Chapter 4
Transformers and Attention Mechanisms

Transformer

- The transformer model is a type of neural network architecture that excels at processing sequential data, most prominently associated with large language models (LLMs).

- Elite performance in such as computer vision, speech recognition and time series forecasting.

- The BERT (or Bidirectional Encoder Representations from Transformers) encoder-decoder model, introduced by Google in 2019, was a major landmark in the establishment of transformers and remains the basis of most modern word embedding applications, from modern vector databases to Google search.

Importance:

- **Parallel processing of sequences (much faster than RNNs/LSTMs).**
- **Handling long-range dependencies (important words in long sentences far apart).**

Transformer: a specific kind of network architecture, like a fancier feedforward network, but based on attention

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

A Very Approximate Timeline

1990 Static Word Embeddings

2003 Neural Language Model

2008 Multi-Task Learning

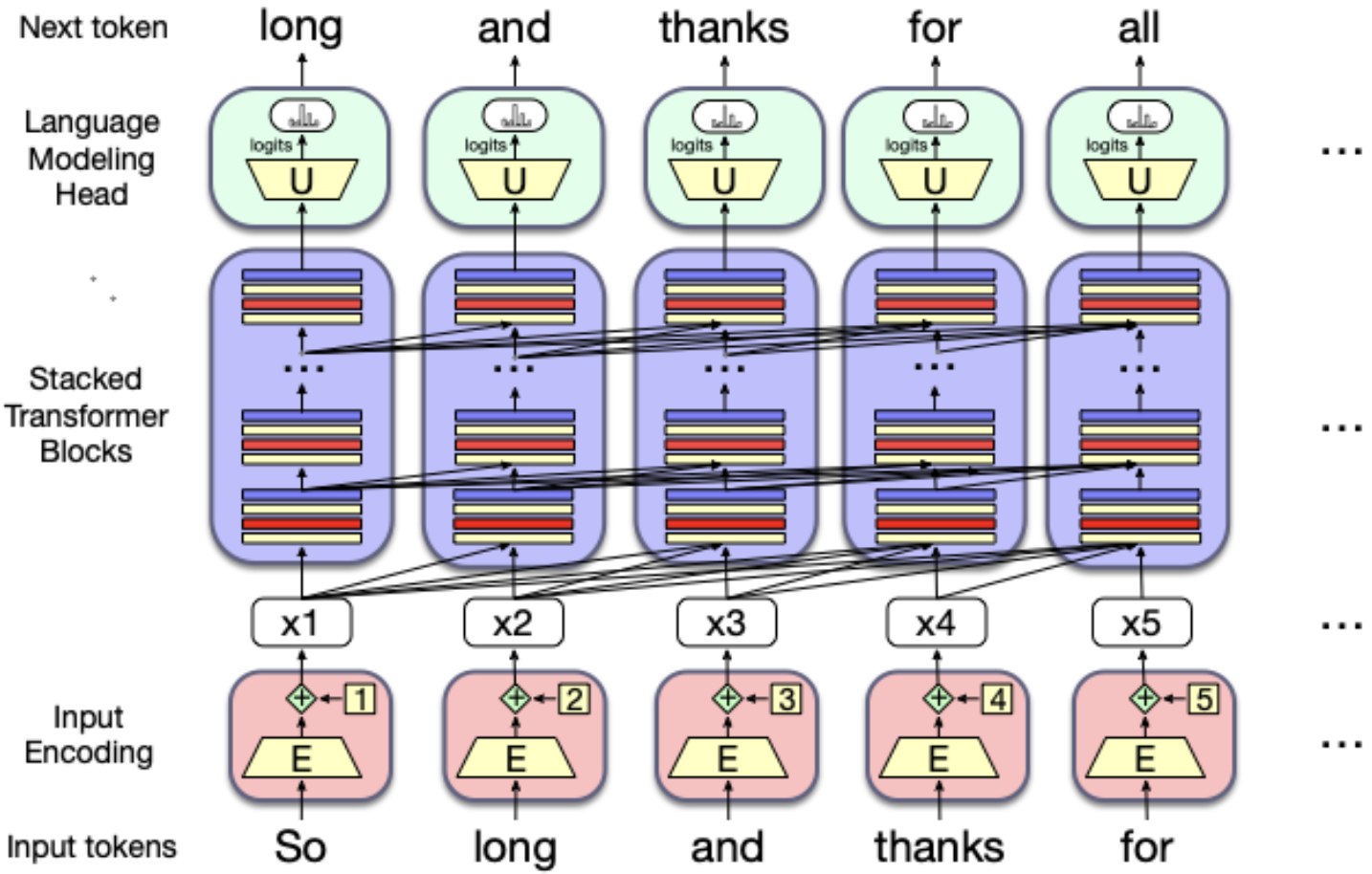
2015 Attention

2017 Transformer

2018 Contextual Word Embeddings and Pretraining

2019 Prompting

Instead of starting with the big picture



Problem with Static Embeddings (Word2Vec)

They are static! The embedding for a word doesn't reflect how its meaning changes in context.

The chicken didn't cross the road because it was too tired

What is the meaning represented in the static embedding for "it"?

Static Embeddings

`The chicken didn't cross the road because it`

What should be the properties of "it"?

`The chicken didn't cross the road because it was too` **`tired`**

`The chicken didn't cross the road because it was too` **`wide`**

At this point in the sentence, it's probably referring to either the chicken or the street

Contextual Embeddings

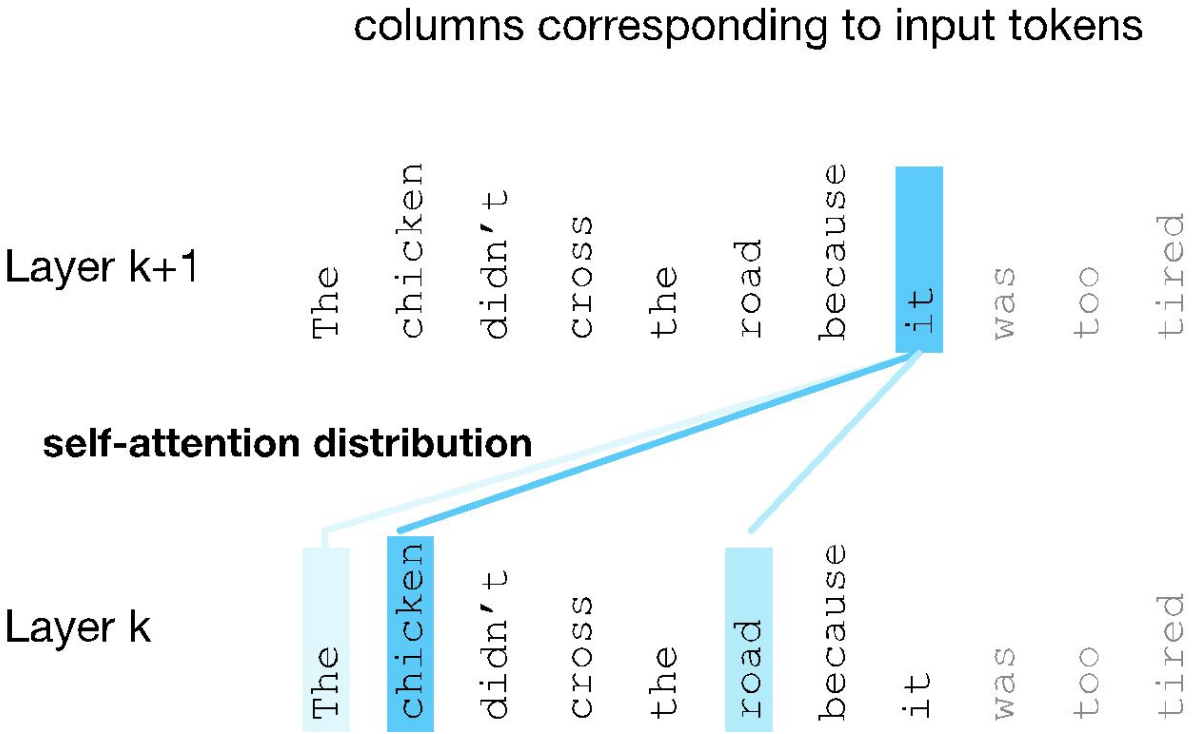
- **Intuition:** a representation of meaning of a word should be different in different contexts!

- **Contextual Embedding**: Each word has a different vector that expresses different meanings depending on the surrounding words

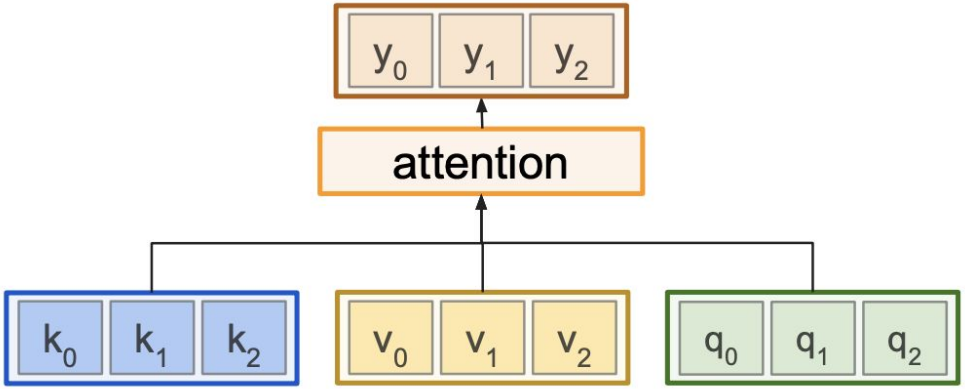
- How to compute contextual embeddings?
  Attention

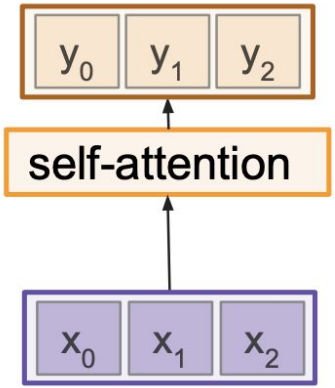
Intuition of Attention

- Build up the contextual embedding from a word by selectively integrating information from all the neighboring words. We say that a word "attends to" some neighboring words more than others.



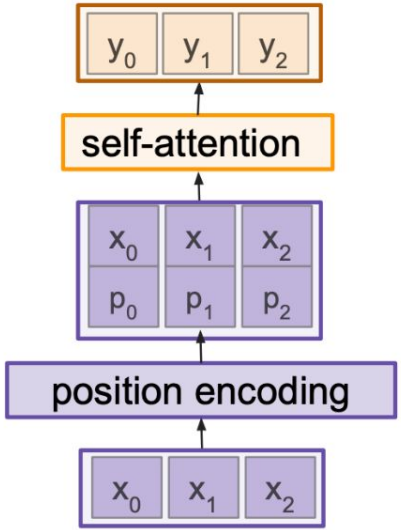
General vs Self-Attention





- **Queries (q_0, q_1, q_2)** → Represent the elements we want to find relevant information for.
- **Keys (k_0, k_1, k_2)** → Represent the identifiers of information (like tags).
- **Values (v_0, v_1, v_2)** → Represent the actual content/information linked with keys.

Position Encoding



Concatenate special positional encoding p_j to each input vector x_j

We use a function *pos*: $N \rightarrow R^d$ to process the position j of the vector into a d-dimensional vector

So, $p_j = pos(j)$

Transformers don't process sequences (like sentences) in order. They look at all the words at once.

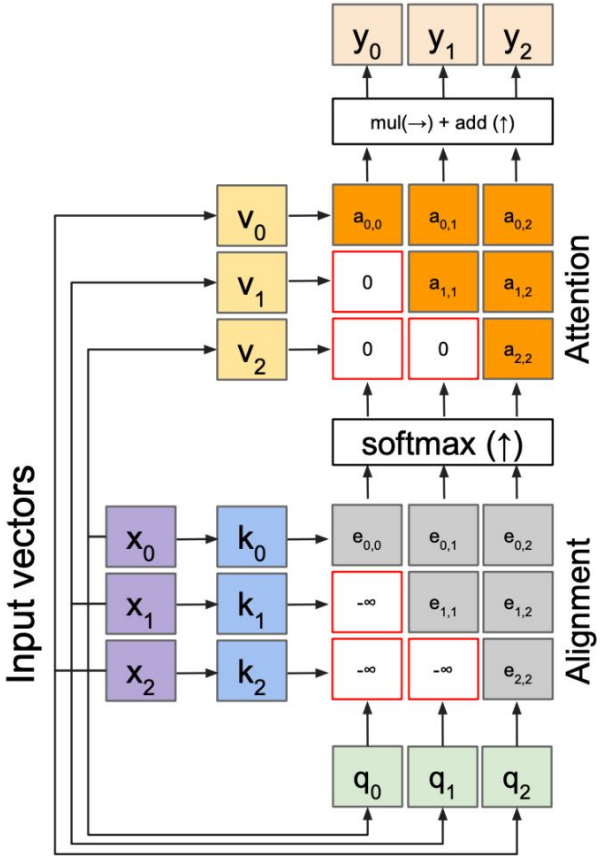
But... words in a sentence have order, like:

"The cat sat on the mat."

Without position information, the model doesn't know which word came first or last.

So we add position encoding to tell the model the order of the words.

Masked Self-Attention Layer



Outputs:
context vectors: **y** (shape: D_v)

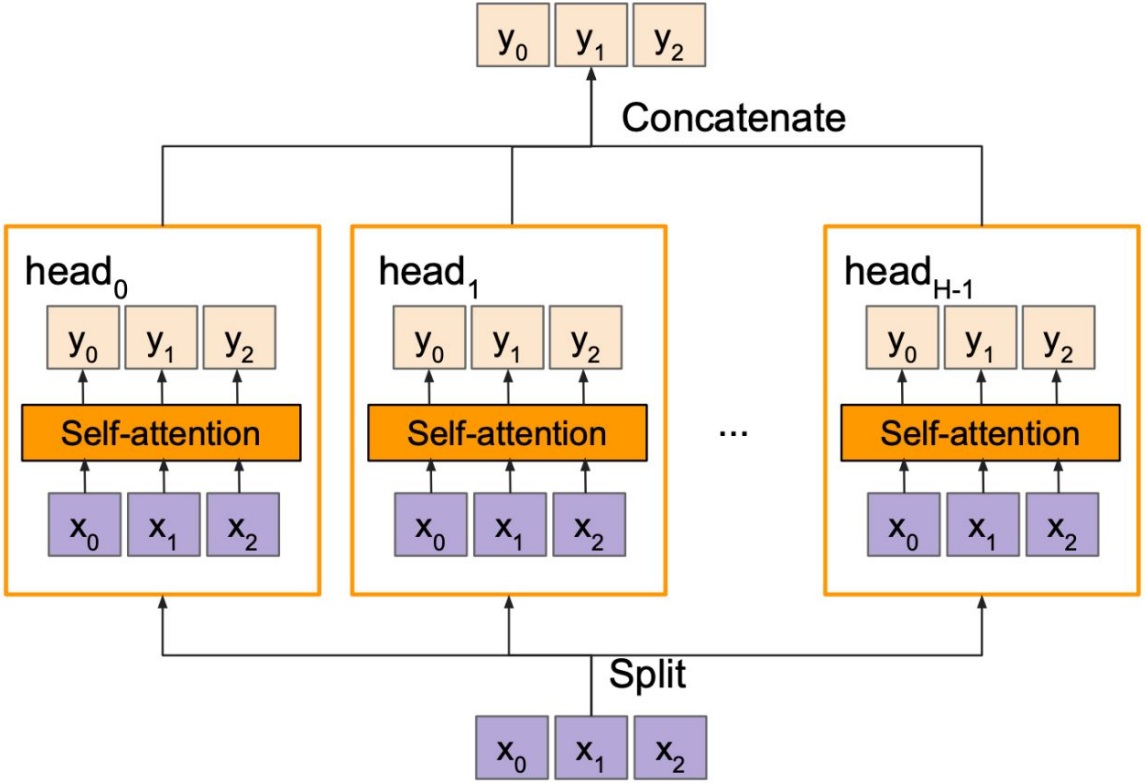
Operations:
Key vectors: $\mathbf{k} = \mathbf{x}W_k$
Value vectors: $\mathbf{v} = \mathbf{x}W_v$
Query vectors: $\mathbf{q} = \mathbf{x}W_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: **x** (shape: N x D)

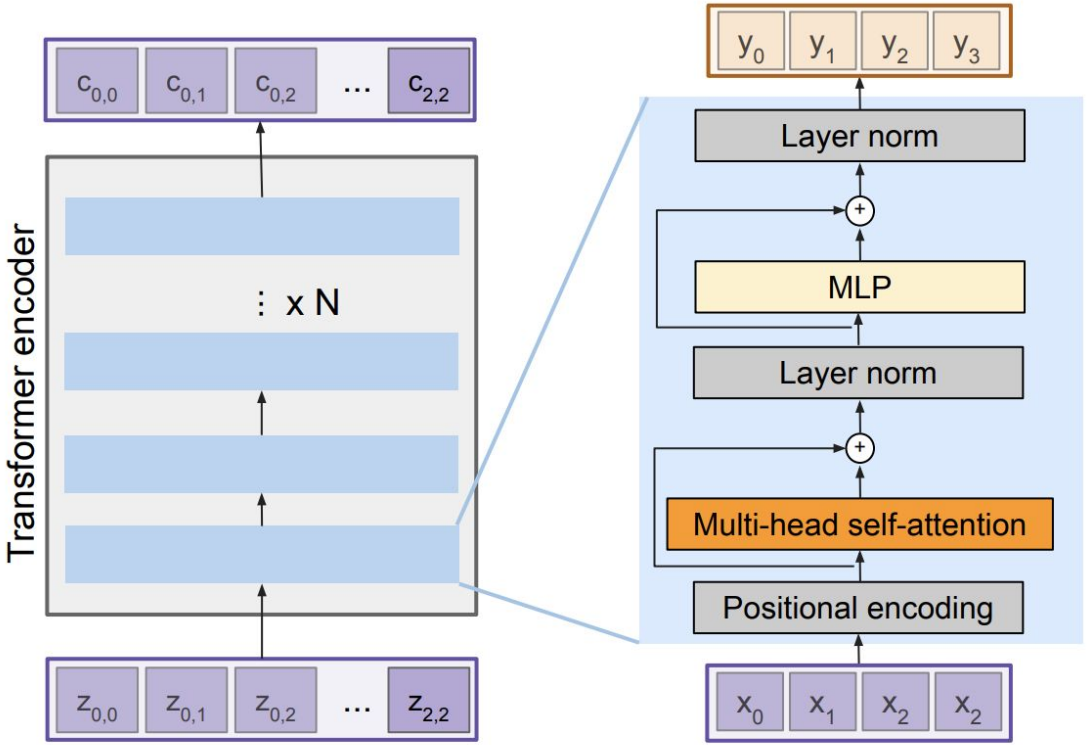
- Prevent vectors from looking at future vectors.
- Manually set alignment scores to -infinity

Multi head Self-Attention Layer

- Multiple self-attention heads in parallel



Transformer Encoder Block



Transformer Encoder Block:

Inputs: Set of vectors **x**
Outputs: Set of vectors **y**

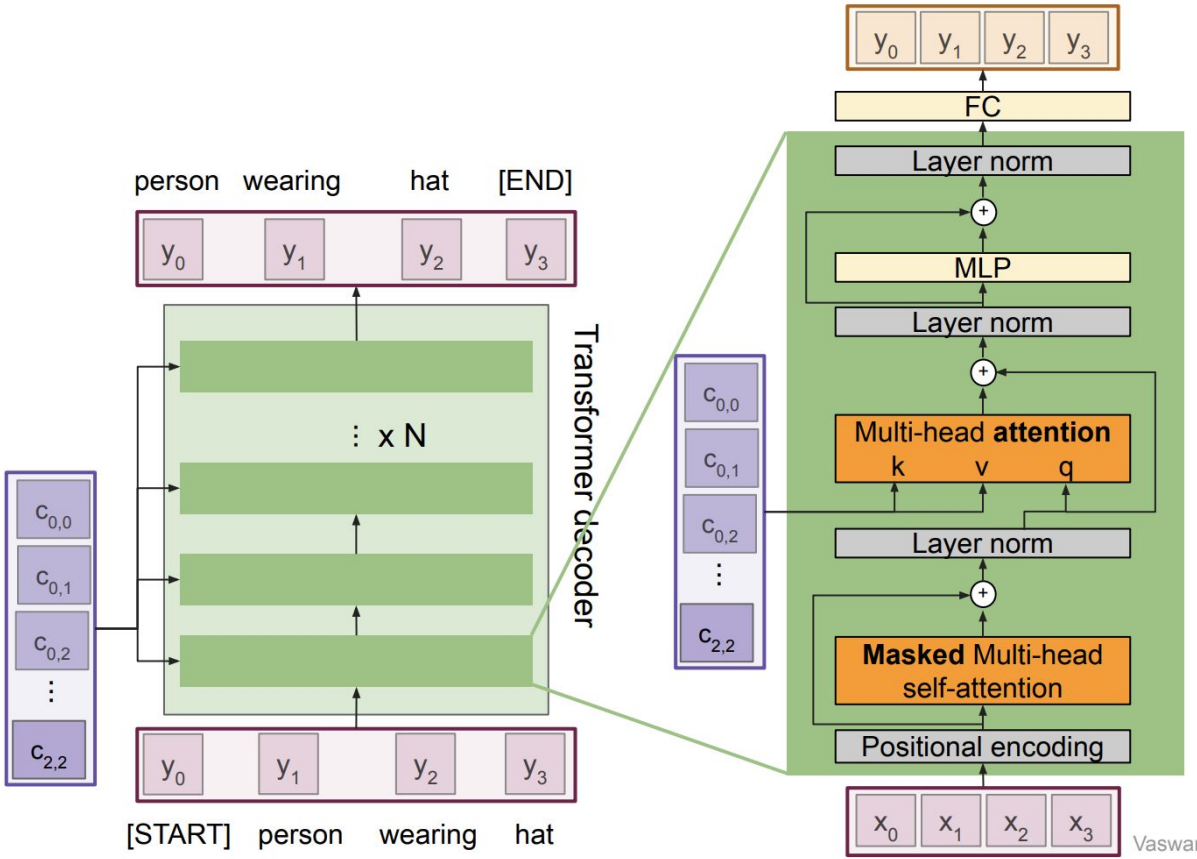
Self-attention is the only interaction between vectors.

Layer norm and MLP operate independently per vector.

Highly scalable, highly parallelizable, but high memory usage.

Vaswani et al, "Attention is all you need", NeurIPS 2017

Transformer Decoder Block



Transformer Decoder Block:

Inputs: Set of vectors **x** and Set of context vectors **c.**
Outputs: Set of vectors **y.**

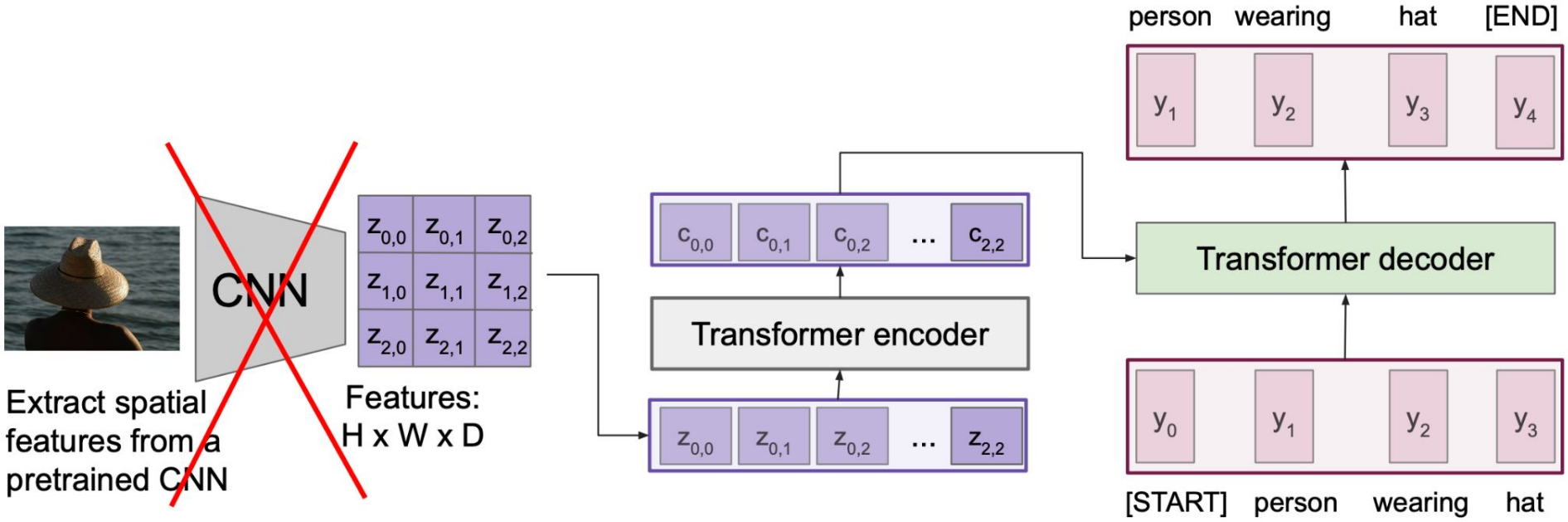
Masked Self-attention only interacts with past inputs.

Multi-head attention block is NOT self-attention. It attends over encoder outputs.

Highly scalable, highly parallelizable, but high memory usage.

Vaswani et al, "Attention is all you need", NeurIPS 2017

Image Captioning Using Transformer



Processes the visual tokens and outputs context-aware representations (c_{i,j}). These representations capture the global context of the image.

The decoder predicts the next word in the caption one token at a time, based on: The encoded image features, and The previously generated caption tokens.

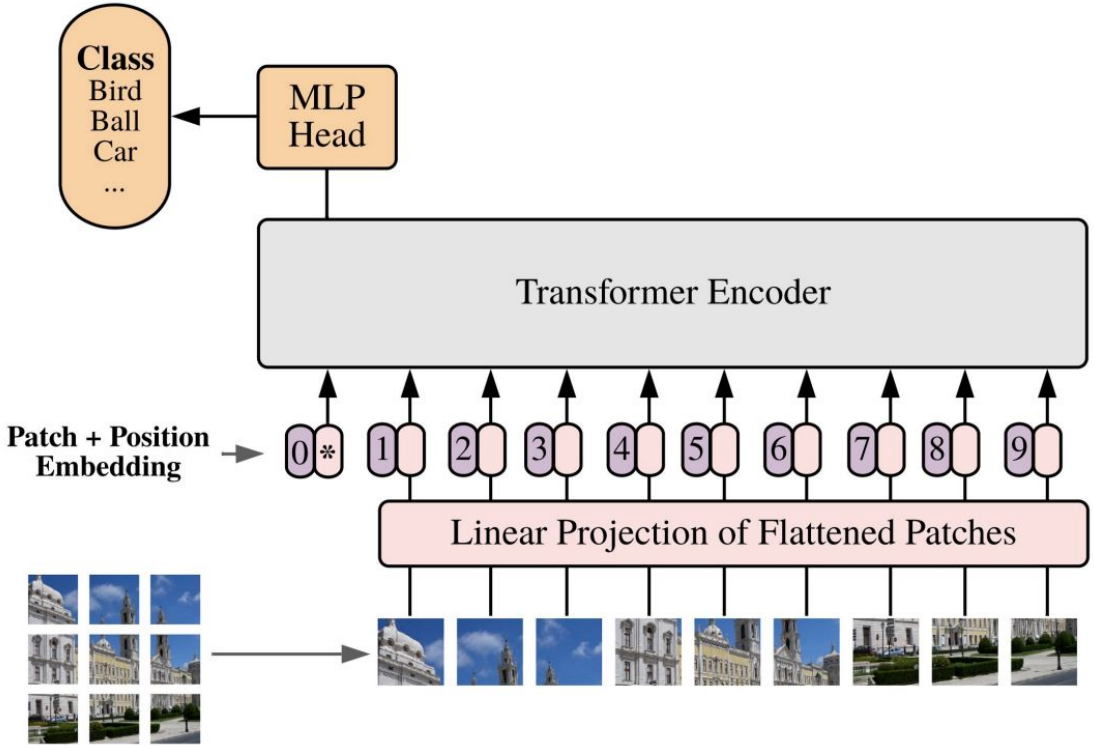
Transformer for Vision

- “LSTM → Transformer” ~ “ConvNet → ??? ”
- Issue with self-attention for vision: computation is quadratic in the input sequence length, quickly gets very expensive (with > few thousand tokens)
  - For ImageNet: 224x224 pixels → ~50,000 sequence length
  - Even worse for higher resolution and video

How can we deal with this quadratic complexity?

Vision Transformer

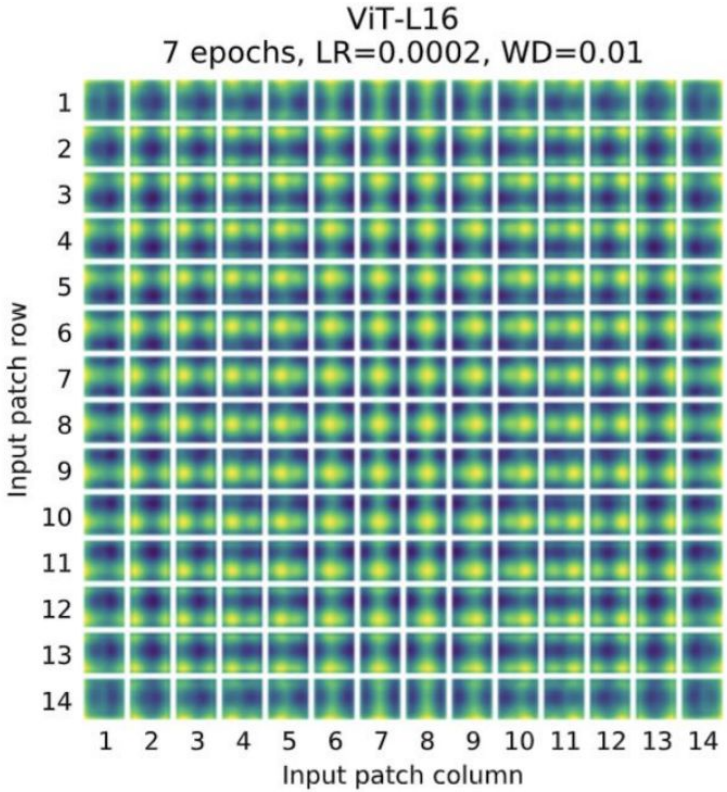
Idea: Take a transformer and apply it directly to image patches



Cordonnier et al., *On the Relationship between Self-Attention and Convolutional Layers*, ICLR 2020

Dosovitskiy et al., *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, ICLR 2021

Analysis: Learned Position Embedding



Conclusion: Learns intuitive local structures, but also deviates from locality in interesting ways