	**83.21**
Resnet101	285.9481	**18.4041**	**93.56**
Resnet152	251.8569	**15.9201**	**93.67**
Vgg19	111.3025	**4.2436**	**96.18**
Average			**89.07**

Data-Driven Quantization noise reduction computed with KLD loss

Dataset	**Imagenet**					
% INT8 filters	**0**	**25%**	**50%**	**75%**	**100%**	**% loss reduced (with 25% FP32)**
1st Conv Layer	**KL Divergence loss (nats)**					
Resnet50v1.5	0	5.144	18.326	**36.48**	**74.37**	**50.95**
Resnet50v1	0	6.395	26.24	**54.71**	**120.45**	**54.58**
Vgg16	0	5.11	14.44	**29.38**	**60.04**	**51.07**
Mobilenetv1-12	0	4.54	26.26	**68.17**	**136.06**	**49.90**
Shufflenet	0	4.34	11.51	**25.35**	**64.37**	**60.62**

Quantization Noise Reduction

All INT-8 quantized with PTR - Heterogenous Quantization Scale



For all INT-8 quantized with heterogenous scale factors,

- noise reduces by **~ 25% with 25% non-power of 2 scale factors**
- noise reduces by **~ 50% with 50% non-power of 2 scale factors**

Average original noise with original power of 2 quantization scales factors
Average noise with 25% non-power of 2 quantization scales factors
Average noise with 50% non-power of 2 quantization scales factors

Results (2/3) – Accuracy Gains with PTC, PTR and PTC+PTR

Model	INT4 Per Channel	PTC INT4 Per Channel
Inception	59.9	61.9
VGG16	28.4	66
Resnet50	49.1	61.7
Squeezenet	38.6	46.8
Densenet121	53.46	65.07
Alexnet	33.2	52.3
VGG19	51	65.1
Resnet 101	53.5	63.2

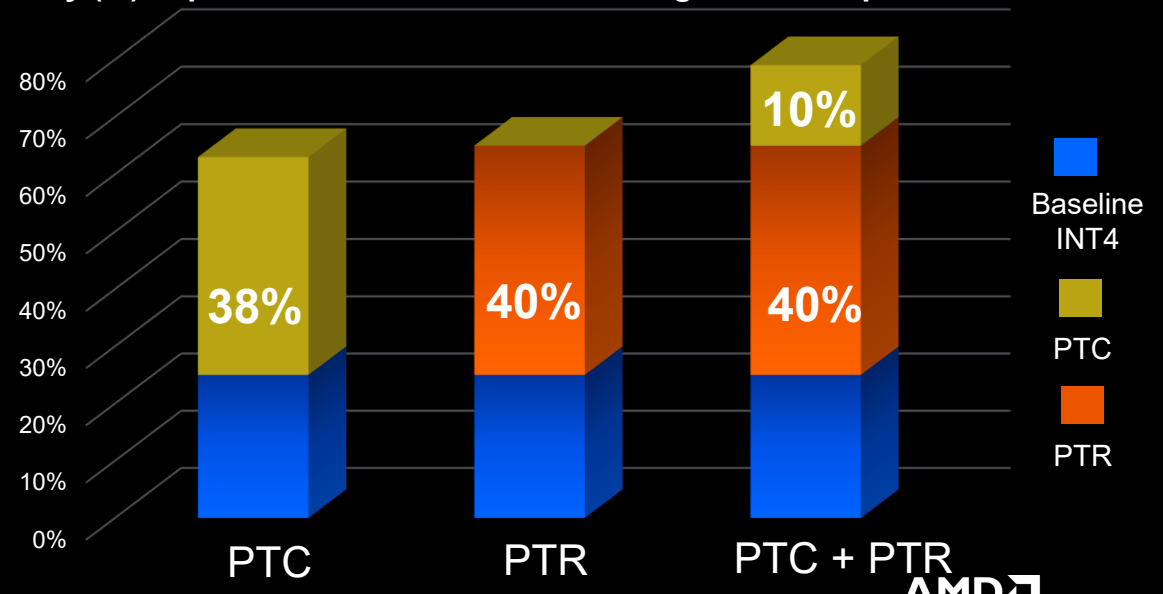
Top-1 Accuracy (%) improvements with PTC for INT-4

Model	INT4 – 100%	INT8 – 100%	INT4: 75%, INT8: 25%	INT4: 50%, INT8: 50%
Alexnet	33.2	56.5	40.2	52.3
Vgg16	28.4	68.3	58.1	65.5
Vgg19	51	68.8	58.6	64.6
Squeezenet	38.6	57.2	46.7	52
Resnet50	49.1	73.2	62.4	64
Resnet101	53.5	73.02	64.7	68.44
Resnet152	57.3	74	67	71.3
Densenet121	53.7	75.5	64.96	70.53
Densenet161	62.6	75.3	66.8	71.1

Top-1 Accuracy (%) improvements with PTR for heterogenous compute: INT4 + INT8

			PTR	PTC + PTR
Model	INT-4	INT-8	INT4: 75%, INT8: 25%	INT4: 75%, INT8: 25%
Alexnet	33.2	56.5	40.2	53
VGG16	28.4	68.3	58.1	64
VGG19	51	68.8	58.6	65.7
Squeezenet	38.6	57.2	46.7	50.9
Resnet50	49.1	73.2	62.4	66.6
Resnet101	53.5	73.02	64.7	68.7
Densenet121	53.7	75.5	64.96	68.4
Densenet161	62.6	75.3	66.8	69.5

Top-1 Accuracy (%) improvements with PTC + PTR for INT4 + INT8



Maximum Top-1 Accuracy (%) improvements over baseline INT4

Results (3/3) – Performance Gains and Model Size Reduction

	CPU + DPU: FP32 + INT8 in 1:2 ratio Mixed precision computation	
Data – Batch, Input Channels, Input Size	% gain over only DPU	% gain over only CPU
256,9,64	50.00	3.8 X
64,11,128	26.36	3.08 X
128,11,128	40.56	3.32 X
256,11,128	53.33	3.79 X
512,11,128	36.36	3.65 X

Simulation results - Performance Improvement with PTRAC: EPYC Milan (FP32) + ALVEO V70 Accelerator (INT8)

	Size in MB		
Models	FP32	INT4: 75%, Float: 25%	INT4: 75%, INT8: 25%
Resnet50	98	33.68	15.31
Resnet101	171	58.78	26.71
Resnet152	232	79.75	36.25
VGG16	528	181.5	82.5
VGG19	549	188.72	85.78
ConvNeXt Large	755	259.53	117.96
ConvNeXtX Large	1310	450.32	204.7

Model Size Reduction with PTRAC

Data-Free

No dependency on training data for PTC and PTR

Can be extended to data-driven techniques like PTQ and QAT

Non-Iterative

One shot weight correction to improve accuracy with low precision

Can be used with retraining and fine-tuning techniques

Accuracy ↑

Accuracies close to high precision model accuracies

FP32 + INT4 accuracy close to FP32 accuracy.

Up to 40% wrt INT4

Performance ↑

Performance close to low precision performance

FP32 + INT4 performance close to INT4 performance

Up to 55% wrt DPU

Up to 3.8X wrt CPU

Model Size ↓

FP32 + INT4 size reduces by ~65% w.r.t. FP32 model size

INT8 + INT4 size reduces by ~84% w.r.t. FP32 model size.

Up to 84%


AMD