Introduction to Visual Transformers and Transformer Attributions

Filip Kołodziejczyk

MI2.AI Seminar, Warsaw, May 13th, 2024

Agenda

- Recap of Transformer architecture
- Introduction to Visual Transformers
- Explaining the Transformers with Attention
- Review of *Deep Learning: Foundations and Concepts*

Recap of Transformer Architecture

based on

Deep Learning: Foundations and Concepts

by C.M. Bishop and H. Bishop

Note: all the figures without explicit source details come from the book; they were published on the book's website: Deep Learning - Foundations and Concepts (bishopbook.com)

Simple definition

Transformer transforms a set of vectors in some representation space into a corresponding set of vectors in some new space. We wish this new space to have a richer internal representation. For that, we use chained transformer layers, which comprise two stages: the attention mechanism, mixing corresponding features, and 2nd stage, processing each row independently.

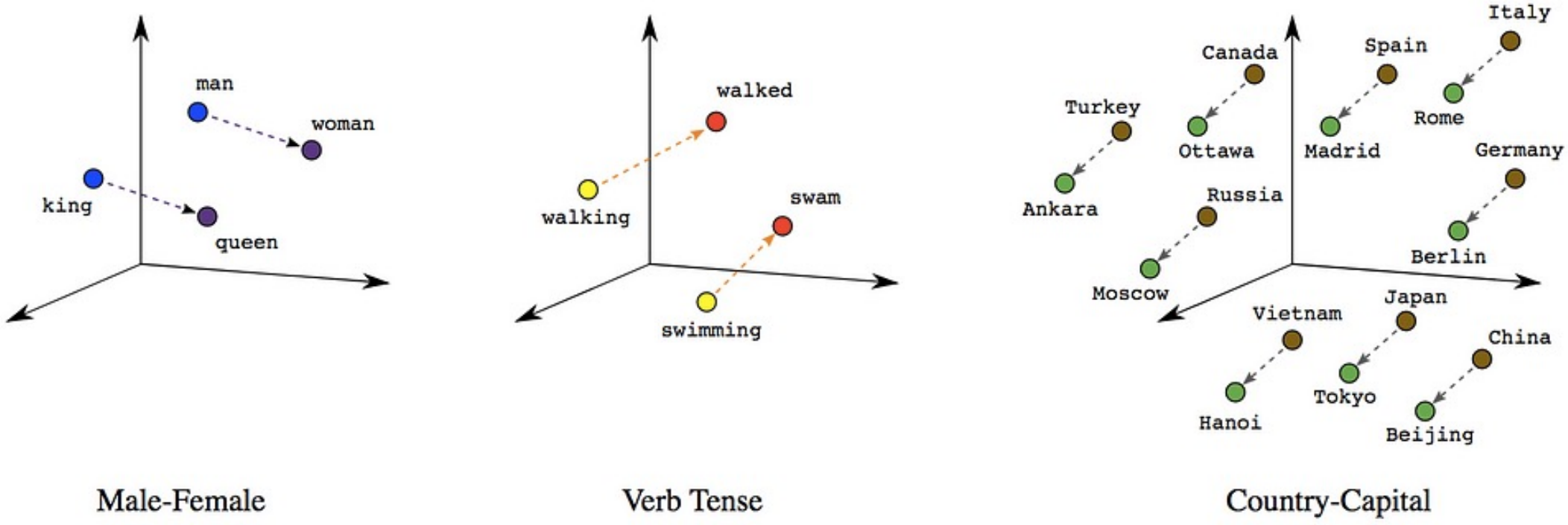
$$\tilde{X} = \mathrm{TransformerLayer}[X]$$

Promises it gives:

- Very effective transfer learning, foundational models reused for many downstream tasks via fine-tuning.
- Possible to train via self-supervised learning to process vast amounts of data.
- Enables parallel processing, so huge models are possible to create.

Transformers, initially created for NLP tasks, achieved SOTA in many domains.

Before the transformers (word embeddings)



Male-Female Verb Tense Country-Capital

Source: A Guide to Word Embedding. What are they? How are they more useful... | by Shraddha Anala | Towards Data Science

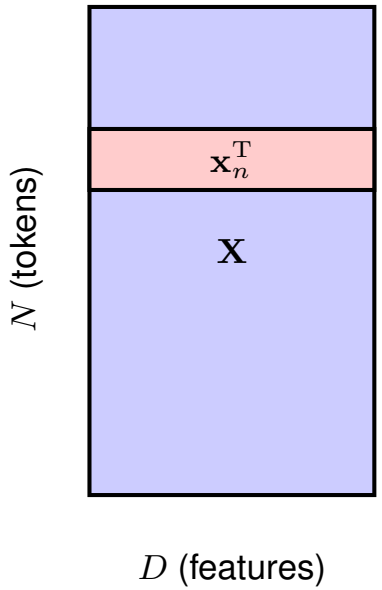
Problem with word embeddings

Word embedding always maps a given word to the same embedding vector, but words change meaning influenced by context. NN should attend to, i.e., rely more on some words from the rest when determining the interpretation of the word of our interest. Moreover, those words' position may vary from sentence to sentence.

*I **swam** across the **river** to get to the other **bank**.*
*I walked across the road to get **cash** from **bank**.*

Attention as solution (naïve aproach)

Transformation layer should transform input – set of vectors (called tokens) $\{x_1, \dots, x_n\} \in \mathbb{M}^{N \times \mathrm{D}}$ (N – #tokens, D – #features) into richer form of embedding, $\{y_1, \dots, y_n\}$ where each y_i depends on all input vectors $\{x_1, \dots, x_n\}$.



$$y_n = \sum_{m=1}^{N} a_{nm} x_m$$

$$a_{nm} \geq 0, \qquad \sum_{m=1}^{N} a_{nm} = 1$$

$$Y = ?$$

Any candidates for weighting coefficient a_{nm}?

Information retrival will help

Problem: Choose which movie to watch on an online movie streaming service.

Solution: Associate each movie with a list of attributes. A user can then provide a list of desired characteristics. We can find the film with the best attributes that match the desired ones. We can even measure the degree of match for each one. As a result, we return this movie. We call the list of movie attributes *keys* K, user list a *query* Q, and, movies corresponding to keys – *values* V.

Mapping to the attention problem:

How to measure the similarity between the x_n and x_m ?

$V - x_n$

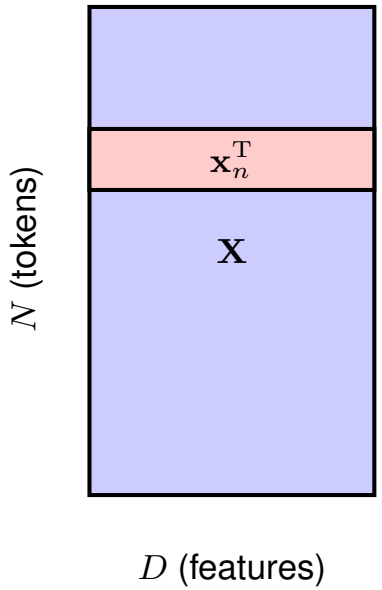
$K - x_n$

$Q - x_m$

We use the dot product between keys and queries, i.e. $x_n^T x_m$.

Attention as solution (naïve aproach)

The transformation layer should transform input – set of vectors (called tokens) $\{x_1, \dots, x_n\} \in \mathbb{M}^{N \times \mathrm{D}}$ (N – #tokens, D – #features) into the richer form of embedding, $\{y_1, \dots, y_n\}$ where each y_i depends on all input vectors $\{x_1, \dots, x_n\}$.



$$y_n = \sum_{m=1}^{N} a_{nm} x_m$$

$$a_{nm} \geq 0, \quad \sum_{m=1}^{N} a_{nm} = 1$$

$$Y = \mathrm{Softmax}[XX^T]X$$

$$a_{nm} = \frac{\exp(x_n^T x_m)}{\sum_{m'=1}^{M} \exp(x_n^T x_m)}$$

Attention as solution - challenges

Issue 1: The transformation is fixed; nothing can be learned. Also, each of the feature values is equally important.

Solution: Introduce a weight parameters $U \in \mathbb{M}^{D \times \mathrm{D}}$. Then, we can use $\tilde{X} = XU$.

$$Y = \mathrm{Softmax}[XX^T]X \longrightarrow Y = \mathrm{Softmax}[XUU^TX^T]XU$$

Issue 2: Matrix XUU^TX^T is symmetric, while the attention mechanism should support asymmetry. E.g. The *chisel* should be strongly related to the *tool* but not vice versa (as there are many tools).

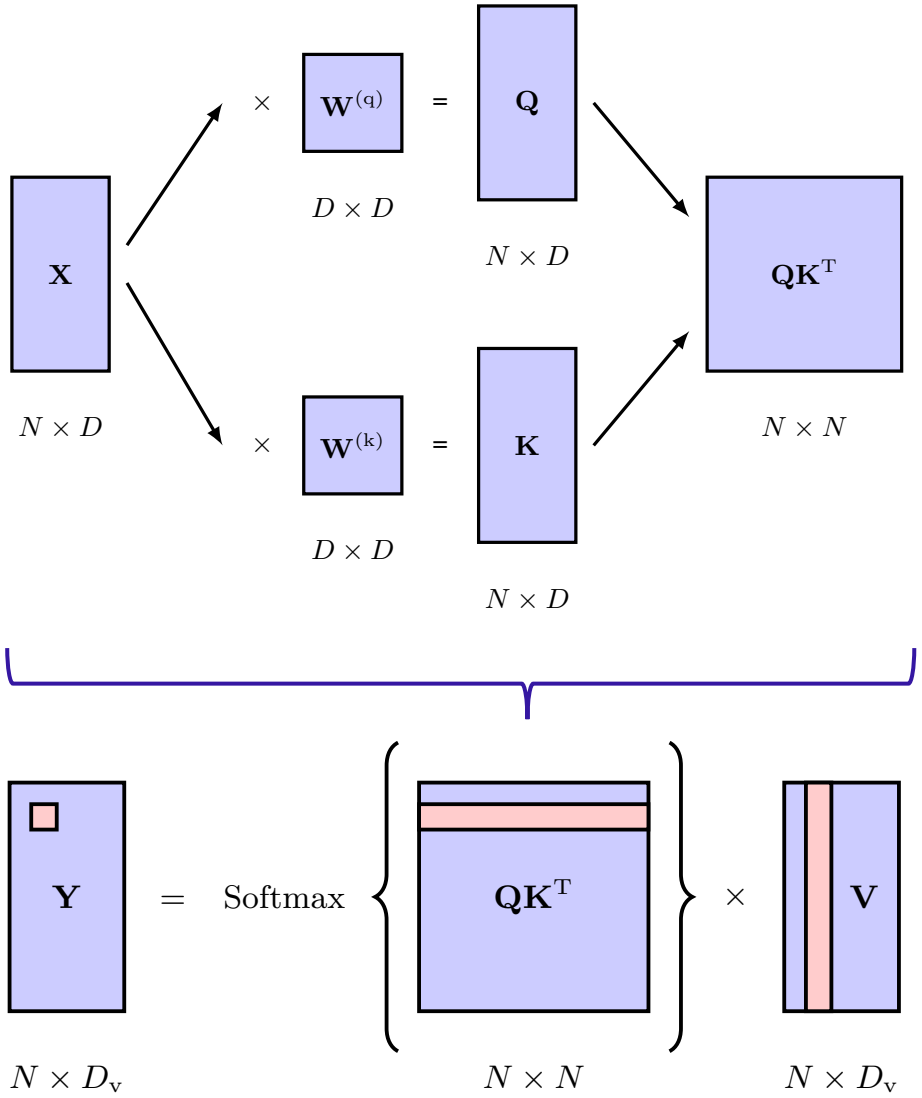
Solution: Introduce separate weights $W^{(q)}, W^{(k)}, W^{(v)}$ and use $Q = XW^{(q)}, K = XW^{(k)}, V = XW^{(v)}$.

$$Y = \mathrm{Softmax}[XUU^TX^T]XU \longrightarrow Y = \mathrm{Softmax}[QK^T]V$$

Self-attention (result)

The name comes from using the same sequence for Q, K, and V.

We often add bias by augmenting X with a column of 1's.



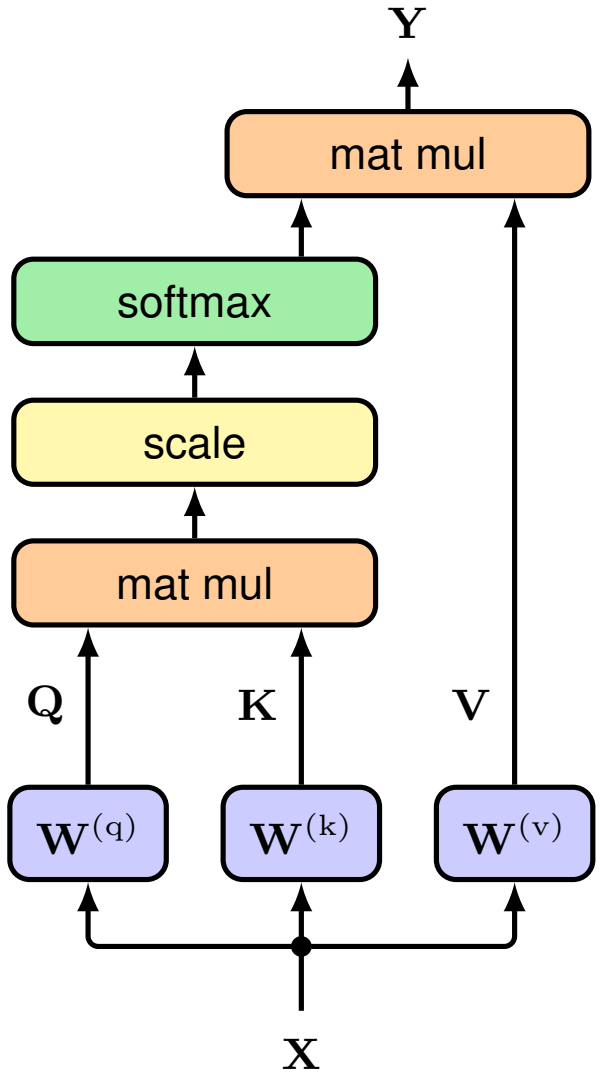
Self-attention - challenges

Issue 3: The gradient of softmax becomes exponentially small for high-magnitude inputs.

Solution: Rescale the QK before the softmax function. Assuming Q and K are the i.i.d. vectors with 0 mean and variance, then their dot product variance would be D.

$$Y = \text{Softmax}[QK^T]V$$

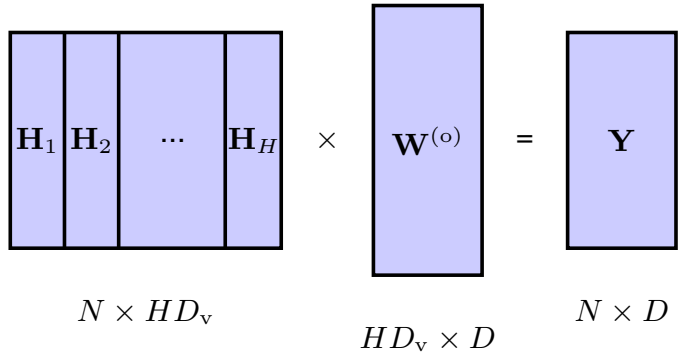
$$Y = \text{Attention}(Q, K, V) \stackrel{\text{def}}{=} \text{Softmax}\left[\frac{QK^T}{\sqrt{D}}\right]V$$



Final form (attention head)

Multi-head attention

Problem: Multiple attention patterns might be relevant simultaneously (e.g., in NLP, some patterns apply to tense, others associated with vocabulary). A single head might average those patterns.

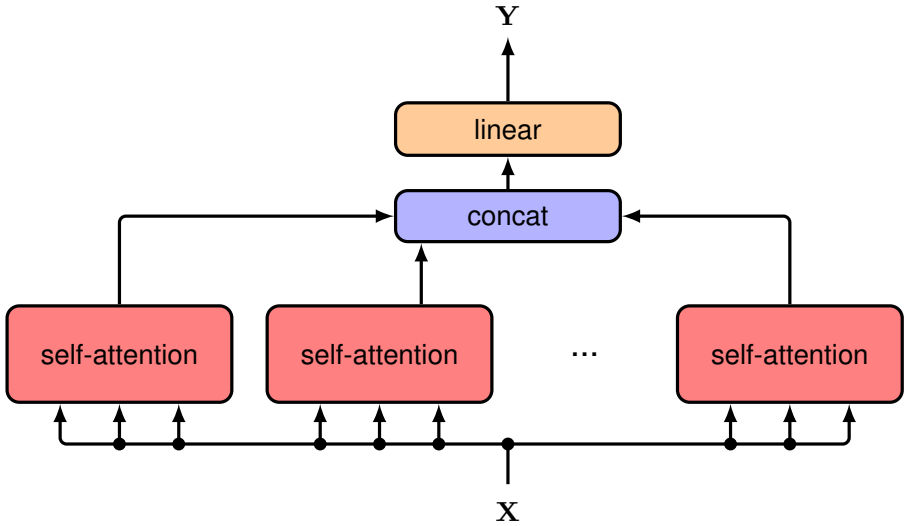


Solution: Use multiple attention heads in parallel, allowing each to focus on its pattern. We define heads as

$$H_h = \text{Attention}(Q_h, K_h, V_h)$$

Where each weight matrix $W_h^{(q)}, W_h^{(k)}, W_h^{(v)}$ is independent. To combine those heads to get single output, we concatenate them and linearly transform using matrix $W^{(\circ)}$ (lerned along with weigh matrices) to the desired dimension.

$$Y(X) = \text{Concat}[H_1, \dots, H_H]W^{(\circ)}$$



Transformer layer

- *Residual connections* are utilized to improve the training (stability and performance). Other training efficiency improvements come from *layer normalization*.

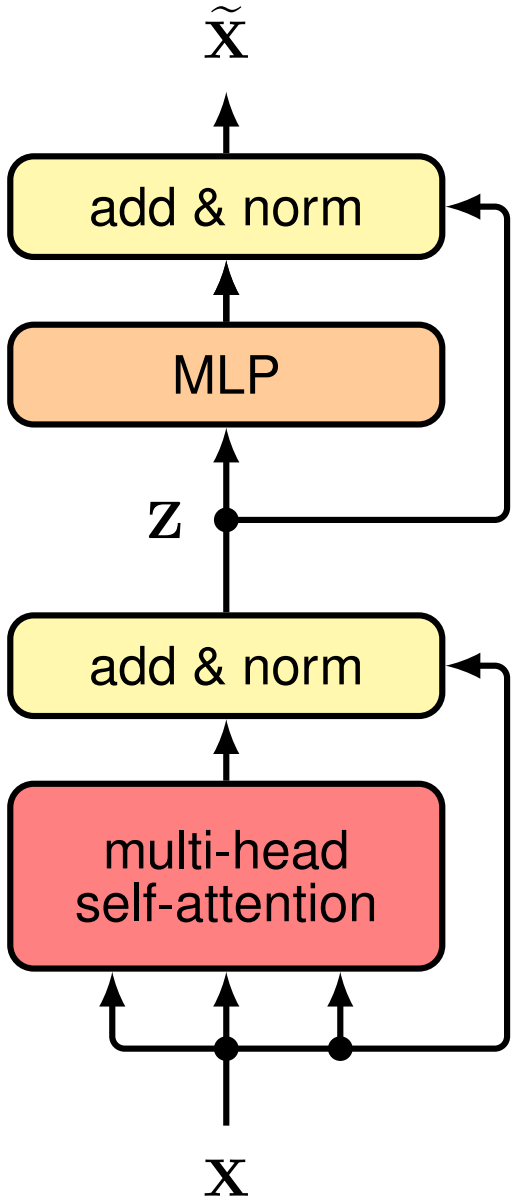
$$Z = \mathrm{LayerNorm}[Y(X) + X]$$

- Sometimes, the order of operations is reversed. This can result in more effective optimization).

$$Z = Y(\mathrm{LayerNorm}[X]) + X$$

- Finally, we can increase the layer flexibility by post-processing the attention result with a standard, fully connected network (multi-layer perceptron), which processes each vector one by one. Again, *layer normalization* and *residual connections* are used (in reversed order, too).

$$\tilde{X} = \mathrm{TransformerLayer}[X] = \mathrm{LayerNorm}[\mathrm{MLP}[\mathrm{Z}] + \mathrm{Z}]$$



Positional encoding

Issue: Transformer is equivariant to input permutations. Lack of dependence on token order limits the sequential capabilities.

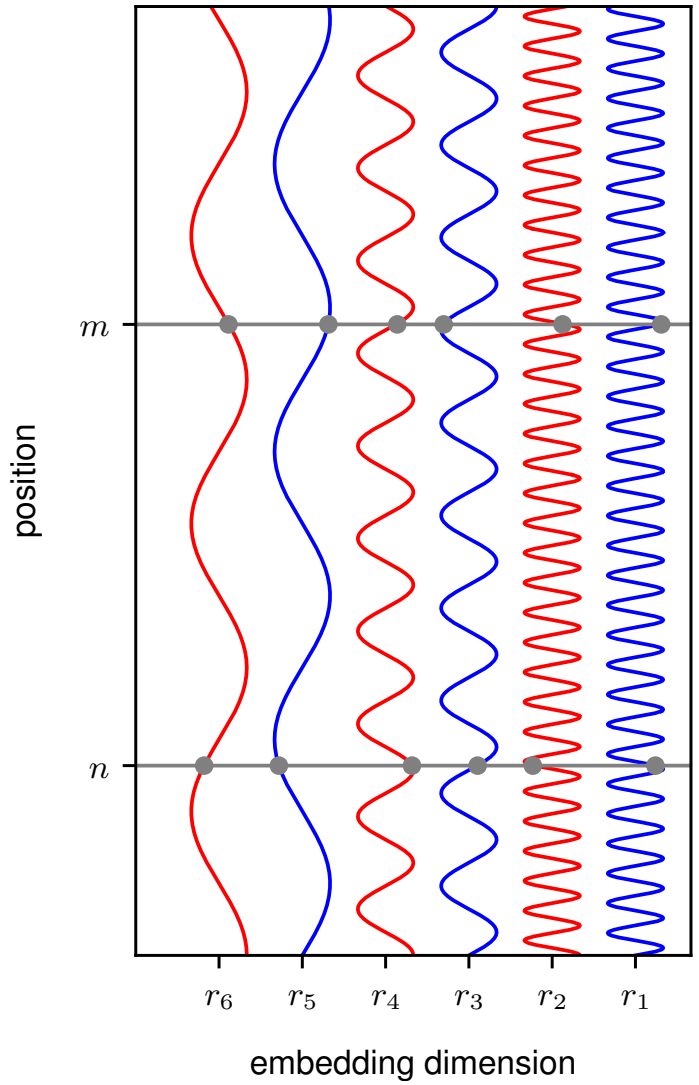
Solution: Construct the position encoding vector r_n and add it to the token vector:

$$\tilde{x}_n = x_n + r_n$$

Thanks to the residual connections, position encoding does not get lost through the consecutive layers.

Positional encoding should be invariant to the sequence length and unique for each position. A common solution is a technique based on sinusoidal functions (i – position, L – input sequence length):

$$r_{ni} = \begin{cases} \sin\left(\frac{n}{L^{i/D}}\right), & \text{if } i \text{ is even} \\ \cos\left(\frac{n}{L^{(i-1)/D}}\right), & \text{if } i \text{ is odd} \end{cases}$$



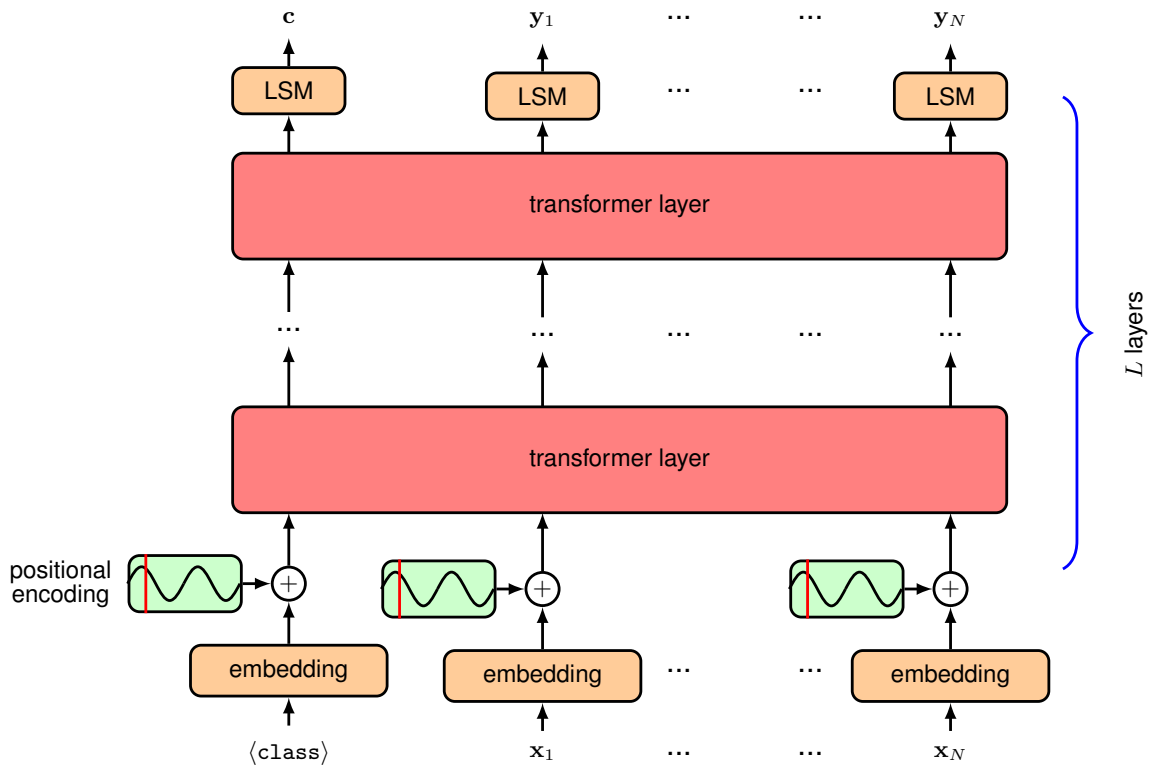
Transformer architecture categories

Depending on the form of the input and output of the model, we distinguish 3 types of transformers:

- **Encoder** – sequence as input and single variable as output
- **Decoder** – single vector as input and sequence as output
- **Sequence-to-sequence** – both the input and output comprise a sequence of variables

Encoder transformers

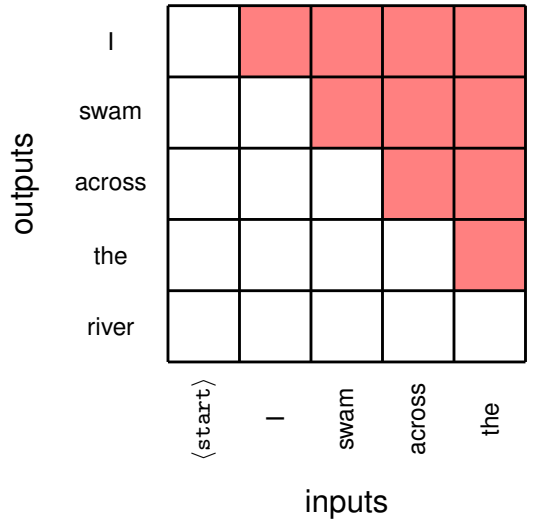
- Uses standard transformer layers, as presented in the previous slides – self-attention is bidirectional, i.e., uses both preceding and following tokens concerning the current token
- Sample use cases: classification, missing word prediction, Named Entity Recognition
- Sample foundational model: BERT

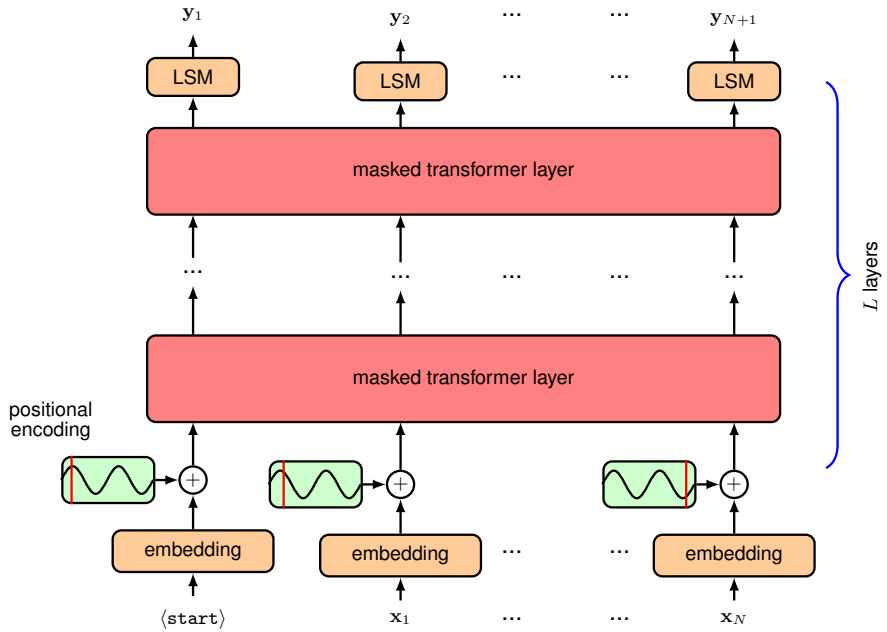


Decoder transformers

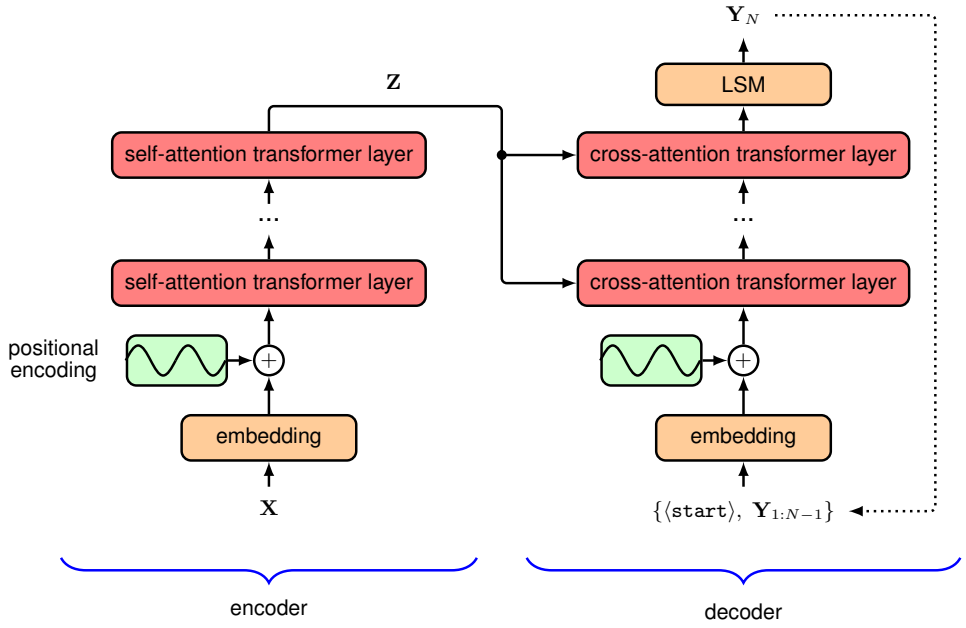
- Main goal is predict future values based on past values (autoregressive model)
- Uses masked transformer layers – self-attention is unidirectional, i.e., uses only preceding tokens. Thanks to that, the model can still be trained in parallel using the whole sequence without cheating: knowledge about the following tokens.
- Masking is done by setting the attention weight of all the following tokens to 0 or $-\infty$.
- While training is parallelized, output generation is sequential
- Sample use cases: next word prediction, sentence completion, text-to-speech
- Sample foundational model: GPT-3

Masking

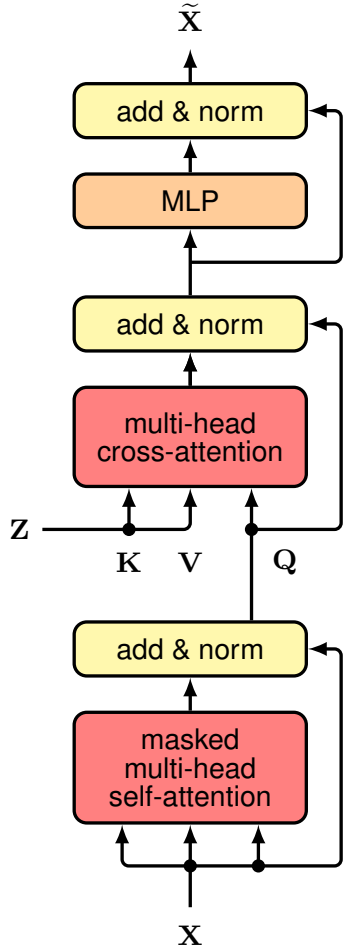




Sequence-to-sequence transformers



- Uses both an encoder and a decoder
- The encoder consumes the input sequence and creates its representation
- Then, the decoder generates an output sequentially, using the so-far generated output
- The decoder's transformer layer is modified: after the masked self-attention layer, there is an additional cross-attention layer
- Cross-attention layer combines the representation of the input sequence created by the encoder with the decoder's masked-layer output
- Sample use cases: machine translation, Q&A, image captioning
- Sample foundational model: BART



Cross-Attention Layer

Introduction to Visual Transformers

based on

Deep Learning: Foundations and Concepts

by C.M. Bishop and H. Bishop

Note: all the figures without explicit source details come from the book; they were published on the book's website: Deep Learning - Foundations and Concepts (bishopbook.com)

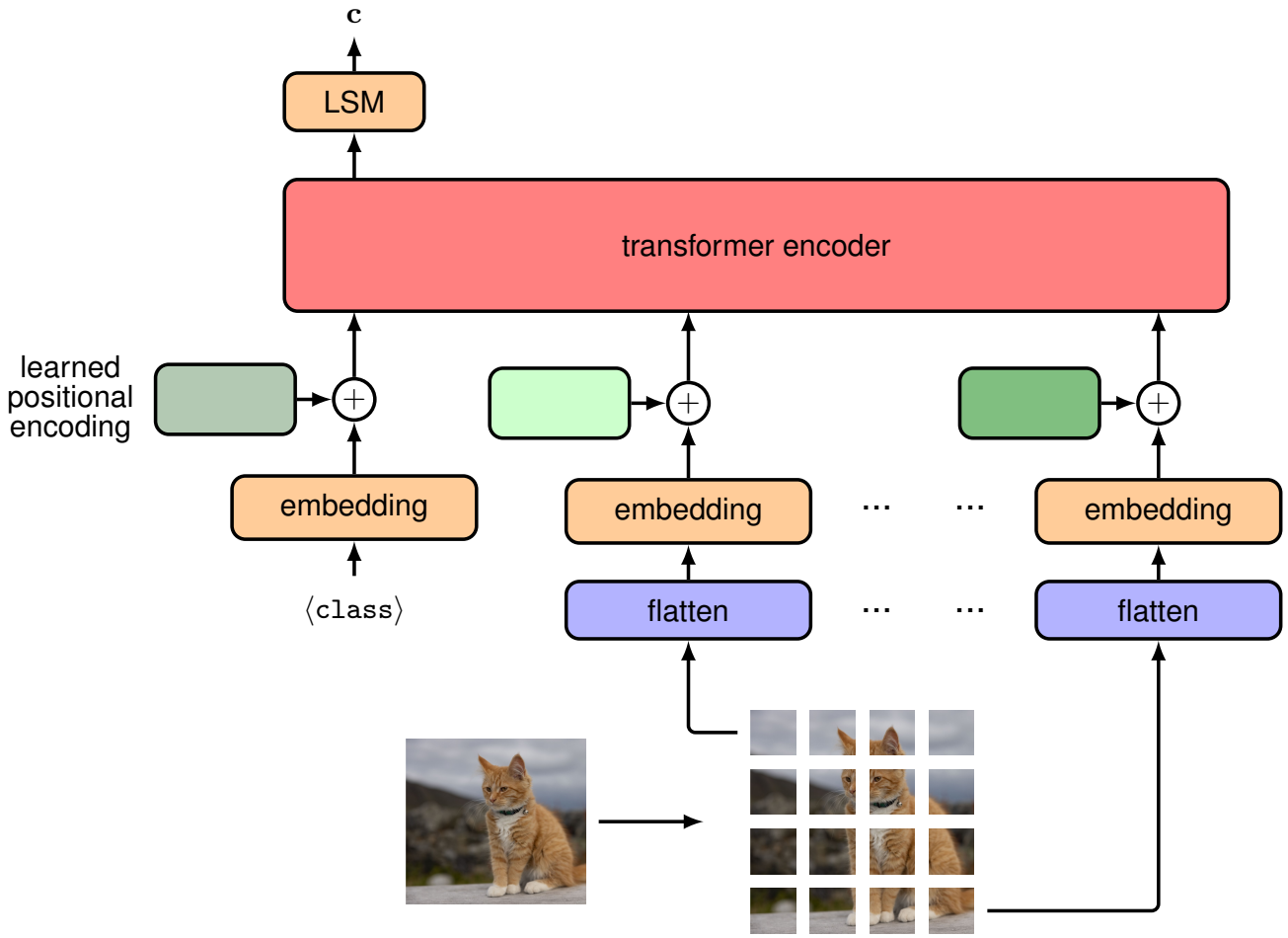
Vision transformers

Transformers make very few assumptions about input data. Therefore, they can be used for a variety of tasks. One of them is CV, where it achieved a SOTA performance. The most common choice is the encoder transformer. This approach is called Vision Transformer (ViT).

When it comes to processing images with transformers, a crucial step is tokenization. The most straightforward approach is to use each pixel as a token. However, the memory requirements of a standard transformer increase quadratically with the number of input tokens, posing a significant challenge. This complexity underscores the need for innovative solutions. Two most common are:

- Split images $x \in \mathbb{R}^{H \times W \times C}$ into a set of patches of the same size. Each image is split into non-overlapping patches of size $P \times P$ and then flattened into a one-dimensional vector, which gives representation $x_p \in \mathbb{R}^{N \times (P^2 C)}$ where the total number of patches for images is $N = \frac{HW}{P^2}$.
- Feed the image into a small CNN. This will downsample the image to the managable numer of tokens.

Vision transformers



Explaining the Transformers with Attention

based on

Quantifying Attention Flow in Transformers

by S. Abnar and W. Zuidema

Note: all the figures and tables without explicit source details come from the paper; published here: [2005.00928] Quantifying Attention Flow in Transformers (arxiv.org)

Authors

- **Samira Abnar** – Apple ML research team member and PhD candidate at the University of Amsterdam; research interests: ML and Cognitive Science
- **Willem Zuidema** - associate professor of NLP, explainable AI , and cognitive modelling at the University of Amsterdam; research interests: NLP, Interpretability, XAI, Cognitive Modelling, Evolution of Language

Setups and Problem Statement

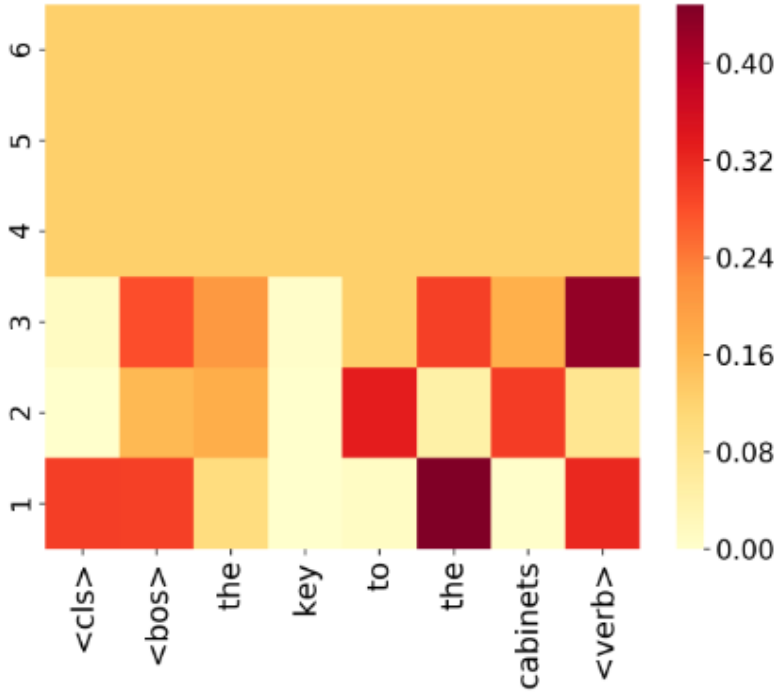
- **Problem:** Verb number prediction task - predicting singularity or plurality of a verb of a sentence, where the input is the sentence up to the verb position
- **Model:** transformer encoder, with GPT2 Transformer blocks (w/o masking), 6 layers, 8 heads, embedding size of 128, using CLS [classification] token
- **Achieved accuracy:** 0.96

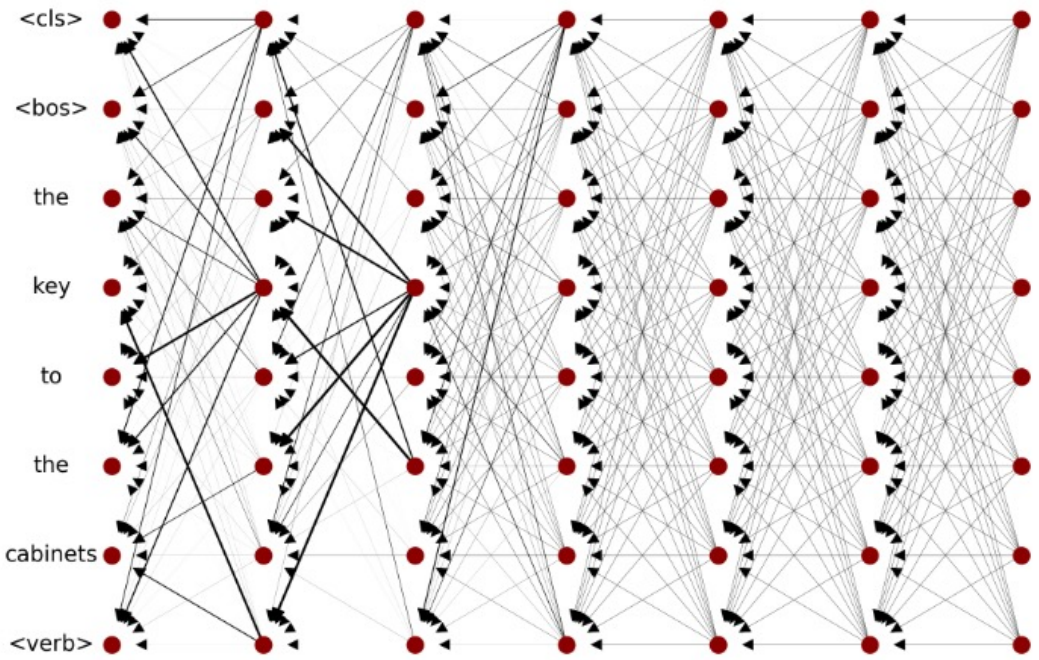
Self-Attention vs Explainability

- Attention visualizations as easy and very popular way to interpret the model
- Due to the nature of self-attention, across the layers of model different tokens get increasingly mixed
- **Which part of the input is the most important?**

Raw Attention

Visualization of correctly classified example. In higher layers, there is a lack of interpretable pattern.





General concept

- Model information flow with a DAG:
  - nodes = input tokens/hidden embeddings
  - edges = attentions for the nodes to the previous layer
- Recursively compute token attentions in each layer
- Include residual connections
- Multi-head attention via averaging the attention (*)
- Computing values in layer $l + 1$:

$$V_{l+1} = V_l + W_{att} V_l, \qquad (W_{att} - \text{attention matrix})$$

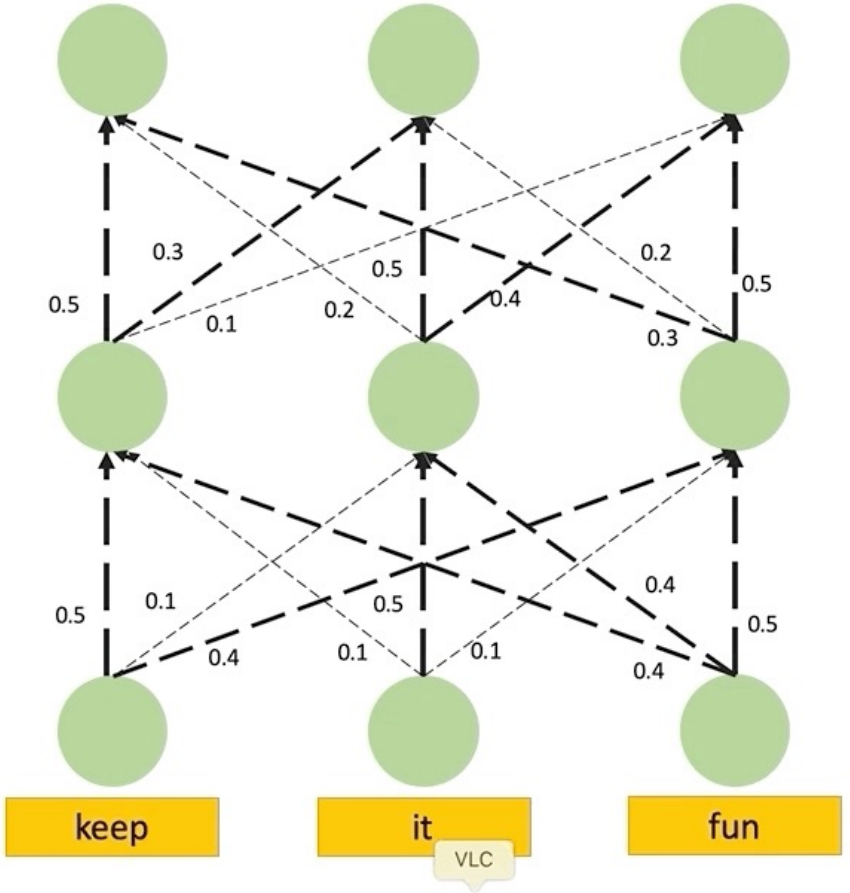
- Only for visualization and debugging, no impact on the model itself (applied post hoc)

Attention rollout

- Path as a proportion of information transferred between nodes (multiply the weights of all edges)
- More than one path possible, the total amount of information is the sum of all possible paths
- Recursive formula
  (l_k- k-th layer, $j \leq i$, A – raw attention):

$$\tilde{A}(l_i) = \begin{cases} A(l_i)\tilde{A}(l_{i-1}) & \text{if } i > j \\ A(l_i) & \text{if } i = k \end{cases}$$

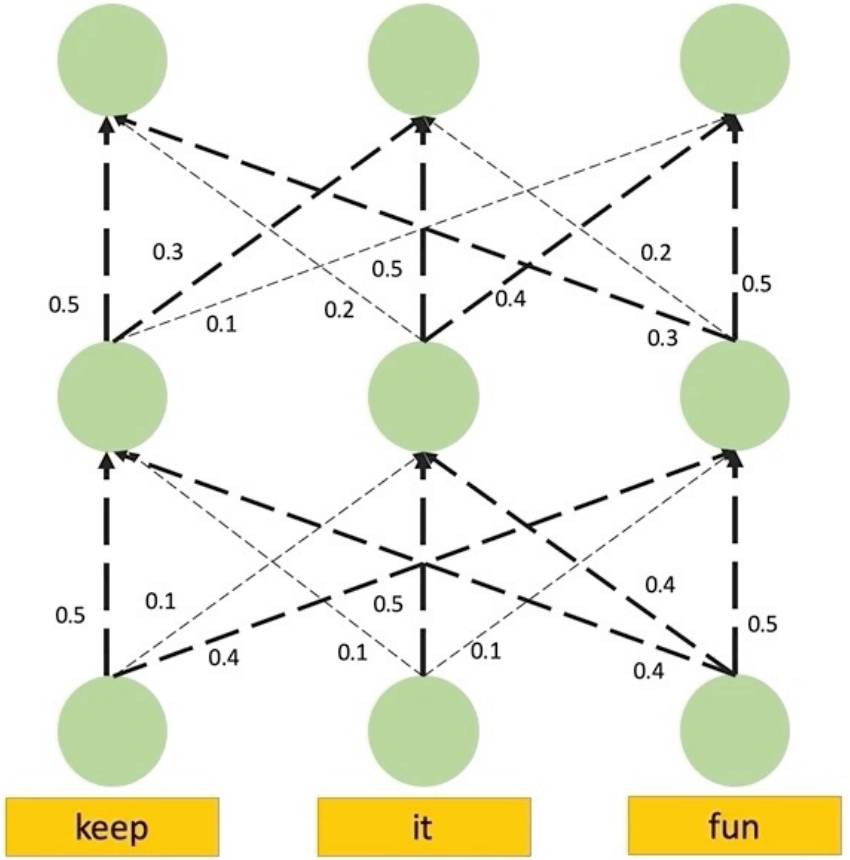
- Polynomial time complexity $O(d * n^2)$



Source: Quantifying Attention Flow in Transformers | Samira Abnar

Attention flow

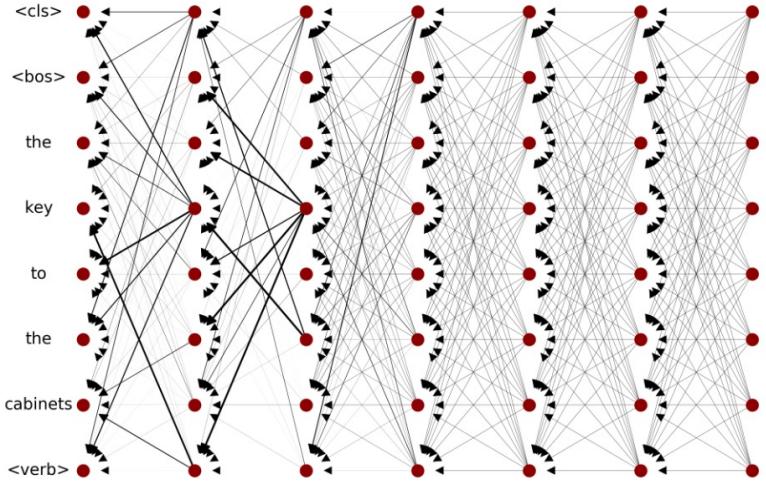
- Treating attention graph as flow network where capacities of edges are attention weights
- We calulate maximum attention flow from any node in any of the layers to any of the input nodes
- Solvable using any maximum flow algorithm e.g. Ford–Fulkerson algorithm
- Compared to attention rollout, the weight of single path is the minimum value of the weights, instead of the product
- Polynomial time complexity $O(d * n^4)$



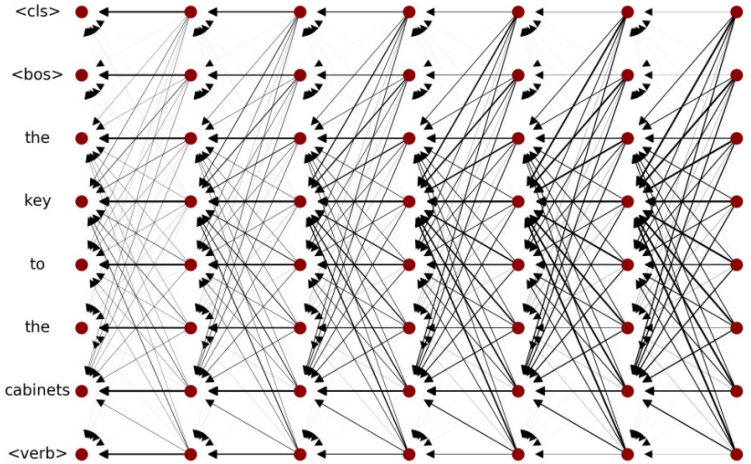
Source: Quantifying Attention Flow in Transformers | Samira Abnar

Comparison

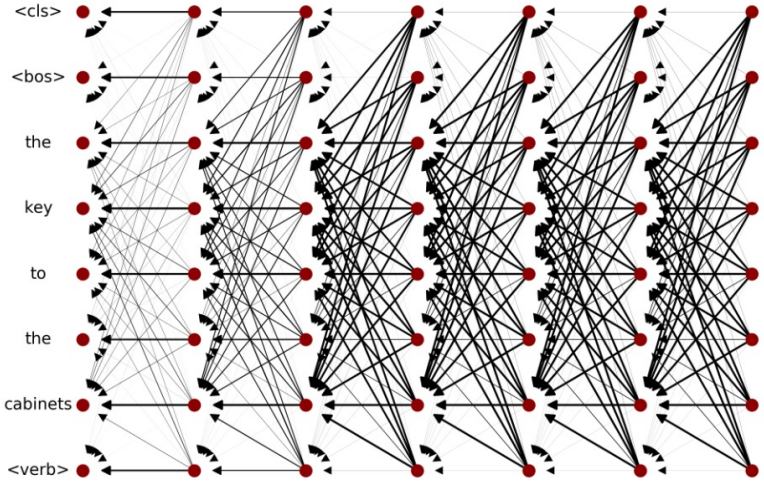
Visualization of correctly classified example.



(a) Embedding attentions



(b) Attention rollout



(c) Attention flow

Comparison cont.

The first and second sentences were classified correctly, while 3rd was misclassified.

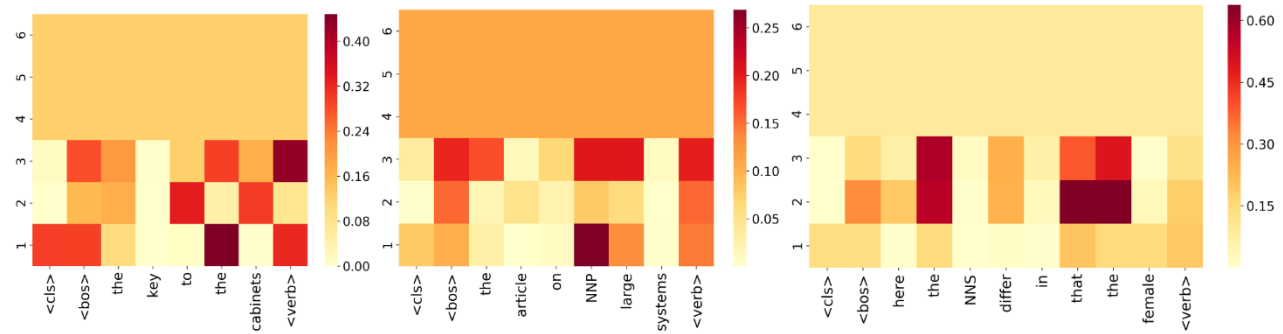


Figure 2: Raw Attention maps for the CLS token at different layers.

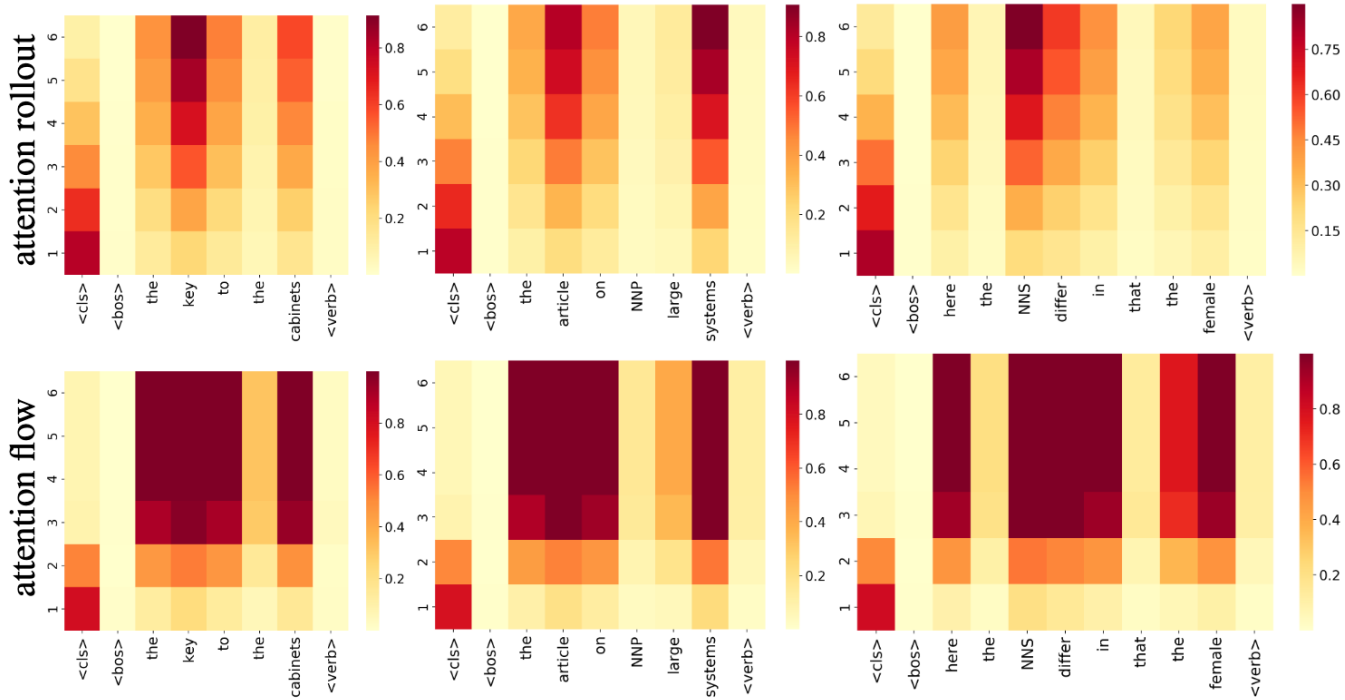


Figure 3: Attention maps for the CLS token

Results

SpearmanR correlation measures rank correlation, i.e., assesses how well the relationship between two variables can be described using a monotonic function.

	L1	L2	L3	L4	L5	L6
Raw	0.69±0.27	0.10±0.43	-0.11±0.49	-0.09±0.52	0.20±0.45	0.29±0.39
Rollout	0.32±0.26	0.38±0.27	0.51±0.26	0.62±0.26	0.70±0.25	0.71±0.24
Flow	0.32±0.26	0.44±0.29	0.70±0.25	0.70±0.22	0.71±0.22	0.70±0.22

Table 1: SpearmanR correlation of attention based importance with blank-out scores for 2000 samples from the test set for the verb number prediction model.

	L1	L2	L3	L4	L5	L6
Raw	0.53±0.33	0.16±0.38	-0.06±0.42	0.00±0.47	0.24±0.40	0.46±0.35
Rollout	0.22±0.31	0.27±0.32	0.39±0.32	0.47±0.32	0.53±0.32	0.54±0.31
Flow	0.22±0.31	0.31±0.34	0.54±0.32	0.61±0.28	0.60±0.28	0.61±0.28

Table 2: SpearmanR correlation of attention based importance with input gradients for 2000 samples from the test set for the verb number prediction model.

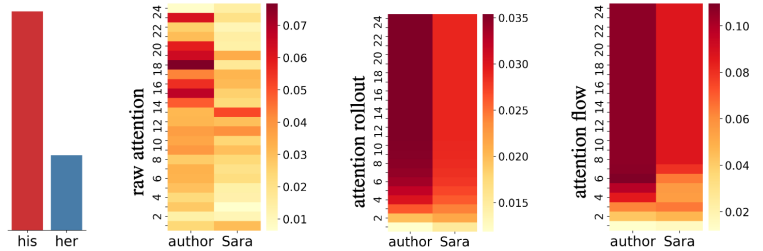
Results (other task)

Result from the sentiment analysis task.

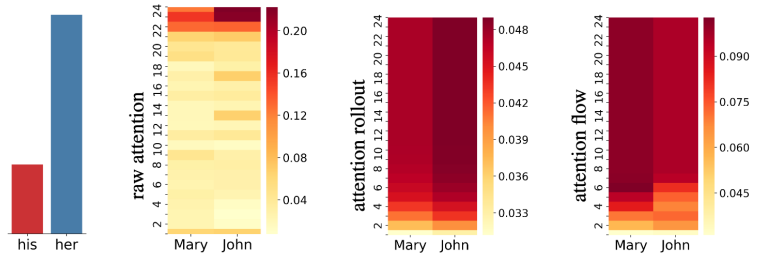
	L1	L3	L5	L6
Raw	0.12 ± 0.21	0.09 ± 0.21	0.08 ± 0.20	0.09 ± 0.21
Rollout	0.11 ± 0.19	0.12 ± 0.21	0.13 ± 0.21	0.13 ± 0.20
Flow	0.11 ± 0.19	0.11 ± 0.21	0.12 ± 0.22	0.14 ± 0.21

Table 3: SpearmanR correlation of attention based importance with input gradients for 100 samples from the test set for the DistillBERT model fine tuned on SST-2.

Result from the pre-trained Bert (resolving pronounce in sentence.



(a) "The author talked to Sara about `mask` book"



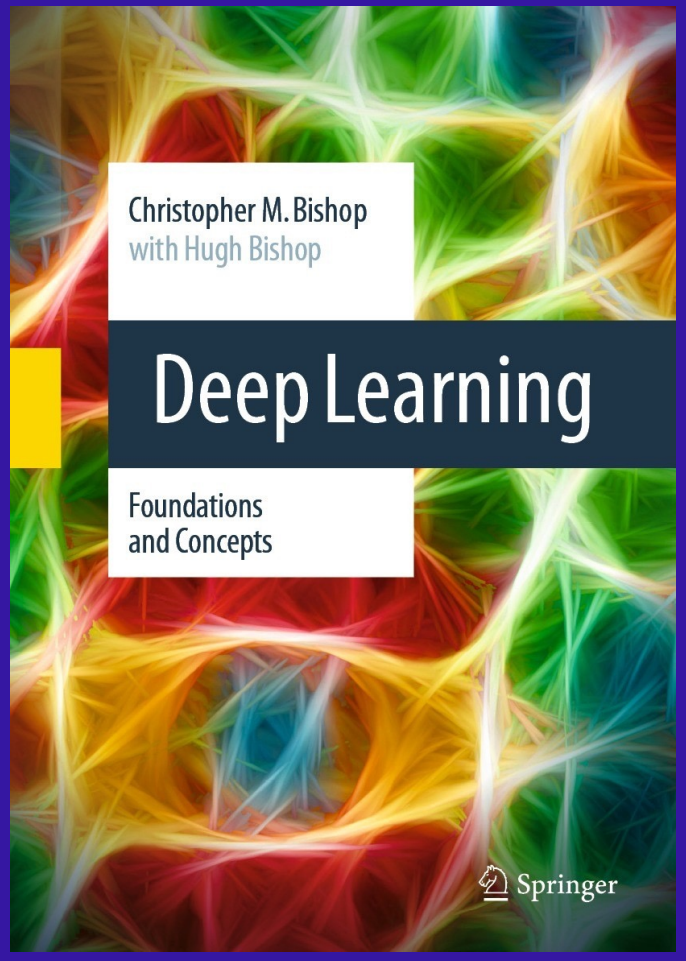
(b) *"Mary convinced John of* `mask` *love"*

Figure 4: Bert attention maps. We look at the attention weights from the `mask` embedding to the two potential references for it, e.g. "author" and "Sara" in (a) and "Mary" and "John" in (b). The bars, at the left, show the relative predicted probability for the two possible pronouns, "his" and "her".

Conclusions

- Proposed methods provide results more correlated with blank-out scores and input gradients than raw attention.
- Attention rollout provides more strict patterns (but not necessarily more accurate results), while attention flow is more flexible.
- Those methods are easy to implement and task-agnostic.
- In paper spplied for encoder-only. To apply for decoder, normalization is needed since both attention rollout and attention flow will be biased toward the input sentence.
- It is possible to analyze each head of attention separately (*).
- As with many other XAI methods, there is a high emphasis that it does not necessarily identify hidden embeddings

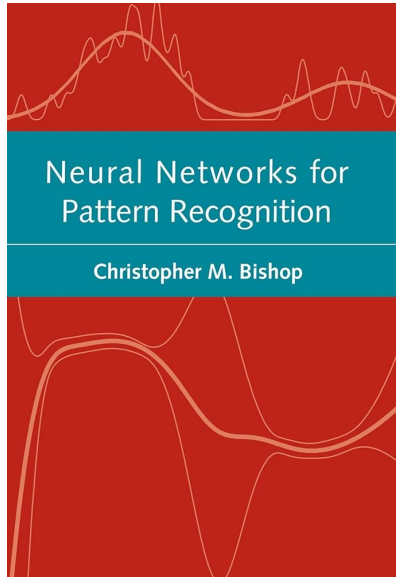
Review of Deep Learning: Foundations and Concepts

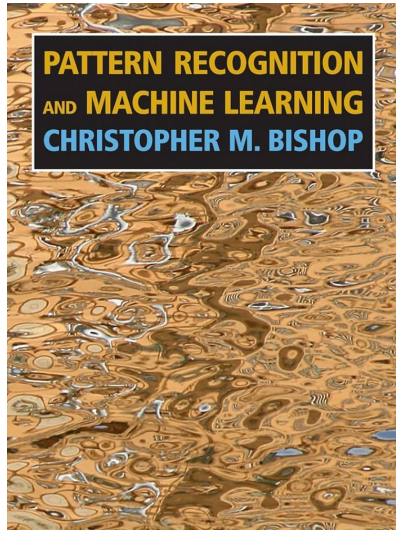


Authors

Chris Bishop *is a Technical Fellow at Microsoft and is the Director of Microsoft Research AI4Science.* Professor of Computer Science, teaching at the University of Edinburgh. Author of two other influential ML textbooks: **Neural Networks for Pattern Recognition** and **Pattern Recognition and Machine Learning**.

Hugo Bishop *is an Applied Scientist at Wayve, a deep learning autonomous driving company in London where he designs and trains deep neural networks.* Cambridge graduate in Machine Learning and Machine Intelligence.





Goals of the book

- Provide a solid basis for deep learning, both for newcomers and those already experienced in the field
- Focus on critical ideas and foundations
- Focus on technicals only, without discussing ethical, societal, or legal aspects
- No literature review due to rapid changes
- Suited for two-semester undergraduate or postgraduate ML course

Book structure

Book structure cont.

Book structure cont.

Book structure cont.

Book structure cont.

Review (pros)

- It starts with the basics: probability, distributions, regression, etc.; the only prerequisite seems to be calculus and linear algebra
- Provides intuitive and easy-to-understand examples and analogies
- Simple (for an academic book) language used
- Lots of figures
- Most of the first-time appearing keywords provided with a comprehensive explanation
- **Available online with a complete set of figures available for download**
- Equiped with the exercies ...

Exercises sneak peak

10.1 (⋆) Consider a fixed weight vector $\mathbf{w}$ and show that the input vector $\mathbf{x}$ that maximizes the scalar product $\mathbf{w}^{\mathrm{T}}\mathbf{x}$, subject to the constraint that $\|\mathbf{x}\|^2$ is constant, is given by $\mathbf{x} = \alpha\mathbf{w}$ for some scalar α. This can most easily be done using a Lagrange multiplier.

10.2 (⋆⋆) Consider a convolutional network layer with a one-dimensional input array and a one-dimensional feature map as shown in Figure 10.2, in which the input array has dimensionality 5 and the filters have width 3 with a stride of 1. Show that this can be expressed as a special case of a fully connected layer by writing down the weight matrix in which missing connections are replaced by zeros and where shared parameters are indicated by using replicated entries. Ignore any bias parameters.

10.3 (⋆) Explicitly calculate the output of the following convolution of a 4×4 input matrix with a 2×2 filter:

$$\begin{array}{|c|c|c|c|} \hline 2 & 5 & -3 & 0 \\ \hline 0 & 6 & 0 & -4 \\ \hline -1 & -3 & 0 & 2 \\ \hline 5 & 0 & 0 & 3 \\ \hline \end{array} \quad * \quad \begin{array}{|c|c|} \hline -2 & 0 \\ \hline 4 & 6 \\ \hline \end{array} \quad = \quad \begin{array}{|c|c|c|} \hline ? & ? & ? \\ \hline ? & ? & ? \\ \hline ? & ? & ? \\ \hline \end{array} \tag{10.18}$$

10.4 (⋆⋆) If an image $\mathbf{I}$ has $J \times K$ pixels and a filter $\mathbf{K}$ has $L \times M$ elements, write down the limits for the two summations in (10.2). In the mathematics literature, the operation (10.2) would be called a *cross-correlation*, whereas a *convolution* would be defined by

$$C(j,k) = \sum_{l} \sum_{m} I(j-l, k-m) K(l,m). \tag{10.19}$$

Write down the limits for the summations in (10.19). Show that (10.19) can be written in the equivalent 'flipped' form

$$C(j,k) = \sum_{l} \sum_{m} I(j+l, k+m) K(l,m) \tag{10.20}$$

and again write down the limits for the summations.

10.5 (⋆) In mathematics, a convolution for a continuous variable x is defined by

$$F(x) = \int_{-\infty}^{\infty} G(y) k(x-y)\, \mathrm{d}y \tag{10.21}$$

where $k(x-y)$ is the kernel function. By considering a discrete approximation to the integral, explain the relationship to a convolutional layer, defined by (10.19), in a CNN.

Review (cons)

- … but they are purely mathematical
- Thus, as ML textbook lacks a practical part (coding and working with real data)
- Despite the name, the authors introduce this as an ML textbook (in the Preface), yet it lacks a few basic ML concepts, e.g., decision tree, SVM, ensemble, and reinforcement learning…
- Relatively not approachable by those without (mathematical) academic background (?)

Thank You for your attention!

Filip Kołodziejczyk

MI2.AI Seminar, Warsaw, May 13th, 2024