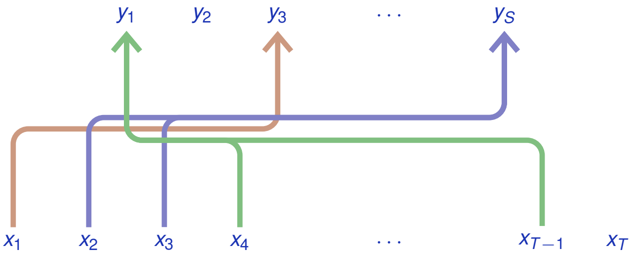
Fast Attention Models

François Fleuret

Joint work with Angelos Katharopoulos,
Apoorv Vyas, and Nikos Pappas.

Attention Layers

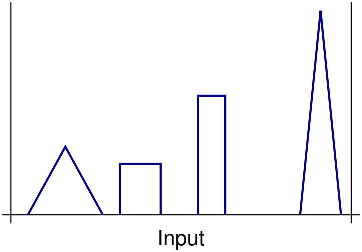
There has been recently a strong interest for attention mechanisms to transport information from parts of the signal to other parts **specified dynamically**.

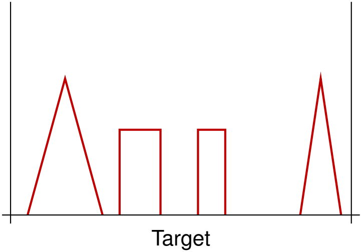


Such mechanisms are very efficient for natural language processing, for which they have replaced recurrent architectures.

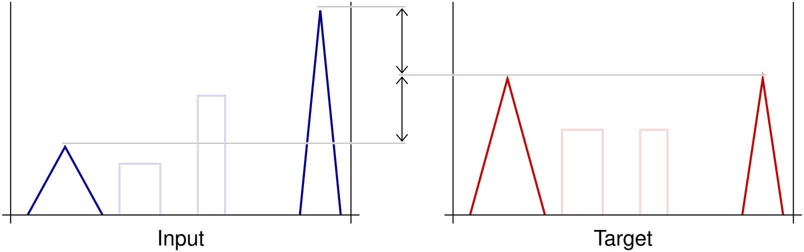
Consider a task with 1d sequences composed of two triangles and two rectangles, where the goal is to average heights in each **pair of shapes**.

Consider a task with 1d sequences composed of two triangles and two rectangles, where the goal is to average heights in each **pair of shapes**.

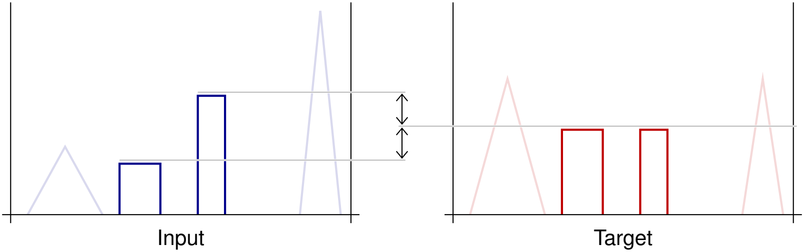




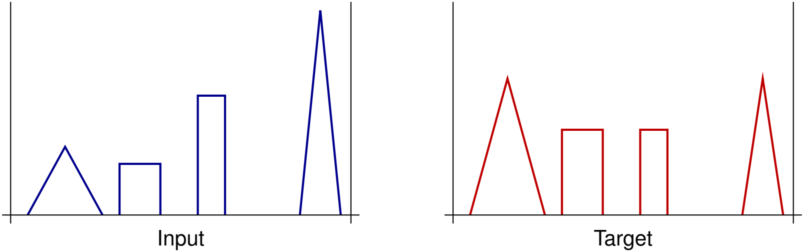
Consider a task with 1d sequences composed of two triangles and two rectangles, where the goal is to average heights in each **pair of shapes**.

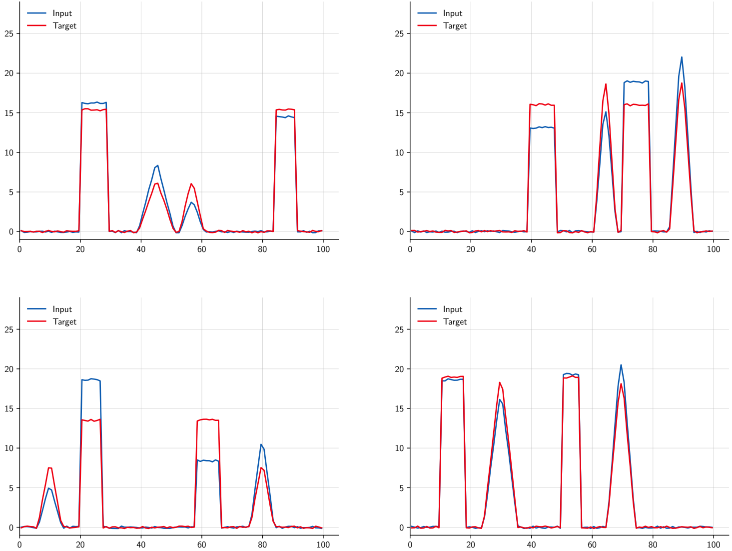


Consider a task with 1d sequences composed of two triangles and two rectangles, where the goal is to average heights in each **pair of shapes**.



Consider a task with 1d sequences composed of two triangles and two rectangles, where the goal is to average heights in each **pair of shapes**.




Input
Target
Input
Target
Input
Target
Input
Target

Given a input sequence $X \in \mathbb{R}^{T \times D}$, a standard convolution layer computes a result $X' \in \mathbb{R}^{T \times D'}$ with

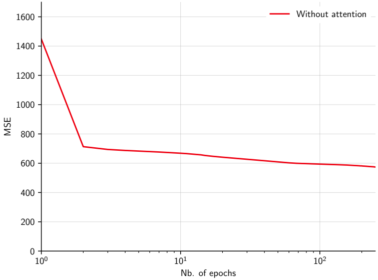
$$\forall t, \; X'_t = \sum_{s=t}^{t+\Delta} W_{s-t} X_s.$$

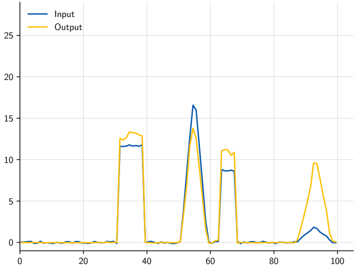
Given a input sequence $X \in \mathbb{R}^{T \times D}$, a standard convolution layer computes a result $X' \in \mathbb{R}^{T \times D'}$ with

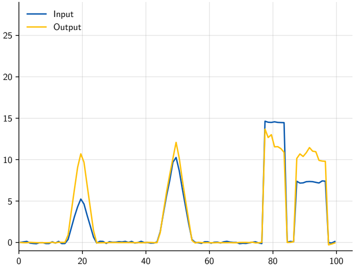
$$\forall t, \; X'_t = \sum_{s=t}^{t+\Delta} W_{s-t} X_s .$$

We test first a 1d convolutional network, with no attention mechanism.

```
Sequential(
  (0): Conv1d(1, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (1): ReLU()
  (2): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (3): ReLU()
  (4): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (5): ReLU()
  (6): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (7): ReLU()
  (8): Conv1d(64, 1, kernel_size=(5,), stride=(1,), padding=(2,))
)

nb_parameters 62337
```


Without attention
MSE
Nb. of epochs
1600
1400
1200
1000
800
600
400
200
0
10^0
10^1
10^2


Input
Output
25
20
15
10
5
0
0
20
40
60
80
100


Input
Output
25
20
15
10
5
0
0
20
40
60
80
100

The poor performance of this model is not surprising given its inability to channel information from “far away” in the signal.

More layers, global averaging, or fully connected layers could possibly solve the problem. However it is more natural to equip the model with the ability to fetch information from parts of the signal that it actively identifies as relevant.

This is exactly what an **attention layer** does.

Given a sequence $X \in \mathbb{R}^{T \times D}$, a standard self-attention layer computes first three sequences:

- the queries: $Q = X W_Q^\top \in \mathbb{R}^{T \times C}$,
- the keys: $K = X W_K^\top \in \mathbb{R}^{T \times C}$,
- the values: $V = X W_V^\top \in \mathbb{R}^{T \times D'}$,

Given a sequence $X \in \mathbb{R}^{T \times D}$, a standard self-attention layer computes first three sequences:

- the queries: $Q = X\, W_Q^\top \in \mathbb{R}^{T \times C}$,
- the keys: $K = X\, W_K^\top \in \mathbb{R}^{T \times C}$,
- the values: $V = X\, W_V^\top \in \mathbb{R}^{T \times D'}$,

from which it computes an attention matrix

$$\forall t, s,\ A_{t,s} = \frac{\exp\left(Q_t\, K_s^\top\right)}{\sum_{u=1}^{T} \exp\left(Q_t\, K_u^\top\right)}$$

where $A_{t,s}$ should be interpreted as **how much position s matters for computing the result at position t.**

Given a sequence $X \in \mathbb{R}^{T \times D}$, a standard self-attention layer computes first three sequences:

- the queries: $Q = X\,W_Q^\top \in \mathbb{R}^{T \times C}$,
- the keys: $K = X\,W_K^\top \in \mathbb{R}^{T \times C}$,
- the values: $V = X\,W_V^\top \in \mathbb{R}^{T \times D'}$,

from which it computes an attention matrix

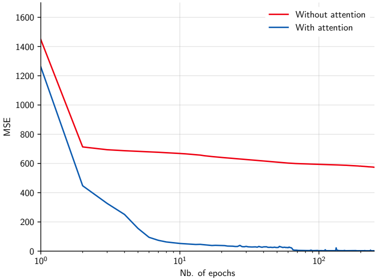
$$\forall t, s,\ A_{t,s} = \frac{\exp\left(Q_t\,K_s^\top\right)}{\sum_{u=1}^{T} \exp\left(Q_t\,K_u^\top\right)}$$

where $A_{t,s}$ should be interpreted as **how much position s matters for computing the result at position t.**

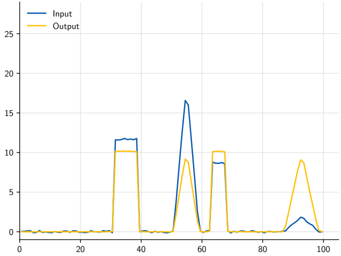
And the resulting sequence $X' \in \mathbb{R}^{T \times D'}$ is

$$\forall t,\ X'_t = \sum_{s=1}^{T} A_{t,s} V_s.$$

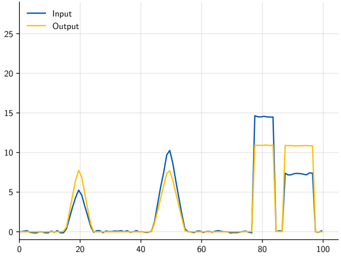
```
Sequential(
  (0): Conv1d(1, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (1): ReLU()
  (2): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (3): ReLU()
  (4): AttentionLayer(in_channels=64, out_channels=64, key_channels=64)
  (5): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (6): ReLU()
  (7): Conv1d(64, 1, kernel_size=(5,), stride=(1,), padding=(2,))
)

nb_parameters 54081
```



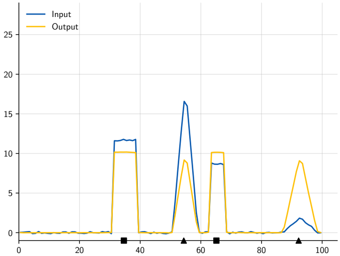
Without attention
With attention
MSE
Nb. of epochs

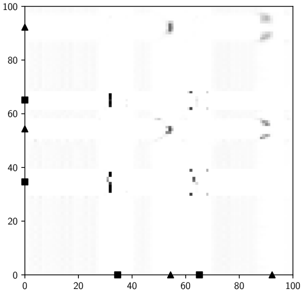


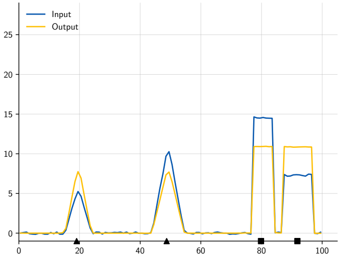
Input
Output

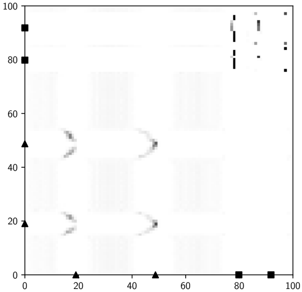


Input
Output

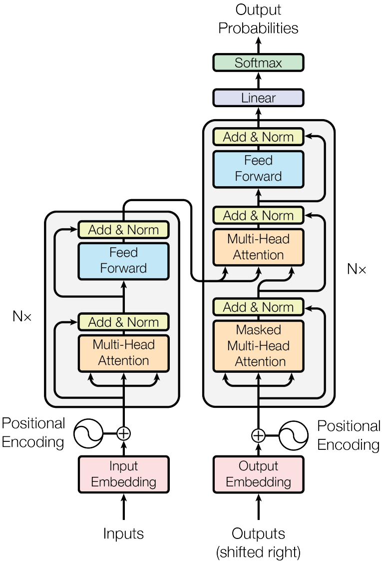

Input
Output
0
5
10
15
20
25
0
20
40
60
80
100


0
20
40
60
80
100
0
20
40
60
80
100


Input
Output
0
5
10
15
20
25
0
20
40
60
80
100


0
20
40
60
80
100
0
20
40
60
80
100

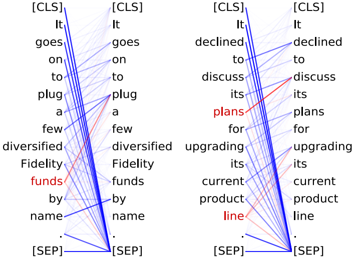
Transformers



(Vaswani et al., 2017)

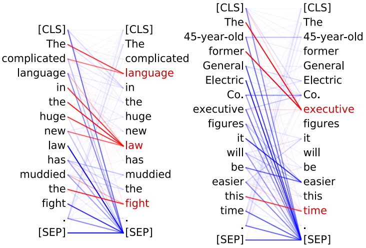
Head 8-10

- **Direct objects** attend to their verbs
- 86.8% accuracy at the `dobj` relation



Head 8-11

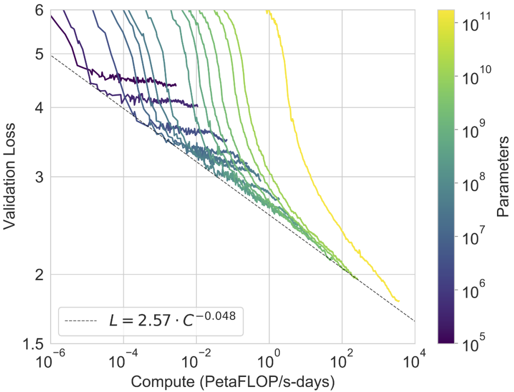
- **Noun modifiers** (e.g., determiners) attend to their noun
- 94.3% accuracy at the `det` relation



(Clark et al., 2019)

Model	Total train compute (PF-days)	Total train compute (flops)	Params (M)	Training tokens (billions)
T5-Small	2.08E+00	1.80E+20	60	1,000
T5-Base	7.64E+00	6.60E+20	220	1,000
T5-Large	2.67E+01	2.31E+21	770	1,000
T5-3B	1.04E+02	9.00E+21	3,000	1,000
T5-11B	3.82E+02	3.30E+22	11,000	1,000
BERT-Base	1.89E+00	1.64E+20	109	250
BERT-Large	6.16E+00	5.33E+20	355	250
RoBERTa-Base	1.74E+01	1.50E+21	125	2,000
RoBERTa-Large	4.93E+01	4.26E+21	355	2,000
GPT-3 Small	2.60E+00	2.25E+20	125	300
GPT-3 Medium	7.42E+00	6.41E+20	356	300
GPT-3 Large	1.58E+01	1.37E+21	760	300
GPT-3 XL	2.75E+01	2.38E+21	1,320	300
GPT-3 2.7B	5.52E+01	4.77E+21	2,650	300
GPT-3 6.7B	1.39E+02	1.20E+22	6,660	300
GPT-3 13B	2.68E+02	2.31E+22	12,850	300
GPT-3 175B	3.64E+03	3.14E+23	174,600	300

(Brown et al., 2020)



(Brown et al., 2020)

Multiple attempts have been made at reducing the computational cost:

- Weight pruning (Michel et al., 2019), weight factorization (Lan et al., 2020), weight quantization (Zafrir et al., 2019).
- Model distillation (Sanh et al., 2019).
- Controlling the attention horizon (Dai et al., 2019; Sukhbaatar et al., 2019).
- Sparse factorization of the attention matrix (Child et al., 2019).
- Hashing (Kitaev et al., 2020).

We have developed one approach that clusters queries to make the cost $O(CT)$ instead of $O(T^2)$ (Vyas et al., 2020), and a second that linearizes the attention score (Katharopoulos et al., 2020).

Linear Attention

An important part of the computation goes into the $O(T^2)$ attention-based processing:

$$X'_t = \frac{\sum_s \exp\left(Q_t K_s^\top\right) V_s}{\sum_s \exp\left(Q_t K_s^\top\right)}.$$

An important part of the computation goes into the $O(T^2)$ attention-based processing:

$$X'_t = \frac{\sum_s \exp\left(Q_t K_s^\top\right) V_s}{\sum_s \exp\left(Q_t K_s^\top\right)}.$$

If we kernelize the similarity measure, the expression becomes linear:

$$X'_t \simeq \frac{\sum_s \left(\Phi(Q_t)\, \Phi(K_s)^\top\right) V_s}{\sum_s \Phi(Q_t)\, \Phi(K_s)^\top}.$$

An important part of the computation goes into the $O(T^2)$ attention-based processing:

$$X'_t = \frac{\sum_s \exp\left(Q_t K_s^\top\right) V_s}{\sum_s \exp\left(Q_t K_s^\top\right)}.$$

If we kernelize the similarity measure, the expression becomes linear:

$$X'_t \simeq \frac{\sum_s \left(\Phi(Q_t)\, \Phi(K_s)^\top\right) V_s}{\sum_s \Phi(Q_t)\, \Phi(K_s)^\top}.$$

And we can use the associativity of the matrix product to reduce the cost

$$\underbrace{\left(\Phi(Q)\Phi(K)^\top\right) V}_{O(T^2 D)+O(T^2 D)} = \underbrace{\Phi(Q) \left(\Phi(K)^\top V\right)}_{O(TD^2)+O(TD^2)}.$$

(Katharopoulos et al., 2020)

In practice we take

$$\Phi(x) = \mathrm{ELU}(x) + 1.$$

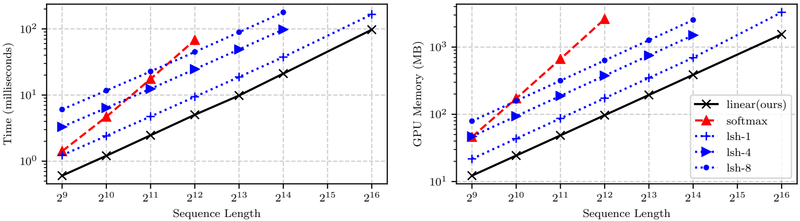


Figure 1: Comparison of the computational requirements for a forward/backward pass for Reformer (lsh-X), softmax attention and linear attention. Linear and Reformer models scale linearly with the sequence length unlike softmax which scales with the square of the sequence length both in memory and time. Full details of the experiment can be found in § 4.1.

(Katharopoulos et al., 2020)

Additionally, when they are used as generative models (*e.g.* translation) transformers require to process the sequence for every new token.

Our linearization allows to keep running quantities:

$$
\begin{aligned}
X'_{t+1} &\simeq \frac{\sum_{s=1}^{t} \Phi(Q_{t+1})\,\Phi(K_s)V_s}{\sum_{s=1}^{t} \Phi(Q_{t+1})\,\Phi(K_s)} \\
&= \frac{\Phi(Q_{t+1})\,\sum_{s=1}^{t} \Phi(K_s)V_s}{\Phi(Q_{t+1})\,\sum_{s=1}^{t} \Phi(K_s)} \\
&= \frac{\Phi(Q_{t+1})\,\left(\left(\sum_{s=1}^{t-1} \Phi(K_s)^\top V_s\right) + \Phi(K_t)^\top V_t\right)}{\Phi(Q_{t+1})\,\left(\left(\sum_{s=1}^{t-1} \Phi(K_s)^\top\right) + \Phi(K_t)^\top\right)}
\end{aligned}
$$

which can be interpreted as the hidden state of a recurrent unit.

(Katharopoulos et al., 2020)

(Katharopoulos et al., 2020)

Method	Bits/dim	Images/sec	
Softmax	0.621	0.45	(1×)
LSH-1	0.745	0.68	(1.5×)
LSH-4	0.676	0.27	(0.6×)
Linear (ours)	0.644	**142.8**	**(317×)**

Table 1: Comparison of autoregressive image generation of MNIST images. Our linear transformers achieve almost the same bits/dim as the full softmax attention but more than 300 times higher throughput in image generation. The full details of the experiment are in § 4.2.1.

Method	Bits/dim	Images/sec	
Softmax	3.47	0.004	(1×)
LSH-1	3.39	0.015	(3.75×)
LSH-4	3.51	0.005	(1.25×)
Linear (ours)	3.40	**17.85**	**(4,462×)**

Table 2: We train autoregressive transformers for 1 week on a single GPU to generate CIFAR-10 images. Our linear transformer completes 3 times more epochs than softmax, which results in better perplexity. Our model generates images 4,000× faster than the baselines. The full details of the experiment are in § 4.2.2.

(Katharopoulos et al., 2020)

We need more fast deep models:

- Lots of promising applications of ML involve very large signals (particle physics, astronomy, microscopy, satellite imaging).
- The trend toward larger models does not seem to slow down.
- Attention mechanisms provide a natural mean to dynamically allocate bandwidth in a model.

The end

References

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. **Language models are few-shot learners**. CoRR, abs/2005.14165, 2020.

R. Child, S. Gray, A. Radford, and I. Sutskever. **Generating long sequences with sparse transformers**. arXiv preprint arXiv:1904.10509, 2019.

K. Clark, U. Khandelwal, O. Levy, and C. Manning. **What does BERT look at? An analysis of BERT's attention**. CoRR, abs/1906.04341, 2019.

Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov. **Transformer-XL: Attentive language models beyond a fixed-length context**. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. **Transformers are RNNs: Fast autoregressive transformers with linear attention**. In Proceedings of the International Conference on Machine Learning (ICML), 2020. (to appear).

N. Kitaev, Ł. Kaiser, and A. Levskaya. **Reformer: The efficient transformer**. arXiv preprint arXiv:2001.04451, 2020.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. **Albert: A lite bert for self-supervised learning of language representations**. In International Conference on Learning Representations, 2020.

P. Michel, O. Levy, and G. Neubig. **Are sixteen heads really better than one?** In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 14014–14024. Curran Associates, Inc., 2019.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. **Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter**. CoRR, abs/1910.01108, 2019.

S. Sukhbaatar, E. Grave, P. Bojanowski, and A. Joulin. **Adaptive attention span in transformers**. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 331–335, Florence, Italy, July 2019. Association for Computational Linguistics.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. **Attention is all you need**. CoRR, abs/1706.03762, 2017.

A. Vyas, A. Katharopoulos, and F. Fleuret. **Fast transformers with clustered attention**. CoRR, abs/2007.04825, 2020.

O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat. **Q8BERT: quantized 8bit BERT**. CoRR, abs/1910.06188, 2019.