# ENEE-719D Integerization Lecture 7

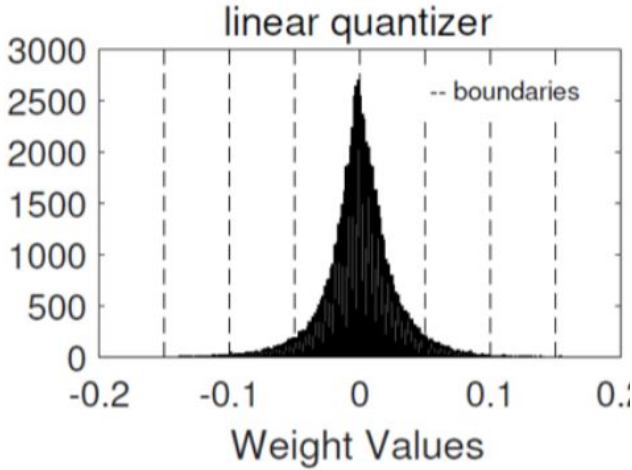
Dr. Ching-Yi Lin

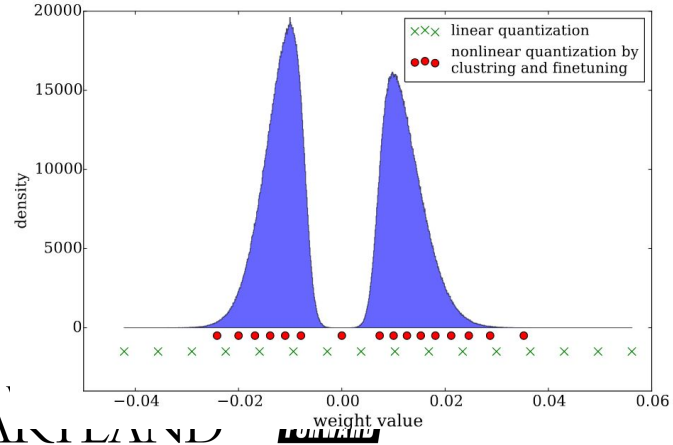


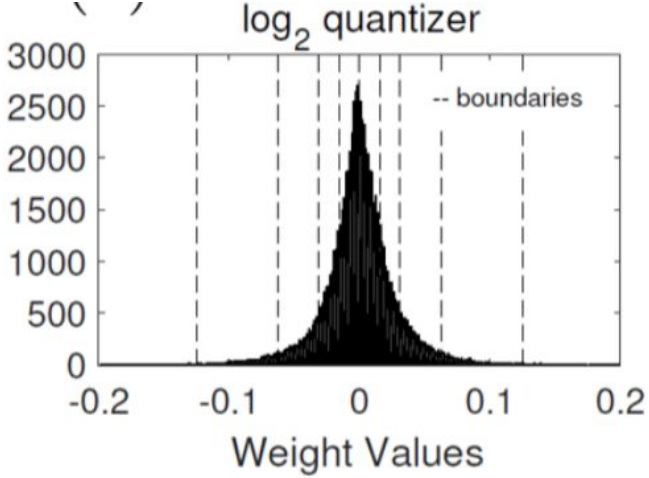



# Recap: Model Quantization

- Linear quantization
- Non-linear quantization
  - Log-quantizer
  - Look-up table based

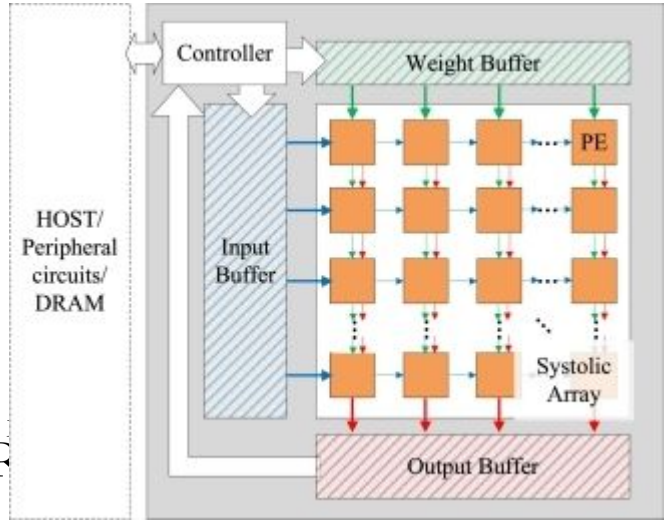


# Recap: Efficient Inference

- MAC operation dominates systolic array
  - MAC = Multiply + Add
  - 1x 32b-multiplier = 16x 8-b multiplier = 32x 32b-adder
- Number of bit in multiplier ↘ Inference efficiency ↗




| Operation: | Energy (pJ) |
|---|---|
| 8b Add | 0.03 |
| 16b Add | 0.05 |
| 32b Add | 0.1 |
| 16b FP Add | 0.4 |
| 32b FP Add | 0.9 |
| 8b Mult | 0.2 |
| 32b Mult | 3.1 |
| 16b FP Mult | 1.1 |
| 32b FP Mult | 3.7 |
| 32b SRAM Read (8KB) | 5 |
| 32b DRAM Read | 640 |

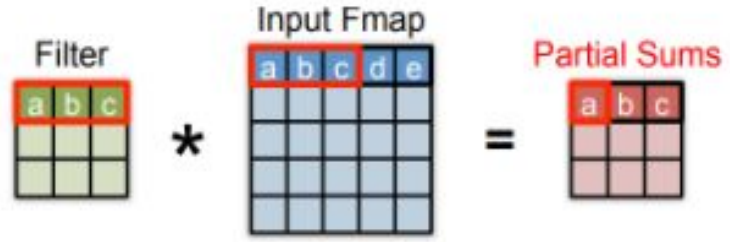


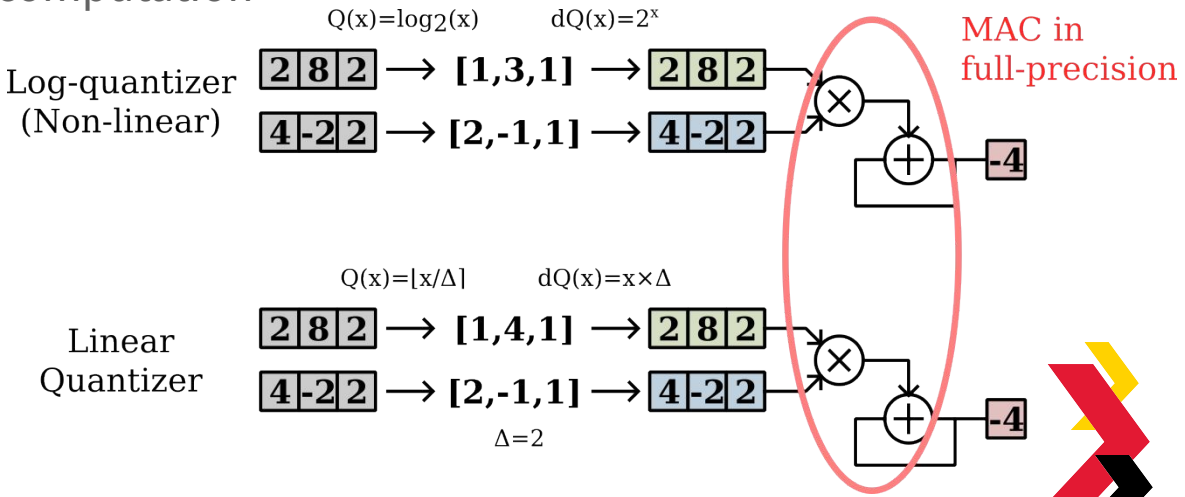

| Area ($\mu m^2$) |
|---|
| 36 |
| 67 |
| 137 |
| 1360 |
| 4184 |
| 282 |
| 3495 |
| 1640 |
| 7700 |
| N/A |
| N/A |



# Quantized Model Inference

- Example of inner product
  - Input X = [2, 8, 2]
  - Weight W = [4, -2, 2]
- Low-bit expression
  - Dequantized before computation
  - Stored in low-bit

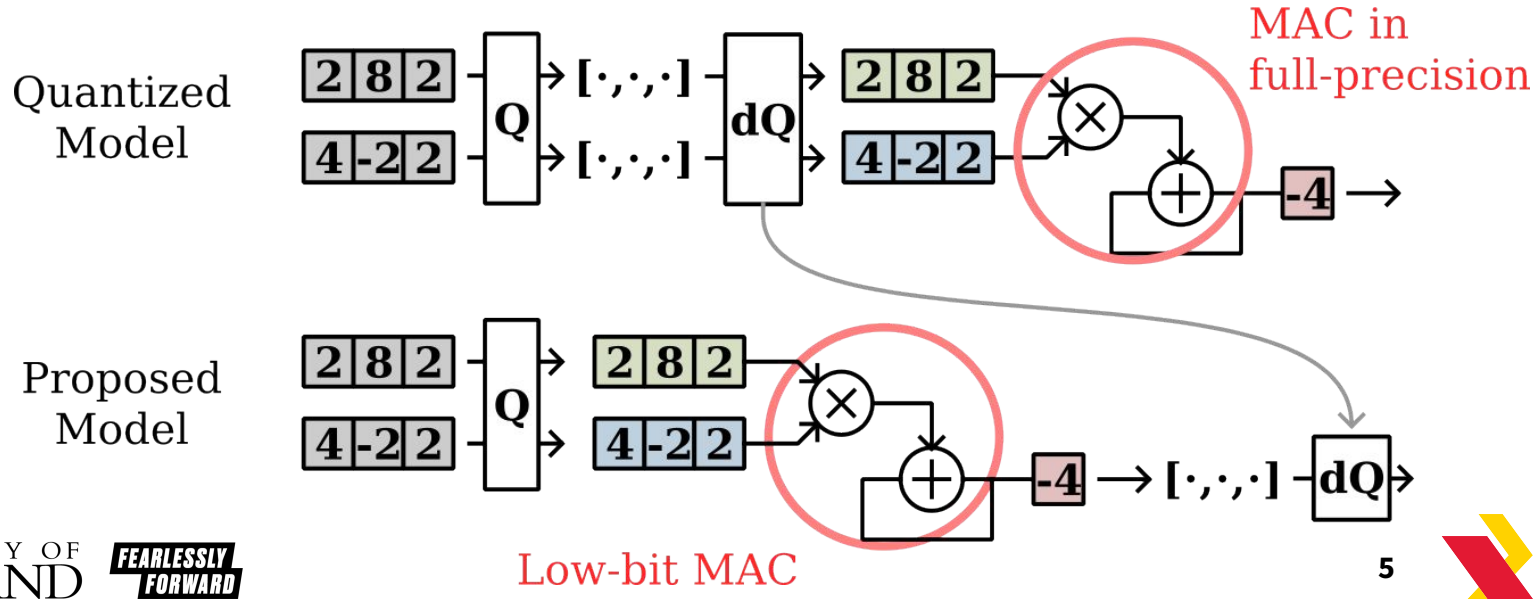



UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

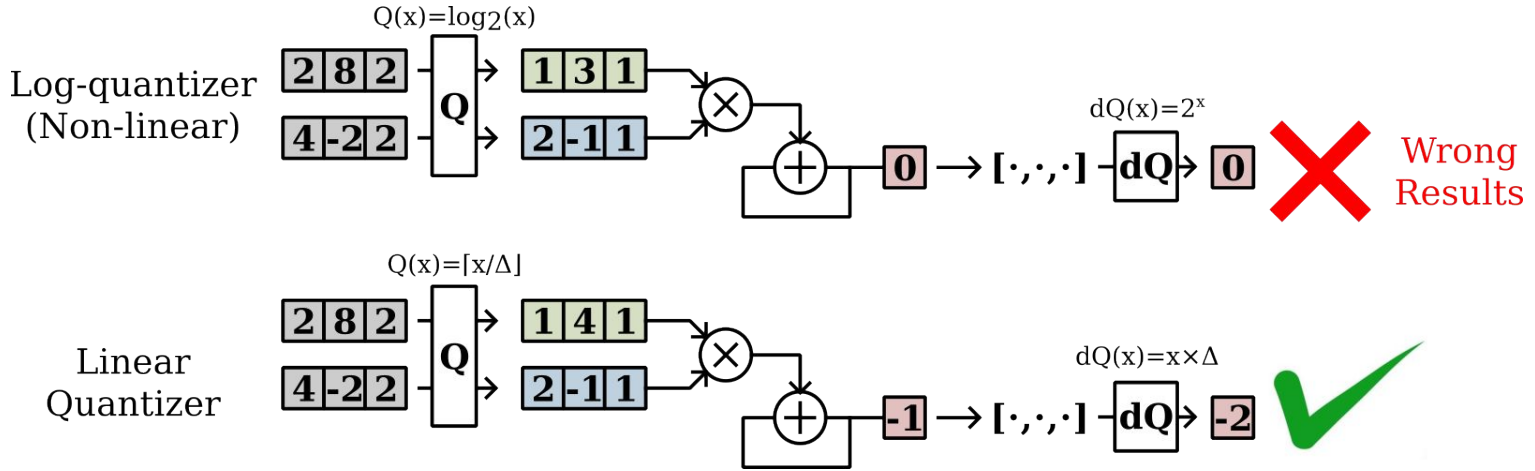
# Integerization Motivation

- Delay dequantization (dQ)
- Make MAC operation in low-precision domain



# Integerization Motivation

- Only few quantizer can swap MAC and dQ
  - Linear quantizer
  - Non-linear quantizer: Wrong results

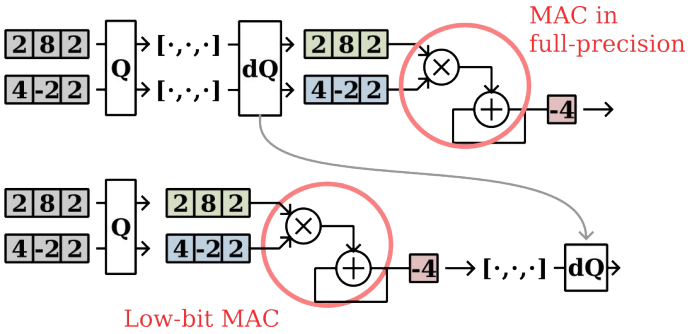


# HW Requirements for Integerization

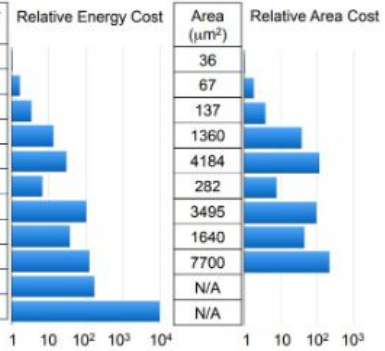
Range

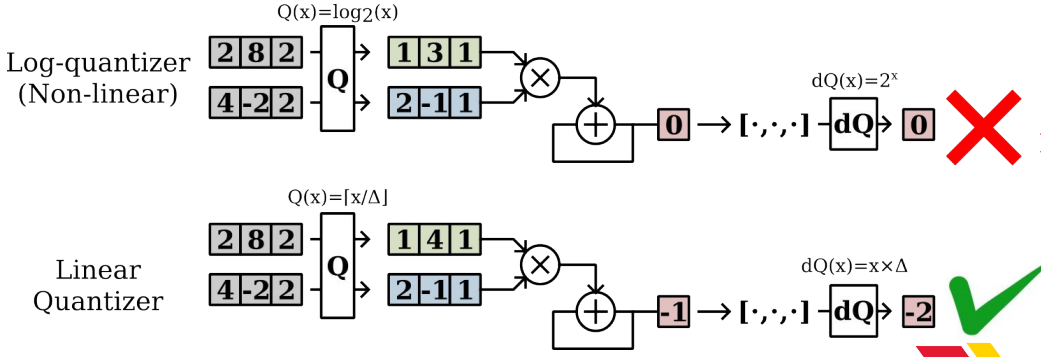
# Part 1

- From quantization to integerization
- Integerization + Neural network
- Integerization + Matrix multiplication
- Model integerization research



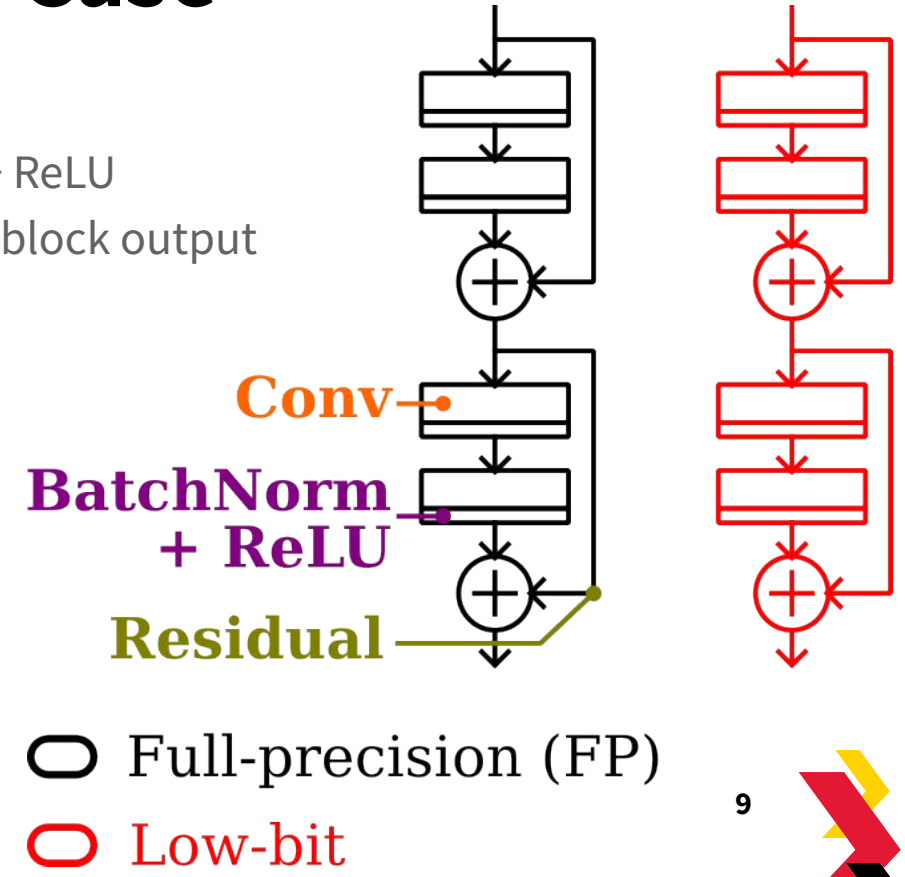

| Operation: | Energy (pJ) |
|---|---|
| 8b Add | 0.03 |
| 16b Add | 0.05 |
| 32b Add | 0.1 |
| 16b FP Add | 0.4 |
| 32b FP Add | 0.9 |
| 8b Mult | 0.2 |
| 32b Mult | 3.1 |
| 16b FP Mult | 1.1 |
| 32b FP Mult | 3.7 |
| 32b SRAM Read (8KB) | 5 |
| 32b DRAM Read | 640 |



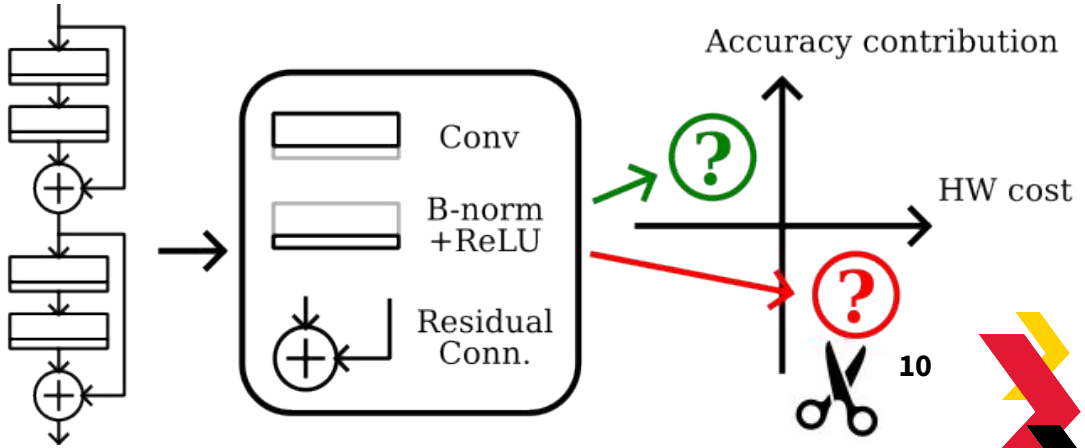
# Integerized NN Ideal Case

- Example model: ResNet-50
  - Each block = Conv + batchnorm + ReLU
  - Residual connection: Merge with block output
- Ideal integerized model
  - Low-bit convolution
  - Low-bit batchnorm + ReLU
  - Low-bit residual connection



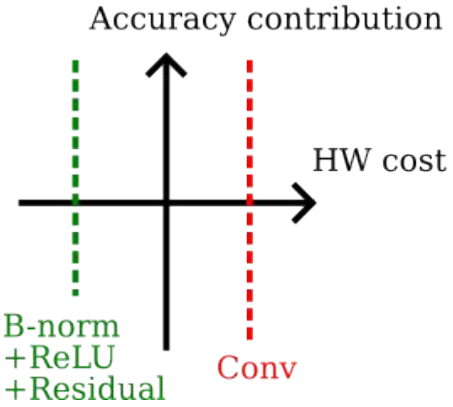
# Hardware-efficient Integerized NN

- Split a model into
  - HW expensive + Accuracy-insensitive modules
  - HW inexpensive + Accuracy sensitive modules
- Integerization strategy
  - Integerize accuracy-insensitive modules

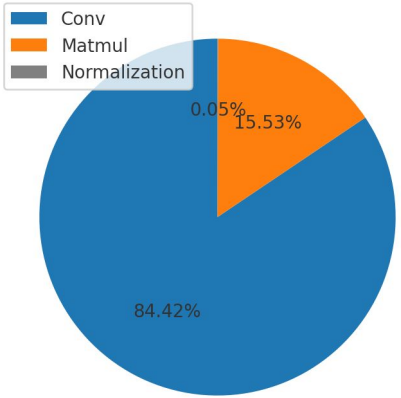


# Hardware Cost Breakdown

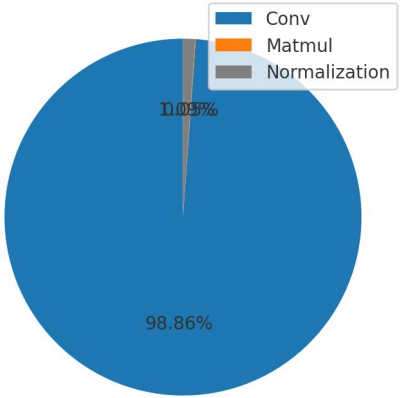
- Conv and matmul dominates HW cost
  - Complexity = $O(N^3)$
- Batchnorm + ReLU + Residual = $O(N^2)$



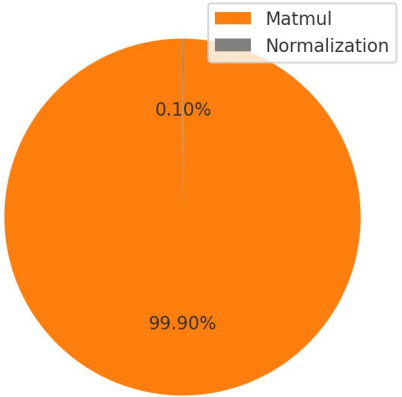

AlexNet MACs Breakdown



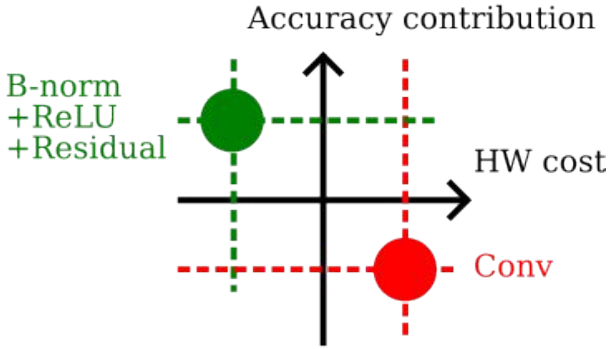

ResNet-50 MACs Breakdown



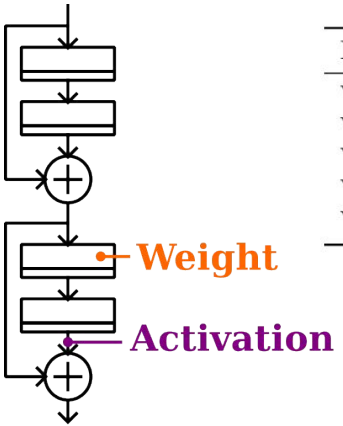

ViT-B/16 MACs Breakdown



# Activation is Important




- Conv and matmul dominates HW cost
- Accuracy: W8A16 > W16A8
  - Activation is more acc-sensitive than weight

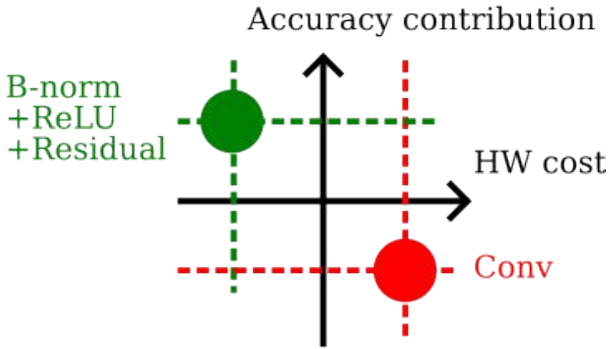


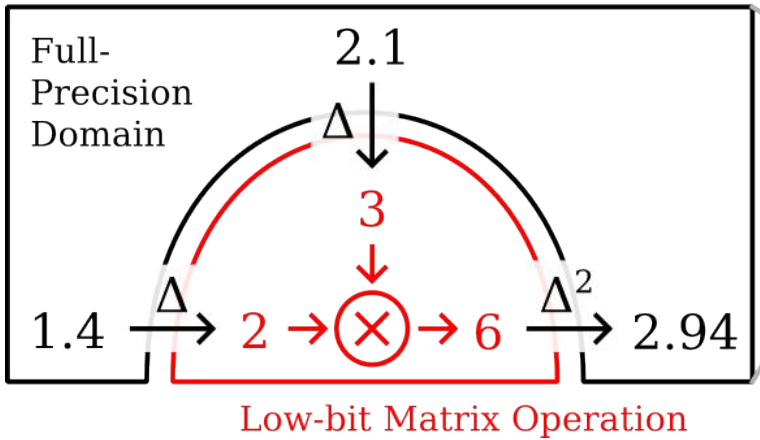

| Precision | Lambada (↑) | PIQA (↑) | OpenBookQA (↑) | RTE (↑) | ReCoRd (↑) | Ave. 19 Tasks (↑) | Wikitext-2 (↓) |
|---|---|---|---|---|---|---|---|
| W16A16 | 49.3 | 66.3 | 29.4 | 53.8 | 75.1 | 38.9 | 21.5 |
| W8A16 | 49.3 | 66.1 | 29.6 | 54.2 | 74.8 | 38.5 | 22.1 |
| W16A8 | 44.7 | 64.8 | 28.2 | 52.7 | 69.2 | 37.8 | 24.6 |
| W8A8 | 42.6 | 64.1 | 28.0 | 53.1 | 67.5 | 37.8 | 26.2 |
| W4/8A16 | 0.00 | 51.4 | 30.2 | 52.7 | 16.1 | 28.9 | 1.76e5 |

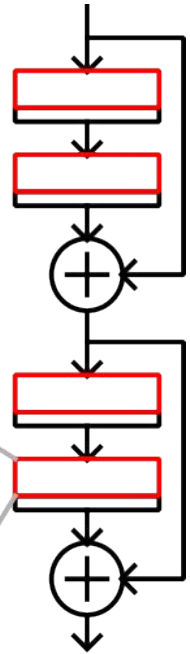
# Full-precision Connection + Q-Matmul

- Conv: HW expensive + Acc-insensitive
- Batchnorm + Activation:

  HW inexpensive + Acc-sensitive
- MAC in low-bit (FP-in, FP-out)

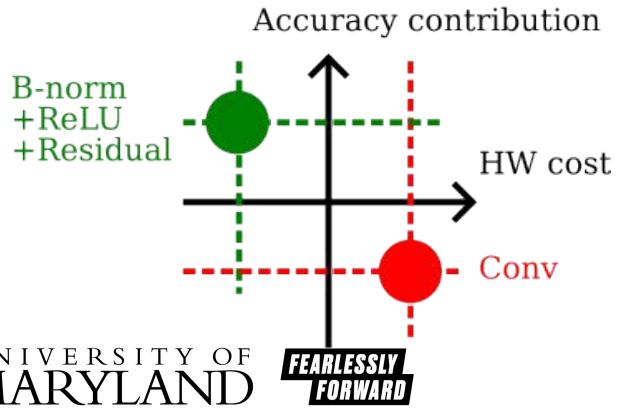


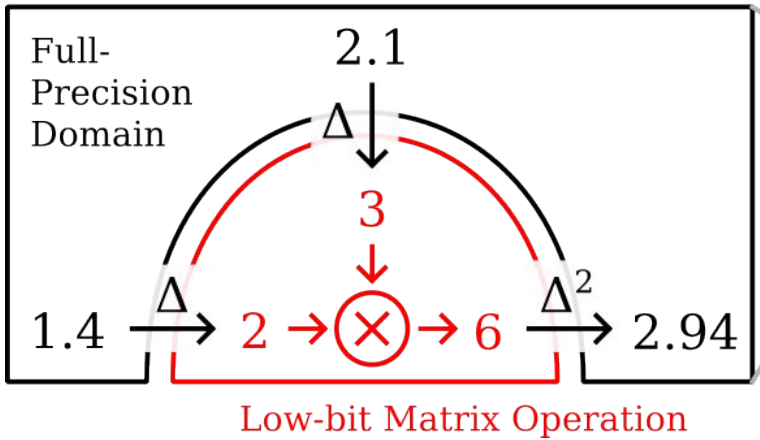



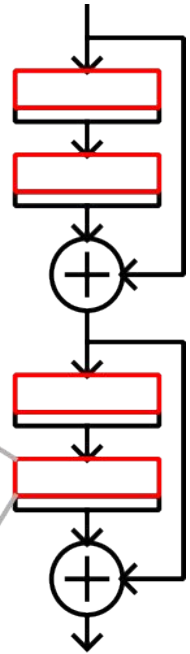



# Part 2

- From quantization to integerization
- Integerization + Neural network
- Integerization + Matrix multiplication
- Model integerization research



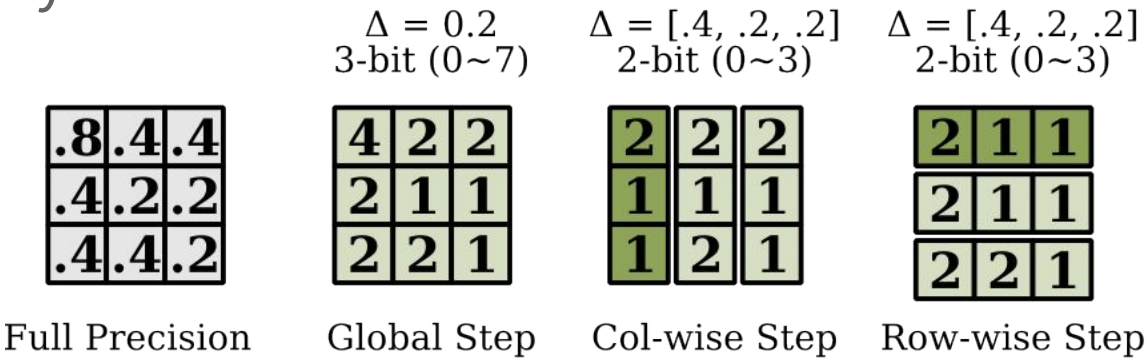



# From Number to Matrix

Quantization granularity

- Global
- Channel-wise
  - Column-wise
  - Row-wise




- Accuracy: Channel-wise > Global
- Size efficiency: Channel-wise > Global

# 2x2 Matrix Multiplication

$$\begin{bmatrix} .1 & .2 \\ .2 & .4 \end{bmatrix} \times \begin{bmatrix} 8 & -4 \\ -2 & 4 \end{bmatrix} = \begin{bmatrix} .4 & .4 \\ .8 & .8 \end{bmatrix}$$

- Each element in output matrix
  - Inner product between input row and weight column
    - $O_{ij} = \Sigma_k (X_{ik} \times W_{kj}) = X_{i1} \times W_{1j} + X_{i2} \times W_{2j}$

$$\begin{bmatrix} .1 & .2 \end{bmatrix} \times \begin{bmatrix} 8 \\ -2 \end{bmatrix} = (.1 \times 8 + .2 \times -2) = .4$$

- If input $(x_{i1}, x_{i2})$ have same $\Delta x$ and weight $(w_{1j}, w_{2j})$ have same $\Delta w$
  - $\Delta x$ and $\Delta w$ can be factored out
  - Low-bit matrix multiplication ($\Delta x = 0.1$, $\Delta w = 2$)

$$[\Delta x \times \begin{bmatrix} 1 & 2 \end{bmatrix}] \times [\Delta w \times \begin{bmatrix} 4 \\ -1 \end{bmatrix}] = \Delta x \times \Delta w \times 1 \times 4 + \Delta x \times \Delta w \times 2 \times -1 = \Delta x \times \Delta w \times 2$$

# Quantization Scope

$$\begin{bmatrix} .1 & .2 \\ .2 & .4 \end{bmatrix} \times \begin{bmatrix} 8 & -4 \\ -2 & 4 \end{bmatrix} = \begin{bmatrix} .4 & .4 \\ .8 & .8 \end{bmatrix}$$

- Global Δx and Δw can be factored out

$$[\Delta x \times \begin{bmatrix} 1 & 2 \end{bmatrix}] \times [\Delta w \times \begin{bmatrix} 4 \\ -1 \end{bmatrix}] = \Delta x \times \Delta w \times 1 \times 4 + \Delta x \times \Delta w \times 2 \times -1 = \Delta x \times \Delta w \times 2$$

Same

- Different Δ in the same MAC cannot be merged
  - The weight has row-wise weight [$\Delta w_1$, $\Delta w_2$, …]
  - In this example, $\Delta w_1$=4 and $\Delta w_2$=2

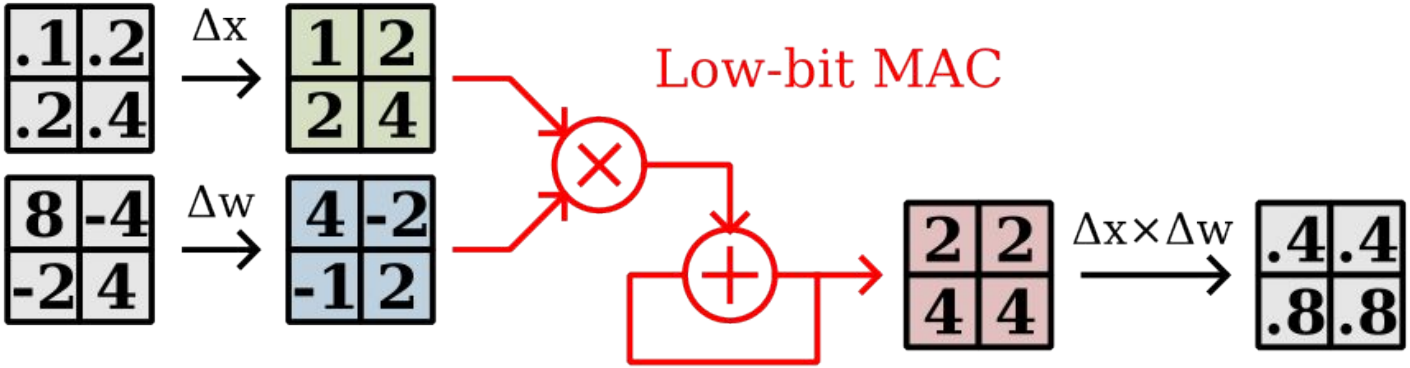
$$\begin{bmatrix} 8 & -4 \\ -2 & 4 \end{bmatrix} \xrightarrow{\overrightarrow{\Delta w}} \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \quad \Delta w_1=4 \quad \Delta w_2=2$$

$$[\Delta x \times \begin{bmatrix} 1 & 2 \end{bmatrix}] \times [\begin{pmatrix} \Delta w_1 \\ \Delta w_2 \end{pmatrix} \times \begin{bmatrix} 2 \\ -1 \end{bmatrix}] = \Delta x \times \Delta w_1 \times 1 \times 2 + \Delta x \times \Delta w_2 \times 2 \times -1 \neq \Delta x \times \Delta w \times 0$$

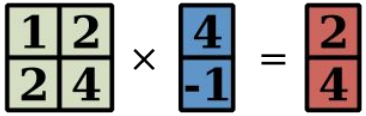
Different

# Global Input + Global Weight

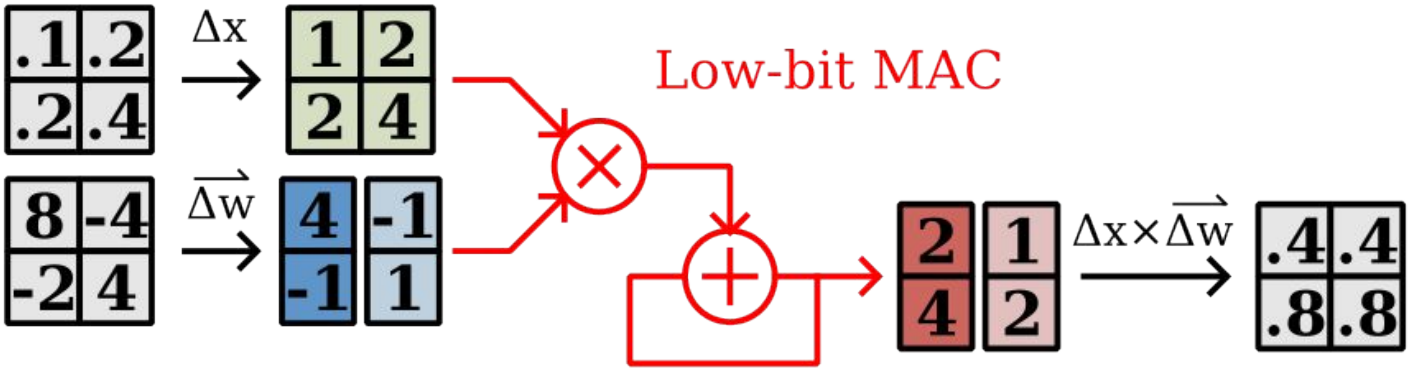
- Global input step size Δx + Global weight step size Δw
- MAC = Low-bit MUL + Full-precision (FP) ADD
- 1× Full-precision scalar scaling (Δx×Δw)



# Global Input + Col-wise Weight

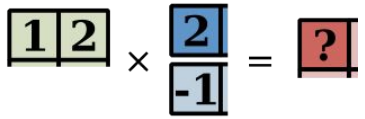



- Global input step size Δx + column-wise weight step size $\Delta\vec{w}$
- MAC = Low-bit MUL + Full-precision (FP) ADD
- 1× Full-precision vector scaling ($\Delta x \times \Delta\vec{w}$)
-

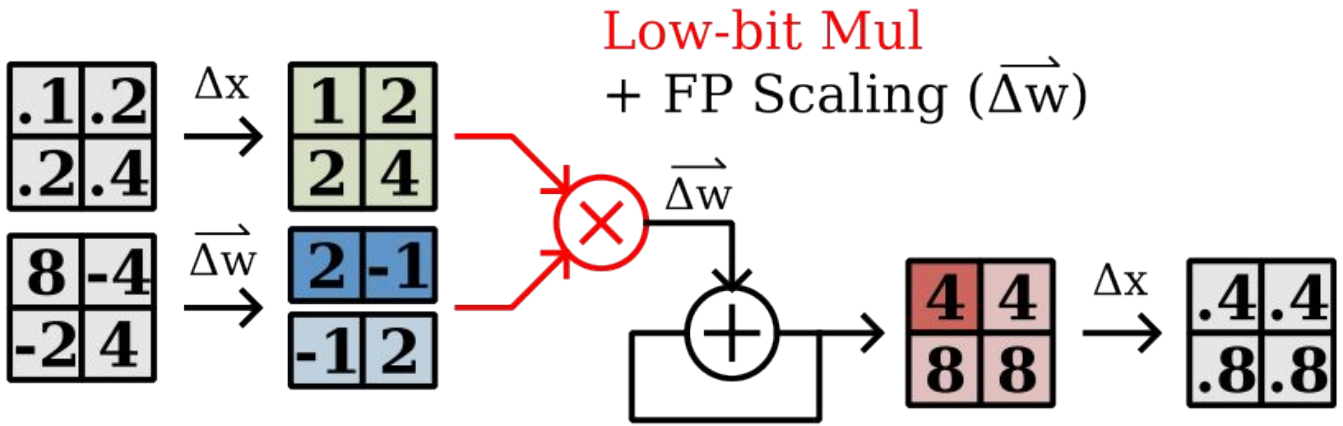


Non-integerized

# Global Input + Row-wise Weight

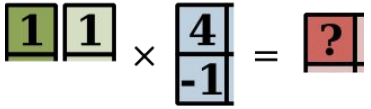



- Global input step size Δx + row-wise weight step size $\vec{\Delta w}$
- MAC = Low-bit MUL + FP scaling ($\vec{\Delta w}$) + FP ADD
- 1× Full-precision scalar scaling (Δx)

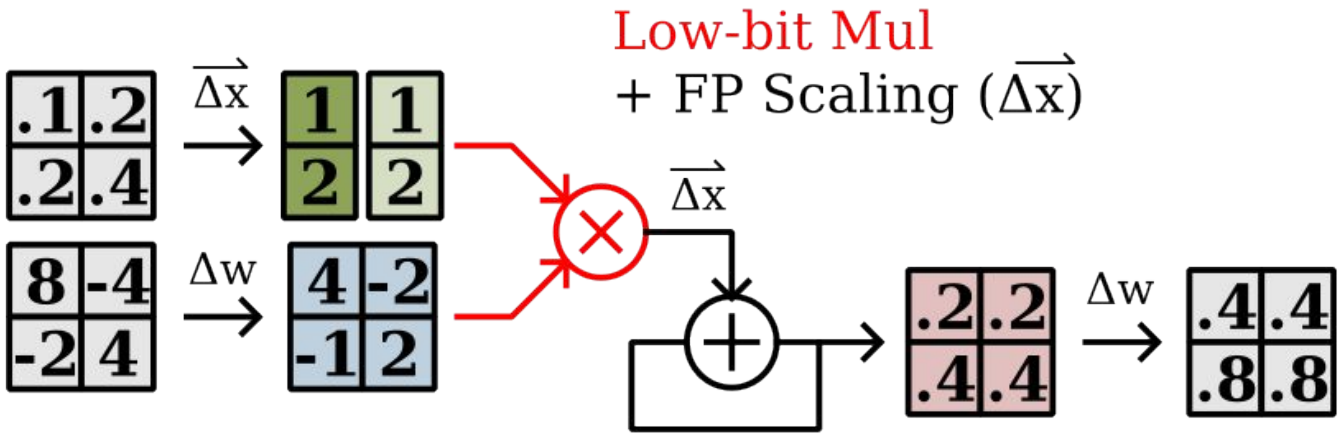


Non-integerized

# Col-wise Input + Global Weight



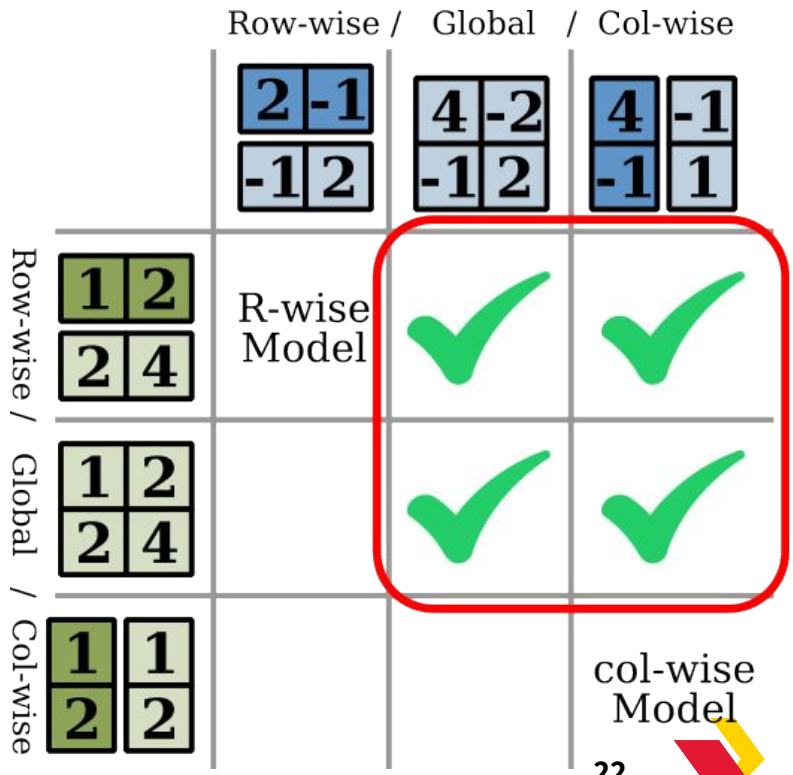

- Col-wise input step size $\vec{\Delta x}$ + Global weight step size $\Delta w$
- MAC = Low-bit MUL + FP scaling ($\vec{\Delta x}$) + FP ADD
- 1× Full-precision scalar scaling ($\Delta w$)



# All Possible Quantization Scheme

- 9 possible quantization schemes
- Schemes able to integerize
  - Input not column-wise
  - Weight not row-wise

- Most common schemes
  - Row-wise model
    - Row-wise input + row-wise weight
  - Column-wise model
    - Col-wise input + col-wise weight

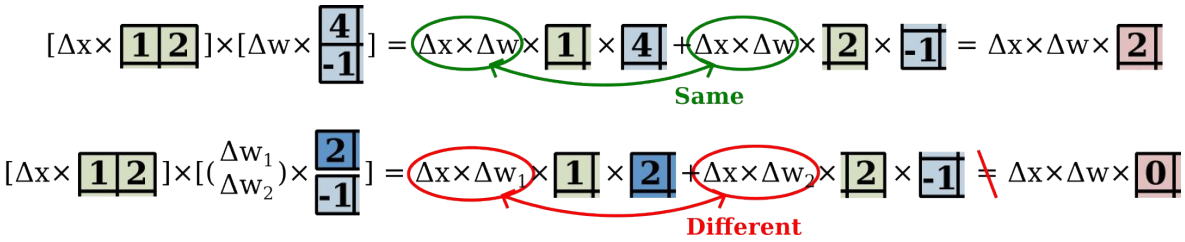


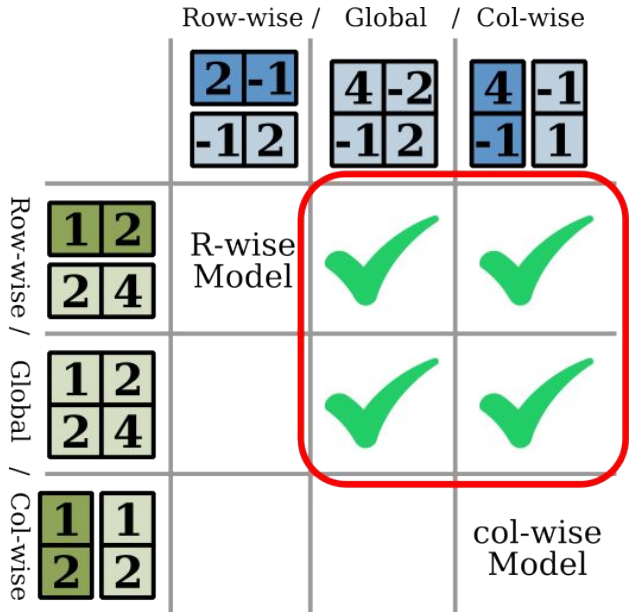



UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

# Part 3

- From quantization to integerization
- Integerization + Neural network
- Integerization + Matrix multiplication
- Model integerization research



# Unsigned or Signed Integerization

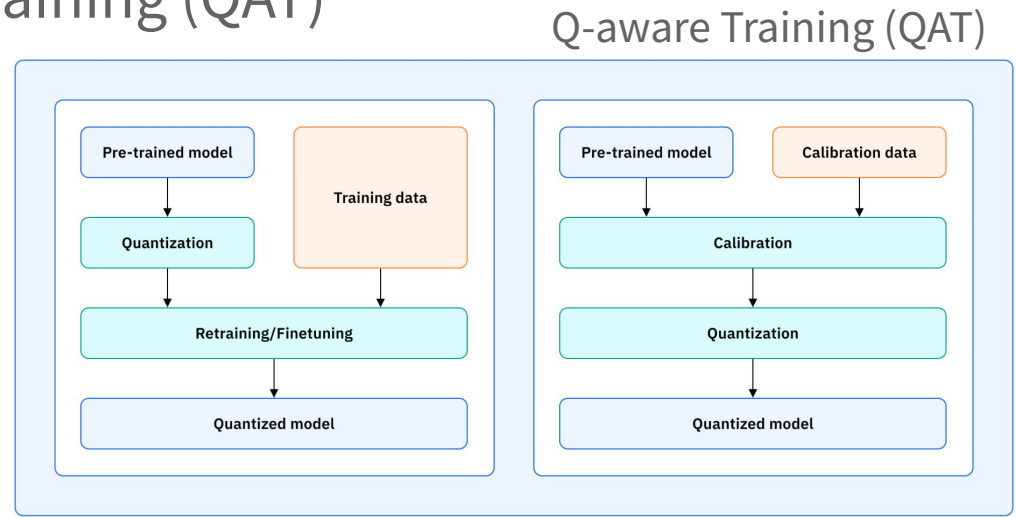
- Range between unsigned and signed
- ReLU and tanh range





# How to train integerized Networks

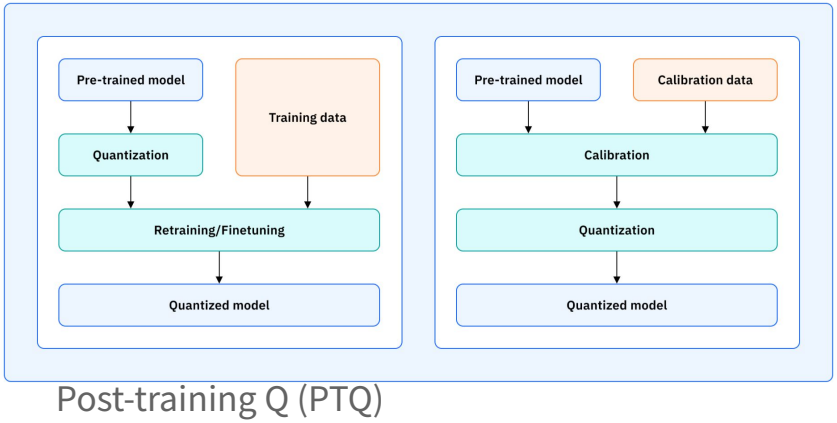
- Post-training Quantization (PTQ)
  - Finetuning from pre-trained models
- Quantization-aware Training (QAT)
  - Train from scratch



# PTQ vs QAT

- Speed (Training time)
  - Finetuning > Re-training
  - PTQ > QAT
- Final accuracy
  - QAT > (PTQ + Full-precision model)
    - Full-precision model is very different to integerized model

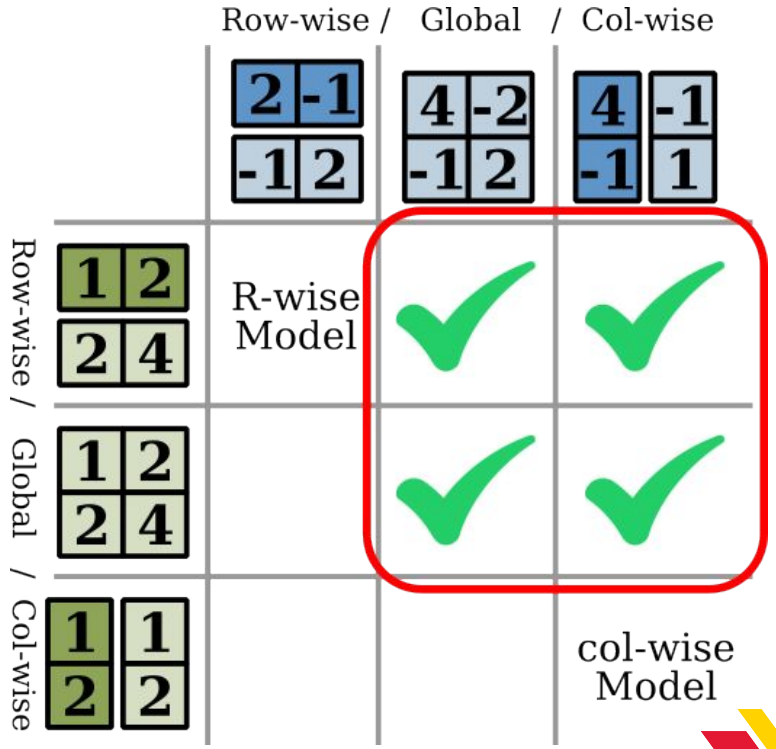
- What if we have a closer pretrained model?



# From Quantized to Integerized

Current quantized model

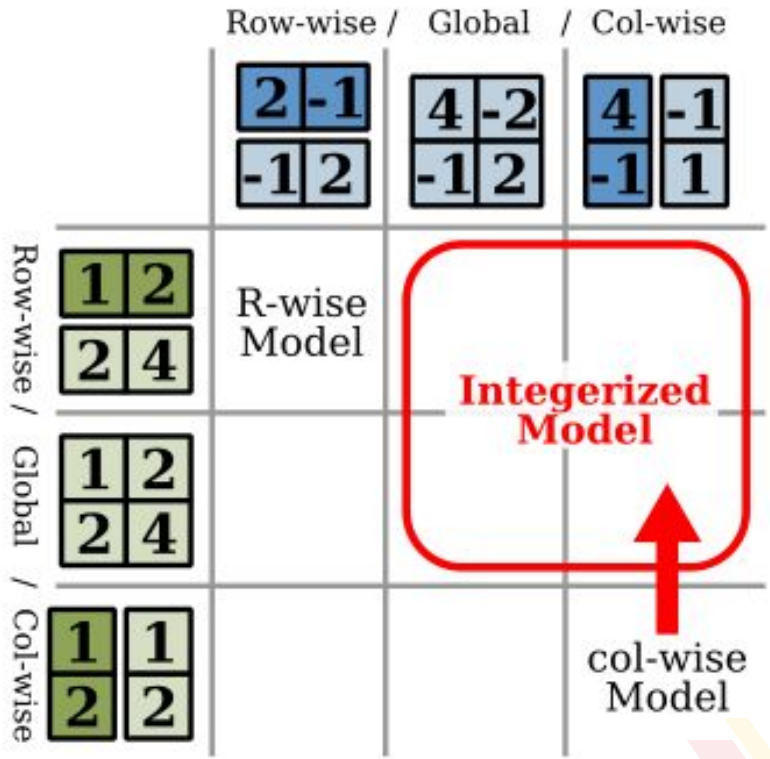
1. Col-wise model
    a. Col-wise Input
    b. Col-wise Weight
2. Row-wise model
    a. Row-wise Input
    b. Row-wise Weight



# Integerize a Quantized Model

Quantize a column-wise model

- Keep weight col-wise
- Make input global quantized
  - Easier than col-wise -> row-wise



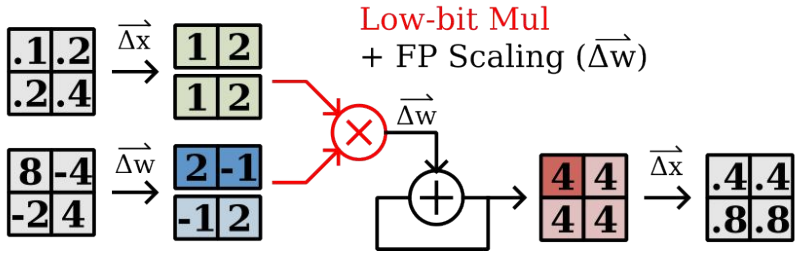


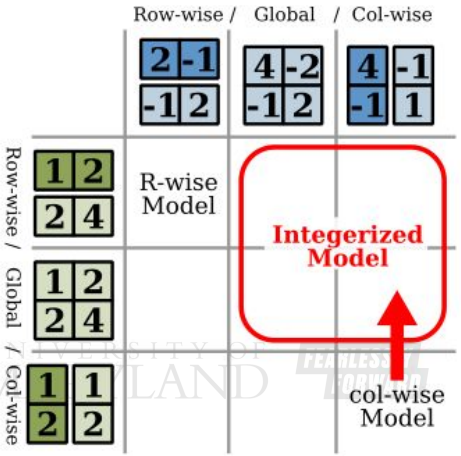

UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

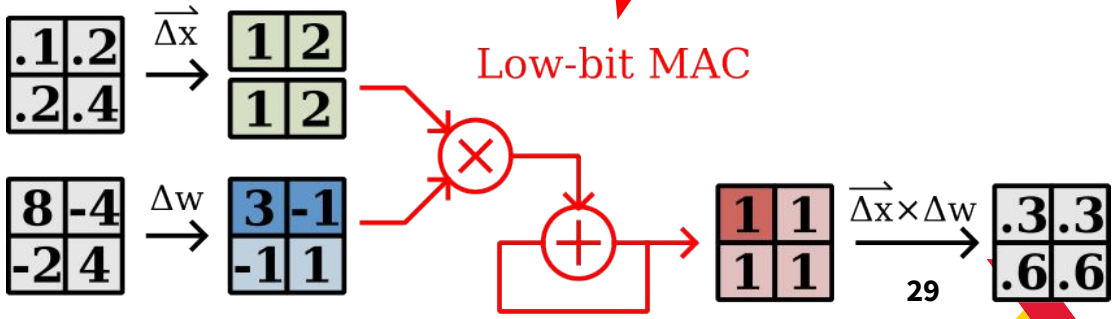
# Integerizing a Quantized Model

From [row-wise $\vec{\Delta x}$ + row-wise $\vec{\Delta w}$] to [row-wise $\Delta x$ + global $\Delta w$]

- $\vec{\Delta w} = [4, 2]$ -> $\Delta w = 3$
- $\vec{\text{Result}}$ with error -> Finetune

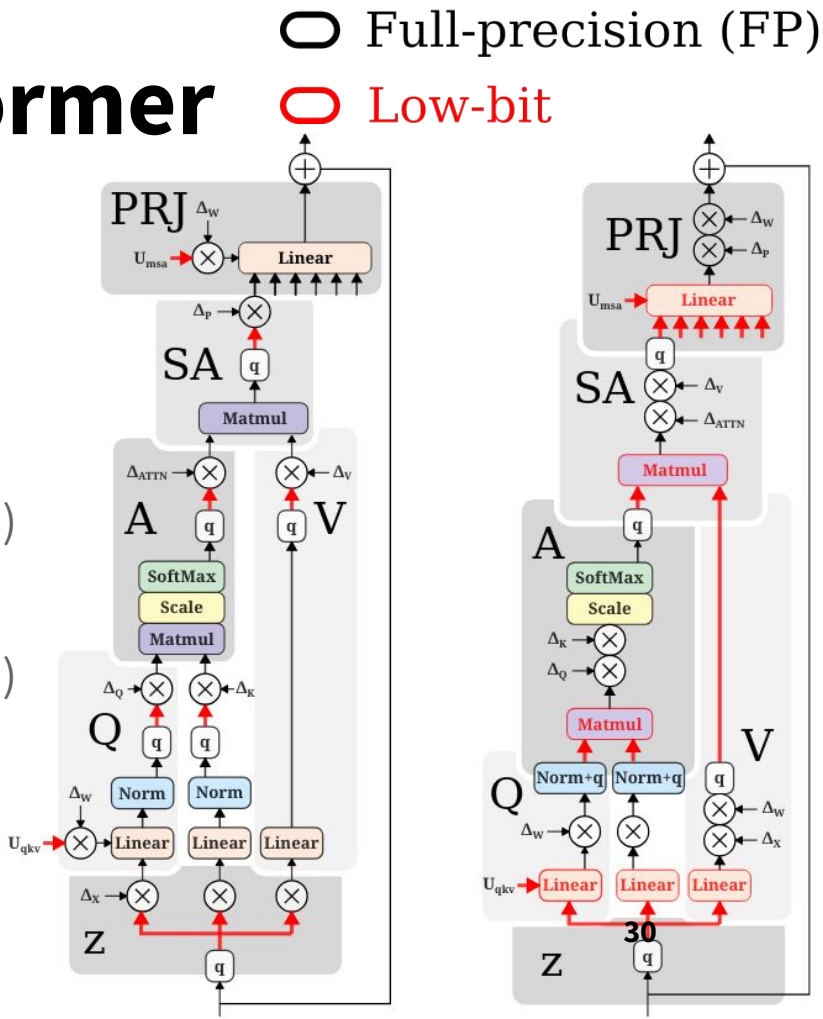


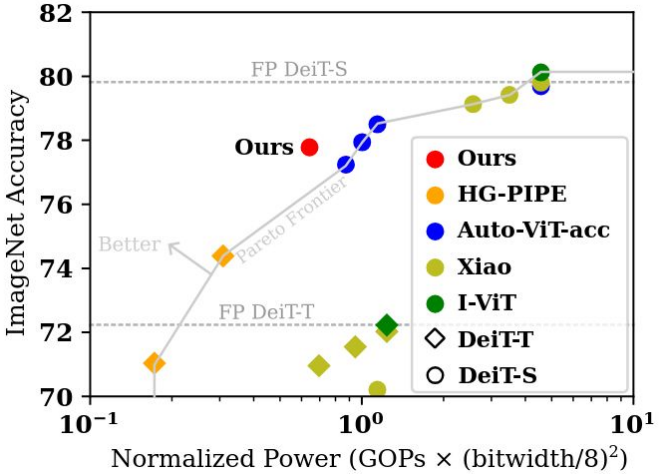
# From Matrix to Transformer

- HW expensive modules:
  - Linear and Matmul
- Quantized model inference
  - Q(q) -> dQ(×) -> Linear/Matmul -> $Q_2$(q)
- Integerized model inference
  - Q(q) -> Linear/Matmul -> dQ(×) -> $Q_2$(q)
  - Merge $dQ_1$ and $Q_2$



# Results – 3b Vision Transformer

- Best acc-pwr tradeoff
- Lower-bit is good

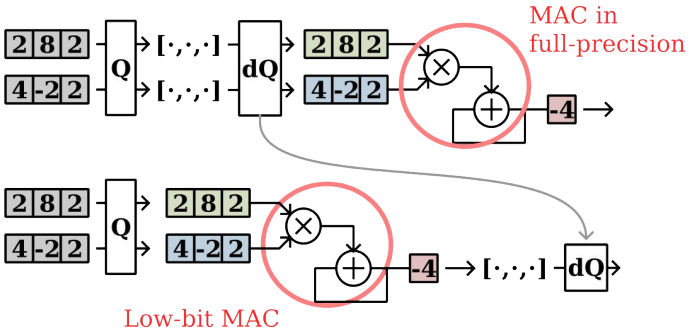


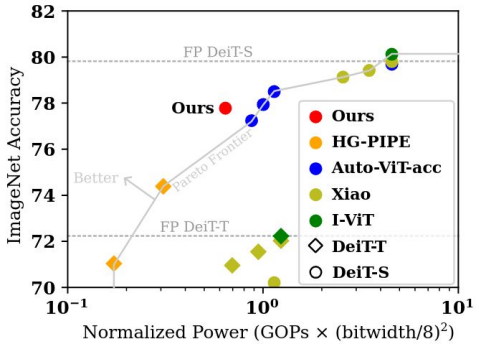

| | Board | Tech | Model | OP Type | clk (MHz) | Power (W) | Peak GOPs/s | Actual GOPs/s | GOPs/s/W (GOPs/J) |
|---|---|---|---|---|---|---|---|---|---|
| ViA [35] | U50 | 16nm | Swin-T | FP16 | 300 | 39 | - | 309.6 | *7.9 |
| DFX [36] | U280 | 16nm | 345M (GPT-2) | FP16 | - | 45 | - | 184.1 | *4.1 |
| Fan [37] | VCU128 | 16nm | FABNet | FP16 | 200 | 11.4 | *128.0 | *128.0 | *11.3 |
| Ye [20] | U250 | 16nm | Rush [38] | INT8 | 300 | 77.2 | 4915 | 1800 | *23.3 |
| Huang [19] | ZCU102 | 16nm | ViT-T | INT8 | 300 | - | 1229 | 616.1 | - |
| | | | ViT-S | | 300 | 29.6 | 1229 | 762.7 | *25.8 |
| HeatViT [39] | ZCU102 | 16nm | DeiT-S | | - | 10.7 | - | *221 | 20.6 |
| | | | DeiT-B | | - | 11.4 | - | *425 | 37.4 |
| Lu [40] | VU13P | 16nm | Transformer (MHA) | INT8 | 200 | 16.7 | 819 | *629 | 37.7 |
| Ftrans [41] | VCU118 | 16nm | RoBERTa base | INT16 | - | 25.1 | - | *1077 | *43.0 |
| Zhang [22] | ZCU9EG | 16nm | ViT-S | FP8 | 300 | *21.4 | - | *1150 | *53.7 |
| | | | ViT-B | | | *21.4 | - | *1254 | *58.6 |
| ME-ViT [24] | U200 | 16nm | ViT-B | INT8 | 150 | 17.8 | - | *1149 | *64.5 |
| | | | | | 300 | 31.8 | - | *2298 | *72.2 |
| Calabash [21] | VU9P | 16nm | BERT | INT16 | 243 | 12.9 | *995 | 880 | *68.2 |
| | ZCU102 | | | | 300 | **6.6** | *614 | 530 | *80.3 |
| Xiao [34] | U250 | 16nm | Deit-T | Custom INT5 | 183 | 40.1 | | 1488 | *37.1 |
| | | | DeiT-S | | 192 | 28.4 | | 2861 | *100.8 |
| | | | DeiT-B | | 179 | 25.3 | | 2332 | *92.4 |
| Auto-ViT-acc [23] | ZCU102 | 16nm | DeiT-S | FP3.5 +FP4 / FP3.5 | 150 | 10.3 | - | *711 | *68.8 |
| | | | DeiT-B | | | 11.0 | - | *996 | *90.3 |
| | | | DeiT-S | | | 6.6 | - | *689 | *105.2 |
| | | | DeiT-B | | | 8.1 | - | *989 | *122.0 |
| HG-PIPE [33] | ZCU102 / VCK190 | 16nm | DeiT-T | FP4 / FP3 | 375 | 21.9 | - | 1974 | *90.1 |
| | | | | | **425** | 43.4 | - | 4536 | *104.5 |
| | | | | | **425** | 46.7 | - | 8898 | *190.5 |
| | | | DeiT-S | | 350 | 48.1 | - | 6854 | *142.5 |
| Ours | U250 | 16nm | 3b DeiT-S (MHA) | INT3 | 400 | 61.83 | **42515** | **13568** | **219.4** |

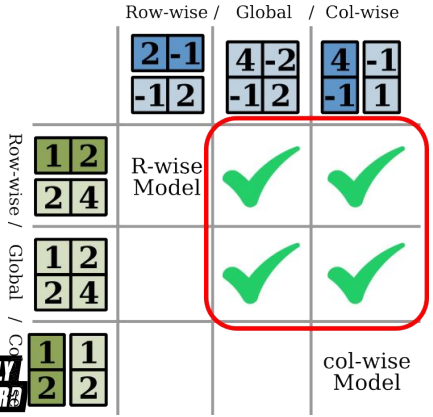
UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

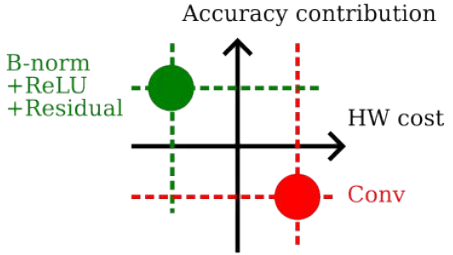
# Last Slide

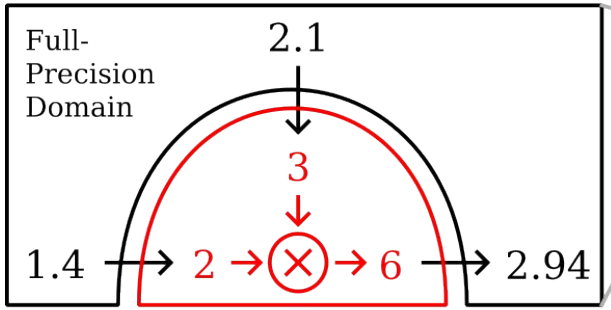
- From quantization to integerization
- Integerization + Neural network
- Integerization + Matrix multiplication
- Model integerization research

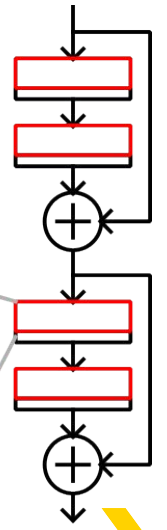


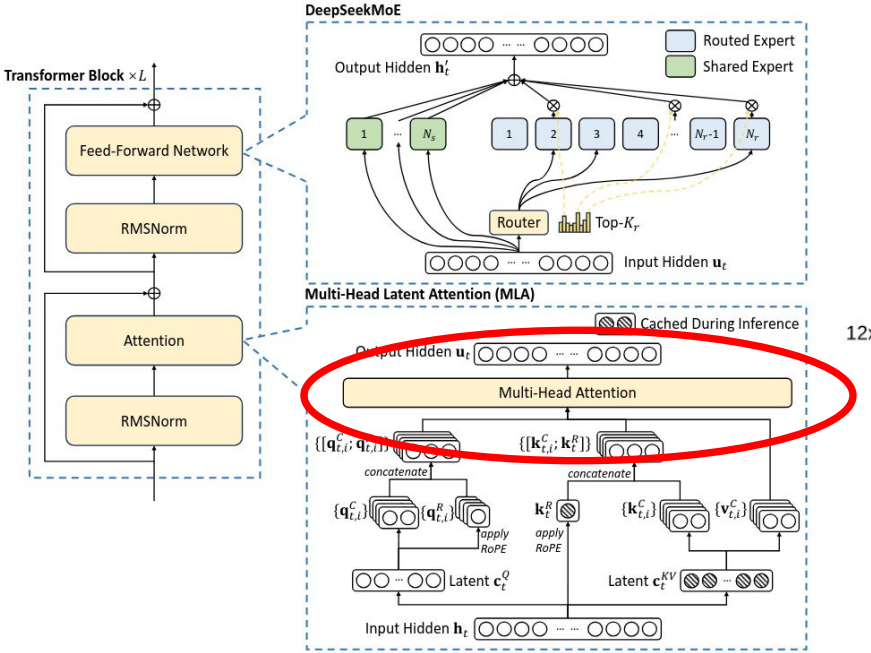



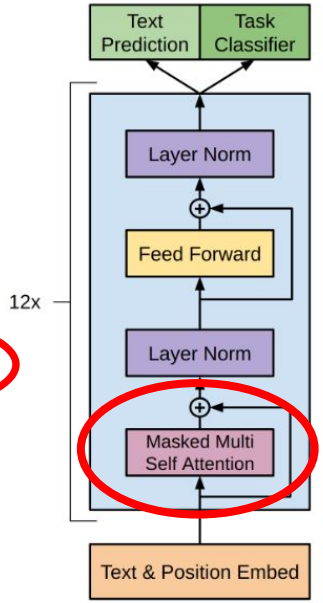
UNIVERSITY OF MARYLAND

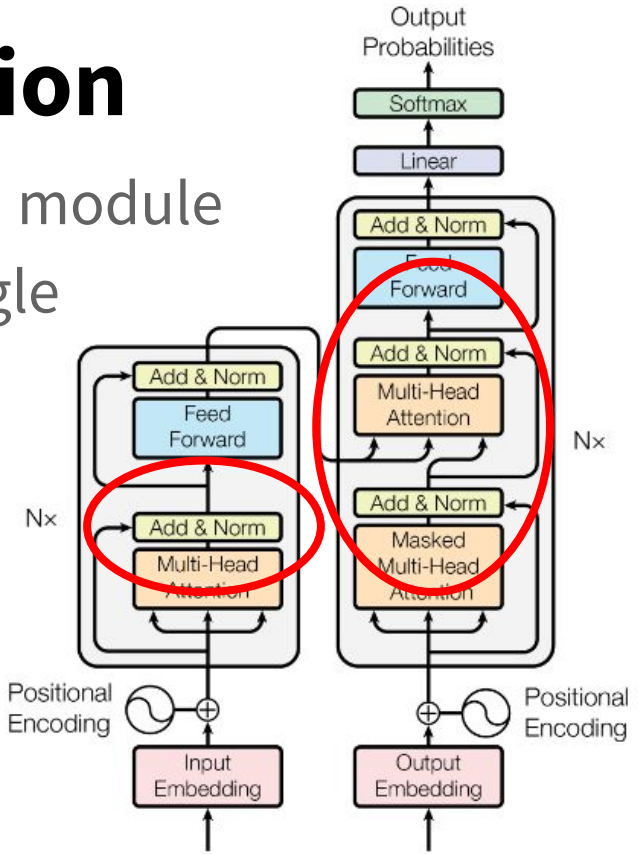
FEARLESSLY FORWARD

# Attention Attention Attention

- All transformer models have attention module
- (L->R) DeepSeek V3 -> Open AI -> Google

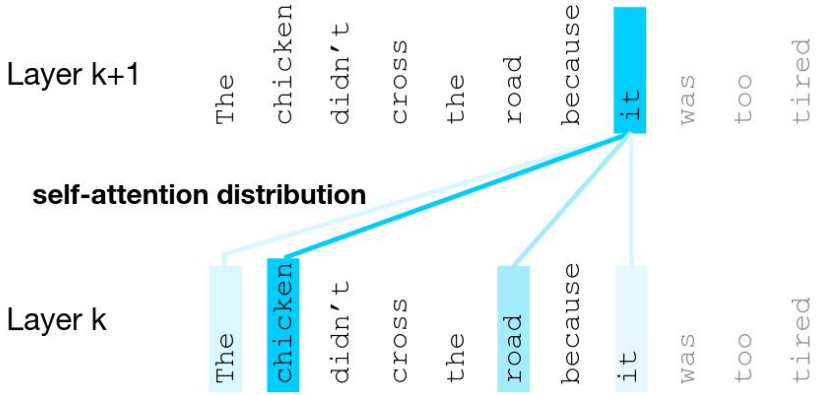


# Motivation of Attention

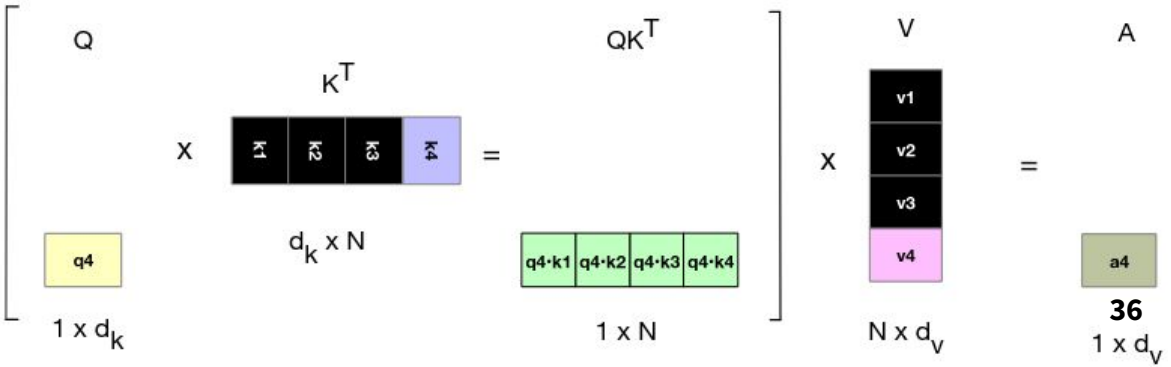
- Word-correspondence within a sentence
- The chicken didn't cross the road because it was too tired
  The chicken didn't cross the road because it was too wide
- Weighted-token relationship



# Two-setp Attention Mechanism

- Each token (word) has their query (Q), key (K), and value (V)

1. Similarity calculated between query and key
   - Inner product between current query (q4) and all keys (k1, k2, k3, and k4)
   - Attention score is a matrix with size 1x4
2. Value (V) is weighted through attention score ($QK^T$)

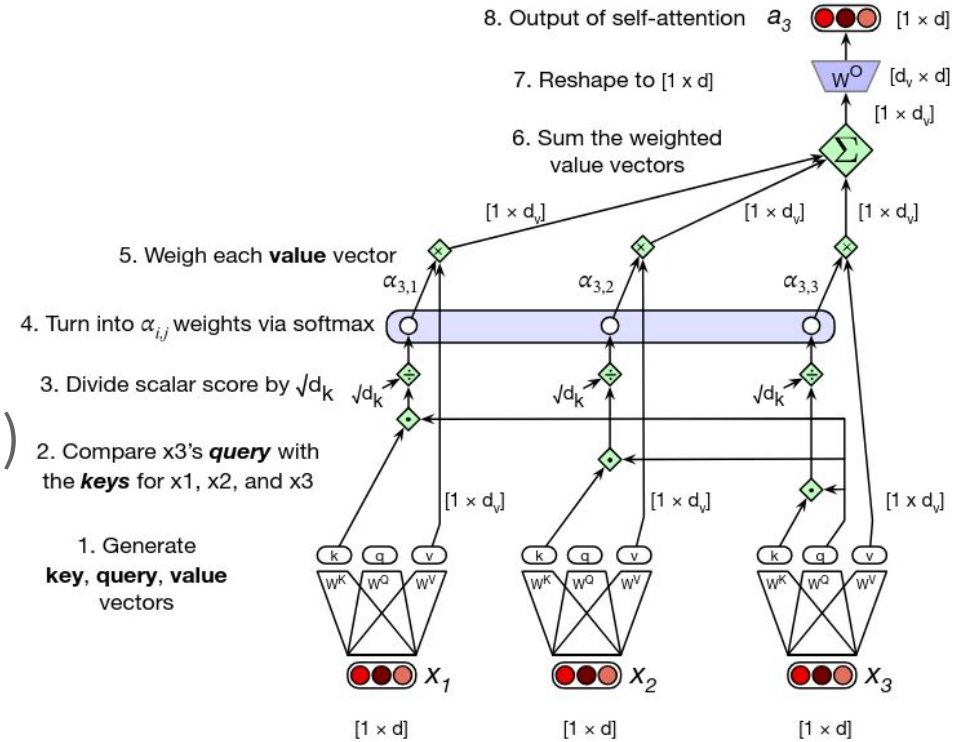



UNIVERSITY OF MARYLAND
FEARLESSLY FORWARD

# Attention Computation

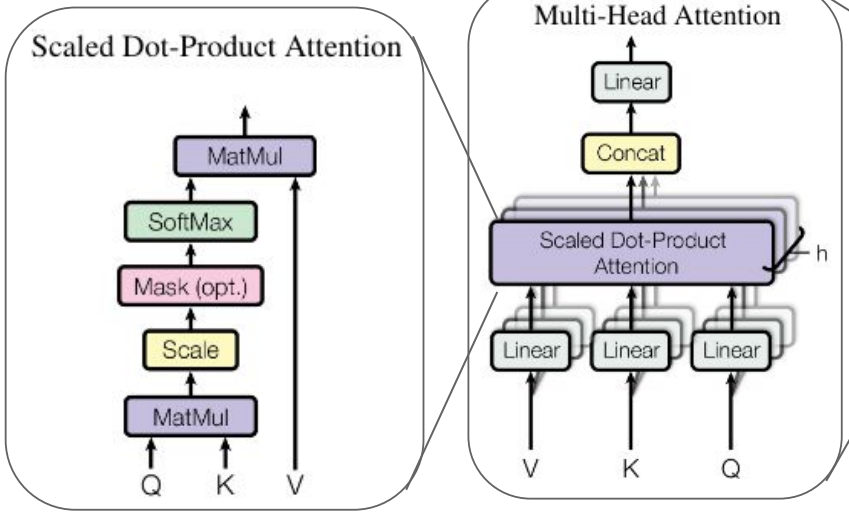
1. Q/K/V from linear layer
2. Q3 queries to K1, K2, K3
3. Attention matrix (Q3K1, Q3K2, Q3K3)
4. Softmax to prob matrix ($\alpha$)
5. Weight v1, v2, v3
6. $\alpha$1v1 + $\alpha$2v2 + $\alpha$3v3

Two matrix multiplication



# Build Transformer from Attn Module

- Each scaled attention = Two matmul (QK and $\alpha$V)
- Each multi-head attention (MHA) = h parallel scaled attn
- Each transformer blk = 1x/2x MHA + Feed-forward
  - Feed-forward = Multi-layer perceptron



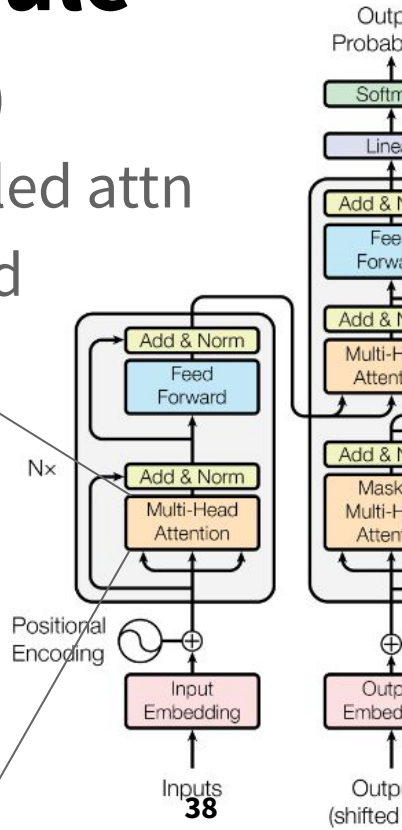



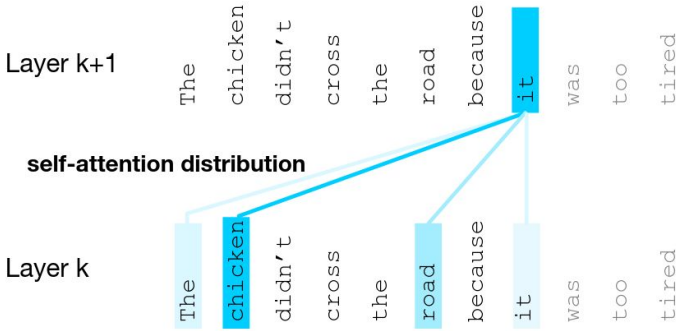

Figure 1: The Transformer - model

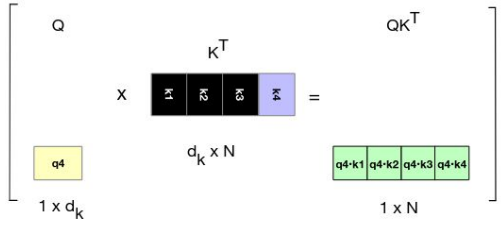


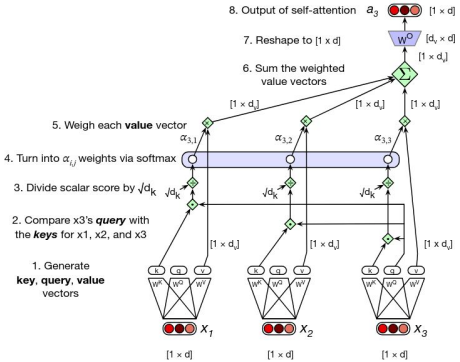
UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

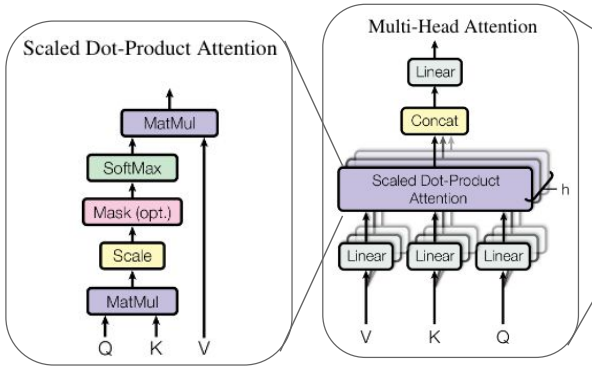
# Quarter 1 Done. Any Question?

1. What is attention? What is transformer?
2. Transformer usage
3. Transformer in hardware
4. Generalized transformer



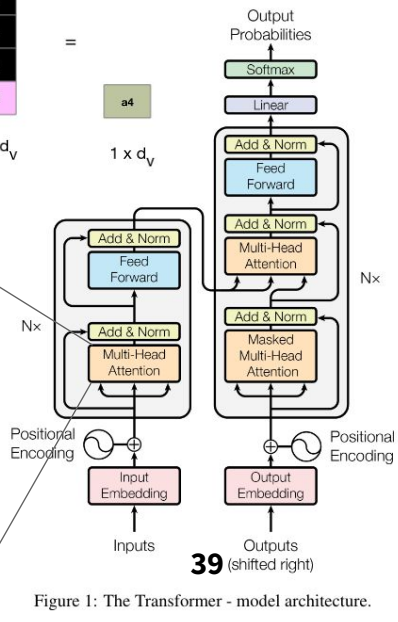



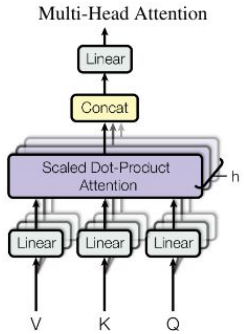

Figure 1: The Transformer - model architecture.



UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

# Interpreting Transformer Model

- Stacking transformer layers
- Encoder (Left)
  - Encode context as keys and values for 2nd MHA
  - Keep reusing in the future inference
- Decoder (Right)
  - Feed input. Generate output
  - Provide query in 2nd MHA
  - 1st MHA = self-attention module



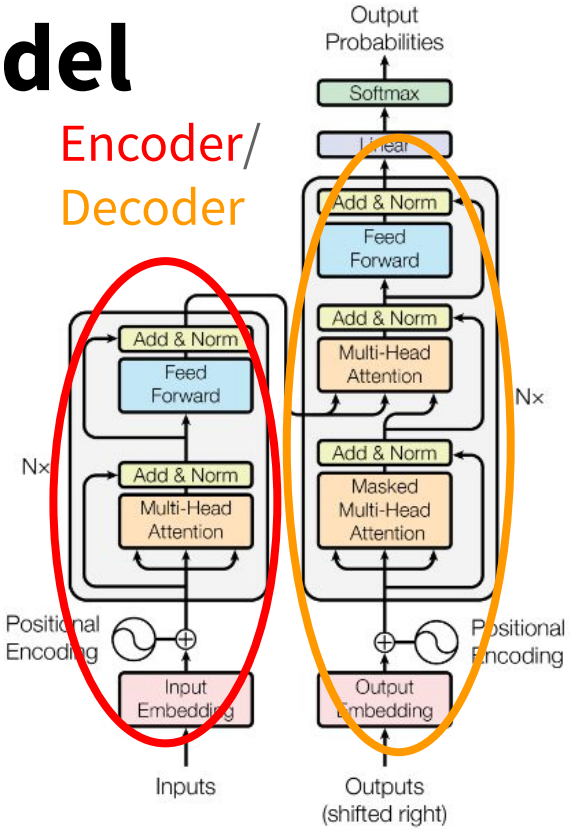



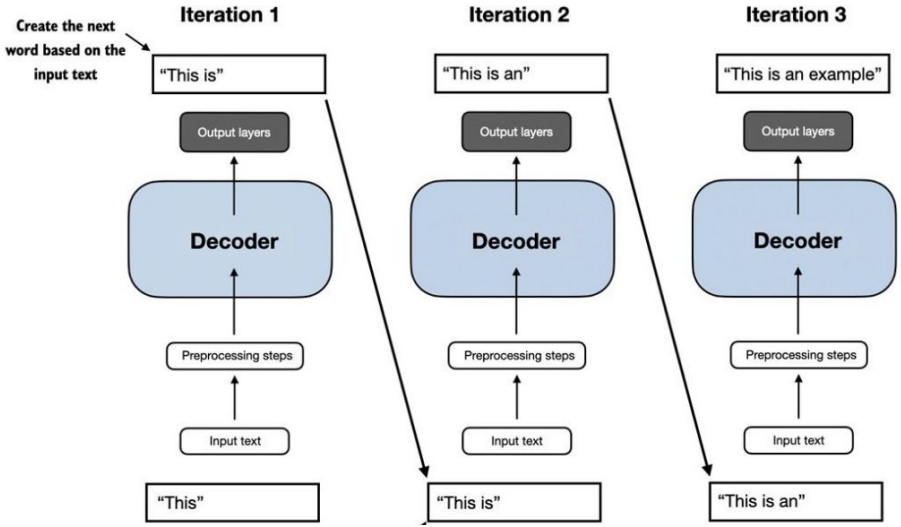

Figure 1: The Transformer - model architecture.

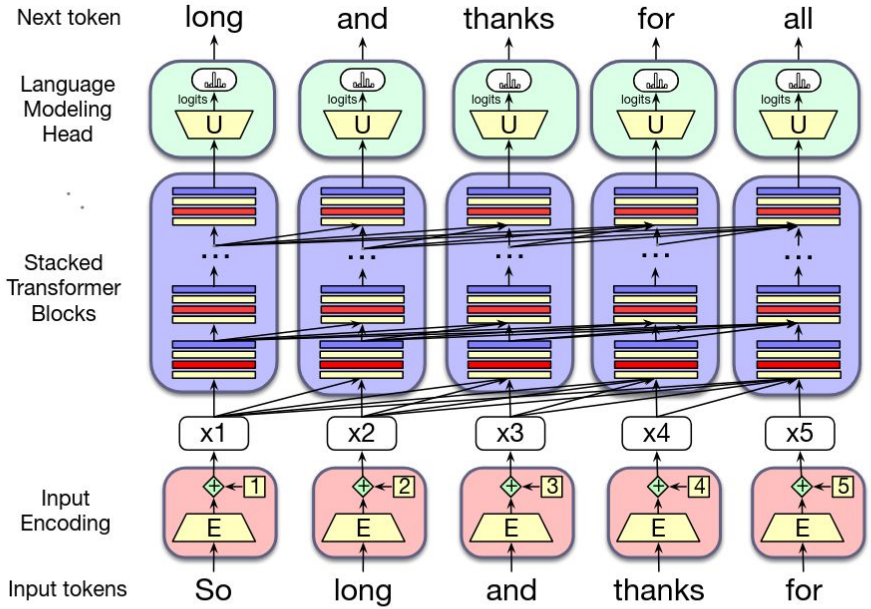


UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

# Next-token Prediction

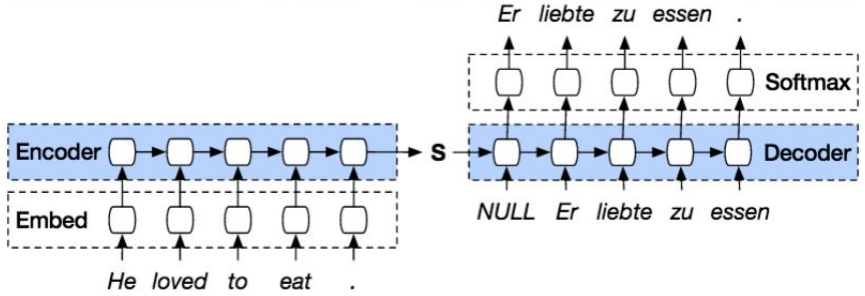
- Each transformer inference generates one token
- Decoder only

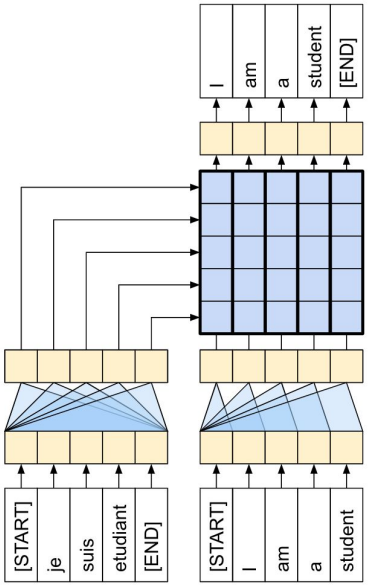


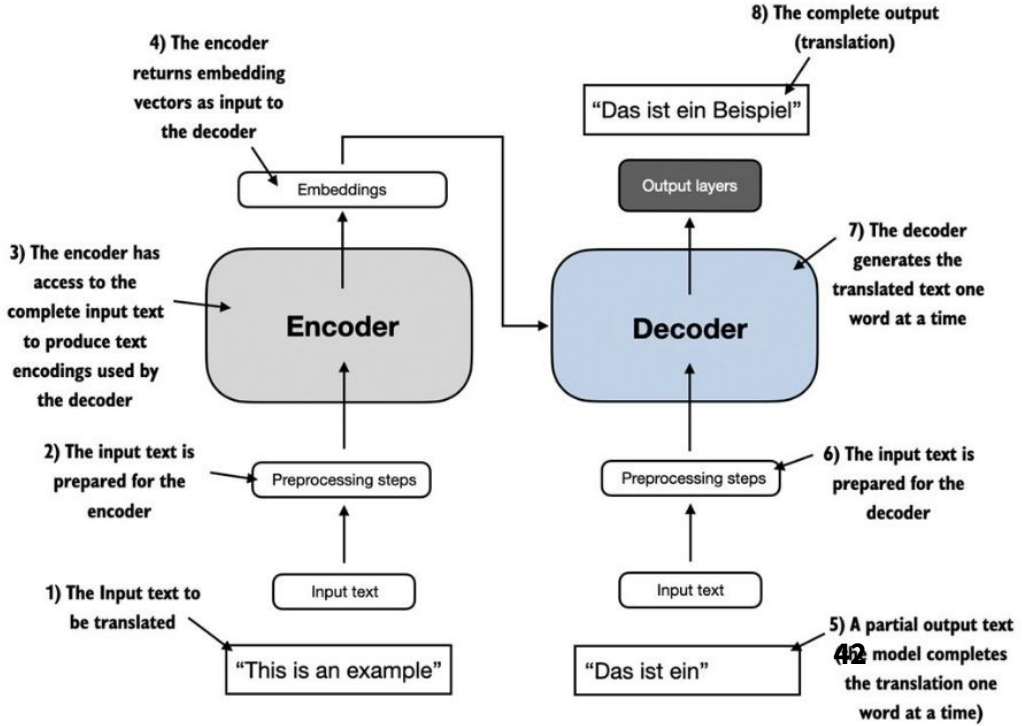



# Machine Translation

- Sequential token generation
- Original texts as context



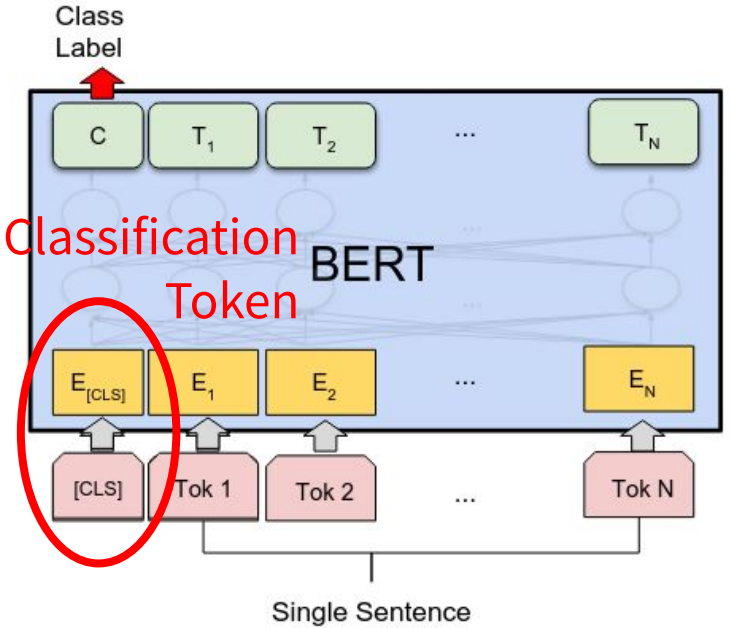



# Single Sentence Classification

- One special token: [CLS]
- Corpus of Linguistic Acceptability (CoLA)
  - 1: Acceptable, 0: Unacceptable

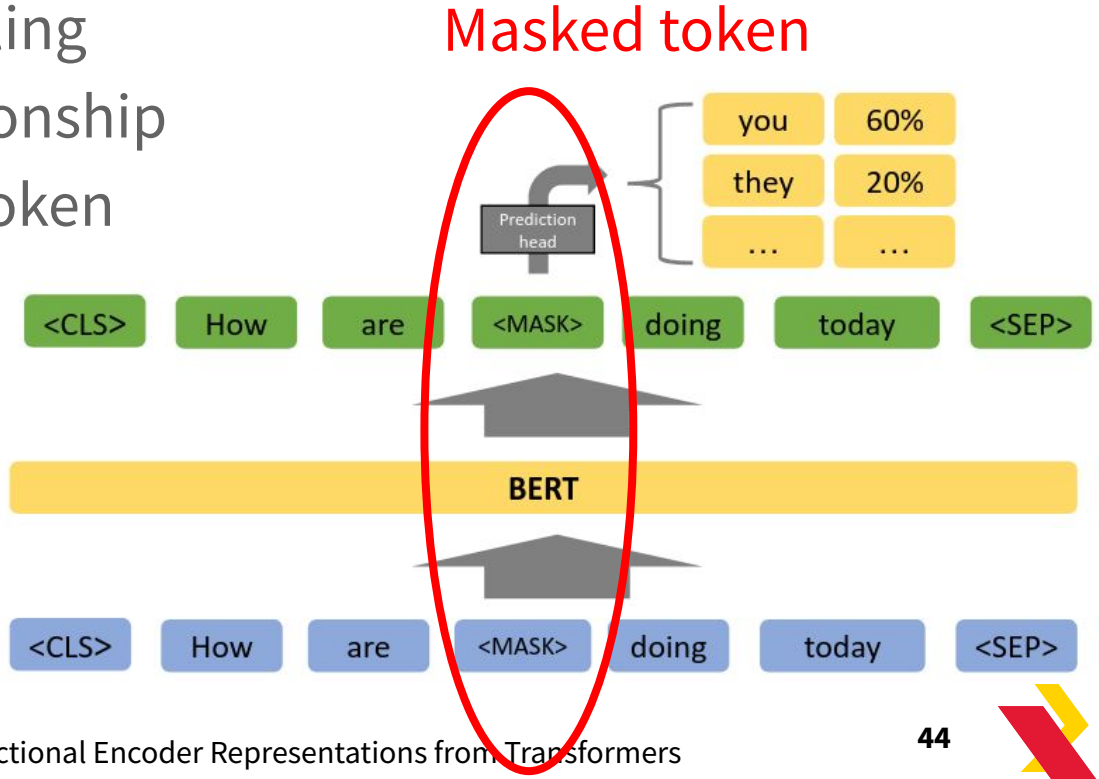
| Label | Sentence |
|---|---|
| 1 | The weights made the rope stretch over the pulley. |
| 1 | The mechanical doll wriggled itself loose. |
| 1 | If you had eaten more, you would want less. |
| 0 | The more you would want, the less you would eat. |



# Model Training (1/2)

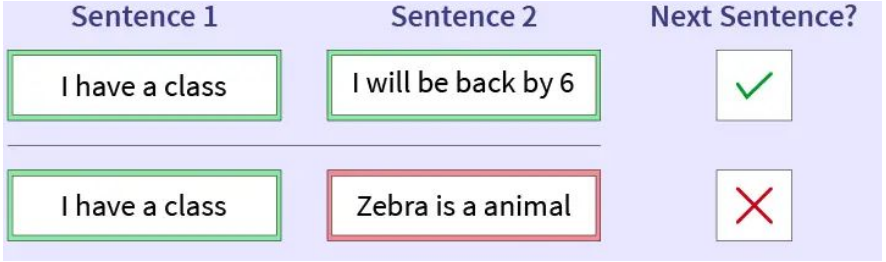
Masked Language Modeling

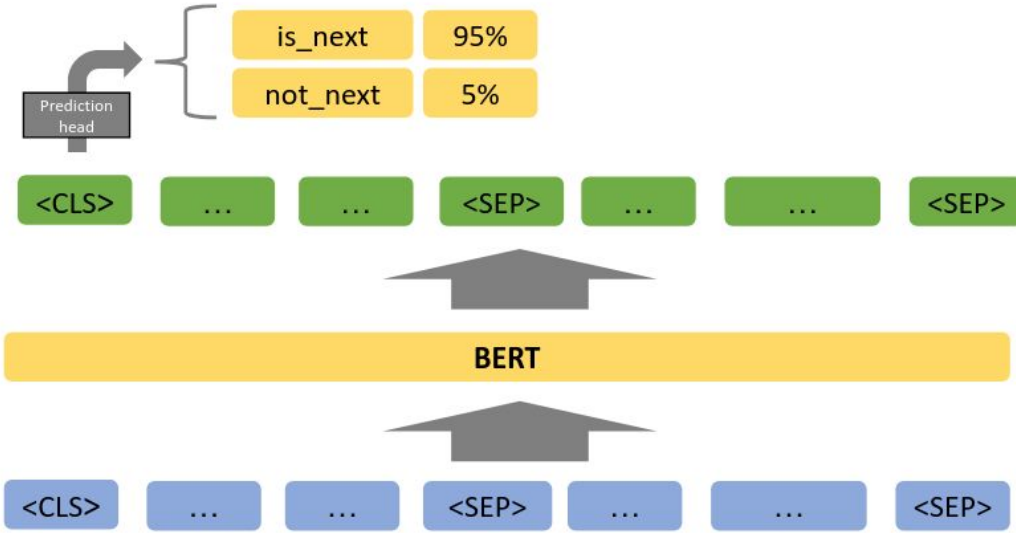
- Train for token relationship
- Use special [MASK] token
- Prediction head
  - Append to [MASK]



# Model Training (2/2)

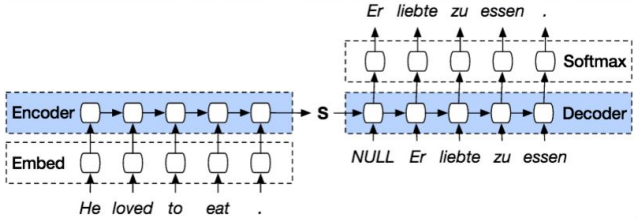
- Next Sentence Prediction
- Input two sentences
  - Separate with a [SEP] token
- Binary label
  - Is_next: Valid next sentence

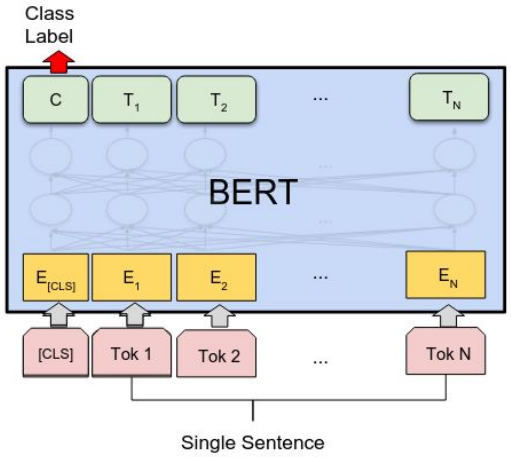


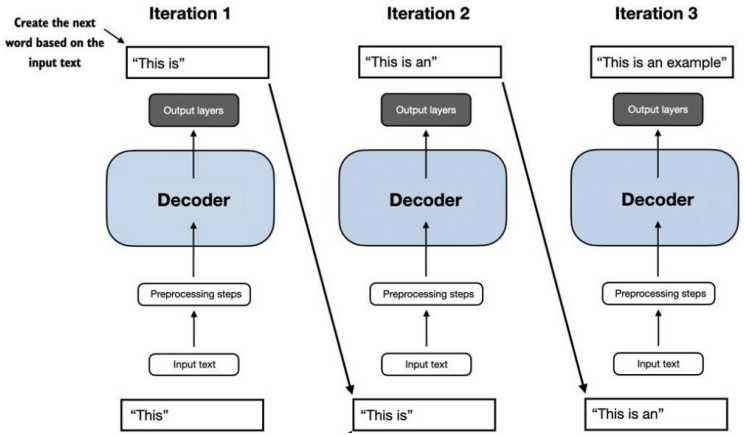



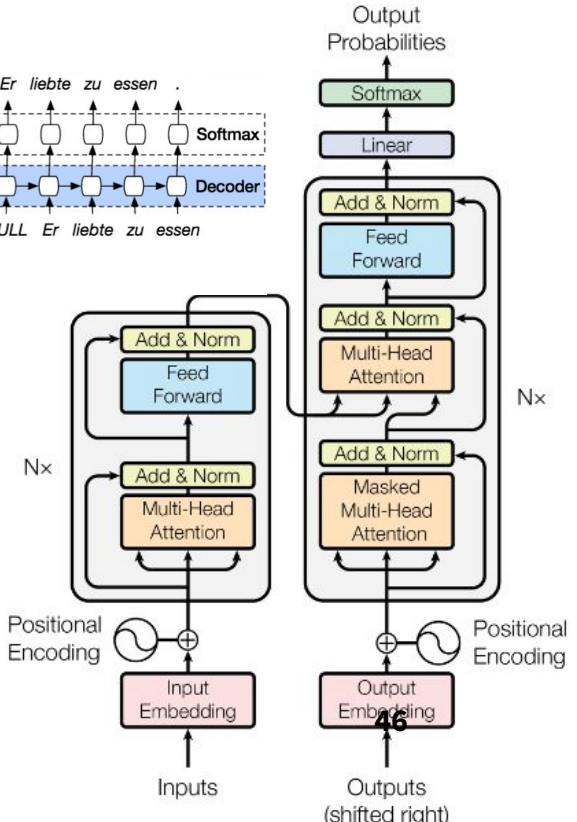
# Quarter 2 Done. Any Question?

1. What is attention? What is transformer?
2. Transformer usage
3. Transformer in hardware
4. Generalized transformer



# Transformer in Hardware

- Embedding
  - From texts to vectors
  - Positional encoding
- Architecture
  - Dynamic matrix multiplication
  - KV cache

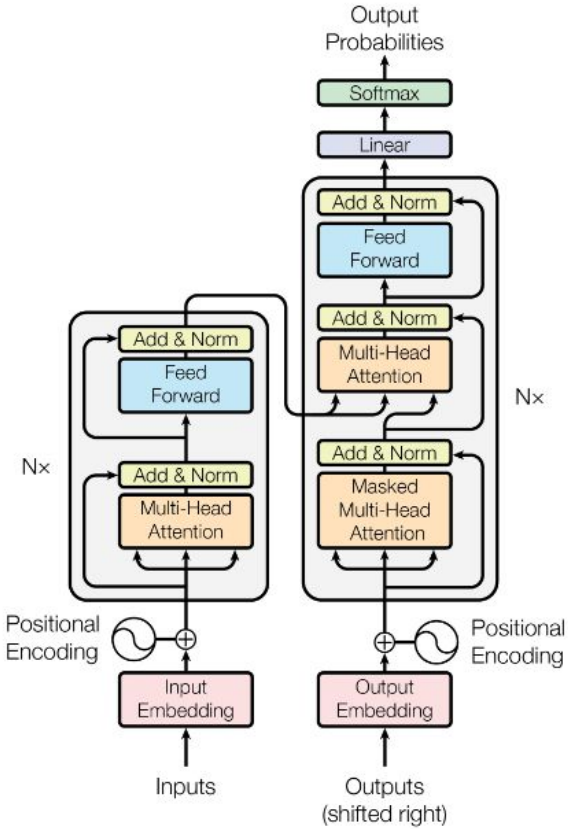


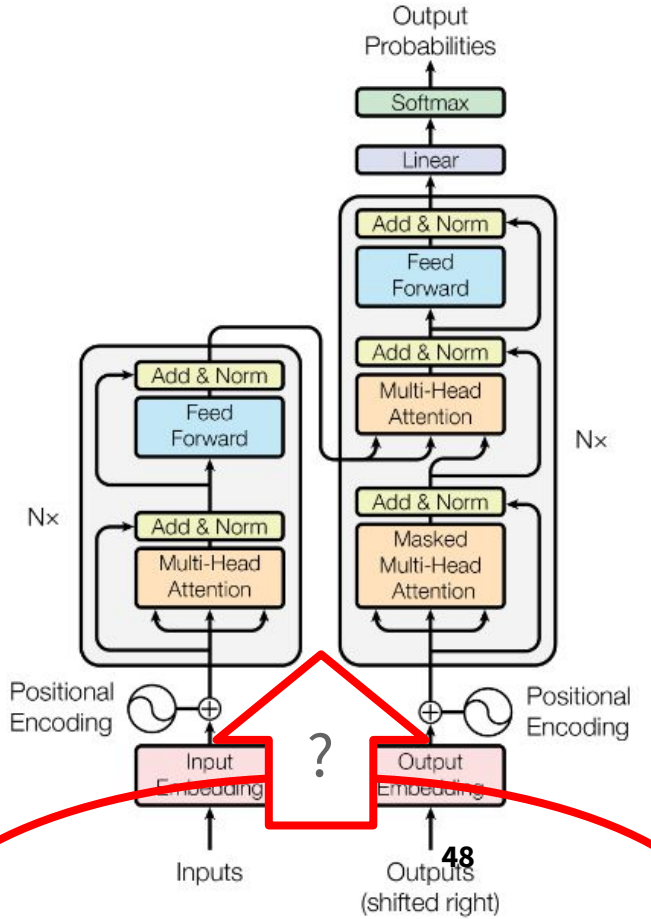

Figure 1: The Transformer - model architecture.



UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

# From Texts to Vectors

- Transformers are matrix-in matrix-out
- How can texts convert to matrix?
- Available tokenization options
  - Character: ['H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd']
  - Word: [‘Tokenizer’, ‘is’, ‘tricky’]
  - Sub-word: [‘Token’, ‘##izer’, ‘is’, ‘tricky’]



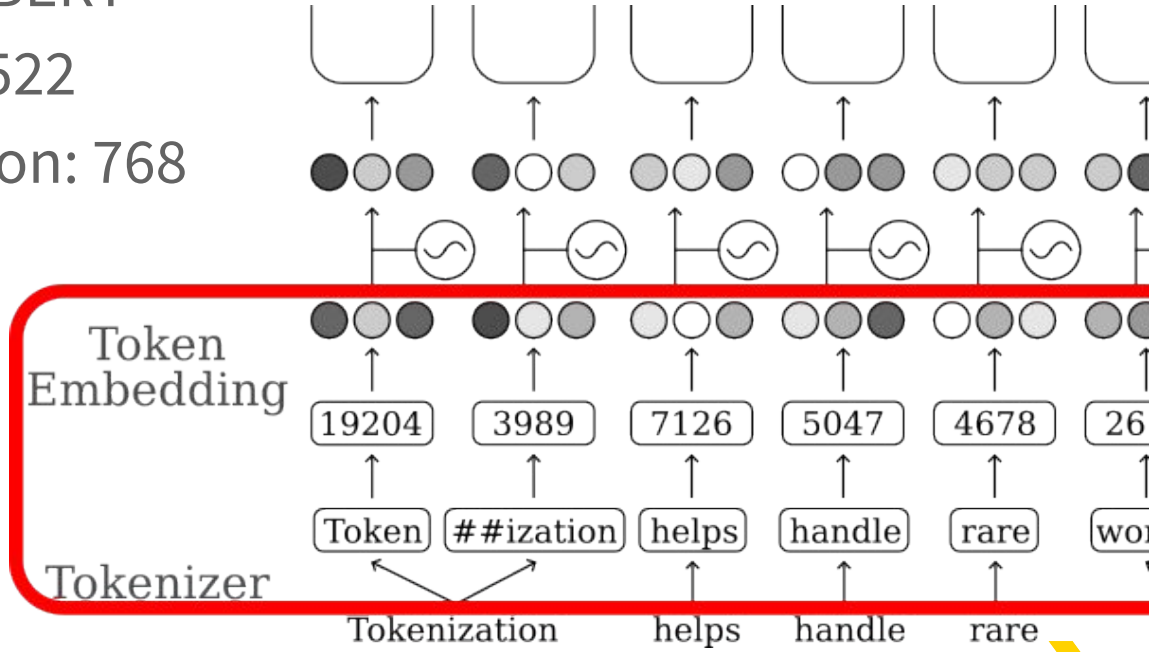
# Token Embedding

WordPiece tokenizer in BERT

- Vocabulary size: 30,522
- Embedding dimension: 768

Embedding Lookup

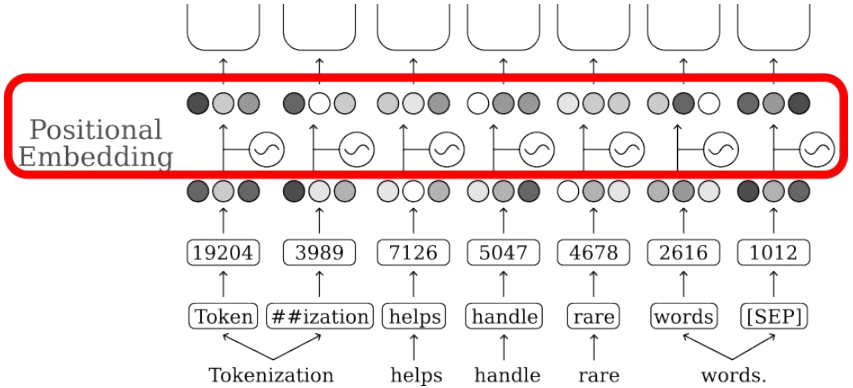
- Token ID -> token

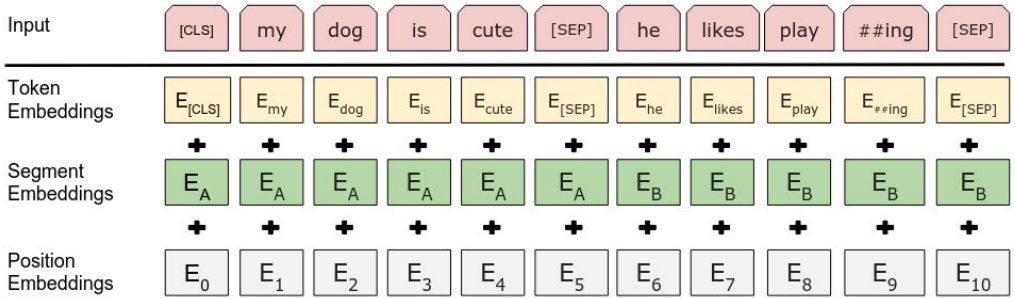


# Positional Encoding – Position Matters

To encode positional information

- Each position maps to a token
  - Called positional embeddings
- Add/Mul to token emb.
- Good positional embedding
  - Non-cyclic
  - Bounded



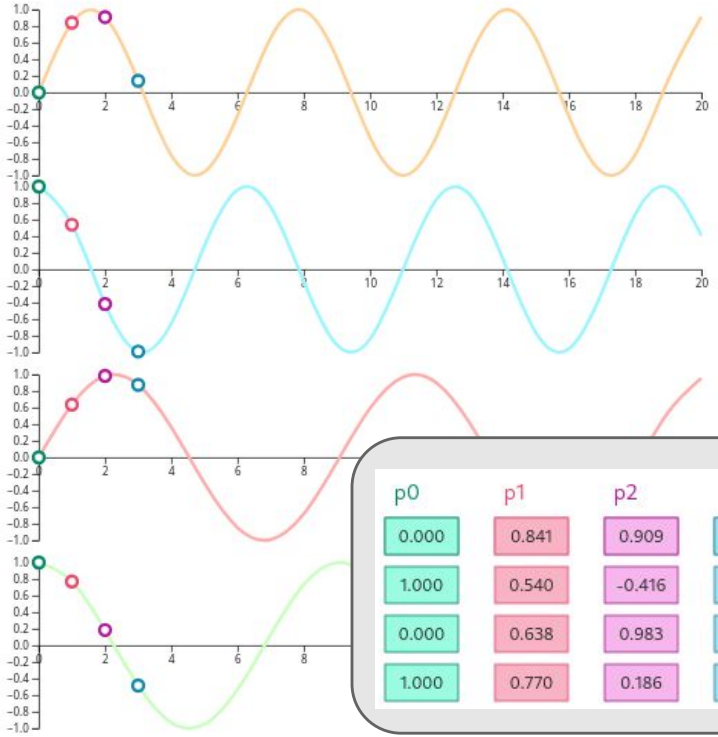

https://erdem.pl/2021/05/understanding-positional-encoding-in-transformers

# Rotary Positional Encoding (RoPE)

- $PE(pos,2i) = \sin(pos \times \theta i)$

  $PE(pos,2i+1) = \cos(pos \times \theta i)$
  - $\theta i = 10000^{-2(i-1)/d}$

- RoPE properties
  - Cyclic but very-large period
  - Linear independent
  - Bounded ($PE \in [-1,1]$)
  - Fix-L2 ($\Sigma PE_i^2$ is const)



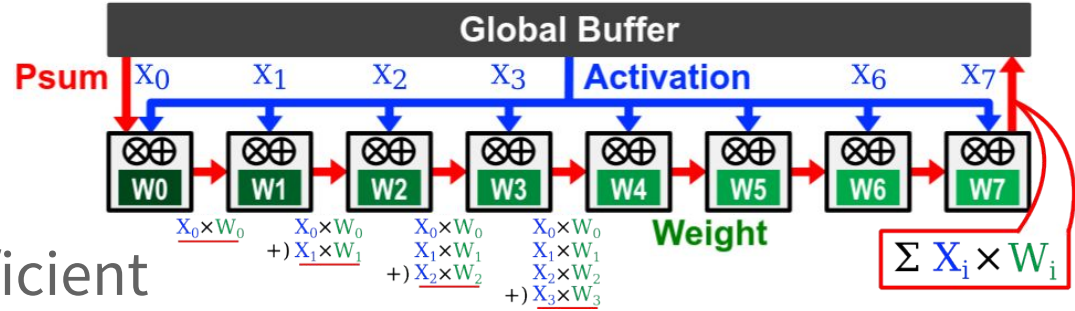
| | p0 | p1 | p2 | p3 | |
|---|---|---|---|---|---|
| | 0.000 | 0.841 | 0.909 | 0.141 | i=0 |
| | 1.000 | 0.540 | -0.416 | -0.990 | i=1 |
| | 0.000 | 0.638 | 0.983 | 0.875 | i=2 |
| | 1.000 | 0.770 | 0.186 | -0.484 | i=3 |

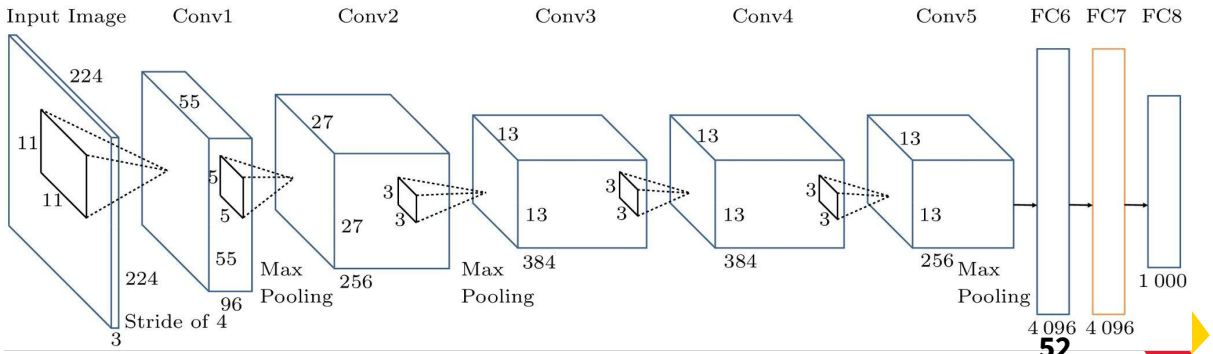


https://erdem.pl/2021/05/understanding-positional-encoding-in-transformers

# Matrix Multiplication: Conv HW Recap

- Conv-based model
  - Small filter size (< 5x5)
  - Large input size
- Weight reuse will be efficient
  - Weight-stationary dataflow
  - Example $\Sigma X_i W_i$



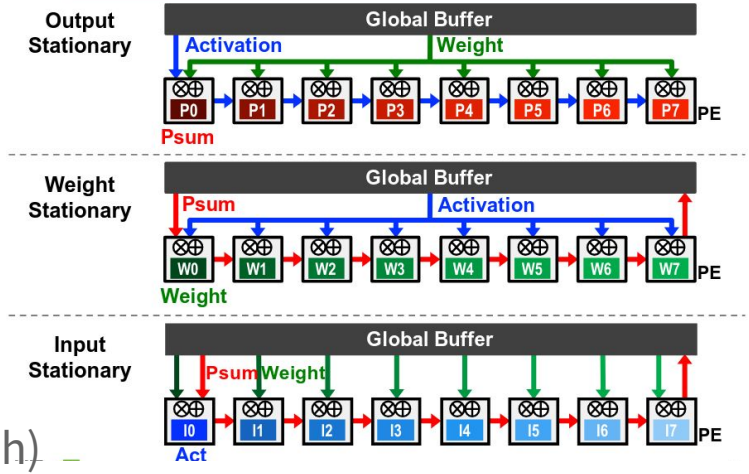



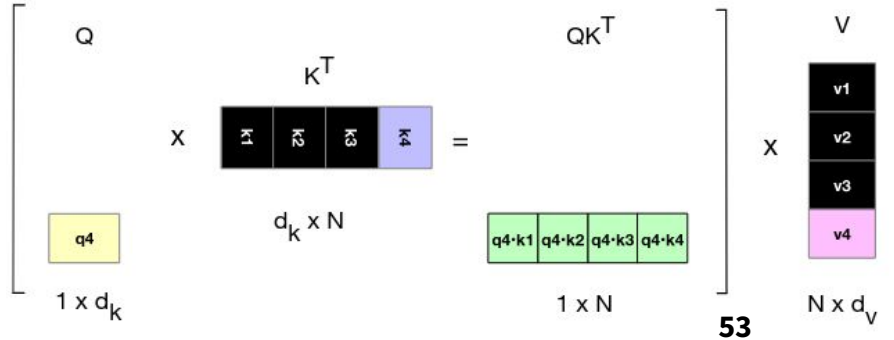



UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

# Matmul in Transformer

- Attention
  - Matmul between Q(1x$d_k$) and K(N x $d_k$)
  - Output has shape (1xN)
  - $d_k^2$xN MACs to generate N outputs ($d_k^2$ each)
- $d_k$ (embedding dim) vs N (attention scope)
  - Large $d_k$ and small N
    - Image-based
    - SwinT-T: $d_k$= 96 and N = 16
  - Small $d_k$ and large N
    - Llama3-70B: $d_k$= 8192, N = 8000



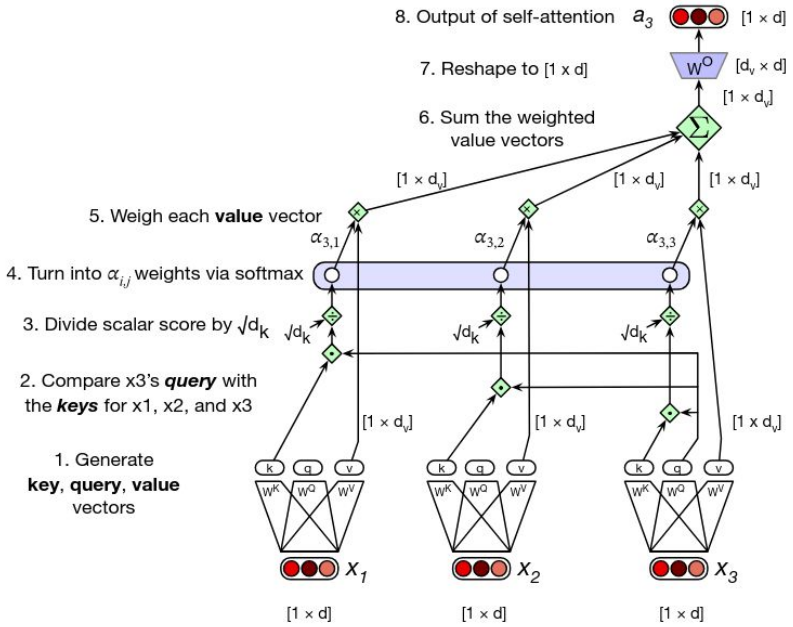



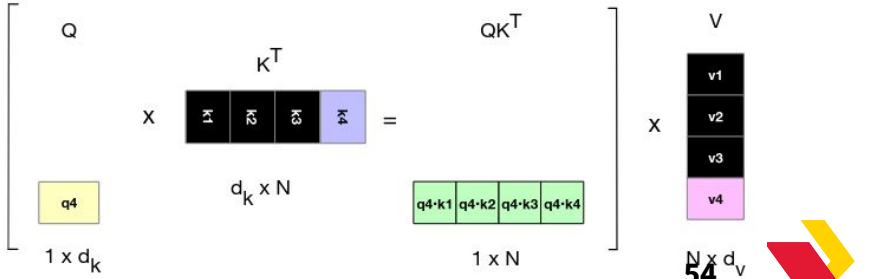



UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

# KV Caching

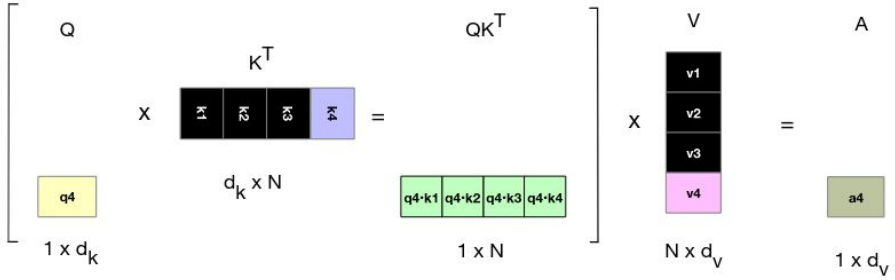
- Step 1 generates Q/K/V from X
- Step 2 use current Q and all Ks
  - One (X→Q) + N× (X→K)
    - (N-1)× (X→K) are computed before
    - Prev N-1 (X→K) were computed
- Step 5 use current $\alpha$ and all Vs
  - N× (X→V)
    - Prev N-1 (X→V) were computed

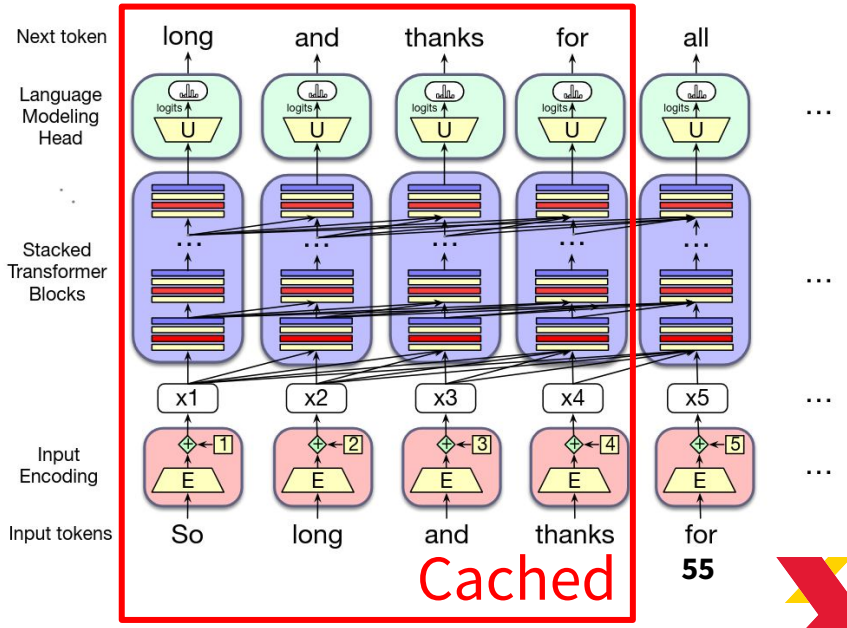


# KV Caching

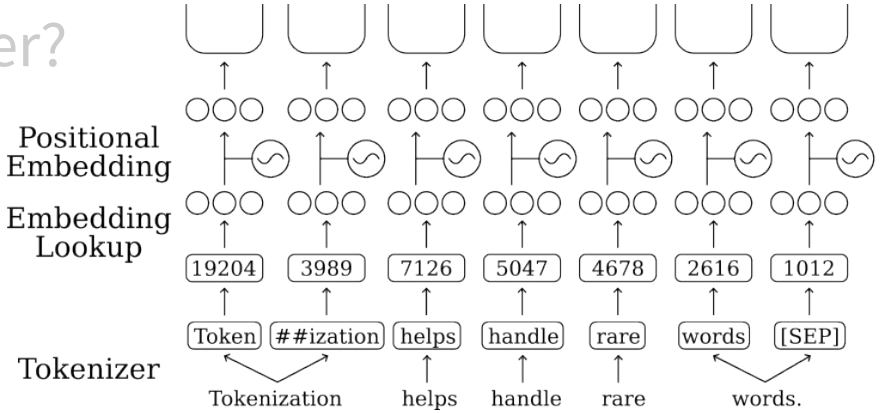
- Cache all K and V from prev N-1 rounds
- Same number of Q/K/V generations every round
- KV cache can be large
  - 2.1GB in LLaMa-7B
  - 19GB in LLaMa-70B

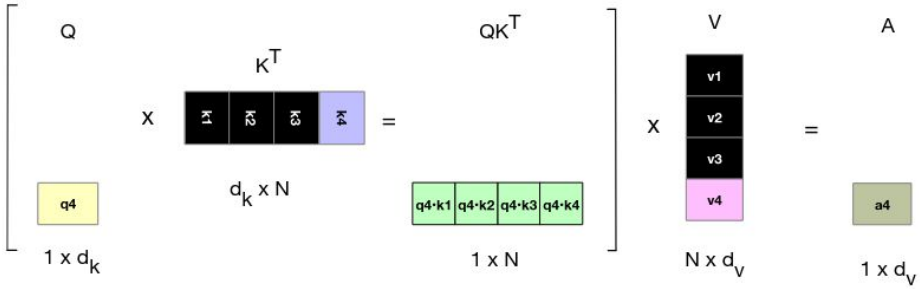


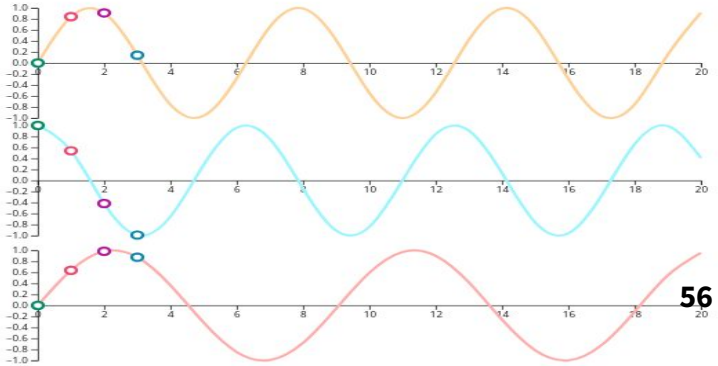



# Quarter 3 Done. Any Question?

1. What is attention/transformer?
2. Transformer usage
3. Transformer in hardware
4. Generalized transformer



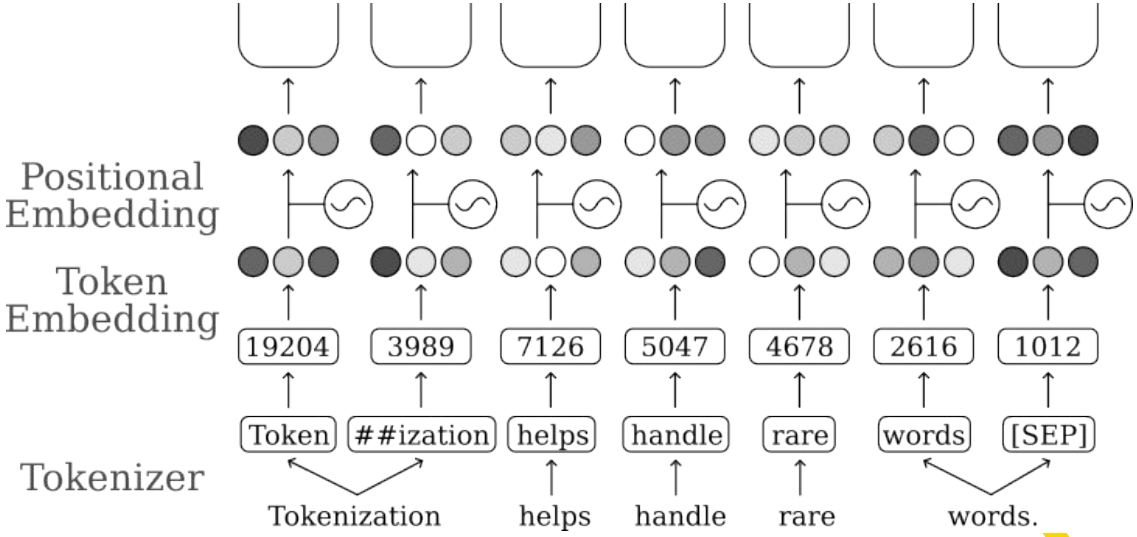



UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

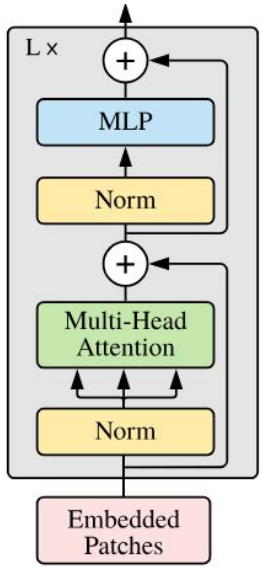
# Recap: Transformer in NLP

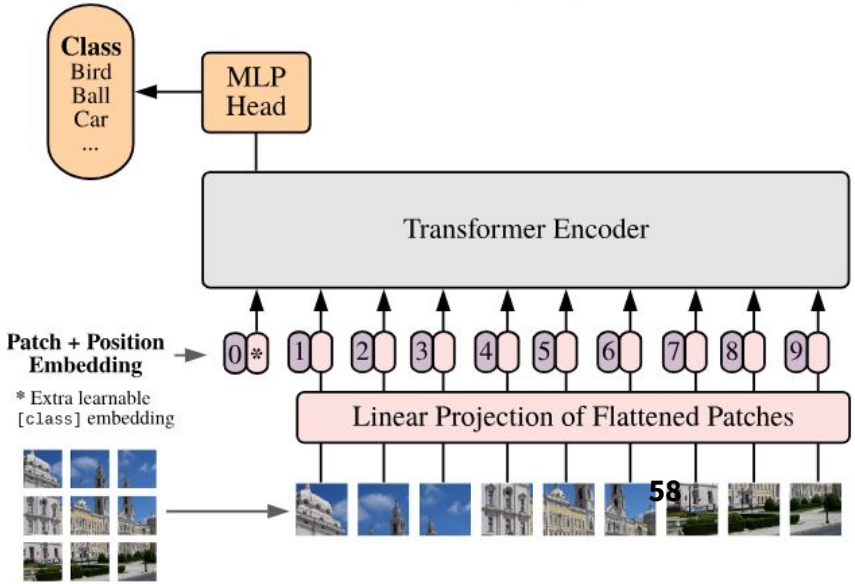
- Token embedding + Positional embedding
- Transformer model
- How to generalize to other applications
  - Vision
  - Audio
  - Multimodal



# Vision Transformer (ViT)

- Token embedding:
  - Image patch -> flattened patch -> linear projection
- Same position embedding
- Model
  - Encoder-only



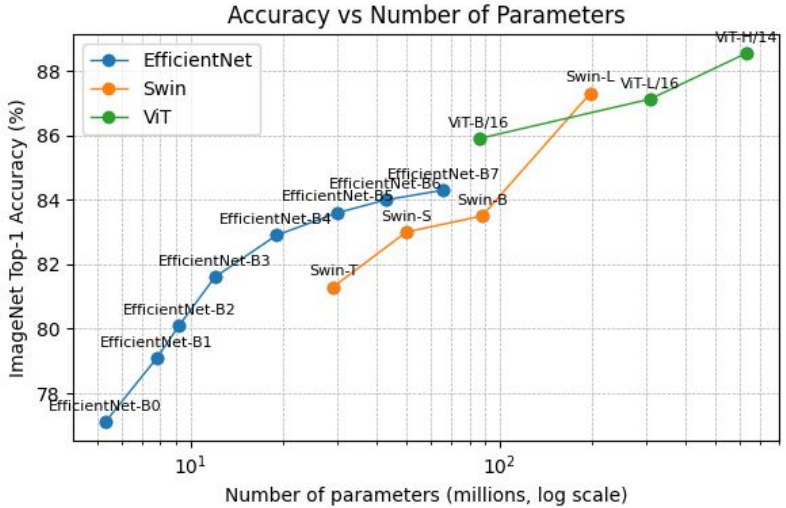






UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

# Results of ViT

- Beat state-of-the-art (SOTA)
- But requires more data and more parameter



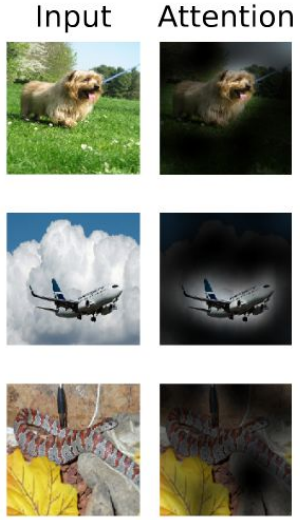



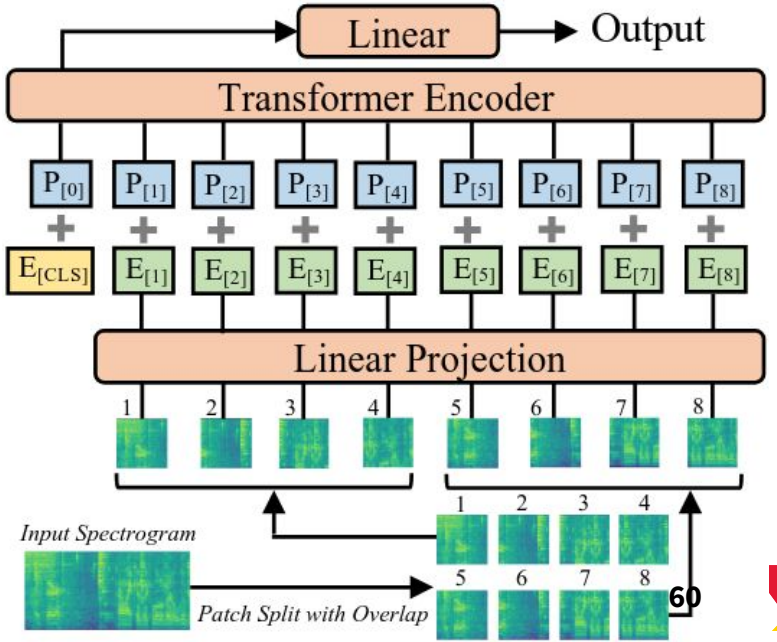

Figure 6: Representative examples of attention from the output token to the input



UNIVERSITY OF MARYLAND
FEARLESSLY FORWARD

# Audio Spectrogram Transformer

- Token embedding:
  - Audio spectrogram -> linear projection
  - Spectrogram (x: time, y:freq)
- Same position embedding
- Model
  - Only transformer encoder
  - With a classification token

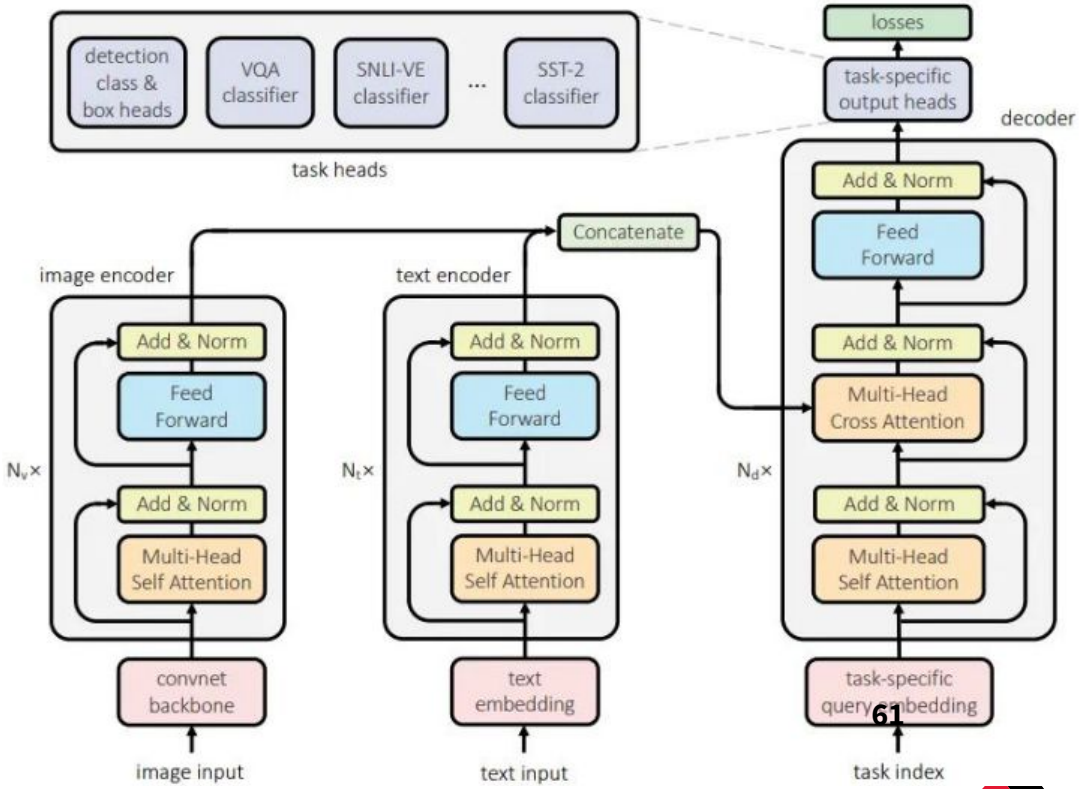





UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

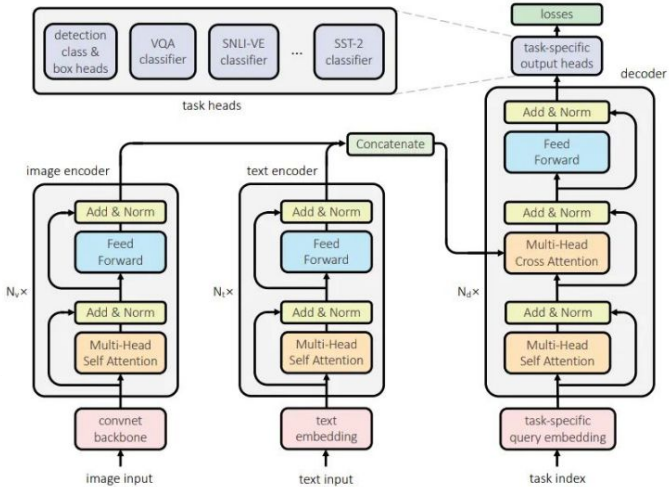
# Multimodal Transformer

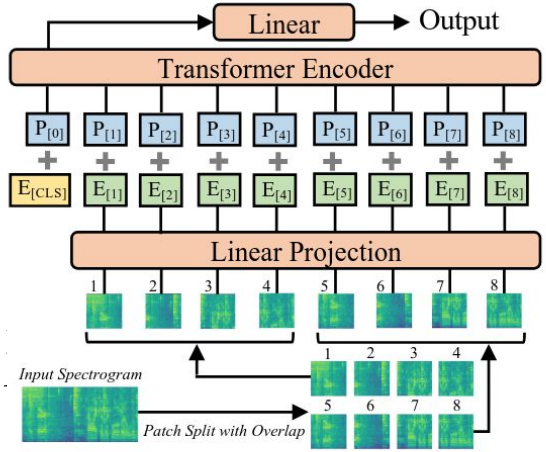
- Token embedding:
  - Image embedding
    + Text embedding
- Model
  - Image encoder
  - Text encoder
  - Joint decoder
    - Concatenated context

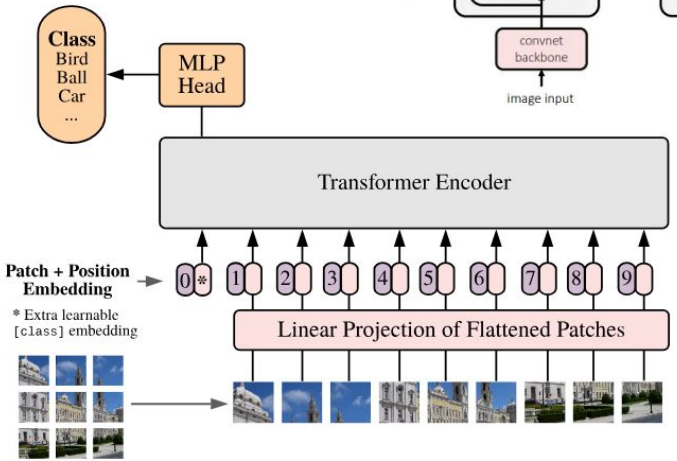


# Quarter 4 Done. Any Question?

1. What is attention/transformer?
2. Transformer usage
3. Transformer in hardware
4. Generalized transformer



# Questions?



UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

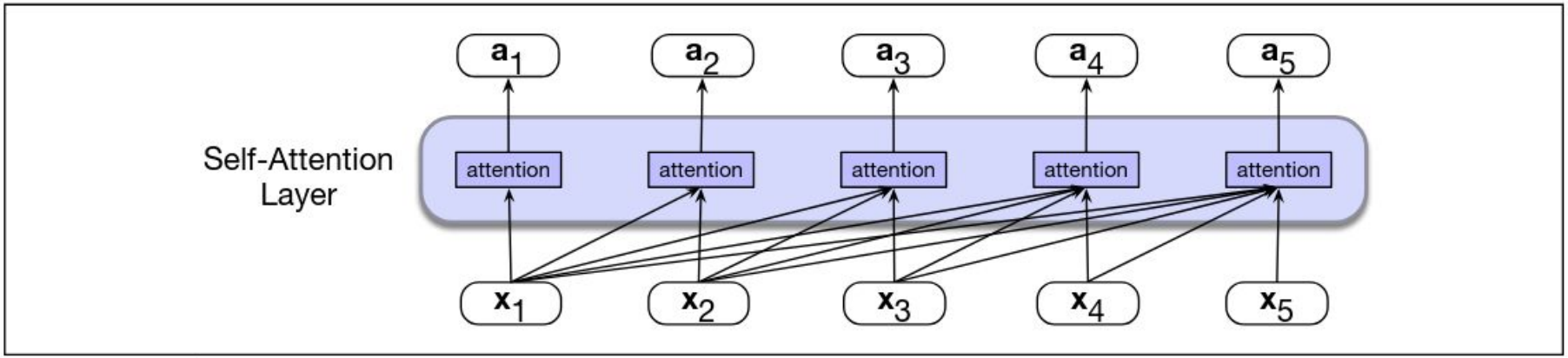



**Figure 8.3** Information flow in causal self-attention. When processing each input $\mathbf{x}_i$, the model attends to all the inputs up to, and including $\mathbf{x}_i$.

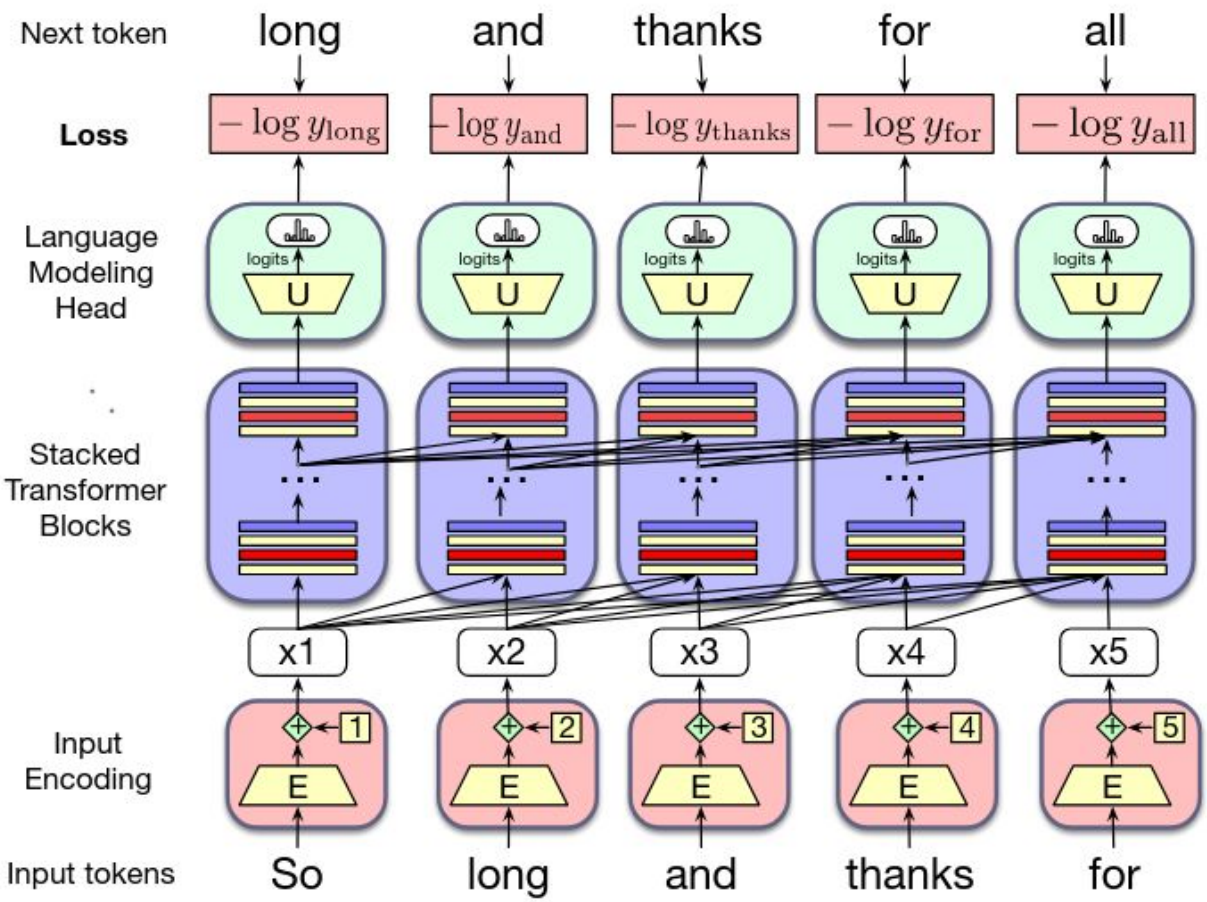



**Figure 8.16** Training a transformer as a language model.

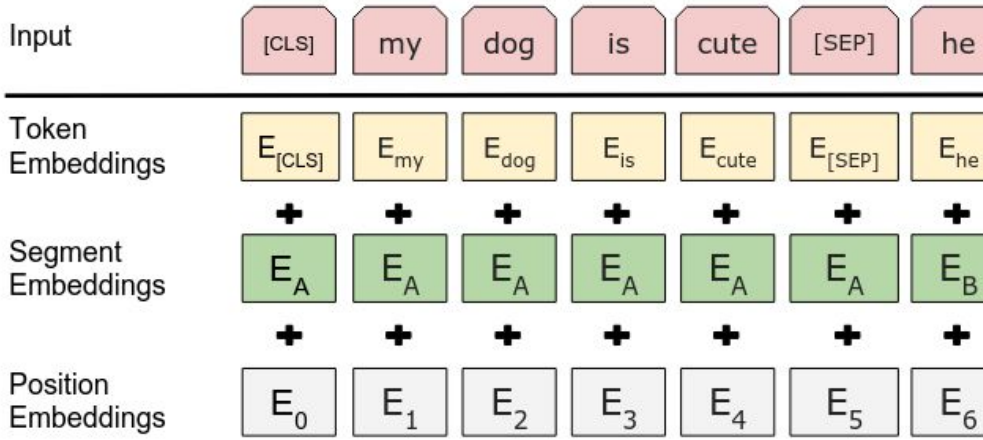



Figure 2: BERT input representation. The input embeddings are the s
tion embeddings and the position embeddings.

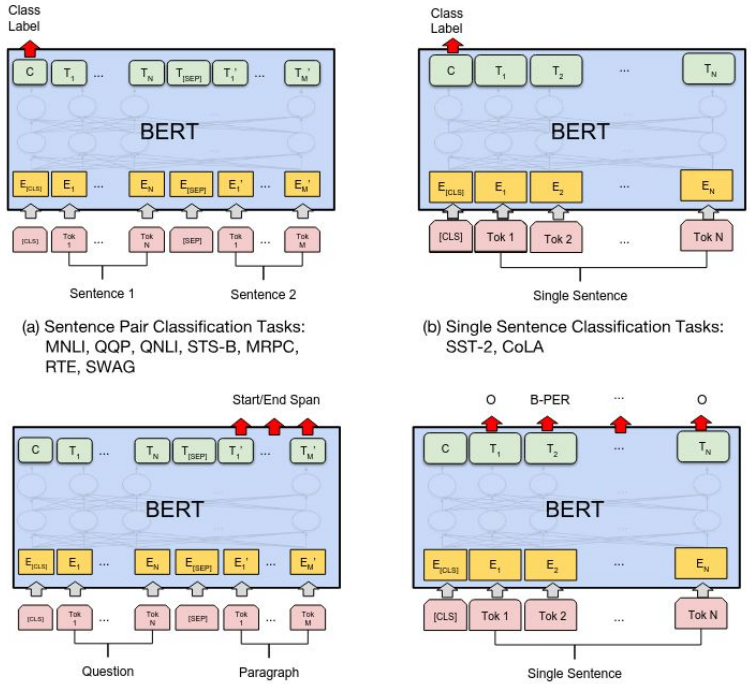



Figure 4: Illustrations of Fine-tuning BERT on Different Tasks.

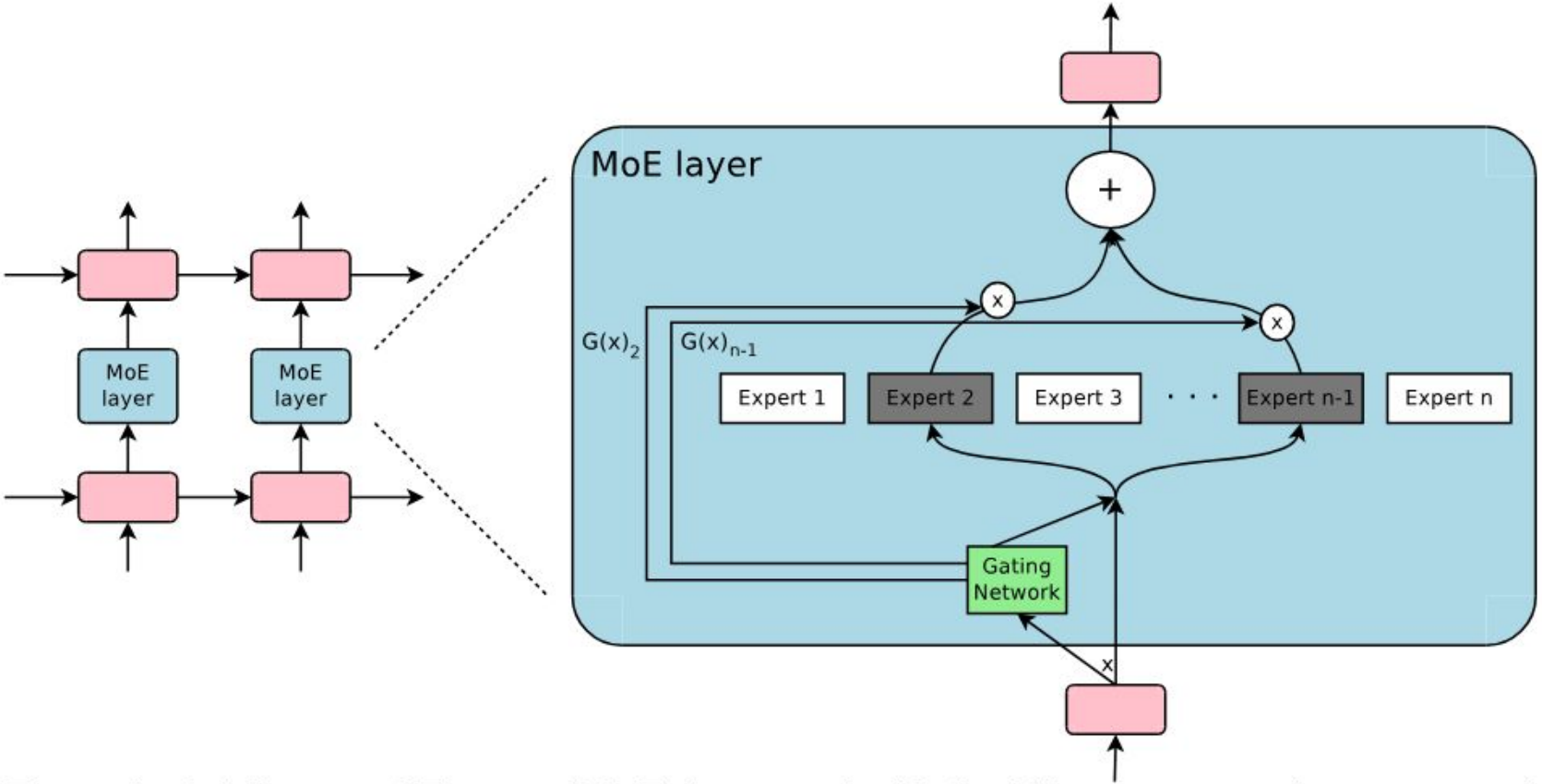
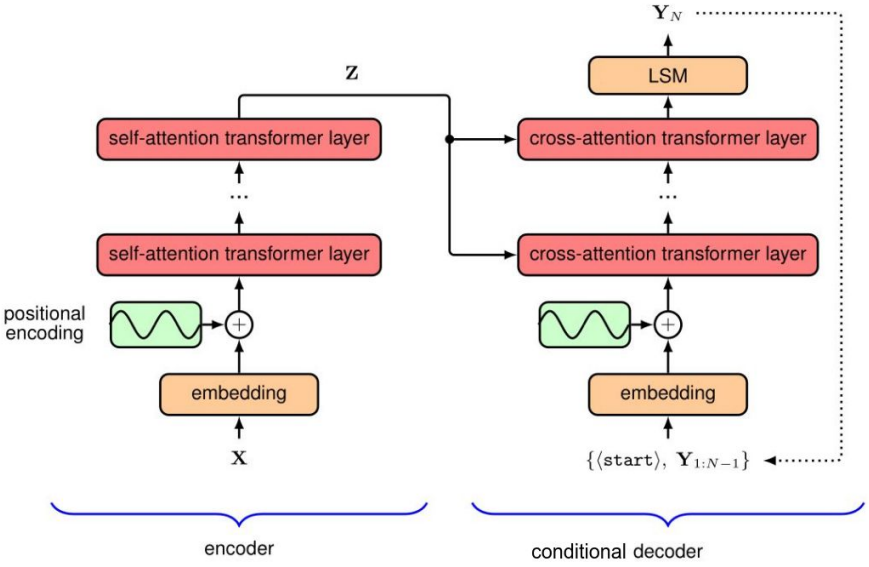
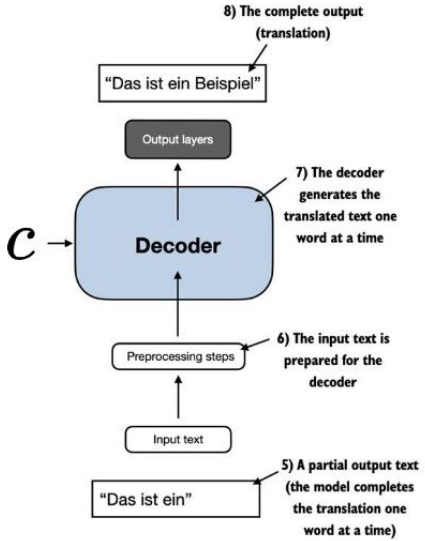



Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.




Approximate

$$p(x_n | x_1, \ldots, x_{n-1}, c)$$

# Multi-token Attention Mechanism

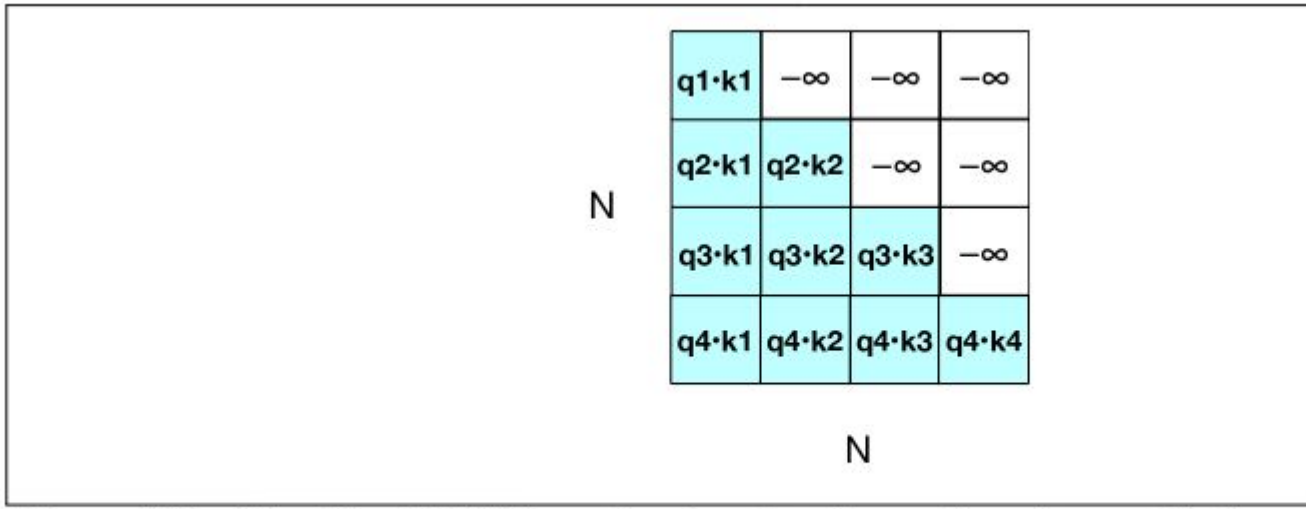



**Figure 8.9** The $N \times N$ $\mathbf{QK}^\intercal$ matrix showing the $q_i \cdot k_j$ values, with the up
tion of the comparisons matrix zeroed out (set to $-\infty$, which the softmax wi