Longformer: The Long-Document Transformer

Abstract

1. Introduction

Existing approaches partition or shorten the long context into smaller sequences that fall within the typical 512 token limit of BERT-style pretrained models. Such partitioning could potentially result in loss of important cross-partition information, and to mitigate this problem, existing methods often rely on complex architectures to address such interactions.

Longformer's attention mechanism is a combination of a windowed local-context self-attention and an end task motivated global attention that encodes inductive bias about the task. The local attention is primarily used to build contextual representations, while the global attention allows Longformer to build full sequence representations for prediction.

2. Related Work

3. Longformer

3.1 Attention Pattern

Sliding Window: Given a fixed window size w, each token attends to $\frac{1}{2}w$ tokens on each side. The computation complexity is $O(w \times n)$. Using multiple stacked layers results in a large receptive field $l \times w$, where top layers have access to all input locations and have the capacity to build representation of entire input.

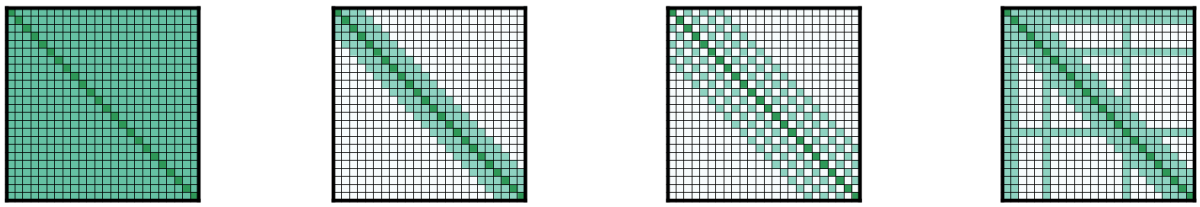
Dilated Sliding Window: The window has gaps of size dilation d. The receptive field is $l \times d \times w$, which can be reach tens of thousands of tokens even for small values of d.

Global Attention: We add "global attention" on few pre-selected input locations. A token with a global attention attends to all tokens across the sequence, and all tokens in the sequence attend to it.

Linear Projections for Global Attention: The Transformer model computes attention scores as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

We use Q_s, K_s, V_s to compute attention scores of sliding window attention, and Q_g, K_g, V_g to compute attention scores for the global attention. Q_g, K_g, V_g are all initialized with values that match Q_s, K_s, V_s.



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Figure 2: Comparing the full self-attention pattern and the configuration of attention patterns in our Longformer.

3.2 Implementation

As shown in Fig. 1, "loop" is a memory efficient PyTorch implementation that supports dilation but is unusably slow and only used for testing, "chunks" only supports the non-dilated case and is used for the pretraining/finetuning setting, and "cuda" is our fully functioning highly optimized custom CUDA kernel implemented using TVM and used for the language modeling experiments.

4. Autoregressive Language Modeling

Autoregressive or left-to-right language modeling is loosely defined as estimating the probability distribution of an existing token/character given its previous tokens/characters in an input sequence.

4.1 Attention Pattern

We use small window sizes (non-dilated) for the lower layers and increase window sizes (dilated) as we move to higher layers.

4.2 Experiment Setup

We focus on character-level LM with text8 and enwik8 dataset.

Training We adopt a staged training procedure where we increase the attention window size and sequence length across multiple training phases. In particular, in the first phase we start with a short sequence length and window size, then on each subsequent phase, we double the window size and the sequence length, and halve the learning rate.

Evaluation We split the dataset into overlapping sequences of size 32256 with a step of size 512 and report the performance on the last 512 tokens on the sequence.

5. Pretraining and Finetuning

We pretrained Longformer on a document corpus and finetuned it for six tasks, including classification, QA and coreference resolution. The resulting model can process sequences up to 4096 tokens long (8 times longer than BERT).

We pretrained Longformer with masked language modeling (MLM), where the goal is to recover randomly masked tokens in a sequence. We continue pretraining from the RoBERTa released checkpoint, while only making the minimal changes necessary to support Longformer's attention mechanism.

Attention Pattern We use sliding window attention with window size of 512, therefore using the same amount of computation as RoBERTa.

Position Embeddings RoBERTa uses learned absolute position embeddings with the maximum position being 512. We initialize the position embeddings copying the 512 position embeddings from RoBERTa mul-tiple times.

7. Longformer-Encoder-Decoder (LED)

Instead of the full self-attention in the encoder, LED uses the efficient local+global attention pattern of the Longformer, the decoder uses the full self-attention. Since pretraining LED is expensive, we initialize LED parameters from the BART. We extend position embedding to 16K tokens (up from BART's 1K tokens) and we initialize the new position embedding matrix by repeatedly copying BART's 1K position embeddings 16 times. We re-lease two model sizes, LED-base and LED-large, which respectively have 6 and 12 layers in both encoder and decoder stacks.

We evaluate LED on the summarization task us-ing the arXiv summarization dataset. The encoder uses local attention with window size 1,024 tokens and global attention on the first $< s >$ token.

	R-1	R-2	R-L
Discourse-aware (2018)	35.80	11.05	31.80
Extr-Abst-TLM (2020)	41.62	14.69	38.03
Dancer (2020)	42.70	16.54	38.44
Pegasus (2020)	44.21	16.95	38.83
LED-large (seqlen: 4,096) (ours)	44.40	17.94	39.76
BigBird (seqlen: 4,096) (2020)	**46.63**	19.02	41.77
LED-large (seqlen: 16,384) (ours)	**46.63**	**19.62**	**41.83**