Optimizing Attention Mechanisms in Transformers

Lightning Talk

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

$$\mathcal{L} = \mathrm{KL}\left(P_{\text{base}} \parallel P_{\text{custom}}\right)$$

Metrics

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

Current Implementation

- Baseline model: GPT-2 (unoptimized attention mechanism)
- Custom attention module: linear combination of candidate masks
  - Learnable weight parameters w/ L1 penalty (independent for each transformer block/layer)
- Dataset: WikiText-2

```
def forward(self, hidden_states, attention_mask=None, **kwargs):
    ...
    candidate_masks = self._get_candidate_masks(seq_length, device=device)
    ...
    w = torch.sigmoid(self.alpha)
    ...
    final_mask = torch.sum(w * candidate_masks, dim=2)
    ...
```

Current Implementation

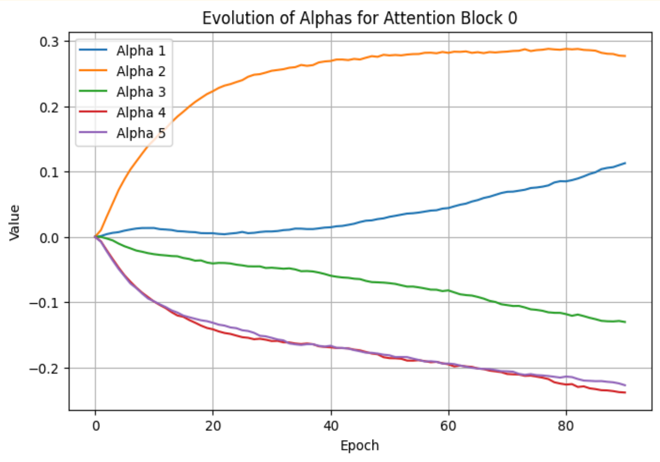
- Loss Computation
  - Compute logits from both models on the same input batch
  - Calculate KL-divergence and minimize

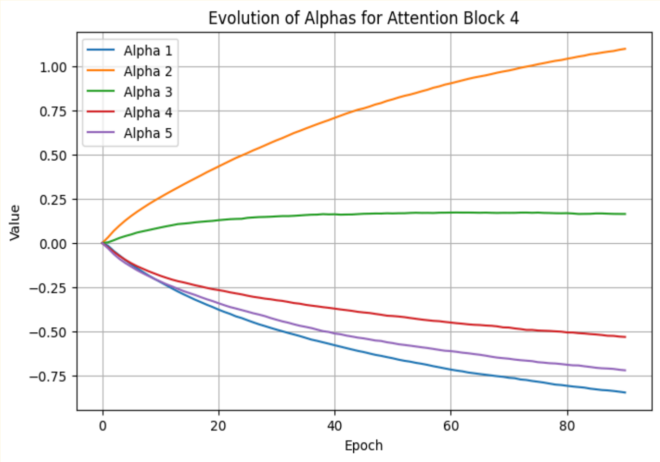
```
def kl_divergence_loss(logits_custom, logits_ref, mask):
    log_probs_custom = F.log_softmax(logits_custom, dim=-1)
    probs_ref = F.softmax(logits_ref.detach(), dim=-1)  # Detach reference model

    # 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
```

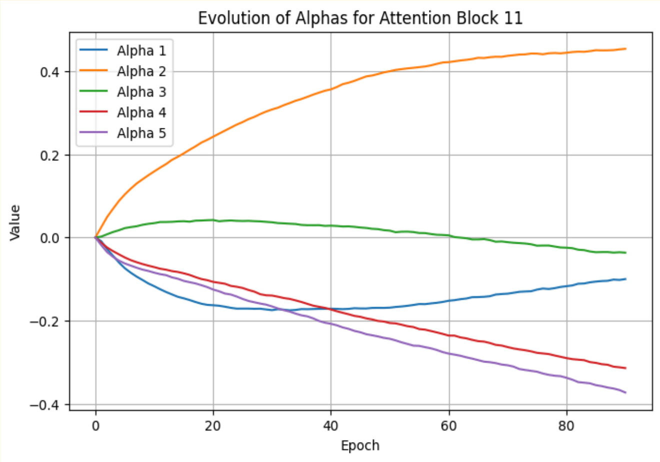
Current Results - Training Progress

- Over 100 epochs, loss (KL-divergence) decreased from 2.1470 to 0.3881 on our dataset
  - Custom attention can mimic the reference model's distributions
  - Model successfully learns sparse attention pattern
- Tested with a few prompts, resulting in output text mimicing style similar to GPT-2, though often less coherent due to the limited context
- L1 regularization experiment: replaced attention layer with 2 possible candidate masks: first token and all tokens (fill attention)

Current Results - 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 - Sample Outputs

```
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 ...
```

- Custom model's outputs sometimes drift or become less coherent
- Follows prompts and produce recognizable English words
- Custom model captures some of GPT-2's output token distribution

Current Limitations

- Limited mask optimization
  - Currently using simple weighted linear combinations of 3-5 fixed attention masks
- Need to train on larger dataset for more epochs
- No measure of memory or speed usage

Next Steps

- Optimize over more varied candidate masks and matrix families
- Testing models other than GPT2
- Extend to full WikiText-2 dataset / more training data
- Measure memory usage and speed improvements
- Optimize hyperparameters
- We've conducted a literature review of recent developments in optimizations that affect attention (i.e. Lexico, NSA), considering trying to implement these / finding ways to imitate these papers

Thank you!

Any questions?