# A Survey on Transformer Models in Machine Learning

**Hannes Stärk**

## Abstract

Transformers make amazing achievements across numerous domains but especially in natural language processing. This new type of neural network processes sequences without using recurrence. Instead, the models process multiple parts of a sequence in parallel, using scaled dot-product attention as their core mechanism. This survey explains in detail how the architecture works, its properties and applications.

## 1. Introduction

A transformer is a special type of neural network that has performed exceptionally well in several sequence-based tasks. Previously, Recurrent Neural Networks (RNNs) and Long Short Term Memory (LSTMs) [10; 20] were the standard for sequential data and natural language processing (NLP). Bahdanau et al. [4] further improved the dominant models by equipping them with an attention mechanism. Transformers removed this recurrence and instead only uses attention that learns to simultaneously route information from an entire segment of a sequence [38]. Transformers are most successfully applied as language models such as BERT and GPT-2 that transformed the field of NLP.

This paper provides a concise but detailed explanation of the original architecture and the most important improvements and innovations building on it. It is discussed how properties arise from components like the attention mechanism and what advantages and disadvantages this entails compared to recurrence-based models. Some widely used models like BERT, GPT-2 or XLNet are covered. While transformers revolutionized NLP, their success in other fields cannot be ignored. Additional interesting and creative applications are highlighted. We consider under which circumstances the models perform well but also where the theoretical limitations and practical problems are. Possible solutions and research ideas are proposed.

## 2. The original Transformer

The transformer of Vaswani et al. [38] consists of an encoder and a decoder and processes multiple tokens of a sequence in parallel. It was trained for machine translation so in our example the tokens are words. An input segment is an English sentence that should be translated into German. Words are embedded as vectors with the same embedding size $s_e$.

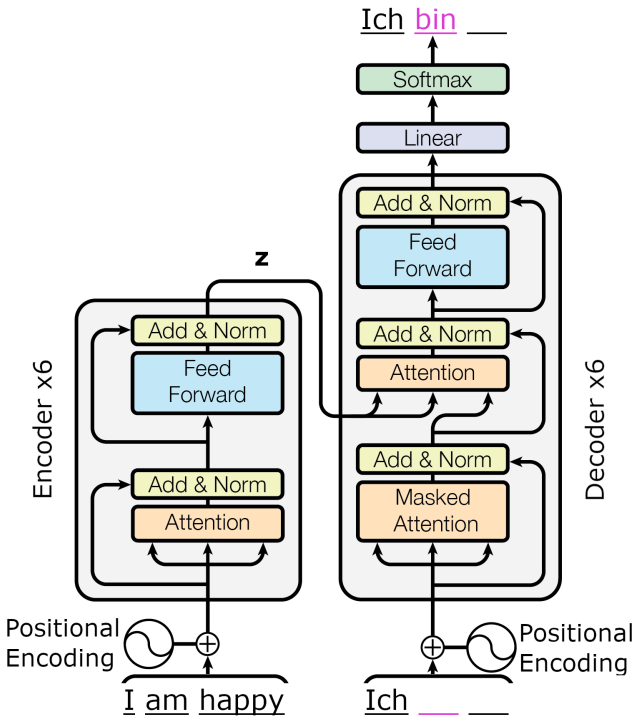


*Figure 1.* Traditional transformer with an encoder layer on the left and a decoder layer on the right. The inputs 'I am happy' and the previously generated token 'Ich' are used to generate the current token 'bin' at inference time. (Image source: Figure 1 in [38])

When translating 'I am happy' into 'Ich bin froh', the encoder takes all word embeddings of the input sentences to produce 3 latent embeddings $\mathbf{z}$ also of size $s_e$. At each timestep, the decoder generates a single token with the same $\mathbf{z}$ as input for every decoder layer. The first decoder layer also has all the previously generated tokens as input. At timestep 2 for example, the decoder takes 'Ich' that was generated at timestep 1 and $\mathbf{z}$ as input to generate the token 'bin' as illustrated in Figure 1.

In the first transformer, the encoder and decoder both contain

6 transformer layers. They themselves contain self-attention layers, a fully connected feed-forward network and residual connections followed by layer normalization [3]. The feed-forward layers have a single network with an input and output dimension of $s_e$ which is applied to every embedding.

**Attention** is the transformer's core component and enables us to attend differently to the input tokens for each generated token. The specific mechanism we use is 'scaled dot-product attention' which first turns the embeddings into queries $Q$, keys $K$ and values $V$ by multiplying them with three different weight matrices. We compare every query with every key to find the highly similar keys for each, meaning that they have a high dot-product. To calculate all those pairwise similarities we multiply the matrices $QK^T$. Every key has a corresponding value that is produced from the same embedding. For each query, our goal is to pay more attention to the values whose keys had high similarity with the query. We do so with softmax, which turns the similarity scores into a distribution of attention scores that is multiplied with the values.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (1)$$

With a higher dimension of the keys or queries $d_k$, the dot-products $QK^T$ are more likely to have very high or low values, meaning small gradients for the softmax. To avoid the small gradients, we divide the dot-product by the square root of $d_k$.

In the encoder, the attention layers always take the same embedding as input to calculate $Q$, $K$ and $V$ which is why it is called self-attention. In contrast, the second attention layer in the decoder receives its inputs from two sources. The keys and values with extracted knowledge about the input tokens come from the encoder. The queries come from the decoder and select keys with corresponding values such that we attend to the right values for generating the current token.

All the mentioned attention layers split the embeddings into 8 parts of size $s_e/8$ and for all 8 parts a set of queries, keys and values are calculated. These are used to calculate 8 different attention results that are concatenated and fed to a linear layer to bring the dimension back to the embedding size. We call this multi-head attention and the idea behind it is allowing our model to learn more attention patterns that give us the right focus for different inputs.

**Training and inference.** During inference, one token is generated at a time using all previously produced tokens as discussed in the translation of 'I am happy'. At training time all output tokens are already known and can be used as input to generate the whole output sentence at once. But for the generation of any token like 'bin', the decoder must not be able to look at the word 'bin' itself or any of the words to the right of it since they would not be known during inference. To prevent the decoder from cheating and looking at future tokens, we mask the self-attention such that for every word the decoder cannot see the word itself and all the following words. The unknown embeddings are ignored by setting the corresponding parts of the dot-product $Q \cdot K^T$ to $-\infty$ such that softmax produces no attention for those spots. This masked attention enables training for multiple tokens in parallel leading to much lower train times. By masking out future tokens during training we also obtain an autoregressive model, meaning that every prediction is only conditioned on the previous tokens.

To convert the output embeddings of the decoder to tokens in the first place, a last linear layer is used that increases the output dimension from the embedding size to the size of a specific dictionary. A softmax turns the large vectors to distributions and the token from the dictionary with the highest probability is chosen as the prediction. For training, the target tokens are one-hot encoded according to the dictionary and their cross-entropy loss with the generated probabilities are optimized via gradient descent.

**Positional encoding.** The transformer should be able to use the information about how the input tokens are ordered since it is valuable knowledge for a sequence. However, the model only uses scaled dot-product attention and fully connected layers which are both permutation invariant. This means no order and distance relations can be captured unlike for example with convolutions or recurrence. The solution is adding a positional encoding that is different depending on a token's position in the sequence. Vaswani's transformer uses a pre-calculated encoding but there are many valid possibilities for either learned or deterministic encodings [8; 14; 30].

## 3. Comparison with traditional Methods

Traditionally most tasks with sequential inputs were tackled with recurrent models (RNNs, LSTMs [20]), often with an encoder-decoder architecture for the common sequence to sequence tasks like machine translation [10]. The encoder RNN would generate a hidden state for each input token one after the other until the end of the input sequence. The last hidden state is used by the decoder RNN to produce an output token and a hidden state for the next token.

Recurrent models are naturally suited to process sequential data like language and transformers lose this valuable recurrence and have to use workarounds like the positional encoding to use information about the sequence order. So how can they outperform traditional models in so many tasks? The great advantage of transformers is the constant path length between the input tokens and the generated token. When translating a 20-word sentence, an RNN has to

remember the semantic of the first word through 20 hidden states until it can generate the first translated word. On this large path, information is lost making it hard to learn and understand long-range dependencies. The problem becomes more severe the longer the input sequences and for instance, if the last generated word strongly depends on the first word in the input. For a transformer, every generated token has the same short pathlength for attending to any input token independent of sequence length, making it easy to learn long-range dependencies. This is crucial for language where a word at the end of a sentence often depends on words at the beginning to define its meaning, just like the word 'its' in this very sentence.

Empirically transformers are more robust against adversarial perturbations than their RNN-based counterparts [21]. For the theoretical reasoning behind that fact, we think about changing a single token of the input to a self-attention layer. The influence on the output depends on the attention scores produced between all queries and the altered key because it determines the attention to the altered value. If the embeddings and are sufficiently spread out in space (not all of them are very similar) the key cannot produce large dot-products with all queries. The attention to the perturbed value is sparse and therefore the influence on the layer's output is sparse. In the recurrent setting, altering a single word changes all the following hidden states and it is easier for an adversarial attacker to impact all the embeddings.

## 4. Transformer Improvements

**Architecture Changes.** Transformers perform well as language models which can predict the next word in a sentence given the previous words such that they can generate text to complete a sentence or answer a question. Most language model transformers drop the encoder and only use the decoder [2; 27] training it to produce the next word of a sentence. Dropping the encoder means half as much computational requirements and even shorter pathlength between the inputs and the embeddings of the generated token.

Transformer-XL [13] is such an autoregressive language model that solves an additional problem. Commonly, transformers process around $n = 512$ tokens in parallel and the input sequence is chunked into this fixed length, disregarding natural boundaries like sentences. The traditional architecture cannot use context from previous segments when processing the current segment, leading to context-breaking in the generated output. The solution is remembering the hidden states of the previous segment and using the saved states of layer $l - 1$ as additional input for layer $l$ in the current segment. This way Transformer-XL can use context from all $L$ previous segments with $L$ as the number of layers. Longformer's [5] solution to context-breaking is making much larger segments with thousands of tokens possible by only calculating the attentions of all $n$ queries to a few keys in a local window around the query. This is comparable to how convolutional kernels in computer vision are slid over images instead of having a fully connected layer. The changed attention is linear in segment length $n$ instead of the standard quadratic self-attention where $n$ queries attend to $n$ keys.

**Unsupervised Training.** Using autoregressive text reproduction as a task to train transformers was made popular with the models GPT [32] and GPT-2 [33]. GPT-2 proved that language models that were trained entirely unsupervised can compete with supervised methods for a variety of tasks. The model solves and generalizes to these tasks in a zero-shot fashion, where the objective is stated implicitly as a text which the model has to complete. For a translation task, the input would be a few pairs of `English sentence = German sentence` and a final prompt of the form `English sentence =`. This makes autoregressive language models extremely flexible in their application.

BERT [14] uses a new bidirectional denoising pretraining approach. In autoregressively trained decoder models with masked attention, a token can only attend to the left of itself. In language, the context from the right also matters. BERT uses encoder layers which have no masking so that the attention goes both directions. Instead, the masking takes place in the pretraining task that randomly replaces $15\%$ of the input tokens with a special token. The model has to correctly generate the original tokens. The pretraining also involves next-sentence prediction where two sentences are used as the input with a separator token between them and a classification token appended at the beginning. For the classification token, BERT has to produce a prediction whether the two sentences followed each other in the text or not. The bidirectionality is natural for language and makes BERT perform better for tasks such as classification. However, the autoregressive property is lost which is inherent to the principle of language models (generating the next word depending on the previous ones). As such, BERT performs worse at text generation [40]. BERT finds a lot of use, is well investigated [22; 23] and a lot of work builds on it like RoBERTa [28] which further tweaks the pretraining tasks and its hyperparameters.

**Computational Requirements**. Transformer language models are often large and during training the attention mechanism needs to store $O(n^2)$ dot-products ($n$ is segment length). Reformer [24] uses attention with $O(n \cdot log(n))$ that assigns similar queries and keys to buckets and only calculates dot-products between keys and queries that are in the same bucket. As a second trick Reformer is made invertible by using reversible residual connections [17]. This means the activations of the forward pass don't have to be stored since they can be recalculated during backpropagation such

that only the activations of a single layer need to be saved simultaneously.

Analyzing the attention patterns of different attention heads shows that some only attend to small parts of the segment suggesting that for most the attention doesn't need to span the whole segment. A learned attention span for each head was proven very successful [36] and can be improved such that each head learns to distribute its attention over sparse slices of the segment [12].

Another research direction is reducing model sizes after training. Transformers can often be compressed substantially with simple methods. For instance, 30% of BERTs weights can be pruned by their magnitude without a significant loss in performance [18]. Especially attention heads are shown to often encode simple positional relations [34] and for most layers only a few heads are relevant meaning that the majority can be pruned [29; 39]. ALBERT [26] is an example that manages to compress BERT's size by 5 times while keeping the same performance.

## 5. Applications of Transformers

Transformers depend on a lot of training data and can capture long-range dependencies. In NLP long-range dependencies are relevant and large text corpora are available, which is why transformers shine in this field. The architecture also works well on many other types of sequential data like in sequence labeling for genomes [11]. The ActionTransformer [15] successfully annotates human actions in videos and for human trajectory prediction a transformer model is the current SOTA [16]. Spatio-temporal data is processed successfully by models where there are two self-attention mechanisms separately attending on the spatial and the temporal dimension [1; 41].

Transformers have shown success in processing images as a sequence of pixels. DEtectionTRansformer [7] performs object detection with an encoder that gathers information from the pixel sequence and a decoder that poses queries for embeddings of bounding boxes. It performs on par with other architectures that are much more complex.

In reinforcement learning, it is valuable to capture how actions influence rewards in the long term and transformers are good at that. Parisotto et al. [31] build on Transformer-XL's memory mechanism and replaces the residual connections with a gating method [9]. The performance is competitive with current LSTMs.

## 6. Transformer Limitations, Challenges and Future Directions

Are some tasks just inherently reliant on recurrence and not fit for transformers? It has been shown that self-attention is computationally restricted and cannot capture some properties that can occur in sequences like recursion [19]. There is empirical evidence that non-recurrent architectures perform worse at tasks where the data contains hierarchical structure [37]. This is relevant for the syntax of natural language which is a hierarchical structure of sentences, phrases and words. Encoding hierarchical structure in the embeddings could increase the performance similar to the positional encoding for capturing word order.

Transformer language models are large because in NLP new SOTAs are often reached by models with ever more parameters. From GPT (110 million) over BERT-Large (340 million) and GPT-2 (1.5 billion) up to GPT-3 [6] (175 billion). Implications are that they can't be trained on a single GPU and that the largest models with the best results are reserved to institutions with vast amounts of resources. Pretraining a single model on hundreds of GPUs for days is not uncommon [35; 42] which is expensive and consumes a lot of energy.

The growing model sizes raise another concern. For instance, GPT-3 claims to learn reasoning which the authors base on results for tasks like basic arithmetic while they can never convincingly prove that GPT-3's predictions were not just part of the training data but the outputs suspiciously look like that is the case. At what point does performance gain just come from having so many parameters that training data is memorized by the network? It would be interesting to use influence functions [25] to test what point of the training data influenced a specific model output the most. That way we learn how much transformers rely on simply reproducing its training data.

## 7. Conclusion

Transformers often replace recurrence based models for processing sequential data because of their ability to better learn long term dependencies and their parallel processing capabilities. At the core of the architecture is scaled dot-product attention which allows beneficial short path lengths but is quadratically expensive with respect to segment length. The most important application is in NLP. There has been a lot of success in pretraining very large transformers as language models that can then be finetuned for any NLP task or even multiple tasks in a flexible way. The limit of performance gain by increasing the model size is not found yet. This upscaling raises problems like cost, energy consumption or accessibility. As a result, lots of methods have been established to make transformers more efficient and smaller. Besides NLP, they are successful for many types of sequential data like video or audio but also for data that can be rewritten as a sequence like images as a string of pixels. The transformer is a young architecture with a lot of potential and active research.

## References


[1] Aksan, E., Cao, P., Kaufmann, M., and Hilliges, O. Attention, please: A spatio-temporal transformer for 3d human motion prediction. 2020. URL https://arxiv.org/abs/2004.08692.

[2] Al-Rfou, R., Choe, D., Constant, N., Guo, M., and Jones, L. Character-level language modeling with deeper self-attention. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 3159–3166. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33013159. URL https://doi.org/10.1609/aaai.v33i01.33013159.

[3] Ba, L. J., Kiros, J. R., and Hinton, G. E. Layer normalization. 2016. URL http://arxiv.org/abs/1607.06450.

[4] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

[5] Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. 2020. URL https://arxiv.org/abs/2004.05150.

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

[7] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. 2020. URL https://arxiv.org/abs/2005.12872.

[8] Chen, K., Wang, R., Utiyama, M., and Sumita, E. Recurrent positional embedding for neural machine translation. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 1361–1367. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1139. URL https://doi.org/10.18653/v1/D19-1139.

[9] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Moschitti, A., Pang, B., and Daelemans, W. (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1724–1734. ACL, 2014. doi: 10.3115/v1/d14-1179. URL https://doi.org/10.3115/v1/d14-1179.

[10] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Moschitti, A., Pang, B., and Daelemans, W. (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1724–1734. ACL, 2014. doi: 10.3115/v1/d14-1179. URL https://doi.org/10.3115/v1/d14-1179.

[11] Clauwaert, J. and Waegeman, W. Novel transformer networks for improved sequence labeling in genomics. *bioRxiv*, 2019. doi: 10.1101/836163. URL https://www.biorxiv.org/content/early/2019/11/13/836163.

[12] Correia, G. M., Niculae, V., and Martins, A. F. T. Adaptively sparse transformers. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 2174–2184. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1223. URL https://doi.org/10.18653/v1/D19-1223.

[13] Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. In Korhonen, A., Traum, D. R., and Màrquez, L. (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp.

2978–2988. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1285. URL https://doi.org/10.18653/v1/p19-1285.

[14] Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

[15] Girdhar, R., Carreira, J., Doersch, C., and Zisserman, A. Video action transformer network. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 244–253. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00033. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Girdhar_Video_Action_Transformer_Network_CVPR_2019_paper.html.

[16] Giuliari, F., Hasan, I., Cristani, M., and Galasso, F. Transformer networks for trajectory forecasting. 2020. URL https://arxiv.org/abs/2003.08111.

[17] Gomez, A. N., Ren, M., Urtasun, R., and Grosse, R. B. The reversible residual network: Backpropagation without storing activations. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 2214–2224, 2017. URL http://papers.nips.cc/paper/6816-the-reversible-residual-network-backpropagation-without-storing-activations.

[18] Gordon, M. A., Duh, K., and Andrews, N. Compressing BERT: studying the effects of weight pruning on transfer learning. 2020. URL https://arxiv.org/abs/2002.08307.

[19] Hahn, M. Theoretical limitations of self-attention in neural sequence models. *Trans. Assoc. Comput. Linguistics*, 8:156–171, 2020. URL https://transacl.org/ojs/index.php/tacl/article/view/1815.

[20] Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[21] Hsieh, Y., Cheng, M., Juan, D., Wei, W., Hsu, W., and Hsieh, C. On the robustness of self-attentive models. In Korhonen, A., Traum, D. R., and Màrquez, L. (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 1520–1529. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1147. URL https://doi.org/10.18653/v1/p19-1147.

[22] Jin, D., Jin, Z., Zhou, J. T., and Szolovits, P. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8018–8025. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6311.

[23] K, K., Wang, Z., Mayhew, S., and Roth, D. Cross-lingual ability of multilingual BERT: an empirical study. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJeT3yrtDr.

[24] Kitaev, N., Kaiser, L., and Levskaya, A. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

[25] Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894. PMLR, 2017. URL http://proceedings.mlr.press/v70/koh17a.html.

[26] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. ALBERT: A lite BERT for self-supervised learning of language representations. In

*8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

[27] Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. Generating wikipedia by summarizing long sequences. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Hyg0vbWC-.

[28] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. 2019. URL http://arxiv.org/abs/1907.11692.

[29] Michel, P., Levy, O., and Neubig, G. Are sixteen heads really better than one? In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pp. 14014–14024, 2019. URL http://papers.nips.cc/paper/9551-are-sixteen-heads-really-better-than-one.

[30] Omote, Y., Tamura, A., and Ninomiya, T. Dependency-based relative positional encoding for transformer NMT. In Mitkov, R. and Angelova, G. (eds.), *Proceedings of the International Conference on Recent Advances in Natural Language Processing, RANLP 2019, Varna, Bulgaria, September 2-4, 2019*, pp. 854–861. INCOMA Ltd., 2019. doi: 10.26615/978-954-452-056-4_099. URL https://doi.org/10.26615/978-954-452-056-4_099.

[31] Parisotto, E., Song, H. F., Rae, J. W., Pascanu, R., Gülçehre, Ç., Jayakumar, S. M., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S., Botvinick, M. M., Heess, N., and Hadsell, R. Stabilizing transformers for reinforcement learning. 2019. URL http://arxiv.org/abs/1910.06764.

[32] Radford, A. Improving language understanding by generative pre-training. 2018.

[33] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

[34] Raganato, A., Scherrer, Y., and Tiedemann, J. Fixed encoder self-attention patterns in transformer-based machine translation. 2020. URL https://arxiv.org/abs/2002.10260.

[35] Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. Megatron-lm: Training multi-billion parameter language models using model parallelism. 2019. URL http://arxiv.org/abs/1909.08053.

[36] Sukhbaatar, S., Grave, E., Bojanowski, P., and Joulin, A. Adaptive attention span in transformers. In Korhonen, A., Traum, D. R., and Màrquez, L. (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 331–335. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1032. URL https://doi.org/10.18653/v1/p19-1032.

[37] Tran, K. M., Bisazza, A., and Monz, C. The importance of being recurrent for modeling hierarchical structure. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 4731–4736. Association for Computational Linguistics, 2018. URL https://www.aclweb.org/anthology/D18-1503/.

[38] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.

[39] Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In Korhonen, A., Traum, D. R., and Màrquez, L. (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 5797–5808. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1580. URL https://doi.org/10.18653/v1/p19-1580.

[40] Wang, A. and Cho, K. BERT has a mouth, and it must speak: BERT as a markov random field language

model. 2019. URL http://arxiv.org/abs/1902.04094.

[41] Xu, M., Dai, W., Liu, C., Gao, X., Lin, W., Qi, G., and Xiong, H. Spatial-temporal transformer networks for traffic flow forecasting. 2020. URL http://arxiv.org/abs/2001.02908.

[42] Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pp. 5754–5764, 2019. URL http://papers.nips.cc/paper/8812-xlnet-generalized-autoregressive-pretraining-for-language-understanding.

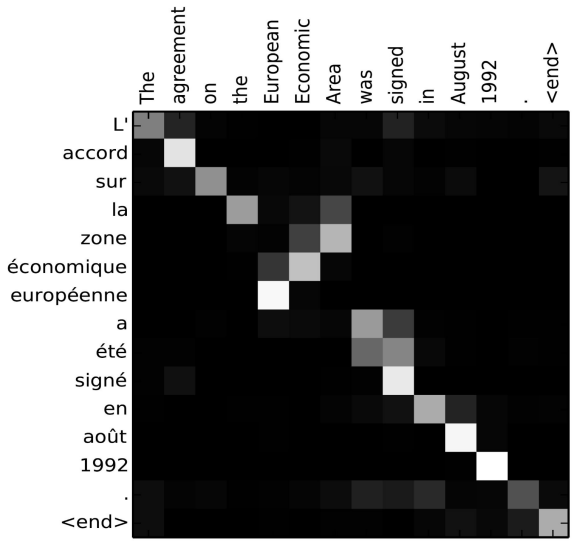


*Figure 2.* Attention layers can be visualized in a simple way. Each pixel shows the softmax result of the queries and keys corresponding to the tokens on the y- and x-axis. (Image source: Figure 3 in [4])

## A. Positional Encoding

One obvious positional encoding would be to directly use a tokens position. This would have the disadvantage of large values added to the token embedding. The model would also have no understanding for positional embeddings that were not encountered during training. Vaswani's transformer calculates the encoding for a token at position $p$ using the highs and lows of different frequency sinusoids depending on the different indices $i$ of a tokens embedding. For the first indices the position $p$ is fed to higher frequency sinusoids and as $i$ increases we use lower frequencies (until $i$ reaches the embedding size $s_e$). Intuitively this is the continuous version of the highs and lows in a binary encoding.

$$Encoding_i(p) = \begin{cases} sin(p/10000^{2i/s_e}) & \text{if i is even} \\ cos(p/10000^{2i/s_e}) & \text{if i is odd} \end{cases} \quad (2)$$