When to Use Efficient Self-Attention?
Profiling Text, Speech and Image Transformer Variants
Anuj Diwan, Eunsol Choi, David Harwath
University of Texas at Austin

Introduction
Many Transformer variants designed to improve the efficiency of self-attention have been proposed in the past several years. We study the efficiency of some of these variants across text, speech and vision, seeking answers to two questions:
1. Is self-attention the true bottleneck, and for what modalities? We visualize layerwise efficiency of models.
2. For what use-cases are these variants useful (or not)? We profile different efficiency metrics for a range of input-lengths.

Efficiency Metrics
Efficiency: umbrella term for a suite of metrics. We profile 4 such metrics:
1. Throughput: Number of examples, with a given sequence length, processed per second, with the max batch size possible for a given GPU
2. Latency: Time (in ms) to process 1 example of a given sequence length
3. # Parameters: Number of model parameters in both train and infer modes. We also profile layerwise latency and # Parameters (separately for Self-Attention, Feedforward, Embedding, etc.).
4. Memory: Allocated GPU memory (in MiB) to process 1 example

Local HuBERT Model
We introduce Local HuBERT, a variant of HuBERT that uses Longformer local-window attention.

Evaluation
We initialize L-HuBERT with pretrained HuBERT weights and evaluate on Librispeech ASR under Frozen (train projection) and Finetune (train all) settings, exploring 32 & 100 token contexts.

<table>
<thead>
<tr>
<th>Model</th>
<th>WER (Frozen)</th>
<th>WER (Finetune)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HuBERT Base</td>
<td>7.09</td>
<td>3.40</td>
</tr>
<tr>
<td>L-HuBERT (32</td>
<td>100)</td>
<td>21.06</td>
</tr>
</tbody>
</table>

Despite a performance gap, L-HuBERT shows reasonable performance and hence we study its computational efficiency.

Implementational Details
Time-based metrics use Pytorch CUDA Events, Max-Memory uses torch.cuda.max_memory_allocated(), # Parameters uses torchinfo, and layerwise metrics use module-level profiling hooks using torchprof.

Evaluation Methodology
Models: Text: BERT, Longformer, Nystromformer (Huggingface); Speech: HuBERT, L-HuBERT (fairseq); Vision: ViT, Swin (Huggingface).

Sequence Length Ranges: Text: 62 to 3362 tokens in steps of 60; Speech: 50-2500 tokens in steps of 25; Vision: 32-1024 pixels in steps of 32.

Layerwise Profiling: Results
1. Non-self-attention components are expensive: Below the avg. seq length of most datasets (1000 tokens for text/speech, 512 pixels for vision), other components take up 35% (text), 58.8% (speech) and 43.75% (image) latency.
2. Optimal strategies can differ across modalities: Embeddings are expensive for Speech but not for others.
3. For variants, attention has large overheads: (see paper!)

Overall Profiling: Results
1. Tipping-Point Analysis: The point at which variants become more efficient that their vanilla counterparts.
   a) High (1.75-2k tokens) for most text/speech datasets.
   b) Reasonable (500-700 px) for high-res image datasets.
   c) Non-existent for the throughput metric.
2. The right model depends on resources: Efficient models are not great for fast training (throughput) but they are pretty good for low-memory inference (max-memory).
3. Possible Reasons: Efficient models suffer from additional overheads (reshaping, preprocessing); plus, local-attention models excessively pad their inputs!

Conclusion
1. Our efficiency analysis reveals differences across modalities and metrics and provides guidance for when a given model should be chosen.
2. Layerwise analysis finds that self-attention is not the only bottleneck, and that the extent of its efficiency cost differs by modality.

We recommend that efficiency papers should include cross-modal & layerwise profiling results to provide a full picture of model benefits.