NYU

Lecture 04: Pruning Strategies for Efficient Deep Neural Network

Notes

- Lab1 will be released this week!
- Lab0 will be posted to help you understand DNN pruning.
- The in-class presentation schedule has been updated.

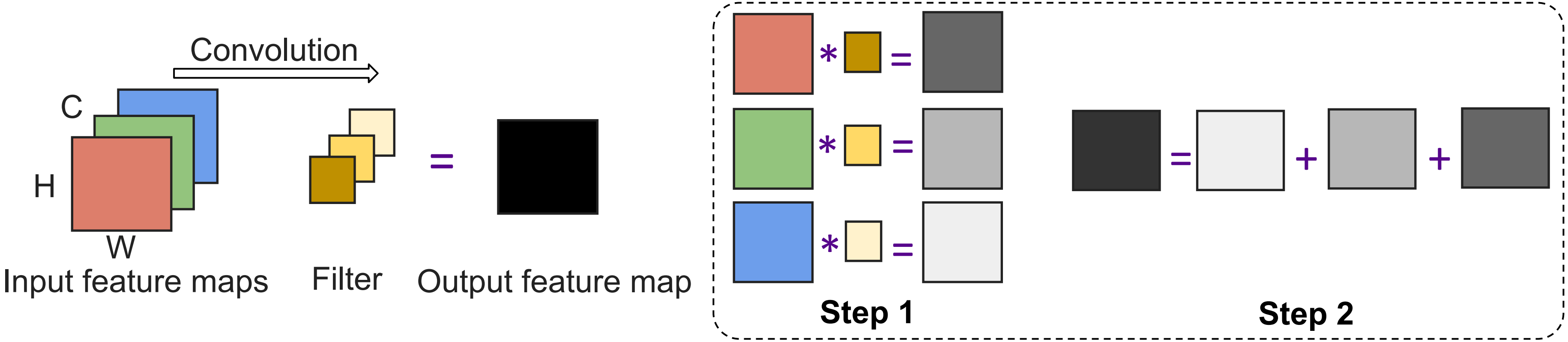
Recap

- Transformer basics
- Bert
- Vision transformer
- Large Language Model
- Self-supervised learning

Topics

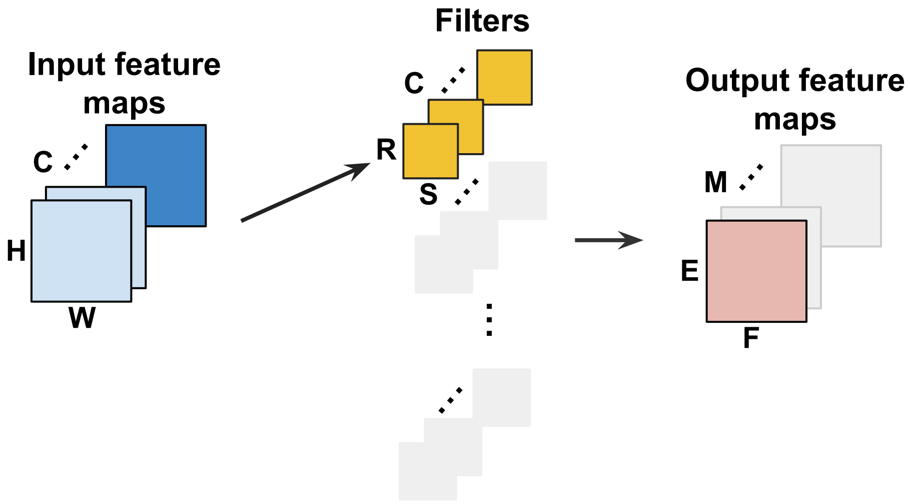
- Why pruning?
  - Reduce running cost
  - Reduce storage
- General pruning techniques
- Transformer pruning

Convolutional Layers



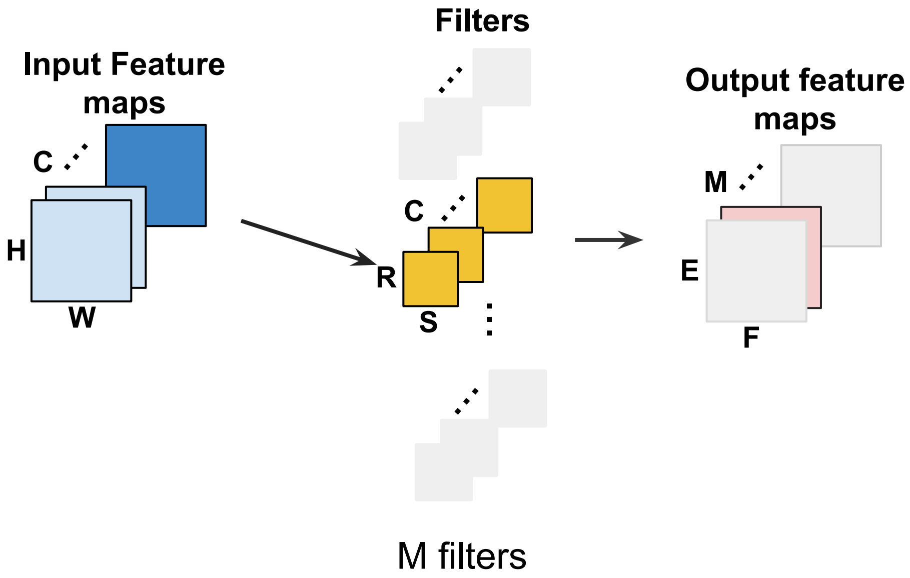
- Core building block of a CNN, it is also the most computational intensive layer.

Convolution

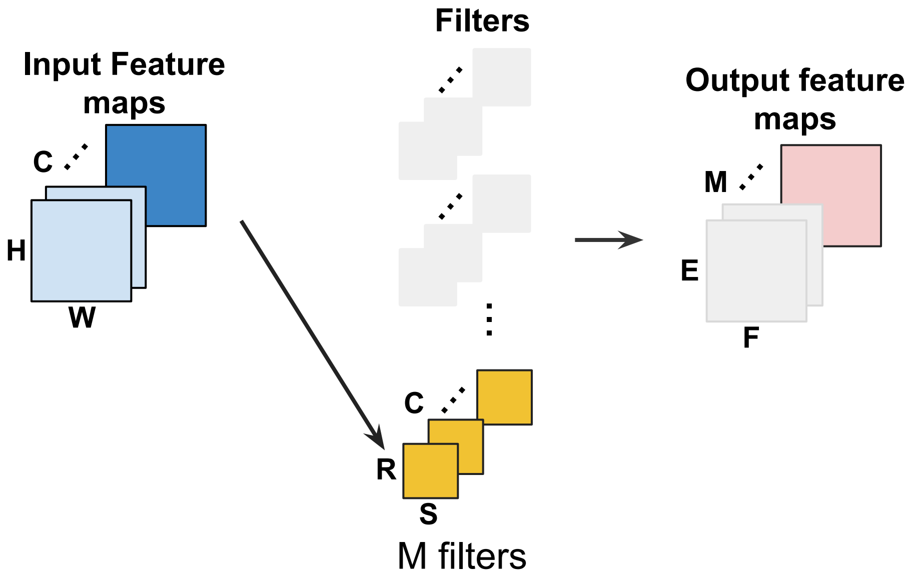


M filters

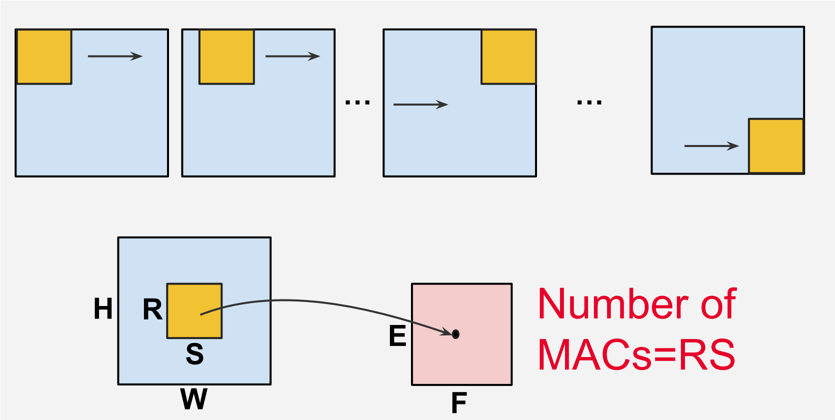
Convolution

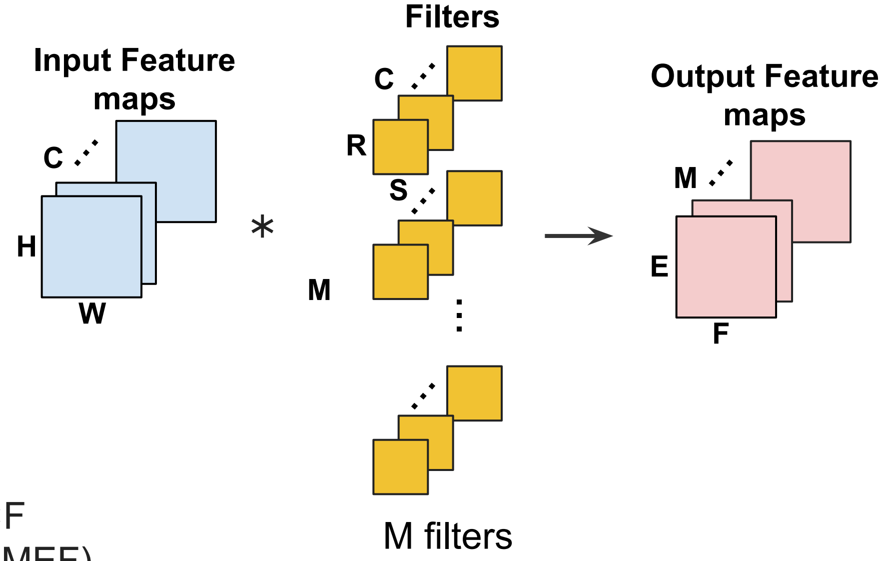


Convolution



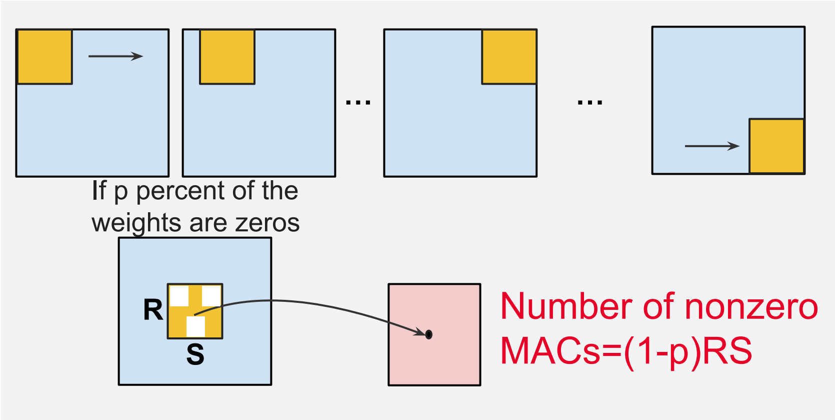
Computational Cost of Convolution





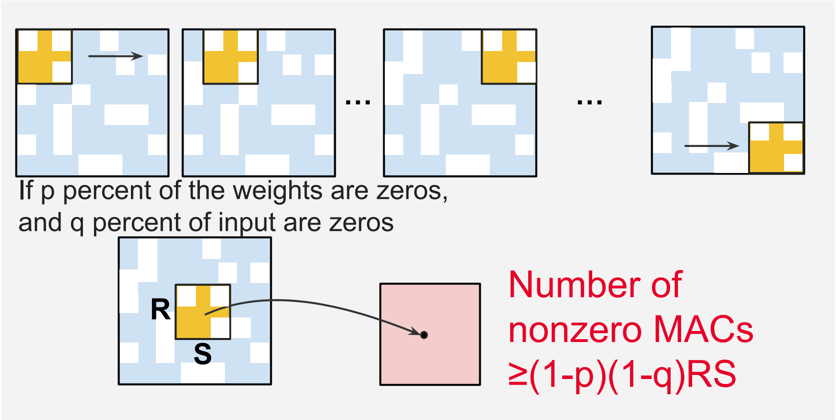
- Number of MACs: M×C×R×S×E×F
- Storage cost: 32×(MCRS+CHW+MEF)
- The input activation and output activations are transient storage, can be eliminated once this layer is finished processing.

Convolution with Sparse Weight



- Number of MACs: (1-p)×M×C×R×S×E×F
- Weight pruning can reduce the computations.
- Sparse weight matrices can be stored more efficiently, which helps minimize memory usage.

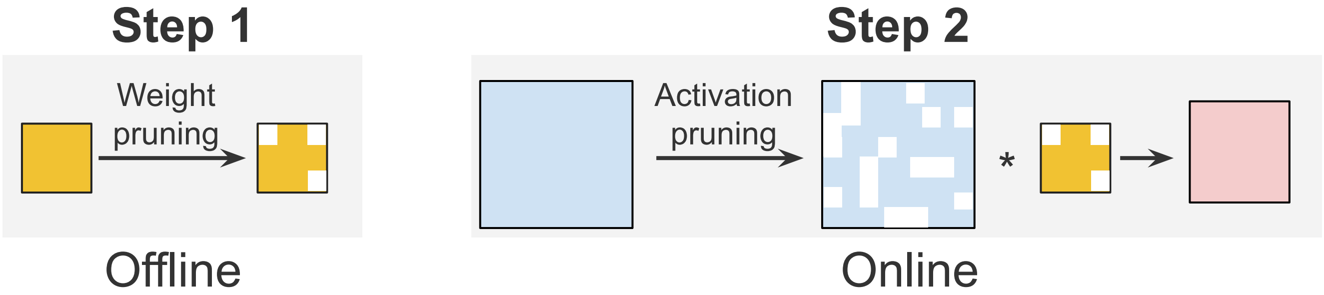
Convolution with Sparse Weight



- 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.

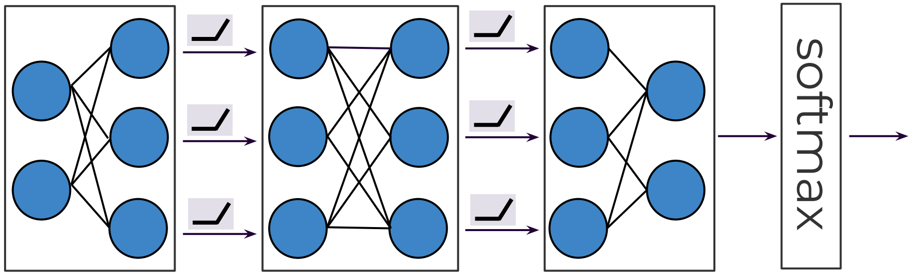
NYU SAI LAB

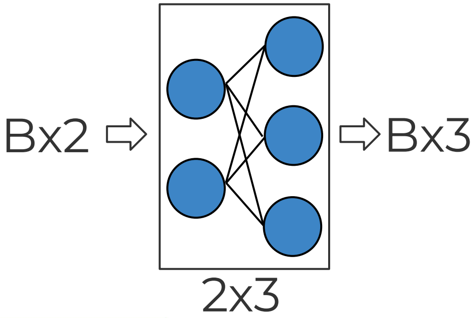
Convolution with Sparse Weight



- Activation sparsity requires online pruning, which leads to additional overhead for sorting.

Computational Cost for MLP

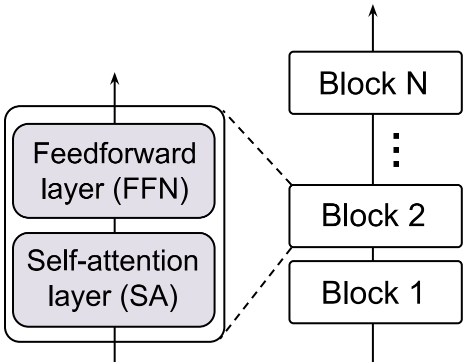




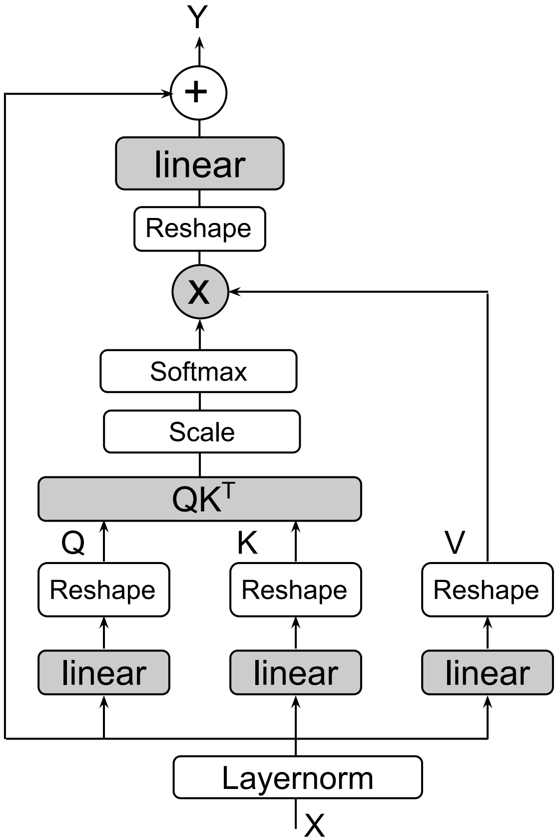
- B is the batch size
- Number of MACs:
  - Bx2x3 = 6B
- Storage cost:
  - 6 x 32 = 192 bits (Weights)
  - (2B + 3B) x 32 bits (Activation)

NYU SAI LAB

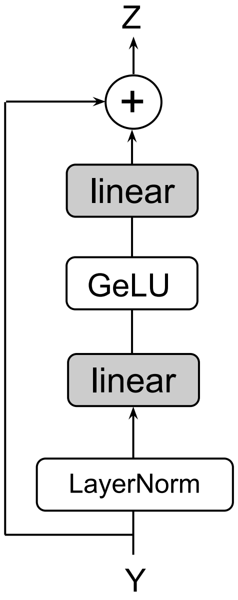
Transformers



- The input sentence has three dimensions:
  - B: batch
  - L: sequence length (number of words)
  - E: embeddings

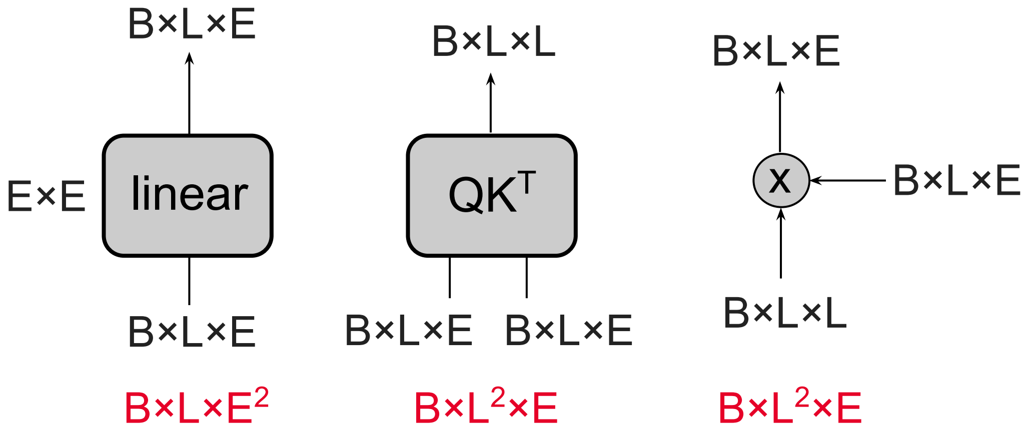


Self attention block (SA)

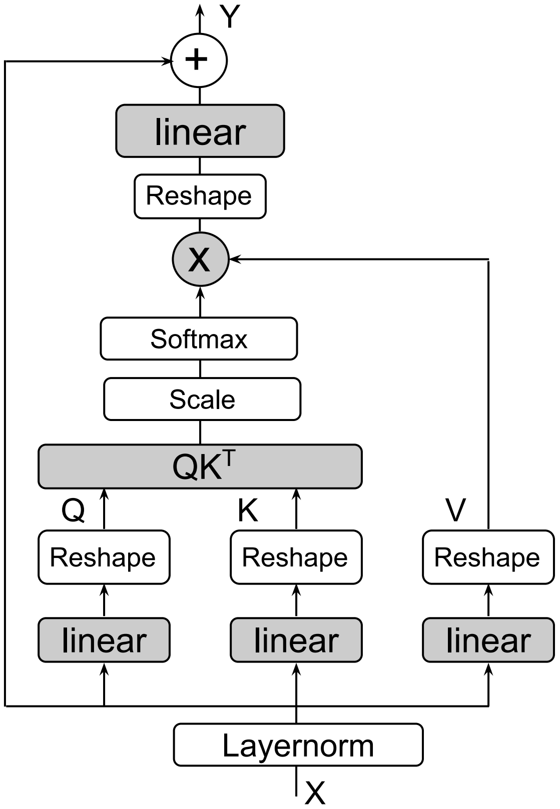


Feed forward block (FFN)

Computational Cost of Transformer

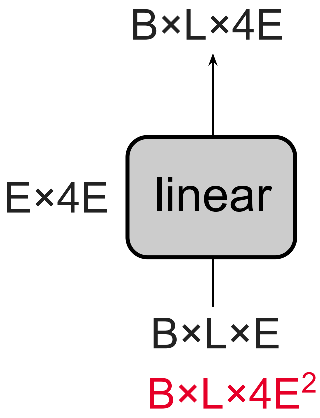


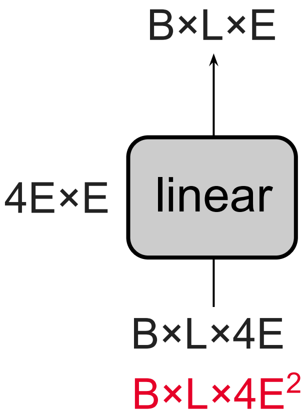
Total = $4B \times L \times E^2 + 2B \times L^2 \times E$

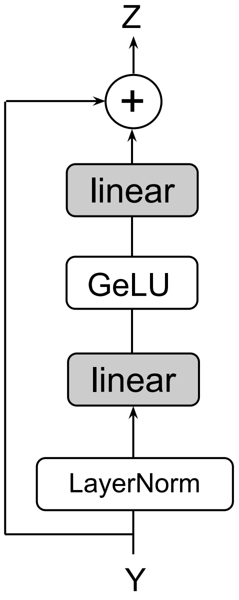


NYU SAI LAB

Computational Cost of Transformer

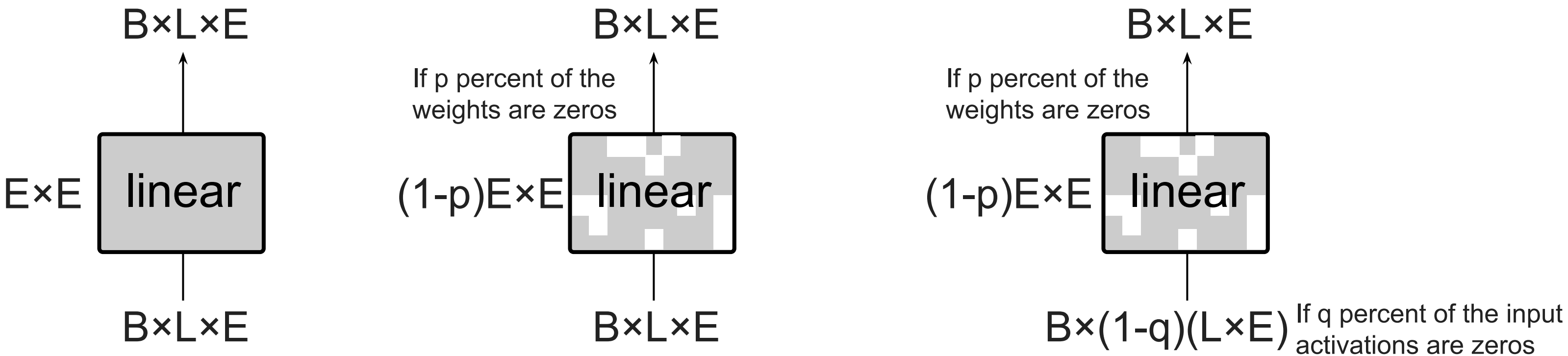






$$\text{Total} = 4B \times L \times E^2 + 2B \times L^2 \times E + 8B \times L \times E^2$$
$$= 12B \times L \times E^2 + 2B \times L^2 \times E$$

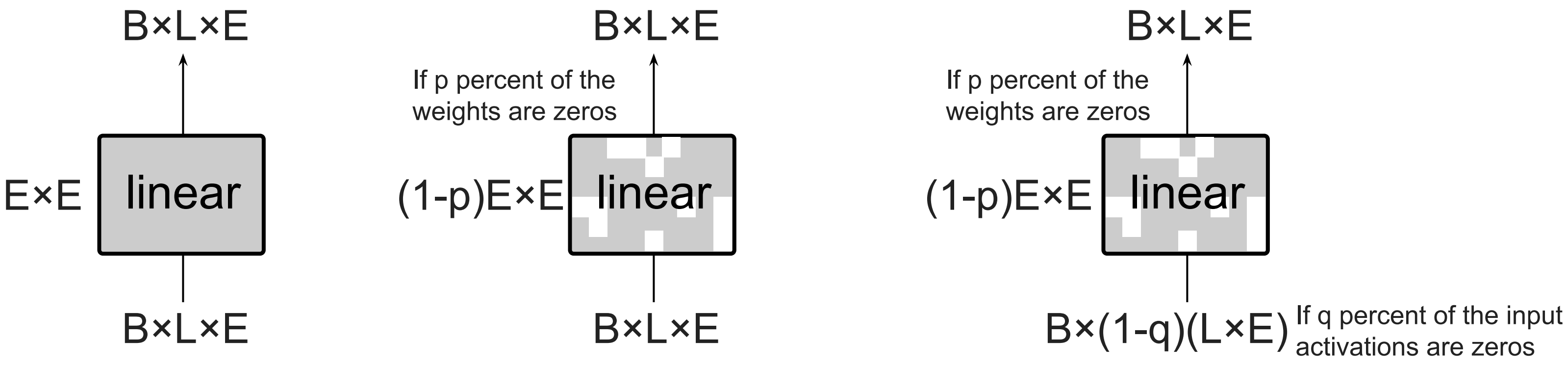
Computational Cost of Transformer



Nonzero MACs
$=(1-p)\times B\times L\times E^2$

Nonzero MACs
$\geq(1-p)(1-q)\times B\times L\times E^2$

Computational Cost of Transformer



Nonzero MACs
$=(1-p)\times B\times L\times E^2$

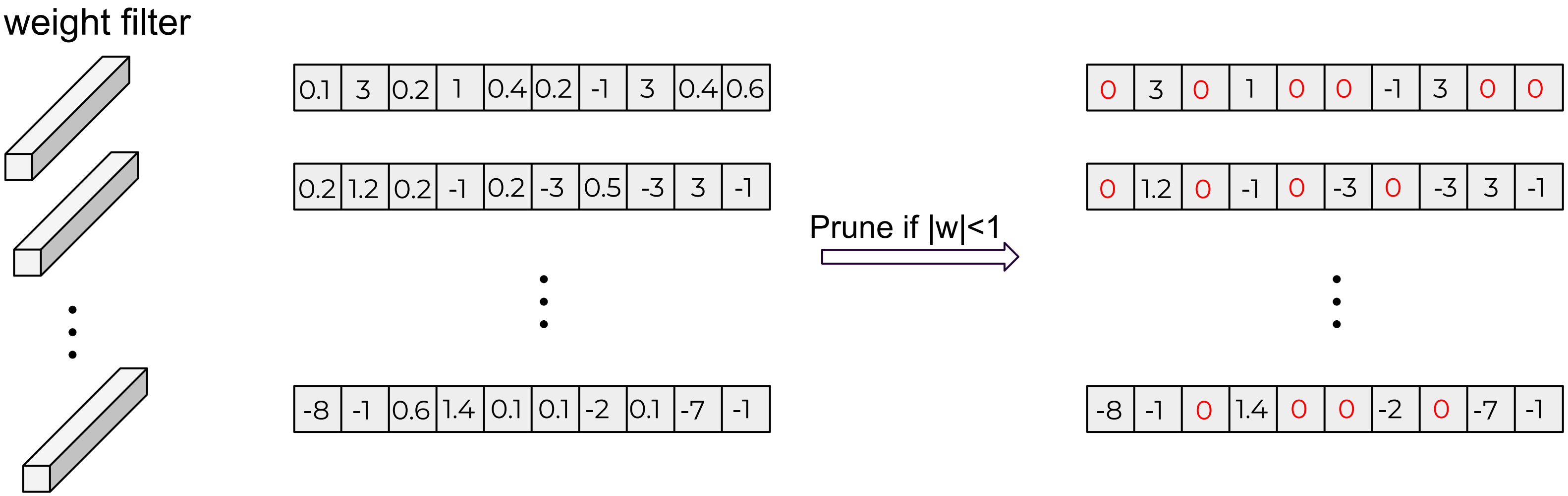
Nonzero MACs
$\geq(1-p)(1-q)\times B\times L\times E^2$

NYU SAI LAB

Topics

- Why pruning?
  - Reduce running cost
  - Reduce storage
- General pruning techniques
- Transformer pruning

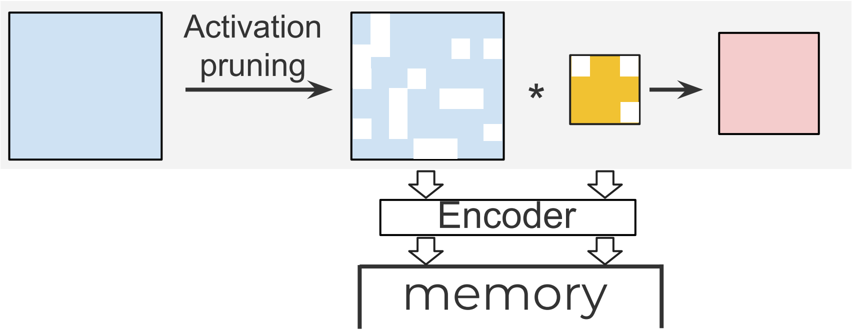
Pruning



- Pruning reduces both computational demands and storage costs.

Benefit of Pruning

- Reduce computational complexity
    - To support sparse matrix with random sparsity pattern, specialized hardware is required.
- Reduce the storage complexity
    - To achieve it, we need to encode the sparse weights.
    - Encoding the activations requires additional computation.



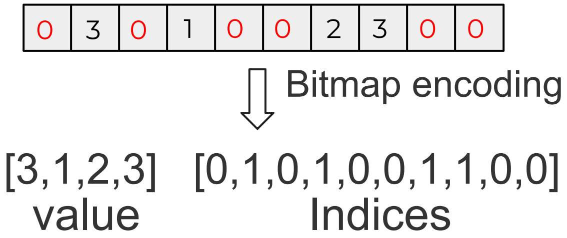
NYU SAI LAB

Sparse Matrices Encodings

- Efficient encoding scheme for sparse matrix storage.
  - Bitmap
  - Run Length Encoding (RLE)
  - Coordinate format (COO)
  - Compressed sparse row (CSR), Compressed sparse column (CSC)

NYU SAI LAB

Bitmap Encoding

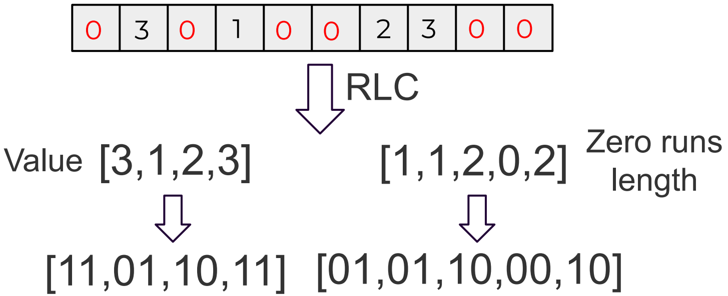


- In summary, the storage cost of bitmap encoding (in bits) is:
  $(1-p)\times L\times n + L$
- n: number bits per value
- L: number of elements
- p: sparsity (%)

- Bitmap is effective for compressing the tensors of low or moderate sparsity.
- Encoding cost is low.

Run Length Encoding (RLC)

- Record the values and length of zero runs between the values.
- Assume 2 bits are used to encode the length of zero runs (0-3).
- Each value requires 2 bits.

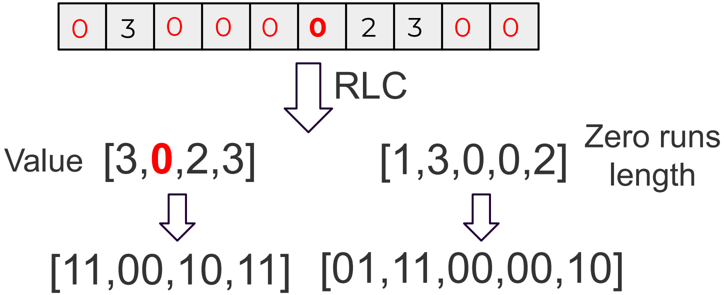


- RLC can reduce storage requirement when sparsity is moderate.

NYU SAI LAB

Run Length Encoding (RLC)

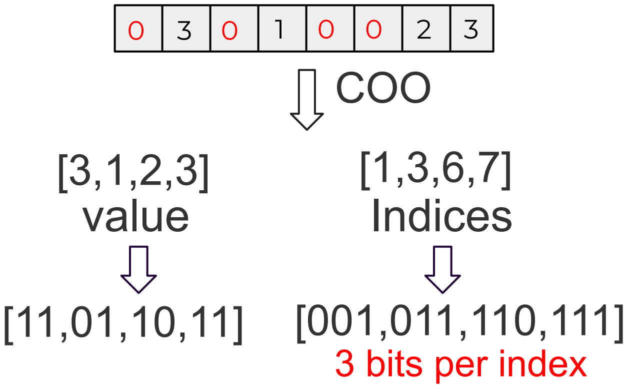
- Record the values and length of zero runs between the values.
- Assume 2 bits are used to encode the length of zero runs (0-3).
- Each value requires 2 bits.



- RLC can reduce storage requirement when sparsity is moderate.
- Hard to formulate it analytically.

NYU SAI LAB

Coordinate Format (COO)



- COO is efficient with the sparsity level is extremely high.
- The storage cost (in bits) is: $(1-p)\times L\times n + (1-p)\times L\times \mathrm{Ceil}(\log_2 L)$
- n: number bits per value
- L: number of elements
- p: sparsity (%)

NYU SAI LAB

Coordinate Format (COO)

0	3	0	1
0	2	3	0

⇩ COO

value	x_Indices	y_Indices
[3,1,2,3]	[0,0,1,1]	[1,3,1,2]
⇩	⇩	⇩
[11,01,10,11]	[01,11,01,10]	[00,00,01,01]

2 bits per index

- COO is efficient with the sparsity level is extremely high.

Compressed Sparse Row/Column (CSR/CSC)

$$\begin{pmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 40 & 0 & 0 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{pmatrix}$$

Sparse matrix

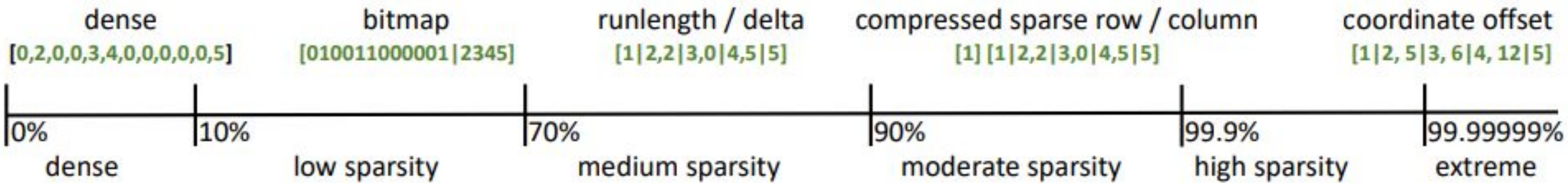
```
V         = [ 10 20 30 40 50 60 70 80 ]
COL_INDEX = [  0  1  1  3  2  3  4  5 ]
ROW_INDEX = [  0  2  4  7  8 ]
```

Encoded form

- CSR/CSC is also suitable for matrices with high sparsity.
- The row index specifies the amount of cumulative nonzero values across each row.

NYU SAI LAB

Encoding Approaches Tailored to Different Sparsity Levels



- Different encoding scheme can be applied for different sparsity levels.

NYU SAI LAB

Hoefler, Torsten, et al. "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks." *Journal of Machine Learning Research* 22.241 (2021): 1-124.

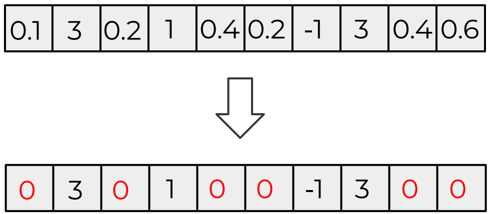
Topics

- Why pruning?
  - Reduce running cost
  - Reduce storage
- General pruning techniques
- Transformer pruning

Pruning Criteria: Magnitude Pruning

- We can prune the weights using the importance score:
  - Magnitude
  - Gradient
  - Hessian
  - …

Magnitude pruning

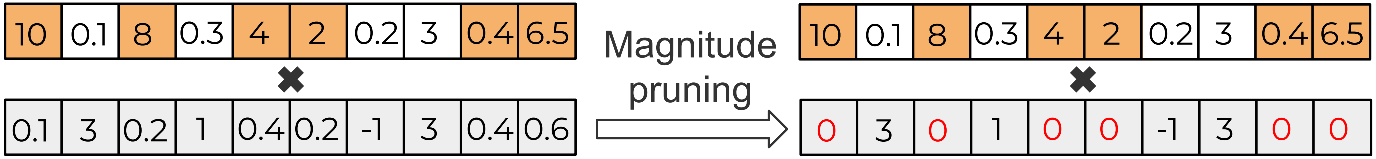


$$m_i = \begin{cases} 1 : if\ \|w_i\|_1 \geq a \\ 0 : if\ \|w_i\|_1 < a \end{cases}$$

NYU SAI LAB

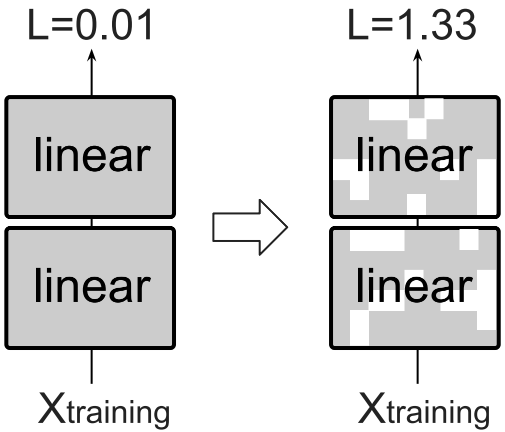
Drawbacks of Magnitude Pruning

- The major drawback of magnitude based pruning is that it does not consider the impact of the input when making the pruning decision.



NYU SAI LAB

Pruning Criteria: Training Loss Change

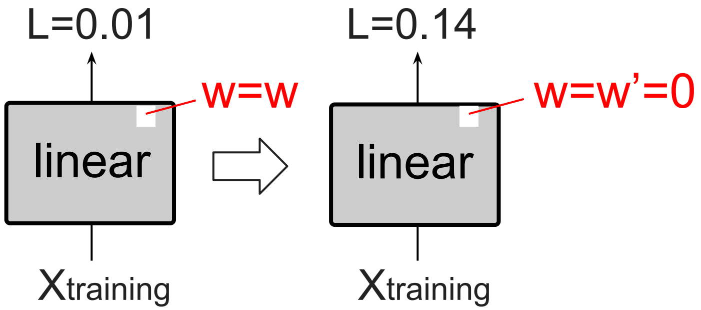


$$\mathcal{L}(w) = \sum_{(x,y)\in D_{training}} l(y, F_w(x))$$

$D_{training}$: is the training dataset
$F_w(.)$: neural network function with parameter w.
L(.): Training loss function

- Another pruning principle is to minimize the impact on training loss as much as possible.
- For a trained DNN, L(.) remains low.
- Training loss is typically computed over either a subset or the entire training dataset.

Pruning Criteria: Training Loss Change



$$\mathcal{L}(w) = \sum_{(x,y)\in D_{training}} l(y, F_w(x))$$

$D_{training}$: is the training dataset
$F_w(.)$: neural network function with parameter w.
l(.): loss function

- Another pruning principle is to minimize the impact on training loss as much as possible.
- For a trained DNN, L(.) remains low.

NYU SAI LAB

Pruning Criteria: Training Loss Change

- Pruning criteria:
  - Keep the change on training loss as small as possible
  - Let L(.) denote the training loss
  - For trained DNN, L(.) will be low.

$$\mathcal{L}(w') = \mathcal{L}(w) + \nabla\mathcal{L}(w)^T(w - w')$$

- If w is pruned, then we have w’=0:

$$\mathcal{L}(0) - \mathcal{L}(w) = \frac{d\mathcal{L}}{dw}w$$

- We can use it as the pruning criteria
  - Sort the weight based on the product of gradient and its value.

NYU SAI LAB

Pruning Criteria: Training Loss Change

$$\mathcal{L}(w') = \mathcal{L}(w) + \nabla\mathcal{L}(w)^T(w - w')$$
$$+\frac{1}{2}(w - w')^T\nabla^2\mathcal{L}(w)(w - w')$$

- When reflecting on each individual value, the pruning criteria becomes:

$$\mathcal{L}(0) - \mathcal{L}(w) = \frac{d\mathcal{L}(w)}{dw}w + \frac{1}{2}\frac{d^2\mathcal{L}(w)}{dw^2}w^2$$

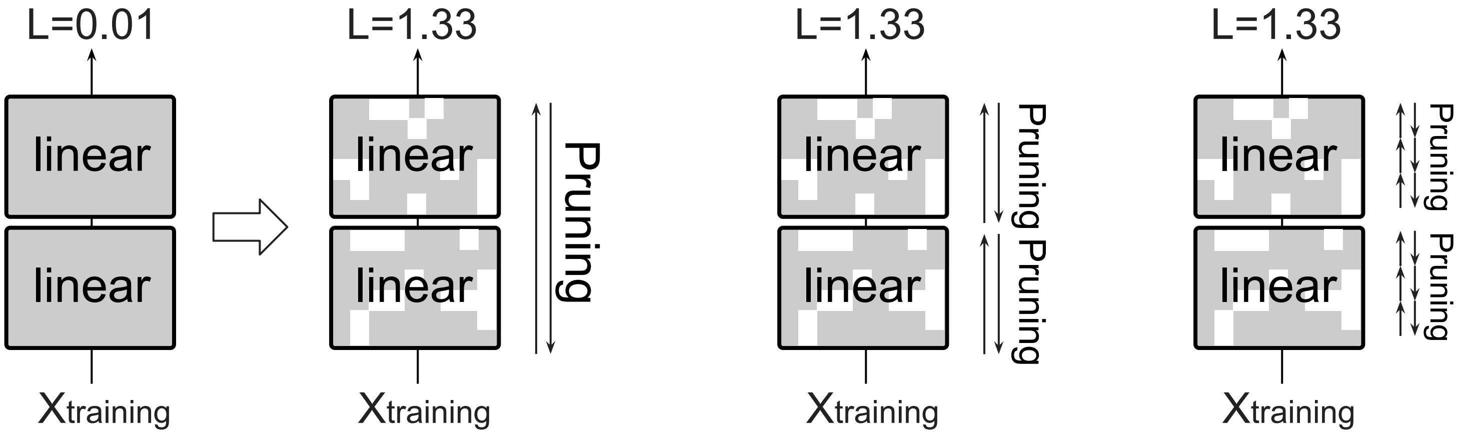
The gradient is usually estimated to zero.

- Multiple approaches have been propose to estimate the Hessian:
  - Empirical Fisher (Outer Product of Gradients)

$$H(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} g_i g_i^\top, \quad g_i = \nabla_\theta \log p(x_i \mid \theta)$$

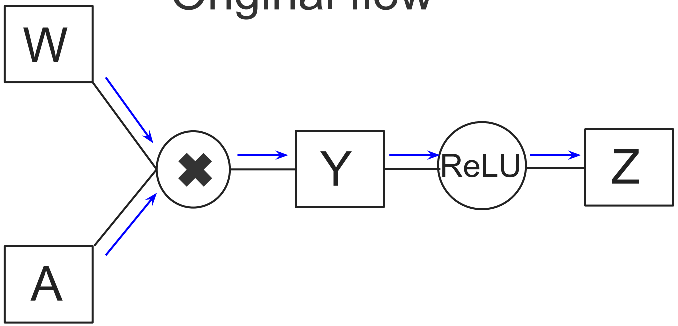
LeCun, Yann, John Denker, and Sara Solla. "Optimal brain damage." *Advances in neural information processing systems* 2 (1989).

Granularity of Pruning



Computational Flow of Pruning

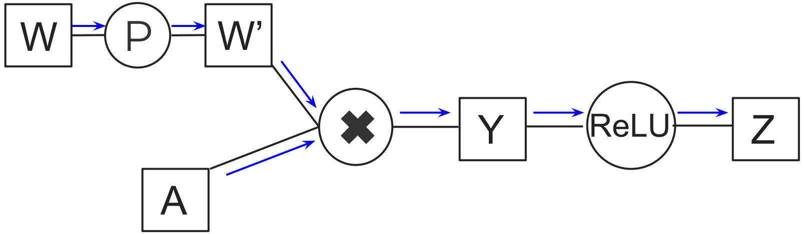
Original flow



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 pruning



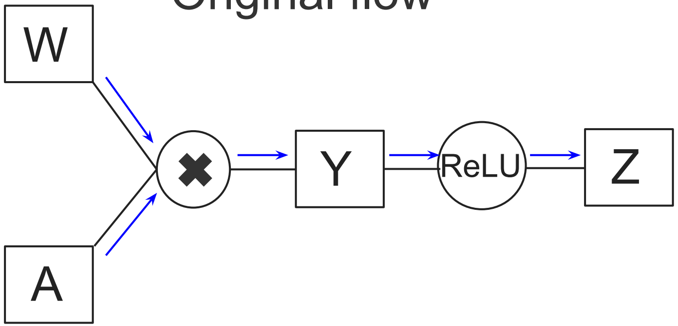
$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Z}\frac{\partial Z}{\partial Y}\frac{\partial Y}{\partial W'}\frac{\partial W'}{\partial W}$$

$\frac{\partial W'}{\partial W}$ = 0 if W is pruned, otherwise = 1

NYU SAI LAB

Computational Flow of Pruning

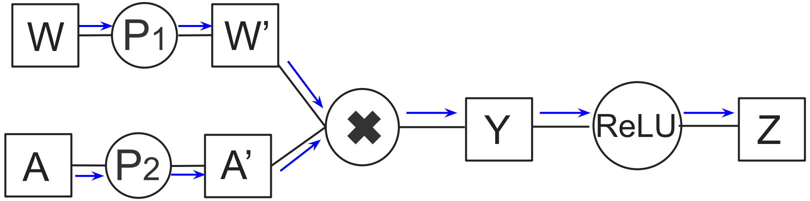
Original flow



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 pruning

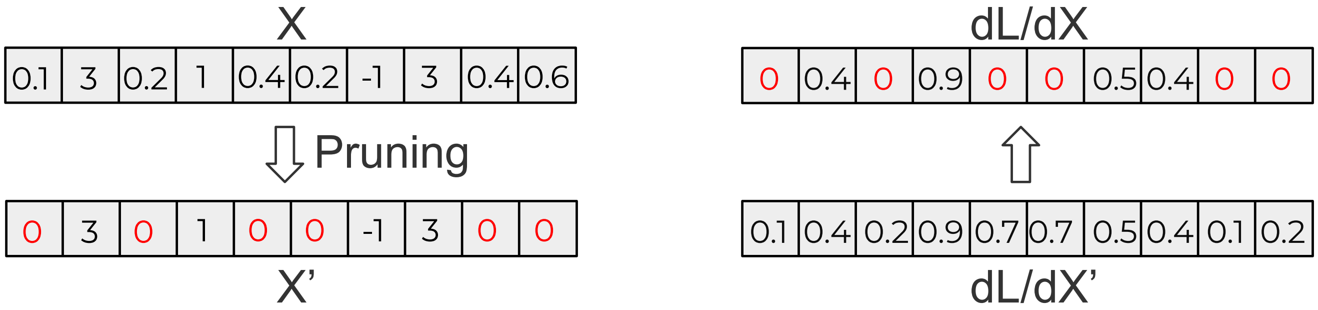


$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Z}\frac{\partial Z}{\partial Y}\frac{\partial Y}{\partial W'}\frac{\partial W'}{\partial W}$$

$\frac{\partial W'}{\partial W}$ = 0 if W is pruned, otherwise = 1

$$\frac{dL}{dA} = \frac{dL}{dZ}\frac{dZ}{dY}\frac{dY}{dA}\frac{dA'}{dA}$$

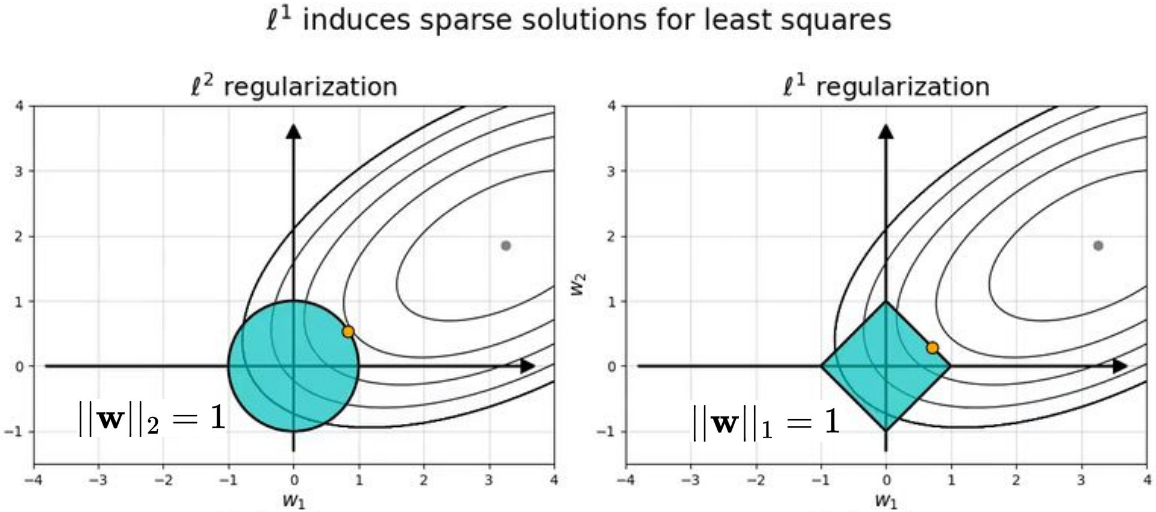
Backward Pass of Pruning Operation



- During backward pass, the gradients of the pruned elements are masked.

NYU SAI LAB

Regularization-based Pruning



$$|\mathbf{x}| = \sqrt{\sum_{k=1}^{n} |x_k|^2}$$

$$|\mathbf{x}|_1 = \sum_{r=1}^{n} |x_r|$$

$$min \quad \mathcal{L}(w) + \lambda||\mathbf{w}||_2 \qquad min \quad \mathcal{L}(w) + \lambda||\mathbf{w}||_1$$ Lasso

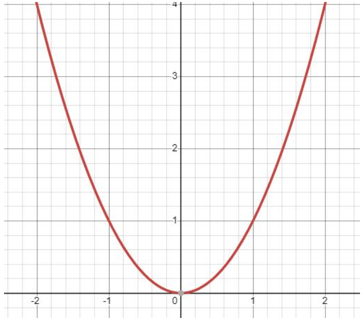
- Add this term can make DNN naturally select the unimportant weight during the training process.

NYU SAI LAB

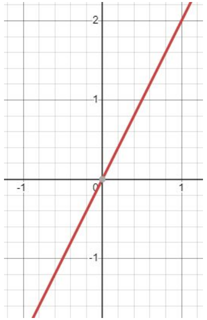
Source:https://satishkumarmoparthi.medium.com/why-l1-norm-creates-sparsity-compared-with-l2-norm-3c6fa9c607f4#:~:text=The%20reason%20for%20using%20the,and%20thus%20a%20sparse%20solution.

Regularization-based Pruning

$y = x^2$

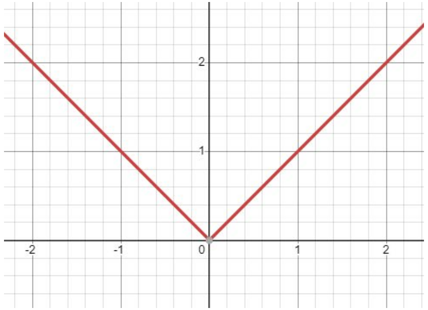


→

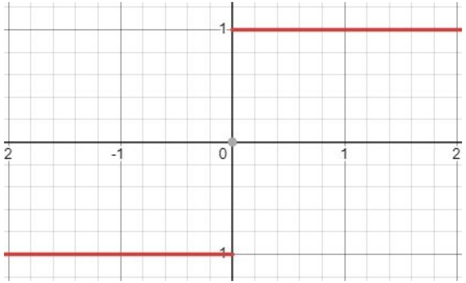


$y = 2x$

$y = |x|$



→



$y = \text{sign}(x)$

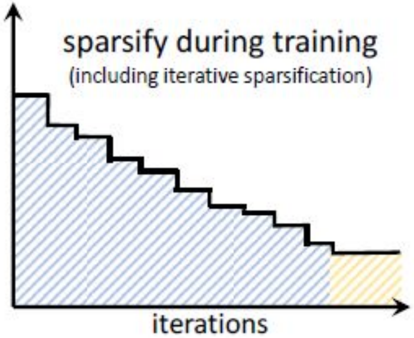
NYU SAI LAB

Taxonomy of Pruning

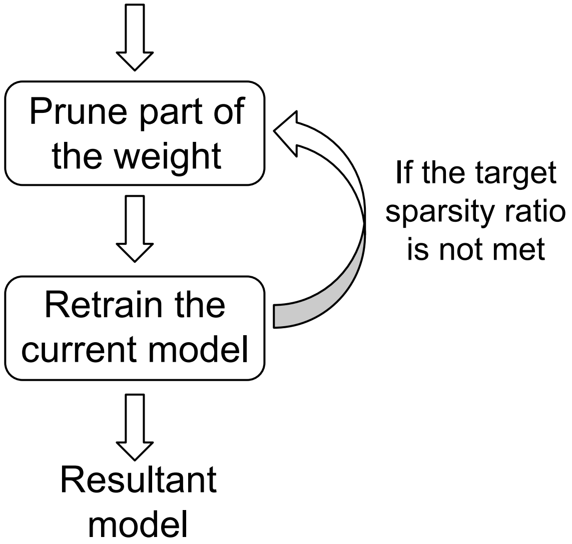
- Pruning techniques can be classified from different perspectives
  - Iterative pruning, zero-shot pruning
  - Structured pruning, unstructured pruning, N:M pruning
  - Weight pruning, activation pruning
  - Static pruning and dynamic pruning
  - Pruning for inference, pruning for training

When to Prune?

Train → Prune
(zero-shot pruning)



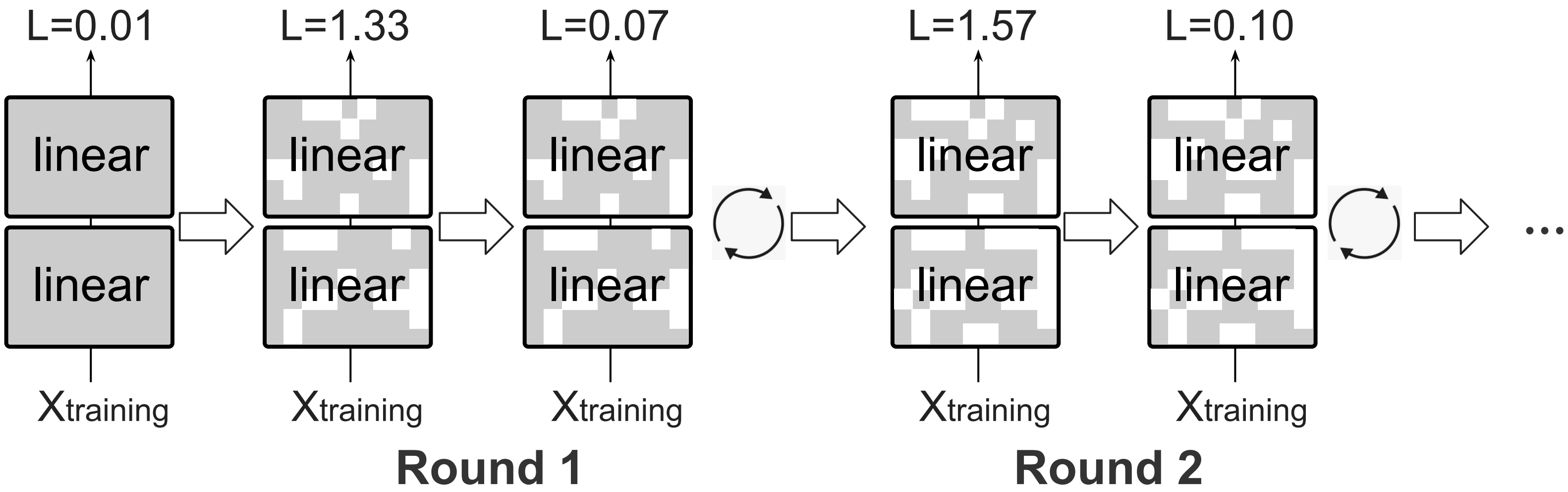
Train⇄Prune
(Iterative pruning)



- Usually interactive pruning has the best accuracy performance, however, it also requires multiple rounds of training and computational cost.
- Zero-shot pruning also termed post-training pruning.

NYU SAI LAB

Iterative Pruning



NYU SAI LAB

Lottery Ticket Hypothesis

"A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations. "

NYU SAI LAB

Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv 2018." *arXiv preprint arXiv:1803.03635* (1810).

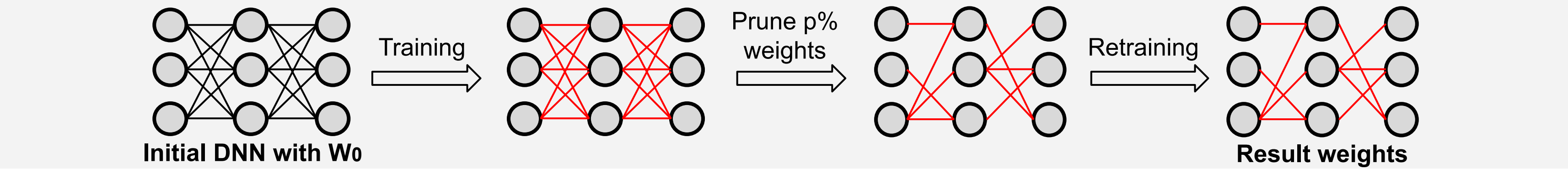
How to Find the Winning Tickets?

- **Iterative Magnitude Pruning (IMP):**
  - Initialized DNN with random weights w_0.
  - While the sparsity level has not reached:
    - Train the DNN with k epochs until convergence
    - prune p% of the nonzero weights.
    - Reinitialize the remaining weights using the values in w_0, finetune the remaining weights for k epochs (**Rewind**).
  - Return the weights.
- Later work has shown that rewind to w_i (i is small) works better for larger networks.

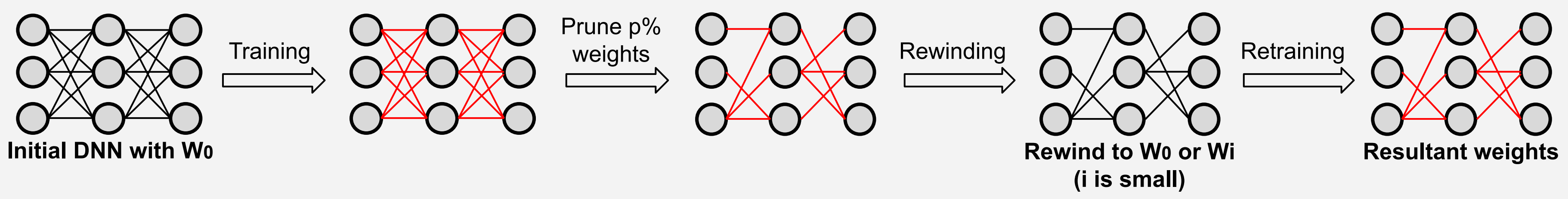
Frankle, Jonathan, et al. "Stabilizing the lottery ticket hypothesis." *arXiv preprint arXiv:1903.01611* (2019).

Weight Rewinding

Conventional iterative pruning



Conventional iterative pruning with weight rewinding



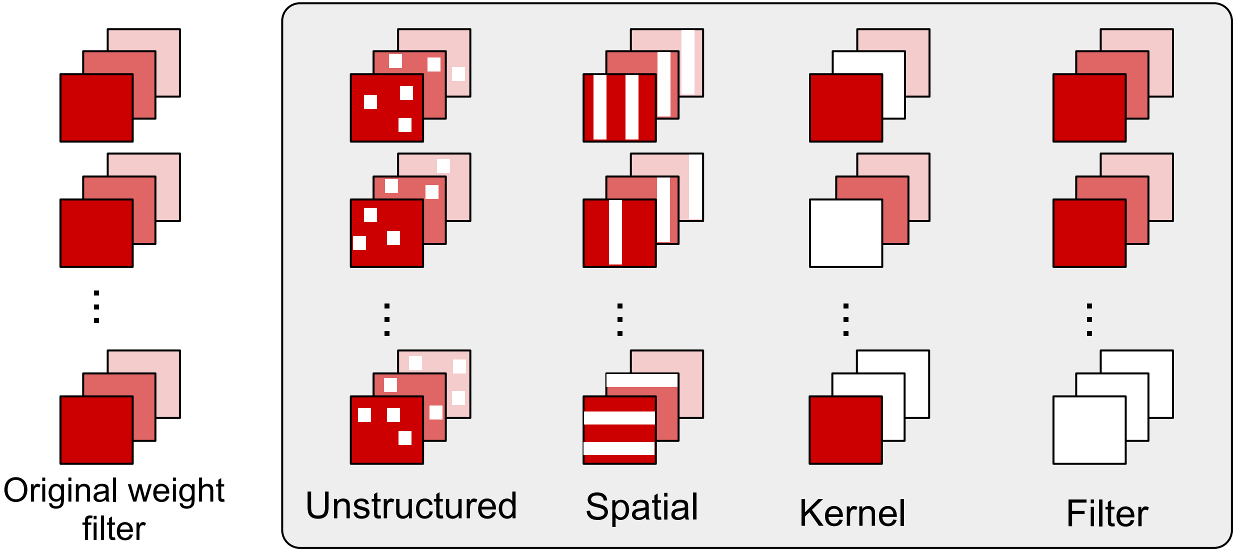
- The pruned architecture itself, rather than a set of inherited “important” weights, is more crucial to the accuracy in the final model, which suggests that in some cases pruning can be useful as an architecture search paradigm.

Liu, Zhuang, et al. "Rethinking the value of network pruning." *arXiv preprint arXiv:1810.05270* (2018).

Taxonomy of Pruning

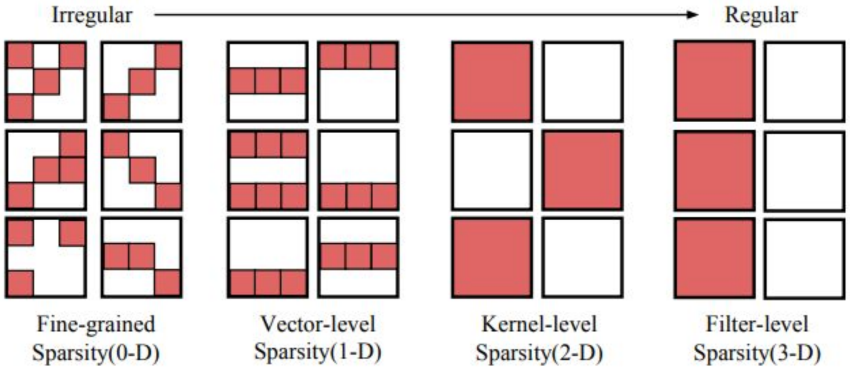
- Pruning techniques can be classified from different perspectives
  - Iterative pruning, zero-shot pruning
  - Structured pruning, unstructured pruning, N:M pruning
  - Weight pruning, activation pruning
  - Static pruning and dynamic pruning
  - Pruning for inference, pruning for training

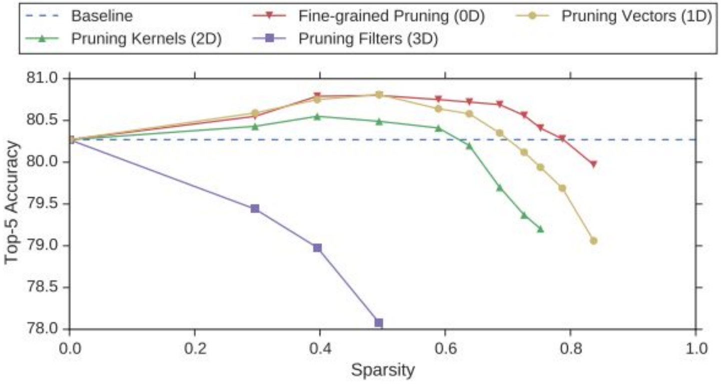
Unstructured/Structured Pruning



- Structured pruning is amenable to hardware performance, due to the regular sparsity distribution.

Unstructured/Structured Pruning

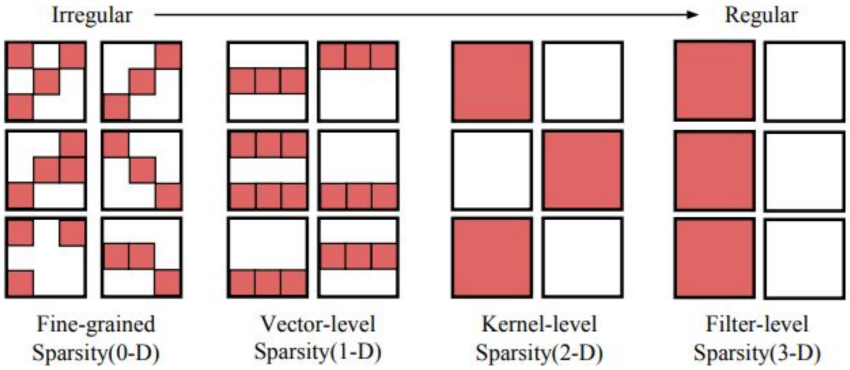


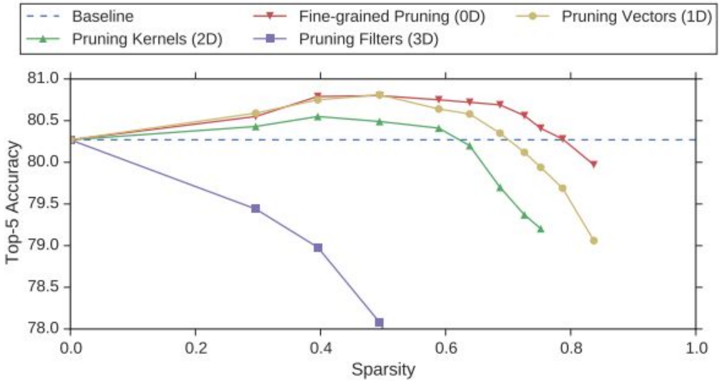


- Unstructured pruning has a better accuracy than structured pruning.
- We can apply the same method as the unstructured pruning to prune a group of parameters.

Mao, Huizi, et al. "Exploring the granularity of sparsity in convolutional neural networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017.

Unstructured/Structured Pruning



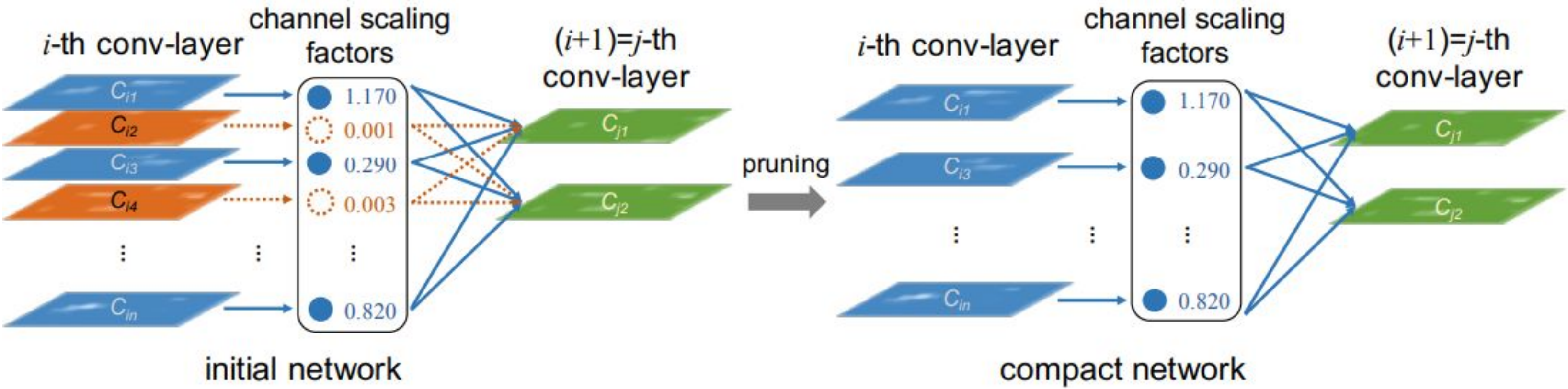


- In magnitude-based weight filter pruning, we first compute the sum of absolute weight values within each filter, then rank the filters and prune those with the smallest sums.
- Similarly, structured pruning can be performed by assessing the importance of a weight group through the sum of their importance scores.

NYU SAI LAB

Mao, Huizi, et al. "Exploring the granularity of sparsity in convolutional neural networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017.

Network Slimming



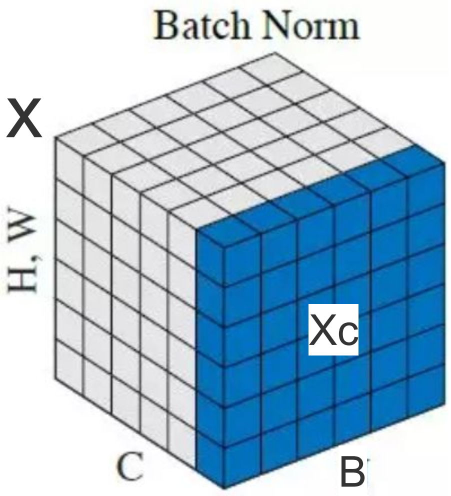
- We associate a scaling factor (from a batch normalization layer) with each filter in convolutional layers. Sparsity regularization is imposed on these scaling factors during training to automatically identify unimportant filters.

$$L = \sum_{(x,y)} l(f(x, W), y) + \lambda \sum_{i} |pi|$$

NYU SAI LAB

Liu, Zhuang, et al. "Learning efficient convolutional networks through network slimming." *Proceedings of the IEEE international conference on computer vision*. 2017.

Batch Normalization



X: HW × B × C

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

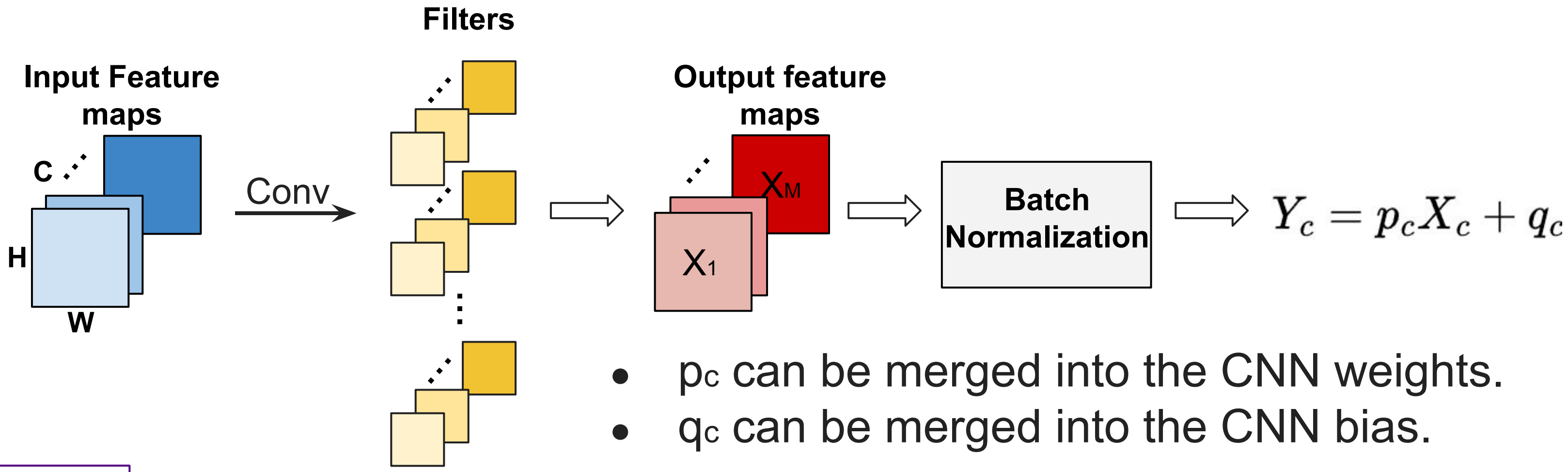
$$\alpha = \{\alpha_c\}, \beta = \{\beta_c\}, \mu = \{\mu_c\}, \sigma = \{\sigma_c\}$$

- 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

Batch Normalization: During Inference

- Given all the parameters are fixed, for each channel c, we have:

$$Y_c = \alpha_c \frac{X_c - \mu_c}{\sigma_c} + \beta_c = \frac{\alpha_c}{\sigma_c} X_c + (\beta_c - \frac{\alpha_c \mu_c}{\sigma_c}) \Longrightarrow Y_c = p_c X_c + q_c$$



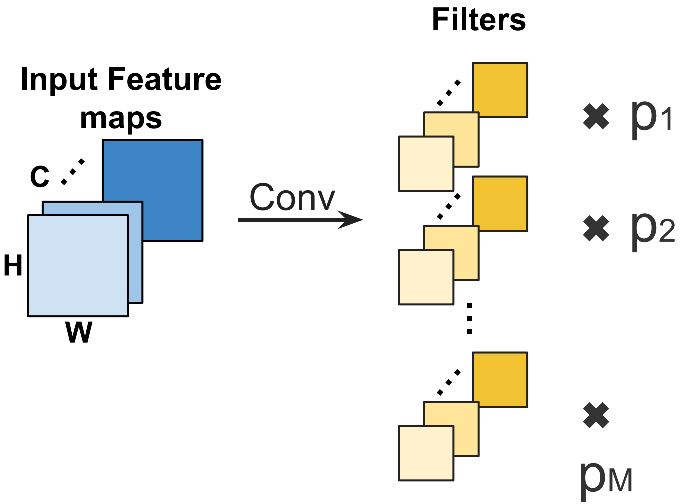
- p_c can be merged into the CNN weights.
- q_c can be merged into the CNN bias.

NYU SAI LAB

Batch Normalization

- For each channel c, we have:

$$Y_c = \alpha_c \frac{X_c - \mu_c}{\sigma_c} + \beta_c = \frac{\alpha_c}{\sigma_c} X_c + (\beta_c - \frac{\alpha_c \mu_c}{\sigma_c}) \Rightarrow Y_c = p_c X_c + q_c$$



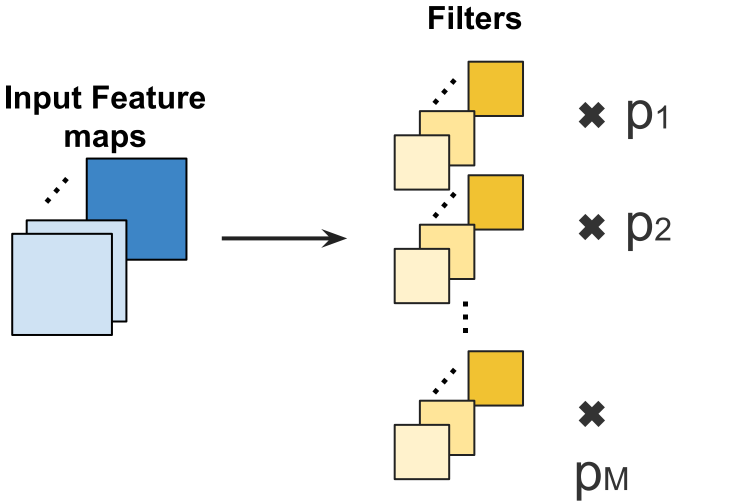
- We can fold in the p and q to the weights and bias of convolutional layer during inference and reduce the online computational cost.

Network Slimming

- Lasso regularization is imposed on the scaling factors of batch normalization during training to automatically identify unimportant weight filters.
- g(.) is the lasso l1-norm $g(.) = \Sigma|p_i|$

$$L = \sum_{(x,y)} l(f(x, W), y) + \lambda \sum_{i} |p_i|$$

- The unimportant weight filters are naturally eliminated during the training process.

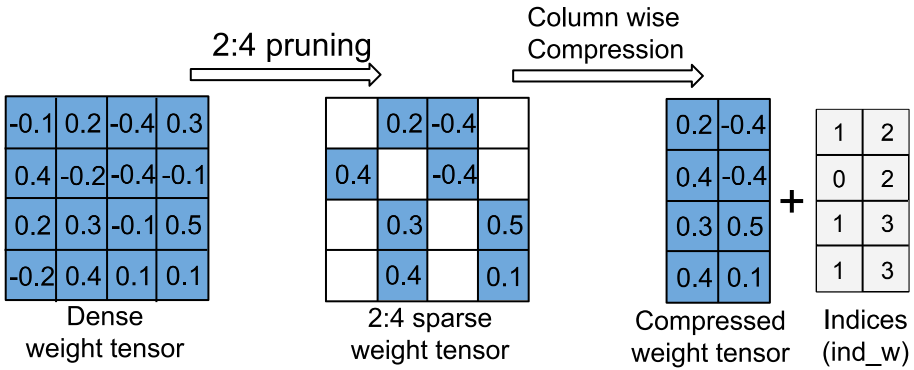


Add a lasso loss on p_i

NYU SAI LAB

Liu, Zhuang, et al. "Learning efficient convolutional networks through network slimming." *Proceedings of the IEEE international conference on computer vision*. 2017.

N:M Sparsity

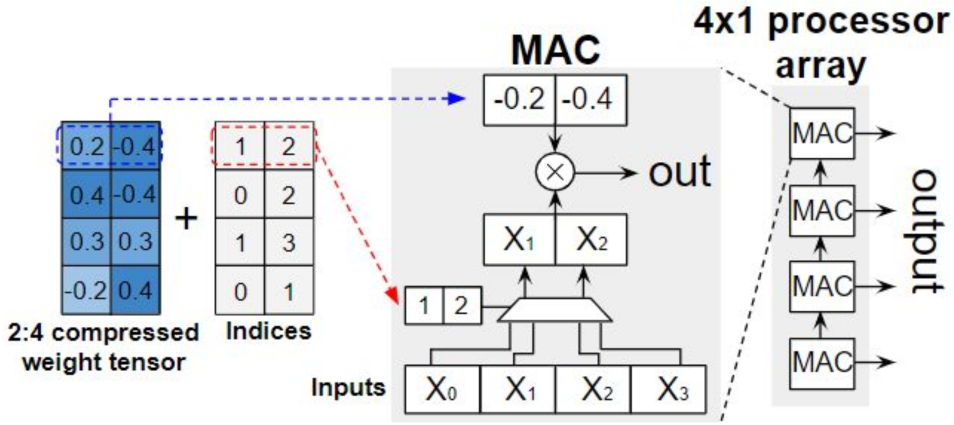


- DNN with structured sparsity can be easily adopted for acceleration, but incur low accuracy.
- On the other hand, DNN with unstructured sparsity is hard to accelerate.

- N:M sparsity is proposed as a middleground between structured and unstructured sparsity.
- 2:4 sparsity is supported in Nvidia V100 GPUs.

Kung, H. T., Bradley McDanel, and Sai Qian Zhang. "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization." *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 2019.

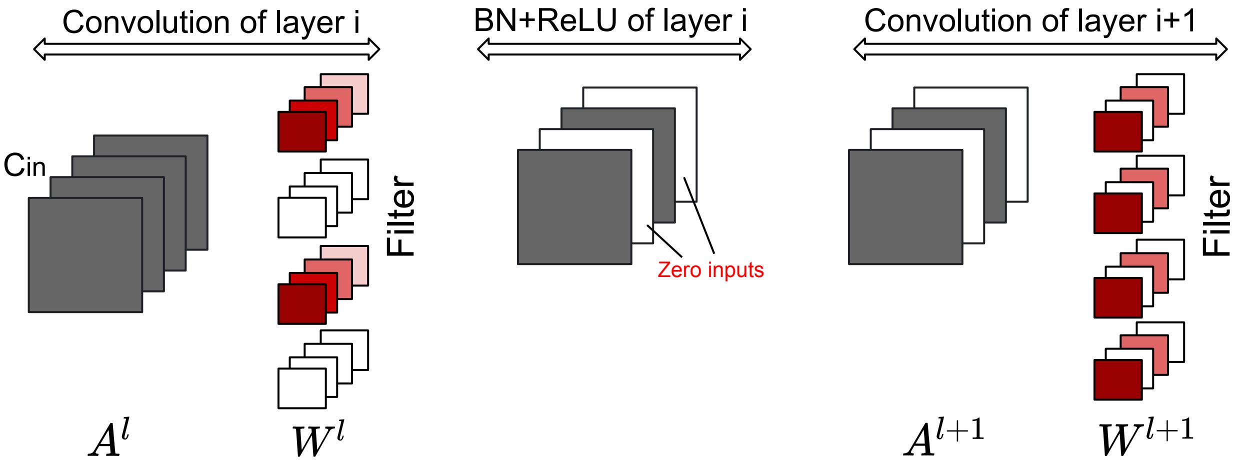
N:M Sparsity



- N:M sparsity is proposed as a middleground between structured and unstructured sparsity.
- 2:4 sparsity is supported in Nvidia V100 GPUs.

Kung, H. T., Bradley McDanel, and Sai Qian Zhang. "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization." *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 2019.

Cascade Effect of Filterwise Pruning in CNN



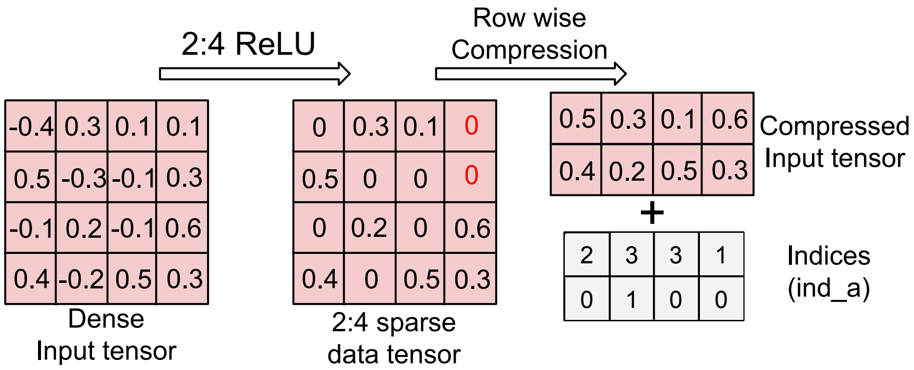
- Assume the bias of the batch normalization is zero.
- Filter pruning at layer I can also result in weight and input sparsity in layer I+1.
- When the bias is not zero, the feature maps of layer i+1 will contain a uniform constant value.

Taxonomy of Pruning

- Pruning techniques can be classified from different perspectives
  - Iterative pruning, zero-shot pruning
  - Structured pruning, unstructured pruning, N:M pruning
  - Weight pruning, activation pruning
  - Static pruning and dynamic pruning
  - Pruning for inference, pruning for training

Pruning on Input Activation

- Why pruning can not be applied to input activation?
    - Large computing cost to determine the importance scores.
    - Due to the usage of ReLU, activation in CNN are 50% sparse, but with irregular sparsity distributions.



NYU SAI LAB

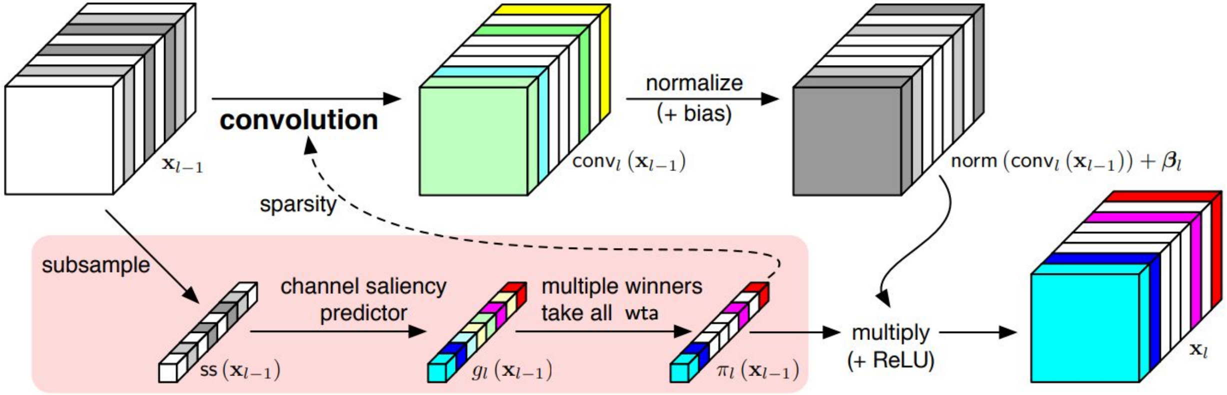
Zhang, Sai Qian, et al. "JointNF: Enhancing DNN Performance through Adaptive N: M Pruning across both Weight and Activation." *Proceedings of the 29th ACM/IEEE International Symposium on Low Power Electronics and Design*. 2024.

Taxonomy of Pruning

- Pruning techniques can be classified from different perspectives
  - Iterative pruning, zero-shot pruning
  - Structured pruning, unstructured pruning, N:M pruning
  - Weight pruning, activation pruning
  - Static pruning and dynamic pruning
  - Pruning for inference, pruning for training

Static vs Dynamic Pruning

- Conventional pruning adopts static pruning criteria and permanently removes components.
- Dynamic pruning exploits input-specific characteristic pruning criteria and preserves the entire network structures and accelerates the networks by dynamically skipping unimportant components.



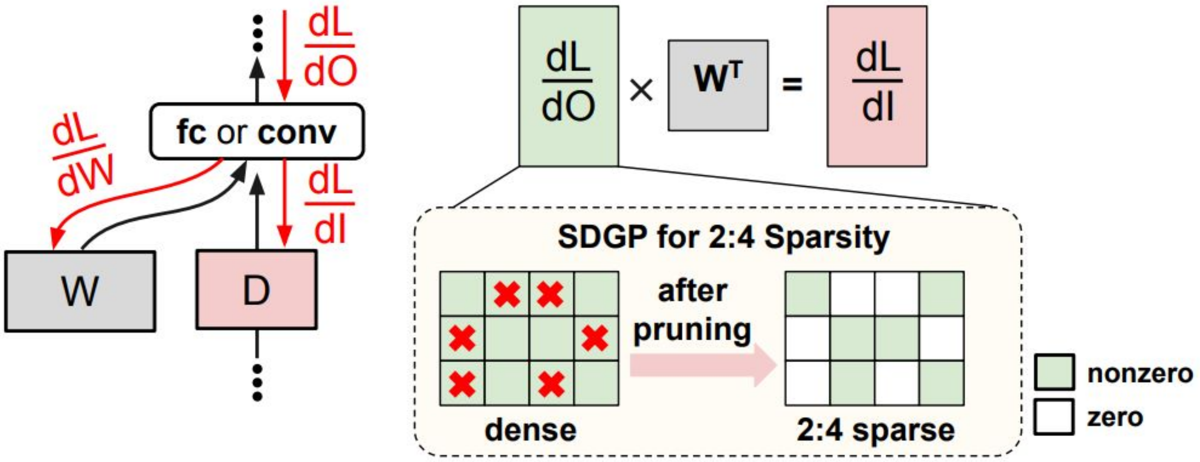
- a channel-wise importance measure is generated.

Gao, Xitong, et al. "Dynamic channel pruning: Feature boosting and suppression." *arXiv preprint arXiv:1810.05331* (2018).

Taxonomy of Pruning

- Pruning techniques can be classified from different perspectives
  - Iterative pruning, zero-shot pruning
  - Structured pruning, unstructured pruning, N:M pruning
  - Weight pruning, activation pruning
  - Static pruning and dynamic pruning
  - Pruning for inference, pruning for training

Pruning during DNN Training



McDanel, Bradley, Helia Dinh, and John Magallanes. "Accelerating dnn training with structured data gradient pruning." *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, 2022.

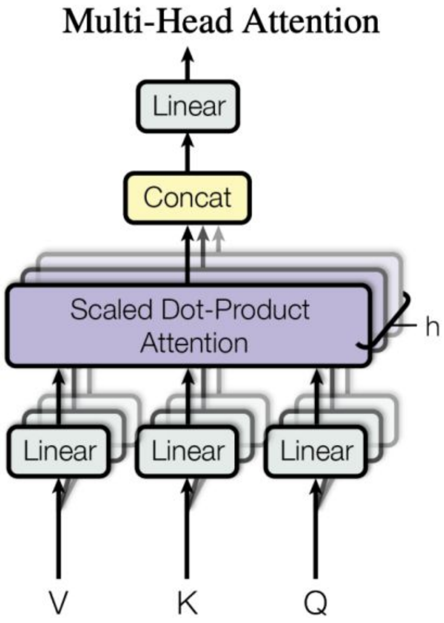
Topics

- Why pruning?
  - Reduce running cost
  - Reduce storage
- General pruning techniques
- Transformer pruning

NYU SAI LAB

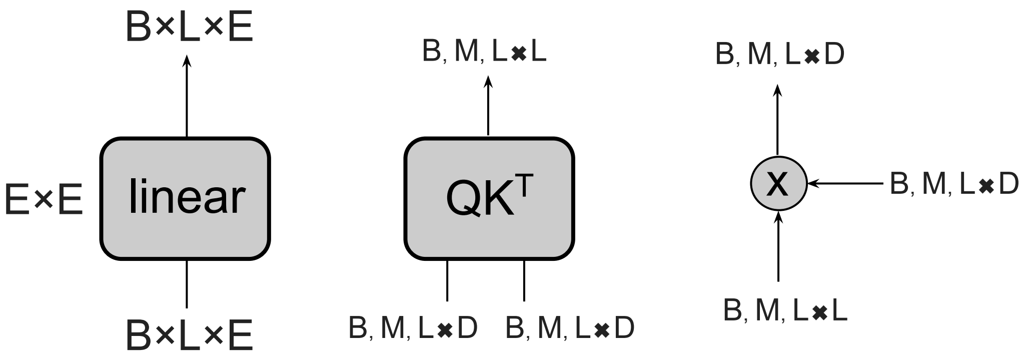
Multi-headed Attention

- Q, K, V tensors are broken into multiple components along the embedding dimension.
  - $(B,L,E) \times (E \times E) \rightarrow (B \times L \times E)$
  - $(B,L,E) \rightarrow (B, M, L, E/M) \rightarrow (B, M, L, D)$, where $D=E/M$
- All the following operations can be performed independently over each head M.
  - $QK^{T} \rightarrow (B, M, L \times D) \times (B, M, D \times L) \rightarrow (B, M, L \times L)$
  - $\mathrm{Softmax}(QK^{T}) \rightarrow (B, M, L \times L)$
  - $\mathrm{Softmax}(QK^{T}) \times V \rightarrow (B, M, L \times L) \times (B, M, L \times D) \rightarrow (B, M, L \times D) \rightarrow (B \times L \times E)$



Michel, Paul, Omer Levy, and Graham Neubig. "Are sixteen heads really better than one?." *Advances in neural information processing systems* 32 (2019).

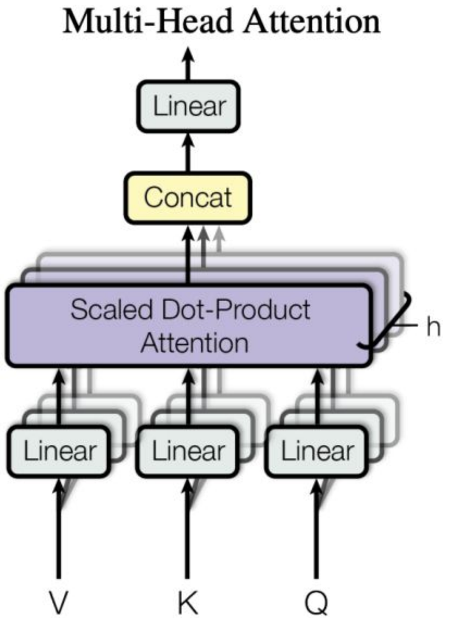
Multi-Head Attention



$B \times L \times E^2$

$B \times M \times L^2 \times D = B \times L^2 \times E$

$B \times M \times L^2 \times D = B \times L^2 \times E$

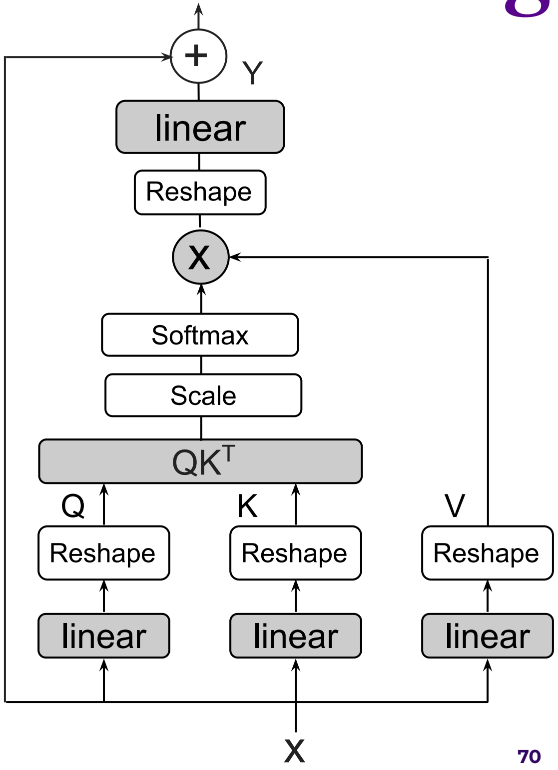


- The introduction of multiple heads do not change the computational cost of the transformer.

NYU SAI LAB

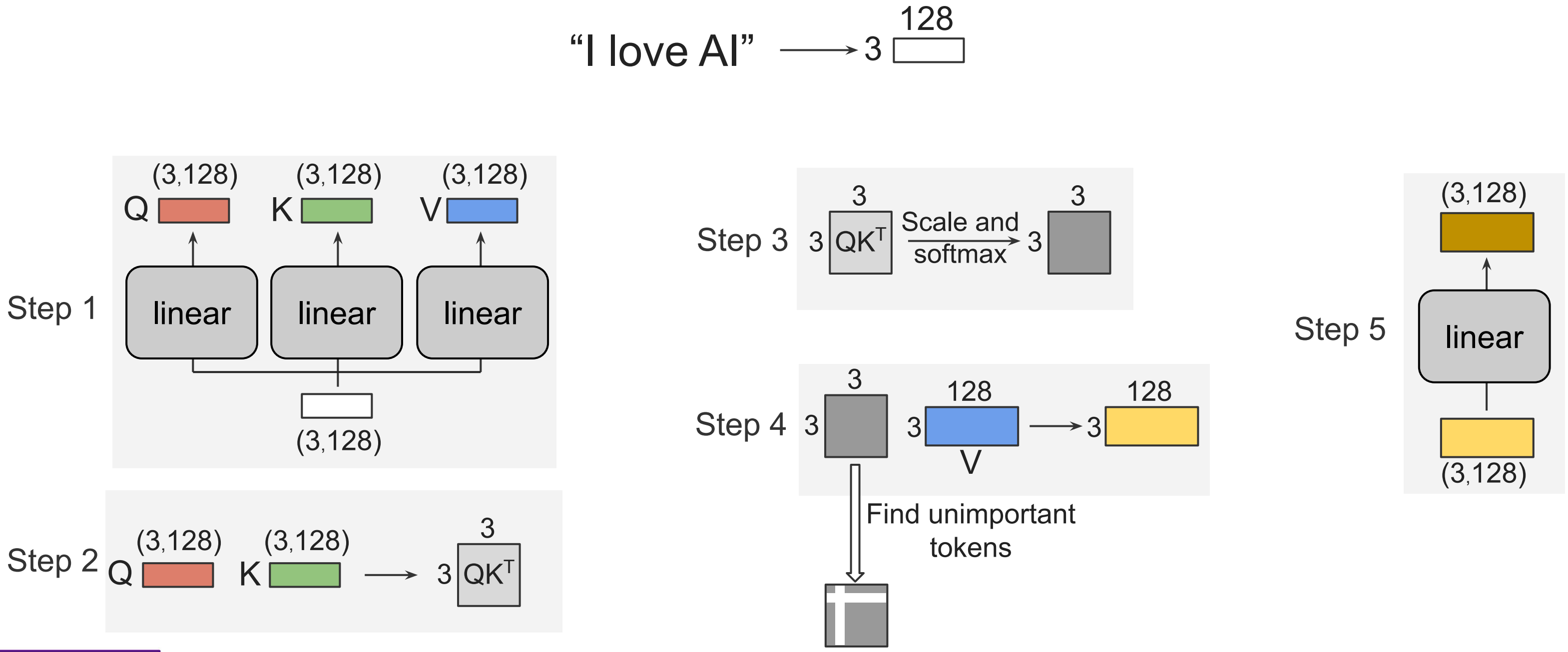
Pruning on Transformers: Token Pruning

- 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) ✖ (E✖E) → (B✖L✖E)
- The second step in calculating self-attention. This will compute the attention score between each pair of input tokens.
  - QK^T→(B, L✖E) ✖ (B,E✖L) → (B, L✖L)
- Scale and normalize the score using softmax.
  - Softmax(QK^T) → (B, L✖L)
- Multiply each value vector by the softmax score.
  - Softmax(QK^T) ✖ V
  - (B, L✖L) ✖ (B, L✖E) → (B, L✖E)
- Pass the result to the linear layer, sum with the input.



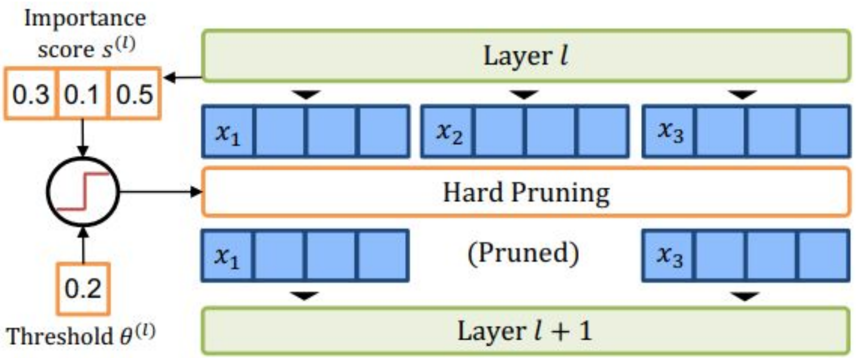
NYU SAI LAB

Pruning on Transformers: Token Pruning



NYU SAI LAB

Pruning on Transformers: Token Pruning



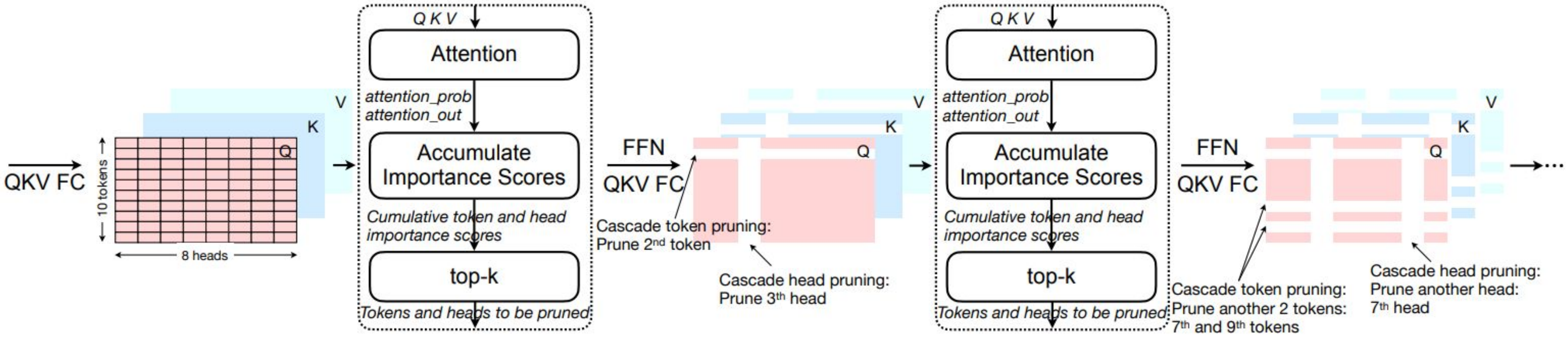
$$s^{(l)}(\mathrm{x}_i) = \frac{1}{N_h}\frac{1}{n}\sum_{h=1}^{N_h}\sum_{j=1}^{n}\mathrm{A}^{(h,l)}(\mathrm{x}_i, \mathrm{x}_j)$$

- One simple approach involves computing the importance score of each token, and remove the tokens whose importance score is lower than a predefined threshold.

NYU SAI LAB

Kim, Sehoon, et al. "Learned token pruning for transformers." *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022.

Pruning on Transformers: Token Pruning



- Tokens and heads can be pruned jointly, the removed tokens and heads will result in a much reduced computation for all the following layers.

Wang, Hanrui, Zhekai Zhang, and Song Han. "Spatten: Efficient sparse attention architecture with cascade token and head pruning." *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2021.

Pruning on Transformers: Token Pruning

- Not all the tokens are necessary to generate the final results.
- Unimportant tokens can be removed progressively as an input sequence passes through transformer layers.

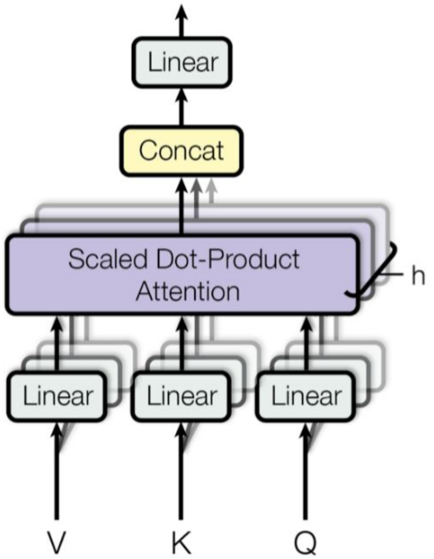
Kim, Sehoon, et al. "Learned token pruning for transformers." *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022.

New Pruning Dimension: Head Pruning

- In additional to the valuewise and channelwise pruning, transformer allows for a new type of pruning: multi-head pruning.

(1, 197, 768) → (1, 12, 197, 64) → (1, 4, 197, 64)

Input — Input with 12 heads — Input with 4 heads

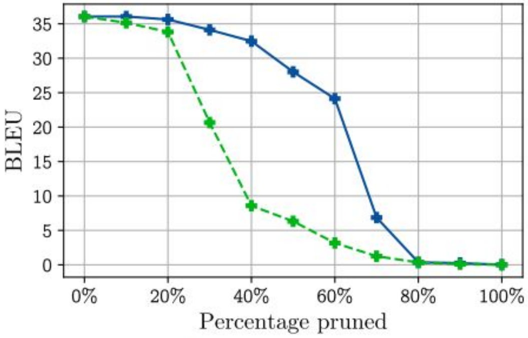


NYU SAI LAB

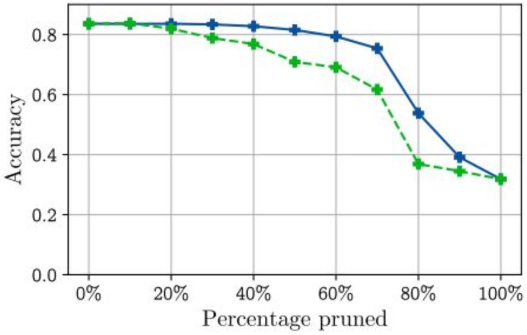
Voita, Elena, et al. "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned." *arXiv preprint arXiv:1905.09418* (2019).

Multi-headed Attention

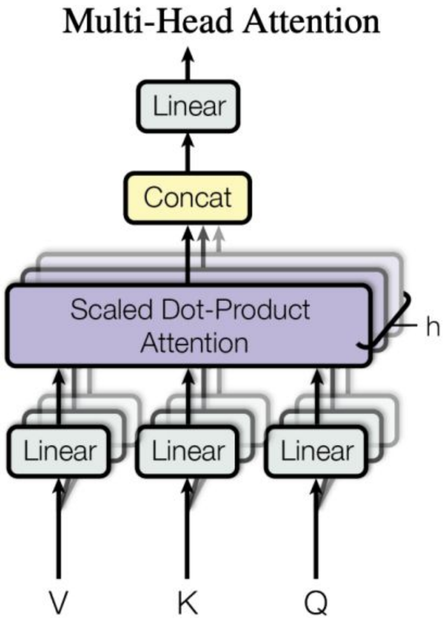
- We observe that the majority of attention heads can be removed without deviating too much from the original score. Surprisingly, in some cases removing an attention head results in an increase in BLEU/accuracy.



(a) Evolution of BLEU score on `newstest2013` when heads are pruned from WMT.

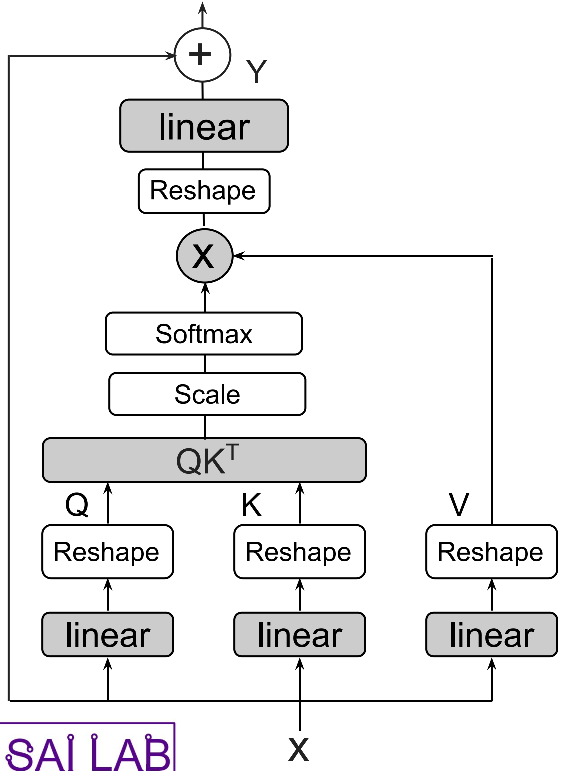


(b) Evolution of accuracy on the MultiNLI-matched validation set when heads are pruned from BERT.



Michel, Paul, Omer Levy, and Graham Neubig. "Are sixteen heads really better than one?" *Advances in neural information processing systems* 32 (2019).

Pruning on Transformers: Head Pruning



- Xi is an embedded vector of ith token.
- There are in total n tokens.
- The output vector of qth token can be expressed as:

$$\text{Att}_{W_k, W_q, W_v, W_o}(\mathbf{x}, q) = W_o \sum_{i=1}^{n} \alpha_i W_v x_i$$

$$\text{where } \alpha_i = \text{softmax}\left(\frac{q^\intercal W_q^\intercal W_k x_i}{\sqrt{d}}\right)$$

- If we further expressed with multi-head attention, the output vector can be expressed as:

$$\text{MHAtt}(\mathbf{x}, q) = \sum_{h=1}^{N_h} \xi_h \text{Att}_{W_k^h, W_q^h, W_v^h, W_o^h}(\mathbf{x}, q)$$

- Where the ε_h are mask variables with values in {0, 1}.

$$I_h = \mathbb{E}_{x \sim X} \left| \frac{\partial \mathcal{L}(x)}{\partial \xi_h} \right|$$

NYU SAI LAB