Overview

This project focuses on optimizing transformer model efficiency through Dynamic Token Pruning, a method that reduces the number of processed tokens dynamically based on their saliency scores. The pruning process effectively reduces computational costs while maintaining model accuracy, making it suitable for real-world applications requiring efficiency and performance.

Key Contributions

1. Efficiency Bottleneck Identification

Profiling revealed self-attention and feedforward layers as the main contributors to FLOPs and runtime.

Baseline Model FLOPs: 12.908G, CUDA Time: 12.490ms.

2. Optimization Method

Dynamic Token Pruning:

Compute token saliency scores using the L2 norm.

Prune tokens with saliency scores below a threshold (e.g., 13.0).

Adjust sequence length dynamically across layers.

3. Results

Accuracy: Maintained at 52.5% for both baseline and pruned models.

Efficiency Improvements:

FLOPs Reduction: 31.2% (12.908G → 8.874G).

CUDA Time Reduction: 20.2% (12.490ms → 9.968ms).

Implementation

Baseline and Pruned Models:

Baseline: Standard transformer encoder.

Pruned: Enhanced with token pruning at each layer.

Profiling: Used PyTorch profiling tools to measure FLOPs and memory usage.

Conclusion

Dynamic Token Pruning offers a scalable approach to optimize transformer efficiency. It significantly reduces computational overhead without compromising model accuracy, achieving both efficiency and performance goals.