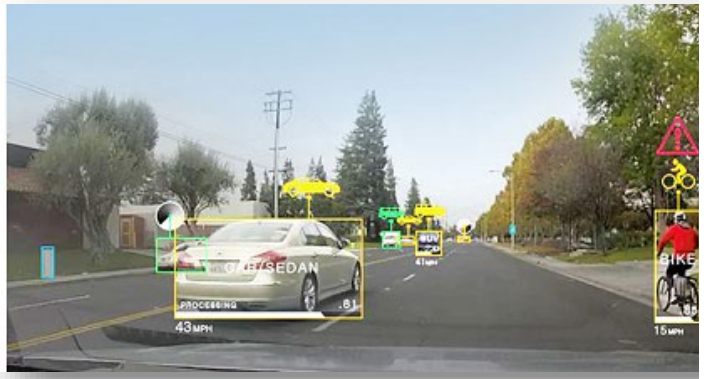
Microsoft

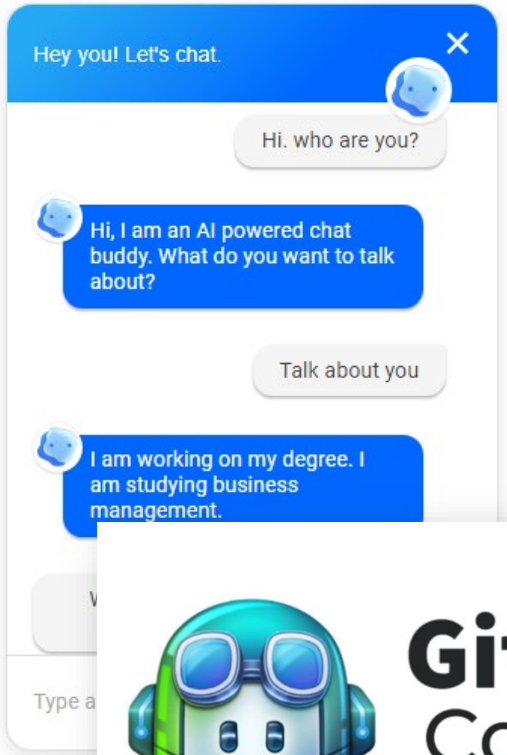
LUT-NN:
Empower Efficient Neural Network Inference With **Centroid Learning and Table Lookups**

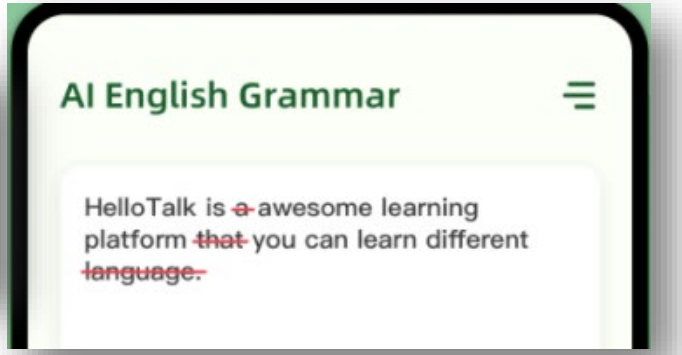
Xiaohu Tang, Yang Wang, **Ting Cao**, Li Zhang, Qi Chen, Deng Cai[1],
Yunxin Liu[2], Mao Yang

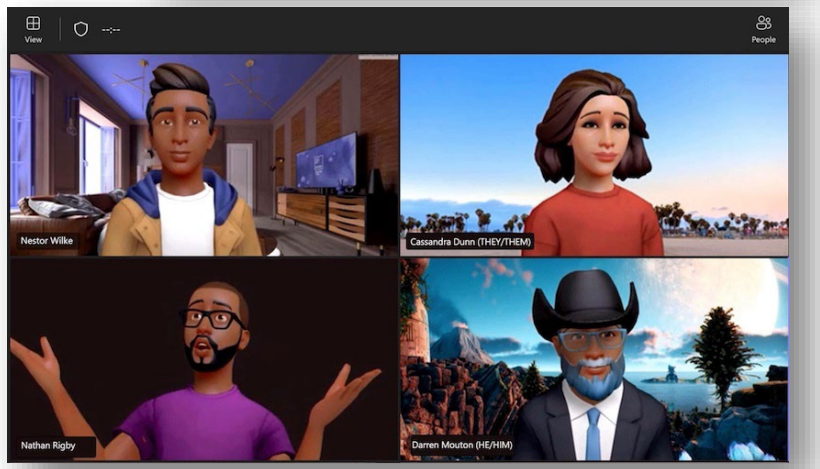
Microsoft Research,
Zhejiang University[1]
Institute for AI Industry Research, Tsinghua University[2]

It is Critical to Enable DNN Inference on Every Device



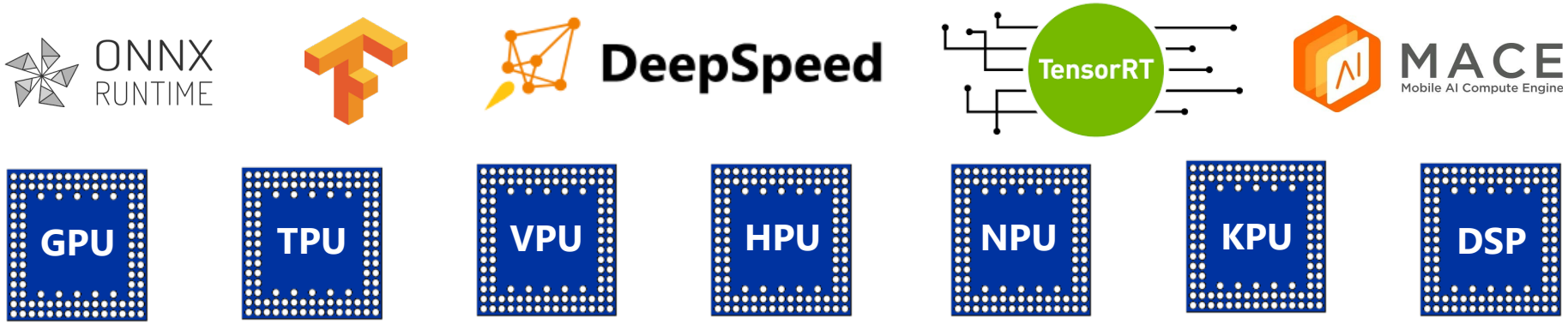






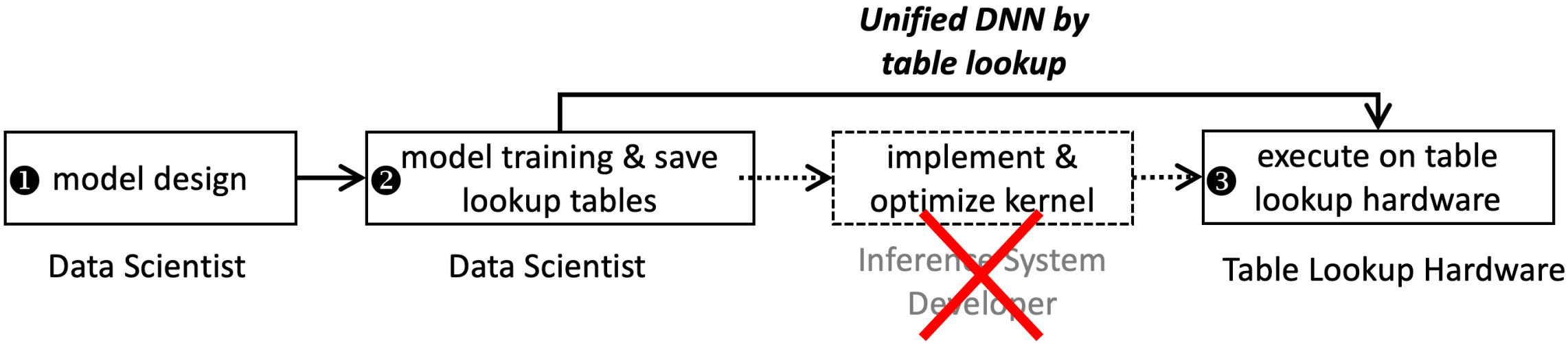
Huge Cost of Current DNN Inference on Devices

- Hardware resources cost
  - More and more computations, memory, power …
  - Even achieve O(TFLOPs/TB)
- Human labor cost:
  - Redesign the kernels, accelerators, for new DNNs
  - O(months/years) to deploy new models to customers



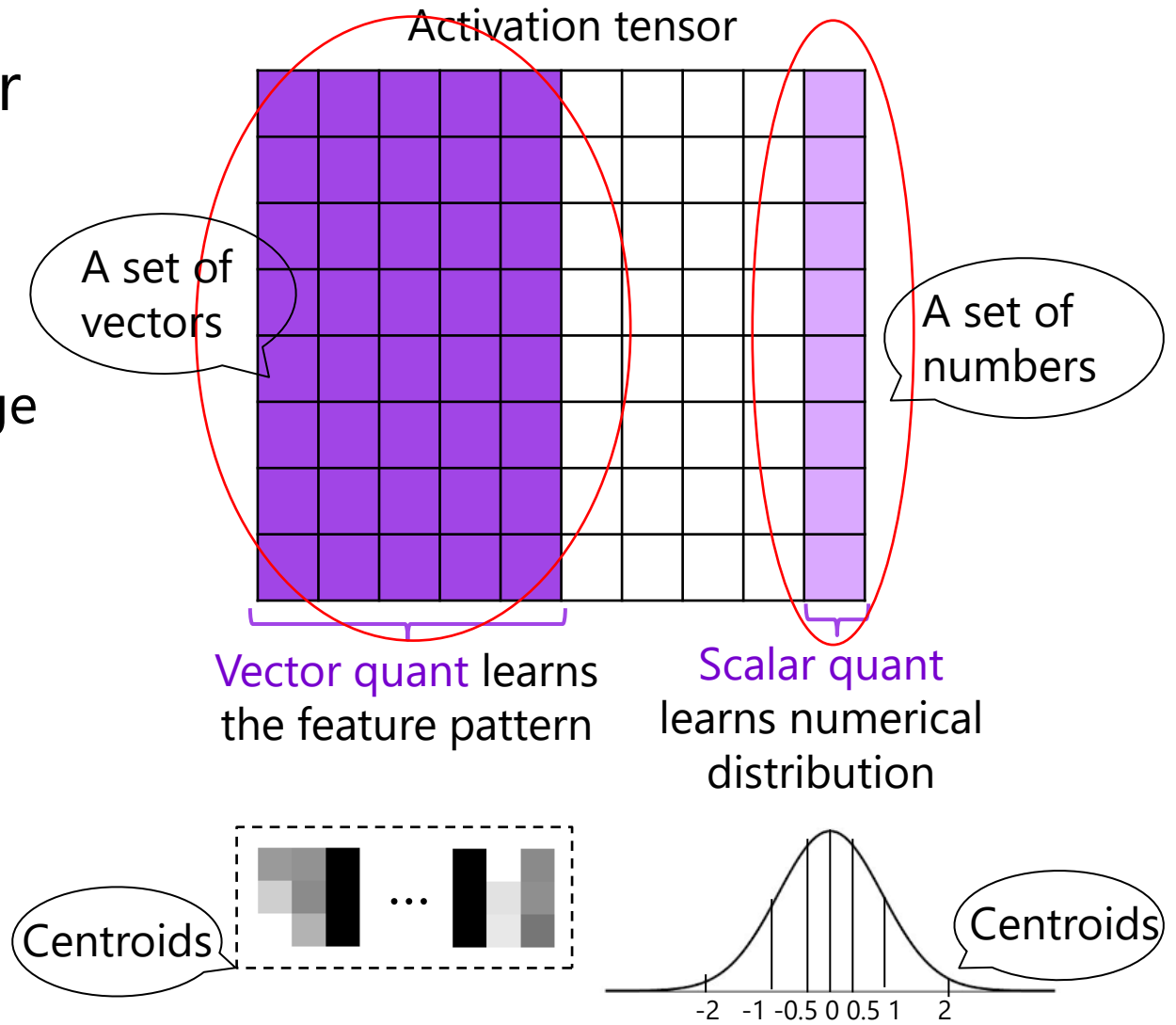
Our Strategy:
Towards DNN Inference by Table Lookup (LUT)

- Precompute results for each kernel and save in the table
- Lookup the tables for results during inference
- Advantages:
  - Much reduced computations, power, memory of DNN inference
  - A trained model can be directly deployed without reimplementing kernels or hardware



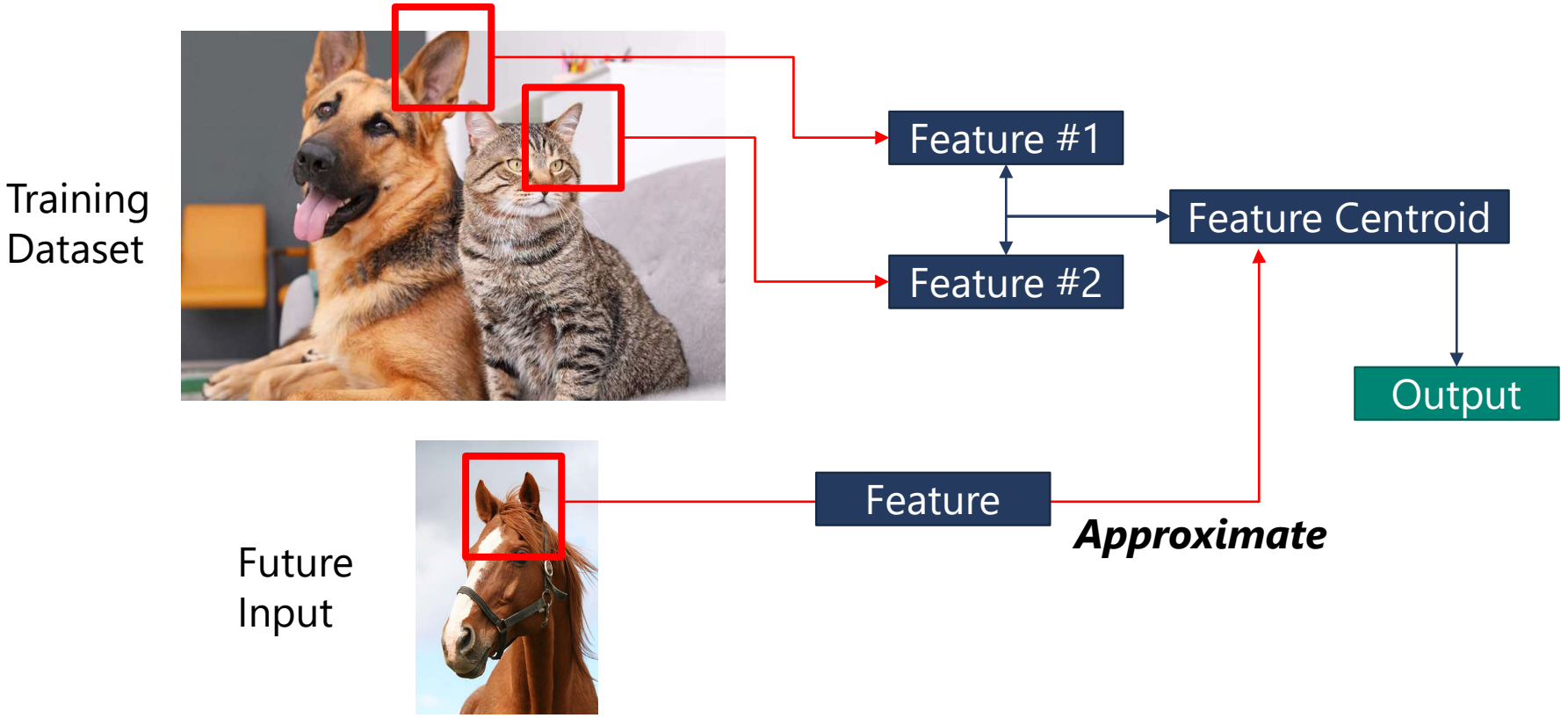
What to Store in the Table?

- A vanilla result table is too large for a computing kernel.
  - For FP16 activation tensor x Weight, table size is 2^{16} x model size
  - Scalar quant activation to FP8, still too large
- Ours: vector quantization (<1bit) for activation
  - e.g., Vector[1:16] (FP16) to 16 centroids, compression ratio 16 x 16 bits / 4 bit = 64
  - Table size is $2^4/16$ = 1 x model size



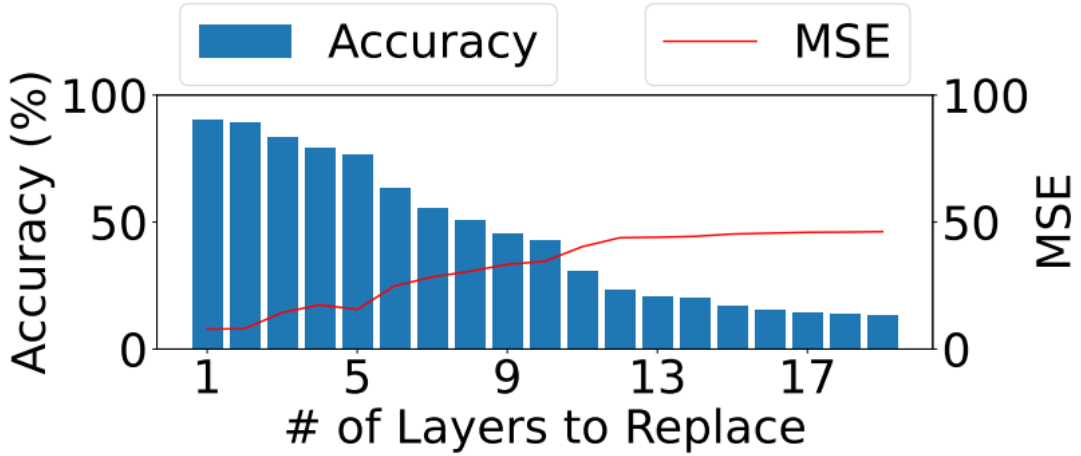
A Semantic Example for Vector Quantization

- The essence of DNN: Each layer of DNN is to extract a level of features, and similar features result in similar outputs.



Related Work Can Only Apply to One Layer

- Maddness[1] initiates to use product quantization (PQ) to replace matrix multiplication by LUT.
  - PQ is an efficient method for vector quantization.
  - It decomposes vector space into sub-vector space and quantize.
- However, Maddness can only replace the last layer of a model by LUT.



Error accumulates and accuracy drops while replace more layers with PQ

[1] Davis Blalock and John Guttag. “Multiplying Matrices Without Multiplying”. ICML 2021.

Mismatched Goal of PQ and DNN Centroid Learning

- Issues for current work: the optimization goal of PQ for a matrix multiplication mismatches with the goal of PQ for DNN.

Goal of product quantization

Learn centroids to minimize error to a matrix multiplication

$$\arg\min_{P} \sum_{c} \sum_{i} \left\| \hat{A}_i^c - P_k^c \right\|^2$$

$\neq$

Goal of DNN

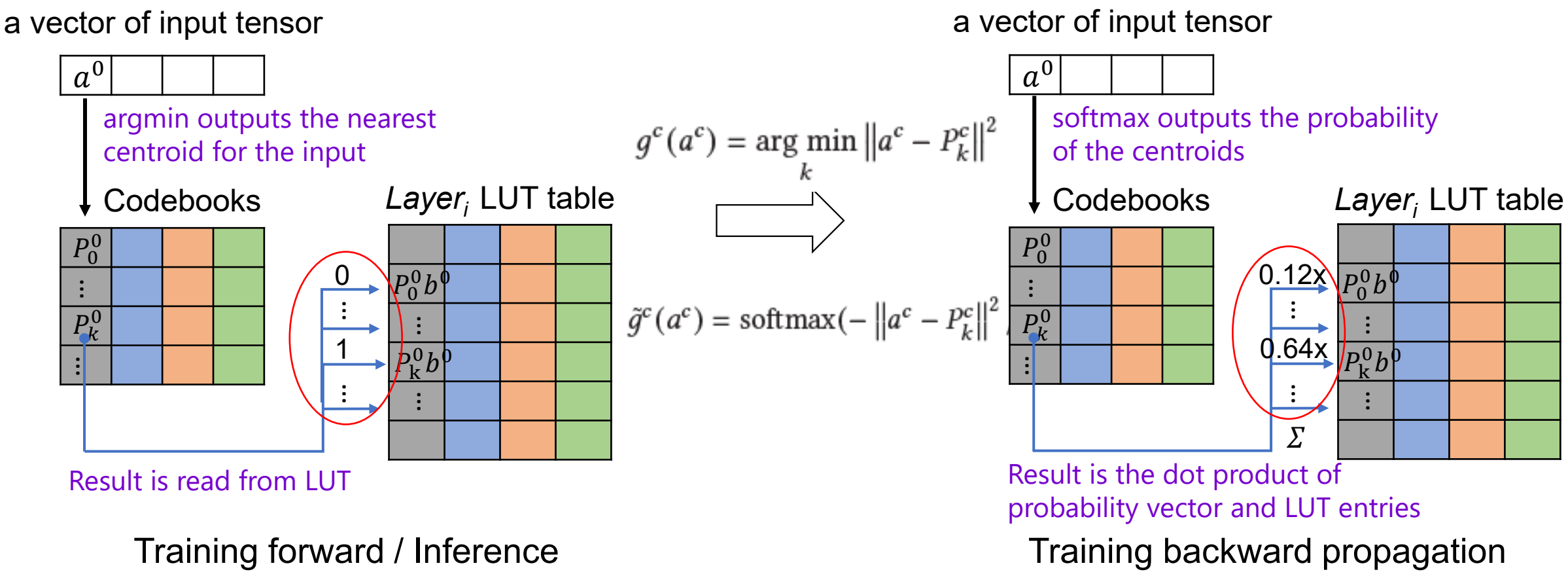
Learn centroids of each layer to minimize the final loss of the model

The Key to Apply PQ to DNN: Backpropagation

- It is necessary to learn centroids of each layer through backpropagation.
  - Pass the final loss to each layer and learn centroids.
- Challenge: vanilla PQ is not differentiable.
  - The argmin of encoding function $g^c(a^c) = \arg\min_k \left\| a^c - P_k^c \right\|^2$
- LUT-NN novel training and inference pipeline:
  - Differentiable centroid learning for training
  - New table lookup operators for inference

LUT-NN Training: Differentiable Centroid Learning

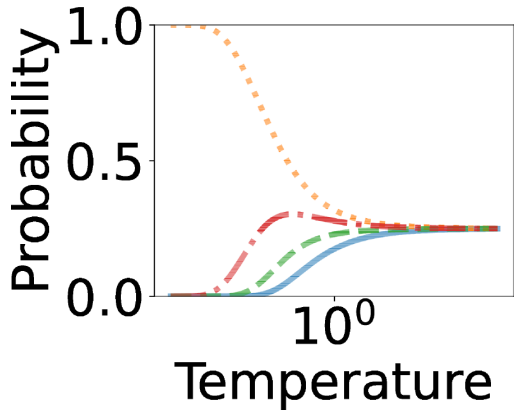
- (1) Soft-PQ centroid learning by Softmax: Softmax for backpropagation



LUT-NN Training: Differentiable Centroid Learning

- (2) Learned temperature: to learn Softmax temperature of each layer during training
  - Temperature controls the approximation of Softmax to argmax.

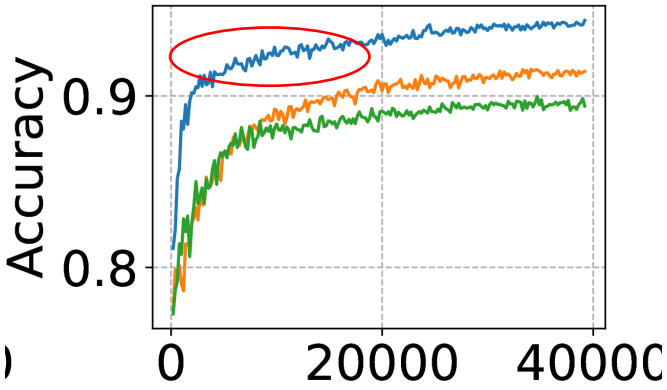
$$soft\max(x)_i = \frac{\exp\left(\frac{x_i}{temperature}\right)}{\sum_{k=1}^{K}\exp\left(\frac{x_k}{temperature}\right)}$$

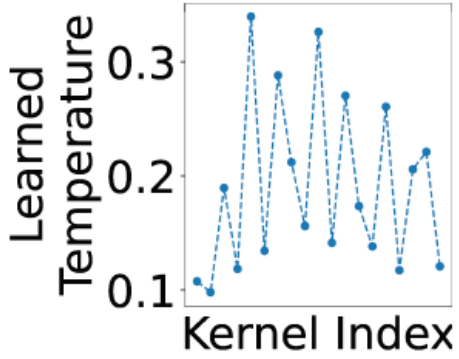


$temperature \to \infty, soft\max(x)_i \to \frac{1}{k}$

$temperature \to 0, soft\max(x) \to onehot(\arg\max(x))$

Our learned temperature achieves higher accuracy



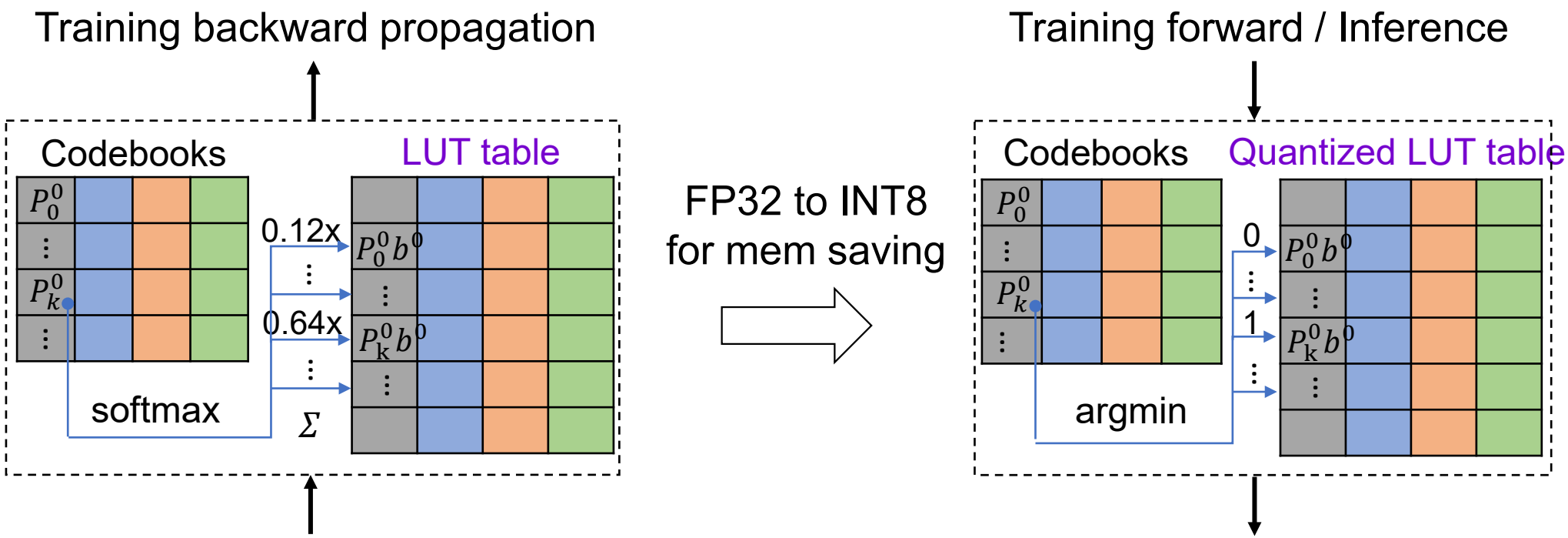


# of training iterations

Learned Temp. Annealing Temp. Temp. is 1

LUT-NN Training: Differentiable Centroid Learning

- (3) Scalar-Quantization Aware Training: adapts the approximation introduced by scalar quantization.
  - Scalar-Quantization level for LUT is not sensitive to final accuracy



LUT-NN Training: Differentiable Centroid Learning

- The training can adapt three levels of approximation

  (1) Approximation introduced by centroids:

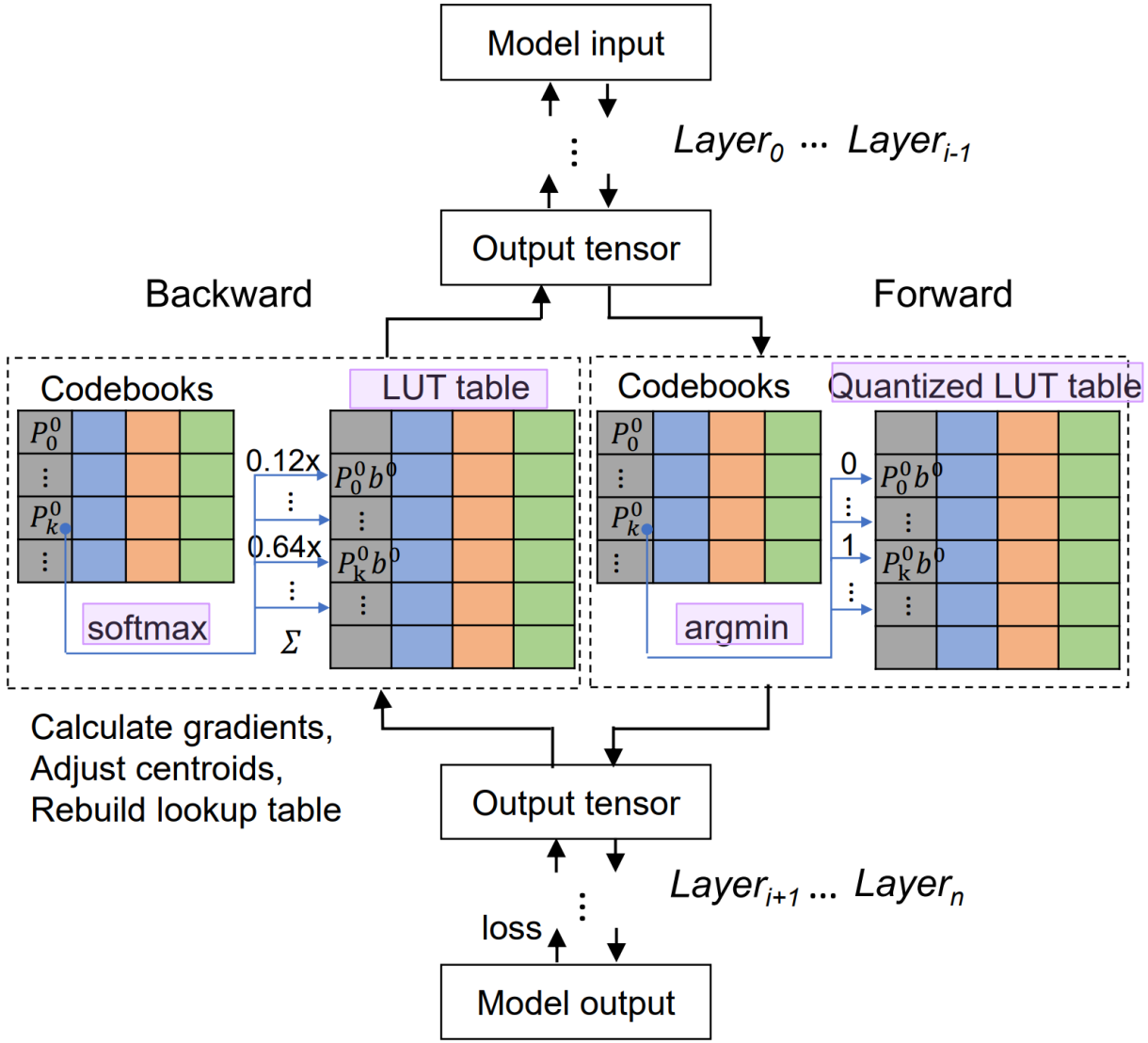
  Soft-PQ centroid learning by Softmax

  (2) Approximation introduced by Softmax :

  Learned temperature

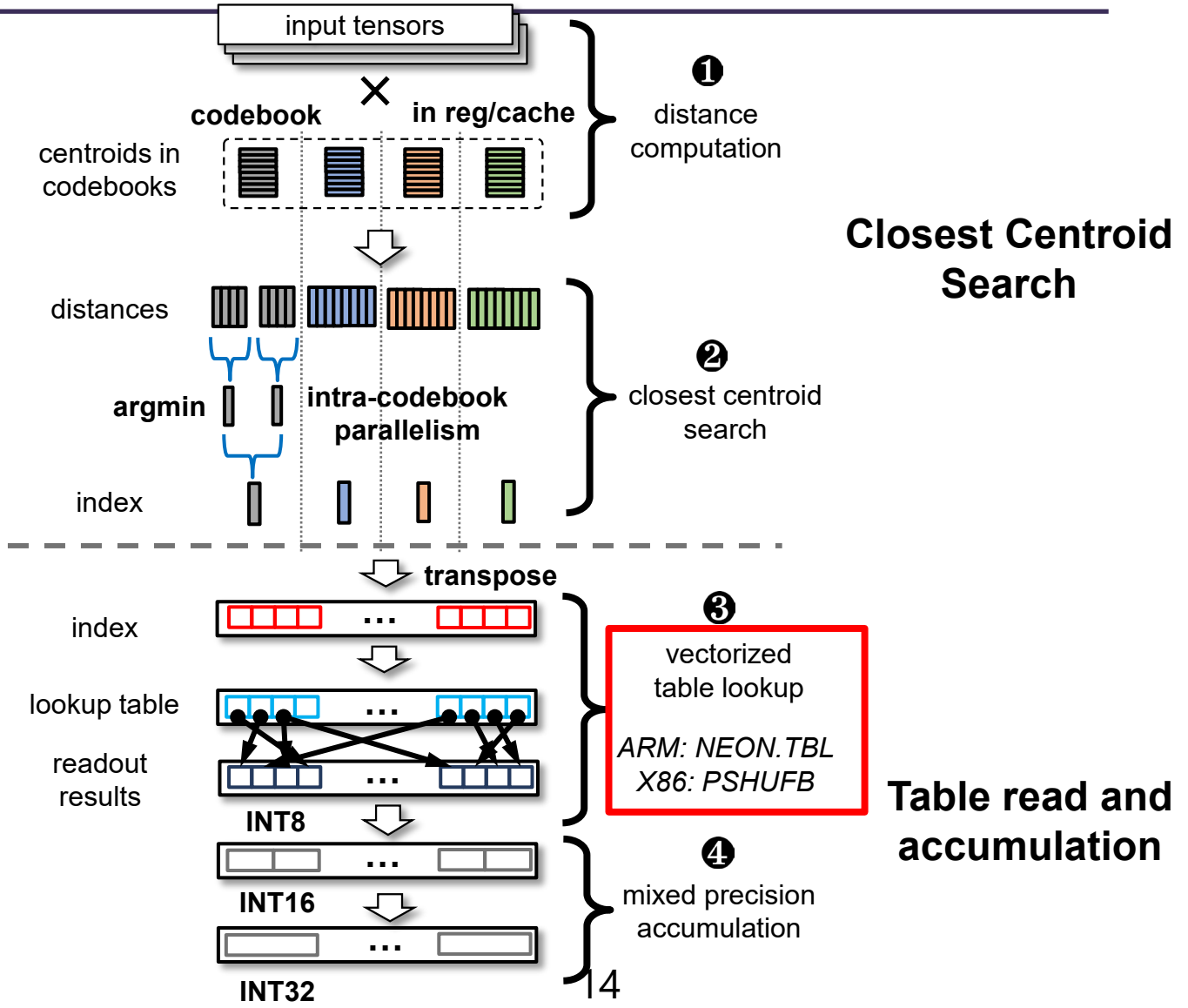
  (3) Approximation introduced by scalar quantization:

  Quantization Aware Training



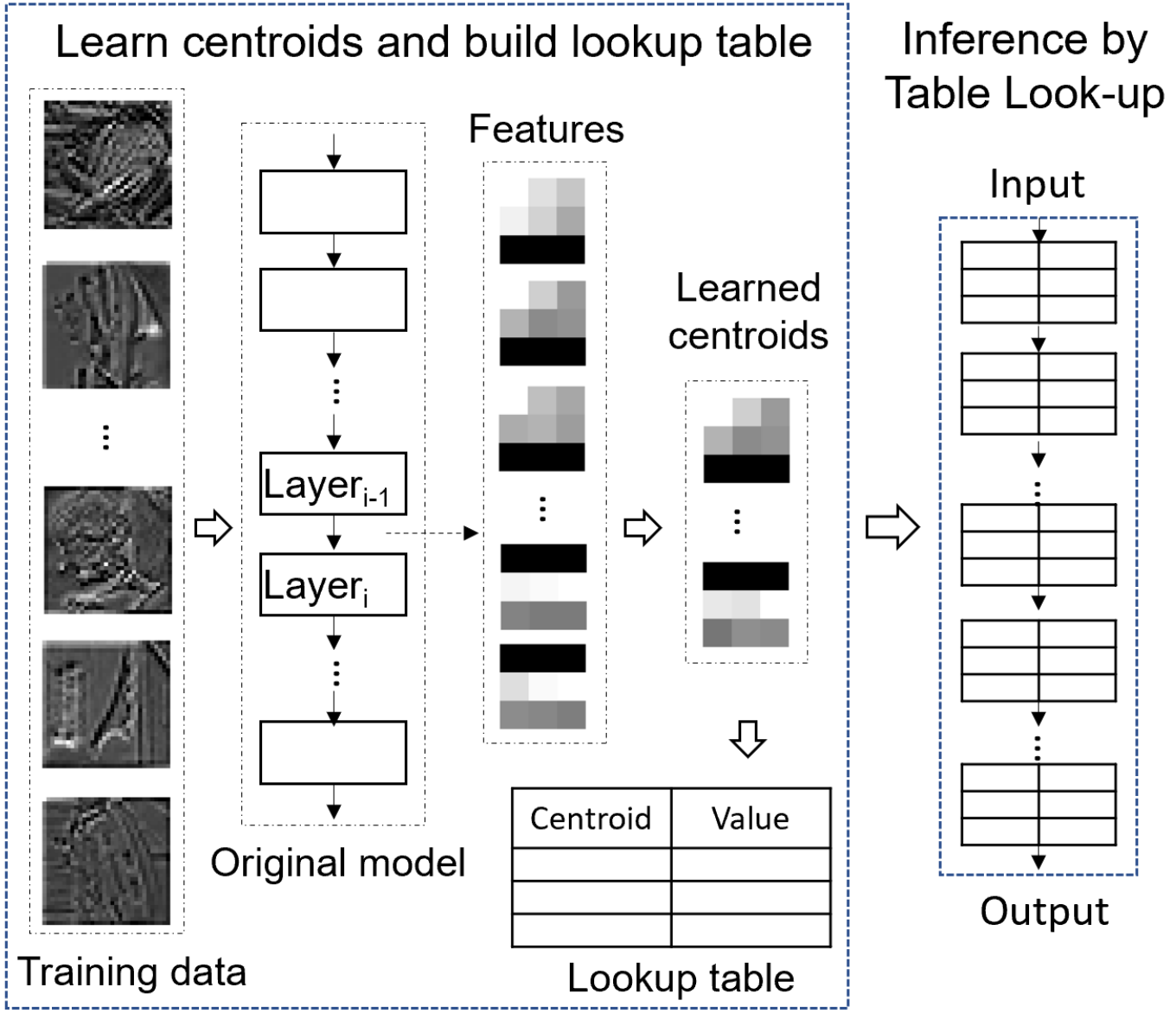
LUT-NN Inference: Table Lookup on CPUs

- Closest Centroid Search
  - Keep frequently accessed centroid table in inner loop
  - Intra-codebook parallelism
- Table read and accumulation
  - SIMD shuffle instructions for parallel read results
  - Mixed precision accumulation



LUT-NN:
The first system to enable pure LUT to replace linear computations in DNNs

Transform the model graph into a graph of LUT

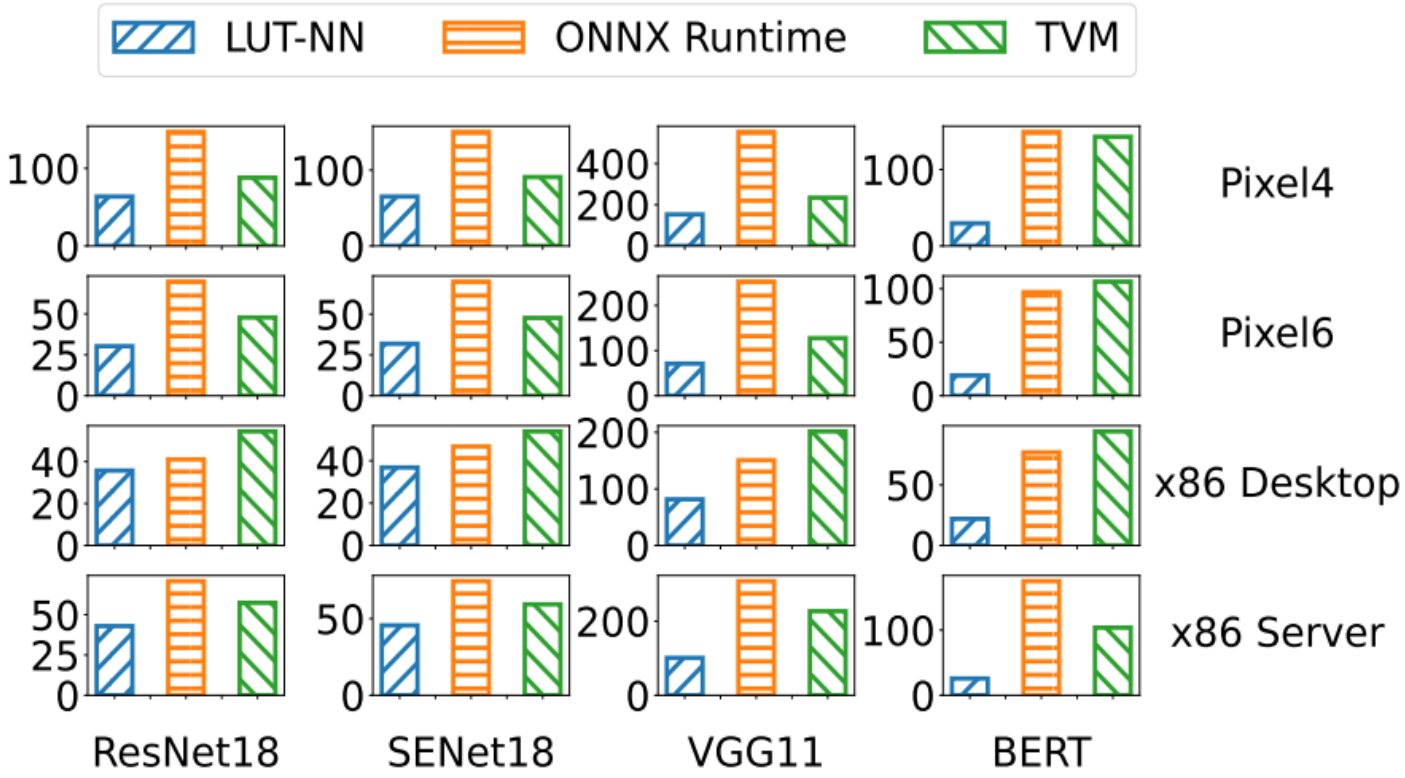


Experiment Settings

- Evaluation tasks: Image recognition, speech recognition, NLP, regression
  - CIFAR, GTSRB, SVHN, ImageNet, Google Speech Command, GLUE, UTKFace
  - Models: SENet, ResNet, VGG, Bert
- Evaluation devices:
  - ARM and x86 CPUs
- Baselines:
  - Accuracy: Maddness and original models
  - Performance: ONNX Runtime and TVM
- Hyper-parameter setting to tradeoff accuracy and cost
  - # centroids = 16, vector length = 9 for 3x3 conv, 4 for 1x1 conv, 32 for fully connect
  - Accuracy: maintain with the original model, +1.98% (speech command) to -2.42% (ImageNet).

Performance Improvement in All Dimensions

- Evaluation metrics:
  - FLOPs: 5.7x to 16x ⬇
  - Model/disk size: 3.4x to 6.8x ⬇
  - Latency: 1.3x to 6.8x ⬇
  - Memory: 1.4x to 6.5x ⬇
  - Power: 15% to 41.7% ⬇



Latency reduction
1.3x (ResNet18) ~ 6.8x (Bert)

Summarization

- LUT-NN is a new paradigm for DNN inference
- It could greatly reduce the cost of DNN deployment
- lutnn (LUT-NN) · GitHub

On-going work

- Applying LUT-NN to large language and diffusion models
- FPGA implementation for LUT-NN accelerator
- Processing in Memory technique for LUT-NN

Thank you!

Welcome researchers and intern students to join our team!

Accuracy

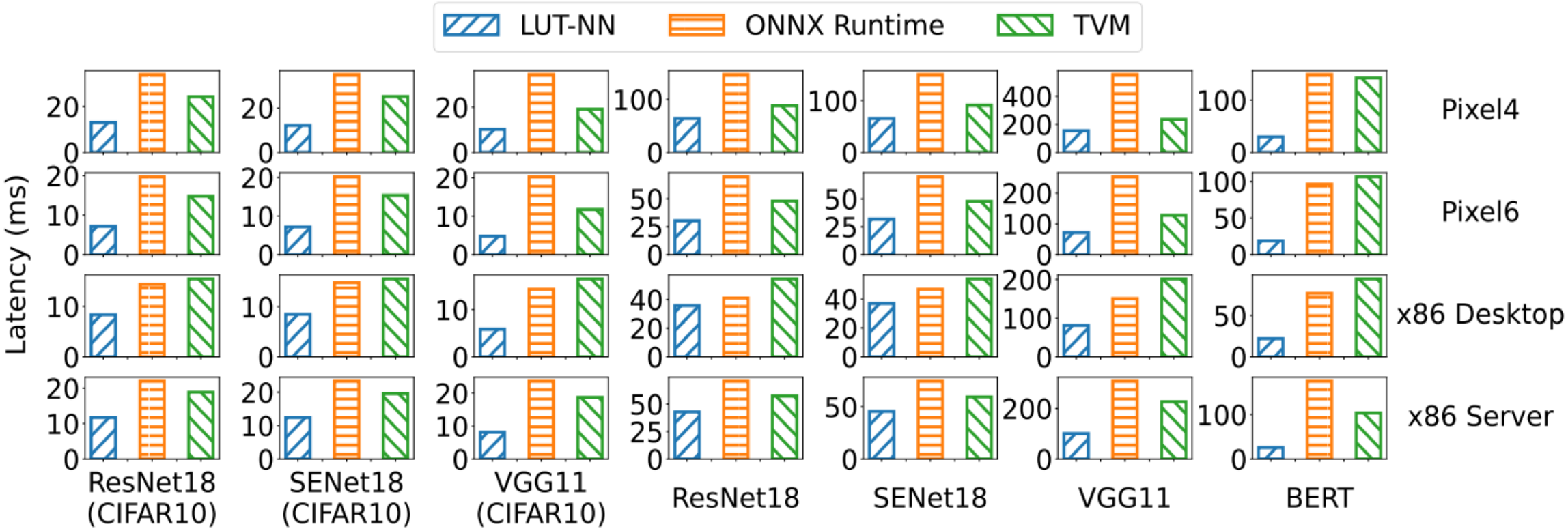
- Accuracy difference compared to original models ranges +1.98% (speech command) to -2.42% (ImageNet).

Model	ResNet18			SENet18			VGG11		
Dataset	LUT-NN	MADDNESS	baseline	LUT-NN	MADDNESS	baseline	LUT-NN	MADDNESS	baseline
CIFAR10	94.40	10.01	95.26	94.48	10.65	95.47	93.89	22.87	95.04
GTSRB	98.73	4.53	98.80	98.36	5.68	98.84	98.55	5.70	99.22
Speech Commands	93.70	1.49	91.72	93.04	1.49	94.36	93.38	1.49	93.11
SVHN	96.00	20.68	96.67	96.22	20.12	96.60	96.23	29.97	96.62
UTKFace	4.91	10.51	5.57	4.74	11.02	5.46	5.69	24.57	5.85
ImageNet	67.38	0.10	69.76	68.21	0.17	70.63	68.04	0.16	68.33

Dataset Task	Single Sentence	Similarity and Paraphrase	Natural Language Inference		
	SST-2	QQP	QNLI	RTE	Average
Training Dataset Size	67k	364k	105k	2.5k	
Test Dataset Size	1.8k	391k	5.4k	3k	
BERT base (%)	93.5	71.2	90.5	66.4	80.4
LUT-NN (%)	92.4	69.6	87.4	64.7	78.5

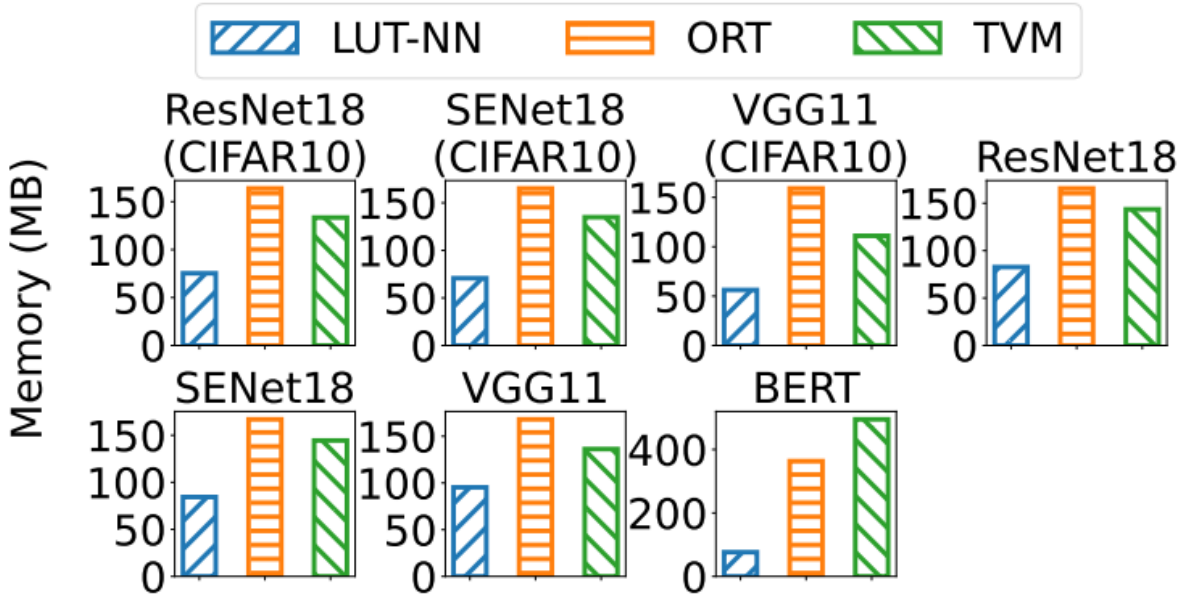
Latency

- Latency reduction is 1.3x (ResNet18) ~ 6.8x (Bert).
  - Bert uses larger weight matrix for matrix multiplications, and longer vector for LUT-NN.



Memory and Power

- Memory saving is 1.4x (VGG11) ~ 6.5x (Bert).



- Power saving is 12% (SeNET18) ~ 42% (Bert).

Model	Method	Power (W)
BERT	LUT-NN	2.6
	TVM	3.7
ResNet18	LUT-NN	2.6
	TVM	3.0
ResNet18 (CIFAR)	LUT-NN	2.6
	TVM	3.3
SENET18	LUT-NN	2.6
	TVM	2.9
SENET18 (CIFAR)	LUT-NN	2.8
	TVM	3.2
VGG11	LUT-NN	2.3
	TVM	2.9
VGG11 (CIFAR)	LUT-NN	2.7
	TVM	3.3