Compressive Transformers for Long-Range Sequence Modelling

- ARUN

Content

- Model
- Compression Function
- Loss functions
- PG-19 Benchmark
- Results
- References

Model

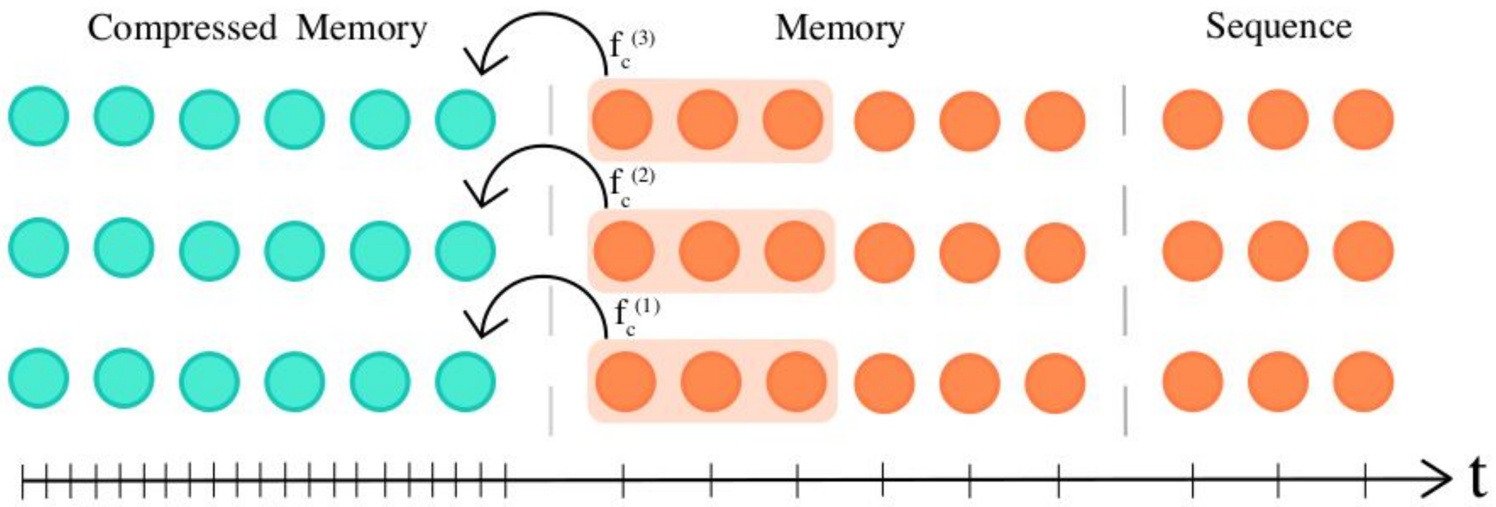

Figure 1: The Compressive Transformer keeps a fine-grained memory of past activations, which are then compressed into coarser *compressed* memories. The above model has three layers, a sequence length $n_s = 3$, memory size $n_m = 6$, compressed memory size $n_{cm} = 6$. The highlighted memories are compacted, with a compression function f_c per layer, to a single compressed memory — instead of being discarded at the next sequence. In this example, the rate of compression $c = 3$.

Compression Function

- Min/Max Pooling
- 1D Convolution
- Dilated convolutions
- Most used (most average attention)

Loss Functions

- BPTT loss (long unrolls)
- Autoencoder loss (compression objective)
  - Original memories from compressed
  - Attempts to retain all information
- Attention-reconstruction loss
  - Content based attention using compressed
- No mixing of losses

Compressive Transformer

Algorithm 1 Compressive Transformer

At time zero

$\mathbf{m_0} \leftarrow \mathbf{0}$ // Initialize memory to zeros ($l \times n_m \times d$)

$\mathbf{cm_0} \leftarrow \mathbf{0}$ // Initialize compressed memory to zeros ($l \times n_{cm} \times d$)

At time t

$\mathbf{h}^{(1)} \leftarrow \mathbf{xW_{emb}}$ // Embed input sequence($n_s \times d$)

for layer $i = 1, 2, \ldots, l$ **do**

  $\mathbf{mem^{(i)}} \leftarrow \text{concat}(\mathbf{cm_t^{(i)}}, \mathbf{m_t^{(i)}})$ // $((n_{cm} + n_m) \times d)$

  $\mathbf{\tilde{a}^{(i)}} \leftarrow \text{multihead\_attention}^{(i)}(\mathbf{h^{(i)}}, \mathbf{mem_t^{(i)}})$ // MHA over both mem types ($n_s \times d$)

  $\mathbf{a^{(i)}} \leftarrow \text{layer\_norm}(\mathbf{\tilde{a}^{(i)}} + \mathbf{h^{(i)}})$ // Regular skip + layernorm ($n_{cm} \times d$)

  $\mathbf{old\_mem^{(i)}} \leftarrow \mathbf{m_t^{(i)}}[: n_s]$ // Oldest memories to be forgotten ($n_s \times d$)

  $\mathbf{new\_cm^{(i)}} \leftarrow f_c^{(i)}(\mathbf{old\_mem^{(i)}})$ // Compress oldest memories by factor c ($\lfloor \frac{n_s}{c} \rfloor \times d$)

  $\mathbf{m_{t+1}^{(i)}} \leftarrow \text{concat}(\mathbf{m_t^{(i)}}, \mathbf{h^{(i)}})[-n_m :]$ // Update memory ($n_m \times d$)

  $\mathbf{cm_t^{(i)}} \leftarrow \text{concat}(\mathbf{cm_t^{(i)}}, \mathbf{new\_cm^{(i)}})[-n_{cm} :]$ // Update compressed memory ($n_{cm} \times d$)

  $\mathbf{h^{(i+1)}} \leftarrow \text{layer\_norm}(\text{mlp}^{(i)}(\mathbf{a^{(i)}}) + \mathbf{a^{(i)}})$ // Mixing MLP ($n_s \times d$)

Attention Reconstruction Loss

Algorithm 2 Attention-Reconstruction Loss

$L^{attn} \leftarrow 0$

for layer $i = 1, 2, \ldots, l$ **do**

  $\mathbf{h^{(i)}} \leftarrow \text{stop\_gradient}(\mathbf{h^{(i)}})$ // Stop compression grads from passing...

  $\mathbf{old\_mem^{(i)}} \leftarrow \text{stop\_gradient}(\mathbf{old\_mem^{(i)}})$ // ...into transformer network.

  $\mathbf{Q}, \mathbf{K}, \mathbf{V} \leftarrow$ stop_gradient(attention params at layer i) // Re-use attention weight matrices.

  def $\text{attn}(\mathbf{h}, \mathbf{m}) \leftarrow \sigma((\mathbf{hQ})\,(\mathbf{mK}))(\mathbf{mV})$ // Use content-based attention (no relative).

  $\mathbf{new\_cm^{(i)}} \leftarrow f_c^{(i)}(\mathbf{old\_mem^{(i)}})$ // Compression network (to be optimized).

  $L^{attn} \leftarrow L^{attn} + ||\text{attn}(\mathbf{h^{(i)}}, \mathbf{old\_mem^{(i)}}) - \text{attn}(\mathbf{h^{(i)}}, \mathbf{new\_cm^{(i)}})||_2$

PG-19 Benchmark

Table 1: Comparison to existing popular language modelling benchmarks.

	Avg. length (words)	Train Size	Vocab	Type
1B Word	27	4.15GB	793K	News (sentences)
Penn Treebank	355	5.1MB	10K	News (articles)
WikiText-103	3.6K	515MB	267K	Wikipedia (articles)
PG-19	69K	10.9GB	(open)	Books

PG-19 Benchmark

Table 2: PG-19 statistics split by subsets.

	Train	**Valid.**	**Test**
# books	28,602	50	100
# words	1,973,136,207	3,007,061	6,966,499

Table 3: Eval. perplexities on PG-19.

	Valid.	Test
36L TransformerXL	45.5	36.3
36L Compressive Transf.	43.4	33.6

Table 4: State-of-the-art results on Enwik8.

Model	**BPC**
7L LSTM (Graves, 2013)	1.67
LN HyperNetworks Ha et al. (2016)	1.34
LN HM-LSTM Chung et al. (2016)	1.32
ByteNet (Kalchbrenner et al., 2016)	1.31
RHN Zilly et al. (2017)	1.27
mLSTM Krause et al. (2016)	1.24
64L Transf. Al-Rfou et al. (2019)	1.06
24L TXL (Dai et al., 2019)	0.99
Sparse Transf. (Child et al., 2019)	0.991
Adaptive Transf. (Sukhbaatar et al., 2019)	0.98
24L TXL (ours)	0.98
24L Compressive Transformer	**0.97**

Table 5: Compression approaches on Enwik8.

Compression fn	**Compression loss**	**BPC**
Conv	BPTT	0.996
Max Pooling	N/A	0.986
Conv	Auto-encoding	0.984
Mean Pooling	N/A	0.982
Most-used	N/A	0.980
Dilated conv	Attention	0.977
Conv	Attention	**0.973**

WikiText-103

Table 6: Validation and test perplexities on WikiText-103.

	Valid.	Test
LSTM (Graves et al., 2014)	-	48.7
Temporal CNN (Bai et al., 2018a)	-	45.2
GCNN-14 (Dauphin et al., 2016)	-	37.2
Quasi-RNN Bradbury et al. (2016)	32	33
RMC (Santoro et al., 2018)	30.8	31.9
LSTM+Hebb. (Rae et al., 2018)	29.0	29.2
Transformer (Baevski and Auli, 2019)	-	18.7
18L TransformerXL, M=384 (Dai et al., 2019)	-	18.3
18L TransformerXL, M=1024 (ours)	-	18.1
18L Compressive Transformer, M=1024	**16.0**	**17.1**

WikiText-103

Table 7: WikiText-103 test perplexity broken down by word frequency buckets. The most frequent bucket is words which appear in the training set more than $10,000$ times, displayed on the left. For reference, a uniform model would have perplexity $|V| = 2.6e5$ for all frequency buckets. *LSTM comparison from Rae et al. (2018)

	$> 10K$	1K–10K	$100 - 1K$	< 100	All
LSTM*	12.1	219	1,197	9,725	36.4
TransformerXL (ours)	7.8	61.2	188	1,123	18.1
Compressive Transformer	**7.6**	**55.9**	**158**	**937**	**17.1**
Relative gain over TXL	2.6%	9.5%	21%	19.9%	5.8%

Attention & Optimization

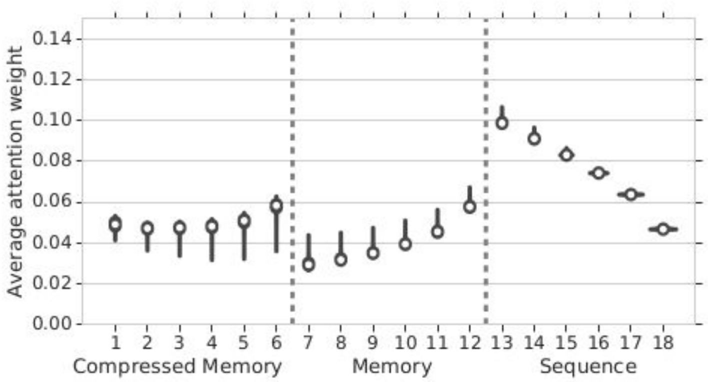

Figure 2: **Attention weight on Enwik8**. Average attention weight from the sequence over the compressed memory (oldest), memory, and sequence (newest) respectively. The sequence self-attention is causally masked, so more attention is placed on earlier elements in the sequence. There is an increase in attention at the transition from memory to compressed memory.

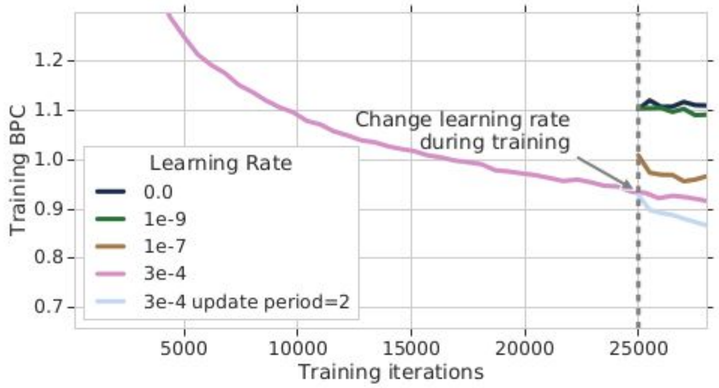

Figure 3: **Learning rate analysis**. Reducing the learning rate (e.g. to zero) during training (on Enwik8) harms training performance. Reducing the frequency of optimisation updates (effectively increasing the batch size) is preferable.

Speech & RL

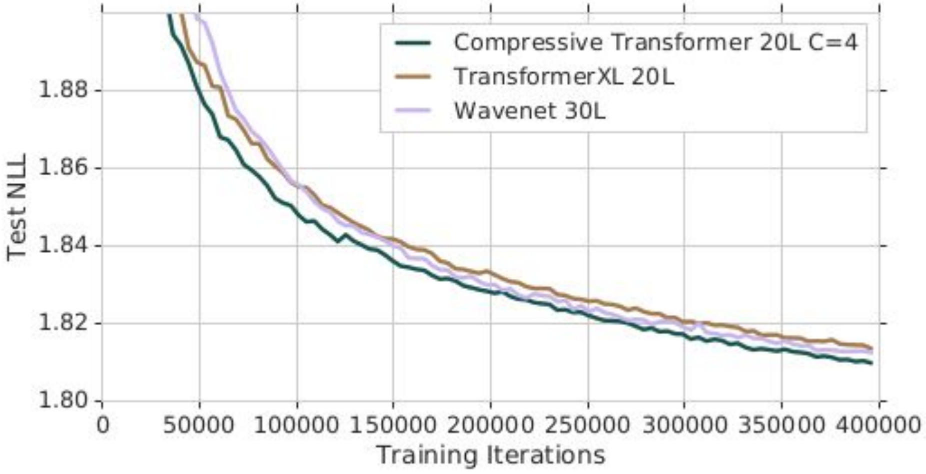

Figure 4: **Speech Modelling.** We see the Compressive Transformer is able to obtain competitive results against the state-of-the-art WaveNet in the modelling of raw speech sampled at 24kHz.

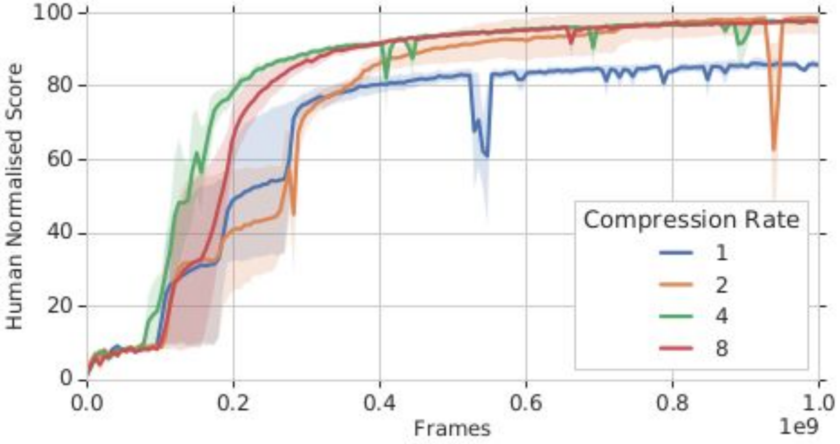

Figure 5: **Vision and RL**. We see the Compressive Transformer integrates visual information across time within an IMPALA RL agent, trained on an object matching task.

References

- https://github.com/UBC-NLP/dlr/blob/master/slides/20190308_transformer-XL.pdf
- https://arxiv.org/pdf/1911.05507.pdf
- https://www.reddit.com/r/MachineLearning/comments/dbqgy4/r_compressive_transformers_for_longrange_sequence/
- https://torontoai.org/2019/09/30/r-compressive-transformers-for-long-range-sequence-modelling/
- https://www.semanticscholar.org/paper/Compressive-Transformers-for-Long-Range-Sequence-Rae-Potapenko/f51497f463566581874c941353dd9d80069c5b77