Transformer Architectures

Nikola Milosavljević
Senior Software Engineer

Microsoft | Development Center Serbia

Overview

- Recurrent neural networks overview
- Transformer architecture for machine translation
  - Attention mechanism
- Transformer-based language models
- Transformer-based image classification

Machine translation

- Motivated development of transformers

Source	Reference
Une fusillade a eu lieu à l'aéroport international de Los Angeles.	There was a shooting in Los Angeles International Airport.
Cette controverse croissante autour de l'agence a provoqué beaucoup de spéculations selon lesquelles l'incident de ce soir était le résultat d'une cyber-opération ciblée.	Such growing controversy surrounding the agency prompted early speculation that tonight's incident was the result of a targeted cyber operation.

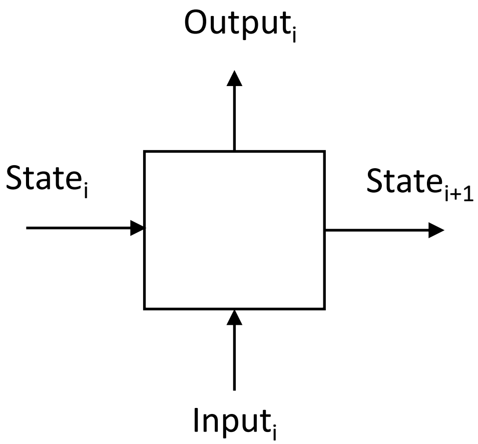
Artetxe *et al.* Unsupervised Neural Machine Translation (2018)

Recurrent neural networks (RNNs)

- Used for problems on sequences

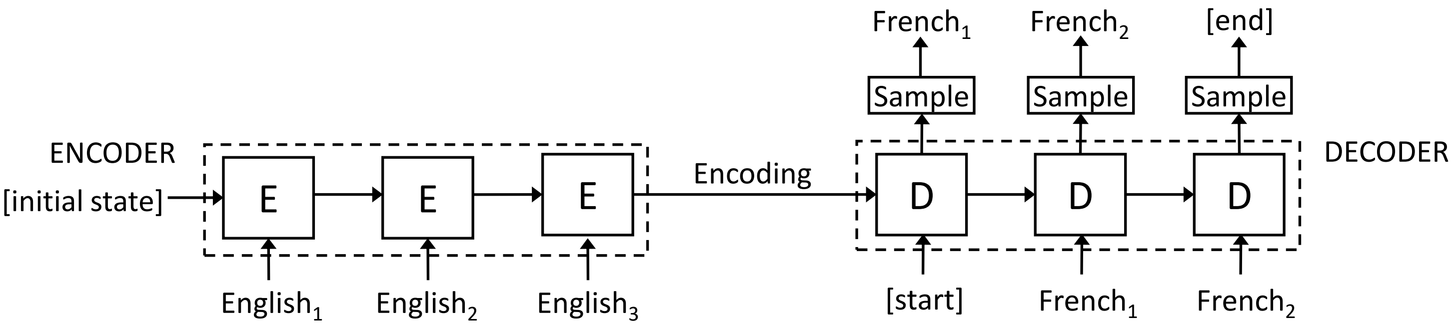
- Define a basic neural network unit (RNN cell)
- Apply the RNN cell to each sequence element

- Can be applied to sequence of any length
- Additional input/output: state vector
  - Captures information about previous elements



Encoder-decoder RNN

- Encoder RNN
  - Consumes input token by token
  - Does not produce output
  - Creates an encoding of input
- Decoder RNN
  - Consumes the encoding
  - Creates output token by token
  - Autoregressive decoding: input in current step is output of the previous step



Sutskever *et al.* Sequence to Sequence Learning with Neural Networks (2014)

Cho *et al.* Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation (2014)

Properties of RNNs

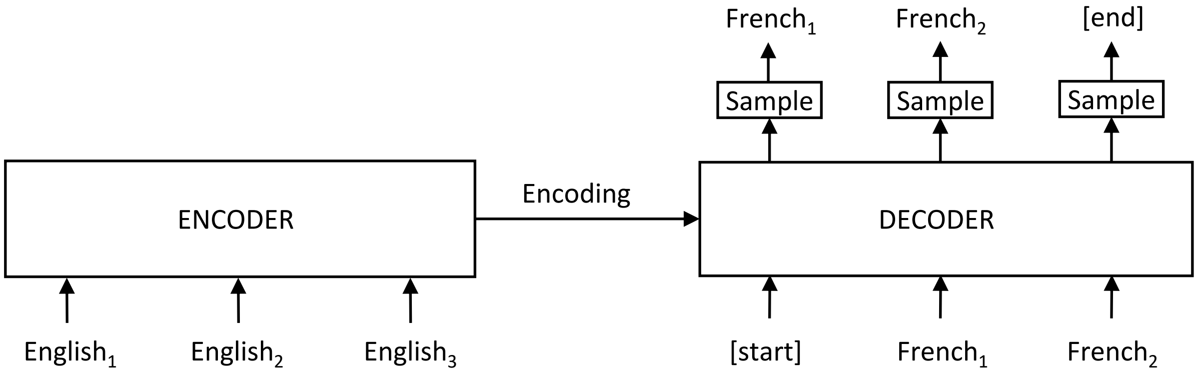
× Serial connections between RNN cells

- Cannot parallelize training and inference
- Hard to capture long-range dependencies
- Hard to backpropagate gradients (“vanishing gradient”)

✓ Compute/memory linear with sequence length

Transformers

- Encoder-decoder architecture with autoregressive decoding
- Proposed by Google in 2017.
- Can be applied to sequence of any length
- Can outperform RNNs on many problems



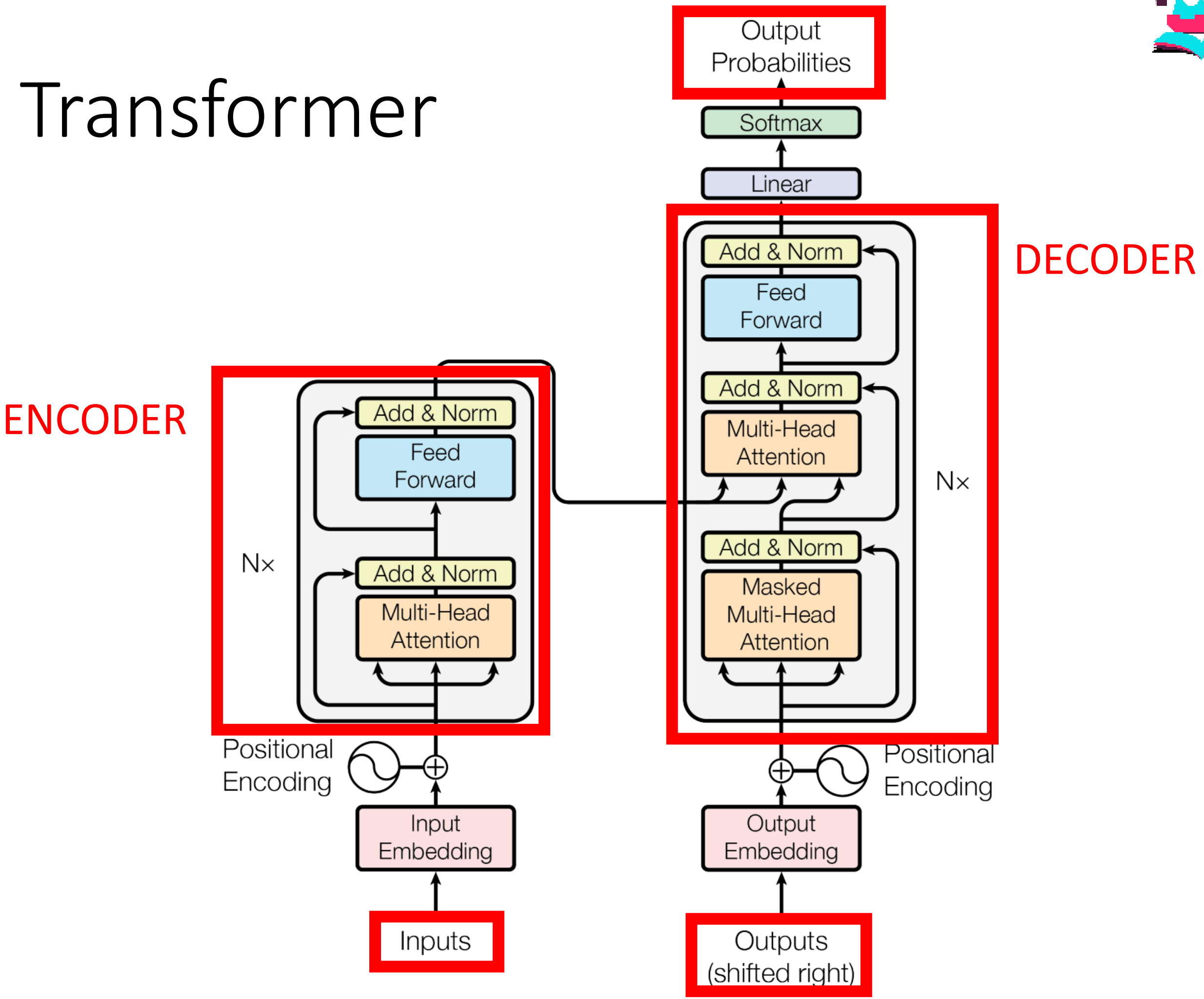
Vaswani et al. Attention is All You Need (2017)

Properties of transformers

✓ No serial connection: any two tokens linked directly

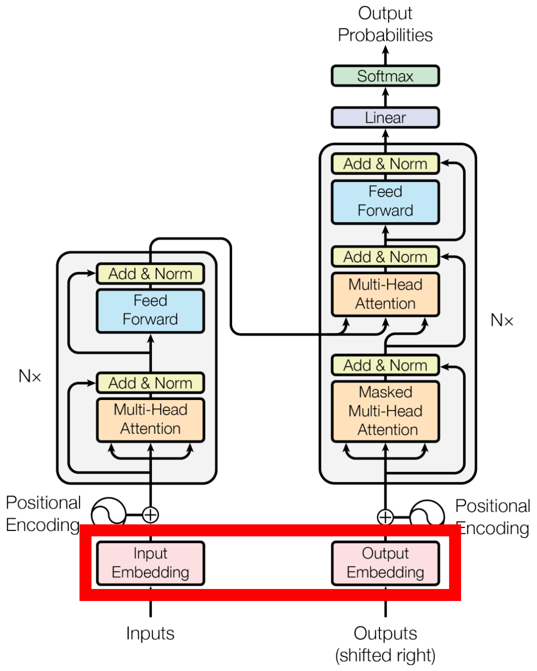
× Compute/memory quadratic in sequence length

The Transformer



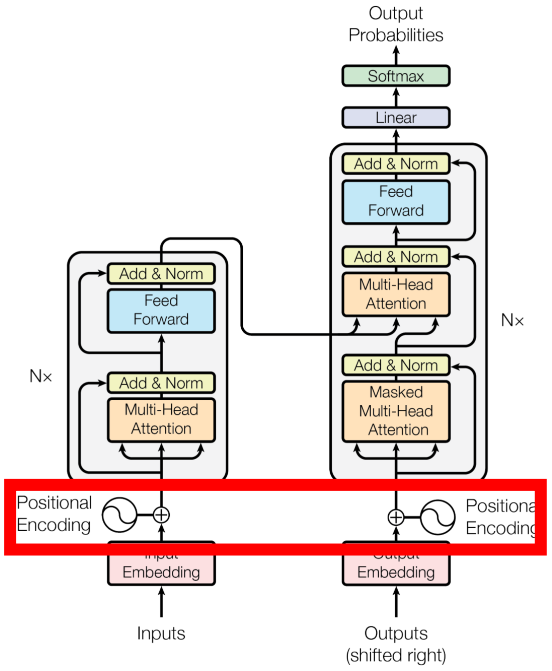
Token embeddings

- Maps a token in vocabulary to a vector
  - Represents the meaning
- Dimension is called model dimension
  - Preserved throughout the network
- Token embeddings are learned during training
  - Represent by a matrix, initialize randomly, optimize by gradient descent



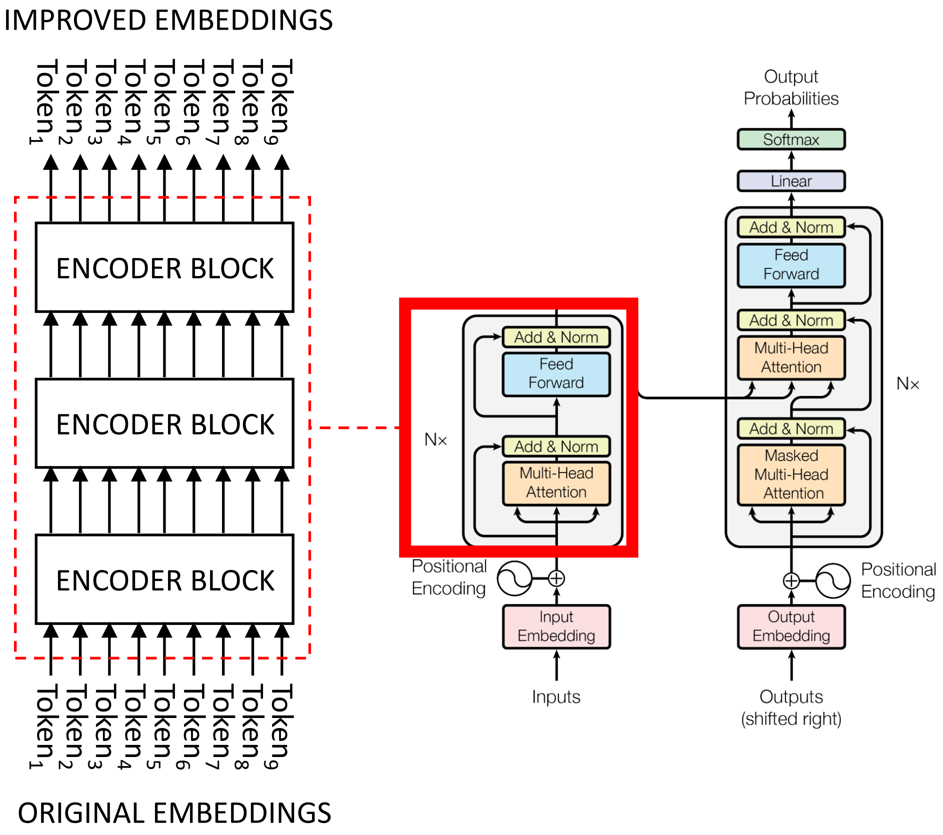
Position embedding

- Maps a position in input sequence to a vector
- Elementwise added to token embedding
- Adds position information to token embedding
  - Prevents permutation-equivariance
- Position embeddings are learned during training
  - Except in the original paper



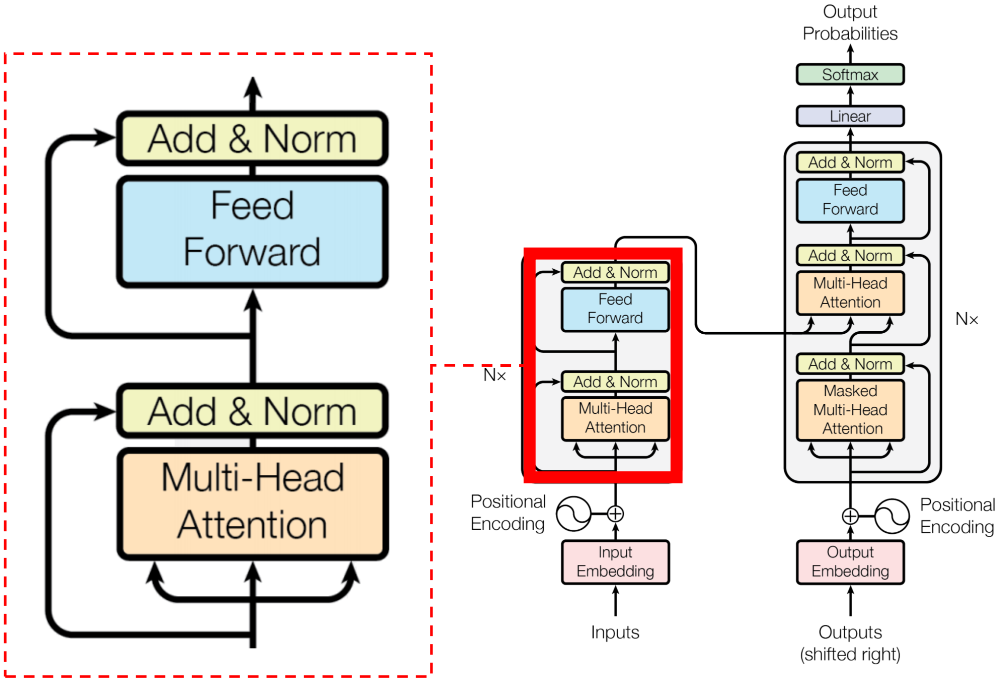
Encoder

- Stacked encoder blocks
  - Same architecture
  - Independent weights
- Input/output of encoder block are embeddings of input (source language) tokens
- Each block improves embeddings by adding context from other input tokens



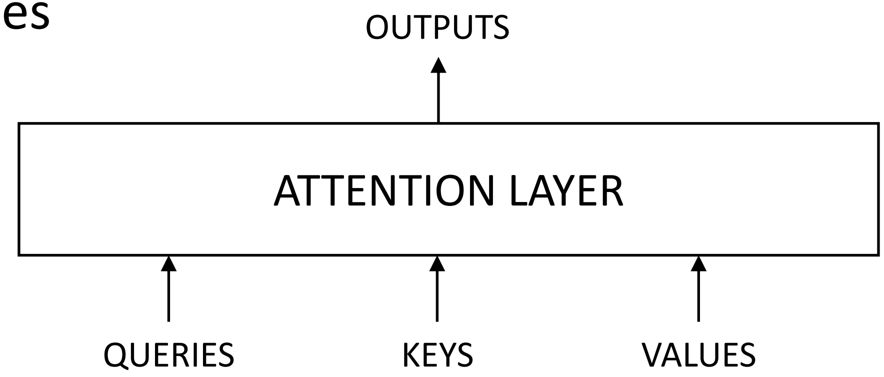
Encoder block

- Attention layer
  - Adds context to input token representations
- Addition + normalization layers
  - Helps convergence (see ResNets)
- Fully connected layers
  - Applied position-wise



Attention

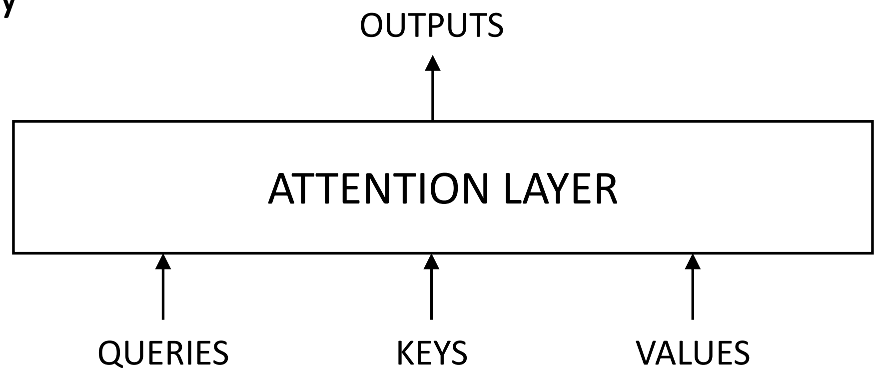
- Older than transformers (used with RNNs too)
- Inputs: queries, keys, values
    - All three are sequences of token embeddings
    - Keys and values form a dictionary data structure (must be equally many)
- Outputs: results of dictionary queries



[Bahdanau, Cho, Bengio: Neural Machine Translation by Jointly Learning to Align and Translate (2014)]

“Soft” dictionary

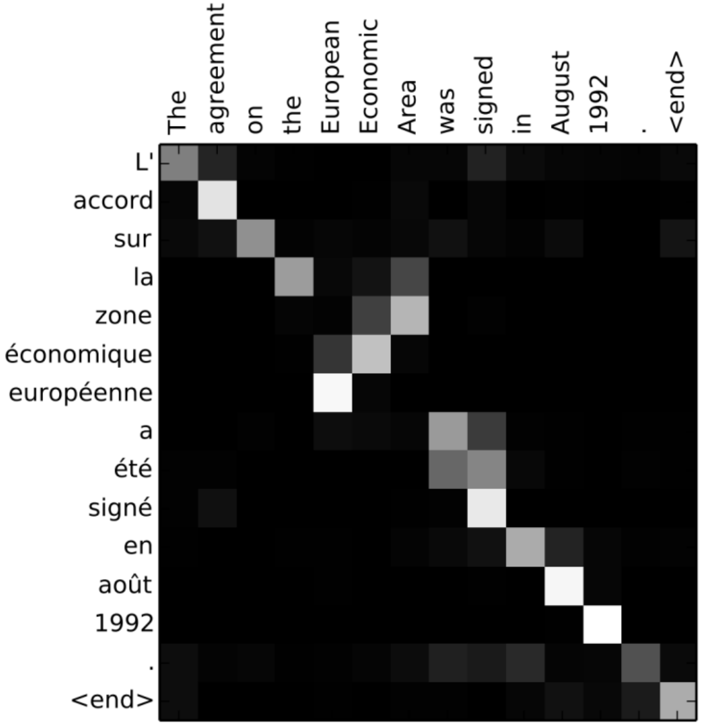
- “Standard” dictionary
  - Query and key match or they don’t
  - Result is the value of the matching key
- “Soft” dictionary
  - Compute matching score for every key
  - Each value influences the result according to its score
  - The scores are called attention values



[Bahdanau, Cho, Bengio: Neural Machine Translation by Jointly Learning to Align and Translate (2014)]

Visualizing attention

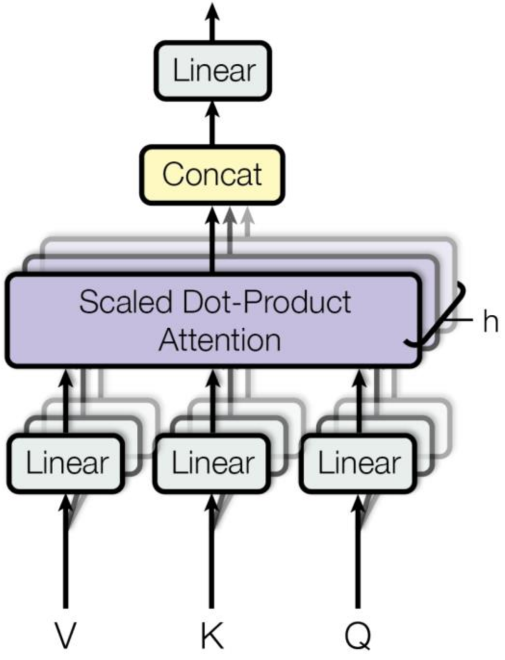
- Attention values can be visualized to show input/output dependencies



[Bahdanau, Cho, Bengio: Neural Machine Translation by Jointly Learning to Align and Translate (2014)]

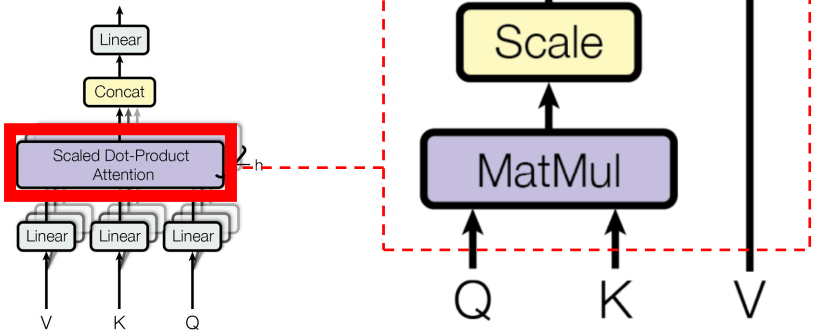
Attention implementation

- h parallel branches ("heads")
- Each "head" focuses on different patterns in input
  - By training different linear layers
- Results of all "heads" combined at the end
- The actual "soft" dictionary logic happens in the scaled dot-product attention block



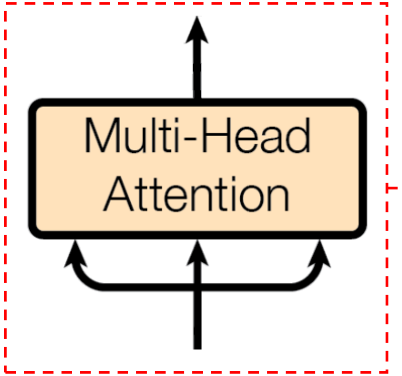
Scaled dot-product attention

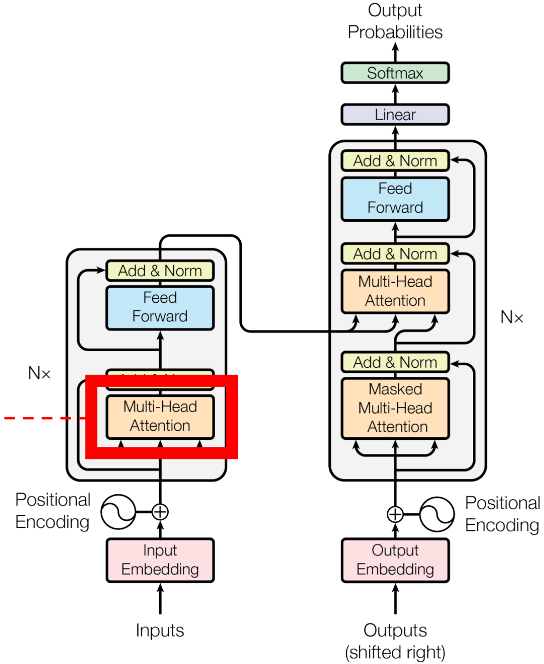
- Attention values are scaled dot-products of queries and keys
  - First MatMul computes query-key dot-products
  - Scaling is for numerical reasons
  - SoftMax computes attention values in the [0, 1] range
  - Second MatMul computes weighted sums



Self-attention

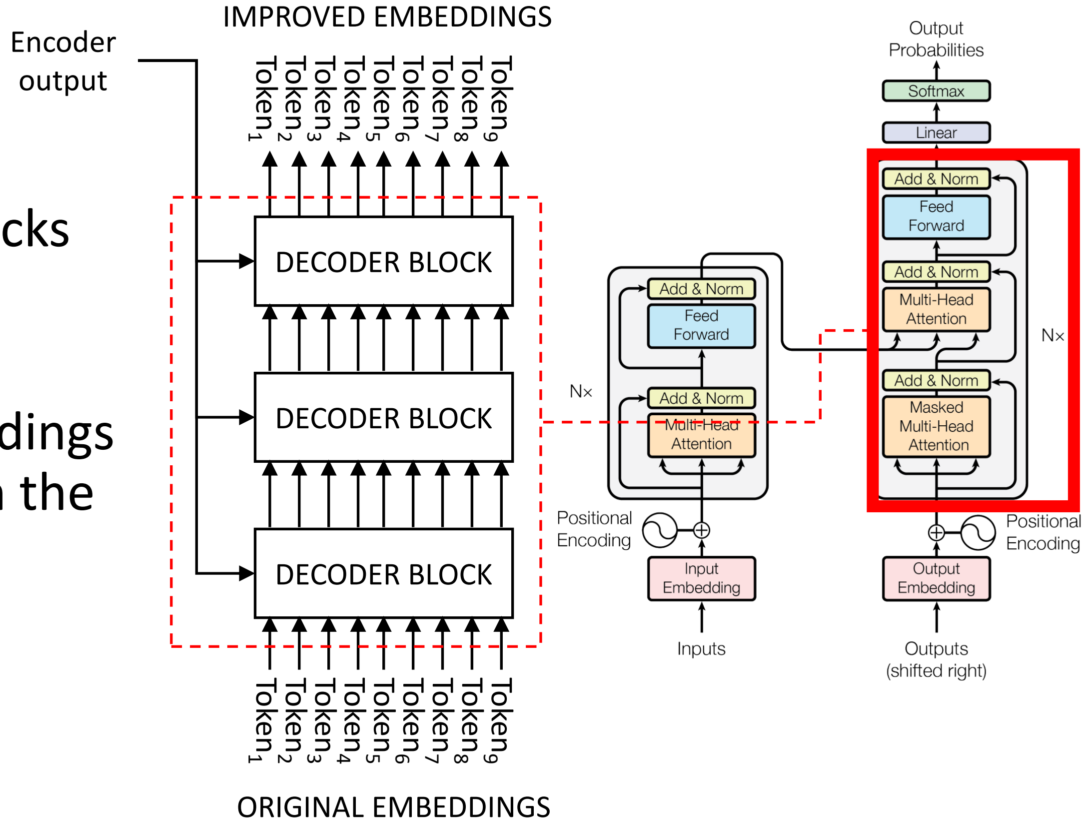
- Queries, keys, values are the same set of vectors
- Input tokens “attend to themselves”





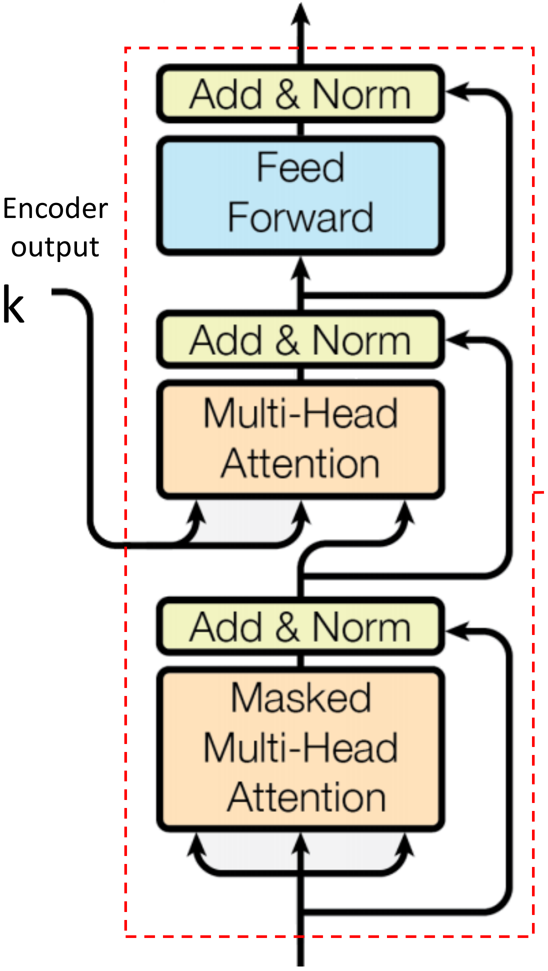
Decoder

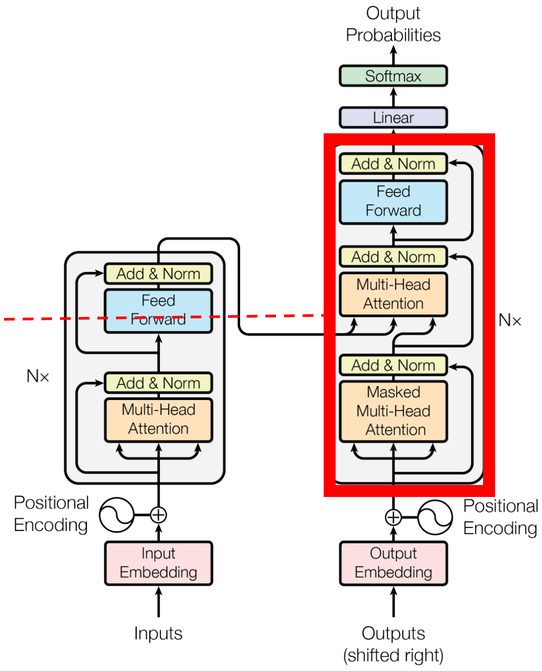
- Very similar to the encoder
- Input/output of decoder blocks are embeddings of output (target language) tokens
- Each block improves embeddings by adding context from both the decoder and the encoder



Decoder block

- Very similar to encoder block
- Cross-attention
  - Queries come from decoder
  - Key-value pairs come from encoder
- Masked attention

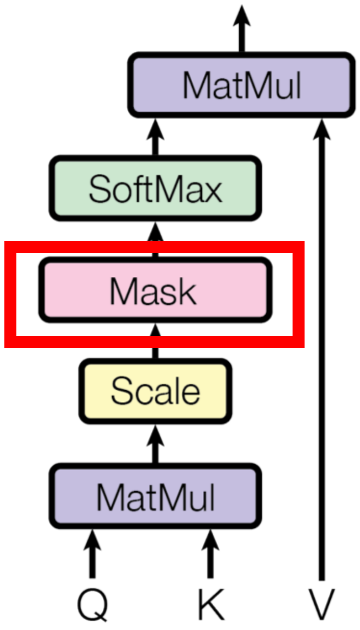




Masked attention

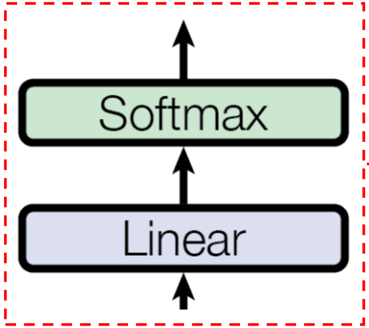
- Prevent a token from attending to its right context
  - During evaluation, right context is not available
  - During training, right context contains the target

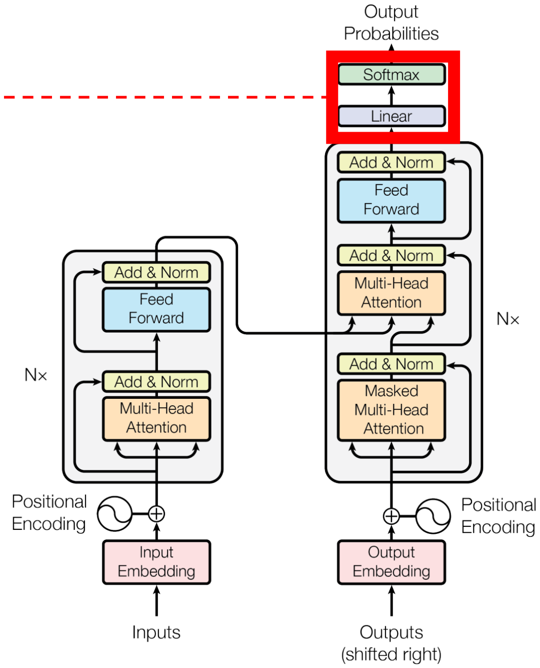
- Masking step within the scaled dot-product
  - Explicitly zeroes out attention values



Final classifier

- Consists of a linear layer and a softmax layer
- Converts token embeddings into probability distribution over the vocabulary





Press and Wolf. Using the output embedding to improve language models (2016)

Language modeling

- Task: predict a token given context
  - Left context (preceding tokens)
  - Bidirectional context (preceding and following tokens)
- Base for solving other NLP problems through transfer learning

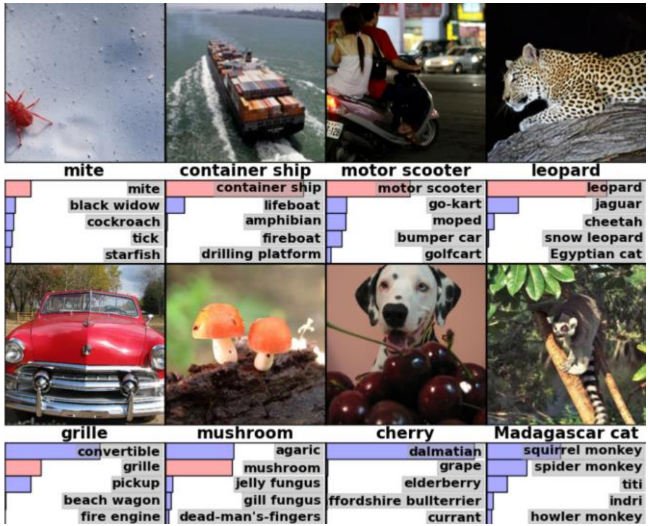
Jay Alammar: The Illustrated GPT-2 (Visualizing Transformer Language Models) (2019)

Transfer learning

- Train for one task (pretraining)
  - Lots of data available
- Finetune for another task (downstream task)
  - Less data available
- Pretrained network's representation is useful for the downstream task
- Replace the last few layers of the pretrained network to match the downstream task
- “Freeze” the remaining layers during finetuning

Transfer learning in CV

- Pretraining task: image classification
- Architecture: convolutional neural network
- Pretraining dataset: ImageNet
  - Web images containing one main object
- Downstream tasks: object detection, semantic segmentation, action recognition…



[ImageNet Large Scale Visual Recognition Challenge (ILSVRC)](#)

[Krizhevsky, Sutskever, Hinton. ImageNet Classification with Deep Convolutional Neural Networks (2012)](#)

Transfer learning in NLP

- NLP’s “ImageNet moment”
- Pretraining task: language modeling
  - Has a lot in common with many NLP tasks
  - Allows self-supervised pretraining
- Architecture: transformers
- Pretraining data: unlabeled internet text
- Downstream tasks: sentiment analysis, sentence similarity, question answering, summarization…

[Sebastian Rudder. NLP's ImageNet moment has arrived (2018)]

GPT

- GPT = **G**enerative **P**re**T**raining
- Published by OpenAI in 2018-2020
- Task: language modeling with left context
  - Of interest both for text generation and as a pretraining task
- GPT-1, GPT-2, GPT-3...

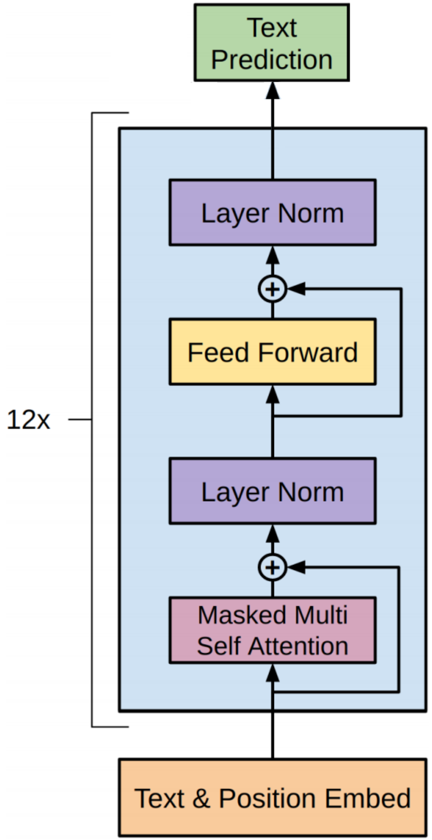
Radford, Narasimhan, Salimans, Sutskever: Improving Language Understanding by Generative Pre-Training (2018)

Radford et al. Language Models are Unsupervised Multi-task Learners (2019)

Brown *et al.* Language Models are Few-Shot Learners (2020)

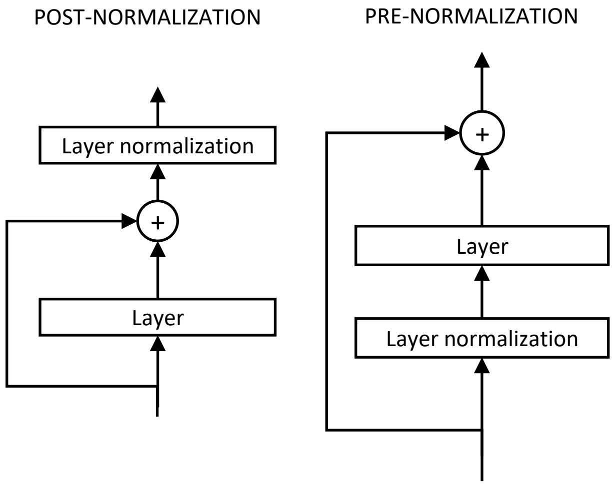
GPT architecture

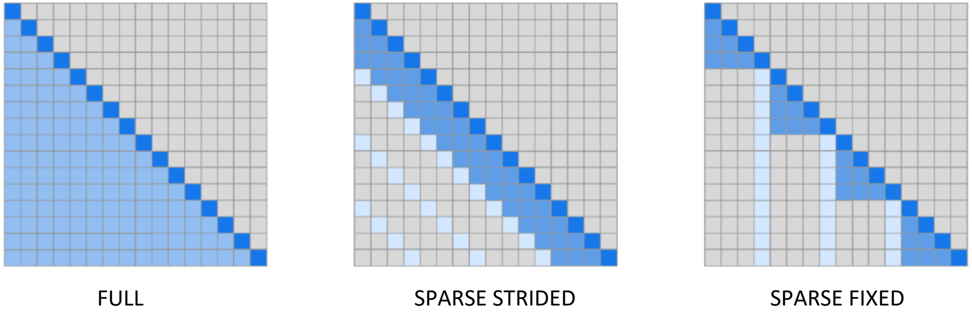
- Based on transformer decoder
  - No cross-attention layer
  - Masked attention forces left context
- Training
  - Input: text passage
  - Output: the same passage shifted left



Subsequent improvements

- Bigger networks
- Data curation
- Scaled down initialization
- Pre-normalization
  - Shuffle layer normalization around
- Sparse attention
  - Replace full attention layer by a constant number of sparse attention layers





GPT-3 training data and models

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}
GPT-3 Small	125M	12	768	12	64
GPT-3 Medium	350M	24	1024	16	64
GPT-3 Large	760M	24	1536	16	96
GPT-3 XL	1.3B	24	2048	24	128
GPT-3 2.7B	2.7B	32	2560	32	80
GPT-3 6.7B	6.7B	32	4096	32	128
GPT-3 13B	13.0B	40	5140	40	128
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128

BERT

- BERT = **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
- Proposed by Google Research in 2018
- Interested in language modeling as a pretraining task
- Pretraining on two tasks
  - Bidirectional language modeling
  - Next sentence prediction

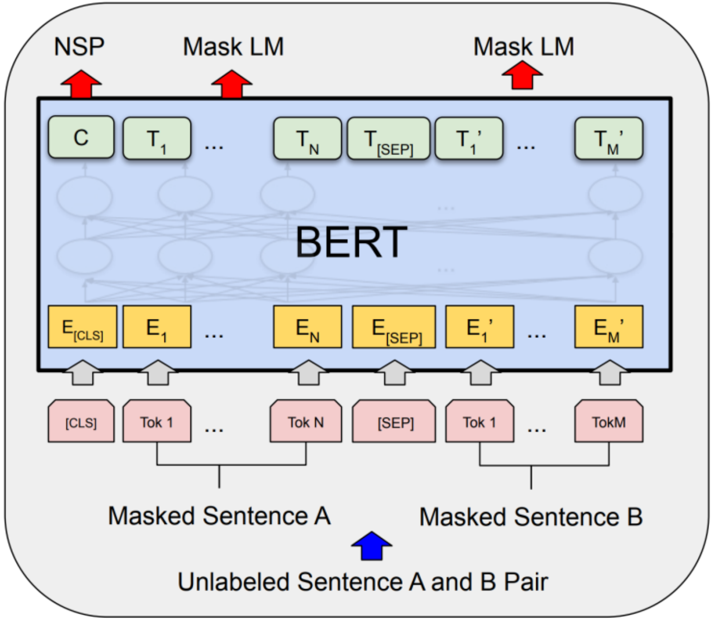
Devlin, Chang, Lee, Toutanova: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2018)

Pretraining tasks

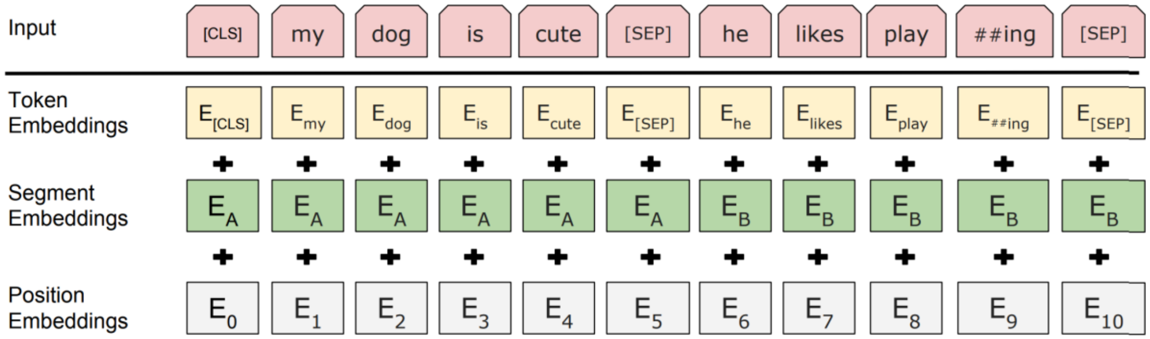
- Masked language model (MLM)
  - Predict a random subset of masked tokens
  - Masking enables training a bidirectional model
    - If allowed to look both left and right, then target token must not be in the input
- Next sentence prediction (NSP)
  - Given sentences A and B, predict if B is the continuation of A
  - Captures semantic relationship between sentences

BERT architecture

- Based on transformer encoder
  - Recall: attention not masked
- Training input: two masked sentences
  - Mask random 15% of tokens
  - Consecutive sentences 50% of the time
  - Special tokens [CLS], [SEP], [MASK]
- Training output:
  - MLM: Unmasked version of input
  - NSP: true/false in [CLS] position



Segment embedding



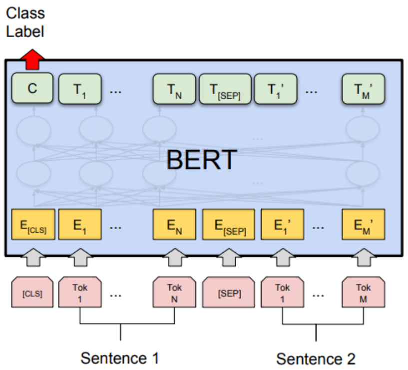
Finetuning

- Goal: solve another NLP task using a pretrained language model

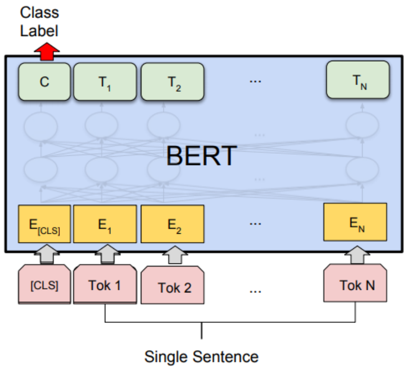
- Convert input example into a single sequence
  - Introduce special tokens if needed
- Modify one or more transformer outputs
  - Typically, just classification heads (linear + softmax)

Finetuning BERT

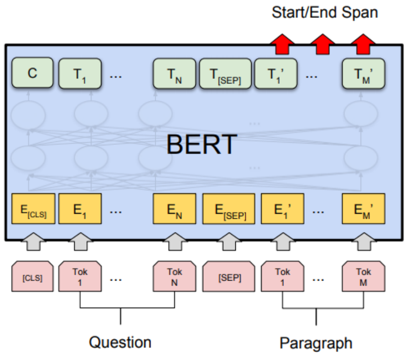
- Classification tasks: use output of a special token
- Tagging tasks: use outputs of each individual token



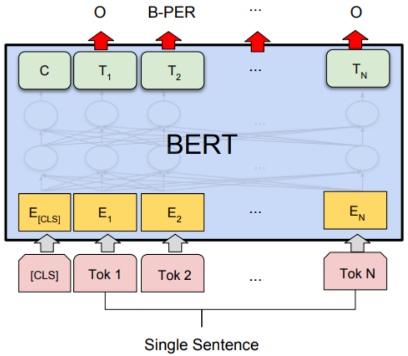
2-sentence classification



1-sentence classification



Question answering



Named entity recognition

Vision Transformer (ViT)

- Proposed by Google Brain in 2020
- Outperforms convolutional networks on several image classification benchmarks
- Pretraining on image classification
  - Very large dataset
- Finetuning on other image classification datasets

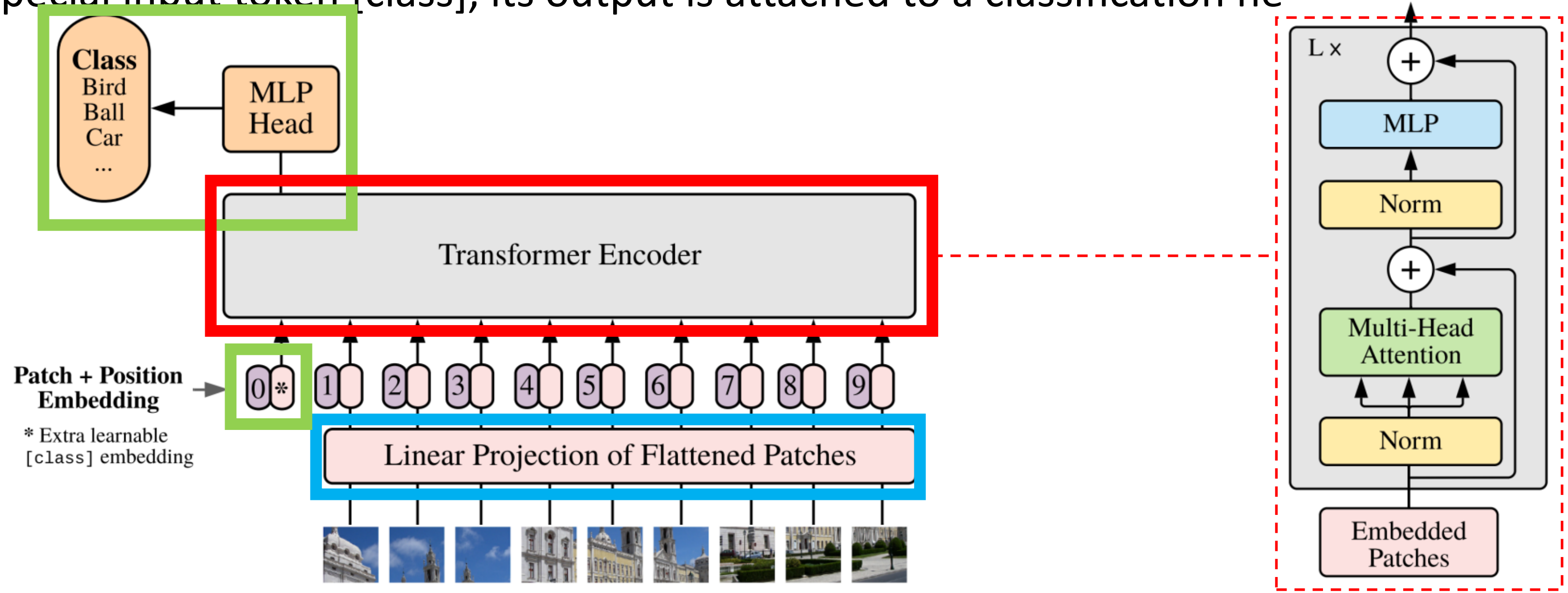
Dosovitskiy *et al*. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale (2020)

Transformers for image data

- Cannot unroll pixels into a sequence
  - Large number of pixels, complexity of attention
- Solution
  - Divide image into a grid of fixed-sized patches
  - Arrange patches into a sequence

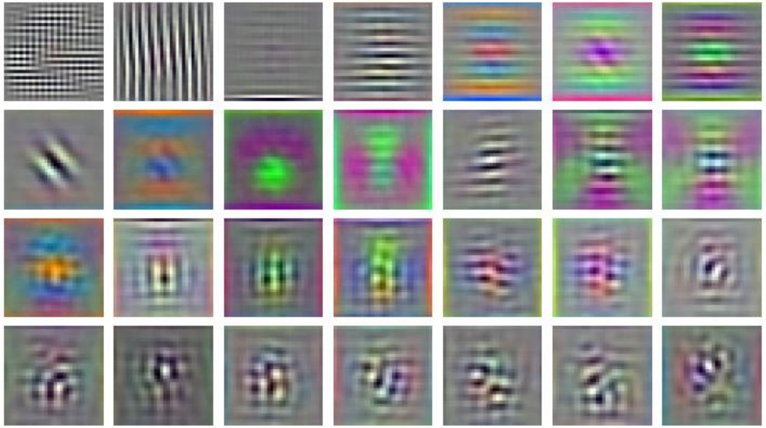
Training

- Architecture: slightly modified transformer encoder
    - A patch is unrolled into vector before applying embedding matrix
    - Special input token [class], its output is attached to a classification network



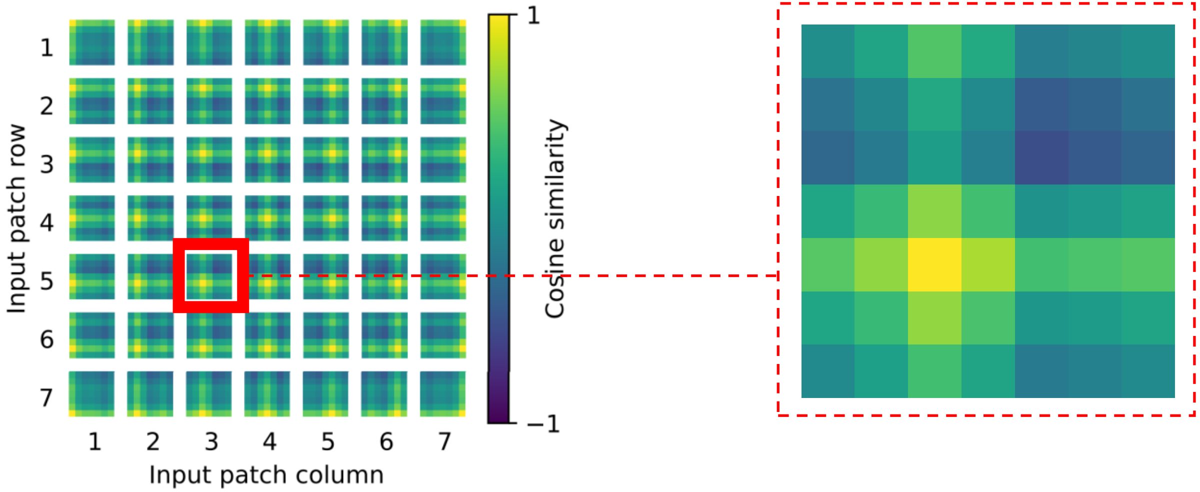
Visualizing patch embedding

- Top principal components of the embedding matrix (rows)
  - Main patterns that the network looks for in a patch
  - Resembles convolutional filters
- Replacing with convolutional network features does not help



Visualizing positional embedding

- Cosine similarity between vectors of patches (i, j) and (i', j')
  - 2D relationship among patches is learned automatically, even though patches are presented to the network as a 1D sequence
- Explicit 2D positional embedding in the network does not help



Pretraining data

- Pretraining transformer on large datasets is needed to beat convolutional network performance

Dataset	Images	Classes	Note
ImageNet ILSVRC 2012	1.3 million	1000	Web images, clean labels
ImageNet-21k	14 million	21841	Web images, clean labels
JFT300M	303 million	18291	Google image search, noisy labels

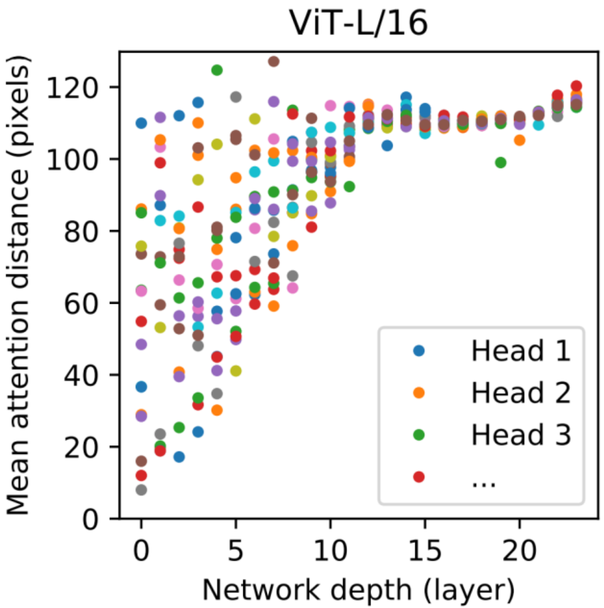
Visualizing attention

- Can be done for images too

[Abnar and Zuidema. Quantifying attention flow in transformers (2020)]

Visualizing attention distance

- Analogous to receptive field for convolutional networks
- Some heads attend to long distances already in early layers



Summary

- Originally developed for machine translation
- Address issues with RNNs
  - Parallel processing of tokens
  - Direct connections between tokens
- Outperforming RNNs in NLP, more recently in computer vision
  - Transfer from language modeling
  - Very few image-specific modifications
- State of the art models are large and require lots of data