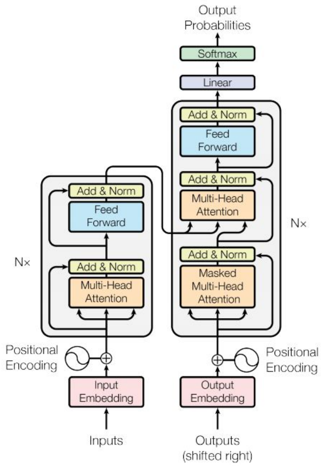
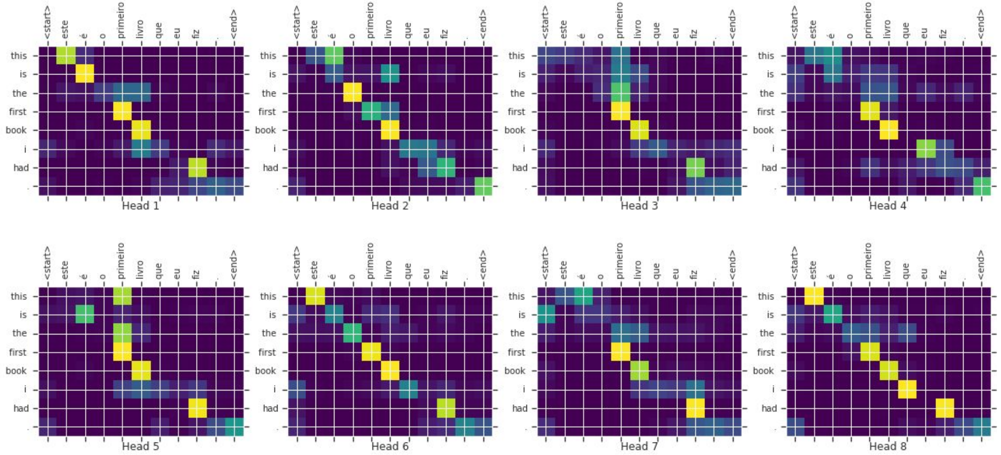
TPrune: Efficient Transformer Pruning for Mobile Devices

Presentation by: Bradford Gill

Transformer Model Significance

- State of the art in NLP
- Accurate machine translation
- GTP-3/BERT/ext achieving near human language abilities, including text generation
- DNA/RNA sequence analysis
- ViT out preforming CNNs in computer vision





Images: https://www.tensorflow.org/text/tutorials/transformer

Clarifications from reviews

- BLEU (bilingual evaluation understudy) score is a measure of distance between machine translation and professional human translation. Higher corresponds to better.
- BSSL was not used to prune, but used to analyze the network.

Motivation

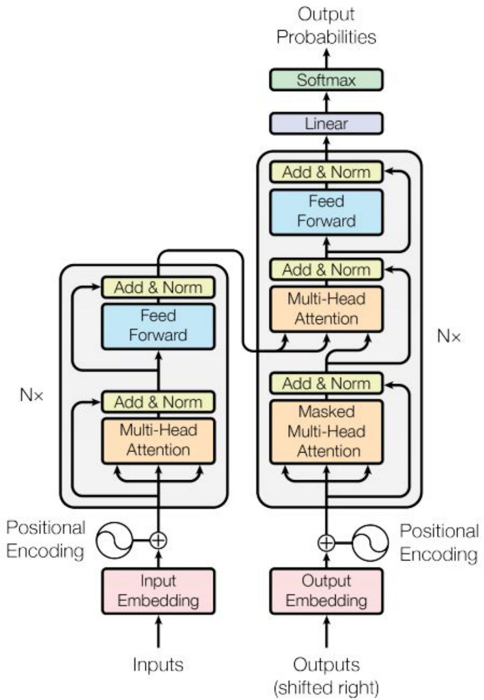
- Translation is vital to enabling human to human activities
- Transformers have achieved SOTA performance in neural machine translation (NMT)
- Transformers are memory intensive, not good for mobile execution
- Cloud computing not always feasible
- Model compression:
    - Transfer learning: (not traditional transfer learning) transferring knowledge from a large model to a small model
    - Efficient transformer alternatives: substituting costly aspects of the transformer with more efficient computations
    - Model Pruning: removing weights or groups of weights from the model. Straightforward compared to other techniques, additionally effective.

Proposed Method/Contributions

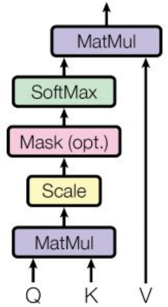
- Structure pruning techniques that exploit redundancy
- Use architecture aware techniques to prune transformer
- Analyze aspects of transformer models using BSSL
- Force sparsity using regularization (SSL and SHS)
- Compared TPrune with other SOTA transformer model techniques

Transformer Model architecture

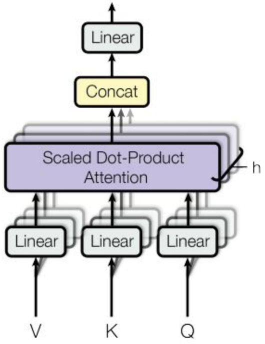
- Encoder/Decoder
- MHA (multi-head attention)
- FFN (feed-forward network)
- Word embeddings vectorize the “meaning” of the word



Scaled Dot-Product Attention

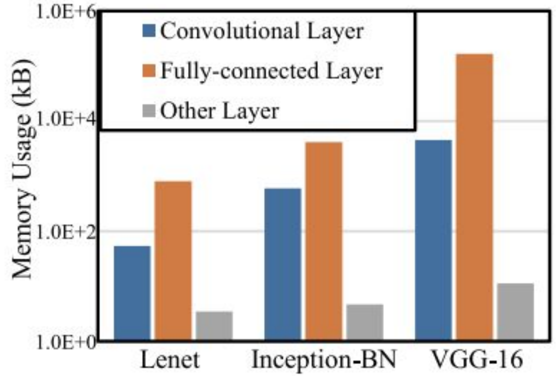


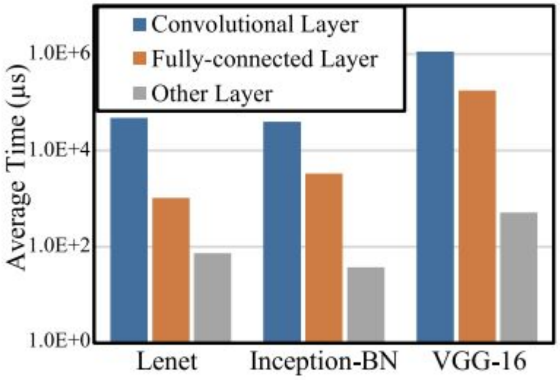
Multi-Head Attention



Images: https://www.tensorflow.org/text/tutorials/transformer

Model Profiling on mobile devices





- Fully connected (FC) layers are less computationally expensive but more memory efficient compared with convolutional layers
- MHA and FFN are reliant on FC layers
- Mobile devices have low memory capabilities

Transformer pruning techniques

- Layer-wise pruning
  - Large granularity
  - Entire layers are dropped
- Head-wise pruning
  - Medium granularity
  - Removes heads from MHA
  - Low BLEU score degradation
- Line-wise pruning
  - Small granularity
  - Removes rows or columns from weight matrices
- Element-wise pruning
  - Prunes on an individual weight bases
  - Comparable or increased BLEU score
  - Speedup not always achievable

- Generally:
  - Larger granularity == larger performance degradation
  - Finer granularity == less parallelism, hence smaller speed-ups
- Fine-tuning allows the “recalibration” of models after pruning to recover performance

Model pruning with regularization

- Regularization can force weights towards zero
- L_0
  - represents the amount of non-zero weights
  - Not differentiable
- L_1
  - Pushes parameters towards 0
  - Differentiable
- Structure pruning
  - Removes groups of weights together
  - Preserves memory locality benefits

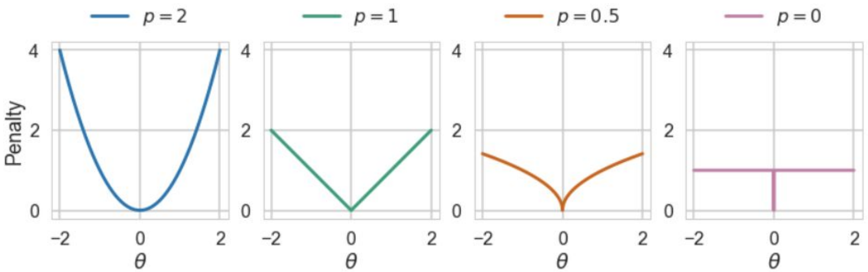


Image: https://openreview.net/pdf?id=H1Y8hhg0b

Structured Hoyer Square regularization

- Abbr: SHS
- L_0 regularization can be used to punish none 0 weights but is non-differentiable so is a challenge with gradient based approaches
- Scale invariant R(aX) = R(X)
- The square of the sum of L_2 norms of each weight group over the L_2 norm of the entire weight matrix
- Force entire groups of weights towards 0, rather than individual weights

$$SHS(W) = \frac{(\sum_{g=1}^{G} ||w^{(g)}||_2)^2}{\sum_{g=1}^{G} ||w^{(g)}||_2^2},$$

$$SHS(W) = \frac{(\sum_{g=1}^{G} ||w^{(g)}||_2)^2}{||W||_2^2}.$$

Analysis and pruning targets

- Matrices W^Q, W^V, W^K, W^O, W_{ffn1}, W_{fnn2} are pruning targets
- Should targets be pruned column-wise or width-wise?
- Should the encoder and decoder be treated equally?
- Are there sparsity differences between deep and shallow layers inn the model?
- Should W^Q, W^V, W^K be pruned with the same sparisty in MHA?

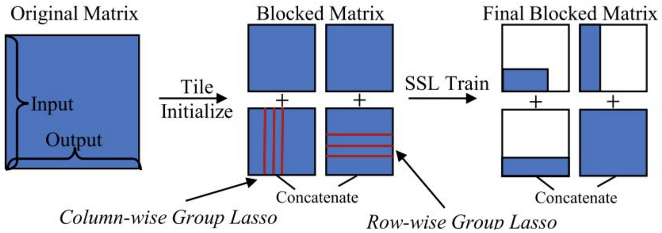
Block-wise Structured Sparsity Learning (BSSL)

- Original layer weight matrix divided into many blocks of equal shape
- Row and column wise penalties applied to each block
- MHA W^Q, W^V, W^K blocks have dims equal to head width allowing us to examine
    - How many heads are needed
    - Needed dimension of heads
- W^o in MHA layer broken up into 2x2 sub blocks
- W_{ffn1} and W_{fnn2} broken into 2x8 and 8x2 sub blocks respectively

$$L_{row}(W) = \sum_{i=1}^{r}\sqrt{\sum_{j=1}^{c}(W[i,j])^2}$$

$$L_{col}(W) = \sum_{i=1}^{c}\sqrt{\sum_{j=1}^{r}(W[j,i])^2}$$

$$L = L_D + \lambda \sum_{i=1}^{l}\sum_{j=1}^{x}\sum_{k=1}^{y}(L_{row}(W_{[i,j,k]}) + L_{col}(W_{[i,j,k]}))$$



Observations of BSSL

Red: No sparsity
Green: Non-zero weights
White: Zero weights

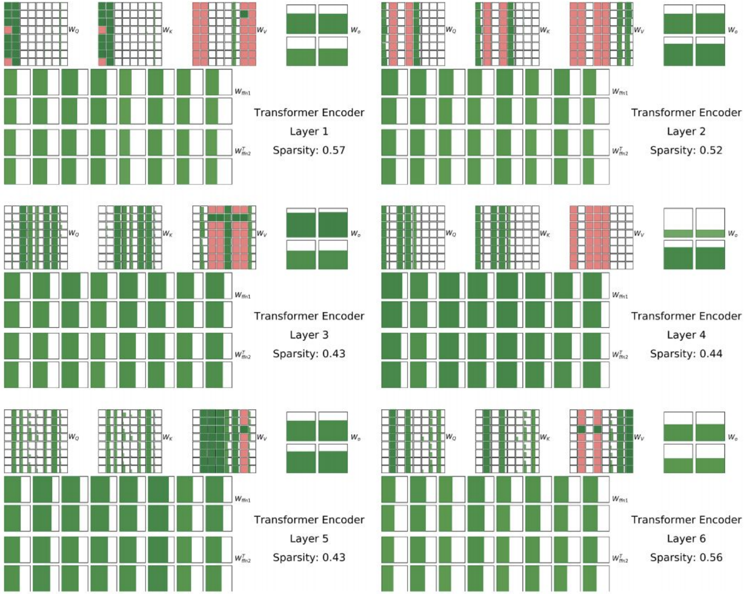


Fig. 5. Sparsity visualization of Transformer encoder with BSSL.

Fig. 6. Sparsity visualization of Transformer decoder with BSSL.

Conclusion of BSSL

- All weight matrices are pruned to certain extent in encoder
- W^Q, W^V, W^K, W_{fnn1} pruned col-wise (to preserve embedding dimension)
- W^O, W_{ffn2} pruned row-wise (to preserve embedding dimension)
- Encoder has higher sparsity then decoder
- Encoder self attention has higher sparsity compared to encoder-decoder attention
- Deeper decoder self attention heads have higher sparsity
- Deeper decoder-encoder attention heads have lower sparsity
- Embedded dimensions are preserved

Transformer pruning/Pruning strategy

- SSL pruning
  - L_1, L_2 norm used as regularization
  - Scale variant, meaning that the gradient is proportional to the magnitude of an individual weight. Can slow trend towards 0
- SHS pruning
  - Structured hoyer square used as regularization
  - Scale invariant, can approach 0 faster.
- Pruning strategy
  - In encoder: W^Q, W^K, W^V, W_{ffn1} and W^O, W_{ffn2} pruned col and row-wise respectively
  - In decoder: only W^Q, W^K are pruned col wise
  - Fine-tuning is done after pruning to recover lost performance

Experimental setup

- TensorFlow tensor2tensor (T2T) research model library used
- TFlite used as mobile framework
- WMT English to German database used
- Original transformer model used
- 4 NVIDIA GTX TITAN X’s used for training
- Nexus 5, Pixel 2, Pixel 3, and LG G8 ThinQ CPUs used for testing

Comparisons of Different Pruning Regularizers

- SHS out preforms SSL in low sparsity high performance (fig 7)
- SHS not monotonically decreasing (fig 8)
- SSL preforms better with high sparsity

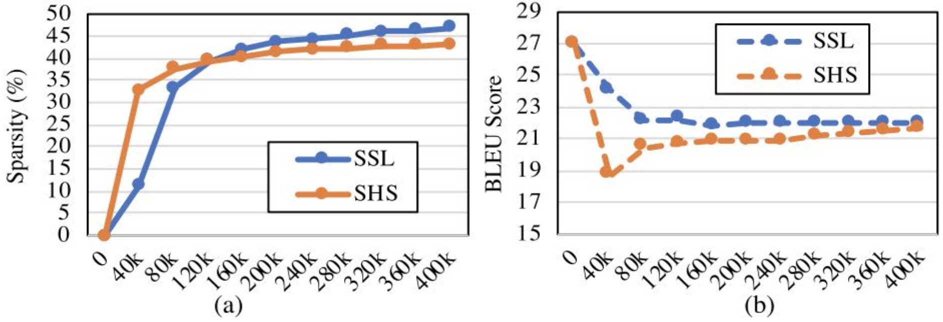


Fig. 8. Sparsity (a) and BLEU (b) between SSL ($\lambda = 5 \times 10^{-5}$) and SHS ($\lambda = 10^{-3}$) with a large λ.

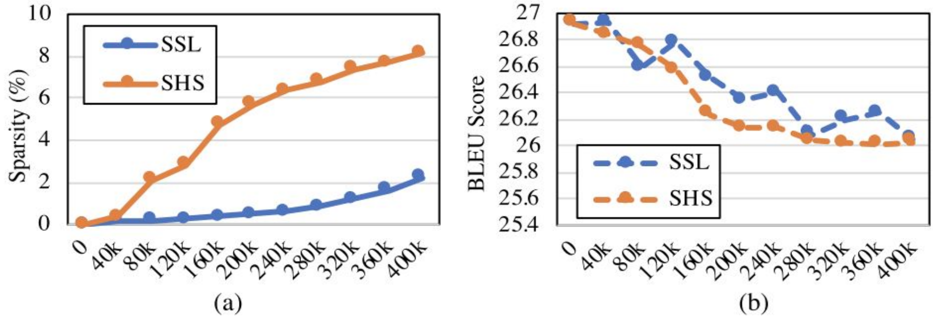
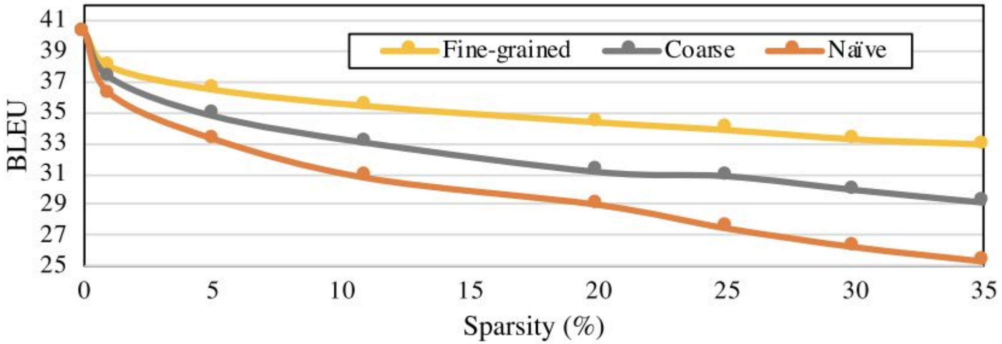


Fig. 7. Sparsity (a) and BLEU (b) between SSL ($\lambda = 10^{-5}$) and SHS ($\lambda = 10^{-4}$) with a small λ.

Evaluation of Pruning Strategies for SHS

- Fine grained
  - TPrune strategy stated above
- Naive
  - Apply row and column penalties to all target matrices
- Course Grained
  - Treat encoder and decoder the same



Evaluation of Layer-wise Sparsity

- W^Q,W^K are always of the same col-wise sparsity
- The col-wise of W^V and row-wise of W^O are the same.
- W_{ffn1} col-wise and W_{ffn2} row-wise sparsity are equal
- Lower sparsities correspond to more important matrices

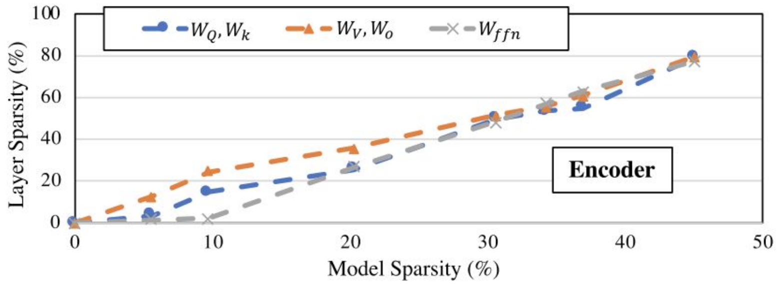


Fig. 10. Layer-wise sparsities of Transformer encoder under different model sparsities.

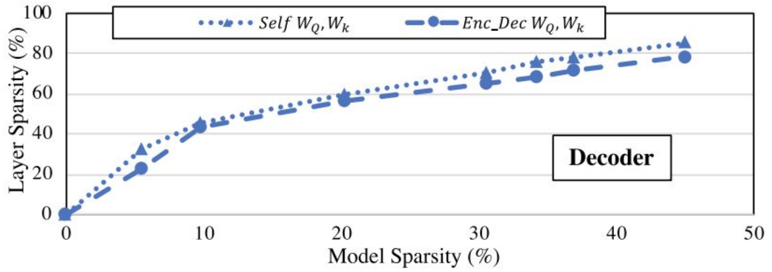


Fig. 11. Layer-wise sparsities of Transformer decoder under different model sparsities.

Evaluation of Speedup on Mobile Devices

- Memory overhead prevents O(n^2) strlen time
- Memory overhead is largest bottleneck

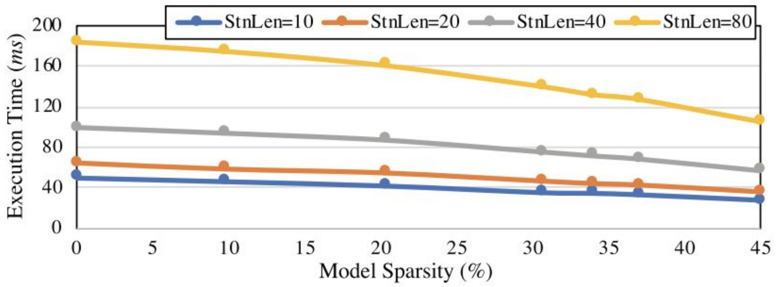


Fig. 12. Execution time with different string lengths.

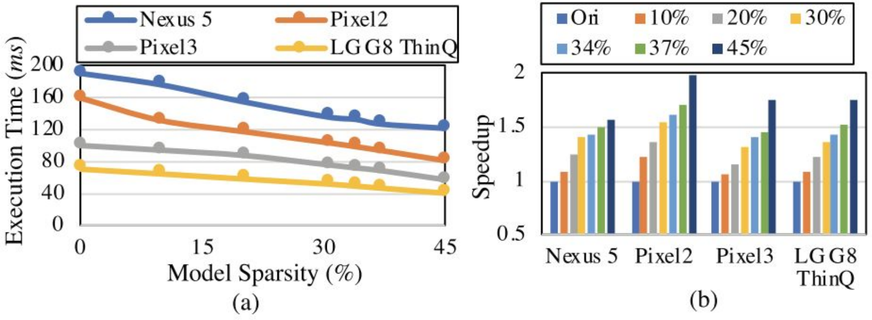
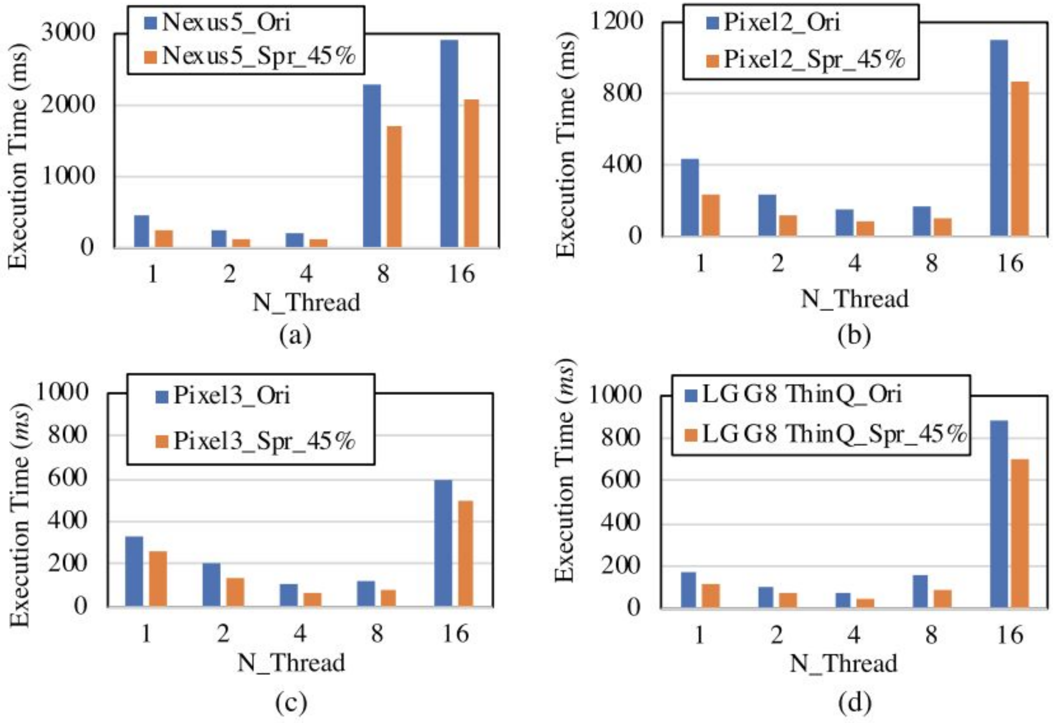


Fig. 13. Execution time (a) and Speedups (b) on different mobile devices.

Evaluation of Speedup on Mobile Devices

Threading can help to an extent, regardless, sparsity always helps



Evaluation of Sparsity-accuracy Trade off

- Insignificant degradation below ~20% sparsity, corresponds to ~1.25x speed up
- Out preforms head wise pruning

Table 2. Summary of Our Line-wise Pruning Results and Previous Head-wise Pruning Results [21]

Dataset	Model	BLEU	BLEU Degradation (%)	Sparsity(%)	Speedup
WMT_EnDe	Baseline	26.92	0	0	1
	Model1	27.14	0	9.76	1.16
	Model2	26.93	0	15.63	1.21
	Model3	26.78	0.52	20.29	1.25
	Model4	26.14	2.9	30.65	1.44
	Model5	25.94	3.58	34.27	1.52
	Model6	25.63	4.79	36.93	1.59
	Model7	24.78	7.95	45.07	1.92
WMT_EnDe	Baseline	26.92	0	0	1
	[21]	26.92	0	4.29	1.05
	[21]	25.19	6.43	8.58	1.15
	[21]	20.74	22.96	17.14	1.33
	[21]	10.1	62.82	25.71	1.56

Comparison with head wise pruning

- Head based pruning can achieve higher speed up on lower sparsity. But from before, the speed-up accuracy trade off gives TPrune the edge

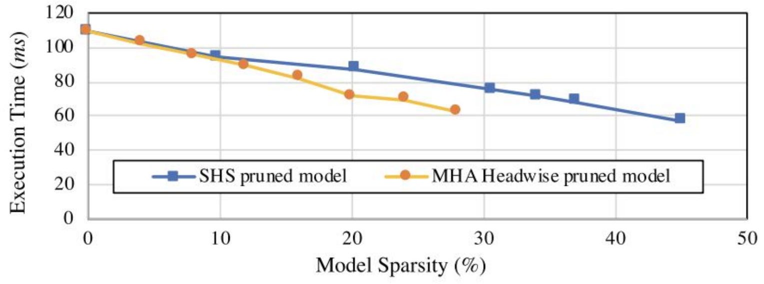


Fig. 15. Execution time when executing pruned models of different sparsity.

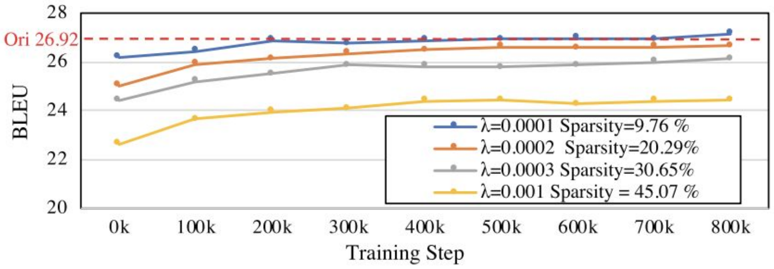


Fig. 16. BLEU score during the fine-tuning procedure.

Questions?