Semiconductor Research Corporation

CoCoSys
CENTER FOR THE CO-DESIGN OF COGNITIVE SYSTEMS

Georgia Tech

SYNERGY

Algorithm-Hardware Co-Design for Data-Free Quantization and Inference using Contrastive Learning and Next Generation Arithmetic

Presenter: Akshat Ramachandran (**Advisor:** Tushar Krishna)

Collaborators: Zishen Wan, Geonhwa Jeong, John Gustafson (ASU) and Souvik Kundu (Intel)

Georgia Institute of Technology

Contact: akshat.r@gatech.edu (Akshat Ramachandran)

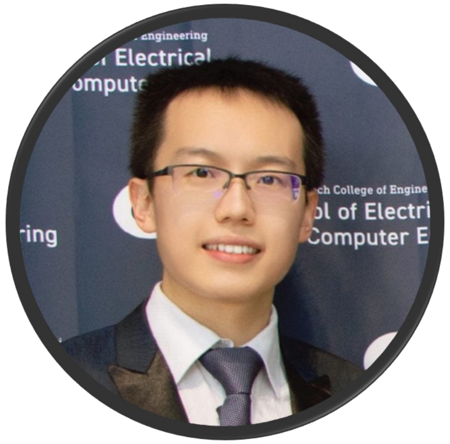
SRC

April 11, 2024

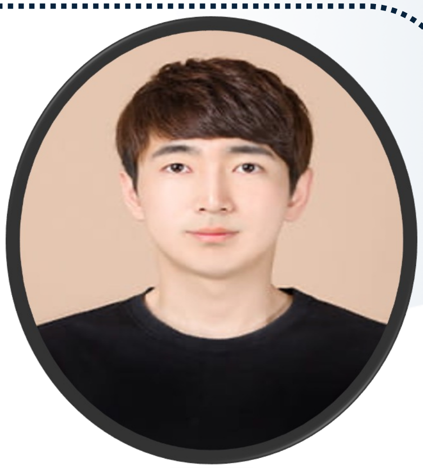
Team

Students

Akshat Ramachandran (GT)



Zishen Wan (GT)



Geonhwa Jeong (GT)

PIs

Tushar Krishna (GT)

John Gustafson (ASU)

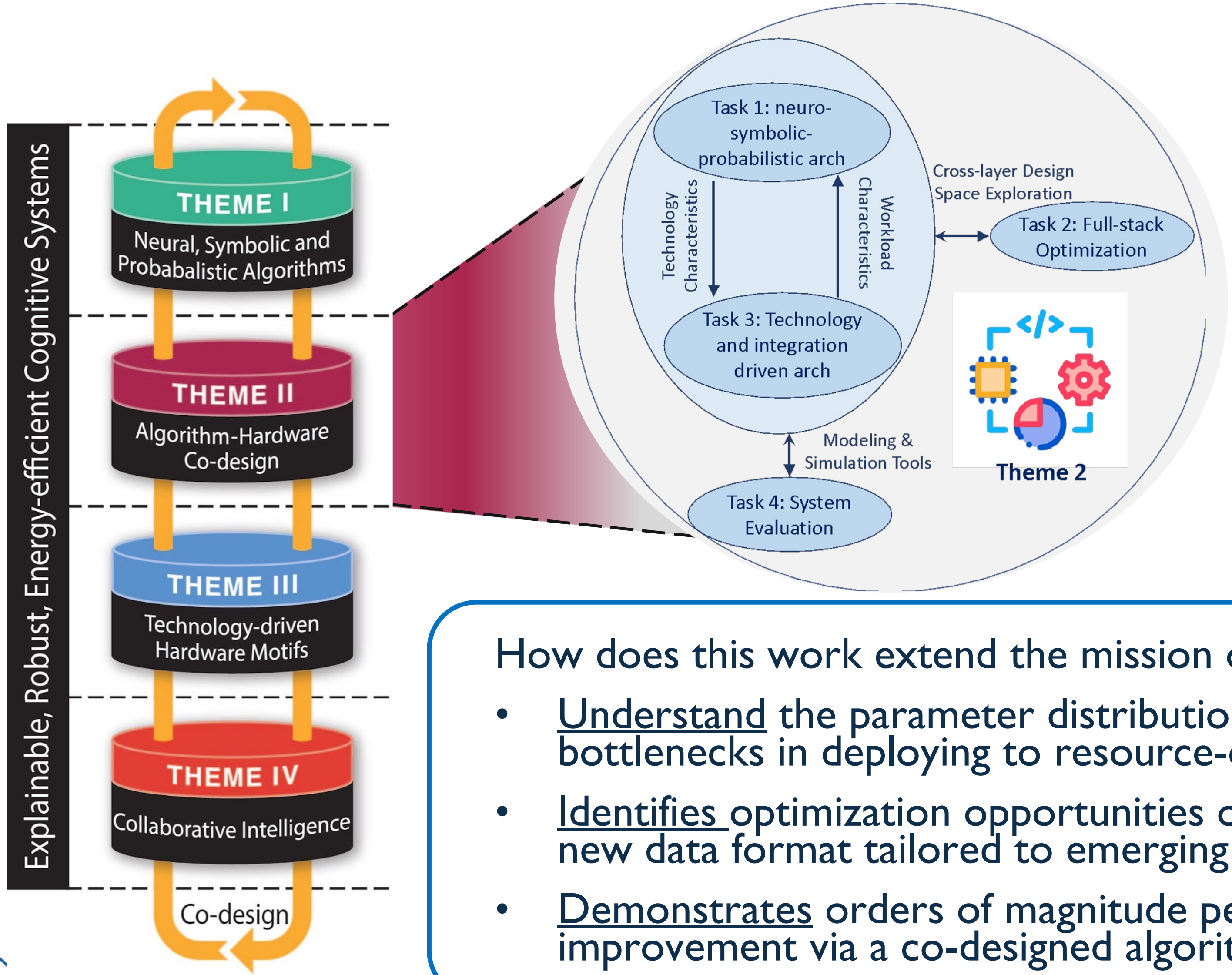
Liaison

Souvik Kundu (Intel)

Executive Summary

- Why ?
  - Existing quantization data-formats lack adaptability and/or are inefficient for implementation in resource-constrained devices.
  - Privacy and security reasons limiting calibration data access for quantization.
  - Lack of a generalized quantization framework in prior art.
- What ?
  - Design of a composite data type called Logarithmic Posits (LP) that blends the adaptability of posits with hardware efficiency of Logarithmic Numbers.
  - Novel genetic-algorithm based data-free quantization framework leveraging contrastive learning for synthetic data-generation.
  - Unified mixed-precision Logarithmic Posit Accelerator (LPA) for high throughput DNN inference.
- How ?
  - Employs a co-design approach, integrating the development of the LP data type, the quantization framework, and the LPA architecture. This integrated approach allows for dynamic adaptation to DNN parameter distributions, resulting in highly efficient DNN inference.

Relevance to the Center Goals



Explainable, Robust, Energy-efficient Cognitive Systems
THEME I
Neural, Symbolic and Probabalistic Algorithms
THEME II
Algorithm-Hardware Co-design
THEME III
Technology-driven Hardware Motifs
THEME IV
Collaborative Intelligence
Co-design
Task 1: neuro-symbolic-probabilistic arch
Technology Characteristics
Workload Characteristics
Cross-layer Design Space Exploration
Task 2: Full-stack Optimization
Task 3: Technology and integration driven arch
Modeling & Simulation Tools
Task 4: System Evaluation
Theme 2

Theme II Mission Statement

The co-design of cognitive hardware driven by the evolution of cognitive workloads as well as the capabilities of future hardware technologies has the potential to unlock quantum improvements in processing efficiency

How does this work extend the mission of CoCoSys?

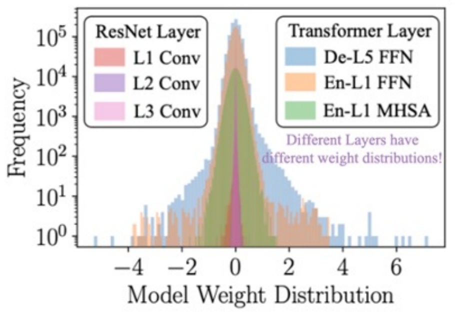
- Understand the parameter distribution variance of neuroinspired models and bottlenecks in deploying to resource-constrained devices.
- Identifies optimization opportunities over existing alternatives by demonstrating a new data format tailored to emerging neuroinspired models.
- Demonstrates orders of magnitude performance and energy efficiency improvement via a co-designed algorithm-hardware approach.

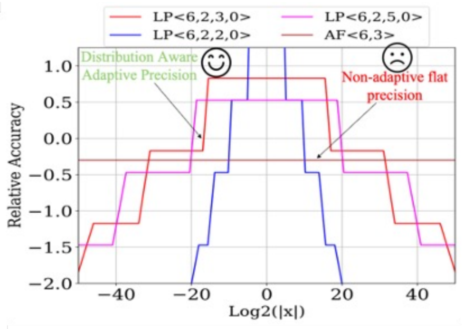
SRC

Outline

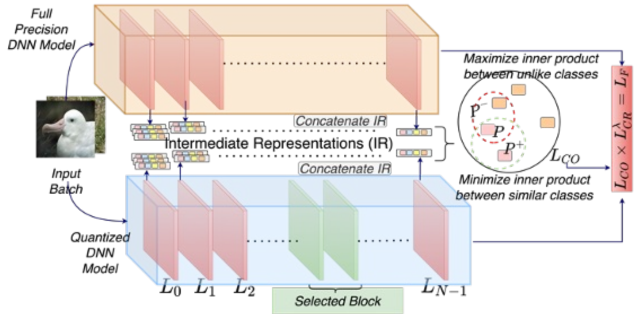
Algorithm

Introduction

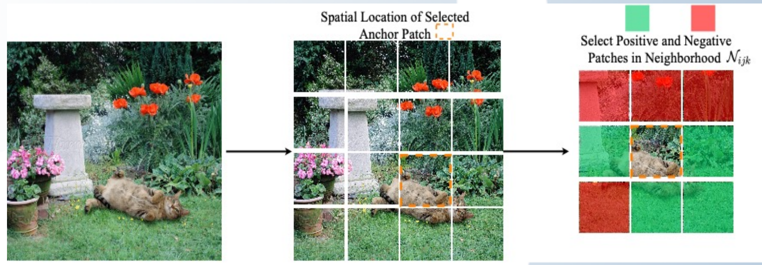




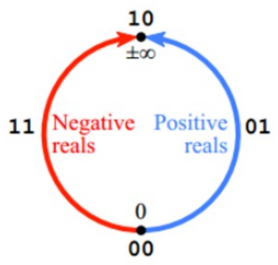
Automated Quantization


Full Precision DNN Model
Input Batch
Quantized DNN Model
Concatenate IR
Intermediate Representations (IR)
Maximize inner product between unlike classes
Minimize inner product between similar classes
Selected Block

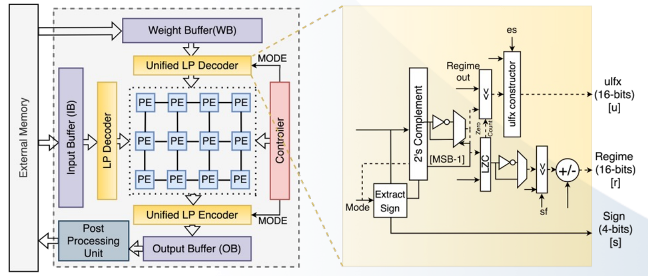
Data-Free Quantization


Spatial Location of Selected Anchor Patch
Select Positive and Negative Patches in Neighborhood

Next Generation Arithmetic: Logarithmic Posits



Hardware: Logarithmic Posit Accelerator

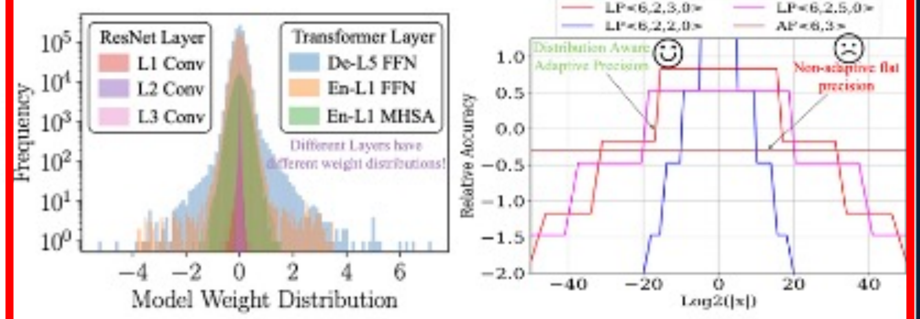


Hardware

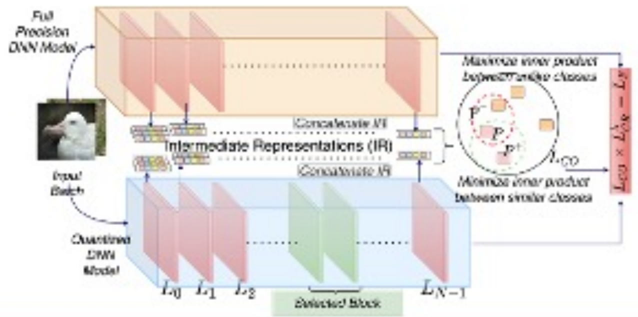
Outline

Algorithm

Introduction



Automated Quantization


Full Precision DNN Model
Input Batch
Quantized DNN Model
Intermediate Representations (IR)
Concatenate IR
Selected Block
Maximize inner product between different classes
Minimize inner product between similar classes

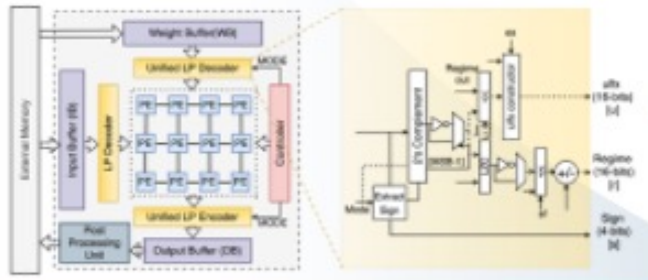
Data-Free Quantization

Spatial Location of Selected Anchor Patch
Select Positive and Negative Patches in Neighborhood

Next Generation Arithmetic: Logarithmic Posits

10
±∞
11
Negative reals
Positive reals
01
0
00

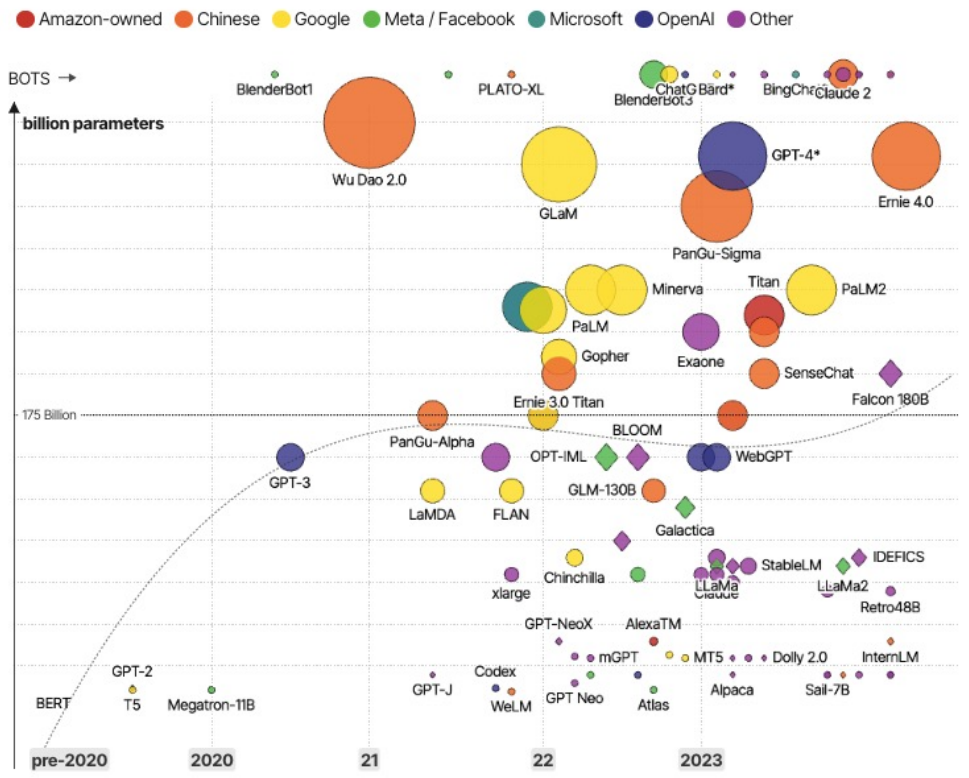
Hardware: Logarithmic Posit Accelerator

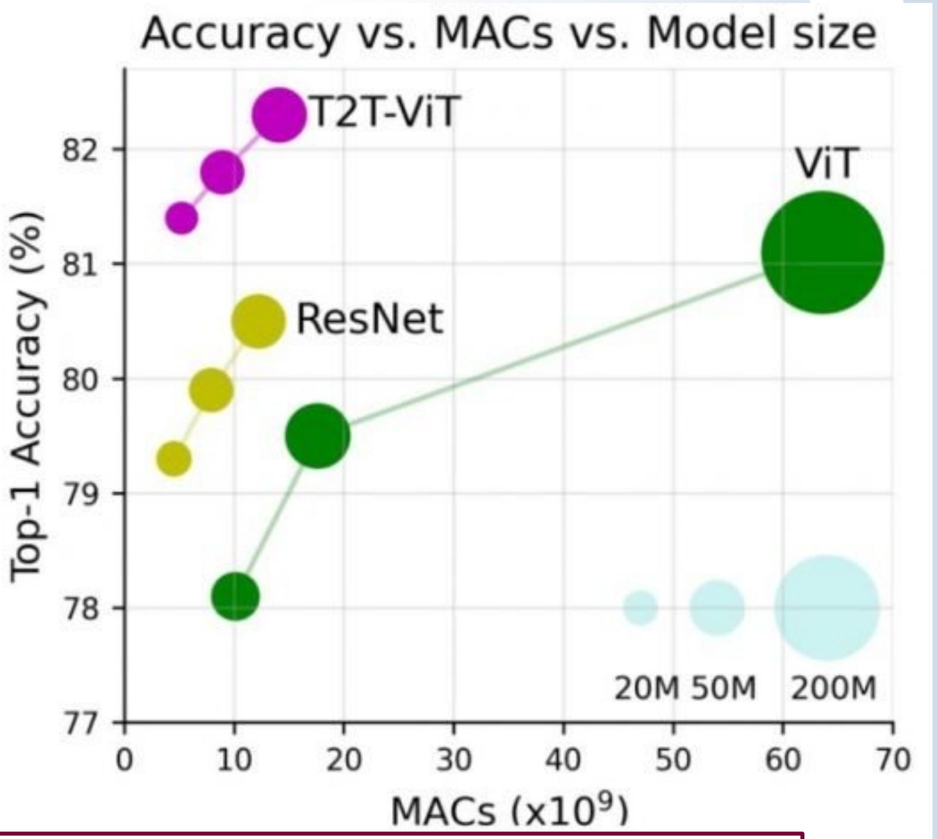


Hardware

SRC

From Bulky DNNs to Sleek Edge Deployment!





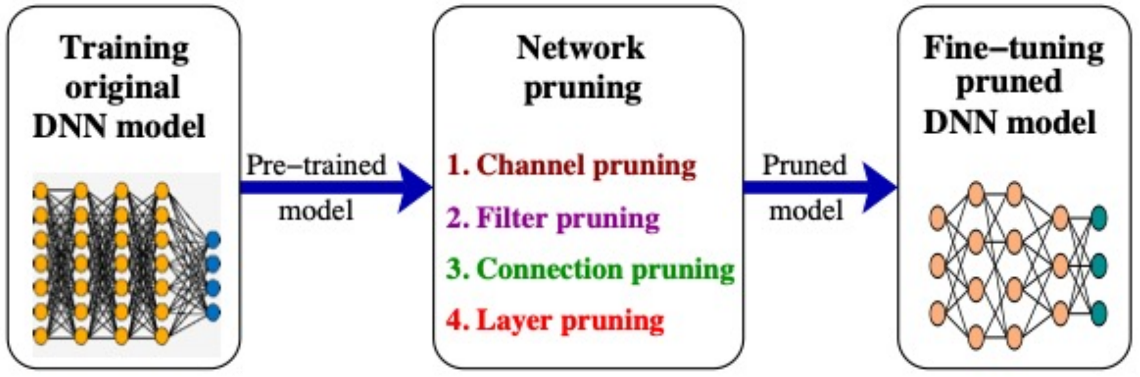
❖ YoY increase in DNN sizes leads to escalating computational and storage demands!
❖ Limited compute, storage resources and energy budget of edge devices (e.g., phones) makes deployment challenging!

SRC

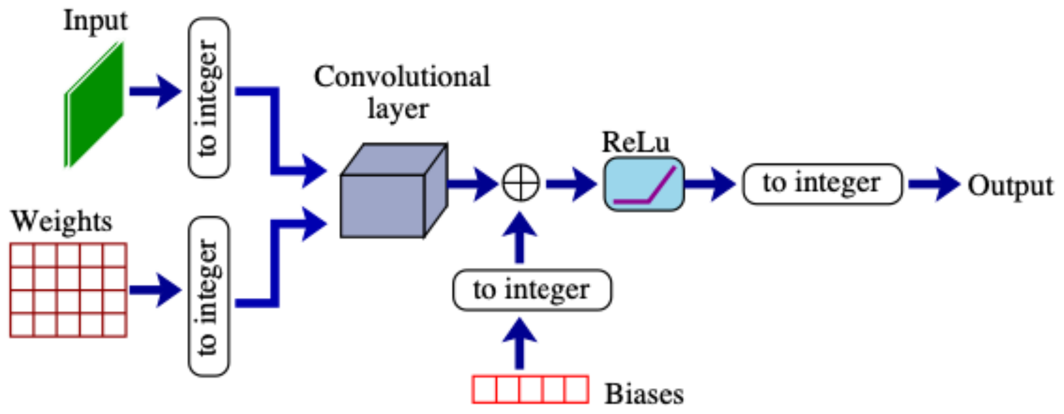
Source: NVIDIA, Viso.ai, LifeArchitect.ai

From Bulky DNNs to Sleek Edge Deployment!

Model Compression techniques for efficient DNN deployment:



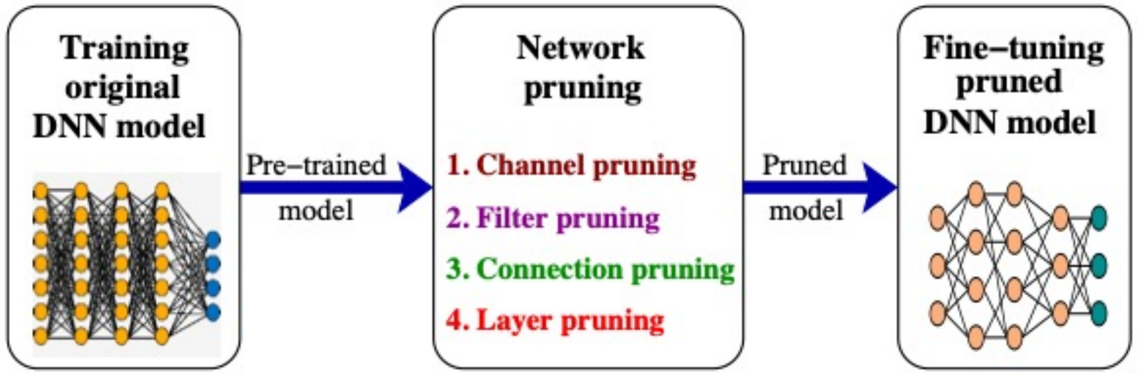
Pruning



Quantization

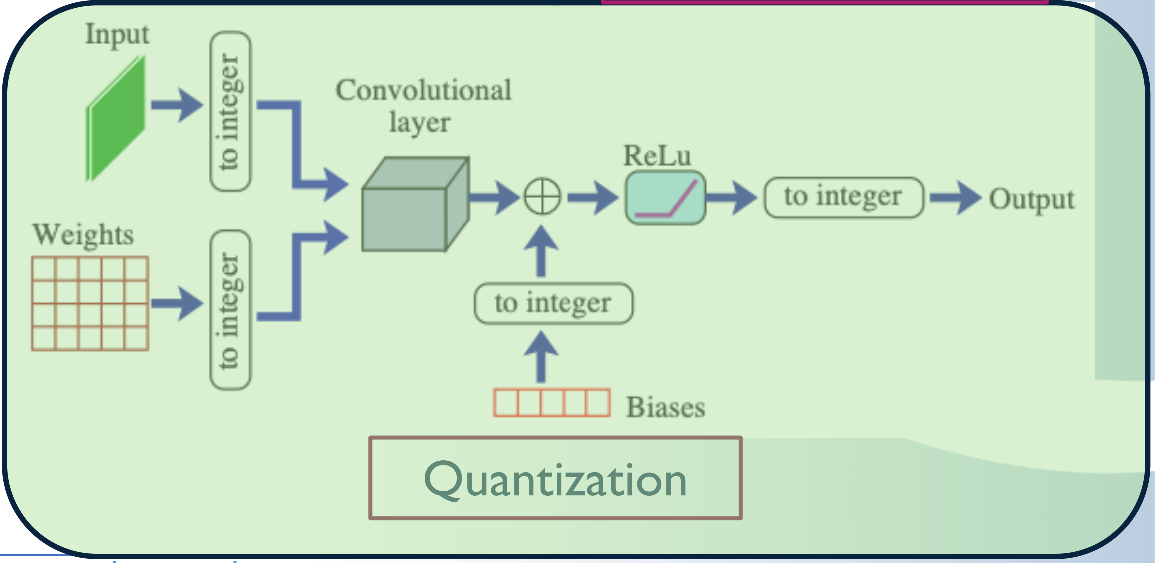
From Bulky DNNs to Sleek Edge Deployment!

Model Compression techniques for efficient DNN deployment:



Pruning

Focus of our work!



Quantization

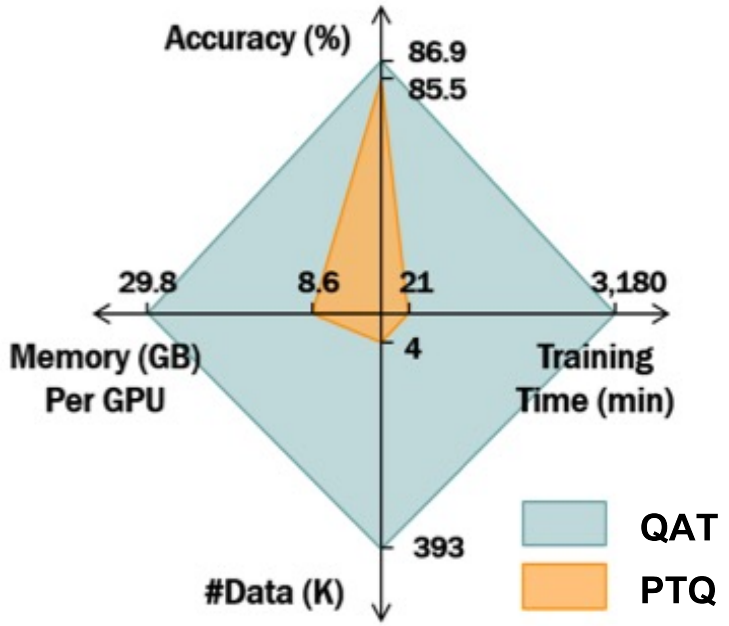
Source: Rahul, M et al. "A survey on deep neural network compression: challenges, overview, and solutions." arXiv preprint arXiv:2010.03954 (2020).

Background: Types of Quantization Techniques

Quantization Aware Training (QAT):

Higher Accuracy
Improved Model Robustness

❌ Increased training complexity and time
❌ Large-scale data dependency



Focus of our work!

Post Training Quantization (PTQ):

Speed and Simplicity
Low data requirement (Can also be data-free)

❌ Potential accuracy drop
❌ Sensitivity to calibration dataset

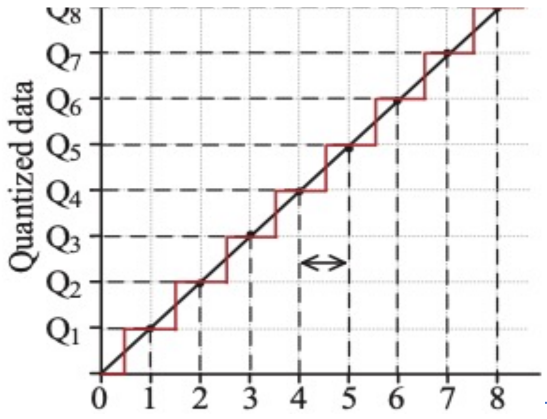
Source: Bai, Haoli, et al. "Towards efficient post-training quantization of pre-trained language models." NeurIPS 2022.

Background: Types of Quantization Formats

Uniform Quantization

- Assigns same quantization step size cross the entire range of values.
- Results in simpler hardware but leads to higher quantization error for distributions with large variances.
- Example: Integers.

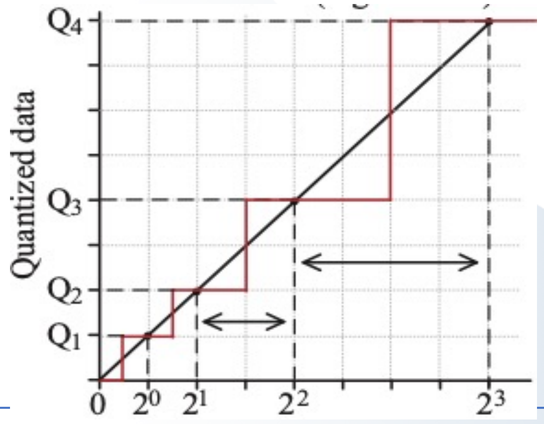
$$Q(X, \gamma, b) = clip(\left\lfloor \frac{X}{\gamma} \right\rceil, -2^{b-1}+1, 2^{b-1}-1)$$



Non-Uniform Quantization

- Has variable distribution intervals, for selective quantization for significant data or extensive dynamic ranges.
- Potentially complex hardware and requires automated quantization algorithms.
- Example: IEEE-754 Floating-Point, Posits, Vector Quantization, Logarithmic Posits.

$$Q(X, b, e, f) = sign(X) \times 2^e \times 1.f$$



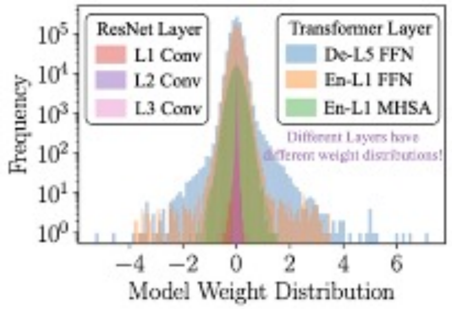
Source: Liu, Fangxin, et al. "Improving neural network efficiency via post-training quantization with adaptive floating-point." CVPR. 2021.

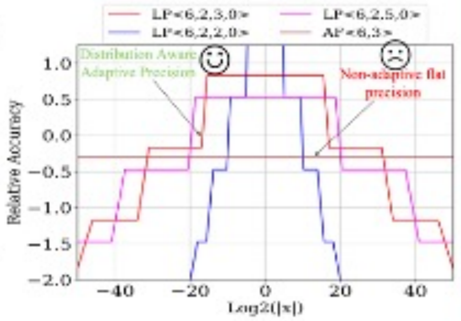
Outline

Algorithm

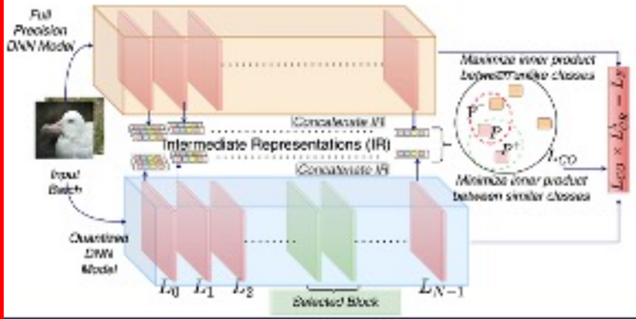
To be presented at DAC'24

Introduction





Automated Quantization


Full Precision DNN Model
Input Batch
Quantized DNN Model
Intermediate Representations (IR)
Concatenate IR
Maximize inner product between unlike classes
Minimize inner product between similar classes
Selected Block

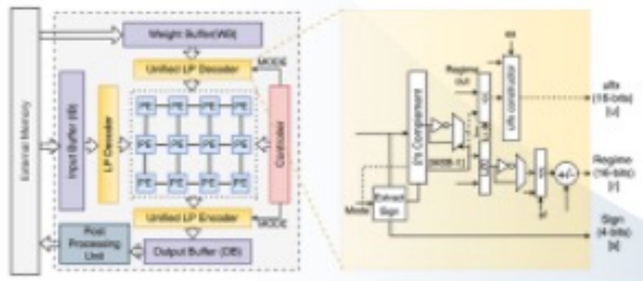
Data-Free Quantization

Spatial Location of Selected Anchor Patch
Select Positive and Negative Patches in Neighborhood

Next Generation Arithmetic: Logarithmic Posits

10
±∞
11
Negative reals
Positive reals
01
0
00

Hardware: Logarithmic Posit Accelerator

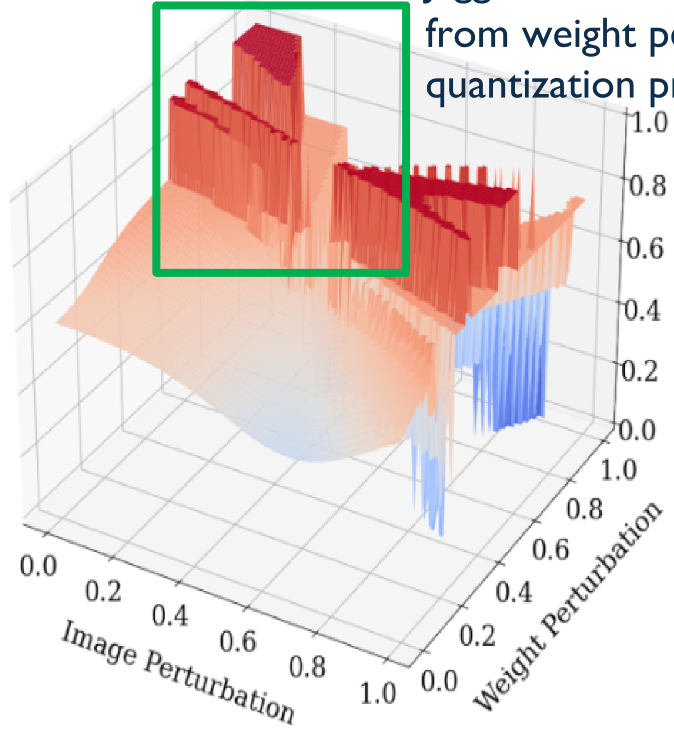


Hardware

Problem: Non-smooth Loss Landscape

DNNs particularly depict a non-smooth quantization loss landscape.

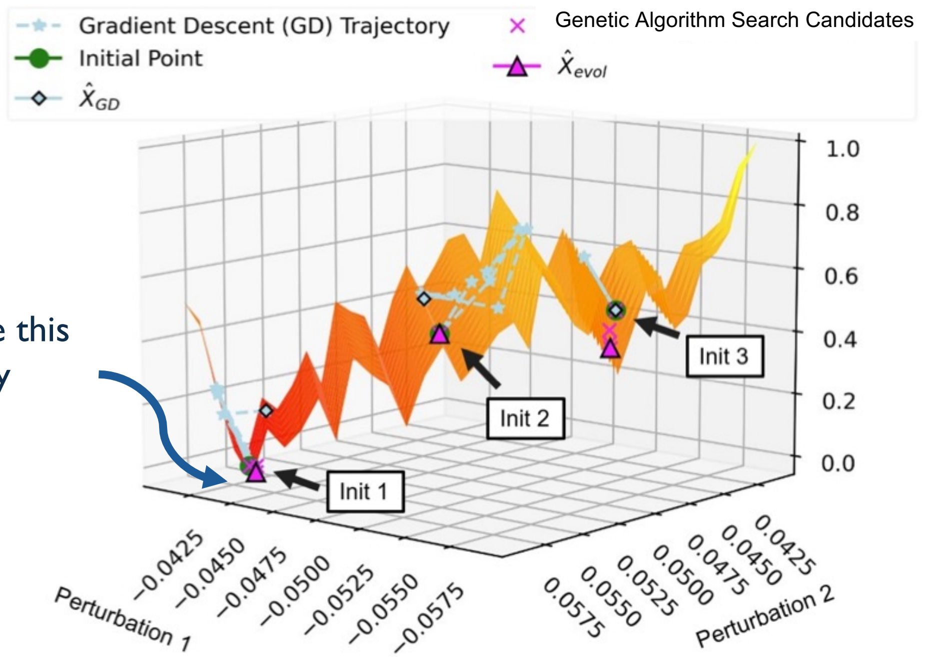
Jagged and non-smooth peaks arise from weight perturbations during the quantization process



Problem: Gradient-Descent Cannot Traverse this Landscape

First and second-order gradient cannot be used satisfactorily to traverse this non-smooth loss landscape because of the presence of multiple local-minima.

GA-based search is able to traverse this loss landscape and optimally identify quantization parameters



Our Solution

We therefore adopt a GA based layer-wise quantization framework.

Why layer wise ? Perturbing too many layers at a time can cause traversal of a high-dimensional search space and does not guarantee convergence.

Background: Genetic Algorithm

Consider a population of randomly rabbits: some individuals are potentially faster and smarter than others.

Background: Genetic Algorithm

- Slower, dumber rabbits are likely to be caught and eaten by foxes.
- Fast, smart rabbits survive to do what rabbits to best: make more rabbits!!

Background: Genetic Algorithm

- The rabbits that survive breed with each other to generate offspring, which starts to mix up their genetic traits
- – Fast rabbits might breed with fast rabbits
- – Fast rabbits with slow rabbits
- – Smart with not-so-smart, etc…

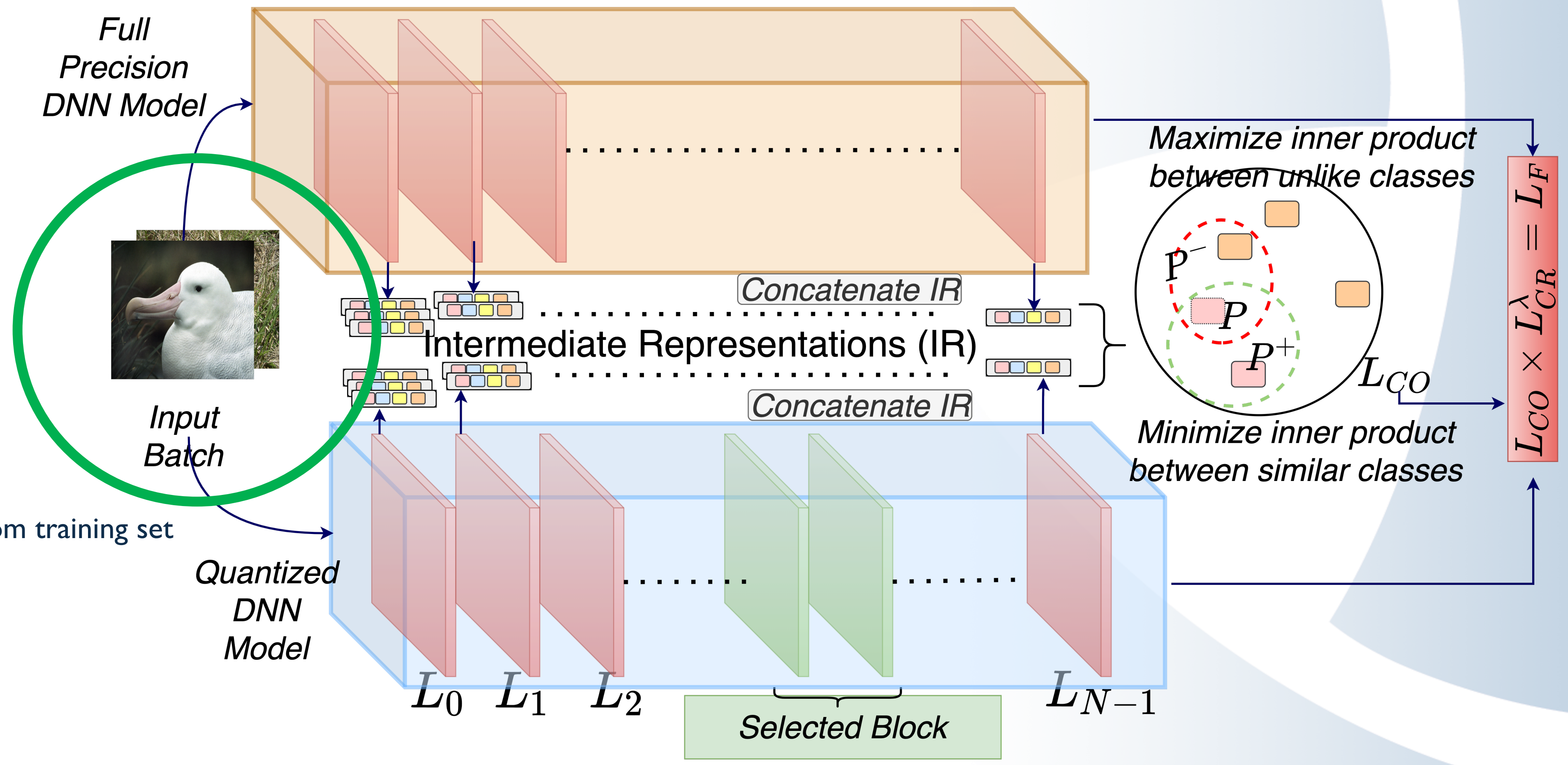
- Furthermore, nature occasionally throws in a "wild hare" because genes can mutate.

Background: Genetic Algorithm

- At the end the rabbits that survive will be the fastest, strongest and smartest among the population because the fox would've eaten the slow, weak and not-so-smart ones.

- In this analogy, an individual rabbit represents a solution to the problem (i.e. a single point in the state space).
- The foxes represent the problem constraints – Solutions that do well are likely to survive.
- We create similar such notions for quantization.

Proposed: Automated Quantization Framework

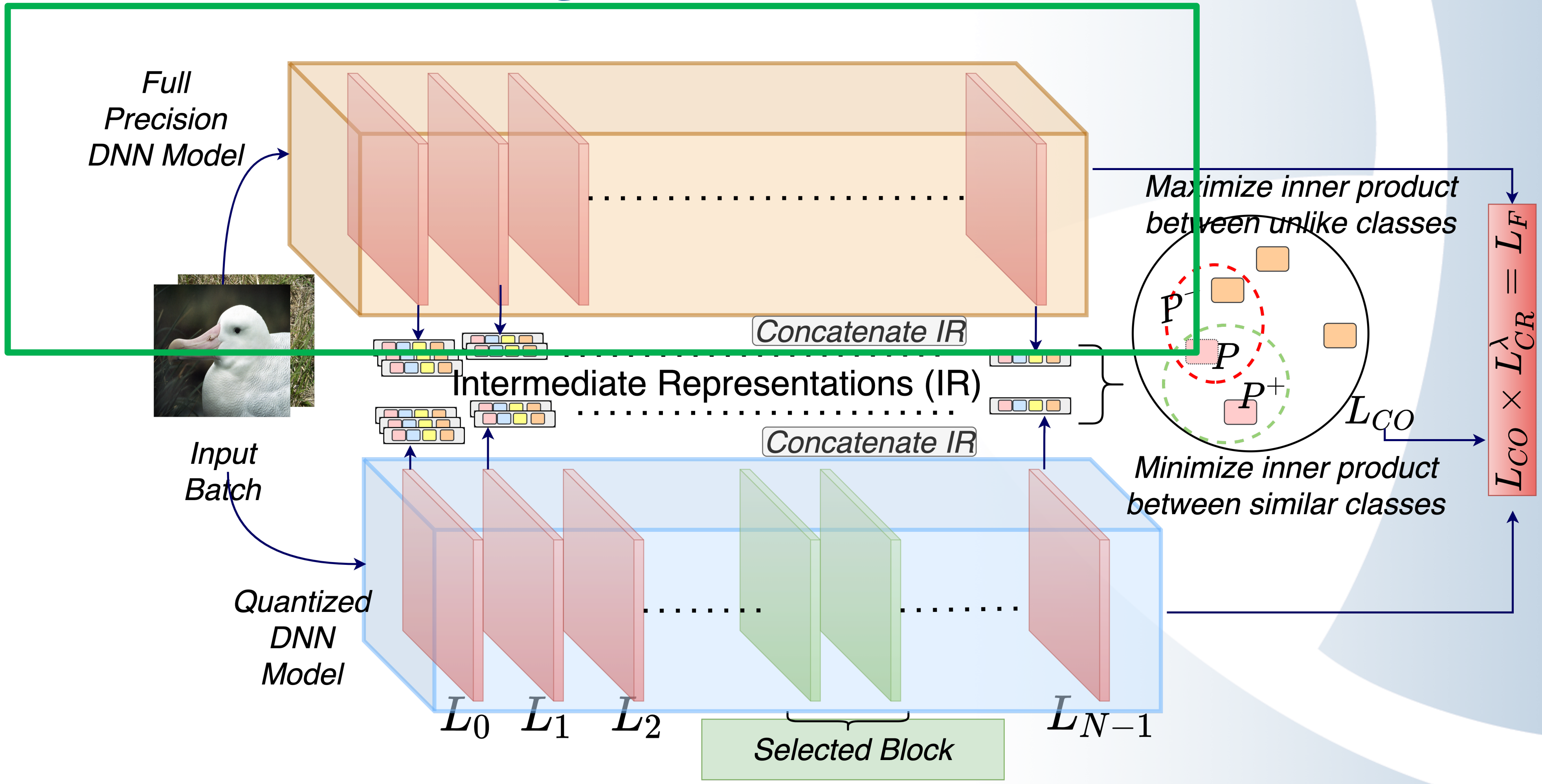


Full Precision DNN Model
Input Batch
Concatenate IR
Intermediate Representations (IR)
Concatenate IR
Maximize inner product between unlike classes
P^-
P
P^+
L_{CO}
Minimize inner product between similar classes
$L_{CO} \times L_{CR}^{\lambda} = L_F$
Quantized DNN Model
L_0
L_1
L_2
Selected Block
L_{N-1}

128 calibration images drawn from training set

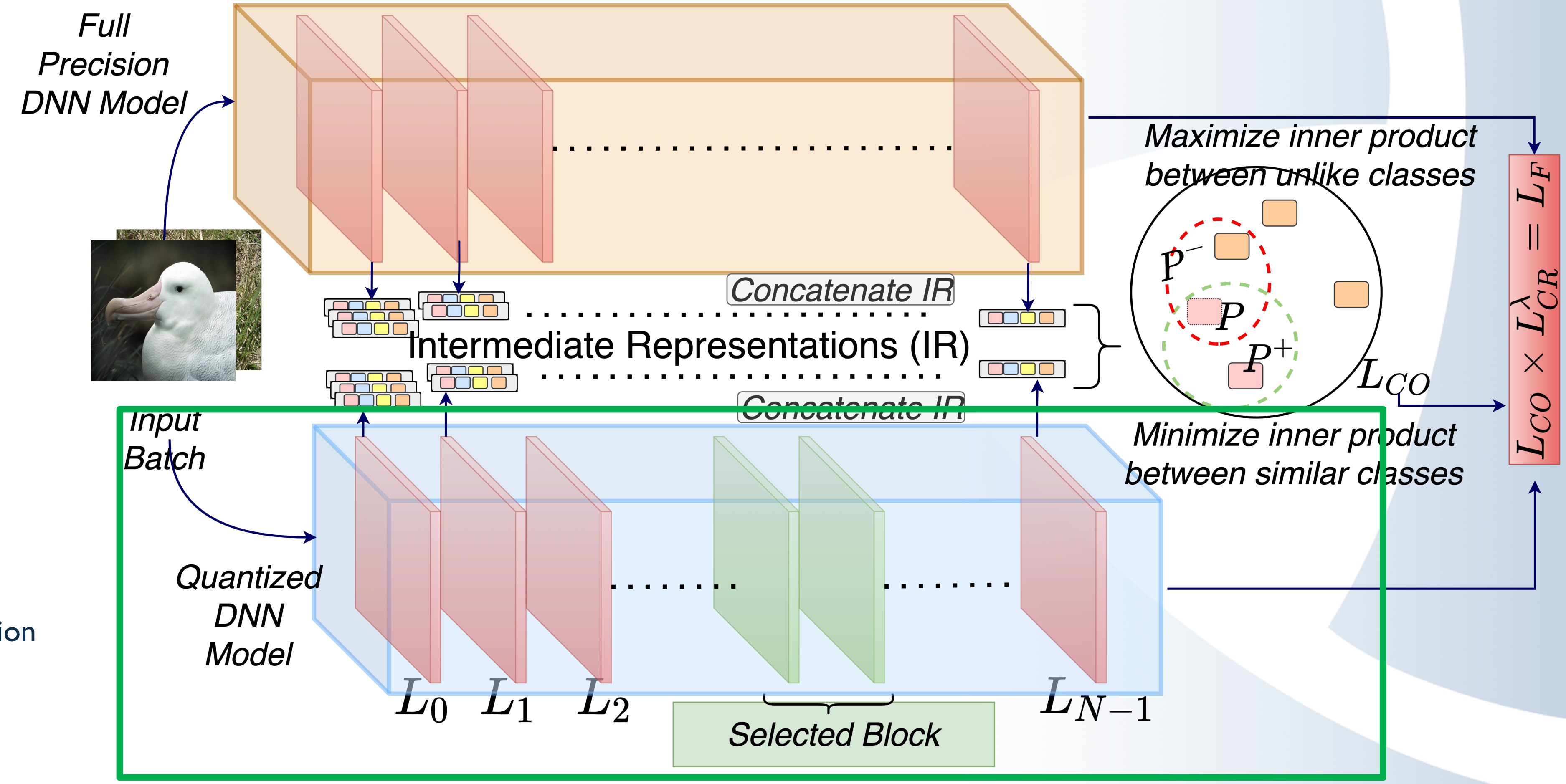
Proposed: Automated Quantization Framework

Full-Precision Model



Full Precision DNN Model
Input Batch
Quantized DNN Model
Concatenate IR
Intermediate Representations (IR)
Concatenate IR
Maximize inner product between unlike classes
Minimize inner product between similar classes
P^-
P
P^+
L_{CO}
$L_{CO} \times L_{CR}^{\lambda} = L_F$
L_0
L_1
L_2
L_{N-1}
Selected Block

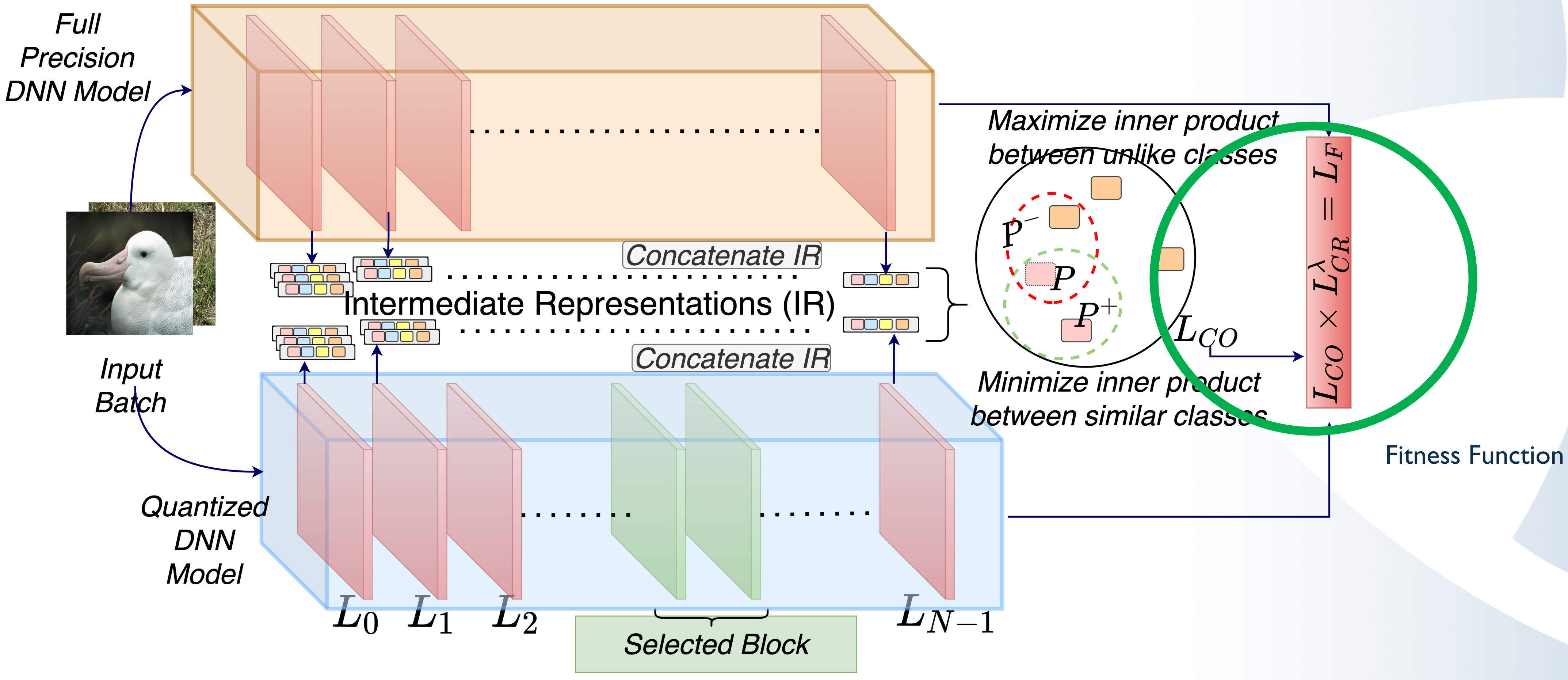
Proposed: Automated Quantization Framework



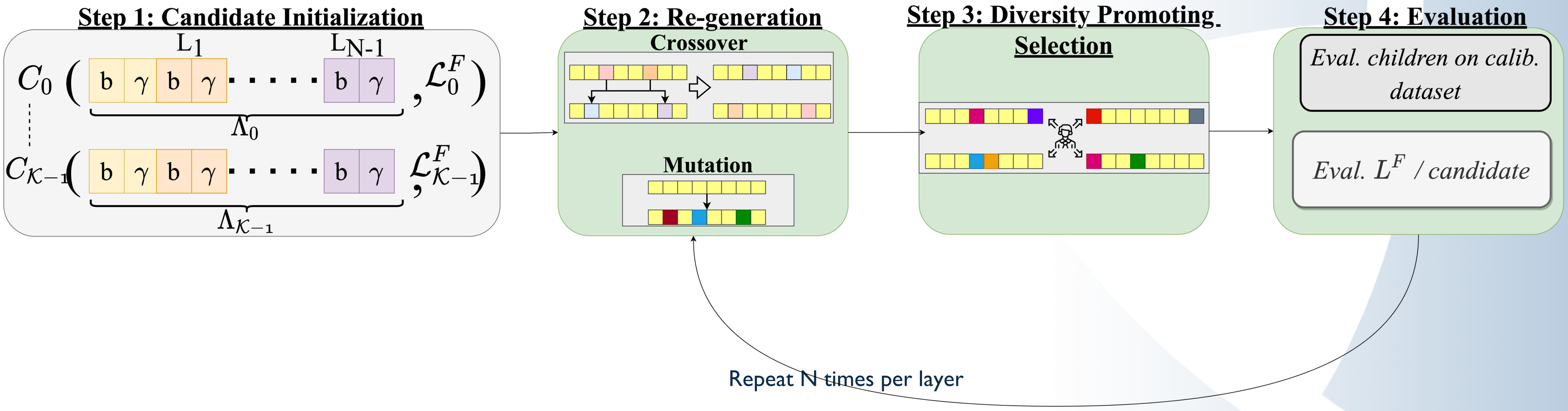
Full Precision DNN Model
Concatenate IR
Intermediate Representations (IR)
Concatenate IR
Input Batch
Maximize inner product between unlike classes
P^-
P
P^+
L_{CO}
Minimize inner product between similar classes
$L_{CO} \times L_{CR}^{\lambda} = L_F$
Quantized DNN Model
L_0
L_1
L_2
L_{N-1}
Selected Block

Randomly initialized model for quantization

Proposed: Automated Quantization Framework

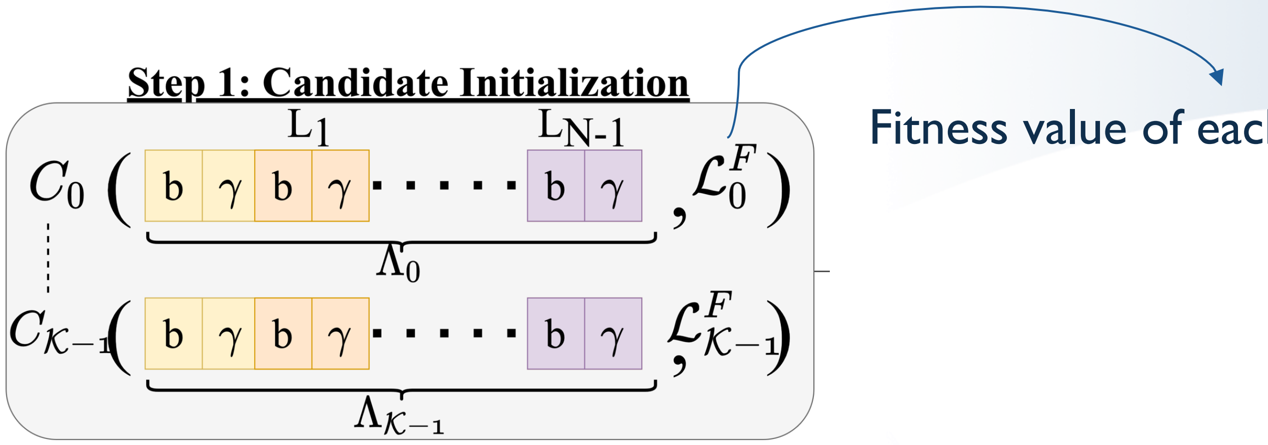

Full Precision DNN Model
Input Batch
Quantized DNN Model
Concatenate IR
Intermediate Representations (IR)
Concatenate IR
L_0
L_1
L_2
Selected Block
L_{N-1}
Maximize inner product between unlike classes
P^-
P
P^+
L_{CO}
Minimize inner product between similar classes
$L_{CO} \times L_{CR}^{\lambda} = L_F$
Fitness Function

Proposed: Automated Quantization Framework



Step 1: Candidate Initialization
L_1
L_{N-1}
$C_0 \left(b \ \gamma \ b \ \gamma \cdots b \ \gamma , \mathcal{L}_0^F \right)$
Λ_0
$C_{\mathcal{K}-1} \left(b \ \gamma \ b \ \gamma \cdots b \ \gamma , \mathcal{L}_{\mathcal{K}-1}^F \right)$
$\Lambda_{\mathcal{K}-1}$
Step 2: Re-generation
Crossover
Mutation
Step 3: Diversity Promoting Selection
Step 4: Evaluation
Eval. children on calib. dataset
Eval. L^F / candidate
Repeat N times per layer

Automated Quantization Framework: Step 1



Step 1: Candidate Initialization
L1
LN-1
C_0
$C_{\mathcal{K}-1}$
b
γ
$\mathcal{L}_0^F$
$\mathcal{L}_{\mathcal{K}-1}^F$
Λ_0
$\Lambda_{\mathcal{K}-1}$

Fitness value of each candidate

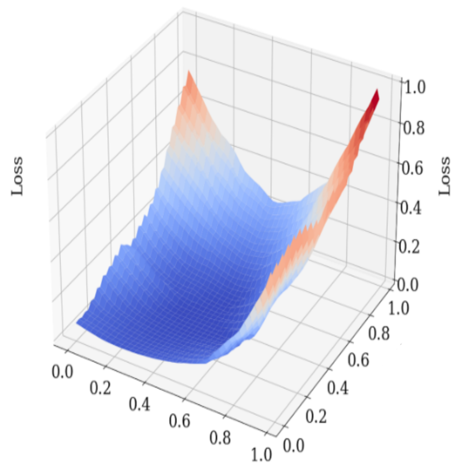
A quantization solution comprises an encoded vector Δ of length $2N$ and each set of 2 values represent the 2 integer quantization parameters of a layer l.

$$Q(X,\gamma,b) = clip(\left\lfloor \frac{X}{\gamma} \right\rceil, -2^{b-1}+1, 2^{b-1}-1)$$

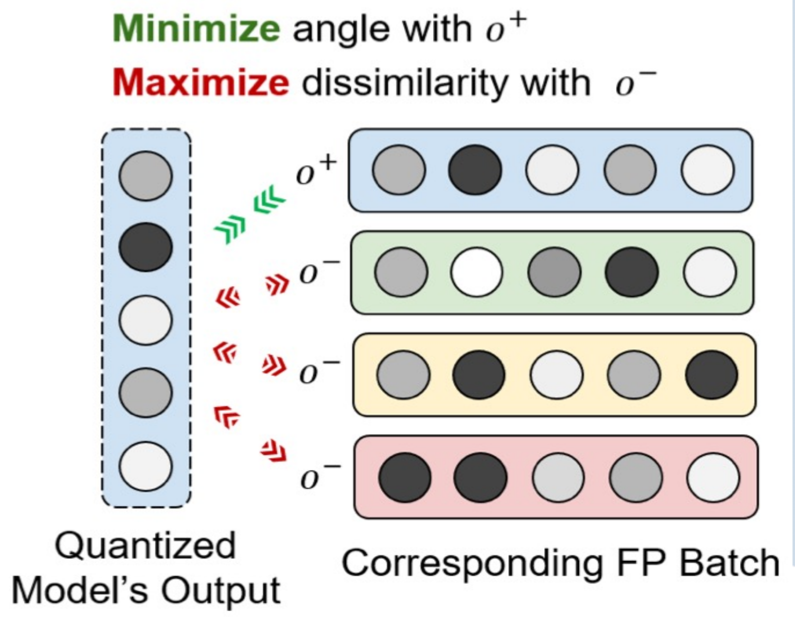
Automated Quantization Framework: Fitness Function Calculation

- Experiments in self-supervised learning and our own experiments suggest that contrastive learning tend to smooth the loss landscape.

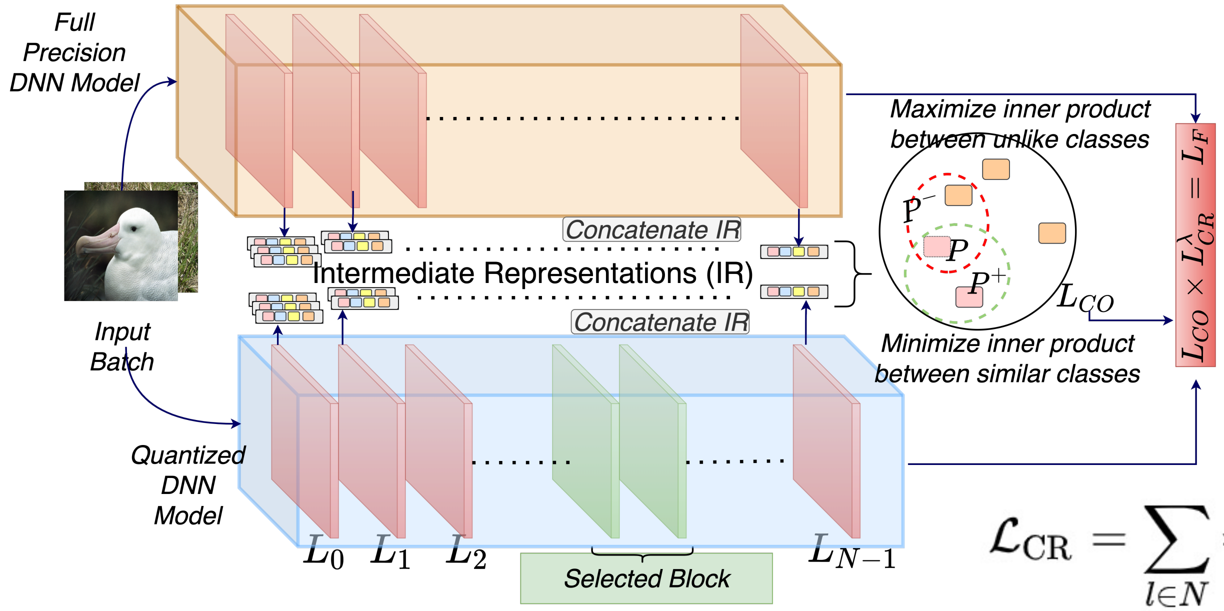
$$\mathcal{L}^{C}_{i,j} = -\log \frac{\sum_{p+} \exp(\lambda^{p}_{i,j} \cdot \lambda^{p+}_{i,j}/\tau)}{\sum_{p+} \exp(\lambda^{p}_{i,j} \cdot \lambda^{p+}_{i,j}/\tau) + \sum_{p-} \exp(\lambda^{p}_{i,j} \cdot \lambda^{p-}_{i,j}/\tau)}$$

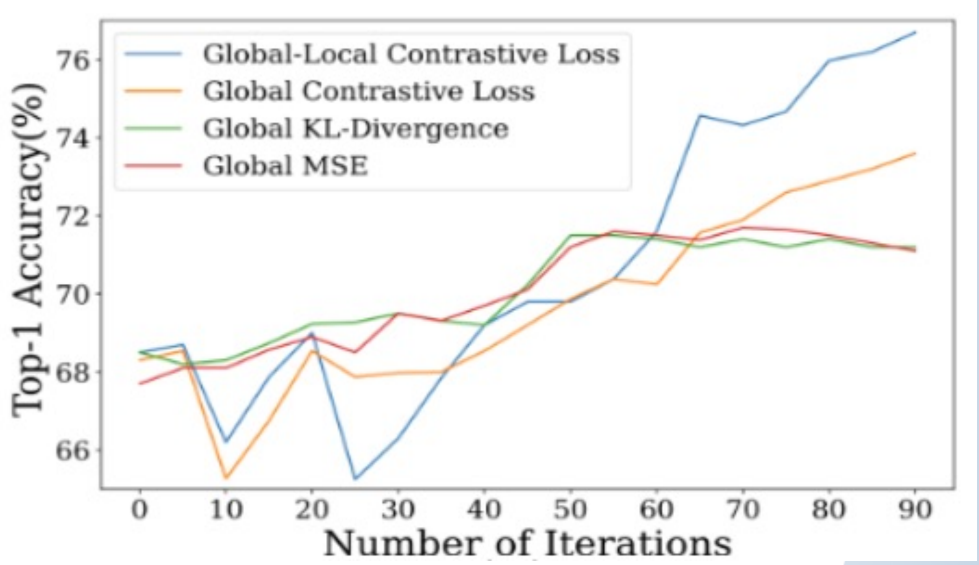


With Contrastive Learning


Minimize angle with o^+
Maximize dissimilarity with o^-
o^+
o^-
o^-
o^-
Quantized Model's Output
Corresponding FP Batch

Automated Quantization Framework: Fitness Function Calculation


Full Precision DNN Model
Input Batch
Quantized DNN Model
Concatenate IR
Intermediate Representations (IR)
Concatenate IR
Maximize inner product between unlike classes
P^-
P
P^+
L_{CO}
Minimize inner product between similar classes
$L_{CO} \times L_{CR}^{\lambda} = L_F$
L_0
L_1
L_2
L_{N-1}
Selected Block

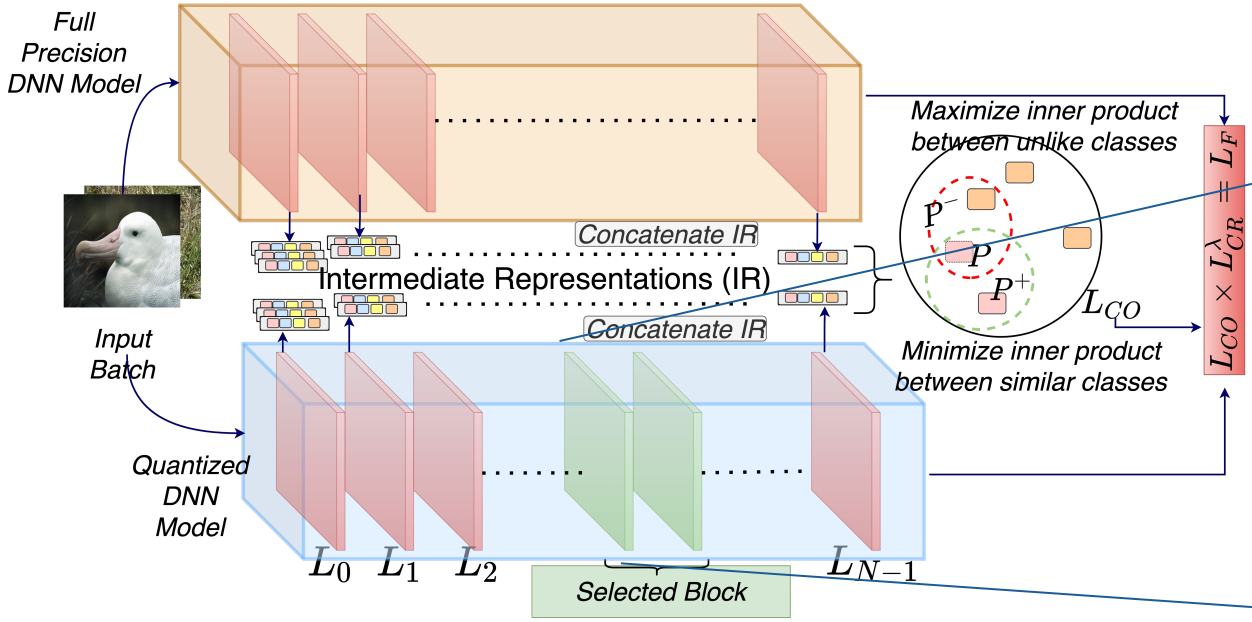


$$\mathcal{L}_{\mathrm{CR}} = \sum_{l \in N} \#\mathrm{PARAM}(\mathbf{H}_l^{\mathrm{FP}}) \times n_l$$

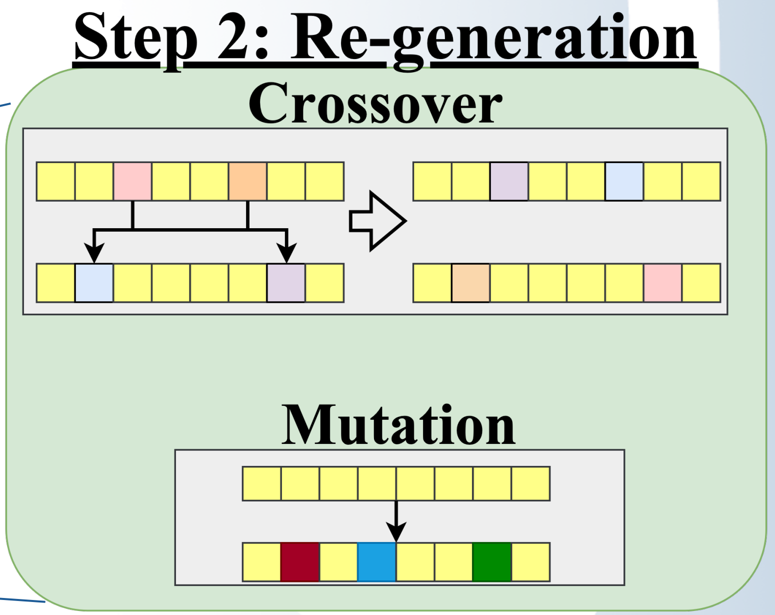
The contrastive component of fitness function aims to align the distribution of quantized model's intermediate representations closely with the FP model, while the compression ratio metric incentivizes lower bit-widths

SRC

Automated Quantization Framework: Step 2



Full Precision DNN Model
Input Batch
Quantized DNN Model
Concatenate IR
Intermediate Representations (IR)
Concatenate IR
Maximize inner product between unlike classes
Minimize inner product between similar classes
P^-
P
P^+
L_{CO}
$L_{CO} \times L_{CR}^{\lambda} + L_F$
L_0
L_1
L_2
L_{N-1}
Selected Block

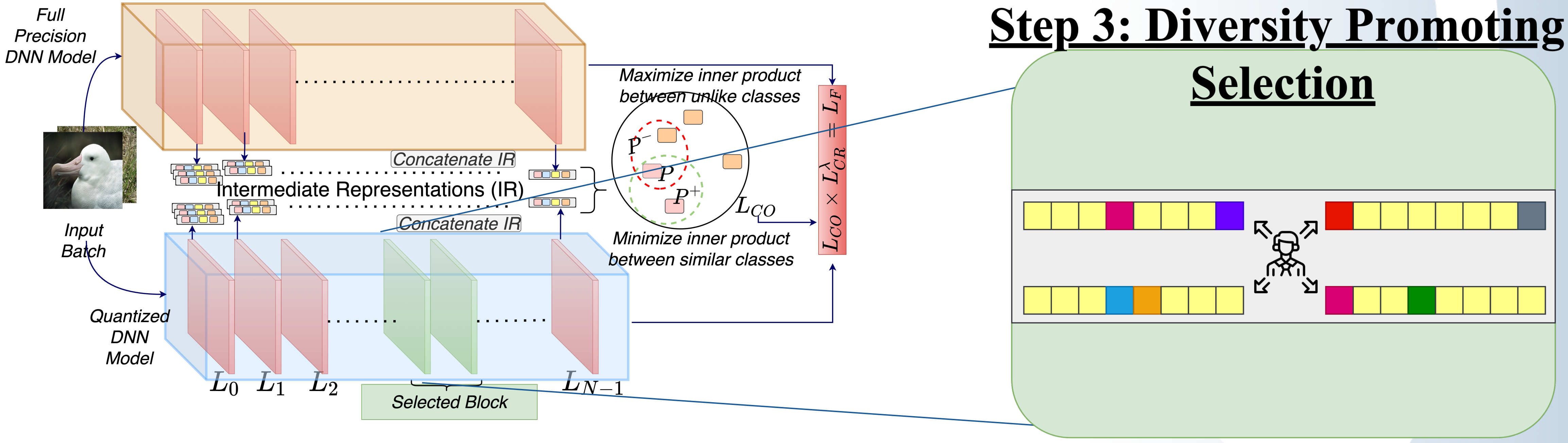


For each selected block perform regeneration and crossover.

$$b_{child} = \mathbf{random}(\mathbf{min}(b_{p1}, b_{p2}) - 1, \mathbf{max}(b_{p1}, b_{p2}) + 1)$$

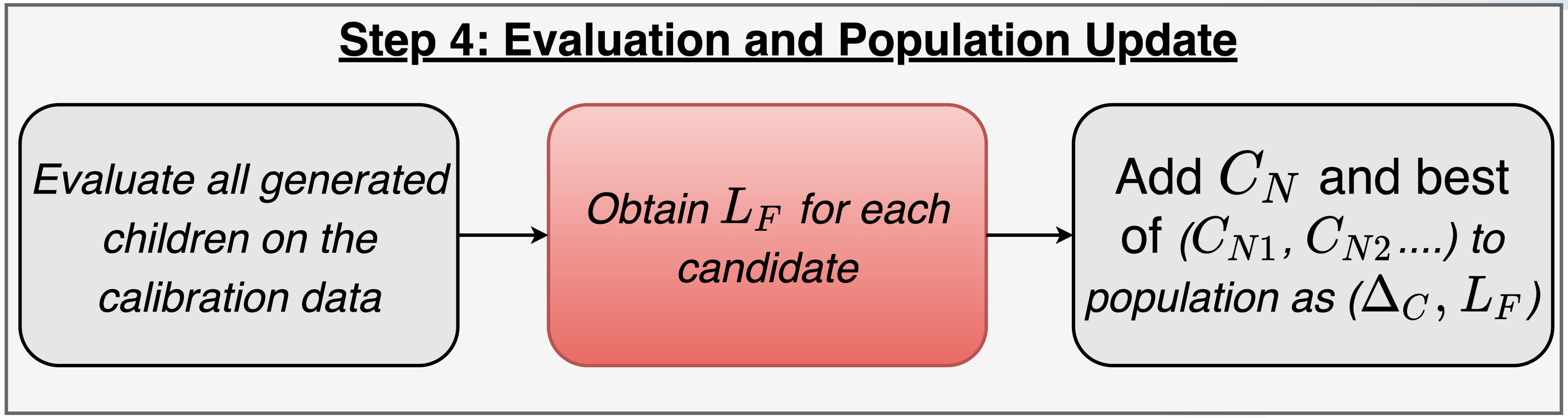
$$\gamma_{child} = \mathbf{mean}(\gamma_{p1}, \gamma_{p2}) + \eta(-10^{-3}, 10^{3})$$

Automated Quantization Framework: Step 3


Full Precision DNN Model
Input Batch
Quantized DNN Model
Concatenate IR
Intermediate Representations (IR)
Concatenate IR
Maximize inner product between unlike classes
Minimize inner product between similar classes
P^-
P
P^+
L_{CO}
$L_{CO} \times L_{CR}^{\lambda} = L_F$
L_0
L_1
L_2
L_{N-1}
Selected Block
Step 3: Diversity Promoting Selection

We create additional random parents and use the regenerated child in the previous stage as the other parent to generate five diverse children.

Automated Quantization Framework: Step 4


Step 4: Evaluation and Population Update
Evaluate all generated children on the calibration data
Obtain L_F for each candidate
Add C_N and best of $(C_{N1}, C_{N2} \ldots.)$ *to population as* (Δ_C, L_F)

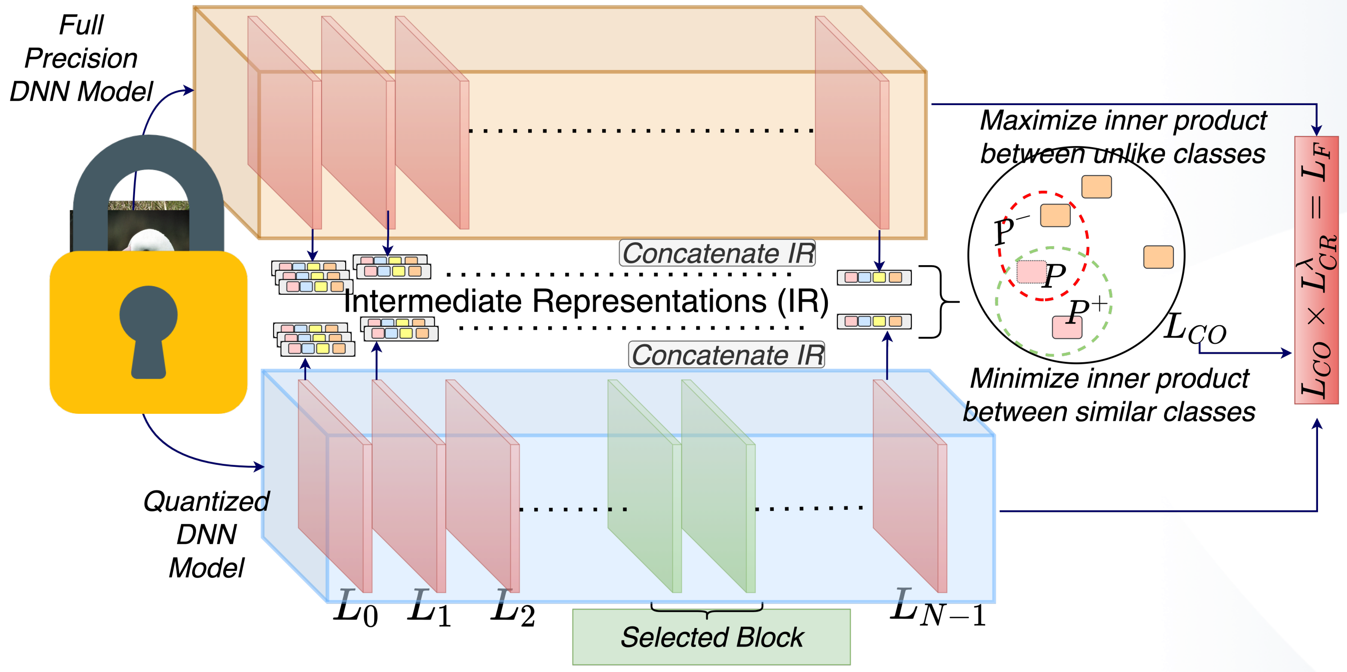
Activation Quantization

Activation quantization sensitivity closely aligns with that of the weight parameters producing them.

The output activation quantization parameters for layer i are determined as,

$$b_{act}[i] = \mathbf{min}(8, b[i] \times 2) \text{ and } \gamma_{act}[i] = \gamma_{act}[i-1] + \gamma[i].$$

Challenges with Current Quantization Scheme


Full Precision DNN Model
Concatenate IR
Intermediate Representations (IR)
Concatenate IR
Maximize inner product between unlike classes
P^-
P
P^+
L_{CO}
Minimize inner product between similar classes
$L_{CO} \times L_{CR}^{\lambda} = L_F$
Quantized DNN Model
L_0
L_1
L_2
Selected Block
L_{N-1}

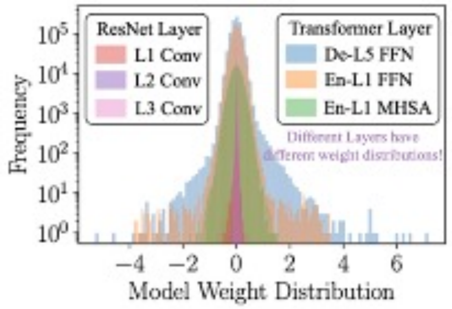
- Current PTQ method requires access to calibration data drawn from the training set.
- However, due to privacy and security concerns the data might not always be accessible.

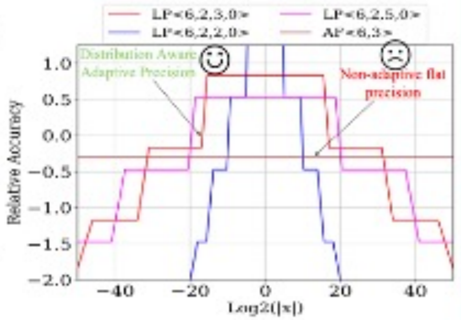
Outline

Algorithm

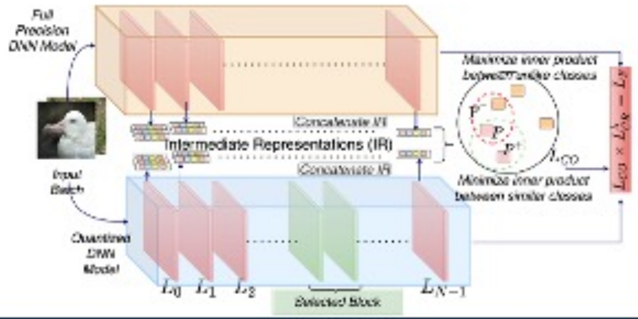
Under Submission

Introduction





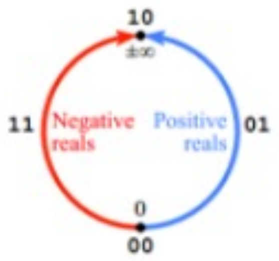
Automated Quantization


Full Precision DNN Model
Input Batch
Quantized DNN Model
Intermediate Representations (IR)
Concatenate IR
Maximize inner product between unlike classes
Minimize inner product between similar classes
Selected Block

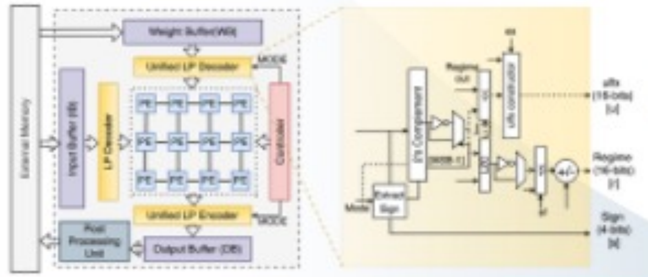
Data-Free Quantization

Spatial Location of Selected Anchor Patch
Select Positive and Negative Patches in Neighborhood

Next Generation Arithmetic: Logarithmic Posits

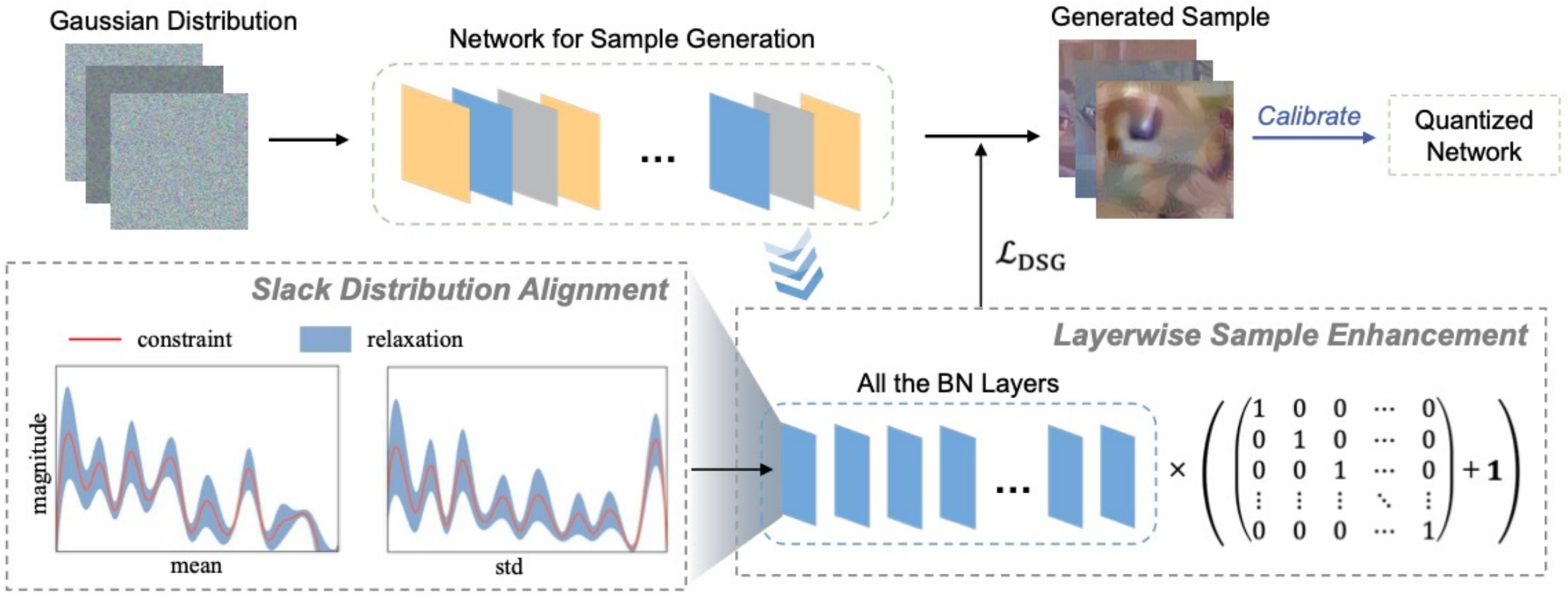


Hardware: Logarithmic Posit Accelerator



Hardware

Prior Art: Data-Free Quantization for CNNs

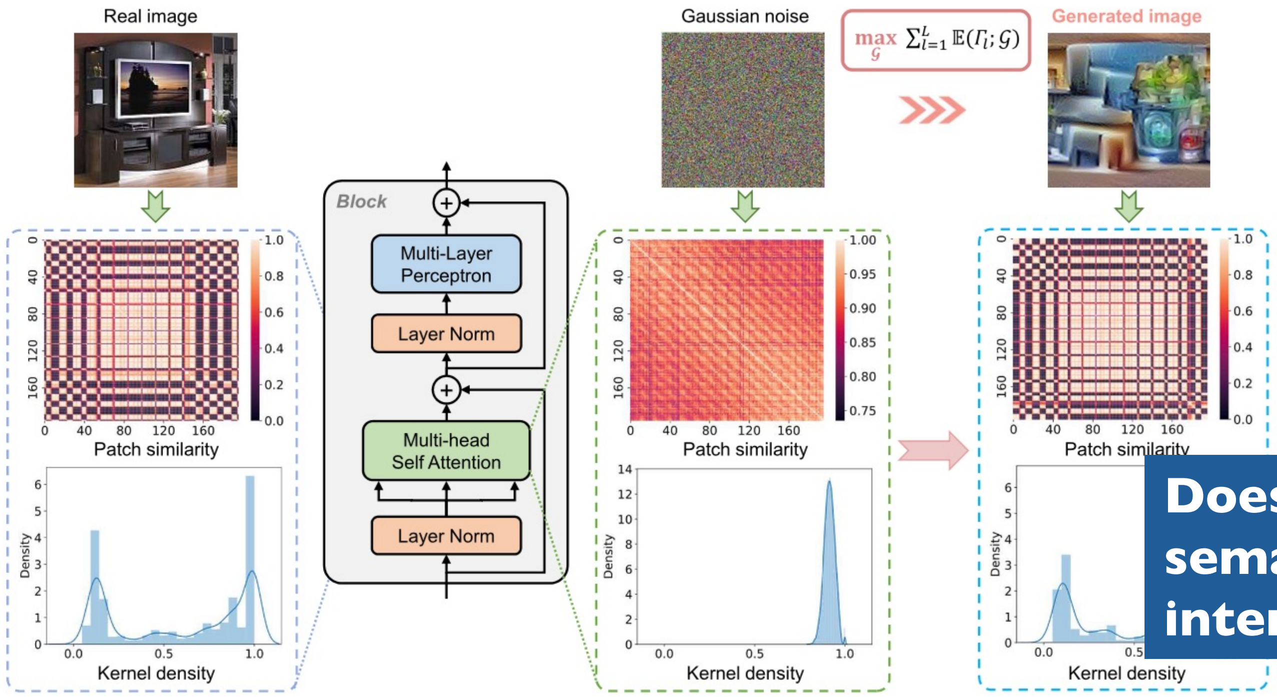

Gaussian Distribution
Network for Sample Generation
...
Generated Sample
Calibrate
Quantized Network
$\mathcal{L}_{DSG}$
Slack Distribution Alignment
constraint
relaxation
magnitude
mean
std
Layerwise Sample Enhancement
All the BN Layers
$\times \left(\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} + \mathbf{1} \right)$

- In CNNs, data-free quantization techniques synthesize data for calibrating the quantized model according to the batch normalization (BN) statistics of FP32.
- This is a reliable and efficient method for distilling data from the FP32 model for CNNs.

SRC

Source: Zhang, Xiangguo, et al. "Diversifying sample generation for accurate data-free quantization." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021..

Data-Free Quantization: Transformer-Based models (ViTs)

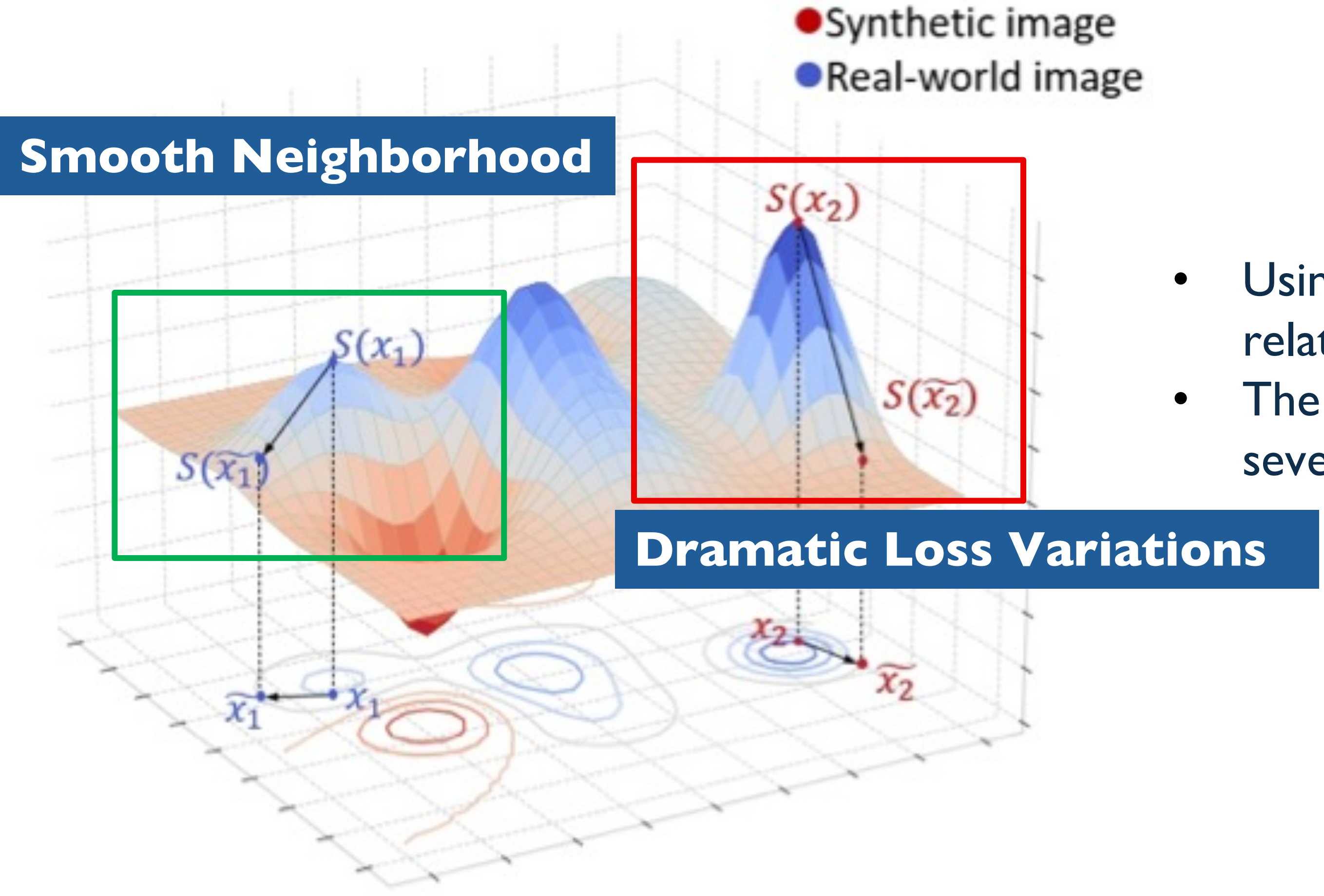


Does not explore semantically meaningful interpatch relations!

- ViTs or transformer-based models do not have a BN layer or any layer that holds statistics of the data it has been trained on.
- Previous methods rely on maximizing the entropy of the self-attention layer outputs.
- This ignores inter-patch relations and generates semantically non-informative and less realistic data.

Source: Li, Zhikai, et al. "Psaq-vit v2: Toward accurate and general data-free quantization for vision transformers." IEEE Transactions on Neural Networks and Learning Systems (2023).

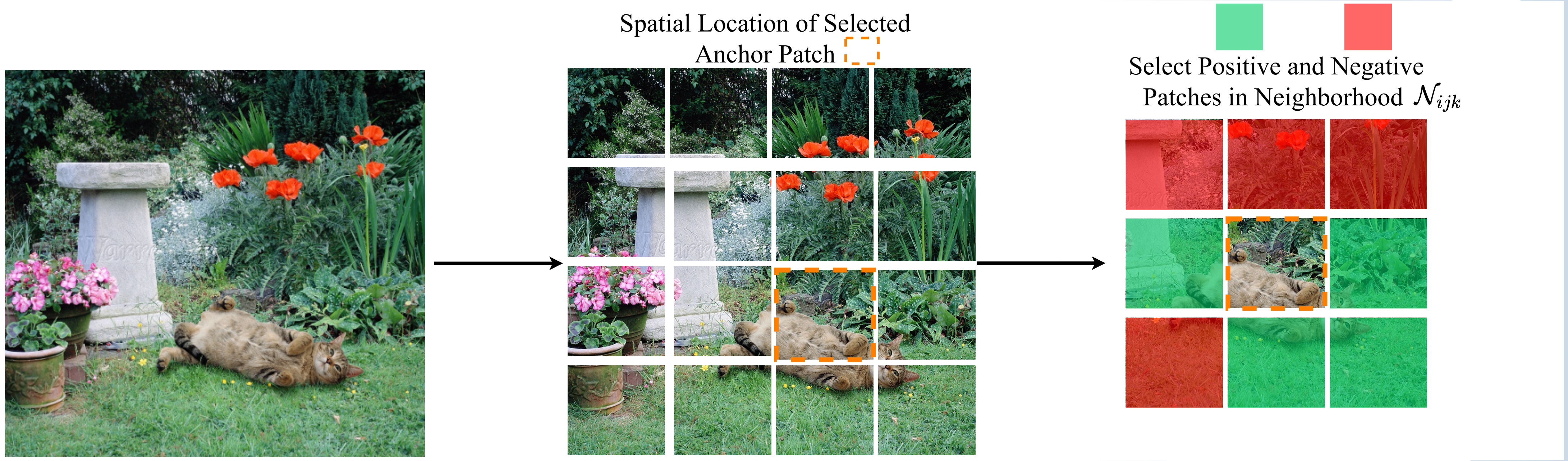
Data-Free Quantization



- Using a global scheme without considering inter-patch relations does not result in realistic and robust images.
- The lower semantic content in generated synthetic images severely impacts quantized model generalizability.

Source: Li, Zhikai, et al. "Psaq-vit v2: Toward accurate and general data-free quantization for vision transformers." IEEE Transactions on Neural Networks and Learning Systems (2023).

Data-Free Quantization: Proposed Approach



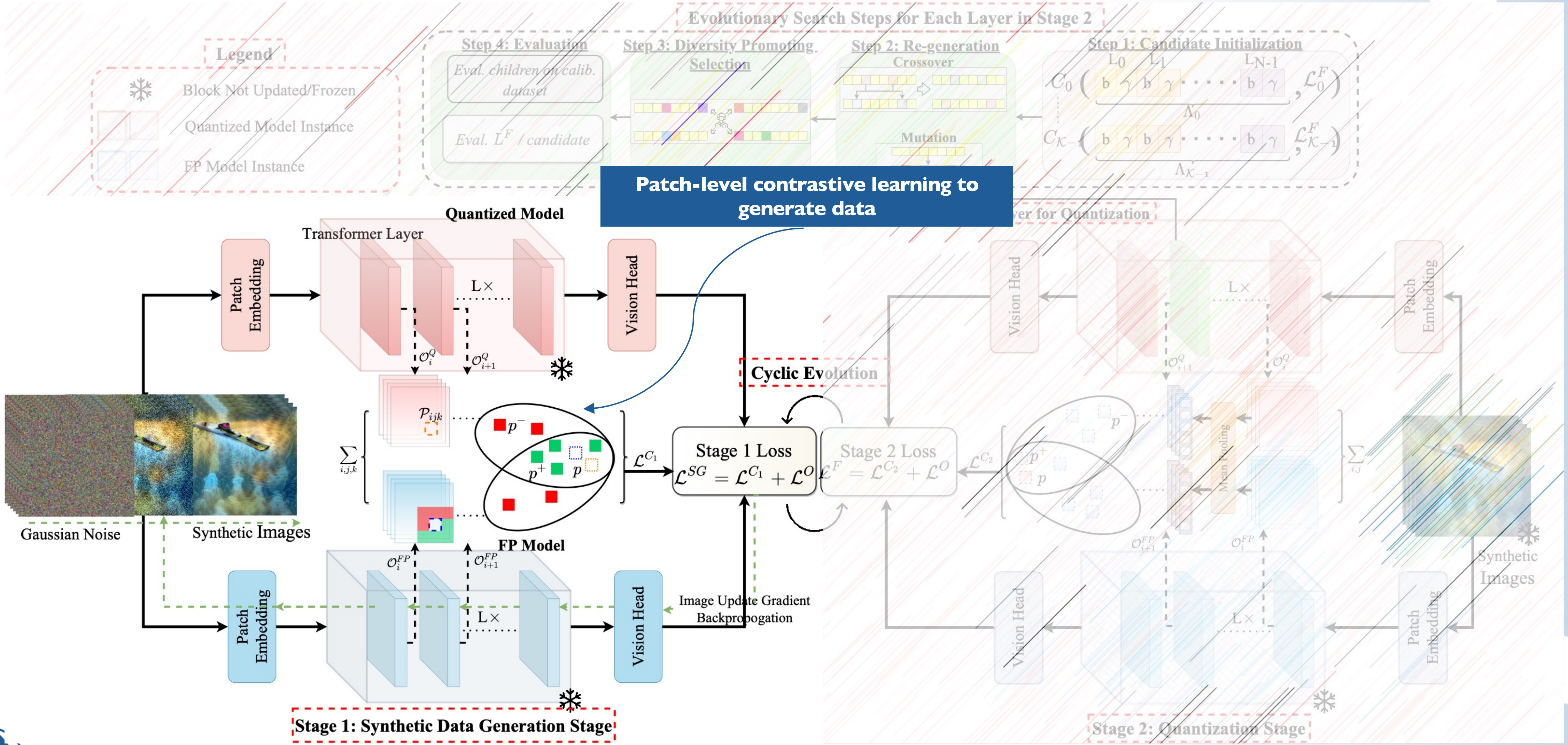
Spatial Location of Selected Anchor Patch
Select Positive and Negative Patches in Neighborhood $\mathcal{N}_{ijk}$

- We propose a data-generation scheme that leverages the architectural characteristics of ViTs i.e, patch-level attention and the inherent properties of real-images to generate semantically rich and meaningful data.

- We develop a contrastive learning scheme that treats semantically similar patches in a neighborhood as positive and others as negative.

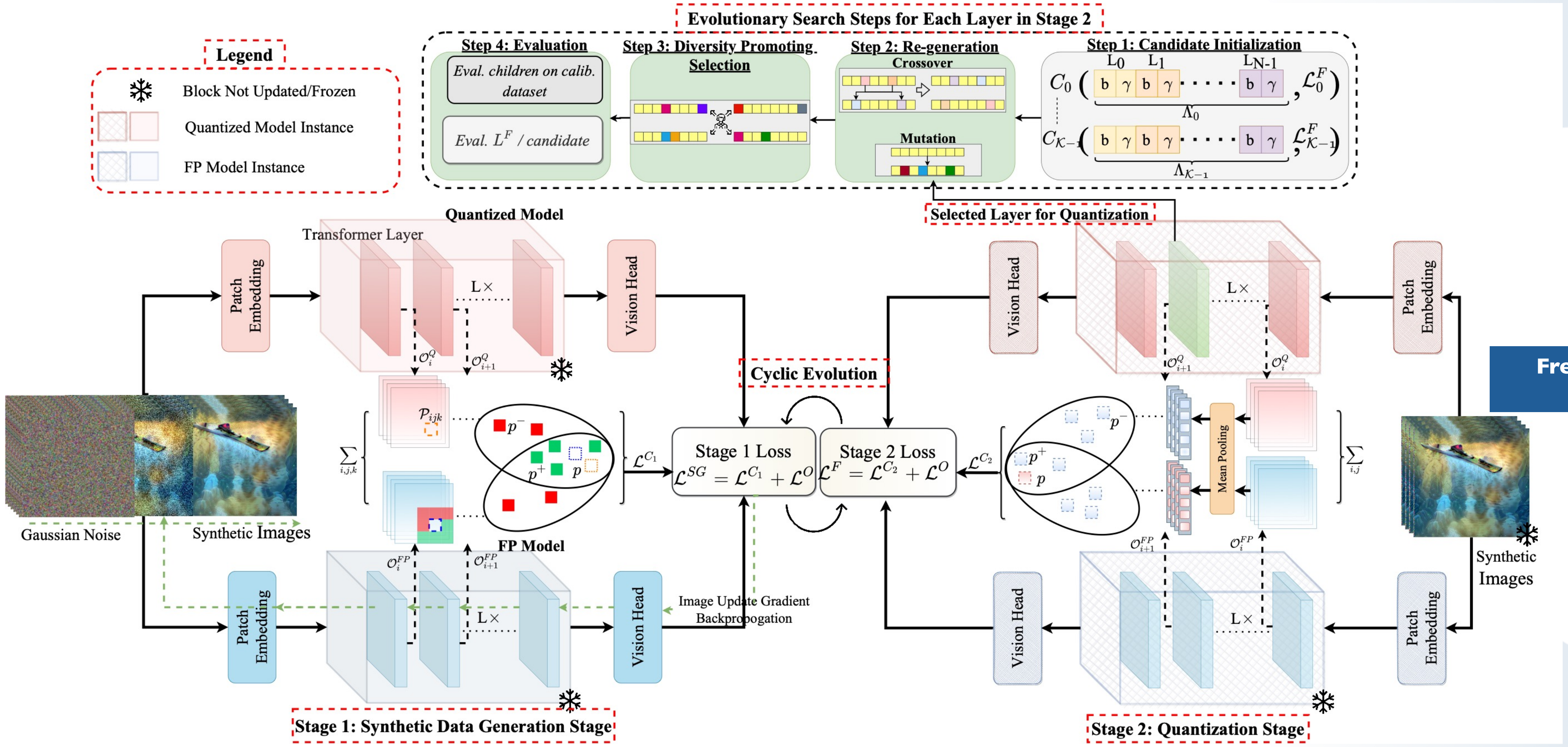
SRC

Data-Free Quantization: Complete Pipeline


Evolutionary Search Steps for Each Layer in Stage 2
Legend
Block Not Updated/Frozen
Quantized Model Instance
FP Model Instance
Step 4: Evaluation
Eval. children on calib. dataset
Eval. L^F / candidate
Step 3: Diversity Promoting Selection
Step 2: Re-generation
Crossover
Mutation
Step 1: Candidate Initialization
Quantized Model
Transformer Layer
Patch Embedding
Vision Head
L ×
Cyclic Evolution
Gaussian Noise
Synthetic Images
Stage 1 Loss
$\mathcal{L}^{SG} = \mathcal{L}^{C_1} + \mathcal{L}^{O}$
Stage 2 Loss
$\mathcal{L}^{F} = \mathcal{L}^{C_2} + \mathcal{L}^{O}$
FP Model
Image Update Gradient Backpropogation
Mean Pooling
Stage 1: Synthetic Data Generation Stage
Stage 2: Quantization Stage

Patch-level contrastive learning to generate data

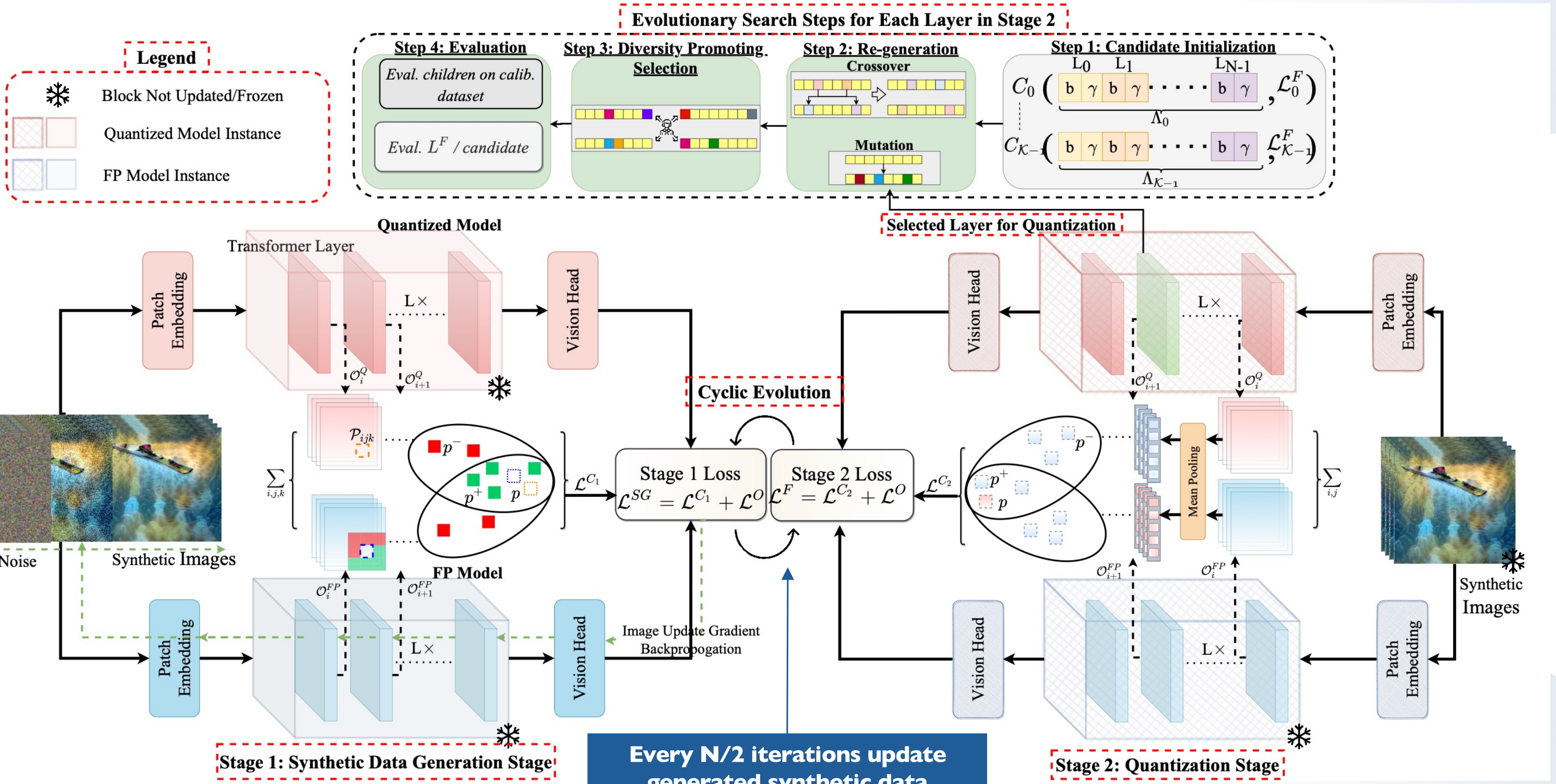
Data-Free Quantization: Complete Pipeline



Evolutionary Search Steps for Each Layer in Stage 2
Step 4: Evaluation
Eval. children on calib. dataset
Eval. L^F / candidate
Step 3: Diversity Promoting Selection
Step 2: Re-generation
Crossover
Mutation
Step 1: Candidate Initialization
Legend
Block Not Updated/Frozen
Quantized Model Instance
FP Model Instance
Selected Layer for Quantization
Quantized Model
Transformer Layer
Patch Embedding
Vision Head
L×
Cyclic Evolution
Stage 1 Loss
$\mathcal{L}^{SG} = \mathcal{L}^{C_1} + \mathcal{L}^{O}$
Stage 2 Loss
$\mathcal{L}^{F} = \mathcal{L}^{C_2} + \mathcal{L}^{O}$
Mean Pooling
Gaussian Noise
Synthetic Images
FP Model
Image Update Gradient Backpropogation
Stage 1: Synthetic Data Generation Stage
Stage 2: Quantization Stage

Freeze Synthetic Images in Stage 2

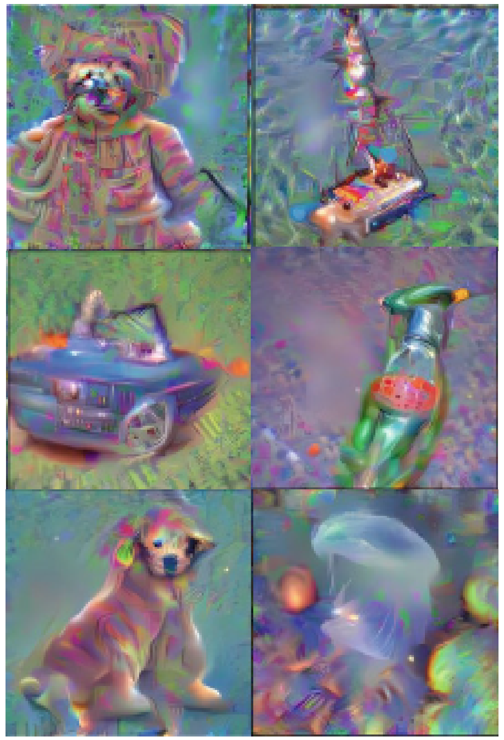
Data-Free Quantization: Complete Pipeline



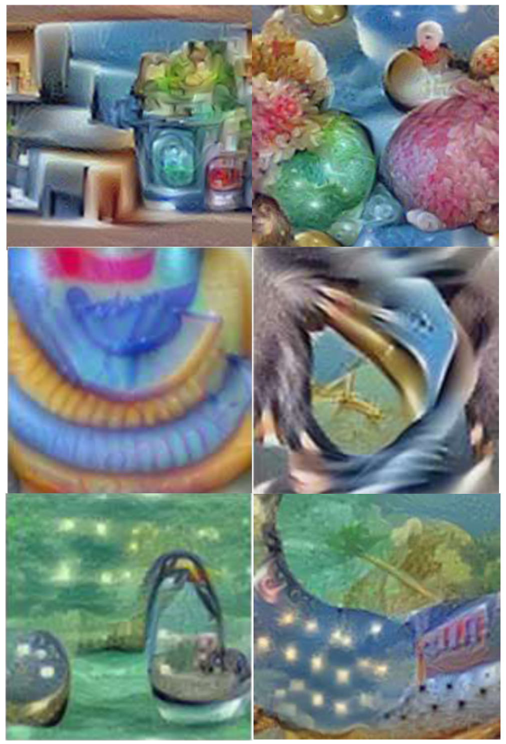
Evolutionary Search Steps for Each Layer in Stage 2
Step 4: Evaluation
Eval. children on calib. dataset
Eval. L^F / candidate
Step 3: Diversity Promoting Selection
Step 2: Re-generation
Crossover
Mutation
Step 1: Candidate Initialization
C_0
C_{K-1}
$\mathcal{L}_0^F$
$\mathcal{L}_{K-1}^F$
Λ_0
Λ_{K-1}
Legend
Block Not Updated/Frozen
Quantized Model Instance
FP Model Instance
Quantized Model
Transformer Layer
Patch Embedding
Vision Head
L×
Selected Layer for Quantization
Cyclic Evolution
Stage 1 Loss
$\mathcal{L}^{SG} = \mathcal{L}^{C_1} + \mathcal{L}^O$
Stage 2 Loss
$\mathcal{L}^F = \mathcal{L}^{C_2} + \mathcal{L}^O$
Mean Pooling
Gaussian Noise
Synthetic Images
FP Model
Image Update Gradient Backpropogation
Stage 1: Synthetic Data Generation Stage
Stage 2: Quantization Stage

Every N/2 iterations update generated synthetic data

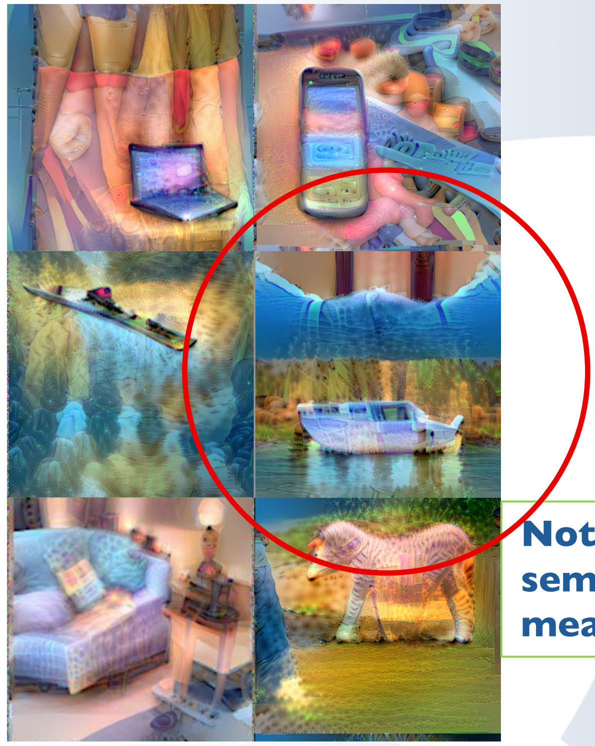
Data-Free Quantization: Generated Samples



PSAQ-ViT v1



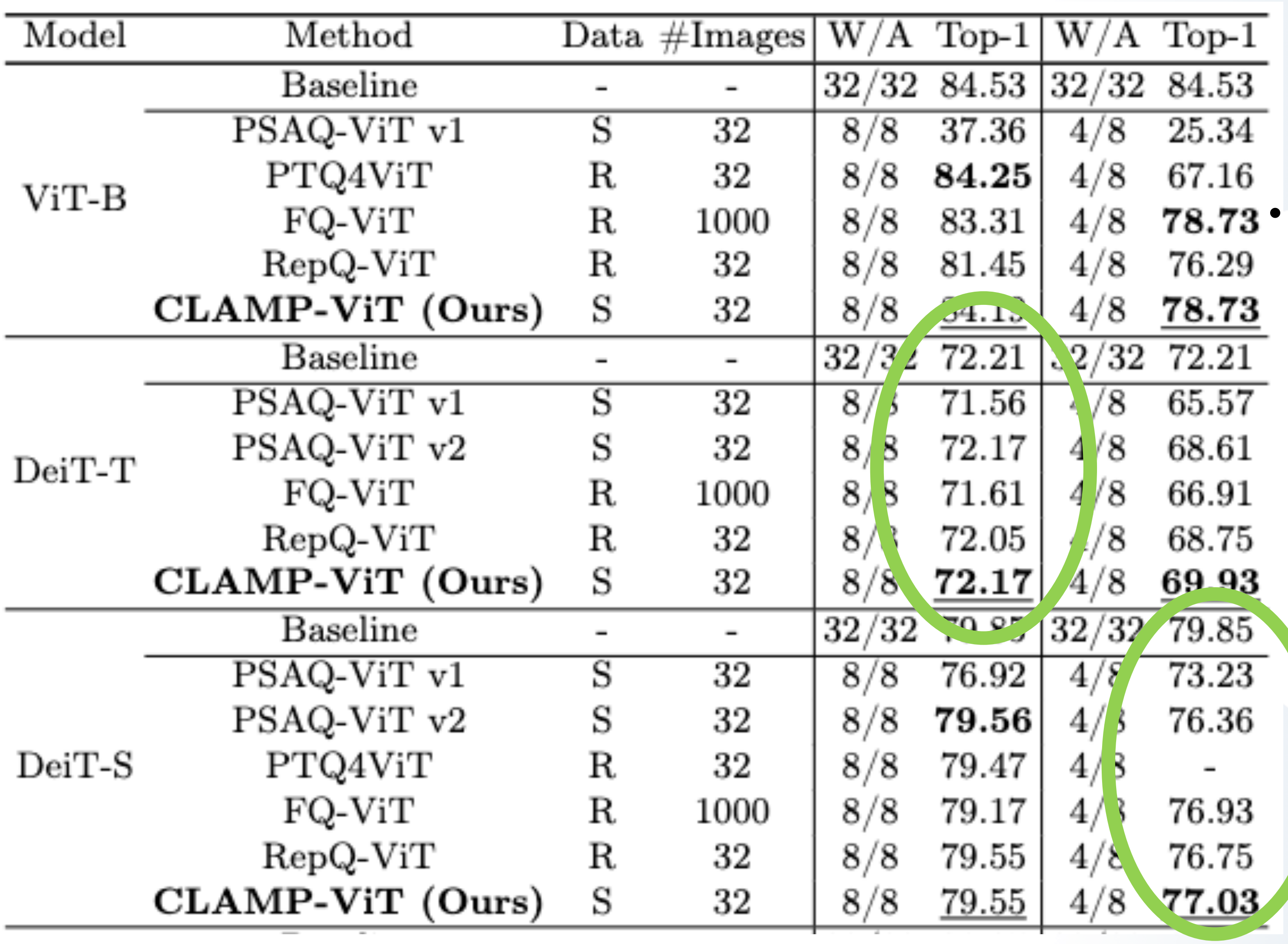
PSAQ-ViT v2



Ours

Notice the semantic meaningfulness!

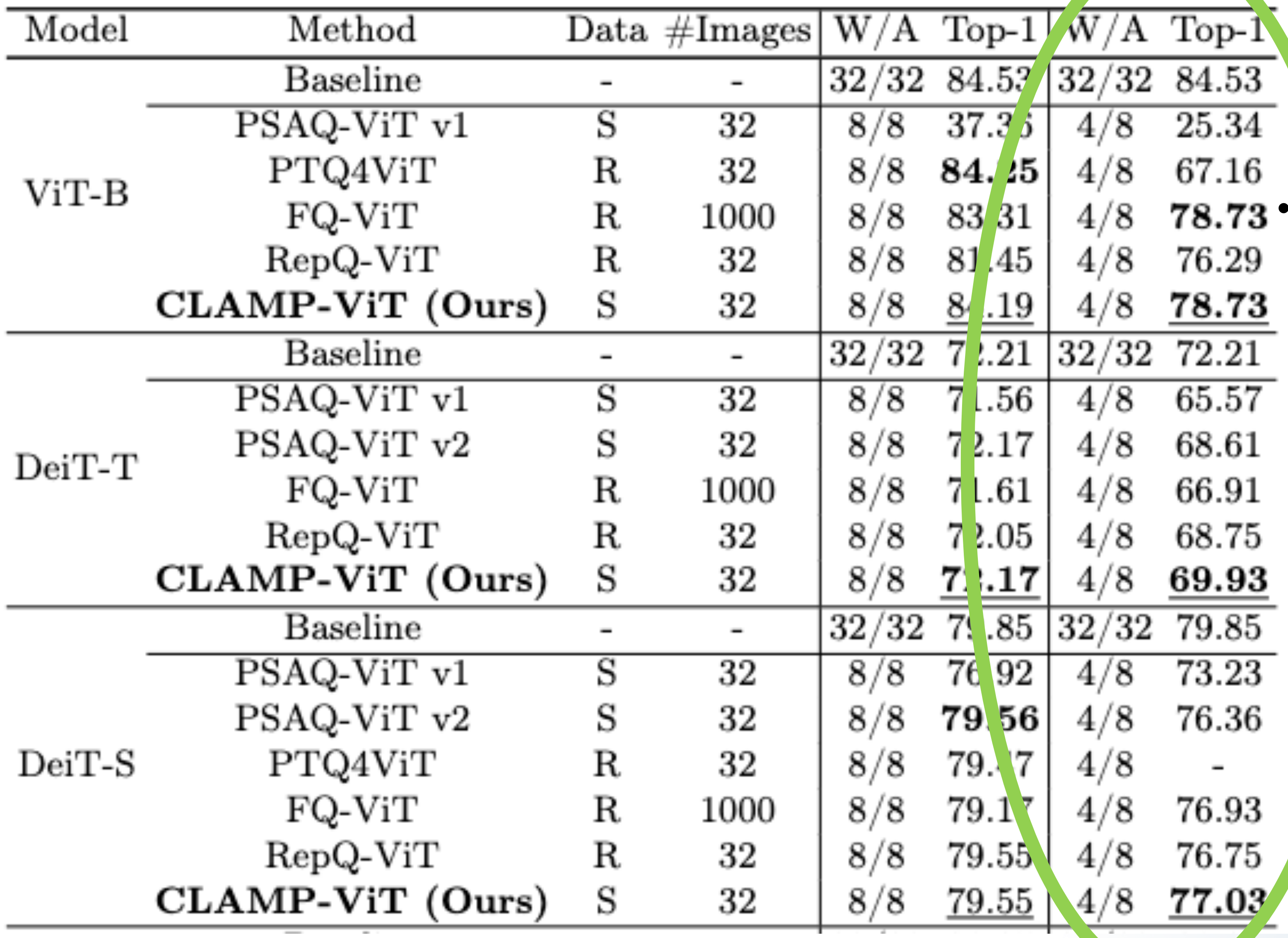
Data-Free Quantization Performance



Model	Method	Data	#Images	W/A	Top-1	W/A	Top-1
ViT-B	Baseline	-	-	32/32	84.53	32/32	84.53
	PSAQ-ViT v1	S	32	8/8	37.36	4/8	25.34
	PTQ4ViT	R	32	8/8	**84.25**	4/8	67.16
	FQ-ViT	R	1000	8/8	83.31	4/8	**78.73**
	RepQ-ViT	R	32	8/8	81.45	4/8	76.29
	CLAMP-ViT (Ours)	S	32	8/8	84.13	4/8	**78.73**
DeiT-T	Baseline	-	-	32/32	72.21	32/32	72.21
	PSAQ-ViT v1	S	32	8/8	71.56	4/8	65.57
	PSAQ-ViT v2	S	32	8/8	72.17	4/8	68.61
	FQ-ViT	R	1000	8/8	71.61	4/8	66.91
	RepQ-ViT	R	32	8/8	72.05	4/8	68.75
	CLAMP-ViT (Ours)	S	32	8/8	**72.17**	4/8	**69.93**
DeiT-S	Baseline	-	-	32/32	79.85	32/32	79.85
	PSAQ-ViT v1	S	32	8/8	76.92	4/8	73.23
	PSAQ-ViT v2	S	32	8/8	**79.56**	4/8	76.36
	PTQ4ViT	R	32	8/8	79.47	4/8	-
	FQ-ViT	R	1000	8/8	79.17	4/8	76.93
	RepQ-ViT	R	32	8/8	79.55	4/8	76.75
	CLAMP-ViT (Ours)	S	32	8/8	79.55	4/8	**77.03**

- **~2.2% Improvement over DFQ and ~1% Improvement over PTQ methods!**

Data-Free Quantization Performance



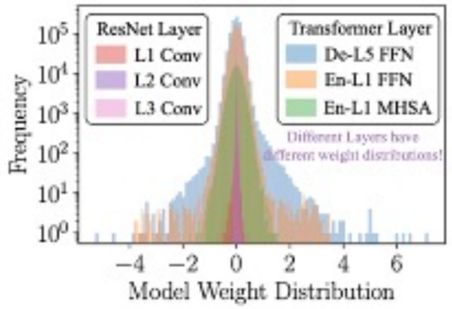
Model	Method	Data	#Images	W/A	Top-1	W/A	Top-1
ViT-B	Baseline	-	-	32/32	84.53	32/32	84.53
	PSAQ-ViT v1	S	32	8/8	37.36	4/8	25.34
	PTQ4ViT	R	32	8/8	**84.25**	4/8	67.16
	FQ-ViT	R	1000	8/8	83.31	4/8	**78.73**
	RepQ-ViT	R	32	8/8	81.45	4/8	76.29
	CLAMP-ViT (Ours)	S	32	8/8	84.19	4/8	**78.73**
DeiT-T	Baseline	-	-	32/32	72.21	32/32	72.21
	PSAQ-ViT v1	S	32	8/8	71.56	4/8	65.57
	PSAQ-ViT v2	S	32	8/8	72.17	4/8	68.61
	FQ-ViT	R	1000	8/8	71.61	4/8	66.91
	RepQ-ViT	R	32	8/8	72.05	4/8	68.75
	CLAMP-ViT (Ours)	S	32	8/8	**72.17**	4/8	**69.93**
DeiT-S	Baseline	-	-	32/32	79.85	32/32	79.85
	PSAQ-ViT v1	S	32	8/8	76.92	4/8	73.23
	PSAQ-ViT v2	S	32	8/8	**79.56**	4/8	76.36
	PTQ4ViT	R	32	8/8	79.47	4/8	-
	FQ-ViT	R	1000	8/8	79.17	4/8	76.93
	RepQ-ViT	R	32	8/8	79.55	4/8	76.75
	CLAMP-ViT (Ours)	S	32	8/8	79.55	4/8	**77.03**

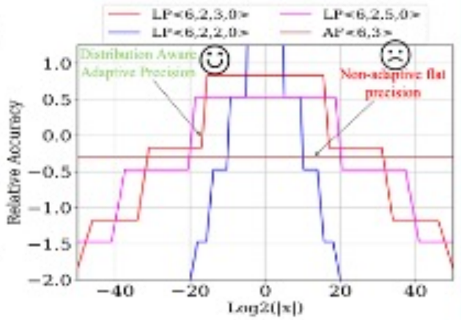
- **For W4/A8 our method shows significant performance boost over all the existing alternatives despite be**

Outline

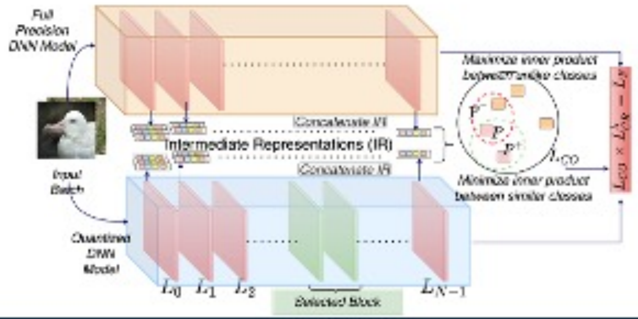
Algorithm

Introduction





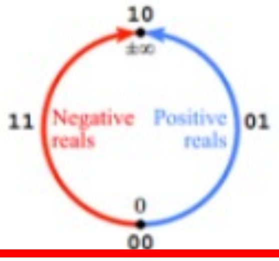
Automated Quantization


Full Precision DNN Model
Input Batch
Intermediate Representations (IR)
Quantized DNN Model
Selected Block
Maximize inner product between unlike classes
Minimize inner product between similar classes

Data-Free Quantization

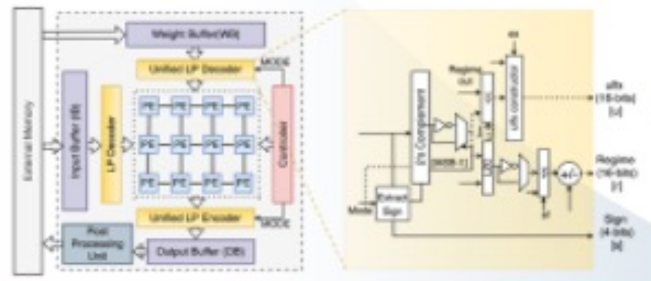
Spatial Location of Selected Anchor Patch
Select Positive and Negative Patches in Neighborhood

Next Generation Arithmetic: Logarithmic Posits



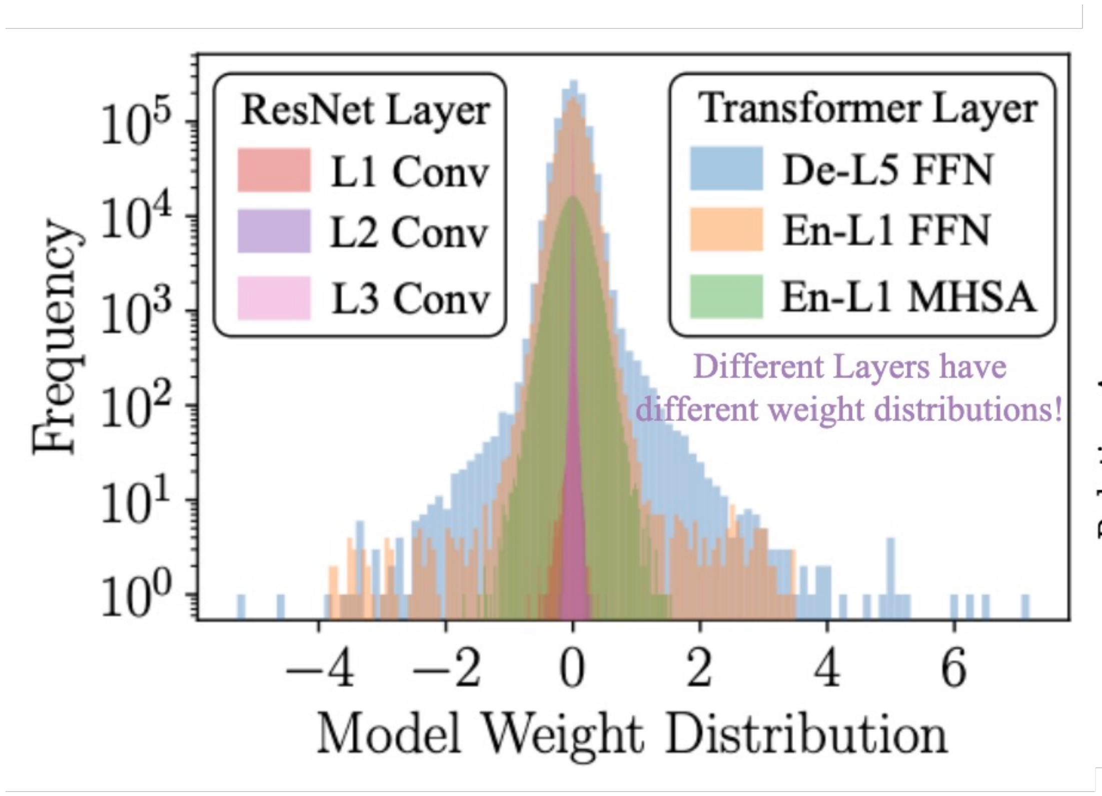
To be presented at DAC'24

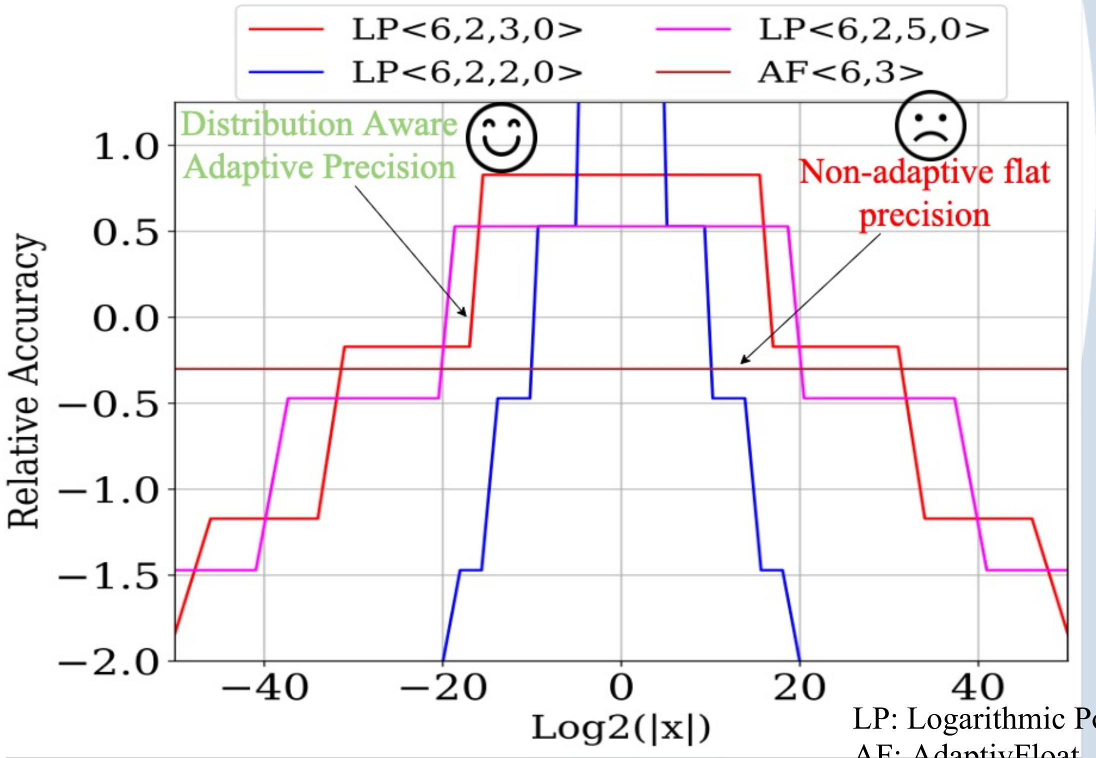
Hardware: Logarithmic Posit Accelerator



Hardware

Challenges with Traditional Quantization

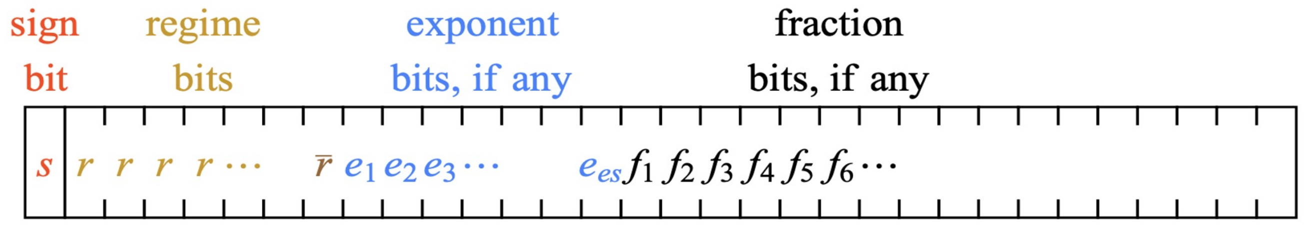




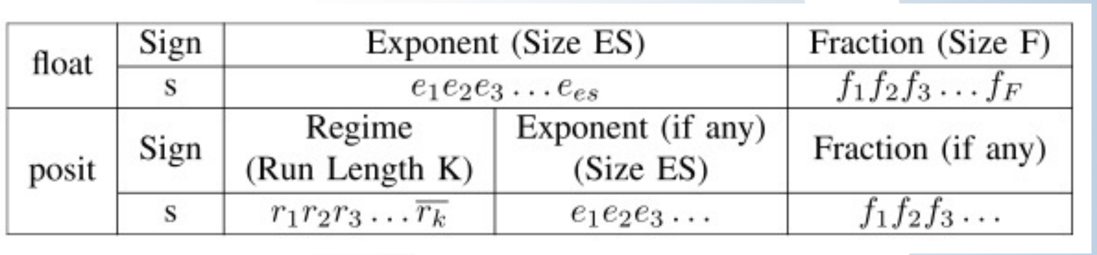
LP: Logarithmic Posit
AF: AdaptivFloat

- Uniform Quantization techniques lack the dynamic range and distributional variance required of DNNs.
- Floating-point based non-uniform quantization techniques fail to adapt to the different DNN distributions and have flat accuracy.
- Vector Quantization is adaptive but introduces additional codebook overhead.

Next Generation Arithmetic: Posits


sign bit
regime bits
exponent bits, if any
fraction bits, if any
s
$r\ r\ r\ r\cdots\ \bar{r}$
$e_1\ e_2\ e_3\cdots\ e_{es}$
$f_1\ f_2\ f_3\ f_4\ f_5\ f_6\cdots$

Generic n-bits Posit format



float	Sign	Exponent (Size ES)		Fraction (Size F)
	s	$e_1e_2e_3\dots e_{es}$		$f_1f_2f_3\dots f_F$
posit	Sign	Regime (Run Length K)	Exponent (if any) (Size ES)	Fraction (if any)
	s	$r_1r_2r_3\dots\overline{r_k}$	$e_1e_2e_3\dots$	$f_1f_2f_3\dots$

$$\texttt{posit val} = (-1)^{sign} * (2^{2^{es}})^k * 2^{expo} * 1.frac$$

- ➢ Possesses the unique field called regime that dynamically varies between [2, n-1] bits. It dynamically varies the exponent and fraction fields to give tapered accuracy and varied dynamic ranges.
- ➢ The regime is a run-length encoding of m 0s (1s) terminated by a 1(0), respectively, or by the final bit. The regime value k is determined as $k = -m$ if the first bit of the regime is 0, or $k = m-1$ otherwise.

SRC

Logarithmic Posits

$$\boxed{x\langle n, es, rs, sf\rangle = (-1)^{sign} \times 2^{2^{es} \times k - sf} \times 2^{\mathbf{ulfx}}}$$

LP value decoding

- A composite datatype that blends the adaptability of Posits with the hardware-efficiency of Logarithmic Number System (LNS).

- Parameterizations introduced for adaptability:
- **Bits (n):** Identify optimal precision for a DNN layer.
- **Exponent Size (es):** Controls dynamic range.
- **Regime Size (rs):** Controls distribution shape.
- **Scale Factor (sf):** Adjusts distribution position.

- Express standard fraction and exponent in the logarithmic domain as a unified fixed-point exponent of the power of two as 2^{ulfx}, where ulfx=e+f.

SRC

Floating-Point v. Posit v. Logarithmic Posit Decoding

Binary Number: 01001110

Floating-Point (E4M3), Bias = 7

0_1001_110

Sign: +

E-b: *1001 - 0111* = 9 - 7 = 2

M: *1.110* = 1.75

Value = +1 * 1.75 * 2^2

Posit (ES = 2)

0_10_01_110

Sign: +

Regime: *10* = 0

E: *01* = 1

M: 1.110 = 1.75

$$\textbf{Value} = +1 * (2^{(2^2)})^0 * 2^1 * 1.75$$

Logarithmic Posit (ES = 1, RS = 7, SF = 0

0_10_0_1110

Sign: +

Regime: *10* = 0

ulfx: *0.1110* = 0.875

$$\textbf{Value} = +1 * (2^{(2^2)})^0 * 2^0 * 2^{0.875}$$

SRC

Floating-Point v. Posit v. Logarithmic Posit Decoding

Binary Number: 01001110

Floating-Point (E4M3), Bias = 7

0_1001_110

Sign: +

E-b: *1001 - 0111* = 9 - 7 = 2

M: *1.110* = 1.75

Value = $+1 * 1.75 * 2^2$

Posit (ES = 2)

0_10_01_110

Sign: +

Regime: *10* = 0

E: *01* = 1

M: 1.110 = 1.75

$$\textbf{Value} = +1 * (2^{(2^2)})^0 * 2^1 * 1.75$$

Logarithmic Posit (ES = 1, RS = 7, SF = 0

0_10_0_1110

Sign: +

Regime: *10* = 0

ulfx: *0.1110* = 0.875

$$\textbf{Value} = +1 * (2^{(2^2)})^0 * 2^0 * 2^{0.875}$$

SRC

Floating-Point v. Posit v. Logarithmic Posit Decoding

Binary Number: 01001110

Floating-Point (E4M3), Bias = 7

0_1001_110

Sign: +

E-b: *1001 - 0111* = 9 - 7 = 2

M: *1.110* = 1.75

Value = $+1 * 1.75 * 2^2$

Posit (ES = 2)

0_10_01_110

Sign: +

Regime: *10* = 0

E: *01* = 1

M: 1.110 = 1.75

$$\textbf{Value} = +1 * (2^{(2^2)})^0 * 2^1 * 1.75$$

Logarithmic Posit (ES = 1, RS = 7, SF = 0

0_10_0_1110

Sign: +

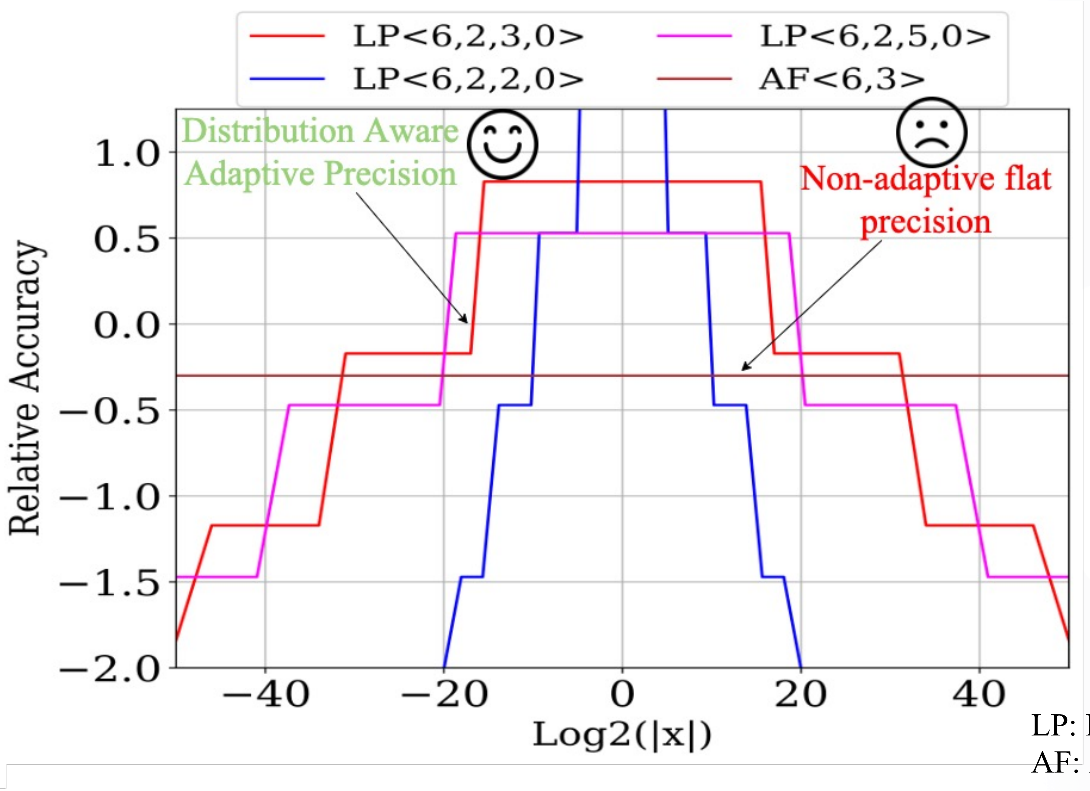
Regime: *10* = 0

ulfx: *0.1110* = 0.875

$$\textbf{Value} = +1 * (2^{(2^2)})^0 * 2^0 * 2^{0.875}$$

SRC

Advantages of Logarithmic Posits

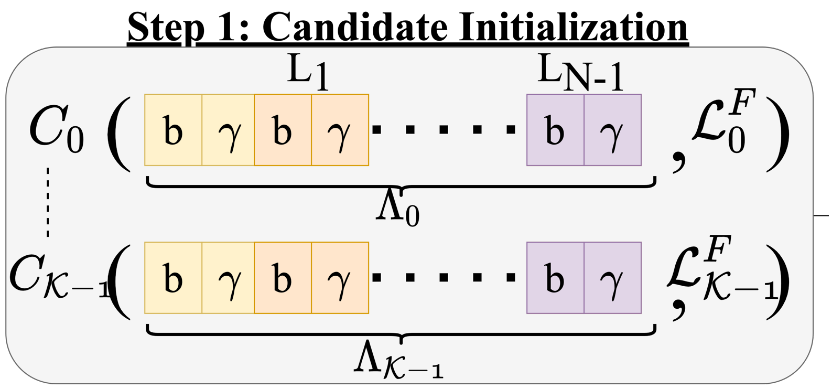


LP: Logarithmic Posit
AF: AdaptivFloat

- By parameterizing the individual bitfields of LP, we are able to adapt the LP representation distribution to the required DNN data distribution.
- While also enjoying the hardware efficiency of LNS.

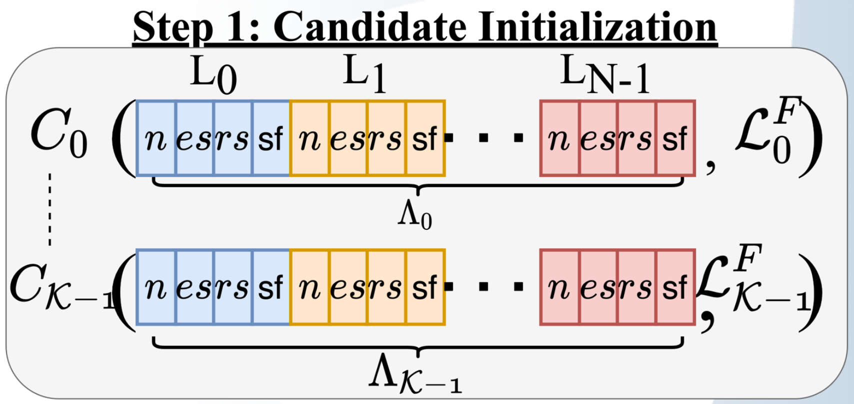
Changes in Algorithm to support LP

Uniform Quantization Candidates



Step 1: Candidate Initialization
L_1
L_{N-1}
C_0
b
γ
$\mathcal{L}_0^F$
Λ_0
$C_{\mathcal{K}-1}$
$\mathcal{L}_{\mathcal{K}-1}^F$
$\Lambda_{\mathcal{K}-1}$

Logarithmic Posit Quantization Candidates



Step 1: Candidate Initialization
L_0
L_1
L_{N-1}
C_0
n
es
rs
sf
$\mathcal{L}_0^F$
Λ_0
$C_{\mathcal{K}-1}$
$\mathcal{L}_{\mathcal{K}-1}^F$
$\Lambda_{\mathcal{K}-1}$

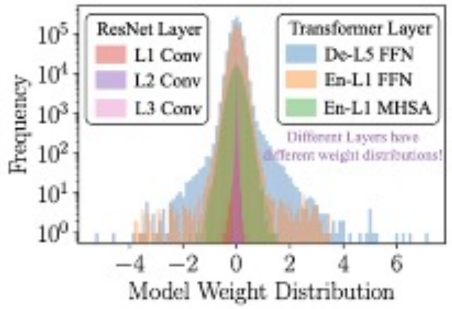
Our algorithm is general and can be easily applied to any quantization format by simply changing the candidate vector!

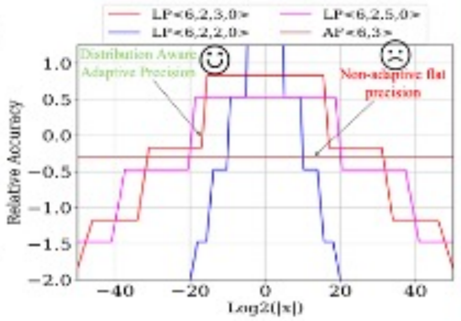
SRC

Outline

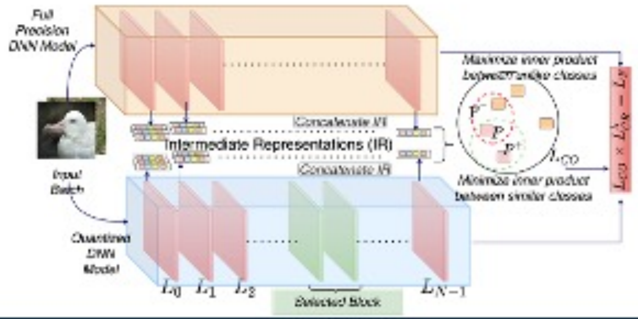
Algorithm

Introduction

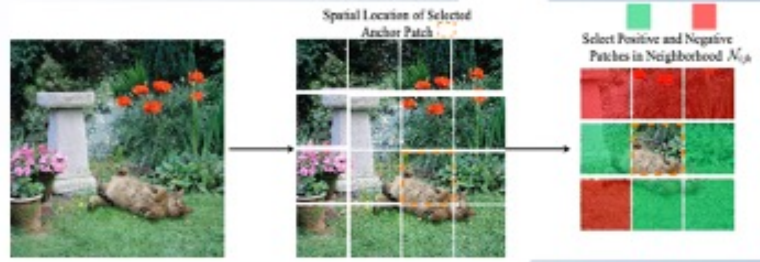




Automated Quantization


Full Precision DNN Model
Input Batch
Quantized DNN Model
Intermediate Representations (IR)
Concatenate IR
Maximize inner product between unlike classes
Minimize inner product between similar classes
Selected Block

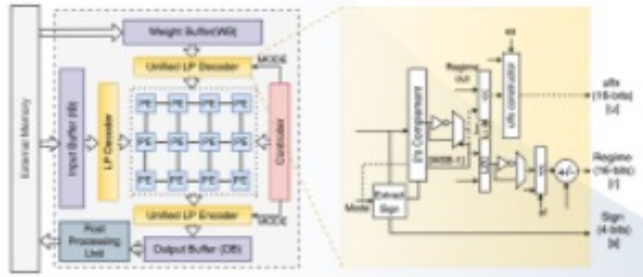
Data-Free Quantization


Spatial Location of Selected Anchor Patch
Select Positive and Negative Patches in Neighborhood

Next Generation Arithmetic: Logarithmic Posits

10
±∞
11
Negative reals
Positive reals
01
0
00

Hardware: Logarithmic Posit Accelerator

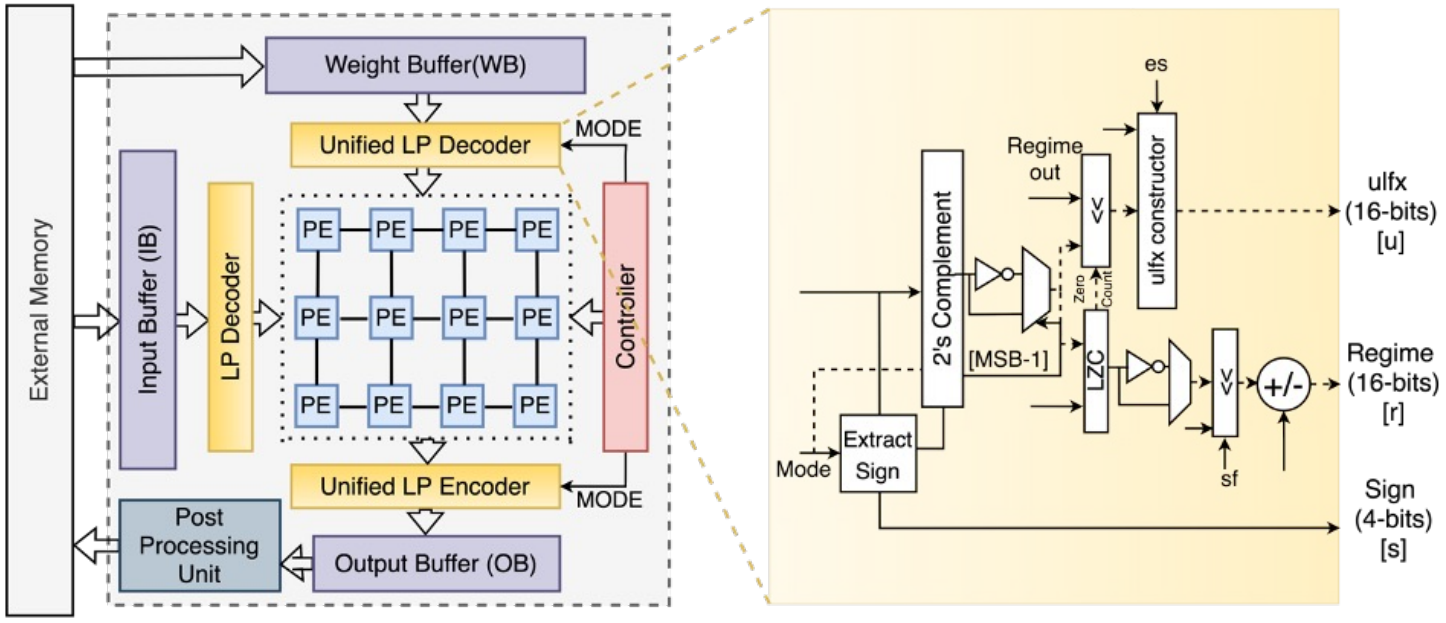


Hardware

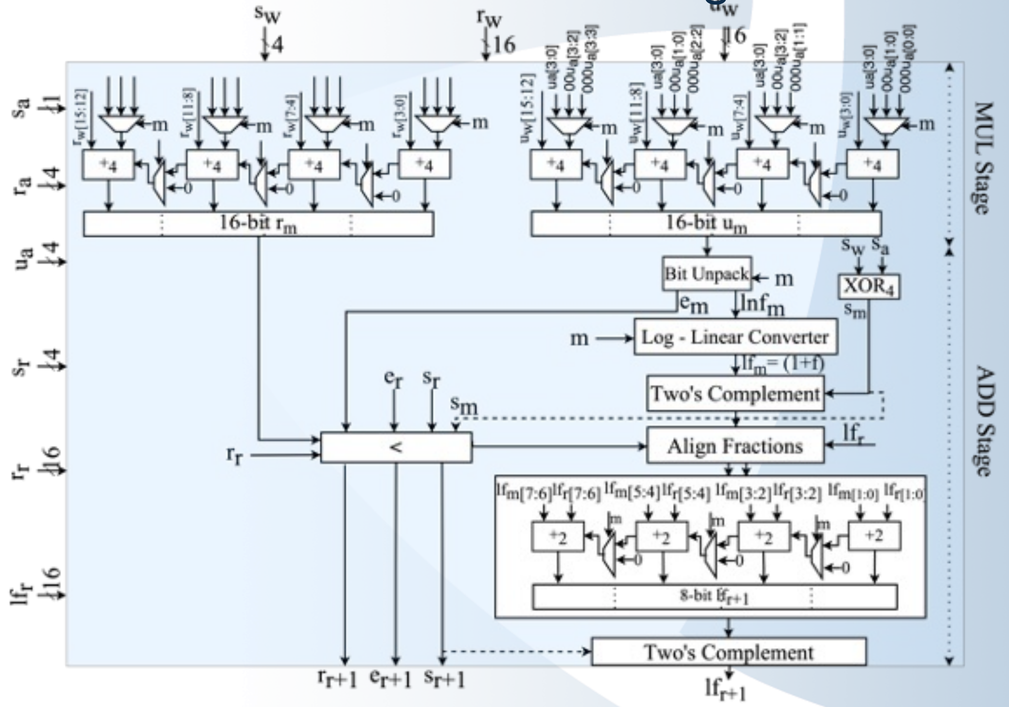
To be presented at DAC'24

Hardware: Logarithmic Posit Accelerator

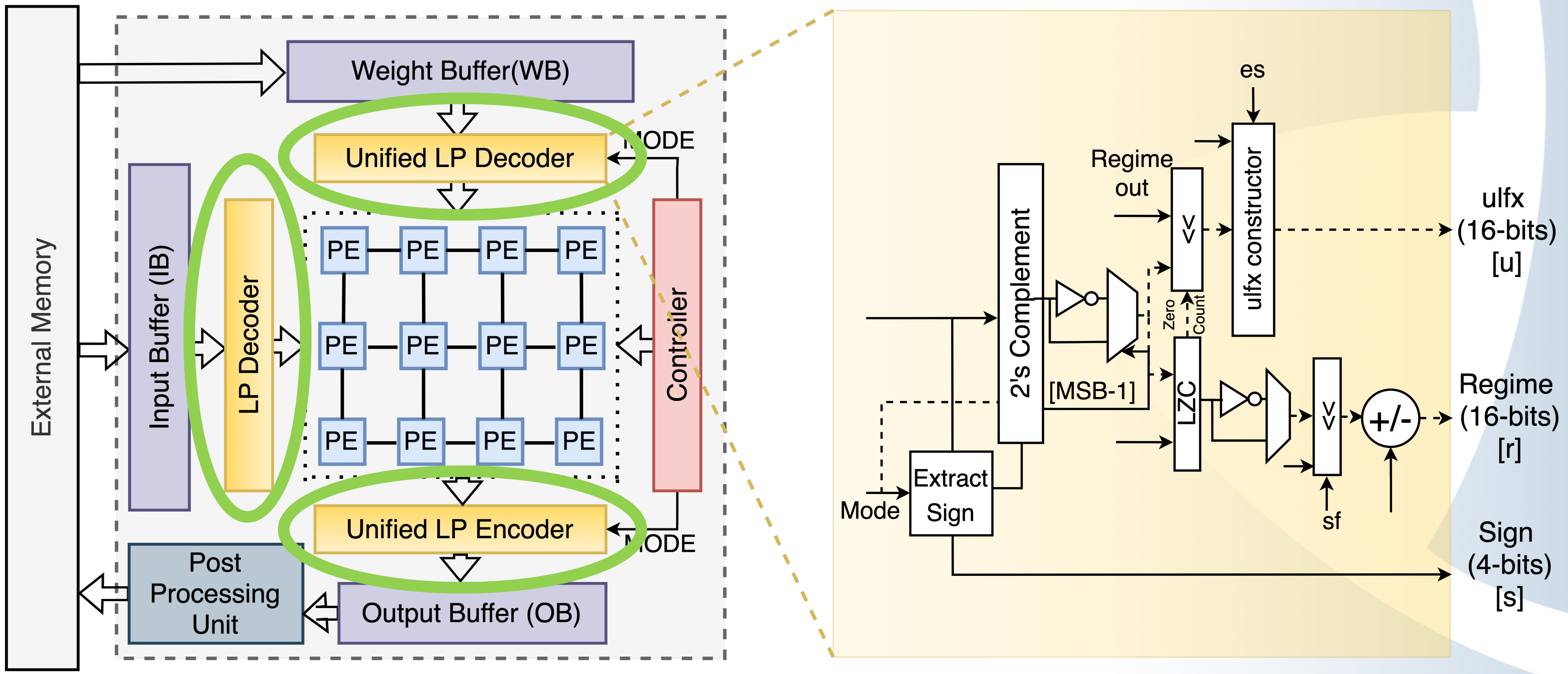
Modifications to a WS Systolic Array



Mixed-Precision PE Design



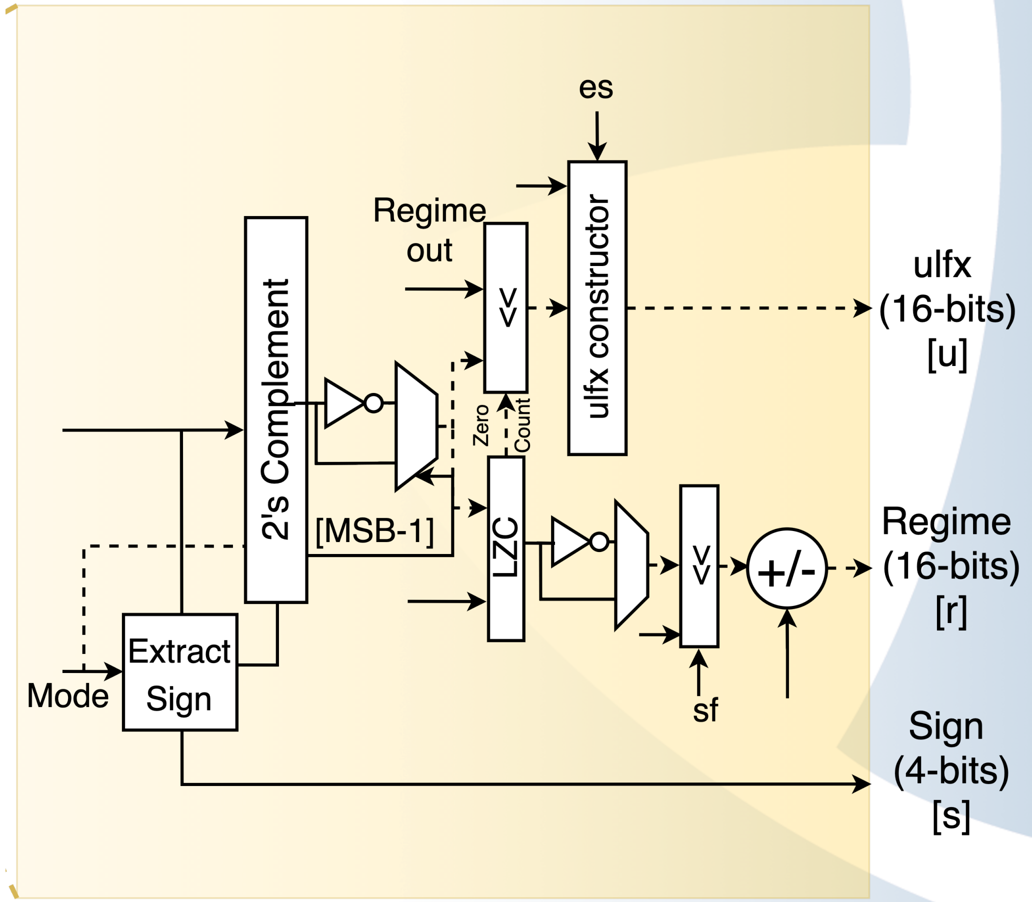
Modifications to Systolic Array



Modifications to Systolic Array

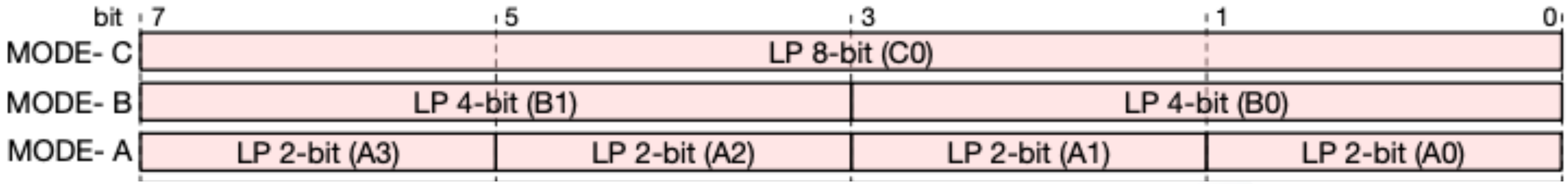
Decoder/Encoder Overhead is 1.03% of compute area for an 8x8 systolic array.

Exhibits weak scaling!!

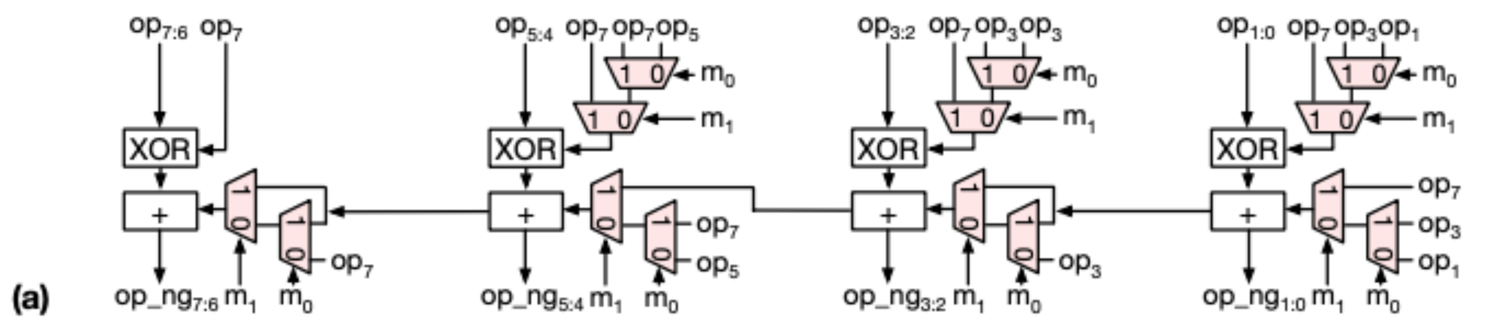


Concept of MODE

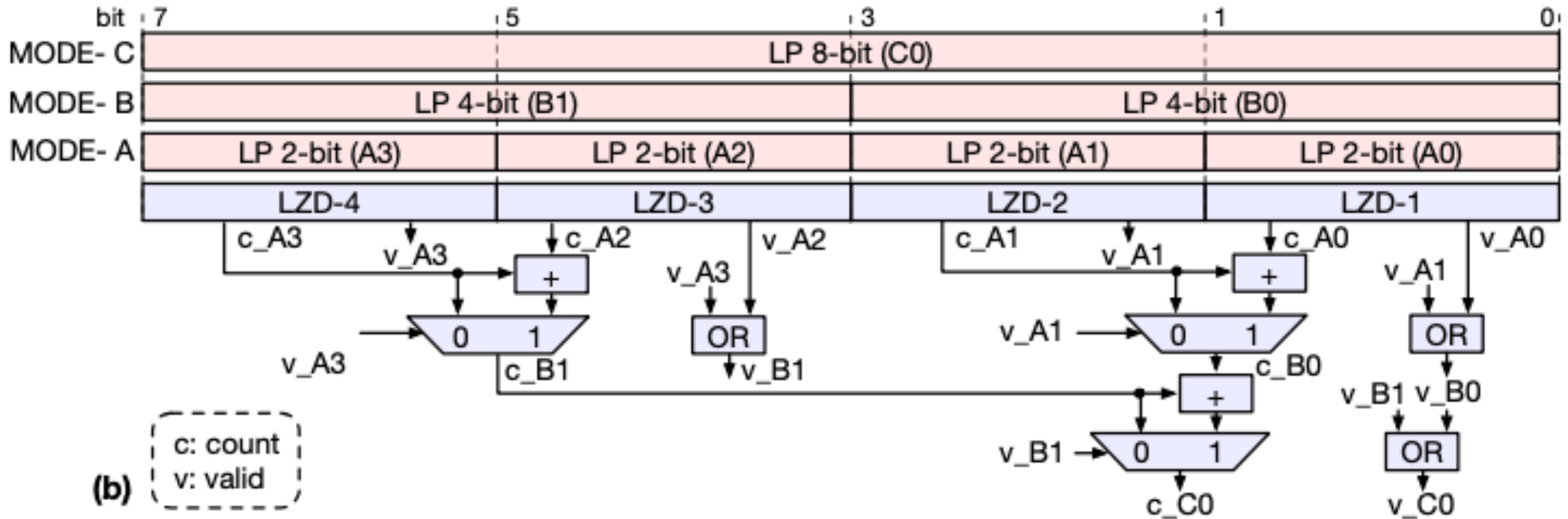
The MODE field is used to identify the precision for computation!



Implementing Mixed-Precision Components



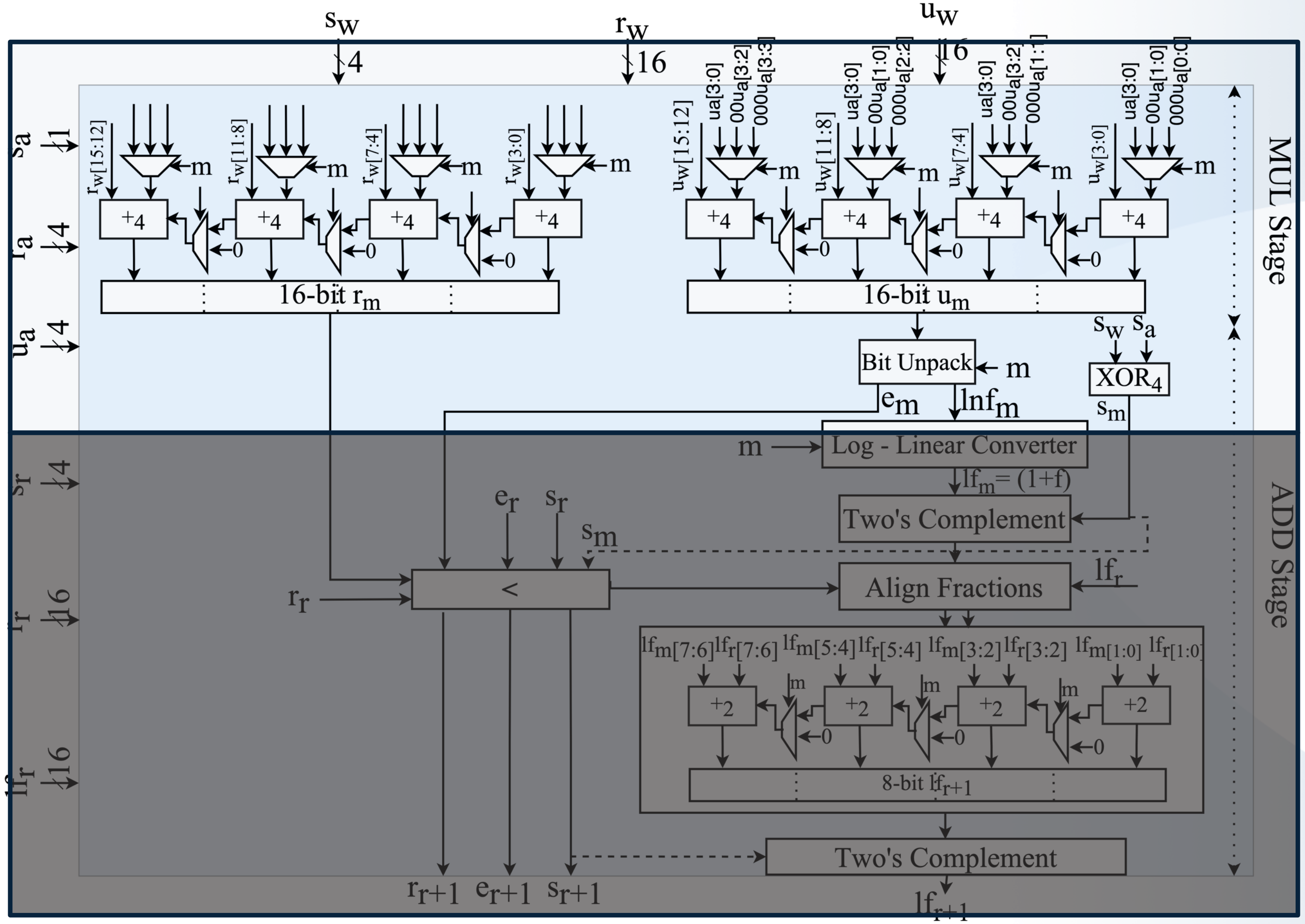
MP 2's Complementor



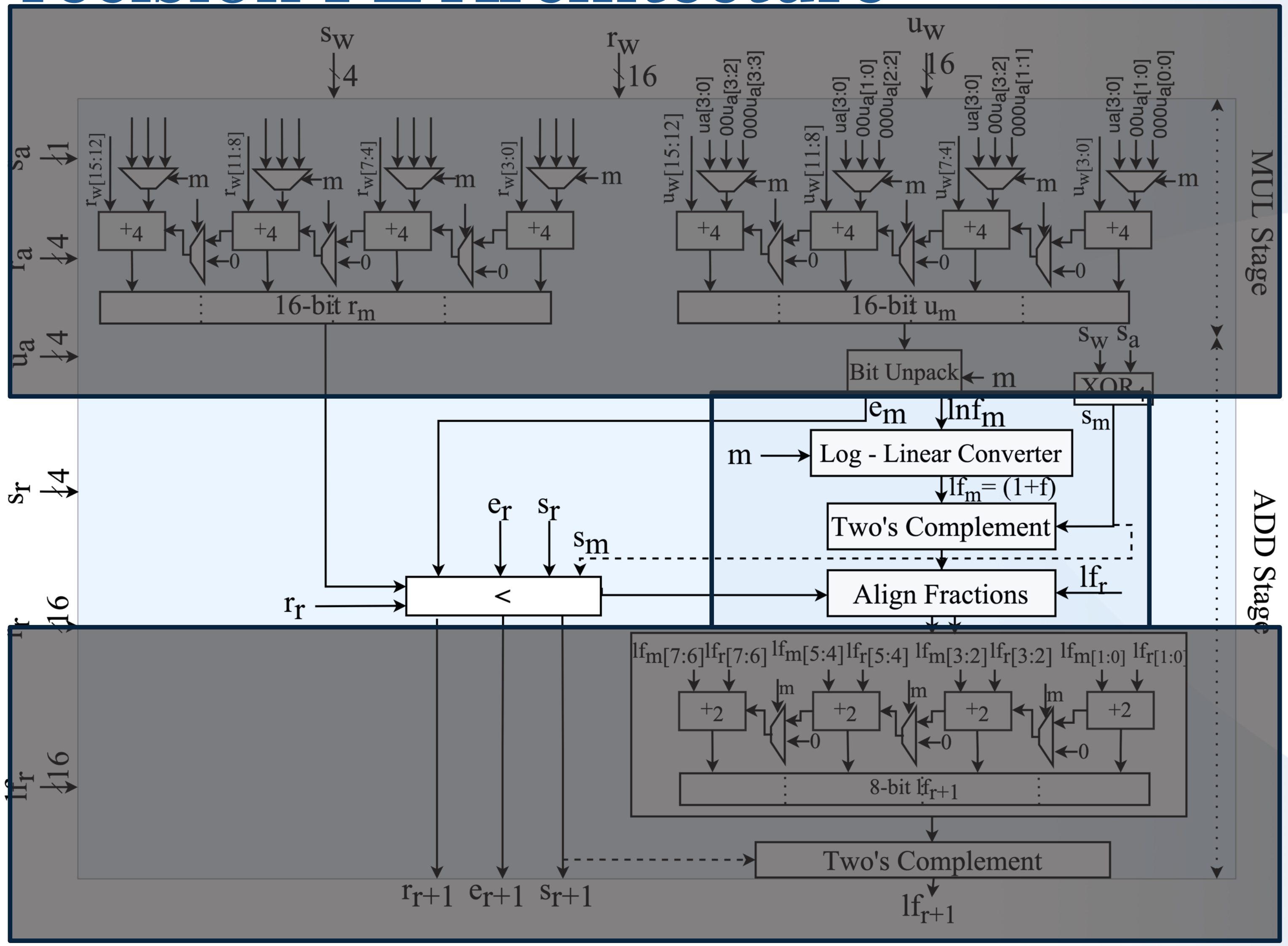
MP Leading Zero Counter (LZC)

Mixed-Precision PE Architecture

Mixed-precision LP multiply-accumulate unit made entirely of 4-bit integer adder building blocks.

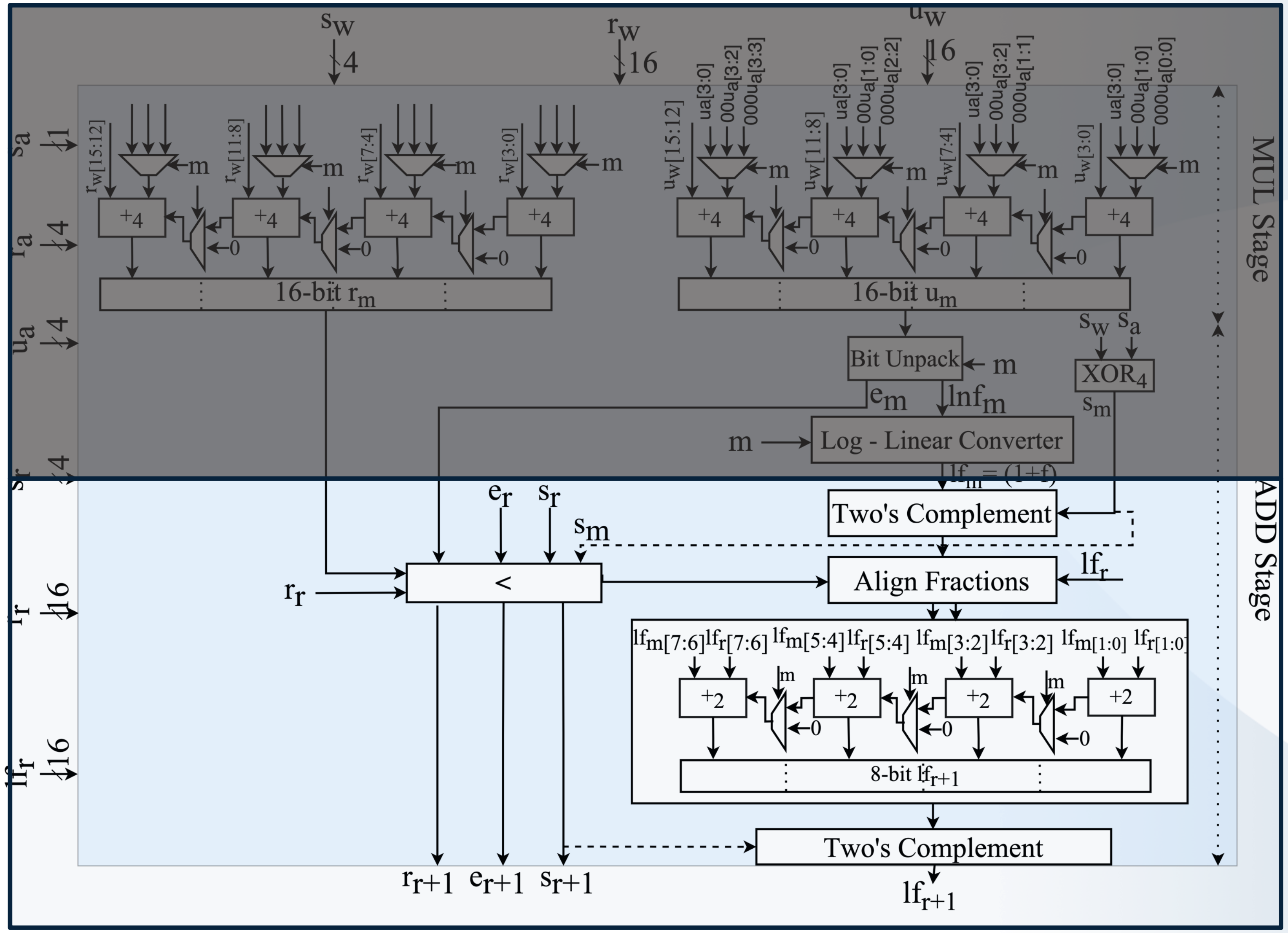


Mixed-Precision PE Architecture



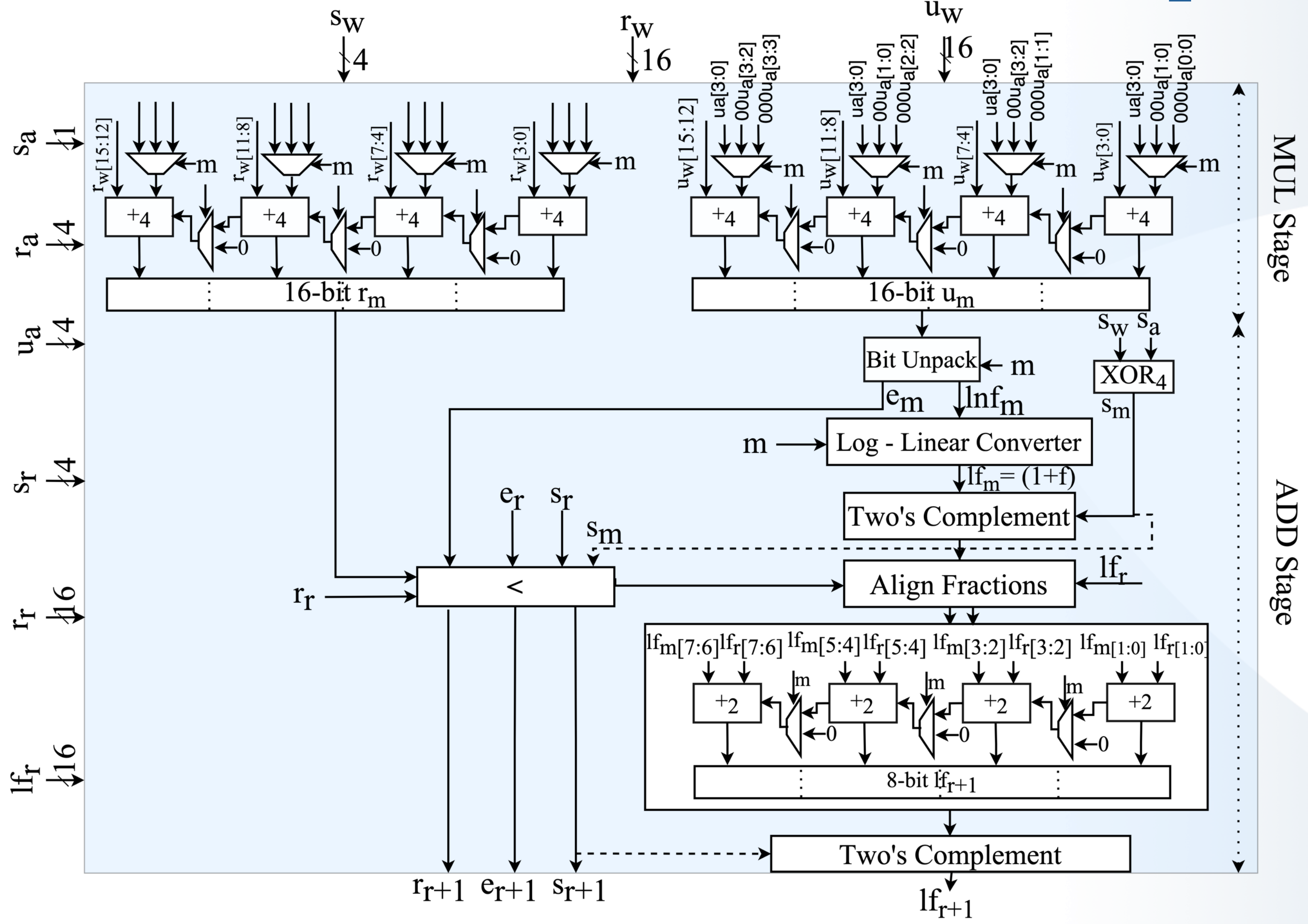
Log-Linear Converter Implemented as Combinational Logic!

Mixed-Precision PE Architecture

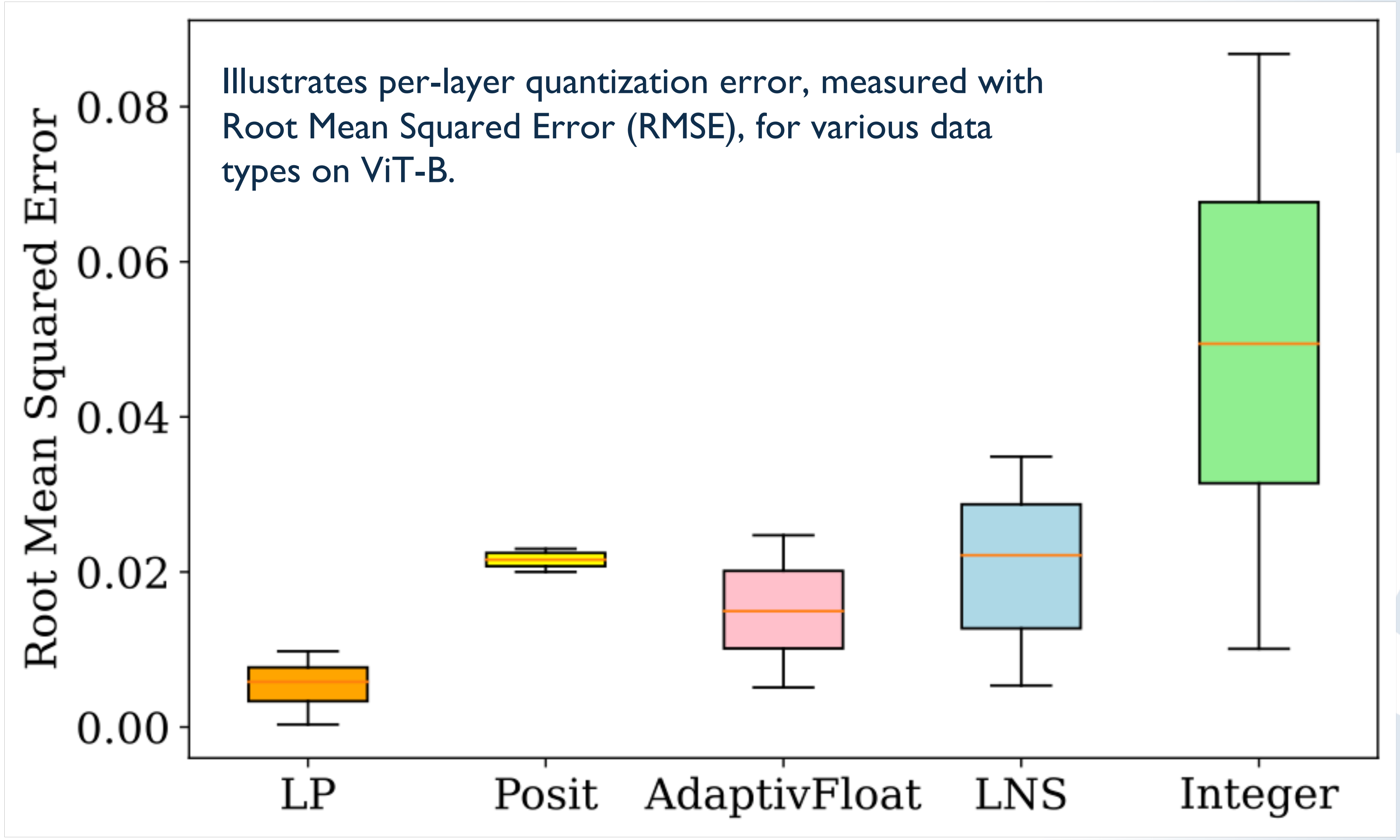


Linear Domain Addition made up of 2-bit adders.

Mixed-Precision PE Architecture: Complete Flow



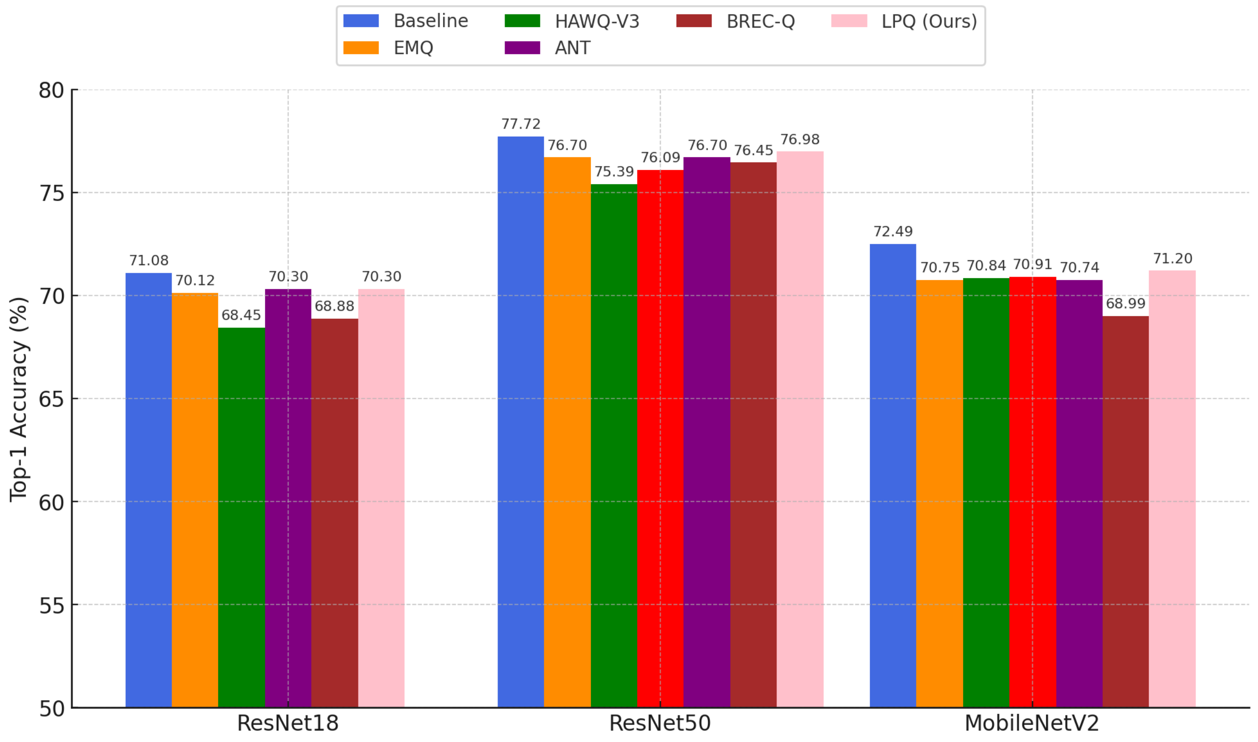
Number Format Comparison



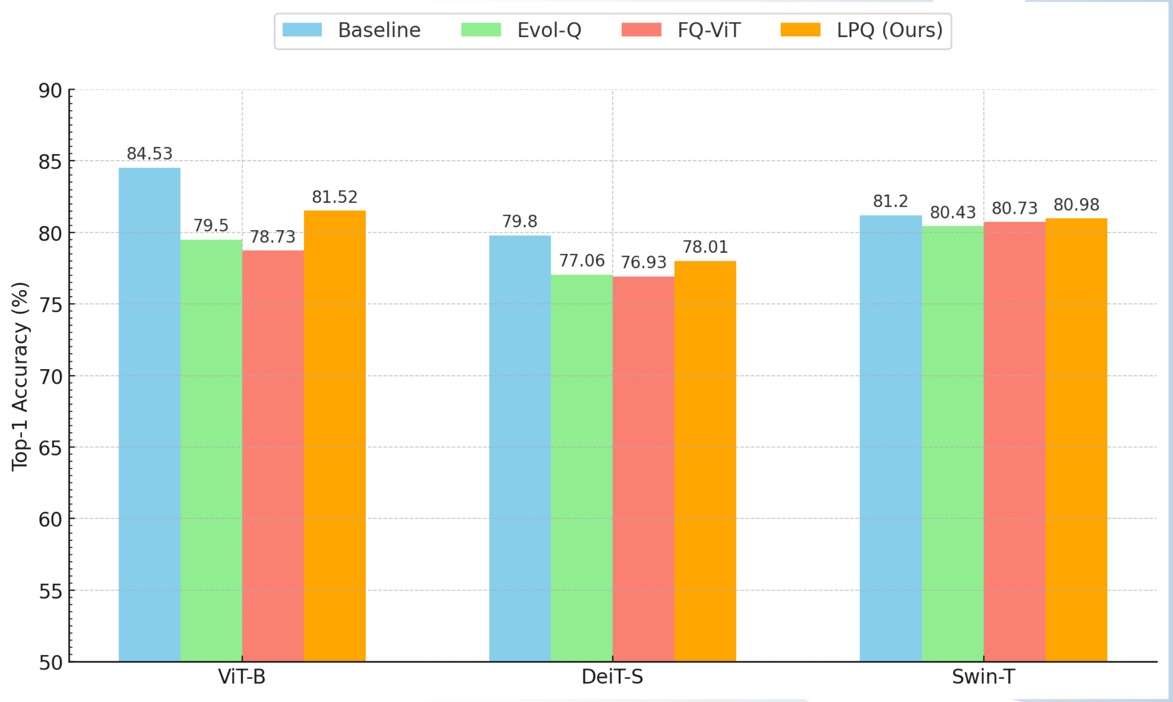
Illustrates per-layer quantization error, measured with Root Mean Squared Error (RMSE), for various data types on ViT-B.

Mixed-Precision Quantization Results: CNN & ViT

CNNs

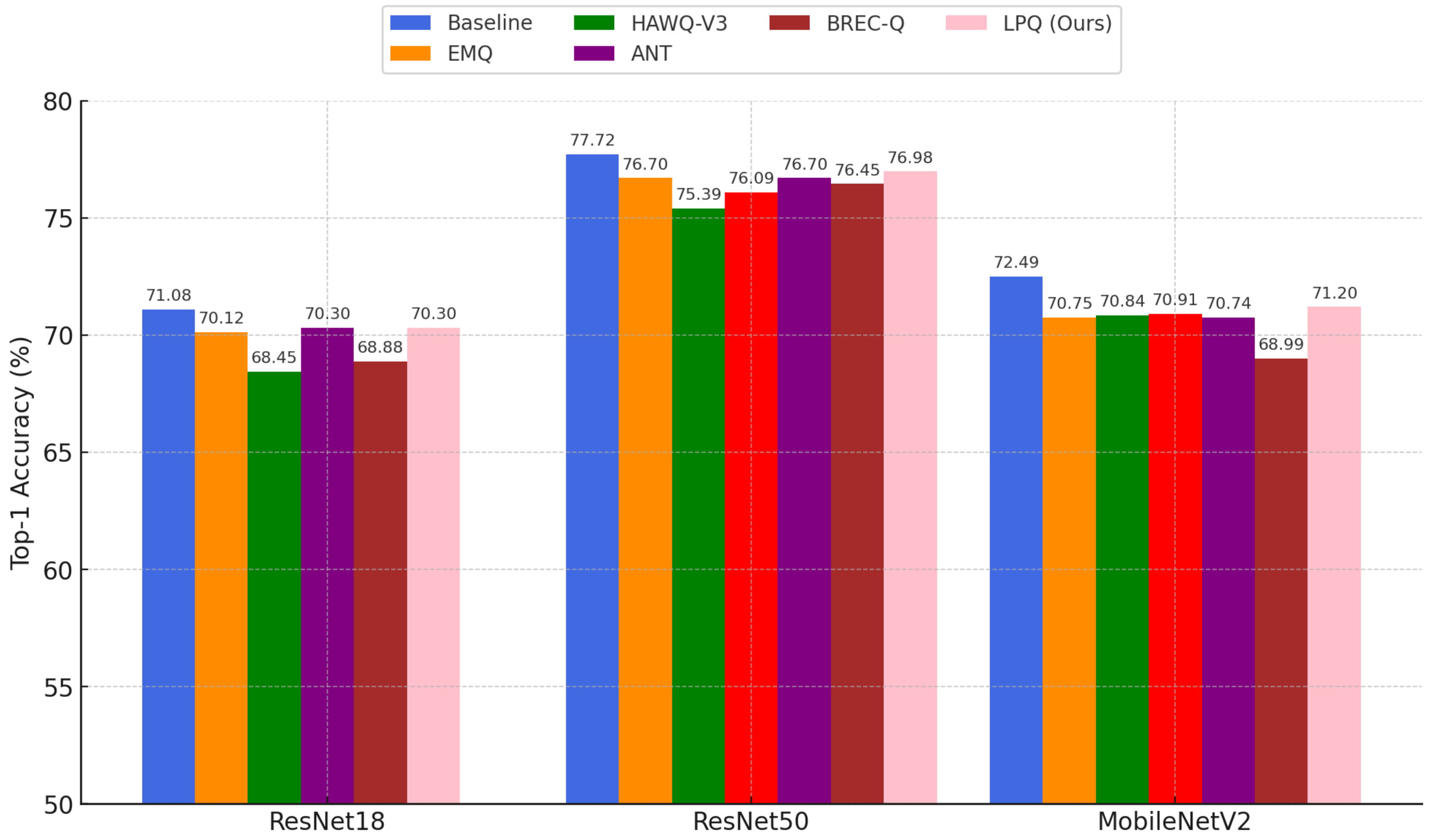


ViTs



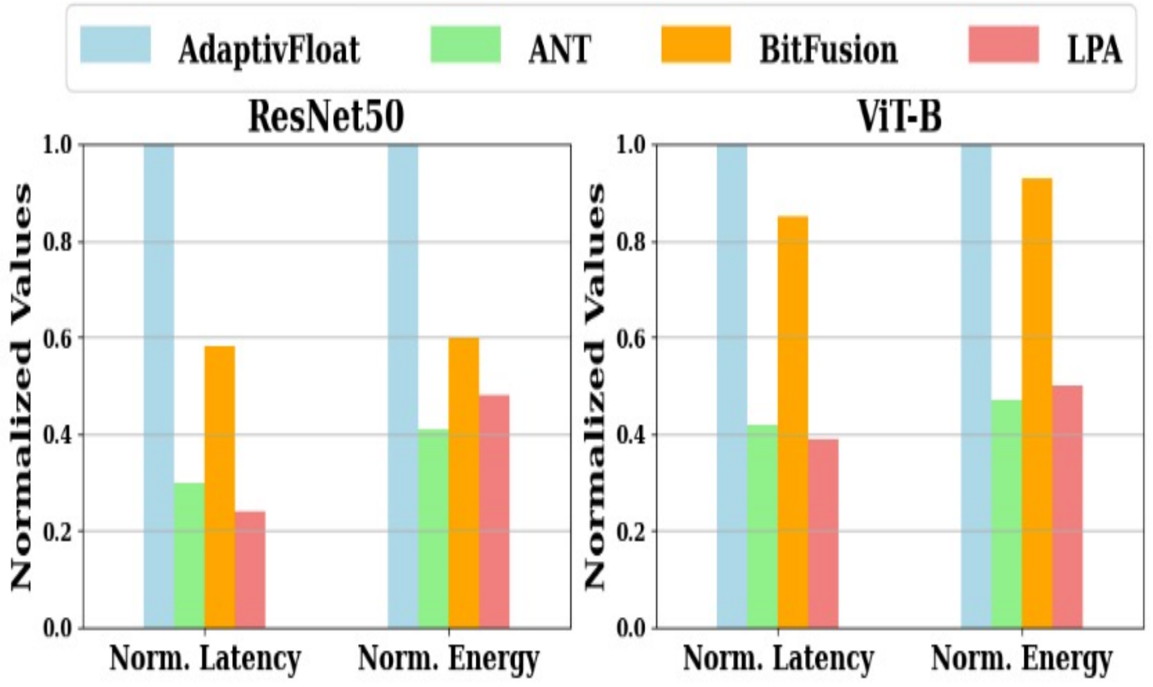
SRC

Mixed-Precision Quantization Results: CNN & ViT

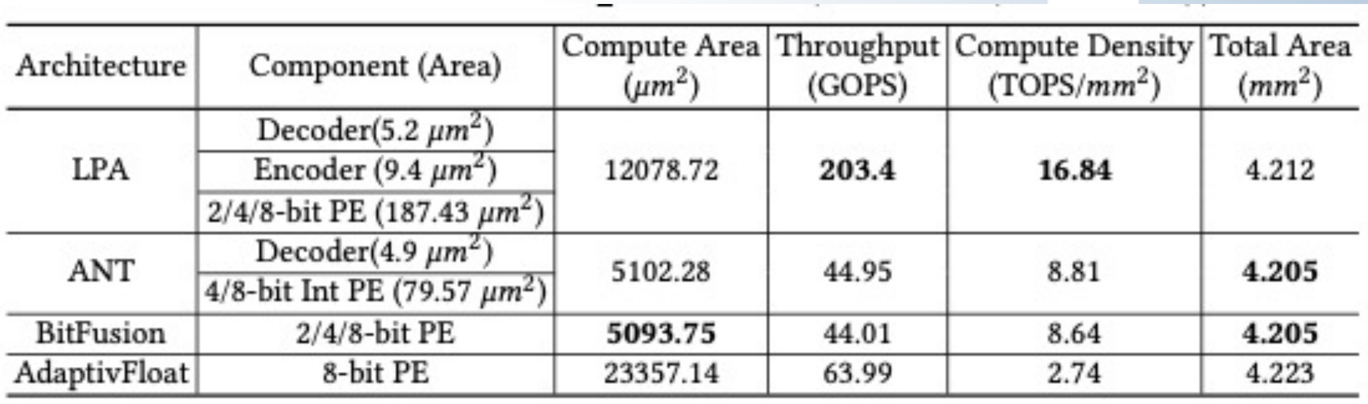


- **Highlights:**
- On average <1% accuracy degradation compared to FP baseline.
- Mixed-Precision LPQ achieves average weight/activation bitwidth of 4.2/5.5.
- 90% reduction in model size.

Logarithmic Posit Accelerator Performance



LPA exhibits the lowest latency across models, with a modest increase in energy consumption over ANT attributed to overheads due to native mixed- precision support and conversion logic.

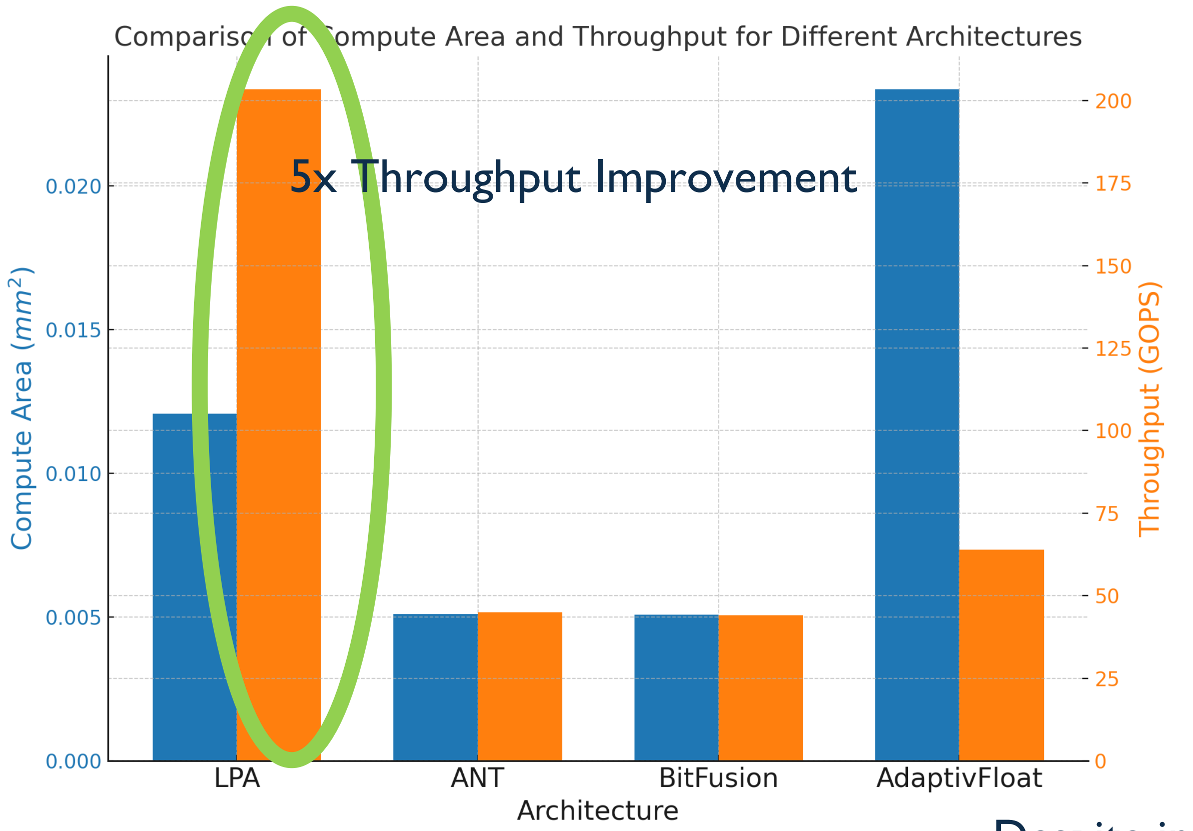


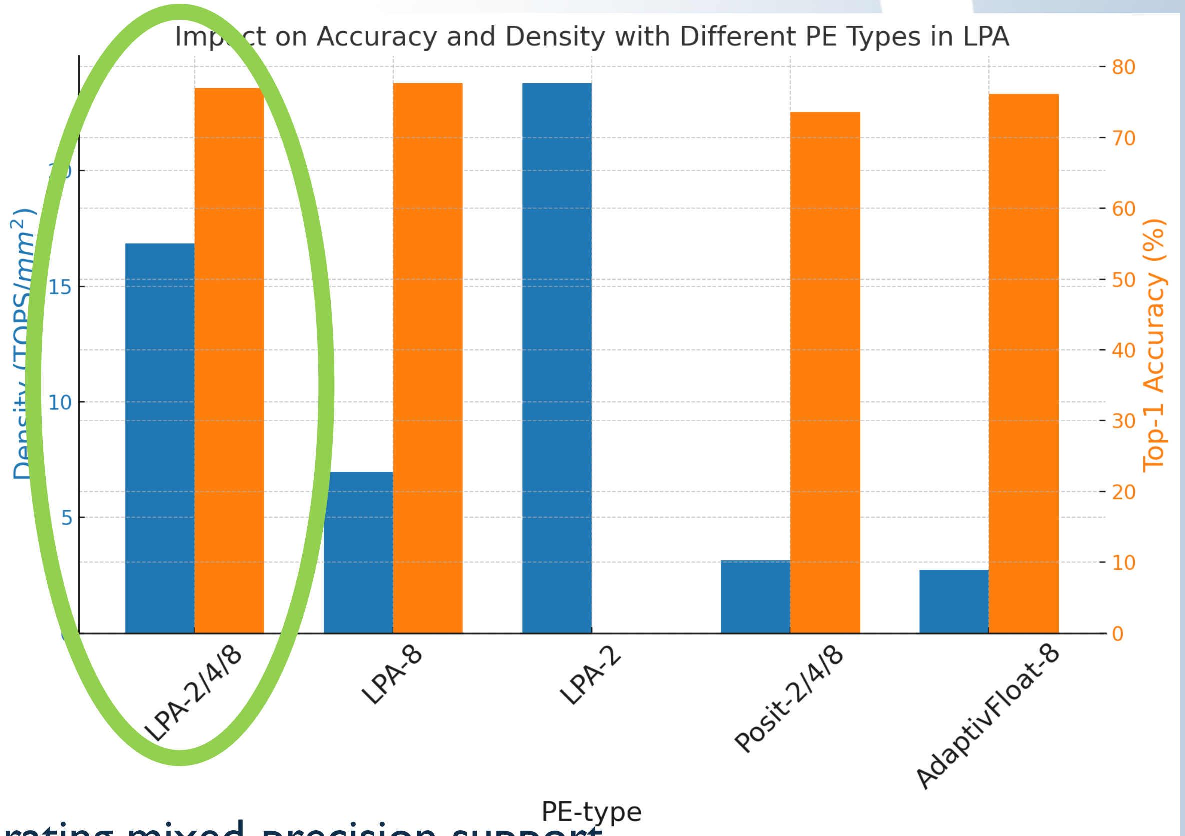
Architecture	Component (Area)	Compute Area (μm^2)	Throughput (GOPS)	Compute Density (TOPS/mm^2)	Total Area (mm^2)
LPA	Decoder(5.2 μm^2)	12078.72	**203.4**	**16.84**	4.212
	Encoder (9.4 μm^2)				
	2/4/8-bit PE (187.43 μm^2)				
ANT	Decoder(4.9 μm^2)	5102.28	44.95	8.81	**4.205**
	4/8-bit Int PE (79.57 μm^2)				
BitFusion	2/4/8-bit PE	**5093.75**	44.01	8.64	**4.205**
AdaptivFloat	8-bit PE	23357.14	63.99	2.74	4.223

Despite ANT and BitFusion exhibiting lower area when compared with LPA for the same number of PEs, LPA results in proportionately higher performance per unit area (TOPS/$mm2$) for mixed-precision DNN inference.

SRC

Logarithmic Posit Accelerator Performance





Despite incorporating mixed-precision support, LPA-2/4/8 achieves accuracy tending to the ideal scenario for both metrics, demonstrating a balanced trade-off.

Summary

CONTEXT

Current Approach and Challenges

- Approach: Algorithm-Hardware Co-Design is a promising area towards efficient DNN inference.
- Challenges: Inefficient data formats and lack of generalizable automated techniques.

APPROACH

The proposed approach:

- Develop an adaptive, hardware-friendly data format.
- Identify existing limitations of automated quantization algorithms.
- Optimize existing hardware to support next-generation data formats.

IMPACT

How do current results advance SOTA?

- <1% Accuracy drop across DNN model families post-quantization with > 15% higher compression ratio than competing methods.
- 2x improvement in PPA and energy-efficiency.

Next Steps

- Extend: Verify adaptability and generalizability to LLMs and VLMs.
- Profile: Deepen understanding of data distribution of modern DNN models to improve existing data-format parameterization.
- Improve runtime of existing algorithm for faster quantization of large-scale models.

ACADEMIA • INDUSTRY • GOVERNMENT

CoCoSys

CENTER FOR THE
CO-DESIGN OF COGNITIVE SYSTEMS