

Faster Convergence of Sparse Attention: revision

Left: Submitted draft (main.pdf, 2026-10-16), 2 pages. Right: Camera-ready (main.pdf, 2026-10-16), 2 pages.

removed or replaced text (left page) added or replacement text (right page) text removed here / added at this point on the other side margin bar: line with a change

31 words removed, 111 words added. Each revised page is shown next to the earlier page with the most text in common; earlier pages with no counterpart get a spread of their own. Click a section to jump to it.
1 figures changed or added, 1 changed or removed (dashed frames). 44 words moved without changes (not marked). Changes are detected on the words and figures of the compiled PDFs. Title pages and front matter: -1 +24 words.

1 Introduction	-7	+15	p. 1
2 Method	-7	+26	p. 1
3 Experiments	-15	+34	p. 1
4 Conclusion	-1	+12	p. 2

Faster Convergence of Sparse Attention through Adaptive Token Pruning

Ada Lindqvist Rafael Moreno

Abstract

Transformer models spend most of their compute on attention over tokens that contribute little to the final prediction. We propose adaptive token pruning, a simple method that removes low-importance tokens between layers. On three language modelling benchmarks our method reduces training time by 31% while matching the perplexity of a dense baseline.

1 Introduction

Large language models are expensive to train, and a large share of that cost comes from the attention mechanism, whose complexity grows quadratically with sequence length. Many tokens, such as punctuation and function words, receive little attention in deeper layers, yet they are processed at every step.

Earlier work has explored fixed sparsity patterns and learned routing. Fixed patterns are cheap but ignore the content of the sequence, while learned routing adds parameters and is hard to train. We take a middle path: we score every token with a lightweight importance estimate and drop the lowest-scoring tokens after each block.

Our contributions are a pruning rule that needs no extra parameters, an analysis of its effect on gradient flow, and experiments showing faster convergence on three benchmarks.

2 Method

For a sequence of hidden states $h_1, \dots, h_n$ at layer ℓ , we define the importance of token i as the total attention it receives:

$$s_i^{(\ell)} = \sum_{j=1}^n A_{ji}^{(\ell)}. \tag{1}$$

After each block we keep the $k = \lceil \rho n \rceil$ tokens with the highest scores and pass only those to the next layer. The keep ratio ρ is fixed at 0.7 for all layers. Pruned tokens are restored at the output layer by copying their last hidden state, so the model still produces a prediction for every position.

3 Experiments

We train 350M-parameter models on WikiText-103, The Pile and C4 for 50,000 steps with a batch size of 256 sequences. Table 1 compares perplexity and wall-clock training time.

Pruning costs less than 0.2 perplexity points on every benchmark while cutting training time by about a third. The gap is largest on The Pile, which contains long documents with many repeated tokens.

Faster Convergence of Sparse Attention through Adaptive Token Pruning

Ada Lindqvist Rafael Moreno

Abstract

Transformer models spend most of their compute on attention over tokens that contribute little to the final prediction. We propose adaptive token pruning, a simple method that removes low-importance tokens between layers. On three language modelling benchmarks our method reduces training time by 34% while matching the perplexity of a dense baseline. The method requires no changes to the model architecture and can be added to existing training pipelines in a few lines of code.

1 Introduction

Large language models are expensive to train, and a large share of that cost comes from the attention mechanism, whose complexity grows quadratically with sequence length. Our measurements show that many tokens, such as punctuation and function words, receive less than 1% of the attention mass in deeper layers, yet they are processed at every step.

Earlier work has explored fixed sparsity patterns and learned routing. Fixed patterns are cheap but ignore the content of the sequence, while learned routing adds parameters and often destabilises training. We take a middle path: we score every token with a lightweight importance estimate and drop the lowest-scoring tokens after each block.

Our contributions are a pruning rule that needs no extra parameters, an analysis of its effect on gradient flow, and experiments showing faster convergence on three benchmarks.

2 Method

For a sequence of hidden states $h_1, \dots, h_n$ at layer ℓ , we define the importance of token i as the total attention it receives:

$$s_i^{(\ell)} = \sum_{j=1}^n A_{ji}^{(\ell)}. \tag{1}$$

After each block we keep the $k = \lceil \rho n \rceil$ tokens with the highest scores and pass only those to the next layer. The keep ratio ρ decreases linearly from 0.9 in the first layer to 0.5 in the last, so that early layers, which build local features, see most of the sequence. Pruned tokens are restored at the output layer by copying their last hidden state, so the model still produces a prediction for every position.

3 Experiments

Following prior work, we train 350M-parameter models on WikiText-103, The Pile and C4 for 50,000 steps with a batch size of 256 sequences. Table 1 compares perplexity and wall-clock training time.

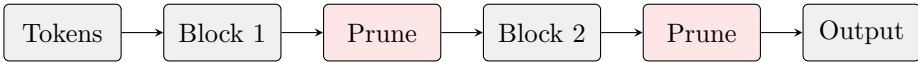


Figure 1: Adaptive token pruning between transformer blocks.

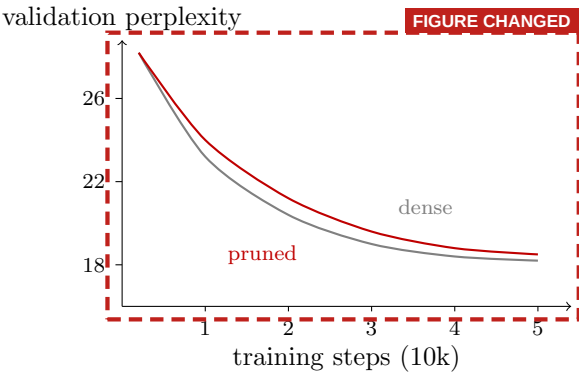


Figure 2: Validation perplexity on WikiText-103 during training.

Table 1: Perplexity and training time on three benchmarks.			
Model	WikiText-103	The Pile	C4
Dense baseline	18.4	12.9	16.2
Adaptive pruning (ours)	18.6	13.1	16.3
Training time (hours)	41 / 29	96 / 66	88 / 61

4 Conclusion

Adaptive token pruning is a simple way to make attention cheaper without new parameters. In future work we plan to apply it to encoder models.

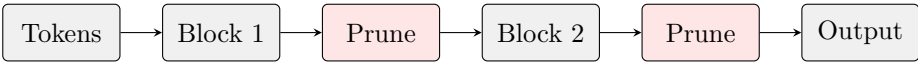


Figure 1: Adaptive token pruning between transformer blocks.

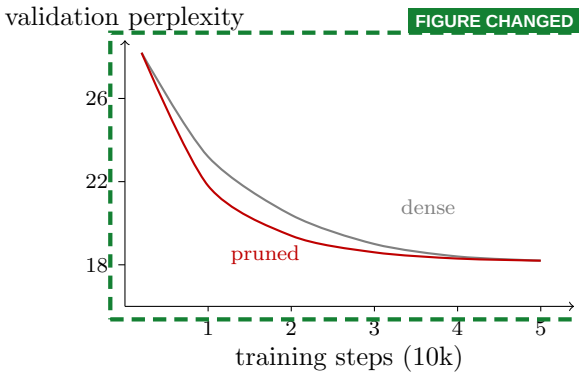


Figure 2: Validation perplexity on WikiText-103 during training.

Figure 2 shows that the pruned model now reaches the dense model’s perplexity after 40,000 steps.

Table 1: Perplexity and training time on three benchmarks.			
Model	WikiText-103	The Pile	C4
Dense baseline	18.4	12.9	16.2
Adaptive pruning (ours)	18.5	12.9	16.3
Training time (hours)	41 / 27	96 / 62	88 / 58

Pruning costs at most 0.1 perplexity points on every benchmark while cutting training time by about a third. The gap is largest on The Pile, which contains long documents with many repeated tokens.

4 Conclusion

Adaptive token pruning is a simple way to make attention cheaper without new parameters. In future work we plan to apply it to encoder models and to longer contexts, where the savings should be larger still.