Optimizing Attention Mechanisms in Transformers

Week 12: Project Overview and Results

Chandler Cheung, Charis Gao, Jordan Hochman

Background

- Transformer models have become central to NLP tasks
- In recent years, size of models has grown exponentially
- Key challenge: $O(n^2)$ complexity in attention mechanism
- Growing model sizes create memory constraints
- Need more efficient attention mechanism without degrading performance

Optimization Problem

- Develop customizable attention mask that learns important tokens in sequence to attend to instead of attending to all tokens
- Train model with optimized attention mask to produce outputs similar to a baseline, unmodified transformer
- Preserve model quality while reducing computational cost

Mathematical Formulation

Core objective: minimize KL-divergence between baseline and custom model over all training examples X

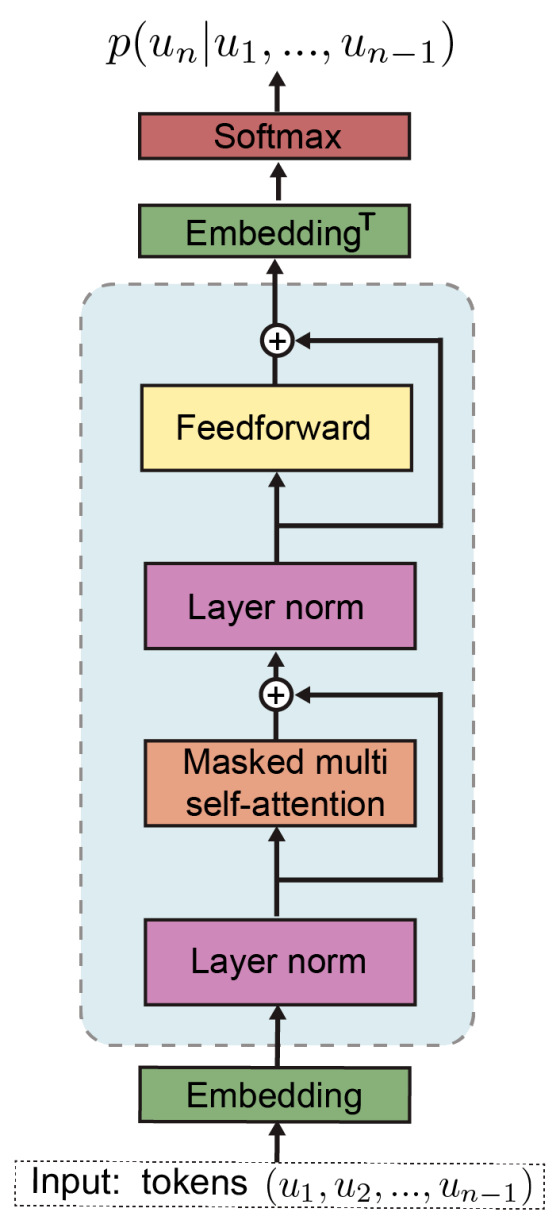
$$\mathcal{L} = \mathrm{KL}\big(P_{\text{base}} \,\|\, P_{\text{custom}}\big)$$

Metrics

- Accuracy retention: comparable performance
- Computational improvement (sub-quadratic): reduced memory and/or speed gains
- Distribution alignment: low KL-divergence

Implementation

- Baseline model: GPT-2 (unoptimized attention mechanism)
- Dataset: WikiText-2
- Custom attention module versions:
  1. Naive linear combination of candidate masks with learnable weight parameters and L1 penalty
  2. Performer -- kernel approximation of attention mechanism with random feature maps
  3. Native Sparse Attention -- hierarchical attention mechanism with a sliding window and selective attention



GPT-2

- Transformer-based language model built using stacked decoder blocks focused on self-attention
- Causal masking – each word in a sequence attends to all previous words using scaled dot-product attention

Loss Computation

- Compute logits from both models on the same input batch
- Calculate KL-divergence and minimize

```
def kl_divergence_loss(logits_custom, logits_ref, mask):
    log_probs_custom = F.log_softmax(logits_custom, dim=-1)
    probs_ref = F.softmax(logits_ref.detach(), dim=-1)  # Detach reference model

    # Calculate per-token KL
    kl = (probs_ref * (probs_ref.log() - log_probs_custom)).sum(-1)

    # Apply padding mask and average
    active_tokens = mask.sum()
    return (kl * mask).sum() / active_tokens
```

Naive Linear Combination of Candidate Masks

- Simple weighted linear combinations of 3-5 fixed candidate masks (eg. past 5 tokens, past 10 tokens, one-hot encoding of tokens, etc.)
- Coefficients are tunable parameters
- L1 penalty so coefficients are not extremely large and so that we can interpret which attention masks are significant

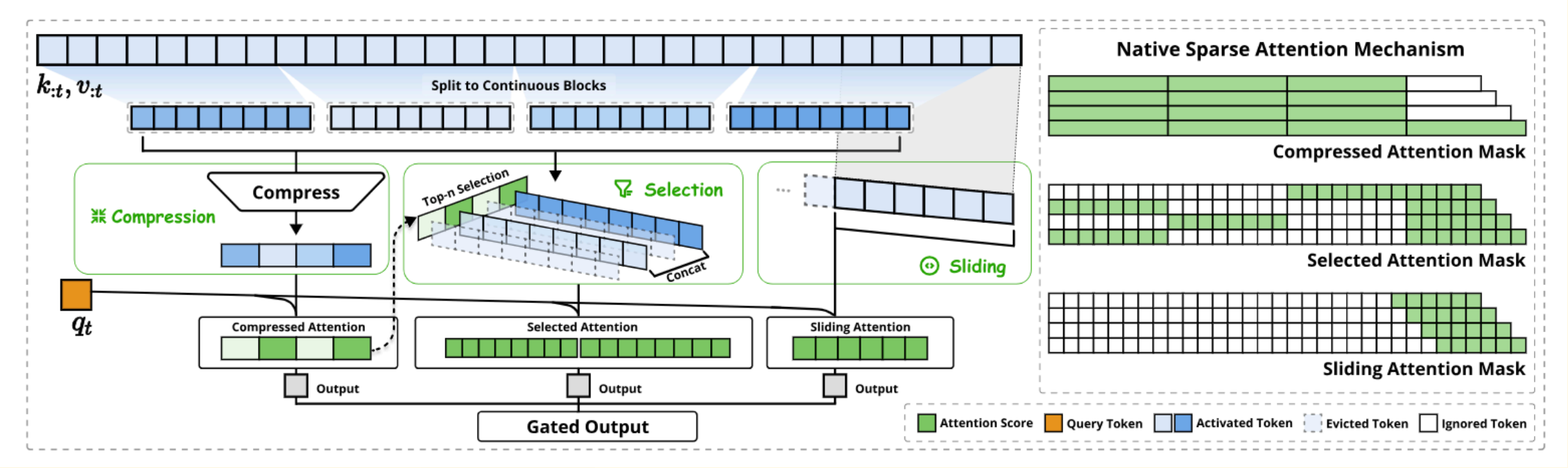
Performers

- Use kernel-based approximations to replace softmax attention– uses FAVOR+ to map Q and K to a different space using random projections (random feature maps)
- Reduces complexity of attention mechanism to linear time

Native Sparse Attention

- Hardware-optimized and end-to-end trainable sparse attention mechanism
- Reduces computational overhead using hierarchical approach
  - Compressed representations of tokens for global context
  - Selectively retains the most relevant tokens for local precision
  - Employs a sliding window mechanism to maintain continuity in processing sequential tokens

Native Sparse Attention - Diagram



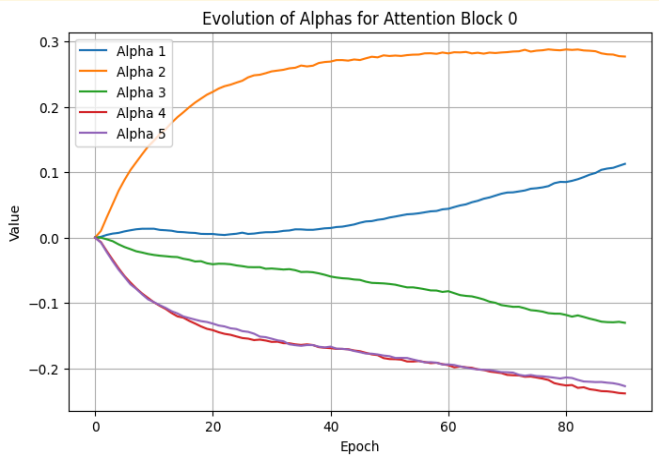
Current Results - Naive Linear Combination

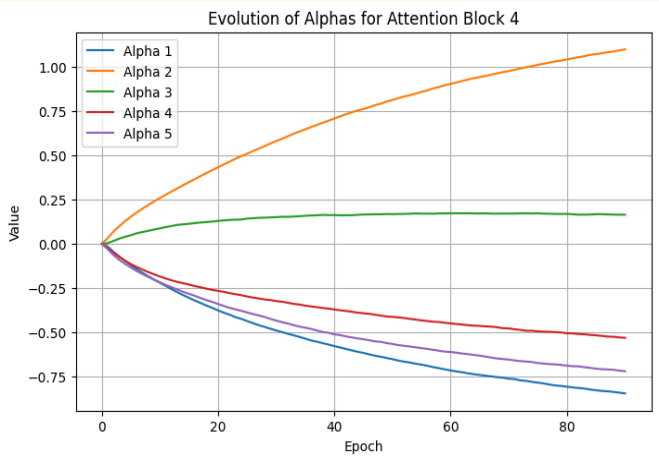
- Over 100 epochs, loss decreased from 2.1470 to 0.3881
  - Custom attention can mimic the reference model's distributions
- Tested with a few prompts, resulting in output text mimicking style similar to GPT-2 (follows prompts + produces recognizable words), though often drifts/is less coherent due to the limited context

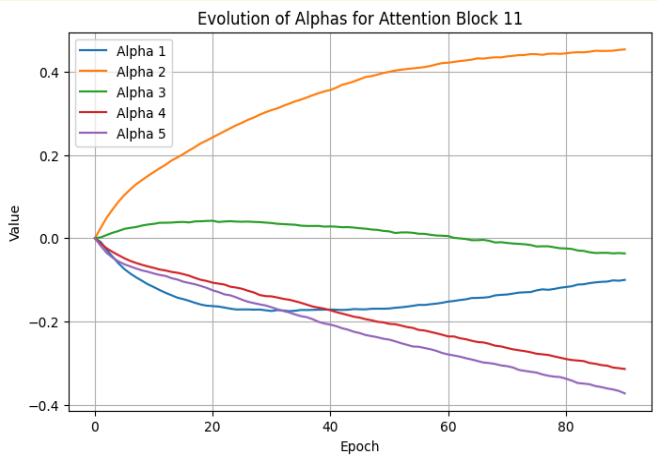
```
Prompt: Hello, my name is
Reference: Aaron. It took just weeks of work to get this script ...
Custom: in German; "Wulf," which means a new kind of word ...

Prompt: The meaning of life is
Reference: different when it comes to death. It involves the beginning and end ...
Custom: not a question, however many people are involved in this matter ...
```

Current Results - Naive Linear Combination Attention Masks Coefficients Convergence







Graphs of evolution of attention mask coefficients during training. Each line represents a coefficient in the attention block. The convergence of these values suggests the model is learning stable attention patterns.

Current Results - Performer

- Work in progress - generates coherent English text
- Trained for 50 epochs, loss decreased from 3.1287 to 2.2994
- Issues with NaN and division by zero addressed by standardization

```
Prompt: The meaning of life is
Reference: that God has created you to live according as he desires. (Deut 4:15.) ...
Custom: also the use to be, where you can find out. The reason for an individual human beings and one who are a major problems with no other than when we're already in some people ...

Prompt: As the sun set behind the towering mountains, the weary traveler finally caught sight of the distant village, its warm lights flickering like tiny stars
Reference: He was alone but in darkness for a moment before he heard his brother's cries and saw him pass by it ...
Custom: around a temple. The second floor which is still standing right and that has an ancient Egyptian tomb ...
```

Current Results - NSA

- Work in progress
- Trained for 5 epochs, loss decreased from 331.665 to 175.68
- Generated outputs becomes incoherent after a few tokens

```
Prompt: Artificial intelligence
Reference: is the key to the future, but it may also be the key to the future with its ability to detect, investigate and manage complex patterns of action. Some scientists, for example, have proposed that artificial intelligence could be  ...
Custom: , will not pay those that they were the most common people could be understood with other other other two-tetetetetetetetetetetetetetetetetetetetetetetetetetetetetetetetetetete  ...
```

Current Limitations and Direct Next Steps

- Standardize all implementations to use same loss function, optimizer, step size, scheduler, etc.
- Modify implementation of Performer so that it doesn't reach noise floor
- train NSA on larger dataset for more epochs and experiment with context length so that coherent English words are produced --> currently accuracy concerns / limited coherence as well as high loss

Further Steps

- Measure memory usage
- Optimize hyperparameters
- Experiment with regularization, penalty, and/or constraints
- Test with different training datasets

Thank you!

Any questions?