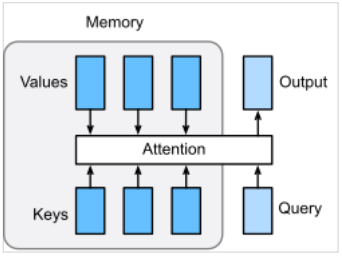
Attention (machine learning)

In machine learning, **attention** is a method that determines the importance of each component in a sequence relative to the other components in that sequence. In natural language processing, importance is represented by "soft" weights assigned to each word in a sentence. More generally, attention encodes vectors called token embeddings across a fixed-width sequence that can range from tens to millions of tokens in size.



Attention mechanism, overview

Unlike "hard" weights, which are computed during the backwards training pass, "soft" weights exist only in the forward pass and therefore change with every step of the input. Earlier designs implemented the attention mechanism in a serial recurrent neural network (RNN) language translation system, but a more recent design, namely the transformer, removed the slower sequential RNN and relied more heavily on the faster parallel attention scheme.

Inspired by ideas about attention in humans, the attention mechanism was developed to address the weaknesses of using information from the hidden layers of recurrent neural networks. Recurrent neural networks favor more recent information contained in words at the end of a sentence, while information earlier in the sentence tends to be attenuated. Attention allows a token equal access to any part of a sentence directly, rather than only through the previous state.

History

1950s–1960s	Psychology and biology of attention. Cocktail party effect[1] — focusing on content by filtering out background noise. Filter model of attention,[2] partial report paradigm, and saccade control.[3]
1980s	Sigma-pi units,[4] higher-order neural networks.
1990s	*Fast weight controllers* and dynamic links between neurons, anticipating key-value mechanisms in attention.[5][6][7][8]
1998	The bilateral filter was introduced in image processing. It uses pairwise affinity matrices to propagate relevance across elements.[9]
2005	Non-local means extended affinity-based filtering in image denoising, using Gaussian similarity kernels as fixed attention-like weights.[10]
2014	seq2seq with RNN + Attention.[11] Attention was introduced to enhance RNN encoder-decoder translation, particularly for long sentences. See Overview section. Attentional Neural Networks introduced a learned feature selection mechanism using top-down cognitive modulation, showing how attention weights can highlight relevant inputs.[12]
2015	Attention was extended to vision for image captioning tasks.[13][14]
2016	Self-attention was integrated into RNN-based models to capture intra-sequence dependencies.[15][16] Self-attention was explored in decomposable attention models for natural language inference[17] and structured self-attentive sentence embeddings.[18]
2017	The Transformer architecture introduced in the research paper Attention is All You Need[19] formalized scaled dot-product self-attention: $$A = \mathbf{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$ Relation networks[20] and set Transformers[21] applied attention to unordered sets and relational reasoning, generalizing pairwise interaction models.
2018	Non-local neural networks[22] extended attention to computer vision by capturing long-range dependencies in space and time. Graph attention networks[23] applied attention mechanisms to graph-structured data.
2019–2020	Efficient Transformers, including Reformer,[24] Linformer,[25] and Performer,[26] introduced scalable approximations of attention for long sequences.
2019+	Hopfield networks were reinterpreted as associative memory-based attention systems,[27] and vision transformers (ViTs) achieved competitive results in image classification.[28] Transformers were adopted across scientific domains, including AlphaFold for protein folding,[29] CLIP for vision-language pretraining,[30] and attention-based dense segmentation models like CCNet[31] and DANet.[32]

Additional surveys of the attention mechanism in deep learning are provided by Niu et al.[33] and Soydaner.[34]

The major breakthrough came with self-attention, where each element in the input sequence attends to all others, enabling the model to capture global dependencies. This idea was central to the Transformer architecture, which replaced recurrence with attention mechanisms. As a result, Transformers became the foundation for models like BERT, T5 and generative pre-trained transformers (GPT).[19]

Overview

The modern era of machine attention was revitalized by grafting an attention mechanism (Fig 1. orange) to an Encoder-Decoder.

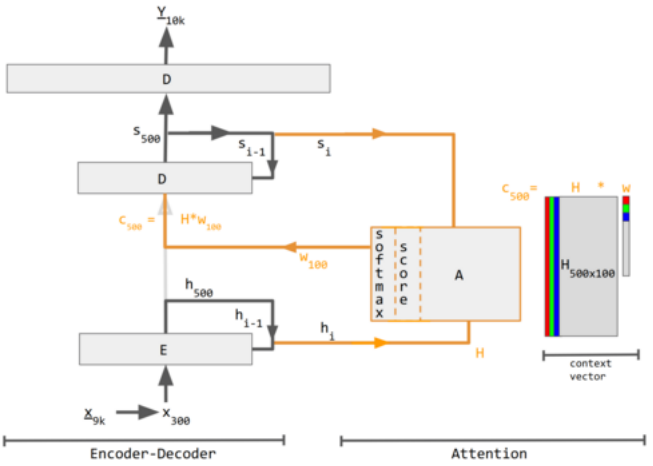


Fig 1. Encoder-decoder with attention.[35] Numerical subscripts (100, 300, 500, 9k, 10k) indicate vector sizes while lettered subscripts i and i − 1 indicate time steps. Pinkish regions in H matrix and w vector are zero values. See Legend for details.

Animated sequence of language translation

Legend

Label	Description
100	Max. sentence length
300	Embedding size (word dimension)
500	Length of hidden vector
9k, 10k	Dictionary size of input & output languages respectively.
x, Y	9k and 10k 1-hot dictionary vectors. x → x implemented as a lookup table rather than vector multiplication. Y is the 1-hot maximizer of the linear Decoder layer D; that is, it takes the argmax of D's linear layer output.
x	300-long word embedding vector. The vectors are usually pre-calculated from other projects such as GloVe or Word2Vec.
h	500-long encoder hidden vector. At each point in time, this vector summarizes all the preceding words before it. The final h can be viewed as a "sentence" vector, or a thought vector as Hinton calls it.
s	500-long decoder hidden state vector.
E	500 neuron recurrent neural network encoder. 500 outputs. Input count is 800–300 from source embedding + 500 from recurrent connections. The encoder feeds directly into the decoder only to initialize it, but not thereafter; hence, that direct connection is shown very faintly.
D	2-layer decoder. The recurrent layer has 500 neurons and the fully-connected linear layer has 10k neurons (the size of the target vocabulary).[36] The linear layer alone has 5 million (500 × 10k) weights – ~10 times more weights than the recurrent layer.
score	100-long alignment score
w	100-long vector attention weight. These are "soft" weights which changes during the forward pass, in contrast to "hard" neuronal weights that change during the learning phase.
A	Attention module – this can be a dot product of recurrent states, or the query-key-value fully-connected layers. The output is a 100-long vector w.
H	500×100. 100 hidden vectors h concatenated into a matrix
c	500-long context vector = H * w. c is a linear combination of h vectors weighted by w.

Figure 2 shows the internal step-by-step operation of the attention block (A) in Fig 1.

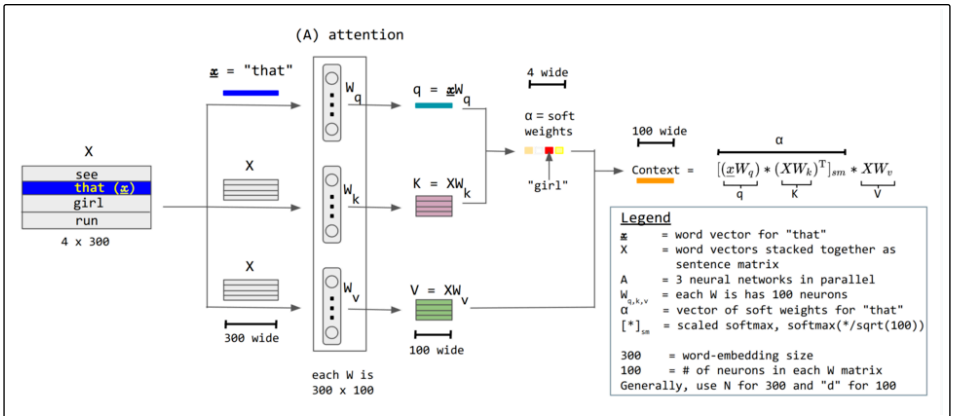


Figure 2. The diagram shows the attention forward pass calculating correlations of the word "that" with other words in "See that girl run." Given the right weights from training, the network should be able to identify "girl" as a highly correlated word. Some things to note:

- This example focuses on the attention of a single word "that". In practice, the attention of each word is calculated in parallel to speed up calculations. Simply changing the lowercase "x" vector to the uppercase "X" matrix will yield the formula for this.
- Softmax scaling $qW_k^T / \sqrt{100}$ prevents a high variance in qW_k^T that would allow a single word to excessively dominate the softmax resulting in attention to only one word, as a discrete hard max would do.
- Notation: the commonly written row-wise softmax formula above assumes that vectors are rows, which runs contrary to the standard math notation of column vectors. More correctly, we should take the transpose of the context vector and use the column-wise softmax, resulting in the more correct form

$$(XW_v)^T * [(W_k X^T) * (\underline{x}W_q)^T]_{sm}$$

.

Interpreting attention weights

In translating between languages, alignment is the process of matching words from the source sentence to words of the translated sentence. Networks that perform verbatim translation without regard to word order would show the highest scores along the (dominant) diagonal of the matrix. The off-diagonal dominance shows that the attention mechanism is more nuanced.

Consider an example of translating *I love you* to French. On the first pass through the decoder, 94% of the attention weight is on the first English word *I*, so the network offers the word *je*. On the second pass of the decoder, 88% of the attention weight is on the third English word *you*, so it offers *t'*. On the last pass, 95% of the attention weight is on the second English word *love*, so it offers *aime*.

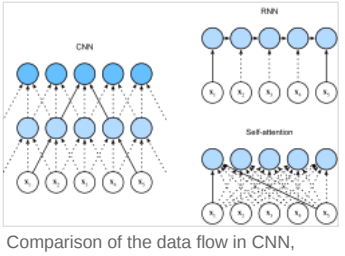
In the *I love you* example, the second word *love* is aligned with the third word *aime*. Stacking soft row vectors together for *je*, *t'*, and *aime* yields an alignment matrix:

	I	love	you
je	0.94	0.02	0.04
t'	0.11	0.01	0.88
aime	0.03	0.95	0.02

Sometimes, alignment can be multiple-to-multiple. For example, the English phrase *look it up* corresponds to *cherchez-le*. Thus, "soft" attention weights work better than "hard" attention weights (setting one attention weight to 1, and the others to 0), as we would like the model to make a context vector consisting of a weighted sum of the hidden vectors, rather than "the best one", as there may not be a best hidden vector.

Variants

Many variants of attention implement soft weights, such as

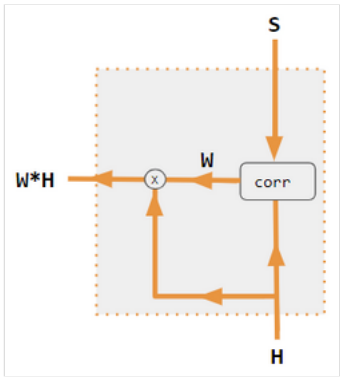
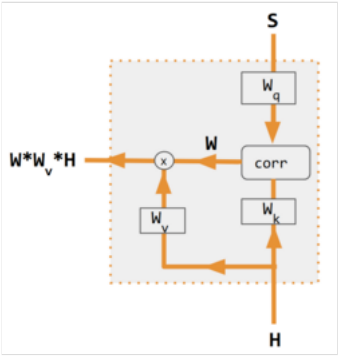
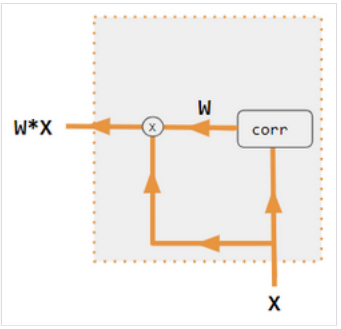
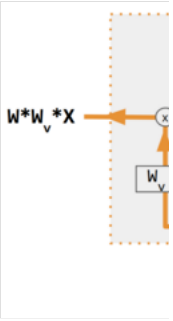


Comparison of the data flow in CNN, RNN, and self-attention

- fast weight programmers, or fast weight controllers (1992).[5] A "slow" neural network outputs the "fast" weights of another neural network through outer products. The slow network learns by gradient descent. It was later renamed as "linearized self-attention".[37]
- Bahdanau-style attention,[11] also referred to as *additive attention*,
- Luong-style attention,[38] which is known as *multiplicative attention*,
- Early attention mechanisms similar to modern self-attention were proposed using recurrent neural networks. However, the highly parallelizable self-attention was introduced in 2017 and successfully used in the Transformer model,
- *positional attention* and *factorized positional attention*.[39]

For convolutional neural networks, attention mechanisms can be distinguished by the dimension on which they operate, namely: spatial attention,[40] channel attention,[41] or combinations.[42][43]

These variants recombine the encoder-side inputs to redistribute those effects to each target output. Often, a correlation-style matrix of dot products provides the re-weighting coefficients. In the figures below, W is the matrix of context attention weights, similar to the formula in Overview section above.

1. encoder-decoder dot product	2. encoder-decoder QKV	3. encoder-only dot product	4. encoder-c
 Both encoder & decoder are needed to calculate attention.[38]	 Both encoder & decoder are needed to calculate attention.[44]	 Decoder is *not* used to calculate attention. With only 1 input into corr, W is an auto-correlation of dot products. $w_{ij} = x_i x_j$.[45]	 Decoder is *not* used t attention.[46]

Legend

Label	Description
Variables X, H, S, T	Upper case variables represent the entire sentence, and not just the current word. For example, H is a matrix of the encoder hidden state—one word per column.
S, T	S, decoder hidden state; T, target word embedding. In the Pytorch Tutorial variant training phase, T alternates between 2 sources depending on the level of teacher forcing used. T could be the embedding of the network's output word; i.e. embedding(argmax(FC output)). Alternatively with teacher forcing, T could be the embedding of the known correct word which can occur with a constant forcing probability, say 1/2.
X, H	H, encoder hidden state; X, input word embeddings.
W	Attention coefficients
Qw, Kw, Vw, FC	Weight matrices for query, key, value respectively. FC is a fully-connected weight matrix.
⊕, ⊗	⊕, vector concatenation; ⊗, matrix multiplication.
corr	Column-wise softmax(matrix of all combinations of dot products). The dot products are $\mathbf{x_i} * \mathbf{x_j}$ in variant #3, $\mathbf{h_i} * \mathbf{s_j}$ in variant 1, and column $_i$ (Kw * H) * column $_j$ (Qw * S) in variant 2, and column $_i$ (Kw * X) * column $_j$ (Qw * X) in variant 4. Variant 5 uses a fully-connected layer to determine the coefficients. If the variant is QKV, then the dot products are normalized by the $\sqrt{d}$ where d is the height of the QKV matrices.

Optimizations

Flash attention

The size of the attention matrix is proportional to the square of the number of input tokens. Therefore, when the input is long, calculating the attention matrix requires a lot of GPU memory. Flash attention is an implementation that reduces the memory needs and increases efficiency without sacrificing accuracy. It achieves this by partitioning the attention computation into smaller blocks that fit into the GPU's faster on-chip memory, reducing the need to store large intermediate matrices and thus lowering memory usage while increasing computational efficiency.[48]

FlexAttention

FlexAttention[49] is an attention kernel developed by Meta that allows users to modify attention scores prior to softmax and dynamically chooses the optimal attention algorithm.

Applications

Attention is widely used in natural language processing, computer vision, and speech recognition. In NLP, it improves context understanding in tasks like question answering and summarization. In vision, visual attention helps models focus on relevant image regions, enhancing object detection and image captioning.

Attention maps as explanations for vision transformers

From the original paper on vision transformers (ViT), visualizing attention scores as a heat map (called saliency maps or attention maps) has become an important and routine way to inspect the decision making process of ViT models.[50] One can compute the attention maps with respect to any attention head at any layer, while the deeper layers tend to show more semantically meaningful visualization. Attention rollout is a recursive algorithm to combine attention scores across all layers, by computing the dot product of successive attention maps.[51]

Because vision transformers are typically trained in a self-supervised manner, attention maps are generally not class-sensitive. When a classification head attached to the ViT backbone, class-discriminative attention maps (CDAM) combines attention maps and gradients with respect to the class `[CLS]` token.[52] Some class-sensitive interpretability methods originally developed for convolutional neural networks can be also applied to ViT, such as GradCAM, which back-propagates the gradients to the outputs of the final attention layer.[53]

Using attention as basis of explanation for the transformers in language and vision is not without debate. While some pioneering papers analyzed and framed attention scores as explanations,[54][55] higher attention scores do not always correlate with greater impact on model performances.[56]

Mathematical representation

Standard scaled dot-product attention

For matrices: $Q \in \mathbb{R}^{m \times d_k}, K \in \mathbb{R}^{n \times d_k}$ and $V \in \mathbb{R}^{n \times d_v}$, the scaled dot-product, or QKV attention, is defined as:

$$\mathbf{Attention}(Q, K, V) = \mathbf{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \in \mathbb{R}^{m \times d_v}$$

where ^{T} denotes transpose and the softmax function is applied independently to every row of its argument. The matrix Q contains m queries, while matrices K, V jointly contain an *unordered* set of n key-value pairs. Value vectors in matrix V are weighted using the weights resulting from the softmax operation, so that the rows of the m-by-d_v output matrix are confined to the convex hull of the points in $\mathbb{R}^{d_v}$ given by the rows of V.

To understand the permutation invariance and permutation equivariance properties of QKV attention,[57] let $A \in \mathbb{R}^{m \times m}$ and $B \in \mathbb{R}^{n \times n}$ be permutation matrices; and $D \in \mathbb{R}^{m \times n}$ an arbitrary matrix. The softmax function is permutation equivariant in the sense that:

$$\mathbf{softmax}(ADB) = A\,\mathbf{softmax}(D)B$$

By noting that the transpose of a permutation matrix is also its inverse, it follows that:

$$\mathbf{Attention}(AQ, BK, BV) = A\,\mathbf{Attention}(Q, K, V)$$

which shows that QKV attention is equivariant with respect to re-ordering the queries (rows of Q); and invariant to re-ordering of the key-value pairs in K, V. These properties are inherited when applying linear transforms to the inputs and outputs of QKV attention blocks. For example, a simple self-attention function defined as:

$$X \mapsto \mathbf{Attention}(XT_q, XT_k, XT_v)$$

is permutation equivariant with respect to re-ordering the rows of the input matrix X in a non-trivial way, because every row of the output is a function of all the rows of the input. Similar properties hold for *multi-head attention*, which is defined below.

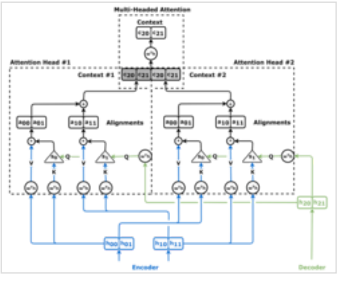
Masked attention

When QKV attention is used as a building block for an autoregressive decoder, and when at training time all input and output matrices have n rows, a masked attention variant is used:

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

where the mask, $M \in \mathbb{R}^{n \times n}$ is a strictly upper triangular matrix, with zeros on and below the diagonal and $-\infty$ in every element above the diagonal. The softmax output, also in $\mathbb{R}^{n \times n}$ is then *lower triangular*, with zeros in all elements above the diagonal. The masking ensures that for all $1 \leq i < j \leq n$, row i of the attention output is independent of row j of any of the three input matrices. The permutation invariance and equivariance properties of standard QKV attention do not hold for the masked variant.

Multi-head attention



Decoder multiheaded cross-attention

Multi-head attention

$$\mathbf{MultiHead}(Q, K, V) = \mathbf{Concat}(\mathbf{head}_1, \ldots, \mathbf{head}_h)W^O$$

where each head is computed with QKV attention as:

$$\mathbf{head}_i = \mathbf{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

and W_i^Q, W_i^K, W_i^V, and W^O are parameter matrices.

The permutation properties of (standard, unmasked) QKV attention apply here also. For permutation matrices, A, B :

$$\mathbf{MultiHead}(AQ, BK, BV) = A\,\mathbf{MultiHead}(Q, K, V)$$

from which we also see that multi-head self-attention:

$$X \mapsto \mathbf{MultiHead}(XT_q, XT_k, XT_v)$$

is equivariant with respect to re-ordering of the rows of input matrix X.

Bahdanau (additive) attention

$$\textbf{Attention}(Q, K, V) = \textbf{softmax}(\tanh(W_Q Q + W_K K))V$$

where W_Q and W_K are learnable weight matrices.[11]

Luong attention (general)

$$\textbf{Attention}(Q, K, V) = \textbf{softmax}(QWK^T)V$$

where W is a learnable weight matrix.[38]

Self-attention

Self-attention is essentially the same as cross-attention, except that query, key, and value vectors all come from the same model. Both encoder and decoder can use self-attention, but with subtle differences.

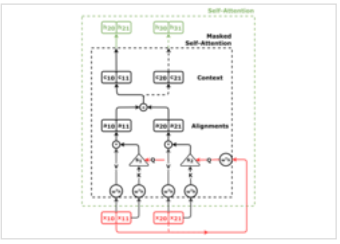
For encoder self-attention, we can start with a simple encoder without self-attention, such as an "embedding layer", which simply converts each input word into a vector by a fixed lookup table. This gives a sequence of hidden vectors $h_0, h_1, \dots$. These can then be applied to a dot-product attention mechanism, to obtain

$$h'_0 = \textbf{Attention}(h_0 W^Q, HW^K, HW^V)$$
$$h'_1 = \textbf{Attention}(h_1 W^Q, HW^K, HW^V)$$
$$\dots$$

or more succinctly, $H' = \textbf{Attention}(HW^Q, HW^K, HW^V)$. This can be applied repeatedly, to obtain a multilayered encoder. This is the "encoder self-attention", sometimes called the "all-to-all attention", as the vector at every position can attend to every other.

Masking

For decoder self-attention, all-to-all attention is inappropriate, because during the autoregressive decoding process, the decoder cannot attend to future outputs that has yet to be decoded. This can be solved by forcing the attention weights $w_{ij} = 0$ for all $i < j$, called "causal masking". This attention mechanism is the "causally masked self-attention".



Decoder self-attention with causal masking, detailed diagram

See also

- Recurrent neural network
- seq2seq
- Transformer (deep learning architecture)
- Attention
- Dynamic neural network

References

1. Cherry, E. Colin (1953). "Some Experiments on the Recognition of Speech, with One and with Two Ears". *The Journal of the Acoustical Society of America*. **25** (5): 975–979. Bibcode:1953ASAJ...25..975C (https://ui.adsabs.harvard.edu/abs/1953ASAJ...25..975C). doi:10.1121/1.1907229 (https://doi.org/10.1121%2F1.1907229). hdl:11858/00-001M-0000-002A-F750-3 (https://hdl.handle.net/11858%2F00-001M-0000-002A-F750-3).
2. Broadbent, Donald E. (1958). *Perception and Communication*. Pergamon Press.
3. Kowler, Eileen (1995). "The control of saccadic eye movements". *Reviews of Oculomotor Research*. **5**: 1–70.
4. Rumelhart, David E.; Hinton, G. E.; Mcclelland, James L. (1987-07-29). "A General Framework for Parallel Distributed Processing" (https://stanford.edu/~jlmcc/papers/PDP/Chapter2.pdf) (PDF). In Rumelhart, David E.; Hinton, G. E.; PDP Research Group (eds.). *Parallel Distributed Processing, Volume 1: Explorations in the Microstructure of Cognition: Foundations*. Cambridge, Massachusetts: MIT Press. ISBN 978-0-262-68053-0.
5. Schmidhuber, Jürgen (1992). "Learning to control fast-weight memories: an alternative to recurrent nets". *Neural Computation*. **4** (1): 131–139. doi:10.1162/neco.1992.4.1.131 (https://doi.org/10.1162%2Fneco.1992.4.1.131). S2CID 16683347 (https://api.semanticscholar.org/CorpusID:16683347).
6. von der Malsburg, Christoph (1981). "The correlation theory of brain function". *Internal Report 81–2, Max-Planck-Institute for Biophysical Chemistry*.
7. Feldman, Jerome A. (1982). "Dynamic connections in neural networks". *Biological Cybernetics*. **46** (1): 27–39. doi:10.1007/BF00335349 (https://doi.org/10.1007%2FBF00335349). PMID 6307398 (https://pubmed.ncbi.nlm.nih.gov/6307398).
8. Hinton, Geoffrey E. (1989). "Connectionist learning procedures". *Artificial Intelligence*. **40** (1–3): 185–234. doi:10.1016/0004-3702(89)90049-0 (https://doi.org/10.1016%2F0004-3702%2889%2990049-0).
9. Tomasi, Carlo (1998). *Bilateral filtering for gray and color images* (https://ieeexplore.ieee.org/document/710815). ICCV.
10. Buades, Antoni (2005). *A non-local algorithm for image denoising* (https://ieeexplore.ieee.org/document/1467423). CVPR.
11. Bahdanau, Dzmitry; Cho, Kyunghyun; Bengio, Yoshua (2014). "Neural Machine Translation by Jointly Learning to Align and Translate". arXiv:1409.0473 (https://arxiv.org/abs/1409.0473) [cs.CL (https://arxiv.org/archive/cs.CL)].
12. Wang, Qian (2014). *Attentional Neural Network: Feature Selection Using Cognitive Feedback*. NeurIPS.
13. Xu, Kelvin; Ba, Jimmy; Kiros, Ryan (2015). *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*. arXiv:1502.03044 (https://arxiv.org/abs/1502.03044).
14. Vinyals, Oriol; Toshev, Alexander; Bengio, Samy; Erhan, Dumitru (2015). "Show and Tell: A Neural Image Caption Generator". *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 3156–3164. doi:10.1109/CVPR.2015.7298935 (https://doi.org/10.1109%2FCVPR.2015.7298935). ISBN 978-1-4673-6964-0.
15. Cheng, Jianpeng (2016). "Long Short-Term Memory-Networks for Machine Reading". arXiv:1601.06733 (https://arxiv.org/abs/1601.06733) [cs.CL (https://arxiv.org/archive/cs.CL)].
16. Paulus, Romain (2017). "A Deep Reinforced Model for Abstractive Summarization". arXiv:1705.04304 (https://arxiv.org/abs/1705.04304) [cs.CL (https://arxiv.org/archive/cs.CL)].

17. Parikh, Anees (2016). *Decomposable Attention Model for Natural Language Inference*. EMNLP. arXiv:1606.01933 (https://arxiv.org/abs/1606.01933).
18. Lin, Zichao (2017). *A Structured Self-Attentive Sentence Embedding*. ICLR. arXiv:1703.03130 (https://arxiv.org/abs/1703.03130).
19. Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N.; Kaiser, Lukasz; Polosukhin, Illia (2017). "Attention is All You Need". arXiv:1706.03762 (https://arxiv.org/abs/1706.03762) [cs.CL (https://arxiv.org/archive/cs.CL)].
20. Santoro, Adam (2017). *Relation Networks for Relational Reasoning*. ICLR. arXiv:1706.01427 (https://arxiv.org/abs/1706.01427).
21. Lee, Juho (2019). *Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks*. ICML. arXiv:1810.00825 (https://arxiv.org/abs/1810.00825).
22. Wang, Xiaolong (2018). *Non-Local Neural Networks*. CVPR.
23. Veličković, Petar (2018). *Graph Attention Networks*. ICLR.
24. Kitaev, Nikita (2020). *Reformer: The Efficient Transformer*. ICLR. arXiv:2001.04451 (https://arxiv.org/abs/2001.04451).
25. Wang, Salah (2020). *Linformer: Self-Attention with Linear Complexity*. ICLR. arXiv:2006.04768 (https://arxiv.org/abs/2006.04768).
26. Choromanski, Krzysztof (2020). *Rethinking Attention with Performers*. ICLR. arXiv:2009.14794 (https://arxiv.org/abs/2009.14794).
27. Ramsauer, Johannes (2021). *Hopfield Networks is All You Need*. NeurIPS. arXiv:2008.02217 (https://arxiv.org/abs/2008.02217).
28. Dosovitskiy, Aleksander (2021). *An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale*. ICLR. arXiv:2010.11929 (https://arxiv.org/abs/2010.11929).
29. Jumper, John (2021). "Highly accurate protein structure prediction with AlphaFold" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8371605). *Nature*. **596** (7873): 583–589. Bibcode:2021Natur.596..583J (https://ui.adsabs.harvard.edu/abs/2021Natur.596..583J). doi:10.1038/s41586-021-03819-2 (https://doi.org/10.1038%2Fs41586-021-03819-2). PMC 8371605 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8371605). PMID 34265844 (https://pubmed.ncbi.nlm.nih.gov/34265844).
30. Radford, Alec (2021). *Learning Transferable Visual Models from Natural Language Supervision*. ICML.
31. Huang, Xiangyu (2019). *CCNet: Criss-Cross Attention for Semantic Segmentation*. ICCV. arXiv:1811.11721 (https://arxiv.org/abs/1811.11721).
32. Fu, Jing (2019). *Dual Attention Network for Scene Segmentation*. CVPR. arXiv:1809.02983 (https://arxiv.org/abs/1809.02983).
33. Niu, Zhaoyang; Zhong, Guoqiang; Yu, Hui (2021-09-10). "A review on the attention mechanism of deep learning" (https://www.sciencedirect.com/science/article/pii/S092523122100477X). *Neurocomputing*. **452**: 48–62. doi:10.1016/j.neucom.2021.03.091 (https://doi.org/10.1016%2Fj.neucom.2021.03.091). ISSN 0925-2312 (https://search.worldcat.org/issn/0925-2312).
34. Soydaner, Derya (August 2022). "Attention mechanism in neural networks: where it comes and where it goes" (https://link.springer.com/10.1007/s00521-022-07366-3). *Neural Computing and Applications*. **34** (16): 13371–13385. arXiv:2204.13154 (https://arxiv.org/abs/2204.13154). doi:10.1007/s00521-022-07366-3 (https://doi.org/10.1007%2Fs00521-022-07366-3). ISSN 0941-0643 (https://search.worldcat.org/issn/0941-0643).
35. Britz, Denny; Goldie, Anna; Luong, Minh-Thanh; Le, Quoc (2017-03-21). "Massive Exploration of Neural Machine Translation Architectures". arXiv:1703.03906 (https://arxiv.org/abs/1703.03906) [cs.CV (https://arxiv.org/archive/cs.CV)].
36. "Pytorch.org seq2seq tutorial" (https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html). Retrieved December 2, 2021.
37. Schlag, Imanol; Irie, Kazuki; Schmidhuber, Jürgen (2021). "Linear Transformers Are Secretly Fast Weight Programmers". *ICML 2021*. Springer. pp. 9355–9366.
38. Luong, Minh-Thang (2015-09-20). "Effective Approaches to Attention-Based Neural Machine Translation". arXiv:1508.04025v5 (https://arxiv.org/abs/1508.04025v5) [cs.CL (https://arxiv.org/archive/cs.CL)].
39. Luo, Fan; Zhang, Juan; Xu, Shenghui (3 July 2024). "Learning Positional Attention for Sequential Recommendation" (https://www.catalyzex.com/paper/learning-positional-attention-for-sequential). *catalyzex.com*.
40. Zhu, Xizhou; Cheng, Dazhi; Zhang, Zheng; Lin, Stephen; Dai, Jifeng (2019). "An Empirical Study of Spatial Attention Mechanisms in Deep Networks". *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. pp. 6687–6696. arXiv:1904.05873 (https://arxiv.org/abs/1904.05873). doi:10.1109/ICCV.2019.00679 (https://doi.org/10.1109%2FICCV.2019.00679). ISBN 978-1-7281-4803-8. S2CID 118673006 (https://api.semanticscholar.org/CorpusID:118673006).
41. Hu, Jie; Shen, Li; Sun, Gang (2018). "Squeeze-and-Excitation Networks". *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 7132–7141. arXiv:1709.01507 (https://arxiv.org/abs/1709.01507). doi:10.1109/CVPR.2018.00745 (https://doi.org/10.1109%2FCVPR.2018.00745). ISBN 978-1-5386-6420-9. S2CID 206597034 (https://api.semanticscholar.org/CorpusID:206597034).
42. Woo, Sanghyun; Park, Jongchan; Lee, Joon-Young; Kweon, In So (2018-07-18). "CBAM: Convolutional Block Attention Module". arXiv:1807.06521 (https://arxiv.org/abs/1807.06521) [cs.CV (https://arxiv.org/archive/cs.CV)].
43. Georgescu, Mariana-Iuliana; Ionescu, Radu Tudor; Miron, Andreea-Iuliana; Savencu, Olivian; Ristea, Nicolae-Catalin; Verga, Nicolae; Khan, Fahad Shahbaz (2022-10-12). "Multimodal Multi-Head Convolutional Attention with Various Kernel Sizes for Medical Image Super-Resolution". arXiv:2204.04218 (https://arxiv.org/abs/2204.04218) [eess.IV (https://arxiv.org/archive/eess.IV)].
44. Neil Rhodes (2021). *CS 152 NN—27: Attention: Keys, Queries, & Values* (https://www.youtube.com/watch?v=rA28vBqN4RM). Event occurs at 06:30. Retrieved 2021-12-22.
45. Alfredo Canziani & Yann Lecun (2021). *NYU Deep Learning course, Spring 2020* (https://www.youtube.com/watch?v=f01J0Dri-6k). Event occurs at 05:30. Retrieved 2021-12-22.
46. Alfredo Canziani & Yann Lecun (2021). *NYU Deep Learning course, Spring 2020* (https://www.youtube.com/watch?v=f01J0Dri-6k). Event occurs at 20:15. Retrieved 2021-12-22.
47. Robertson, Sean. "NLP From Scratch: Translation With a Sequence To Sequence Network and Attention" (https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html). *pytorch.org*. Retrieved 2021-12-22.
48. Mittal, Aayush (2024-07-17). "Flash Attention: Revolutionizing Transformer Efficiency" (https://www.unite.ai/flash-attention-revolutionizing-transformer-efficiency/). *Unite.AI*. Retrieved 2024-11-16.
49. "FlexAttention: The Flexibility of PyTorch with the Performance of FlashAttention – PyTorch" (https://pytorch.org/blog/flexattention/).
50. Dosovitskiy, Alexey; Beyer, Lucas; Kolesnikov, Alexander; Weissenborn, Dirk; Zhai, Xiaohua; Unterthiner, Thomas; Dehghani, Mostafa; Minderer, Matthias; Heigold, Georg (2021-06-03), *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, arXiv:2010.11929 (https://arxiv.org/abs/2010.11929)
51. Abnar, Samira; Zuidema, Willem (2020-05-31), *Quantifying Attention Flow in Transformers*, arXiv:2005.00928 (https://arxiv.org/abs/2005.00928)
52. Brocki, Lennart; Binda, Jakub; Chung, Neo Christopher (2024-10-25), *Class-Discriminative Attention Maps for Vision Transformers*, arXiv:2312.02364 (https://arxiv.org/abs/2312.02364)
53. Gildenblat, Jacob (2025-07-21), *jacobgil/pytorch-grad-cam* (https://github.com/jacobgil/pytorch-grad-cam), retrieved 2025-07-21
54. Mullenbach, James; Wiegreffe, Sarah; Duke, Jon; Sun, Jimeng; Eisenstein, Jacob (2018-04-16), *Explainable Prediction of Medical Codes from Clinical Text*, arXiv:1802.05695 (https://arxiv.org/abs/1802.05695)
55. Bahdanau, Dzmitry; Cho, Kyunghyun; Bengio, Yoshua (2016-05-19), *Neural Machine Translation by Jointly Learning to Align and Translate*, arXiv:1409.0473 (https://arxiv.org/abs/1409.0473)
56. Serrano, Sofia; Smith, Noah A. (2019-06-09), *Is Attention Interpretable?*, arXiv:1906.03731 (https://arxiv.org/abs/1906.03731)
57. Lee, Juho; Lee, Yoonho; Kim, Jungtaek; Kosiorek, Adam R; Choi, Seungjin; Teh, Yee Whye (2018). "Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks". arXiv:1810.00825 (https://arxiv.org/abs/1810.00825) [cs.LG (https://arxiv.org/archive/cs.LG)].

External links

- Olah, Chris; Carter, Shan (September 8, 2016). "Attention and Augmented Recurrent Neural Networks" (https://distill.pub/2016/augmented-rnns/). *Distill*. **1** (9). Distill Working Group. doi:10.23915/distill.00001 (https://doi.org/10.23915%2Fdistill.00001).
- Dan Jurafsky and James H. Martin (2022). *Speech and Language Processing* (3rd ed. draft, January 2022) (https://web.stanford.edu/~jurafsky/slp3/) — Chapter 10.4 (Attention) and Chapter 9.7 (Self-Attention Networks: Transformers)
- Alex Graves (2020). Attention and Memory in Deep Learning (https://www.youtube.com/watch?v=AIiwuClvH6k) — video lecture from DeepMind / UCL