Midterm Coverage

General Information

- In-class exam on Mar 27 5:10pm-6:30pm
- 80 mins
- Close book, you can bring calculator
- No cheat sheet is allowed
- Five questions

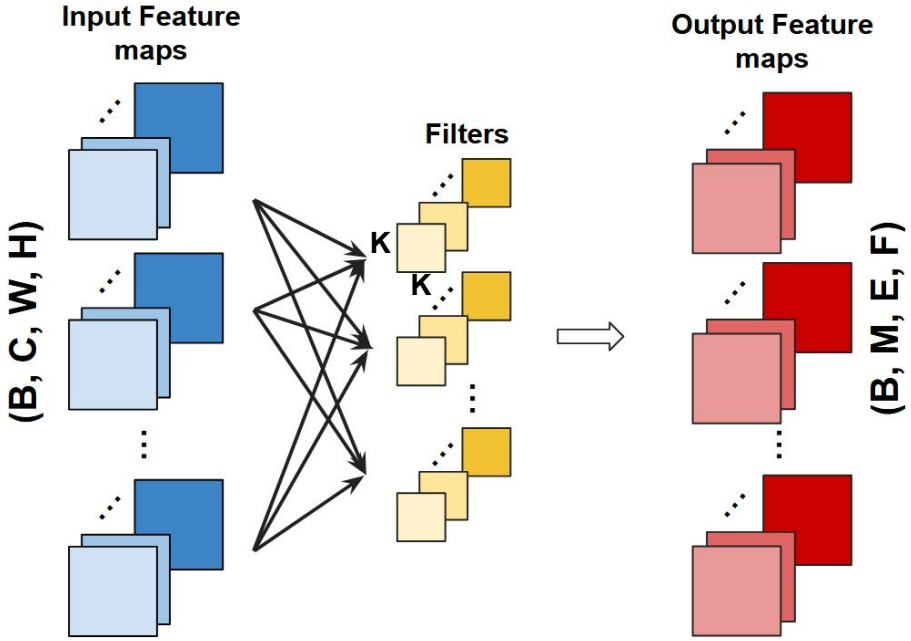
Recap (Lecture 1: P65-P98)

- Neural Network Basics
  - MLP
  - Forward and backward propagation of MLP
  - Weight decay, dropout
  - The training optimizer: SGD, RMSProp, Adam
  - Multistage learning rate scheduler
  - Initialization

Recap (Lecture 2: P1-P69, P75-P78)

- Conv2D operation
  - How the computation is performed
  - Input dimension, weight dimension, output dimension
  - Computational cost
- BatchNorm
  - Parameter folding-in during inference
- ResNet, MobileNet, ShuffletNet, SqueezeNet, DenseNet
  - Depthwise Separable Conv
  - Groupwise Convolution
- Other tasks: segmentation

Computational Cost: Standard Convolution

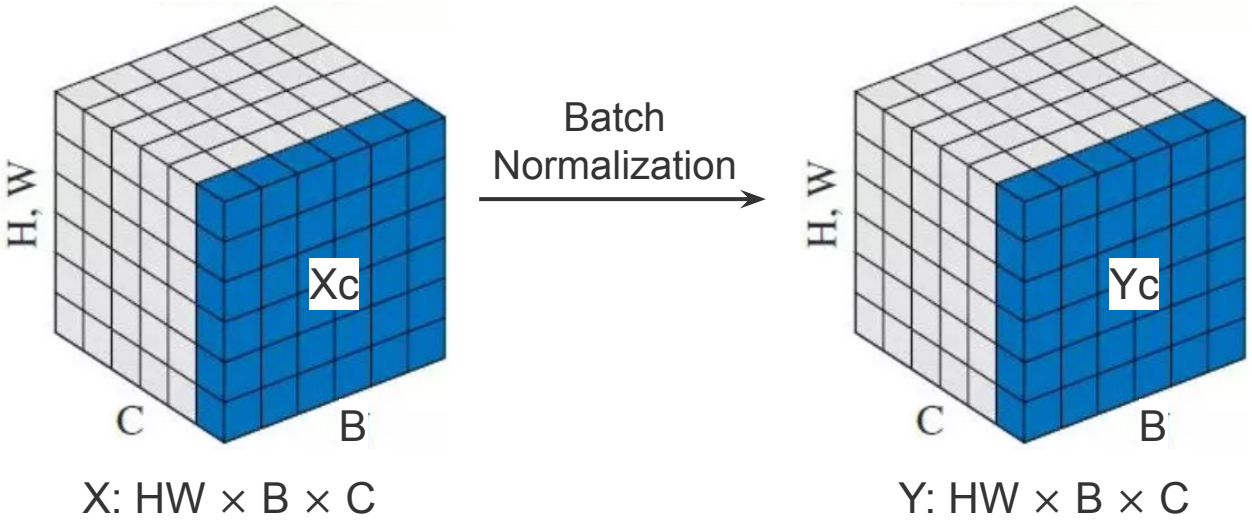


- Number of MACs: $B \times M \times K \times K \times C \times E \times F$
- Storage cost: $32 \times (M \times C \times K \times K + B \times C \times H \times W + B \times M \times E \times F)$

B: batch size
C: number of input channels
H,W: size of the input feature maps
M: number of weight filters
K: weight kernel size
E,F: size of the output feature maps

NYU SAI LAB

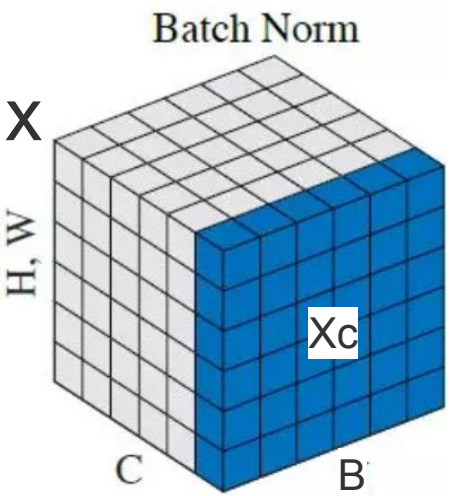
Batch Normalization



- **Batch Normalization (BatchNorm)** is a technique used in deep learning to improve the training stability and performance of neural networks.

Ioffe, Sergey. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

Batch Normalization



X: HW × B × C

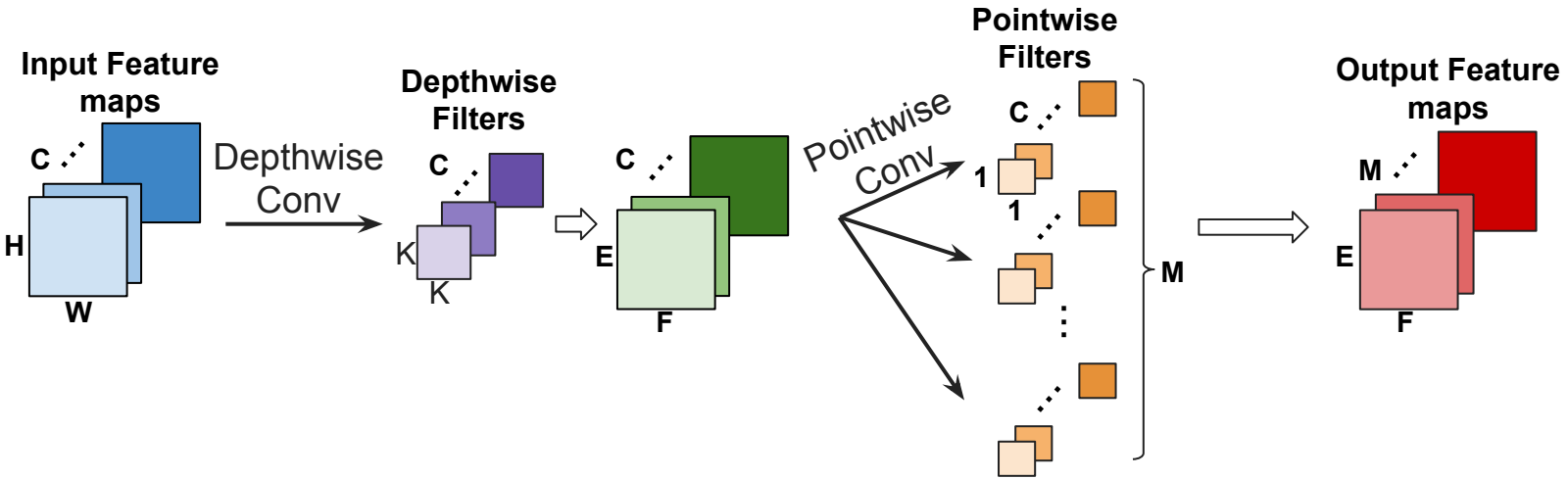
$$Y_c = \alpha_c \frac{X_c - \mu_c}{\sigma_c} + \beta_c \qquad \text{For each } c \in C$$

$$\alpha = \{\alpha_c\}, \beta = \{\beta_c\}, \mu = \{\mu_c\}, \sigma = \{\sigma_c\}$$

- For each channel c, we have:
  - Xc: (HW x B)
  - μ_c and δ_c are the mean and standard deviation of Xc.
  - α_c and β_c are learnable parameters
  - α_c, β_c, μ_c, δ_c are scalers
- Overall, we have:
  - μ, δ, α and β all have a length of of C
  - μ, δ, α and β are all fixed during the inference
  - μ, δ are statistics based on the training dataset

NYU SAI LAB

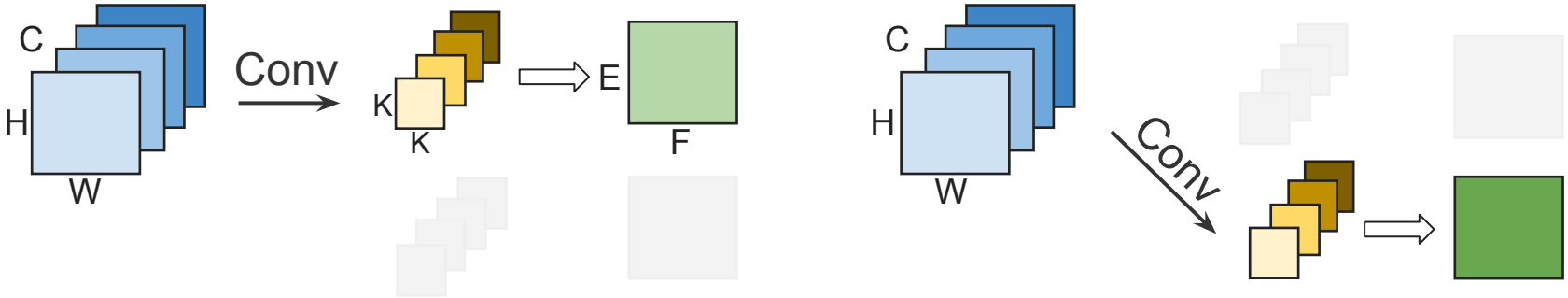
Depthwise Separable Convolution



- Number of MACs: K×K×C×E×F + M×C×E×F
- Storage cost: 32×(C×H×W+C×K×K+C×E×F+M×C+M×E×F)

NYU SAI LAB

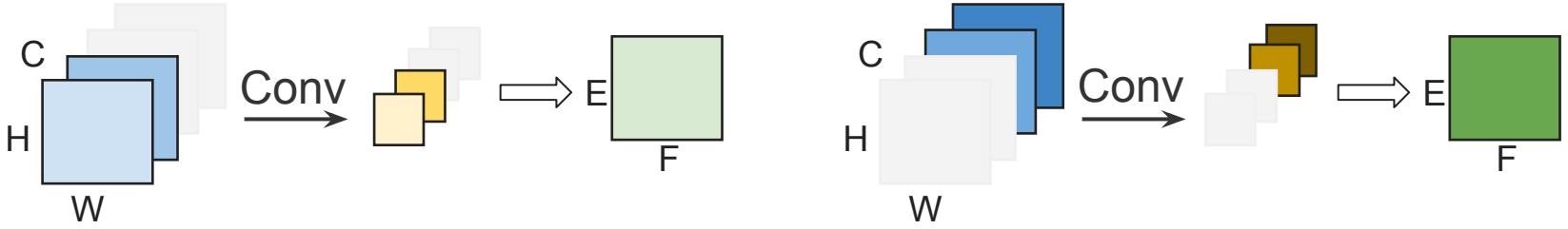
Group Convolution



- The original MAC: E×F×K×K×C×M

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." *Communications of the ACM* 60.6 (2017): 84-90.

Group Convolution



- Group size = 2
- Each group of feature maps within the input only convolved with partial weight kernels.
- This will lead to a large saving on memory consumption and computational cost.
- The number of MAC: E×F×K×K×C×M/G

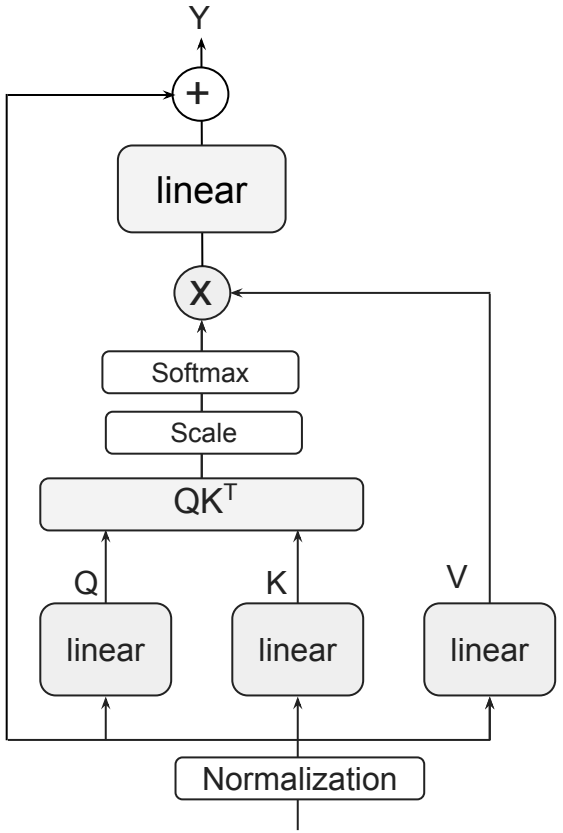
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." *Communications of the ACM* 60.6 (2017): 84-90.

Recap (Lecture 3: P1-P69, P75-P78)

- Transformers
  - How the computation is performed and why
  - Multi-headed attention, FFN
  - LayerNorm, RMSNorm, GeLU
  - Positional embedding, Word embedding
- Vision Transformer
  - How to convert image into visual tokens
- LLM
  - Prefilling, decoding
  - KV cache
- SSL
  - Basics

Self-Attention Block

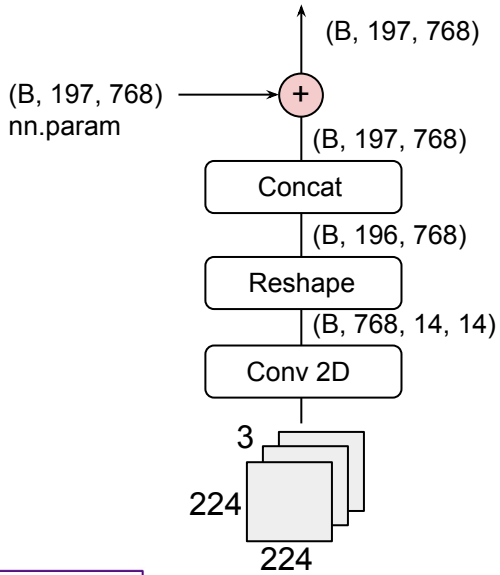
- Given input x, the first step in calculating self-attention is to create three vectors from each of the input x', denoted as: Query (Q), Key (K), Value (V).
  - $(B,L,E) \times (E \times E) \rightarrow (B \times L \times E)$
- The second step in calculating self-attention. This will compute the attention score between each pair of input tokens.
  - $QK^T \rightarrow (B, L \times E) \times (B,E \times L) \rightarrow (B, L \times L)$
- Scale and normalize the score using softmax.
  - $\text{Softmax}(QK^T) \rightarrow (B, L \times L)$
- Multiply each value vector by the softmax score.
  - $\text{Softmax}(QK^T) \times V$
  - $(B, L \times L) \times (B, L \times E) \rightarrow (B, L \times E)$
- Pass the result to the linear layer, sum with the input.



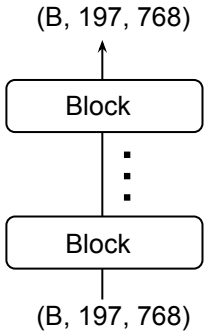
NYU SAI LAB

Vision Transformer

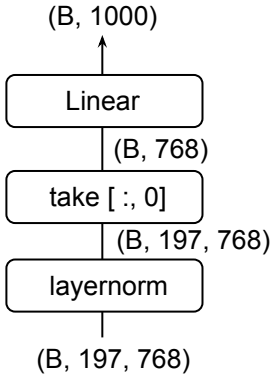
- Transformer architecture can also be applied over the computer vision tasks.



(part 1)



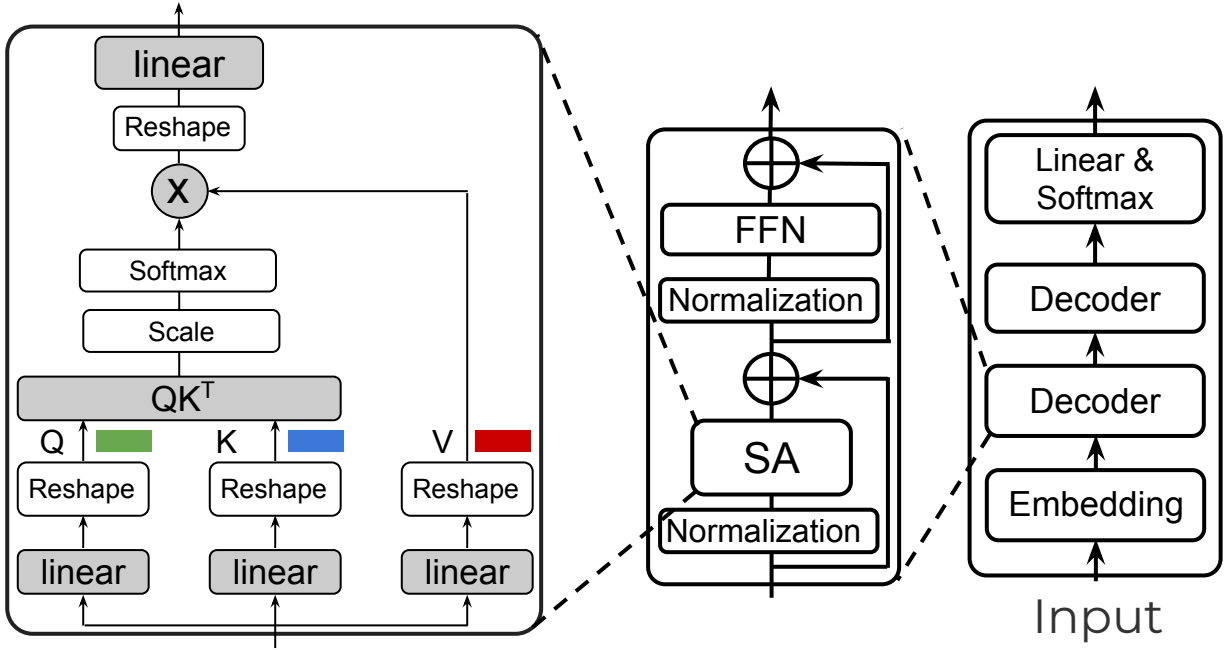
(part 2)



(part 3)

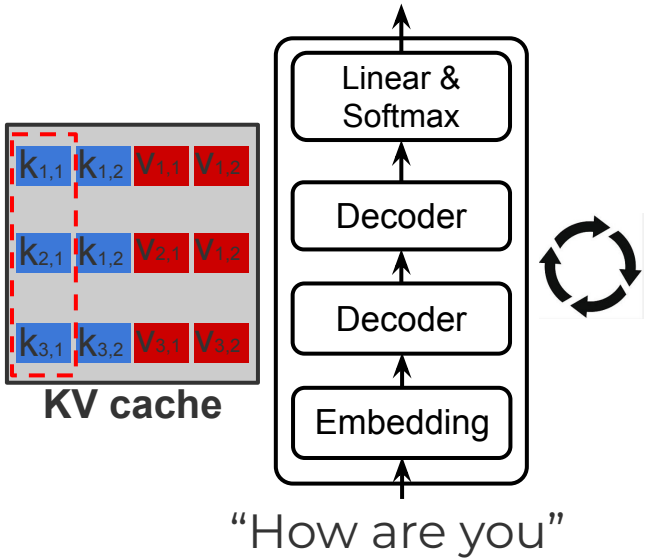
NYU SAI LAB

Transformers as a Generative AI Tool



- We need to buffer the v and k for later usage.

GPT-2: Prefilling



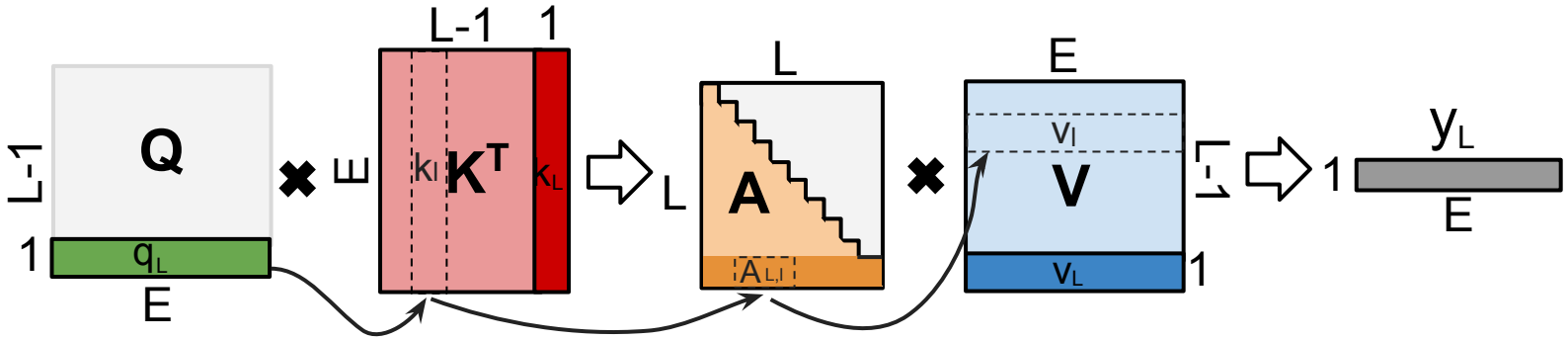
$k_{i,j}$ Key vector for i^{th} token in j^{th} layer
$(1 \times E)$

$v_{i,j}$ Value vector for i^{th} token in j^{th} layer
$(1 \times E$
$)$

- During the prefilling stage, LLM processes the entire prompt, or context tokens jointly, saving the KV vectors into the memory.

NYU SAI LAB

Why KV Cache Saves Computation?



- During the decoding phase, new tokens are continuously generated and must be processed using the buffered K and V vectors to generate subsequent tokens.
- Without a KV cache, all previous K and V vectors must be recomputed, resulting in significant computational overhead.

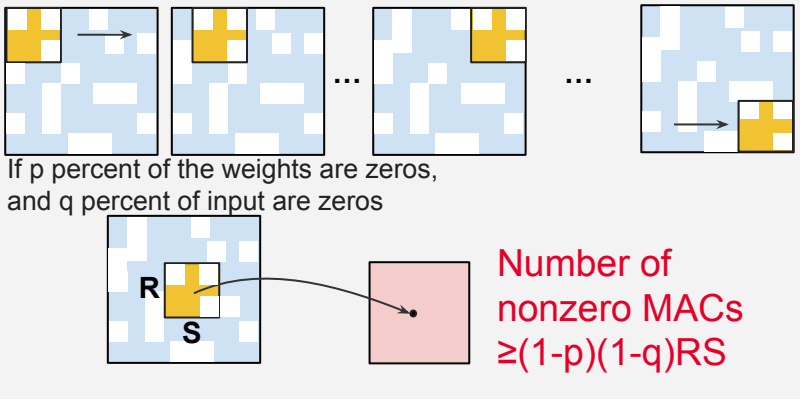
NYU SAI LAB

Recap (Lecture 4: P1-P65, P68-P75)

- Computational cost saving with pruning
  - CNN & Transformer
- Sparse matrix encoding
  - Bitmap, Run-length encoding, COO
- General pruning techniques
  - Magnitude pruning, gradient-based, Hessian-based pruning
  - Lasso
  - Taxonomy of Pruning
  - Network Slimming, N:M sparsity
  - Cascade effect of pruning
- Transformer pruning
  - Token pruning
  - Head pruning

NYU SAI LAB

Convolution with Sparse Weight



- Number of MACs≥(1-p)×(1-q)×M×C×R×S×E×F
- Input pruning can also reduce the computations.
- Sparse input and weight matrices can be stored more efficiently, which helps minimize memory storage.

Sparse Matrices Encodings

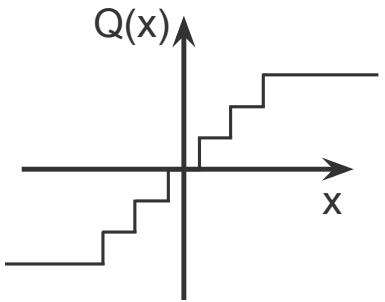
- Efficient encoding scheme for sparse matrix storage.
  - Bitmap
  - Run Length Encoding (RLE)
  - Coordinate format (COO)

Recap (Lecture 5: P1-P70)

- Basic Data Formats
  - Fixed point (INT), Floating point (FP), Block floating point (BFP)
- Quantization
  - Unsymmetrical & Symmetrical
  - Why fixed, FP, BFP & logarithm quantization can save computation?
- STE
- Taxonomy of Quantization
- Qunatization during training
  - Stochastic quantization
- Learnable adaptive quantization scheme

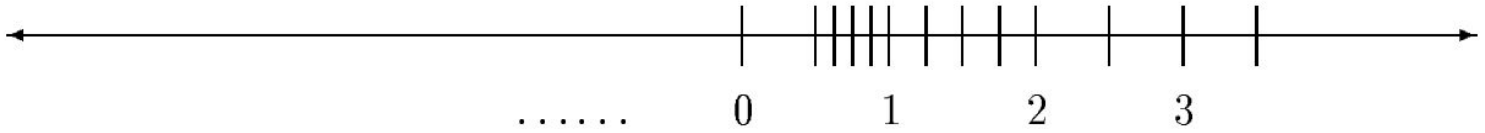
Fixed-Point Format (Symmetrical)

- How to convert a number x to INT representation?
  - Set the clipping range: (-L, L), bitwidth: b
  - Compute the scale: $s = 2L/(2^b - 2)$
  - Clip the input x: $x_c = Clip(x, L, -L)$
  - Calculate the INT representation: $x_{int} = round(x_c/s)$
  - Rescale: $x_q = sx_{int}$



- Have a uniform representation power within the clipping range.
- All the computations can be performed using x_{int}

Floating Point Arithmetic



- Have better representation power for values with small magnitudes.
- How to convert a real number x to FP representation?

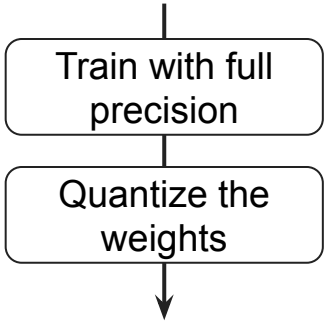
x = |x| s = sign(x)

$$a = \lfloor log_2 x \rfloor \quad e = a + bias \quad m = \frac{x}{2^a} - 1$$

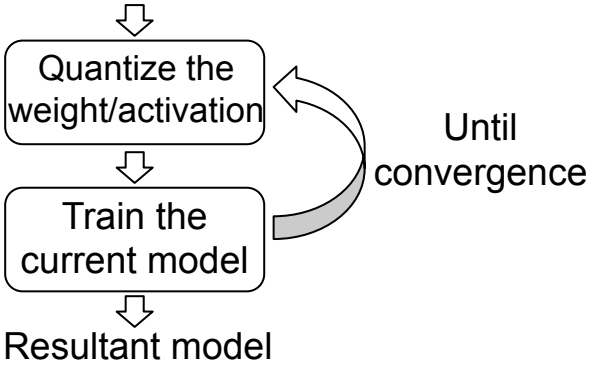
NYU SAI LAB

When to Quantize?

Post-training quantization (PTQ)



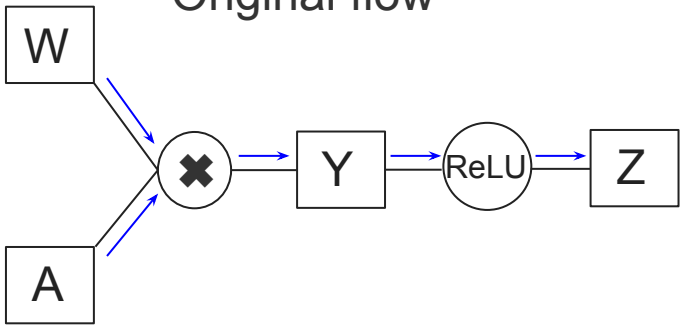
Quantization-aware Training (QAT)



- PTQ has lower computational cost, but accuracy is also lower.
- For the model which is expensive to train (LLM), PTQ is applied to facilitate their implementations.

Another Way to Look at Quantization

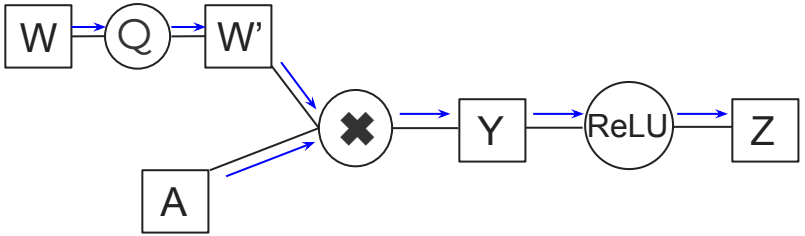
Original flow



Y = WA, Z = ReLU(Y)

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Z}\frac{\partial Z}{\partial Y}\frac{\partial Y}{\partial W}$$

Flow with quantization

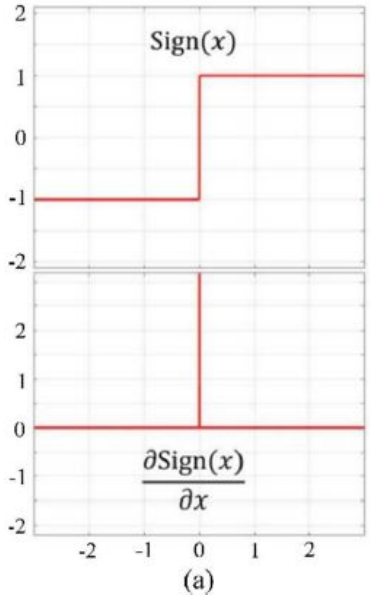


$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Z}\frac{\partial Z}{\partial Y}\frac{\partial Y}{\partial W'}\frac{\partial W'}{\partial W}$$

How to compute $\frac{\partial W'}{\partial W}$?

NYU SAI LAB

Straight Through Estimator (STE)



- Staircase function has a derivative of 0 at most of the values. This will makes the DNN not trainable.
- We instead use STE to estimate the gradient of a non-differentiable quantized function in the backward pass.

$$\frac{\partial W'}{\partial W} = 1$$

- During the forward pass, apply quantization, for backprop, ignore it.

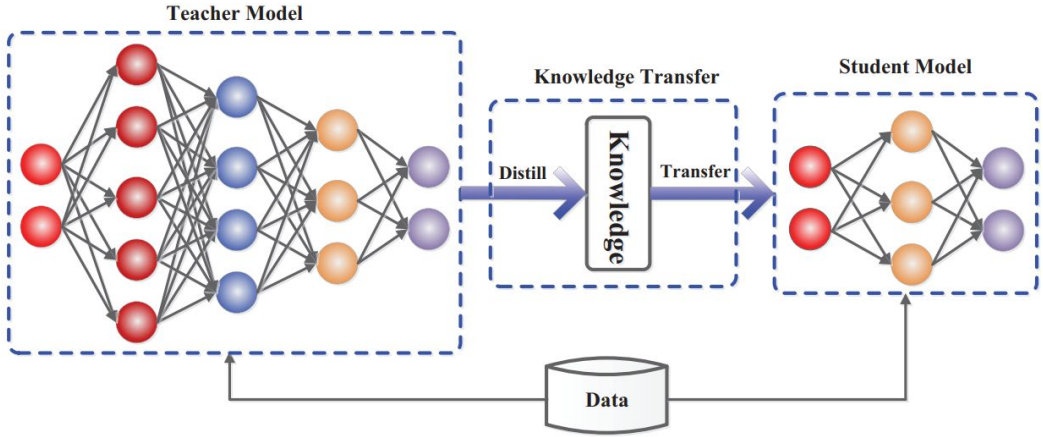
Li, Hao, et al. "Training quantized nets: A deeper understanding." *Advances in Neural Information Processing Systems* 30 (2017).

Recap (Lecture 6: P1-P32, P50-P60, P63-P68)

- Distillation
  - Feature-Based Knowledge Distillation
  - Online distillation
  - Self distillation
  - Multi-teacher, multi-student, cross-modal
- Once-for-all NAS
- Low-rank Decomposition
- Dynamic / Conditional Computing

Knowledge Distillation Basics

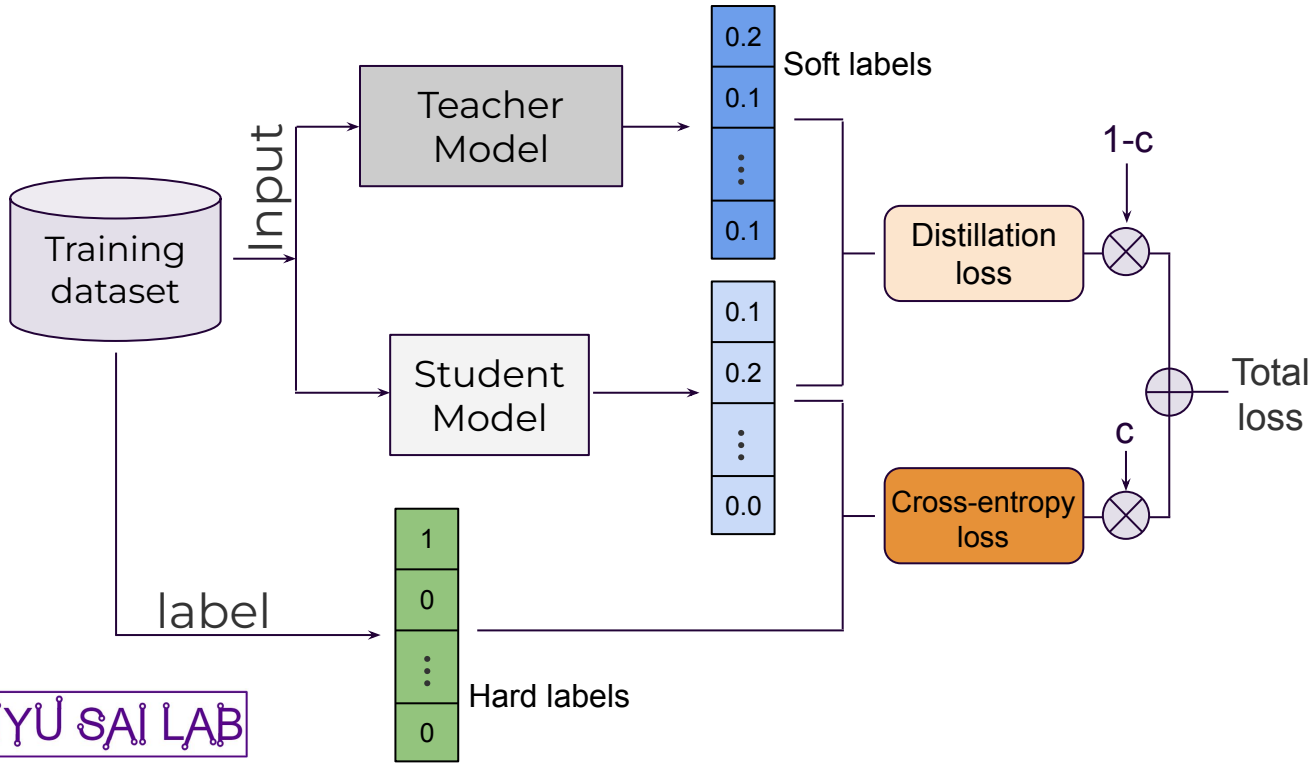
- Transferring knowledge from a large and complex model or set of models to a single, smaller model that can be effectively deployed in real-world scenarios with practical limitations.



NYU SAI LAB

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." *arXiv preprint arXiv:1503.02531* (2015).

Response-Based Knowledge Distillation



- During the training process, only the weights within the student model got updated.
- The teacher model can be either pretrained or trained together with student.
- c tends to be close to 1: ~0.9.

NYU SAI LAB

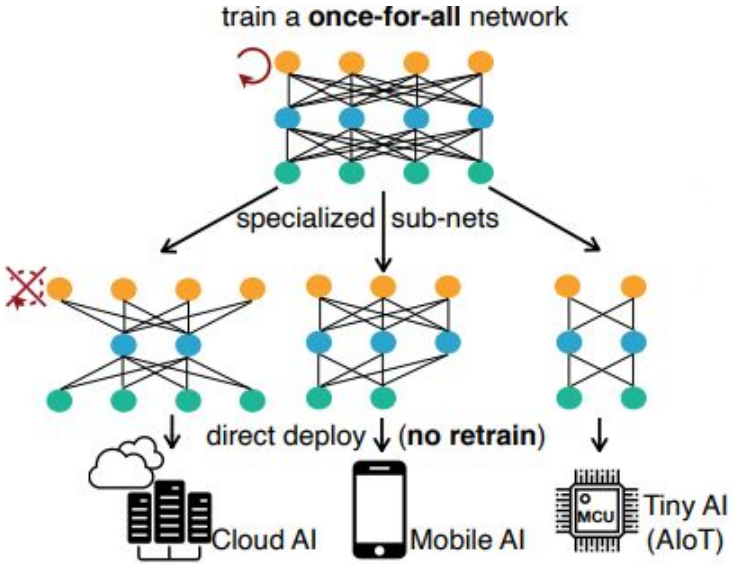
KL Divergence

- In mathematical statistics, the Kullback–Leibler (KL) divergence, denoted $D_{KL}(P \,//\, Q)$, is a measure of how much a model probability distribution Q is different from a true probability distribution P, where both P and Q are discrete.

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x)\ \log\left(\frac{P(x)}{Q(x)}\right)$$

- If P(x) = Q(x) for all x, then the KL divergence equals 0.
- Otherwise, KL divergence is greater than 0.

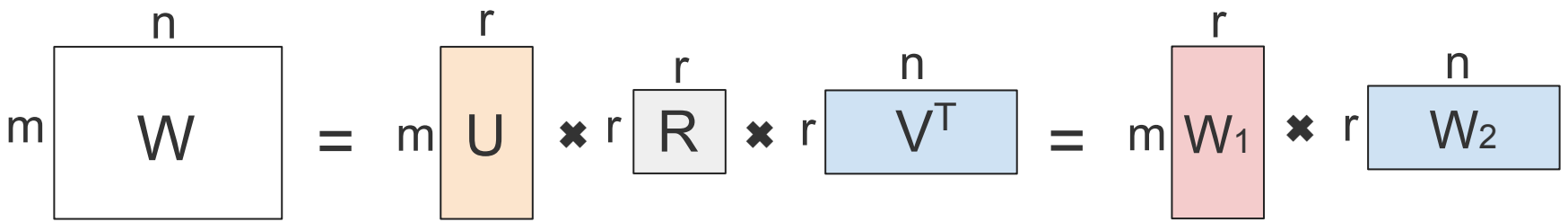
Once-For-All: Train One Network And Specialize It for Efficient Deployment



- A supernet is train which contained multiple smaller subnetworks.
- All the subnetworks are trained jointly.

Cai, Han, et al. "Once-for-all: Train one network and specialize it for efficient deployment." *arXiv preprint arXiv:1908.09791* (2019).

Singular Value Decomposition



- W: Input matrix
  - m×n matrix
- V (n×r matrix) contains the orthonormal eigenvectors of W^TW
- U (m×r matrix) contains the orthonormal eigenvectors of WW^T
- R: Singular value matrix
  - r×r diagonal matrix, r is the rank of W

Before: mn

After: mr + rn = r(m+n) = min(m,n)(m+n)

assume W is full rank

- Without rank truncation, the number of parameters increases.

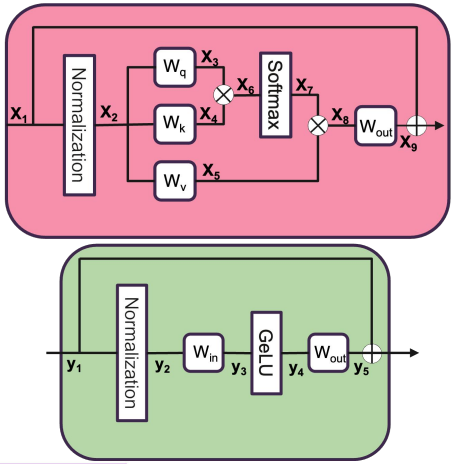
NYU SAI LAB

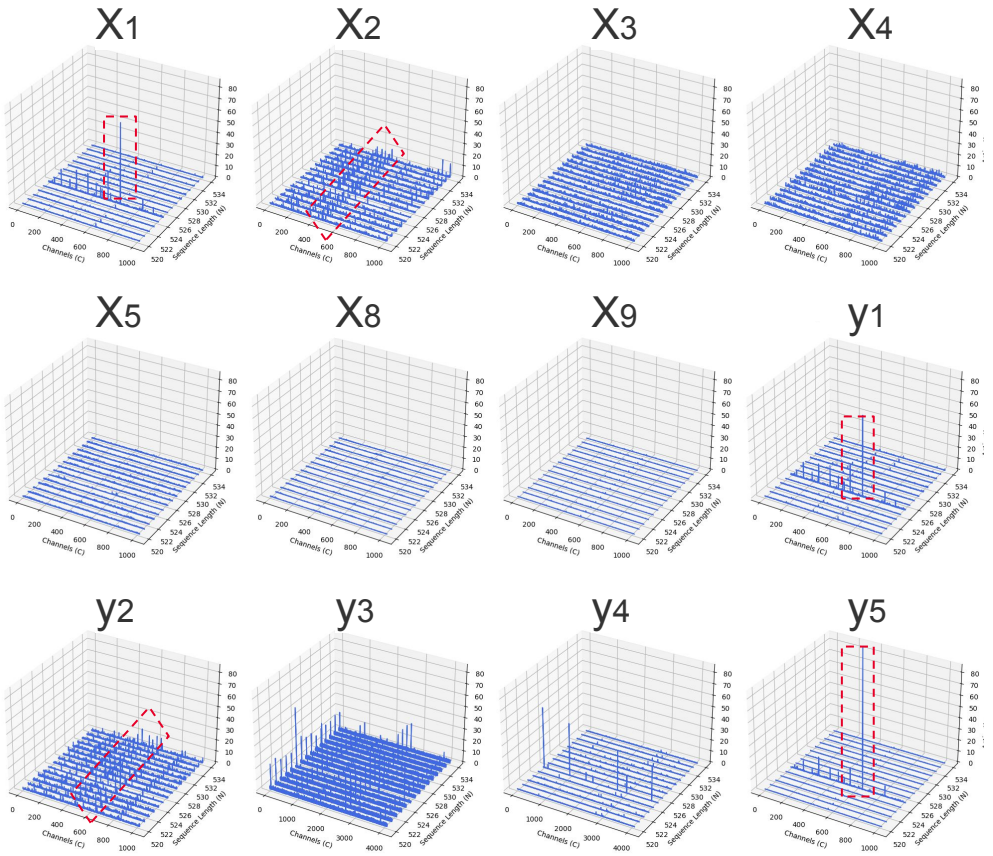
Recap (Lecture 7: P1-P41, P53-P68)

- Outlier Distribution in LLM
  - Massive activation
  - Channelwise outlier
- Quantization and smoothing techniques for large models
- LLM Pruning
- Low-rank decomposition for LLM

Outlier Study: CLIP Activations

- Outliers with large magnitudes appear at positions x1, y1, and y5, referred to as **massive activations**.

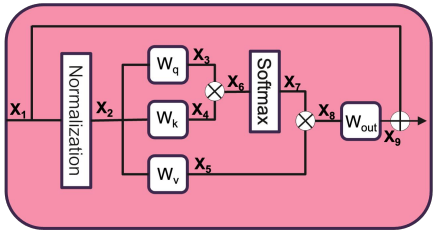




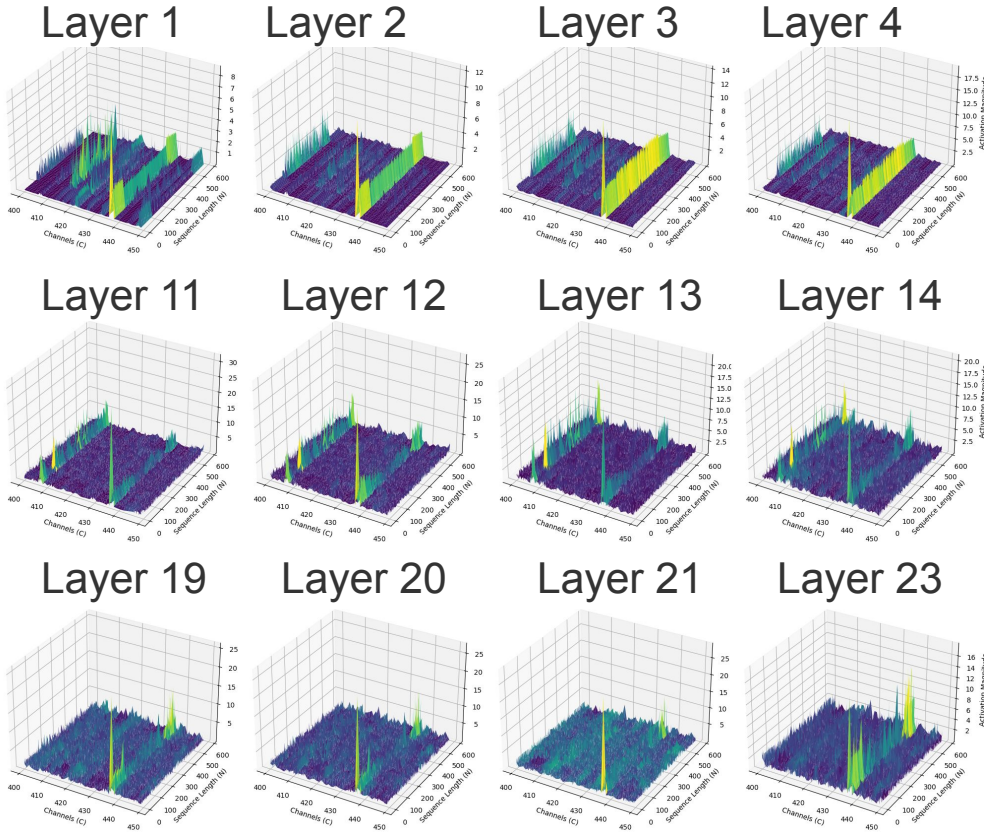
NYU SAI LAB

Outlier Study: CLIP Activations

- 3D plots of x2 across layers.



- x2 exhibits channel wise outlier



NYU SAI LAB

Impact of Massive Activation

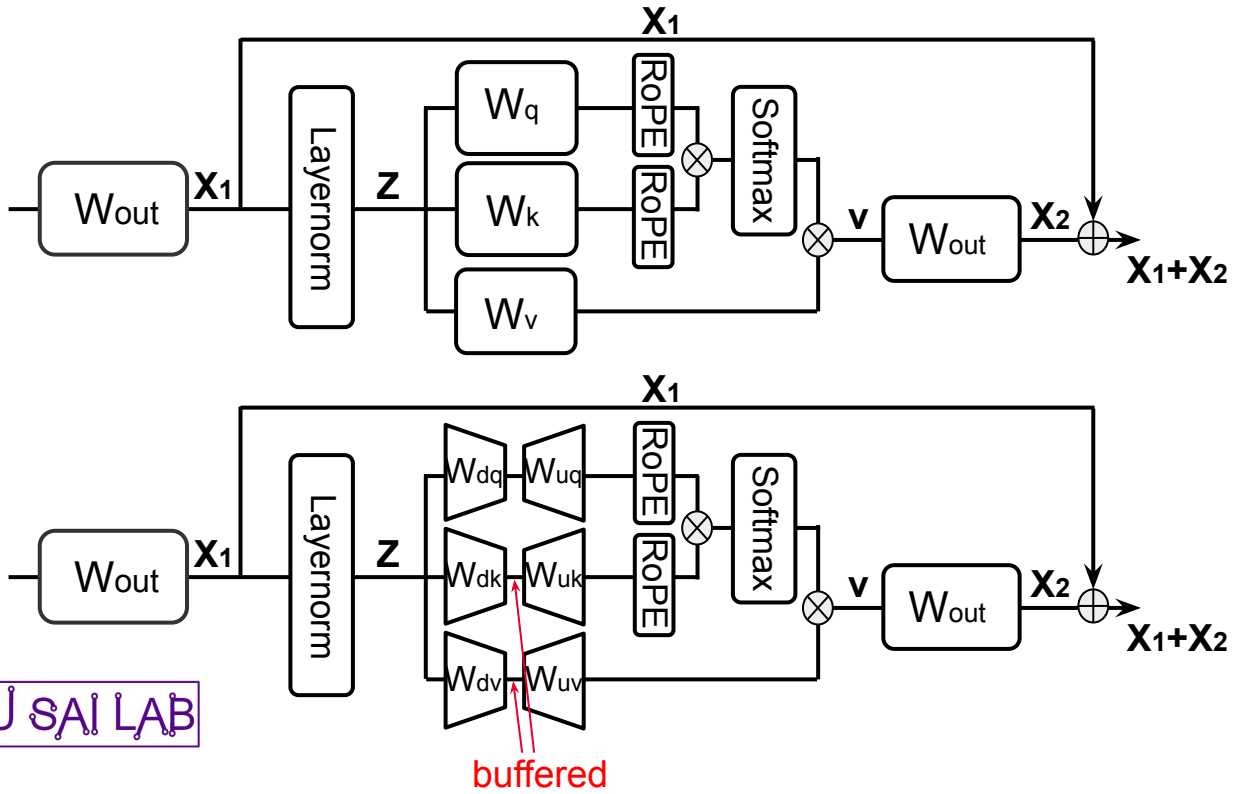
	LLaMA3.2-3B		LLaMA3.1-8B		LLaMA2-13B		GPT-2		Qwen2.5-7B	
Intervention	**WikiText**	**C4**	**WikiText**	**C4**	**WikiText**	**C4**	**WikiText**	**C4**	**WikiText**	**C4**
Original	5.567	10.790	6.941	9.046	4.355	6.405	14.795	19.460	6.520	11.773
TMAs to mean at y_7	1124111.75	21046.82	21281.49	1301562.25	1301562.25	6469.42	14.841	19.560	71216.17	66588.86
TMAs to zeroes at y_7	1138151.23	21951.41	21601.10	1302018.53	1309211.61	7128.32	14.911	19.928	71835.61	67518.35

- The truncation of massive activation will cause the significant accuracy degradation of the LLM.
- Massive activations also occur in other types of foundation models that utilize attention-based architectures.

NYU SAI LAB

Raman, Rahul, Khushi Sharma, and Sai Qian Zhang. "Rethinking the Outlier Distribution in Large Language Models: An In-depth Study." *arXiv preprint arXiv:2505.21670* (2025).

Singular Value Decomposition



SVD decomposition can potentially save the MAC operations, memory storage and KV cache size.

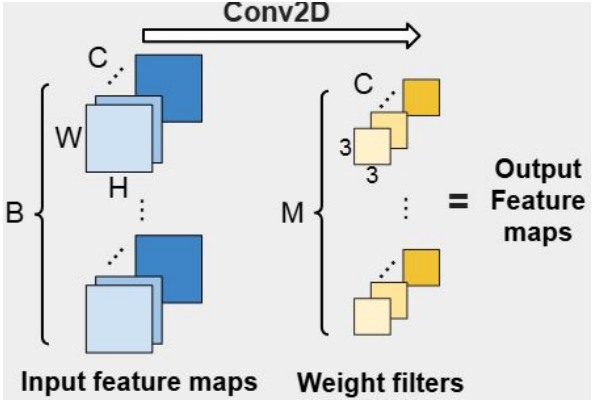
NYU SAI LAB

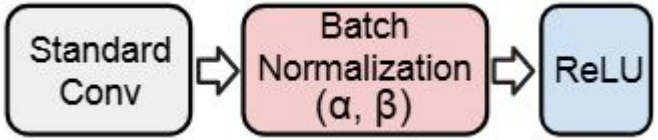
Recap (Lecture 8: P1-P50)

- Efficient training of DNNs
  - Efficient computing
  - Efficient storage
- Parameter efficient finetuning
- Basics of Speculative Decoding

Example Question

Figure below describes a standard Conv2D operation where the input has dimensions B x C x H x W, and the weight filters have sized M x C x 3 x 3. Here, B represents the batch size, C is the number of input channels, and H and W denote the spatial dimensions of the input feature map. Assume a padding size and stride of 1. No bias terms is involved.





1. How many multiply-accumulate (MAC) operations are required for conventional Conv2D?
2. If p% of the filters within the weight are pruned (meaning every elements within the weight filters are set to zero), how does this affect the computational cost of the convolution? Additionally, what is the new total of MAC operations for the convolution?
3. What is the size of the output generated by the batch normalization operation? During inference, is the BN operation explicitly computed? What if changing to layer normalization?

Example Question

as illustrated in figure below. The input to the SA layer is shaped as BxLxE, where B represents the batch size, L denotes the token length, and E is the embedding size. Each of the weight matrices W_Q, W_K, and W_V has dimensions ExE, and the number of attention heads is assumed to be 1.

1. Could you explain the purpose and underlying intuitior of the QK^T operation in the self-attention mechanism?

2. What is the total count of MAC operations in this SA block? Please express it using the terms B, L, and E. How does the computational complexity evolve as the token length L increases? You don't need to account for the cost of nonlinear operations within softmax, layernorm, and scaling operations.