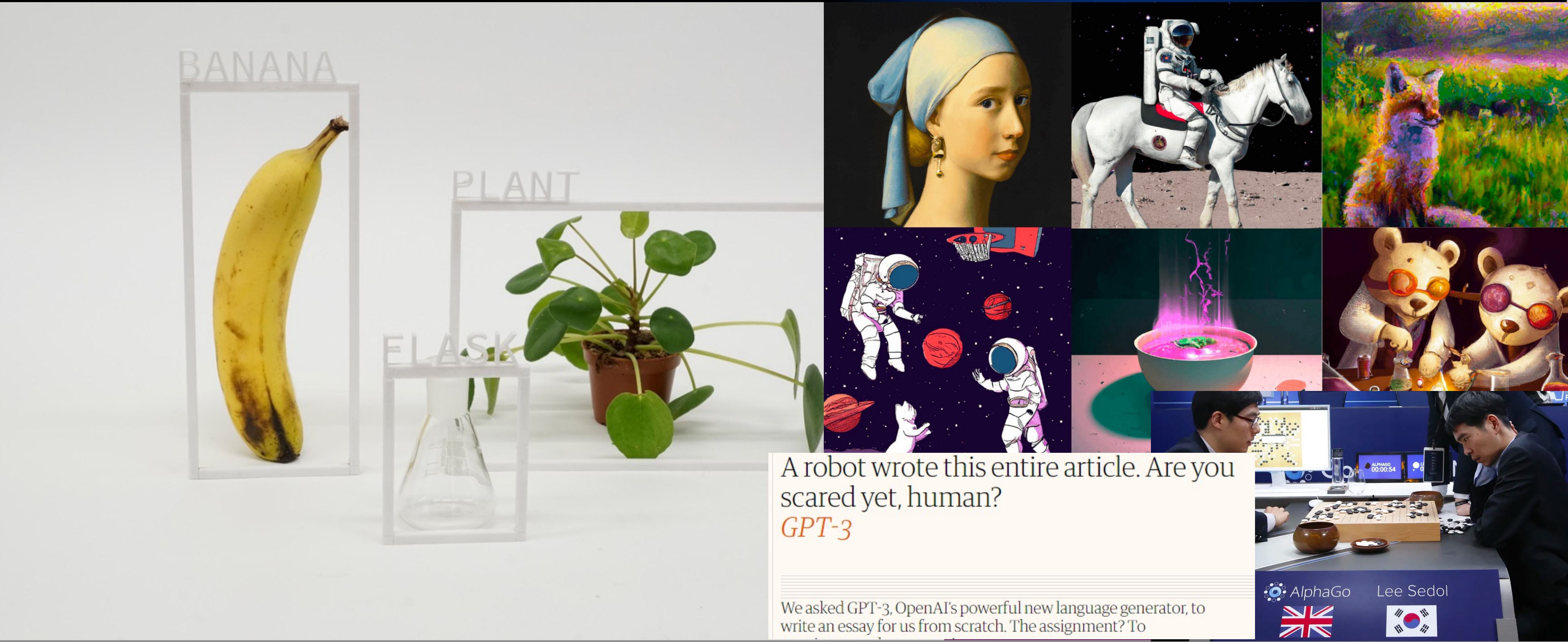


Lecture 7: Attention & Transformers

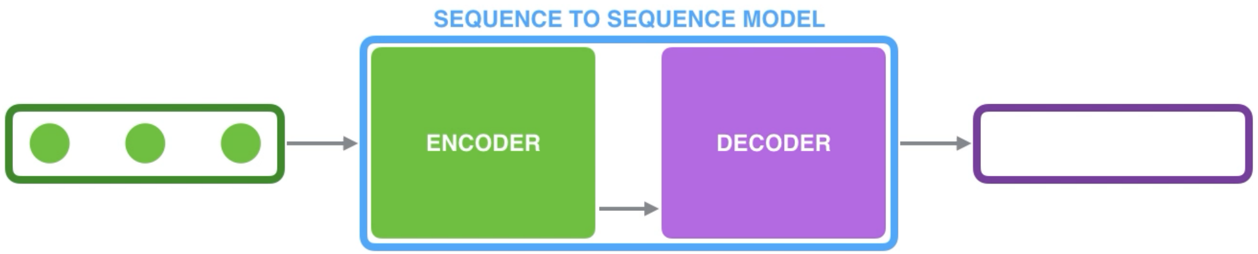
Ivona Najdenkoska
Yuki M. Asano

Lecture overview

- Seq2seq
- Seq2seq for NMT
- Issues of Seq2seq
- Attention mechanism
- Self-attention mechanism
- Transformer
- Language Transformers
- Multimodal Transformers
- Vision Transformer
- Summary

Seq2seq models

- Seq2seq models are encoder-decoder models that consist of 2 parts:



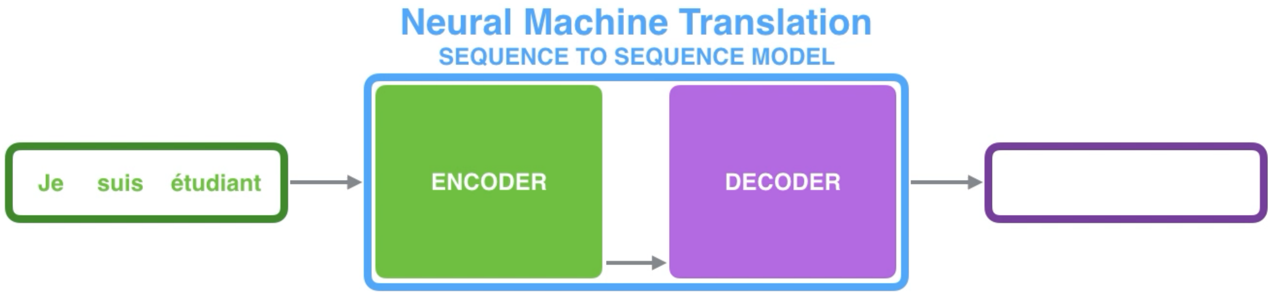
  - Encoder: takes a variable-length sequence of elements as the input and transforms it into a context representation with a fixed-size.
  - Decoder: maps the encoded state of a fixed size to a variable-length sequence of elements.

- Success in machine translation, text summarization and image captioning.

https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Neural machine translation with a seq2seq model

- An encoder neural network reads and encodes a source sentence into a fixed-length context vector.
- A decoder then outputs a translation from the encoded context vector.



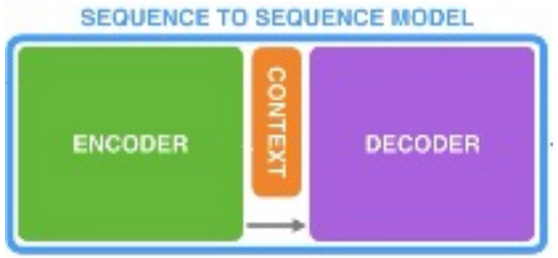
Defining seq2seq for NMT[1]: Encoder

- An encoder encodes the input sentence, a sequence of vectors $x = (x_1, x_2, \dots x_{Tx})$, into a context vector c.
- A common approach is to use an RNN/LSTM, such that:

$$h_t = f(x_t, h_{t-1}),$$

$$c = q(\{h_1, \dots h_{Tx}\}).$$

  - $h_t \in \mathbb{R}$ is a hidden state at time-step t,
  - c is the context vector generated from the sequence of hidden states,
  - f and q are nonlinear functions.



$h_t = f(x_t, h_{t-1}) \quad c = q(\{h_1, \dots h_{Tx}\})$

[1]Neural Machine Translation by Learning to Align and Translate, Bahdanau et al. ICLR (2015)

Defining seq2seq for NMT: Decoder

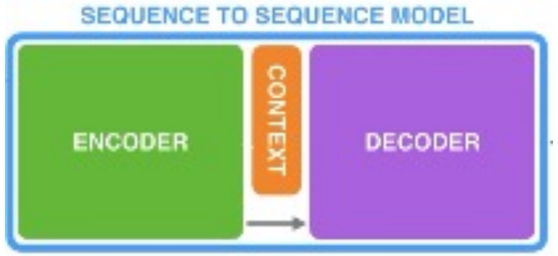
- The decoder is trained to predict the next word y_t, given the context vector c and all the previously predicted words $\{y_1, \ldots, y_{t-1}\}$.
- It defines a probability over the translation y by decomposing the joint probability into the conditionals:

$$p(y) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \ldots, y_{t-1}\}, \mathbf{c}) \text{, where } y = (y_1, \ldots, y_{Ty}).$$

- With an RNN decoder, each conditional probabilty is modeled as:

$$p(y_t \mid \{y_1, \ldots, y_{t-1}\}, \mathbf{c}) = g(y_{t-1}, s_t, \mathbf{c}) \text{, where}$$

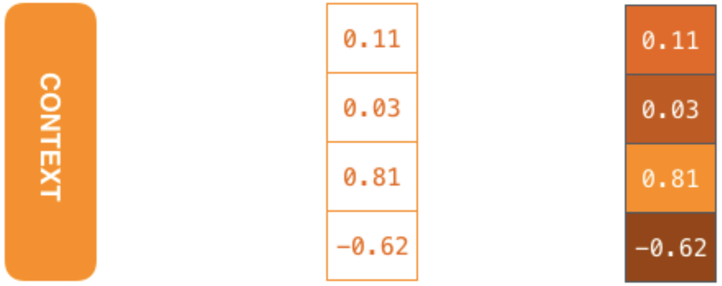
  - g is a nonlinear (multi-layered) function
  - s_t is the hidden state of the RNN.



$h_t = f(x_t, h_{t-1}) \quad c = q(\{h_1, \ldots h_{Tx}\})$

Issue of seq2seq models

- The model needs to compress all necessary information of a source sequence into the fixed-length context vector c.
  - The context vector c is seen as a *bottleneck.*



The context is a vector of floats, and its size is the number of hidden units in the encoder RNN.

- When dealing with long sequences, it is challenging for the model to compress all information into a fixed-length context - due to the vanishing gradient problem.

Attention

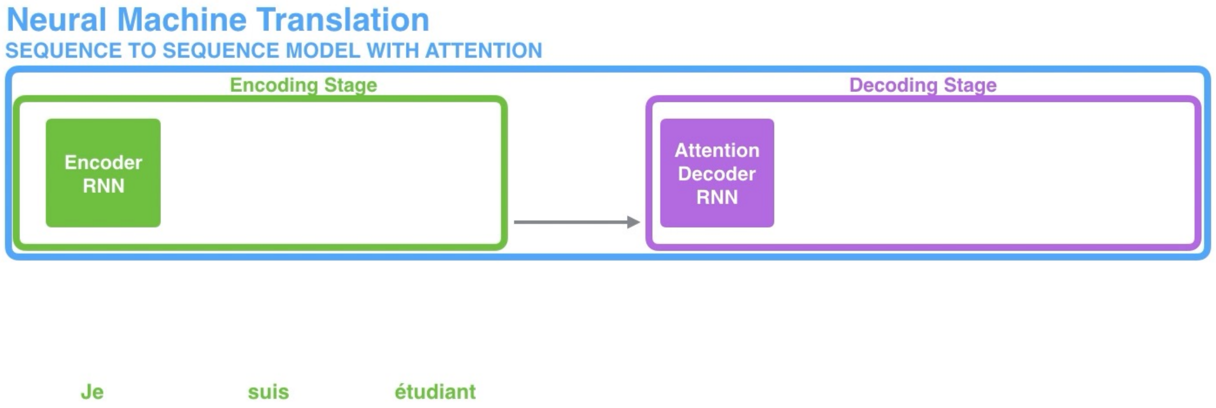
- The **attention** mechanism[1] overcomes the bottleneck issue of encoder-decoder.
- Intuitively it allows the model to focus on relevant parts of the input, while decoding the output.

- In the context of NMT: Each time the model generates a word, it searches for a set of positions in the source sentence where the most relevant information is concentrated.
- **Breakthrough** idea in NLP – as impactful as CNN in computer vision!

[1]Neural Machine Translation by Learning to Align and Translate, Bahdanau et al. ICLR (2015)

Attention

- The encoder–decoder with attention does not need to encode the whole input into a single fixed-length vector.
- It instead encodes the input into a sequence of vectors,
- Then it chooses a subset of these vectors adaptively while decoding the output.
- This frees the model from having to **squash all the information** of the source into a fixed-length vector.



Formal definition of Attention

- Now each conditional probability of the decoder is defined as:

  $$p(y_t|\{y_1, \ldots, y_{t-1}\}, x) = g(y_{t-1}, s_i, ci), \text{ where}$$

  s_i is an RNN hidden state for time i, computed by:

  $$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

- The probability is conditioned on a distinct context vector c_i for each target word y_i (unlike the conventional encoder–decoder).
- The context vector c_i depends on a sequence of annotations $(h_1, \ldots, h_{t-1})$ to which an encoder maps the input sentence.
- Each annotation h_i contains information about the whole input,
  - with a strong focus on the parts surrounding the i-th word of the input sequence.

Formal definition of Attention

- The context vector c_i is computed as a *weighted sum* of these annotations h_j:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j,$$

- where the weight α_{ij} of each annotation h_j is computed by the probability:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

where $e_{ij} = a\left(s_{i-1}, h_j\right)$ is an alignment model,

- It scores how well the inputs around position *j* and the output at position *i* match.
- The alignment model a is parametrized as a feedforward neural net,
- And is jointly trained with all the other components of the model.

Formal definition of Attention

- The weight α_{ij} reflects the importance of the annotation h_j, with respect to the previous hidden state s_{i-1} when deciding the next state s_i and generating y_i.

- This implements a mechanism of attention in the decoder.
- The decoder decides which parts of the source sentence to pay attention to.

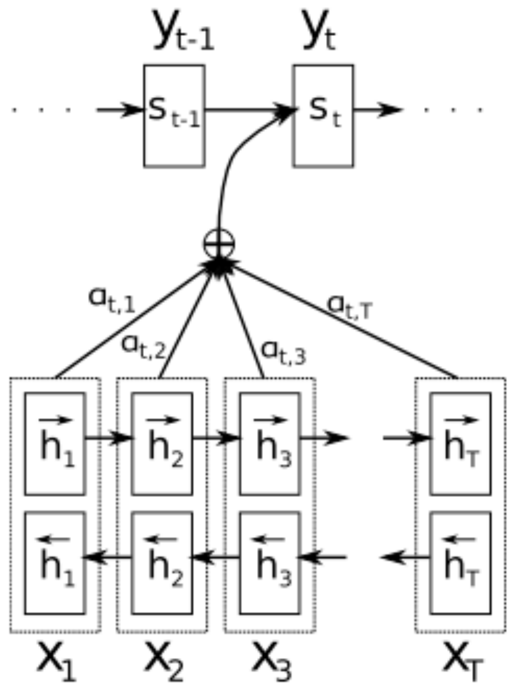
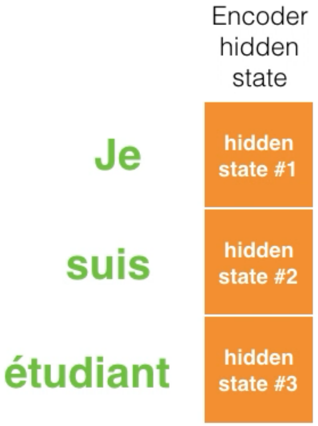


Fig. 1 Graphical illustration of attention model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \dots, x_T)$[1]

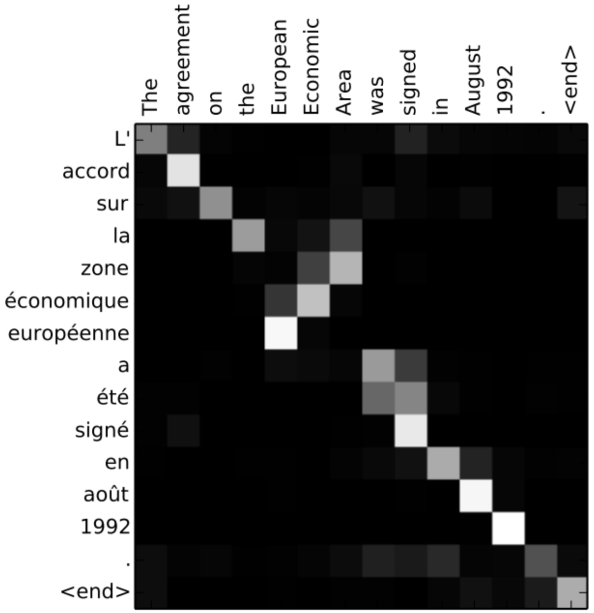
[1]Neural Machine Translation by Learning to Align and Translate, Bahdanu et al. ICLR (2015)

Why attention?

- The encoder is **relived from the burden** of having to encode all information of the input into a fixed-size vector.



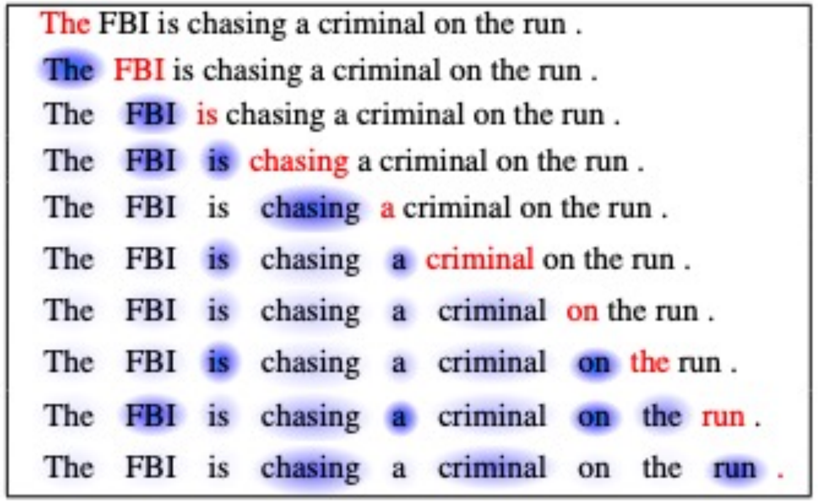
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

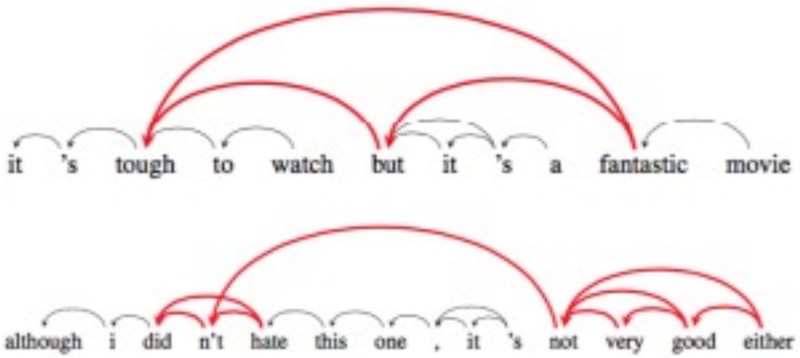


The x-axis and y-axis correspond to the words in the source sentence (English) and the generated translation (French). Each pixel shows the weight α_{ij} of the annotation of the j-th source word for the i-th target word.

Self-attention

- **Self-attention**[1] (or intra-attention): relates parts of sequence with each other.
- Results is a representation of the whole sequence.
- In general terms, it can be seen as an operation on sets of elements.



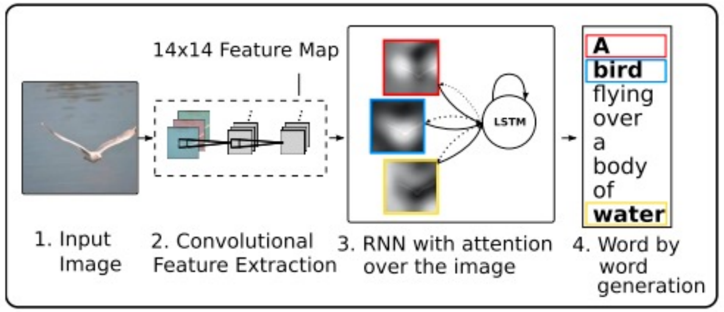


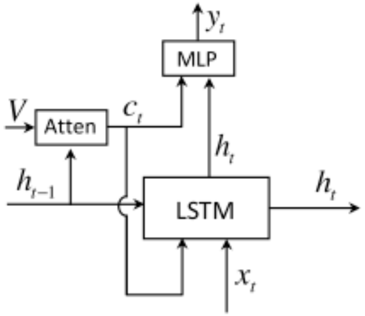
Useful for NLP tasks, such as sentiment analysis

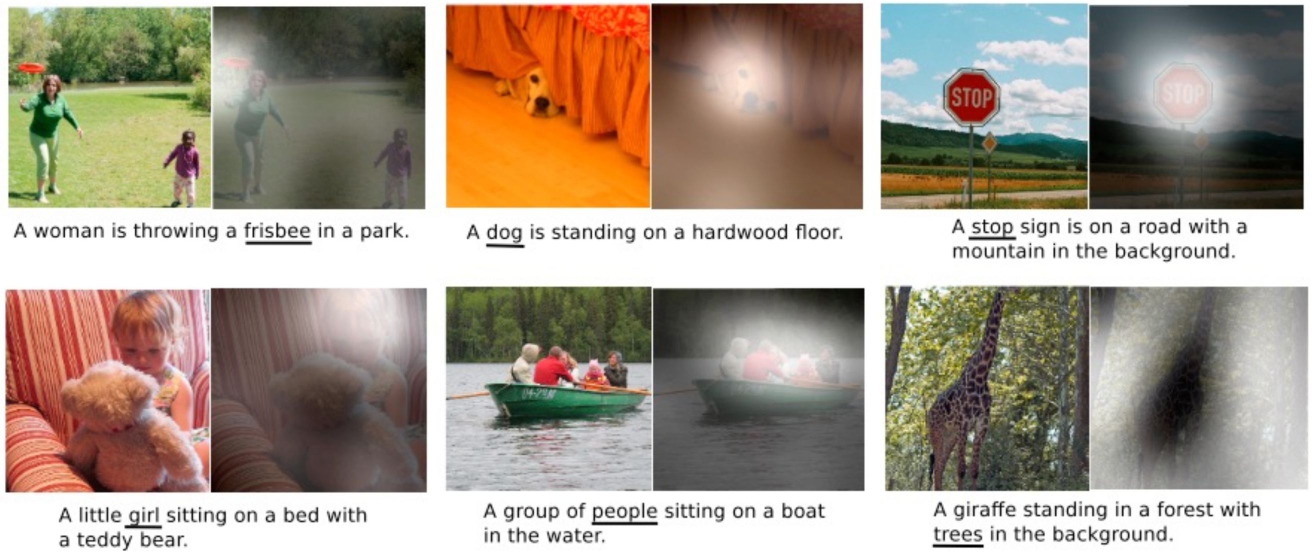
[1] Long Short-Term Memory-Networks for Machine Reading, Cheng et al. (2016)

Paying attention in vision

- Attention is applied to many other tasks, inspired by its success in NMT.
- Example in vision: image captioning[1], where the input is image, and the output is a description of the image.







Examples of attending to the correct object (white indicates the attended regions, underlines indicated the corresponding word)

[1] Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, Xu et al. (2016)

Attention is all you need

- **Transformer**[1]– encoder-decoder model based on (self-)attention mechanisms
  - Without any recurrence and convolutions
  - Referred to as NLP's ImageNet moment
  - It completely changed the landscape of deep learning models!
  - SOTA in NLP tasks and recently in computer vision

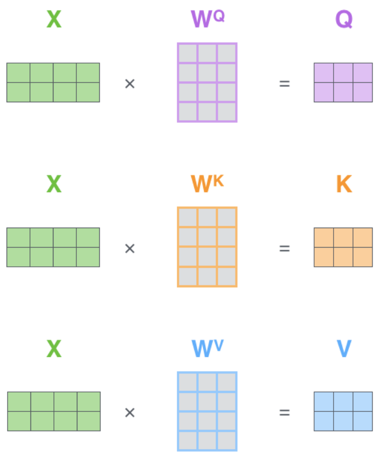
- Important concepts:
  - Queries, keys, values
  - Scaled dot-product attention
  - Multi-head (self-)attention

[1] Attention Is All You Need, Vaswani et al. (2017)

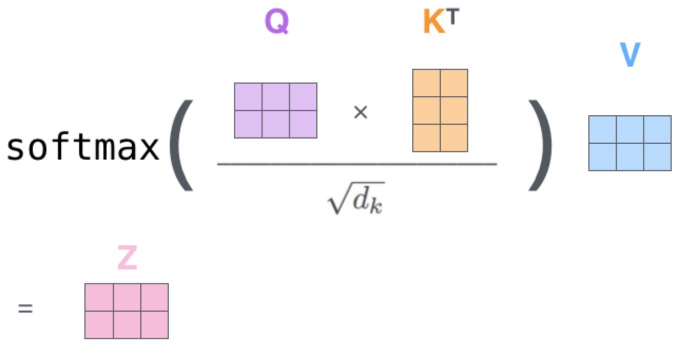
Queries, keys and values

- The Transformer paper redefined the attention mechanism by providing a generic definition based on ***queries, keys, values.***
- Intuition: Use the **query** of the target and the **key** of the input to calculate a matching score,
- These matching scores act as the weights of the **value** vectors.

For self-attention: Q=K=V=X



The input consists of queries and keys of dimension d_k, and values of dimension d_v.



Compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

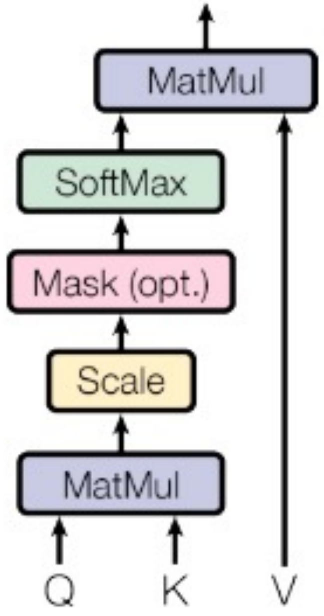
https://jalammar.github.io/illustrated-transformer/

Scaled Dot-Product Attention

- The attention mechanism used in Transformer is referred to as *"Scaled Dot-Product Attention"*.

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

- For large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients.
- To counteract this effect, the dot product is scaled by $\frac{1}{\sqrt{d_k}}$.

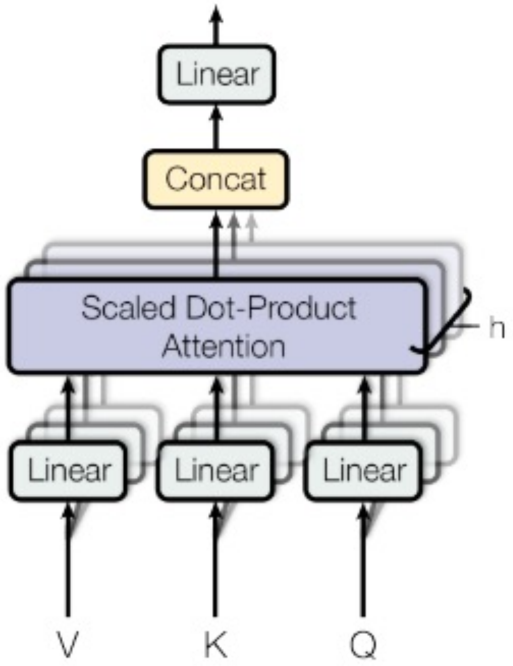


Multi-head attention

- It is beneficial to linearly project the Q, K and V, h times with different, linear projections to d_k, d_k and d_v dimensions.
- The attention function is performed in parallel, on each of these projected versions of Q, K and V.
- These are concatenated and again projected, resulting in the final values.
- This increases the learning capacity of the model.

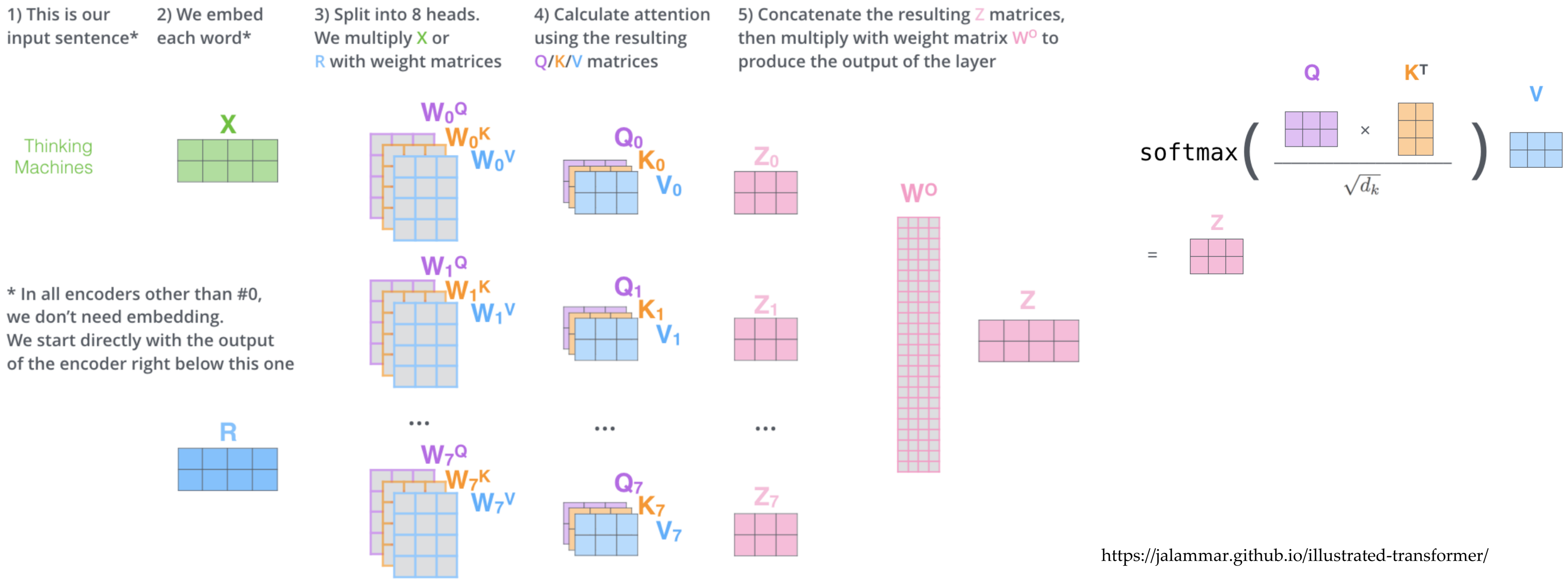
$$MultiHead(Q, K, V) = Concat(head_1, \dots, headh)W^O,$$
$$\text{where } head_i = Attention(QW^Q{}_i, KWKi, VWVi),$$

and the projections are parameter matrices $W^Q{}_i \in \mathbb{R}^{d_{model} \times d_k}$, $W^K{}_i \in \mathbb{R}^{d_{model} \times d_k}$, $W^V{}_i \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O{}_i \in \mathbb{R}^{hd_v \times d_{model}}$.



Multi-head **self-**attention

- For multi-head self-attention: the ***queries, keys, values*** are equal to the input representation or from the previous (encoding/decoding) layer.

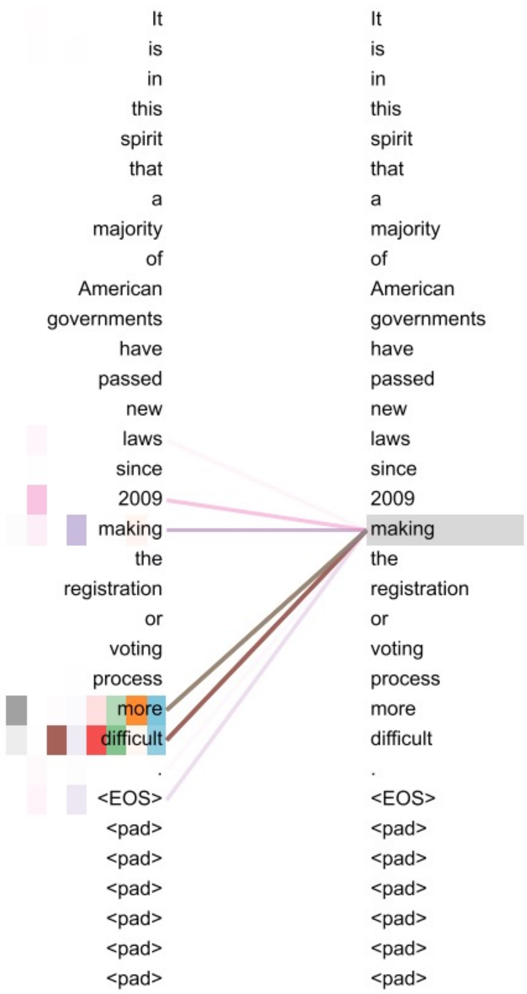


https://jalammar.github.io/illustrated-transformer/

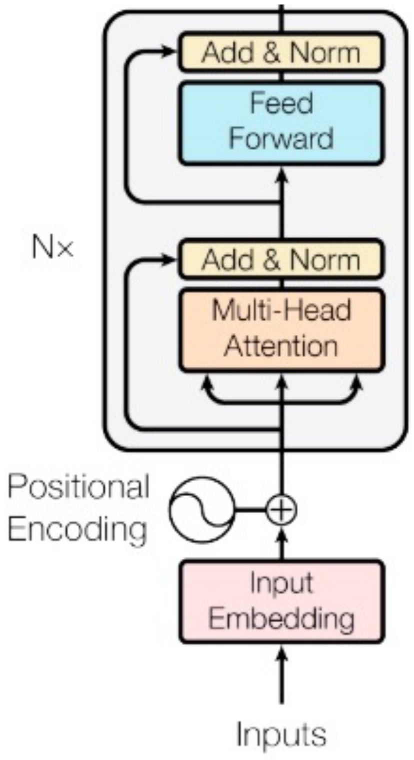
Multi-head **self-**attention

- Many of the attention heads attend to a distant dependency of the verb "*making*", completing the phrase "*making…more difficult*".
- Self-attention can yield more interpretable models.
- Because attention values can be seen as importance weights.

Multi—head self-attention visualization

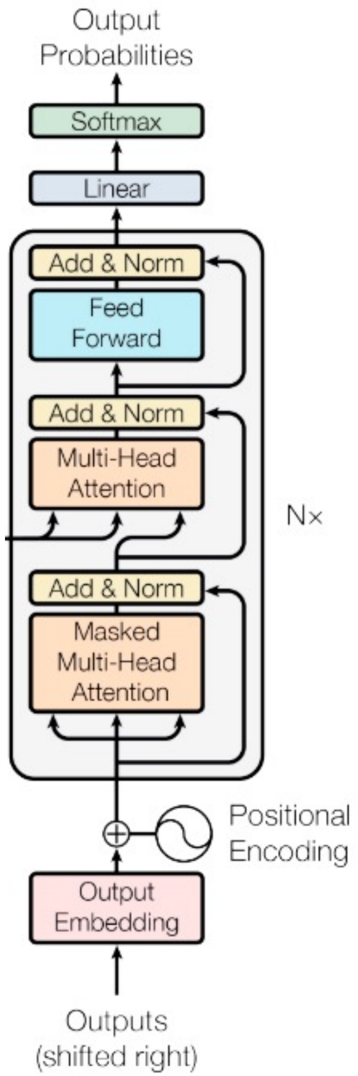


Transformer encoder

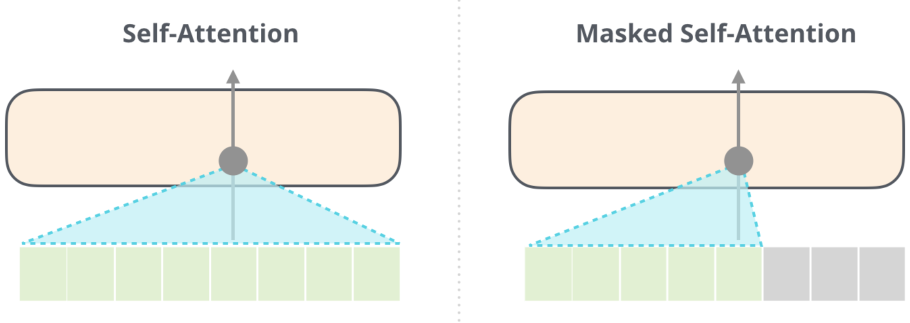


- The encoder consists of N = 6 identical layers
  - Each encoder layer has 2 sub-layers: multi-head attention and fully connected feed-forward network.
- Each sub-layer has a residual connection around it, followed by layer normalization.

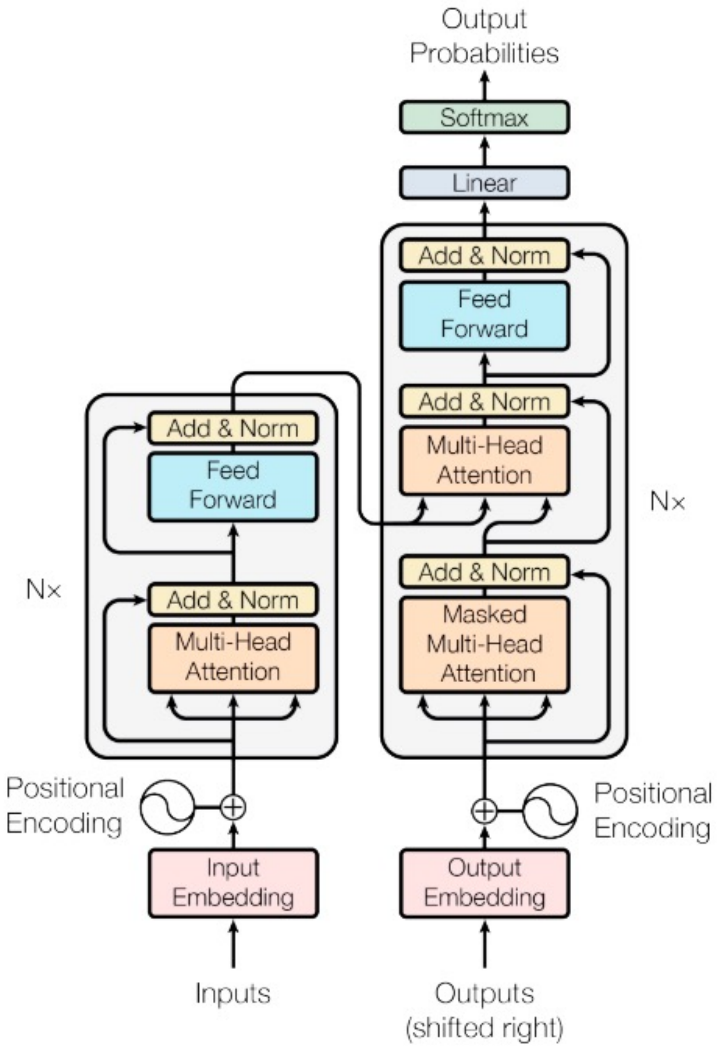
Transformer decoder



- The decoder also consists of N = 6 identical layers
  - A decoder layer is identical to the encoder layer,
  - It has an additional 3rd sub-layer,
  - Which performs multi-head attention over the output of the encoder.
- The masked self-attention sub-layer in the decoder prevents positions from attending to subsequent positions.
  - the predictions for position i can depend only on the known outputs at positions < i.



The full Transformer



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 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

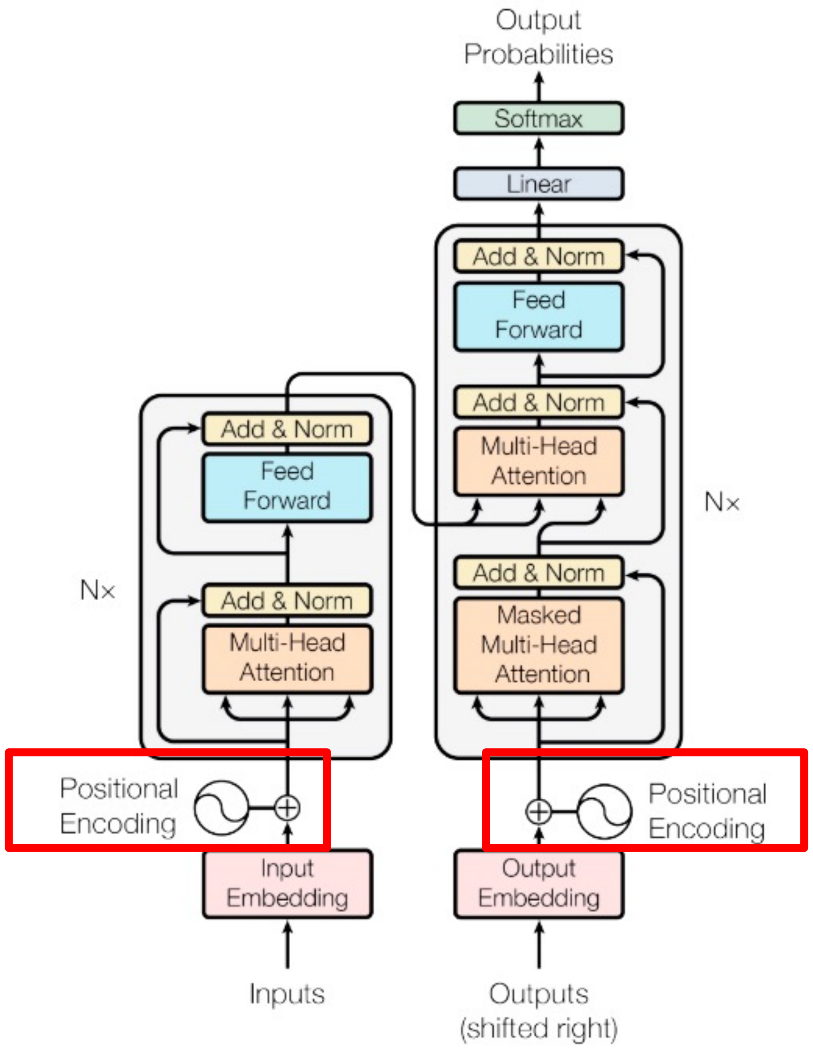
Coding a Transformer (PyTorch): init

```
class Transformer(nn.Module):
    def __init__(self,
            d_model: int = 512, # the number of expected features in the encoder/decoder inputs
            nhead: int = 8, # the number of heads in the multihead-attention models
            num_encoder_layers: int = 6, # the number of sub-encoder-layers in the encoder
            num_decoder_layers: int = 6, # the number of sub-decoder-layers in the decoder
            dim_feedforward: int = 2048):  # the dimension of the feedforward network model

    encoder_layer = TransformerEncoderLayer(d_model, nhead, dim_feedforward, …)
    encoder_norm = LayerNorm(d_model, eps=layer_norm_eps, **factory_kwargs)

    decoder_layer = TransformerDecoderLayer(d_model, nhead, dim_feedforward, …)
    decoder_norm = LayerNorm(d_model, eps=layer_norm_eps, **factory_kwargs)

    self.encoder = TransformerEncoder(encoder_layer, num_encoder_layers, encoder_norm)
    self.decoder = TransformerDecoder(decoder_layer, num_decoder_layers, decoder_norm)
```

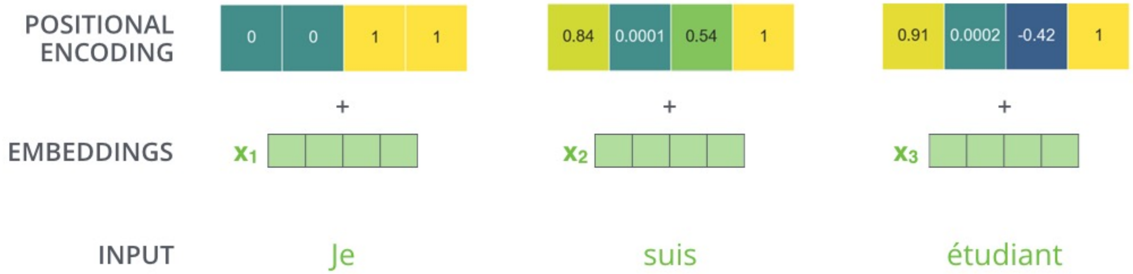
Coding a Transformer (PyTorch): forward pass

```
def forward(self,
        src: Tensor,  # the sequence to the encoder (required)
        tgt: Tensor,  # the sequence to the decoder – target (required)
        src_mask: Optional[Tensor] = None, # the additive mask for the src sequence (opt)
        tgt_mask: Optional[Tensor] = None, …):  # the additive mask for the tgt sequence (opt)

        memory = self.encoder(src, mask=src_mask, …)
        output = self.decoder(tgt, memory, tgt_mask=tgt_mask, …)
        return output
```

Transformer: Positional encodings



- Attention is a permutation-invariant operation.
- A pure attention module will return the same output regardless of the order of its inputs.
- As a solution: Positional encodings are added to the input in order to make use of the order of the sequence.



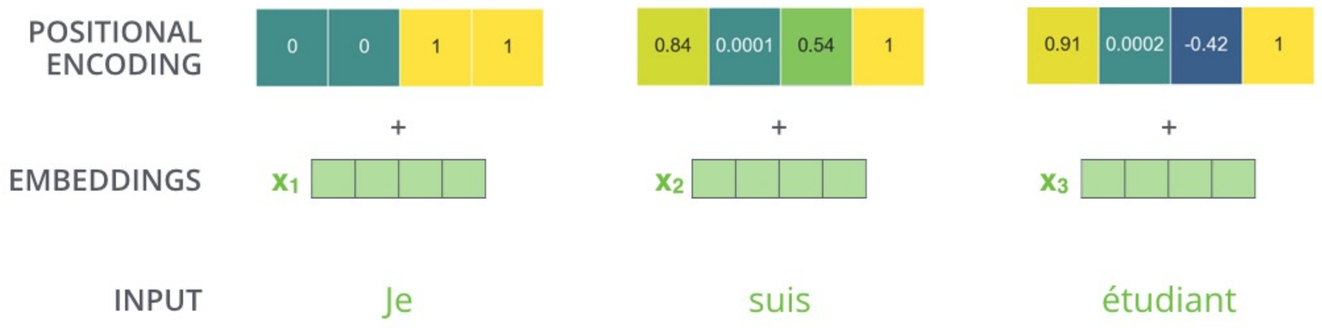
Transformer: Positional encodings

- Intuitively: Positional encodings follow a specific pattern that the model learns
  - To determine the position of each word / the distance between words in the sequence.
- They can encode spatial, temporal, and modality identity… they can be learned or fixed.

- The original Transformer uses sine and cosine functions of different frequencies:

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$



Pros & Cons

Pros:

- Transformer operates on data in parallel which accelerates the learning process, compared to RNN which operates sequentially
- Transformer can deal with long-term dependencies in sequences

Cons:

- Transformer scales quadratically with the number of inputs
- They are memory-intense and require lots of data and long training

Summary

Encoder-decoder is a useful architecture that can be applied to many deep learning problems

Traditional encoder-decoder for NMT has the issue with the context vector

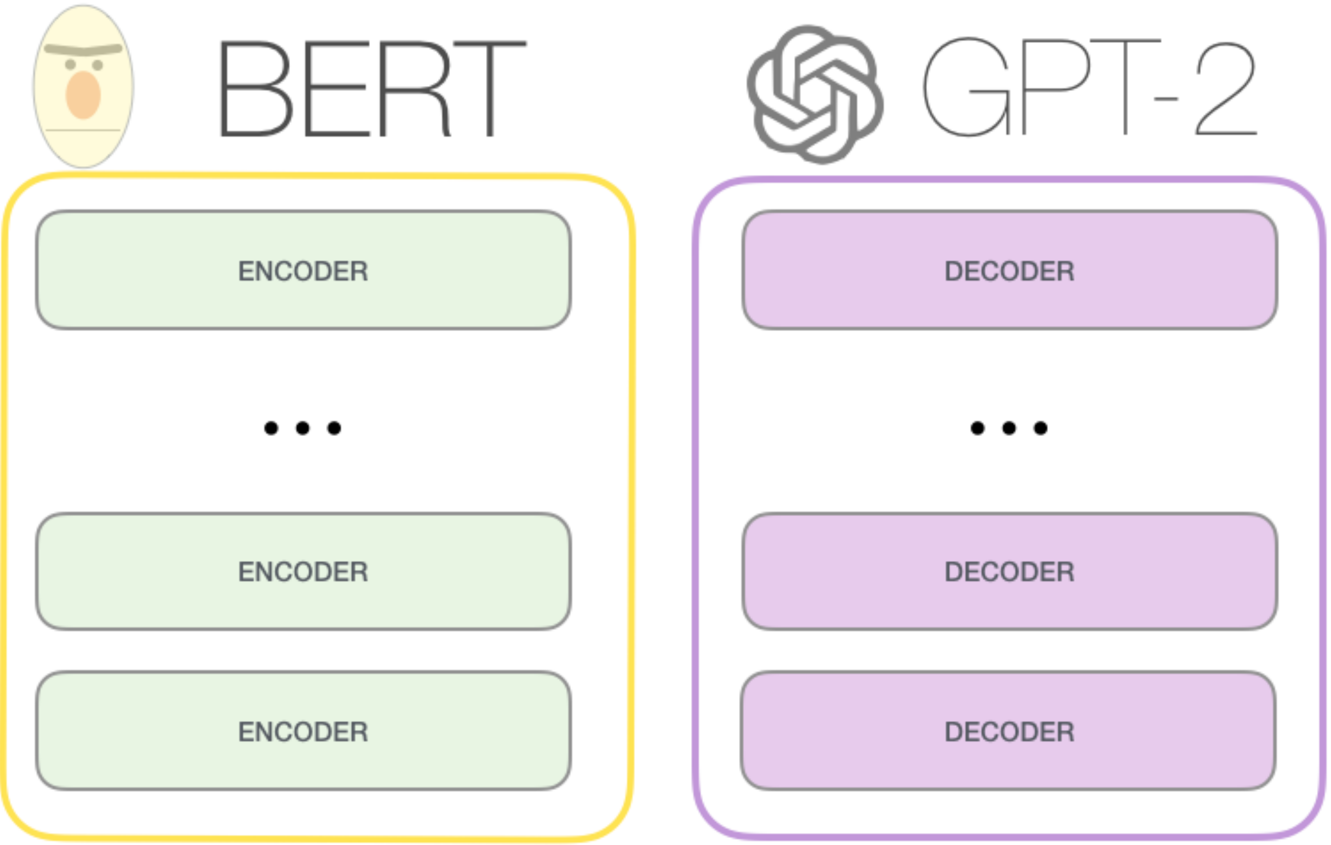
Attention mechanism overcomes the problem, by learning to select important features

Transformer is the first model that entirely relies on attention.

Recommended papers

- Neural Machine Translation by Learning to Align and Translate, Bahdanau et al. ICLR (2015)
- Long Short-Term Memory-Networks for Machine Reading, Cheng et al. (2016)
- Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, Xu et al. (2016)
- Attention Is All You Need, Vaswani et al. (2017)

Family of Transformer models



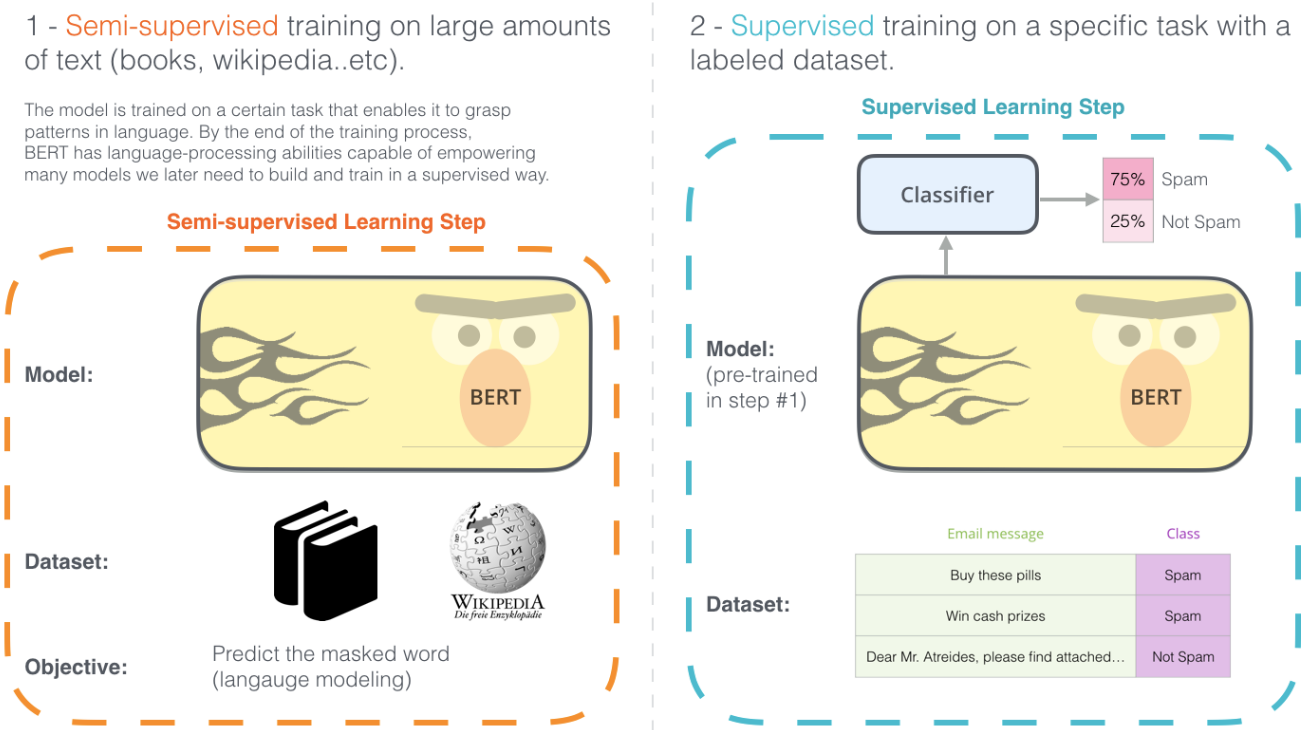
Lecture overview

- Language Transformers
  - BERT
  - GPT
- Multimodal Transformer
  - CLIP
  - Flamingo
- Vision Transformer
- The Perciever (IO)

BERT

- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers == BERT[1]
  - pre-trained Transformer encoder

- BERT is pre-training bidirectional representations from unlabeled text, by jointly conditioning on **both left and right context**.
- A pre-trained BERT model can be fine-tuned with just one additional output layer to create SOTA models for NLP tasks.

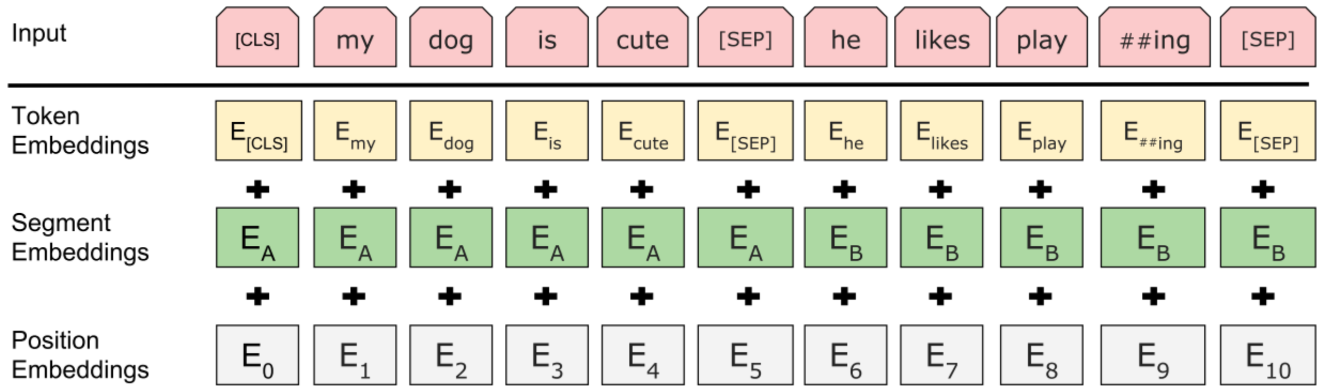


https://jalammar.github.io/illustrated-bert/

[1] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. (2018)

BERT input representation

- The input representation can represent a single sentence OR a pair of sentences in one sequence.
- The complete input is the sum of the token embeddings, the segmentation embeddings and the position embeddings.
- The first token of every sequence is always a special classification token ([CLS]).
  - The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks.



Sentence pairs are packed together into a single sequence, separated with a special token ([SEP]).

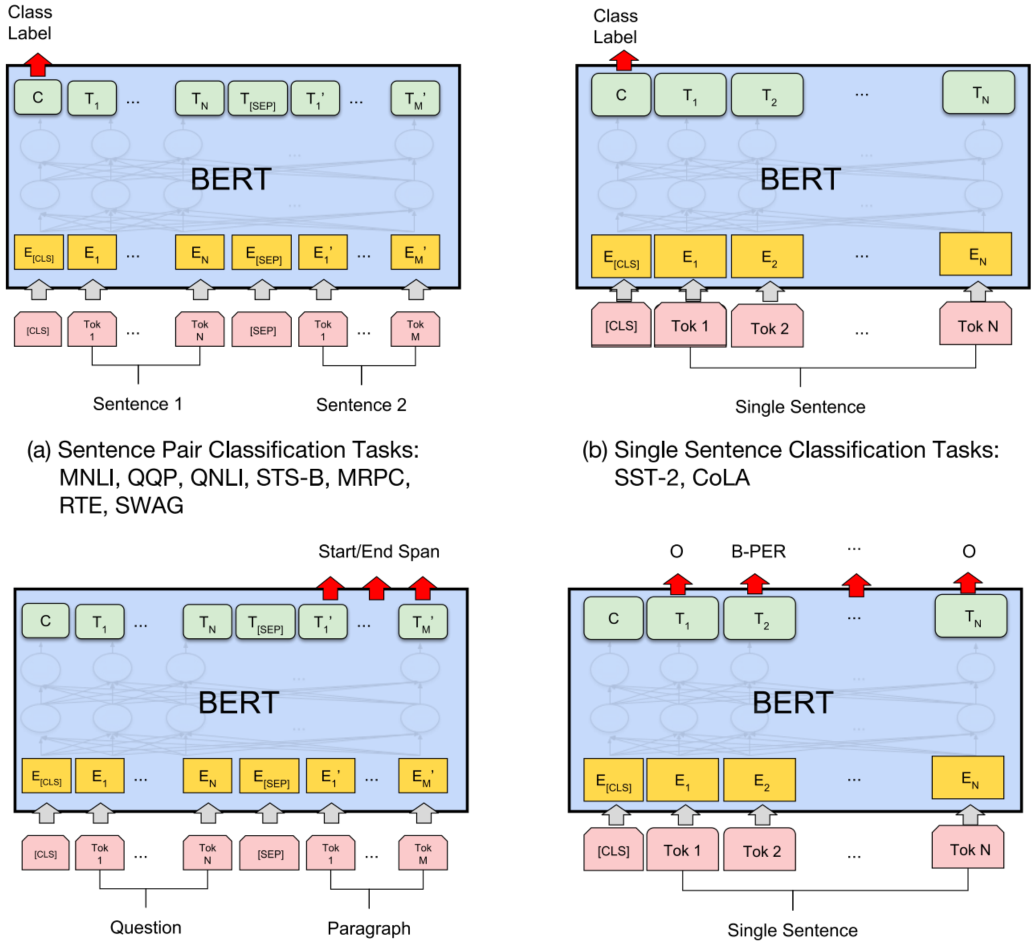
Also, a learned embedding is added to every token indicating whether it belongs to sentence A or sentence B.

BERT pre-training

- Task #1: **M**asked **L**anguage **M**odelling (MLM)
  - Mask a percentage of the input tokens at random with a special token [MASK], and then predict those masked tokens.
- Task #2: **N**ext **S**entence **P**rediction (NSP)
  - Binary prediction whether the next sentences is a correct one.
  - When choosing the sentences A and B, 50% of the time B is the actual next sentence that follows A and 50% of the time it is a random sentence from the corpus.
- Datasets: BooksCorpus (800M words) and Wikipedia (2,500M words).

BERT fine-tuning

- Fine-tuning is straightforward since the self-attention mechanism allows BERT to model many downstream tasks
  - whether they involve single text or text pairs.
- It is relatively inexpensive compared to pre-training



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

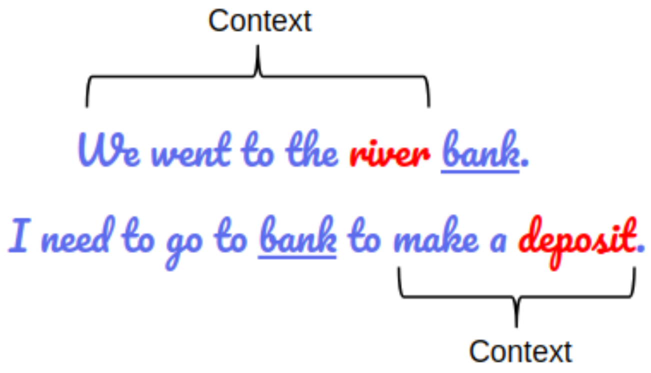
(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

BERT for feature extraction

- Word2Vec/GloVe models produce embeddings to properly represent words.
- They capture *semantic* or meaning-related relationships.
- However, these models produce the same representation for a given word independent of the context.
- Pre-trained BERT can create *contextualized* word embeddings.
- For example, the word "*bank*" will have different embeddings for the 2 sentences, because the context is different.



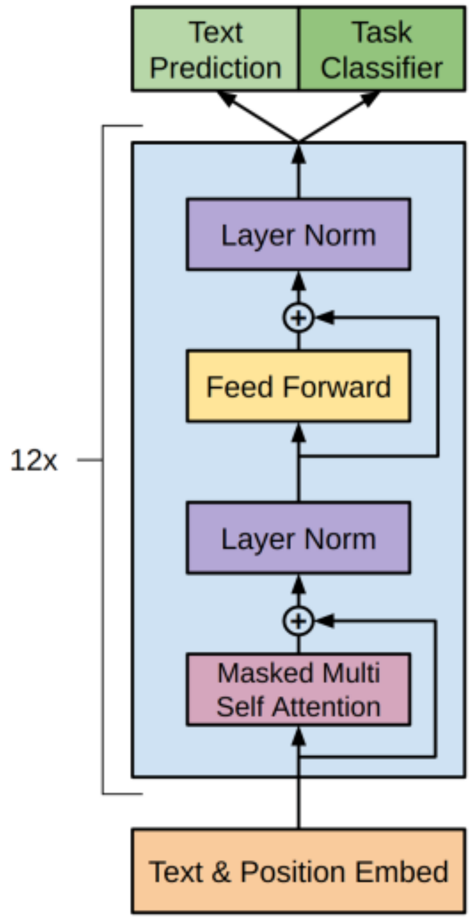
https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/

BERTology

- RoBERTa: A Robustly Optimized BERT Pretraining Approach,
  - More data, Longer training, Larger batches
- ALBERT: A Lite BERT for Self-supervised Learning of Language Representations
- DeBERTa
- DistilBERT
- CamamBERT
- RoBERT
- ClinicalBERT
- ∞

GPT-{1, 2, 3}

- **G**enerative **P**retraining by **T**ransformers == GPT[1]
  - A pre-trained unidirectional Transformer decoder
- The idea: To train a generative language model using unlabeled data
- And then fine-tune it on specific downstream tasks.
- Unsupervised pre-training
  - Given an unsupervised corpus of tokens, use a standard language modeling objective (to predict the next token in the sequence given previous tokens)
- Supervised fine-tuning
  - To adapt the parameters to the supervised task.



Transformer decoder architecture

[1] Improving Language Understanding by Generative Pre-Training, Radford et al. (2018)

GPT-{1, 2, 3}

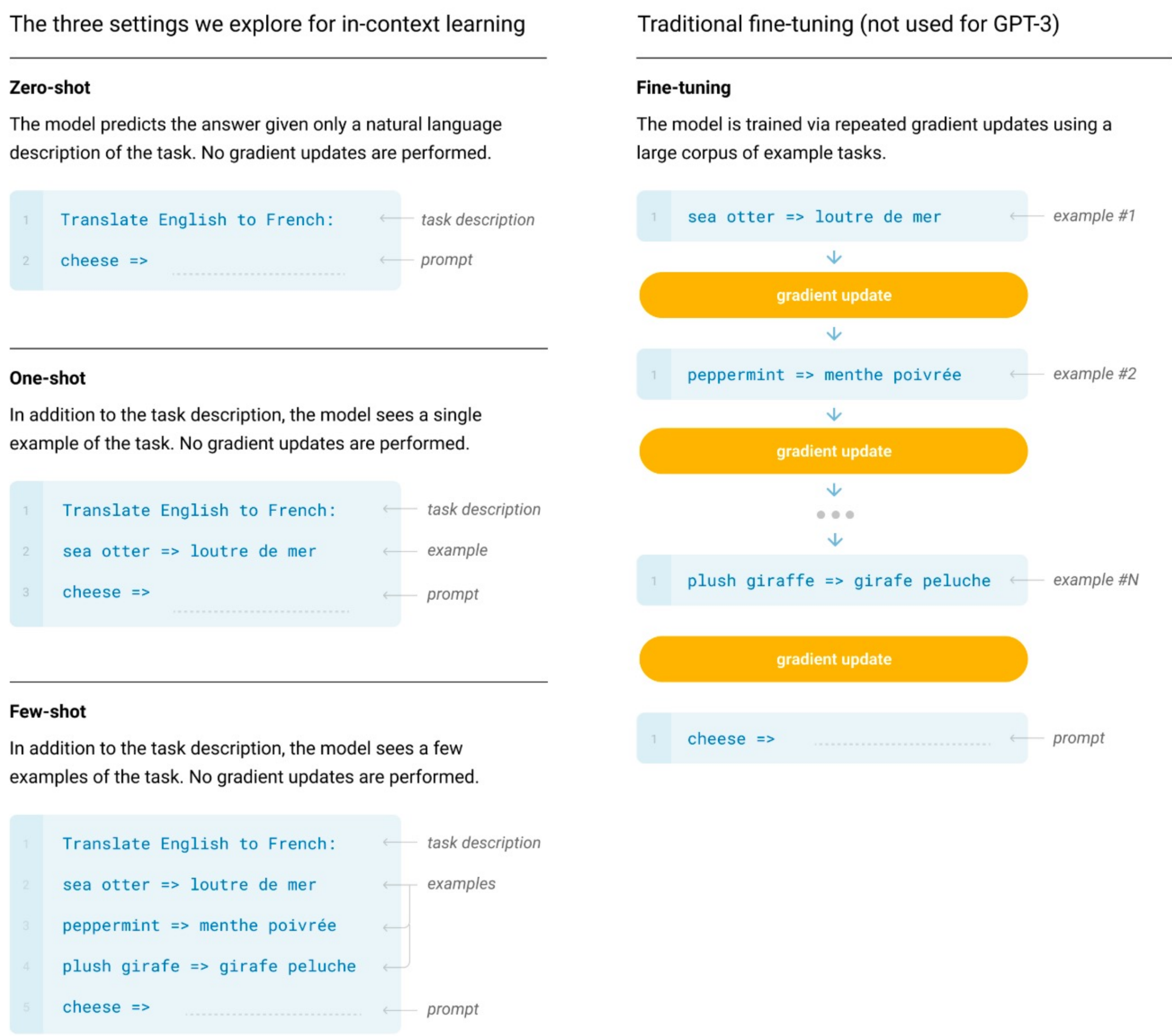
- **GPT1**[1] : Proves that language modeling serves as an effective pre-training objective which helps the model to generalize well
  - A significant achievement is its ability to carry out zero-shot performance on various tasks
- **GPT2**[2] : uses a larger dataset for training and adds additional parameters to build a stronger language model.
- **GPT3**[3] : even larger than GPT2, can automatically generate high-quality paragraphs.
  - Performs well on tasks on which it was never explicitly trained on, like writing SQL queries and codes given natural language description of task.

	GPT-1	GPT-2	GPT-3
Parameters	***117 Million***	***1.5 Billion***	***175 Billion***
Decoder Layers	12	48	96
Hidden Layer	768	1600	12288
Batch Size	64	512	3.2M

[1] Improving Language Understanding by Generative Pre-Training, Radford et al. (2018)
[2] Language Models are Unsupervised Multitask Learners, Radford et al. (2019)
[3] Language Models are Few-Shot Learners, Radford et al. (2020)

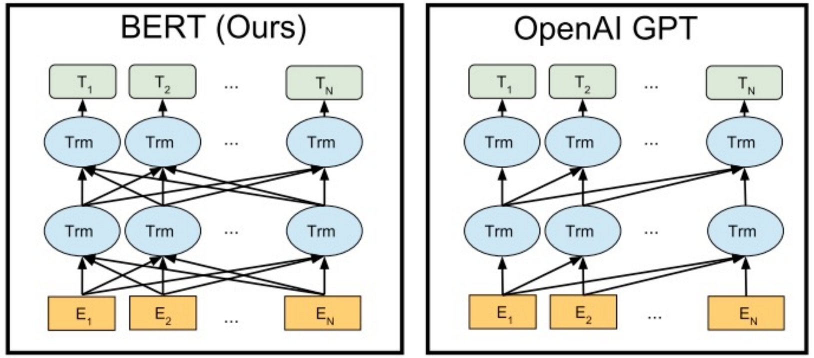
GPT: In-context learning

- GPT models have the ability to perform *in-context learning*.
- First: The model is conditioned on a natural language instruction and/or a few demonstrations of the task.
- Then: it completes the task by predicting what comes next in an autoregressive manner.



GPT vs BERT

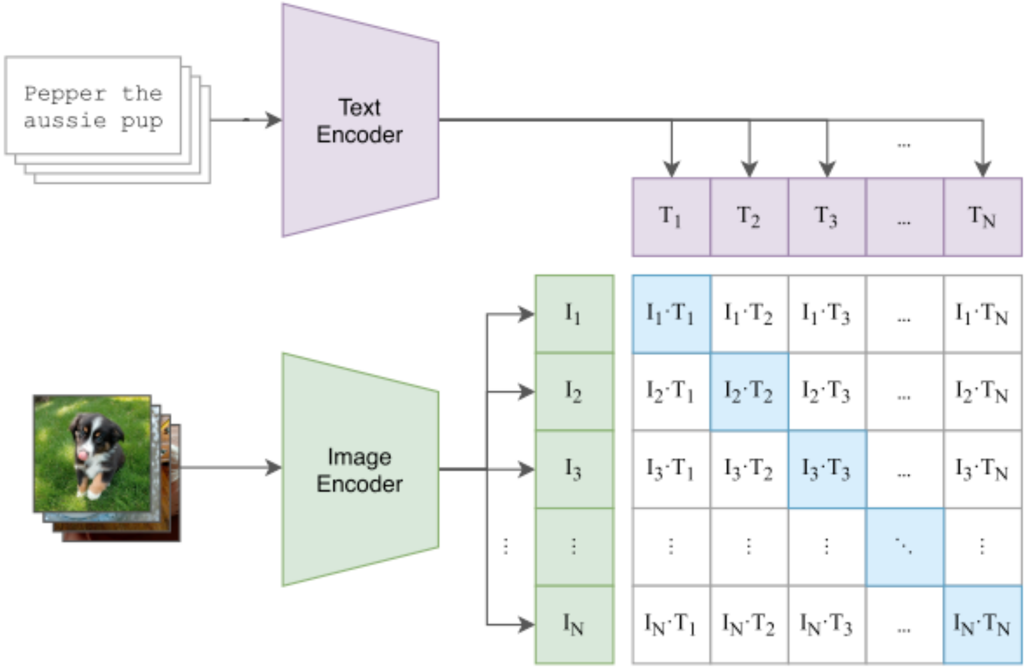
- The key difference between the BERT and GPT, is that GPT is a unidirectional Transformer decoder, whereas BERT is bidirectional Transformer encoder.
- GPT outputs one token at a time (decoding procedure), just like traditional language models.
  - After each token is produced, that token is added to the sequence of inputs.
  - That new sequence becomes the input to the model in its next step.
  - This is an idea called "auto-regression".
- In losing auto-regression, BERT gained the ability to incorporate the context on both sides of a word.



Differences in pre-training model architectures. BERT uses a bidirectional Transformer. GPT uses a left-to-right Transformer.

Multimodal Transformer architecture: CLIP

- CLIP or **C**ontrastive **L**anguage-**I**mage **P**re-training[1]
  - Consists of Image Encoder (CNN/ViT) and Text Encoder (Transformer).
  - Given a pair (image, caption), CLIP processes each modality with the corresponding encoder – yielding a specific embedding for each.

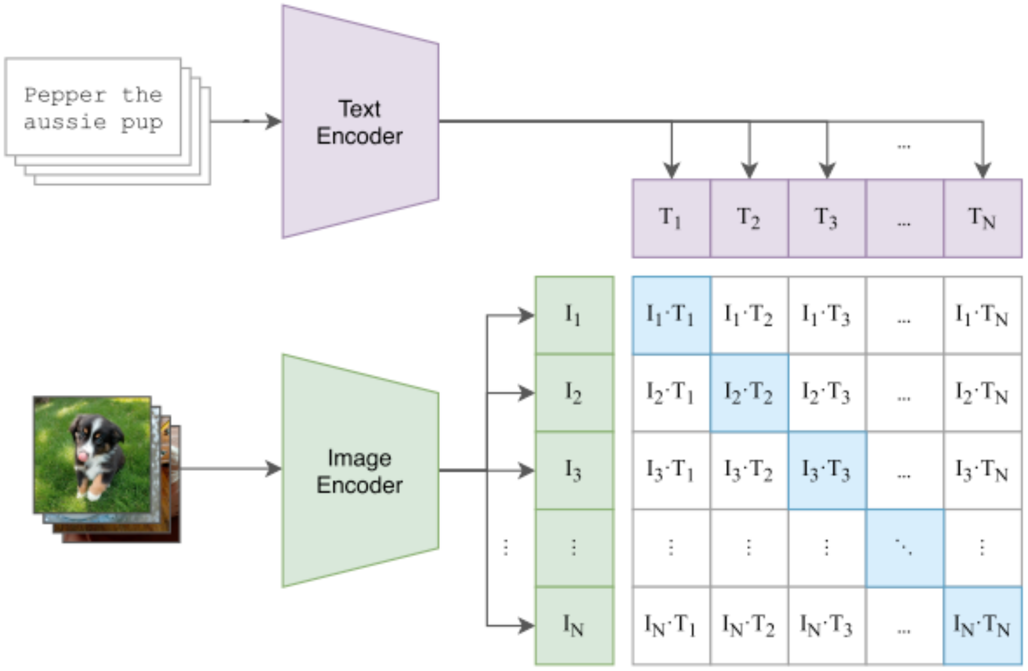


I_1 = ImageEncoder(image$_1$);
I_2 = ImageEncoder(image$_2$)
…
T_1 = TextEncoder(caption$_1$);
T_2 = TextEncoder(caption$_2$)
…

[1] Learning Transferable Visual Models From Natural Language Supervision, Radford et al. (2021)

Multimodal Transformer architecture: CLIP

- CLIP solves an easier proxy pre-training task of predicting which text as a whole, is paired with which image.

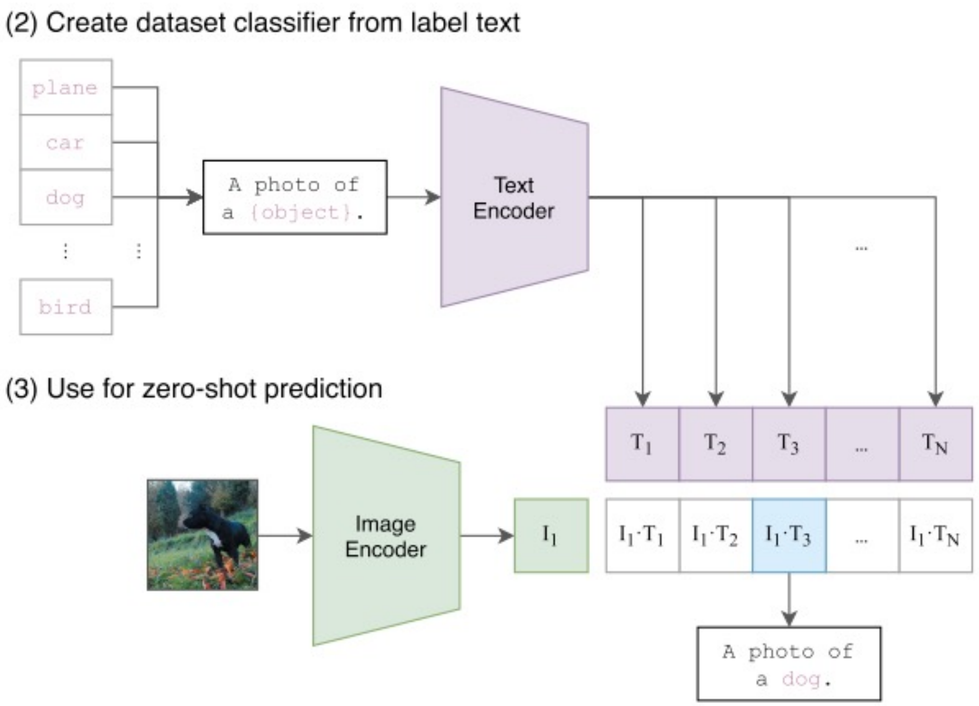


- Maximize the cosine similarity of the image and text embeddings of true pairs (I_1 * T_1)
- Minimize the cosine similarity of the embeddings of incorrect pairs (I_1 * T_2)

- Formally said - CLIP optimizes a contrastive loss.

Multimodal Transformer architecture: CLIP

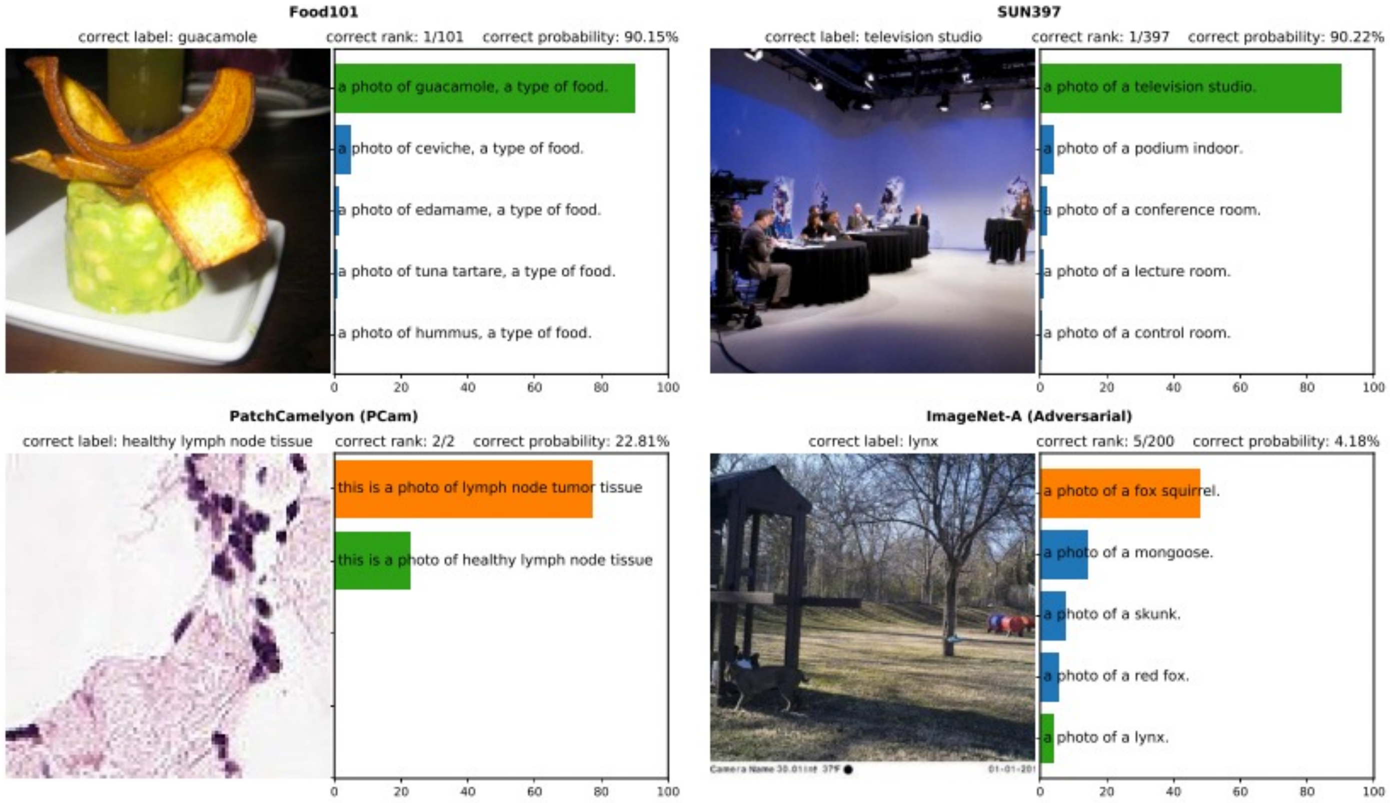
- At inference time: CLIP shows zero-shot classification abilities
  - Predicting labels which were never observed during training
- It uses the names of all the classes in the dataset as the set of potential text pairings: a prompt template “A photo of a {label}.”



- First: compute the feature embedding of the image: I_1 and the feature embedding of all possible texts T_1, T_2, T_3 …
- Then: compute the cosine similarity of these embeddings, normalized into a probability distribution via a softmax.
- This gives the most probable (image, text) pair, hence the predicted class.

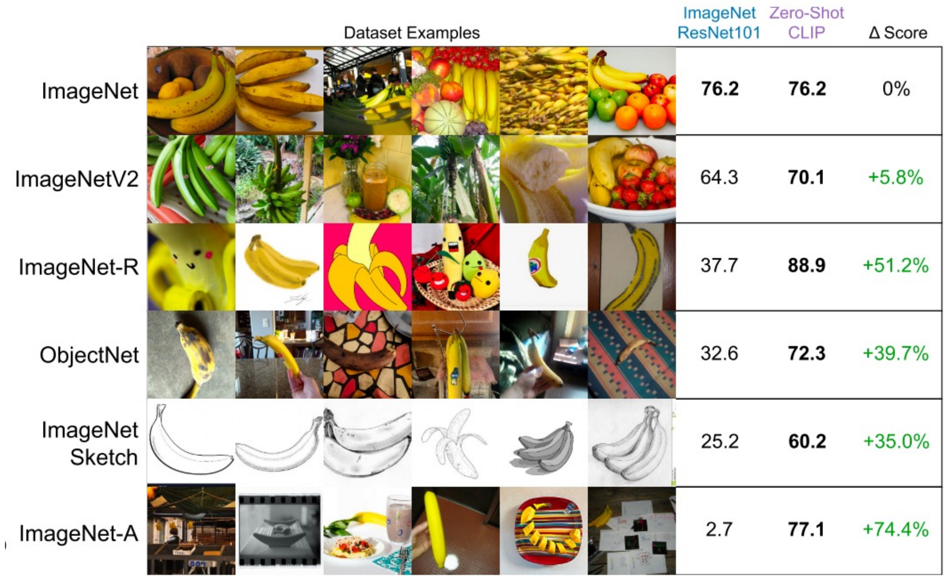
CLIP: Zero-Shot Examples

- Visualization of predictions from CLIP zero-shot classifiers
  - The predicted probability of the top 5 classes is shown along with the text used to represent the class.



CLIP: Robustness

- Zero-shot CLIP models are much more robust than equivalent supervised ImageNet models.
- This suggests that zero-shot evaluation is much more representative of a model's capability.



Visualizing distribution shift for bananas, a class shared across 5 of the 7 natural distribution shift datasets.

		Food101	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech101	Flowers	MNIST	FER2013	STL10*	EuroSAT	RESISC45	GTSRB	KITTI	Country211	PCAM	UCF101	Kinetics700	CLEVR	HatefulMeme	SST	ImageNet
LM RN50		81.3	82.8	61.7	44.2	69.6	74.9	44.9	85.5	71.5	82.8	85.5	91.1	96.6	60.1	95.3	93.4	84.0	73.8	70.2	19.0	82.9	76.4	51.9	51.2	65.2	76.8	65.2
CLIP-RN	50	86.4	88.7	70.3	56.4	73.3	78.3	49.1	87.1	76.4	88.2	89.6	96.1	98.3	64.2	96.6	95.2	87.5	82.4	70.2	25.3	82.7	81.6	57.2	53.6	65.7	72.6	73.3
	101	88.9	91.1	73.5	58.6	75.1	84.0	50.7	88.0	76.3	91.0	92.0	96.4	98.4	65.2	97.8	95.9	89.3	82.4	**73.6**	26.6	82.8	84.0	60.3	50.3	68.2	73.3	75.7
	50x4	91.3	90.5	73.0	65.7	77.0	85.9	57.3	88.4	79.5	91.9	92.5	97.8	98.5	68.1	97.8	96.4	89.7	85.5	59.4	30.3	83.0	85.7	62.6	52.5	68.0	76.6	78.2
	50x16	93.3	92.2	74.9	72.8	79.2	88.7	62.7	**89.0**	79.1	93.5	93.7	98.3	**98.9**	68.7	98.6	97.0	91.4	89.0	69.2	34.8	83.5	88.0	66.3	53.8	71.1	**80.0**	81.5
	50x64	94.8	94.1	78.6	77.2	81.1	90.5	67.7	**88.9**	**82.0**	94.5	95.4	98.9	**98.9**	**71.3**	99.1	97.1	92.8	90.2	69.2	40.7	83.7	89.5	69.1	55.0	**75.0**	**81.2**	83.6
CLIP-ViT	B/32	88.8	95.1	80.5	58.5	76.6	81.8	52.0	87.7	76.5	90.0	93.0	96.9	**99.0**	69.2	98.3	97.0	90.5	85.3	66.2	27.8	83.9	85.5	61.7	52.1	66.7	70.8	76.1
	B/16	92.8	96.2	83.1	67.8	78.4	86.7	59.5	**89.2**	79.2	93.1	94.7	98.1	**99.0**	69.5	99.0	97.1	92.7	86.6	67.8	33.3	83.5	88.4	66.1	**57.1**	70.3	75.5	80.2
	L/14	95.2	98.0	87.5	77.0	**81.8**	**90.9**	69.4	**89.6**	**82.1**	**95.1**	**96.5**	99.2	**99.2**	**72.2**	**99.7**	**98.2**	94.1	**92.5**	64.7	42.9	85.8	**91.5**	72.0	**57.8**	**76.2**	**80.8**	83.9
	L/14-336px	**95.9**	97.9	87.4	**79.9**	**82.2**	**91.5**	**71.6**	**89.9**	**83.0**	**95.1**	**96.0**	99.2	**99.2**	**72.9**	**99.7**	**98.1**	**94.9**	**92.4**	69.2	**46.4**	85.6	**92.0**	**73.0**	**60.3**	**77.3**	**80.5**	85.4

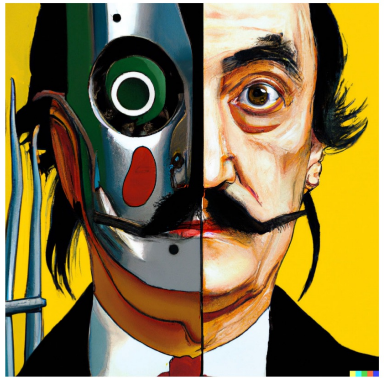
Performance of various pre-trained models over 27 datasets

CLIP: Usage in other models

- The joint multimodal embedding space of CLIP enables its usage in many other downstream tasks in a zero-shot fashion.

"Vibrant portrait painting of Salvador Dalí with a robotic half."

- DALLE-2[1] – a text-guided image generation model,
- CLIP4Clip[2] - video-language retrieval model,
- GroupViT[3] – semantic segmentation model - in a zero-shot manner.

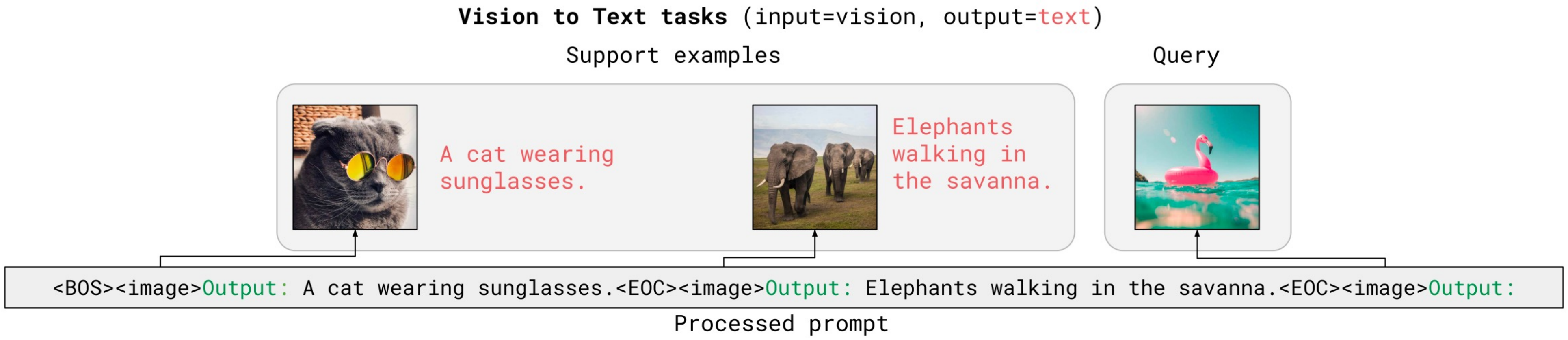


vibrant portrait painting of Salvador Dalí with a robotic half face

[1] Hierarchical Text-Conditional Image Generation with CLIP Latents, Ramesh et al. (2022)
[2] CLIP4Clip: An Empirical Study of CLIP for End to End Video Clip Retrieval , Luo et al. (2022)
[3] GroupViT: Semantic Segmentation Emerges from Text Supervision, Xu et al. (2022)

CLIP: Shortcomings

- Models like CLIP simply provide a similarity score between text and image.
- Able to tackle limited use cases such as classification, where a finite set of outcomes is provided beforehand.
- Lack the ability to generate language - less suitable to more open-ended tasks.

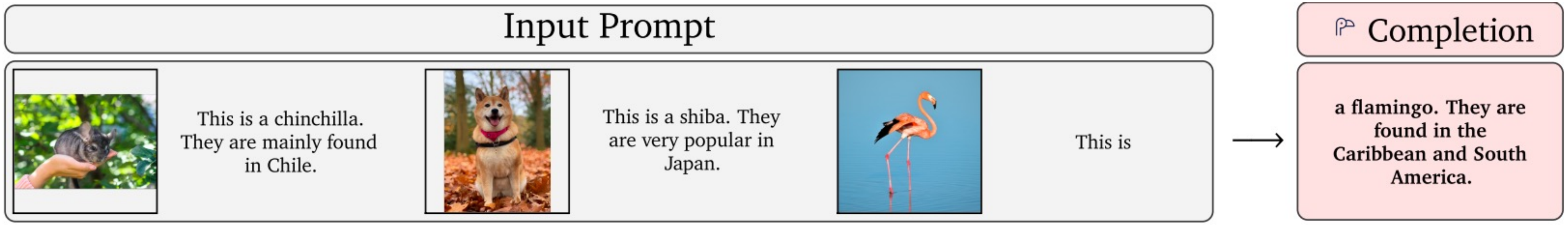


As a potential solution: Visual Language Models (VLMs), such as Flamingo[1].

[1] Flamingo: a Visual Language Model for Few-Shot Learning, Alayrac et al. (2022)

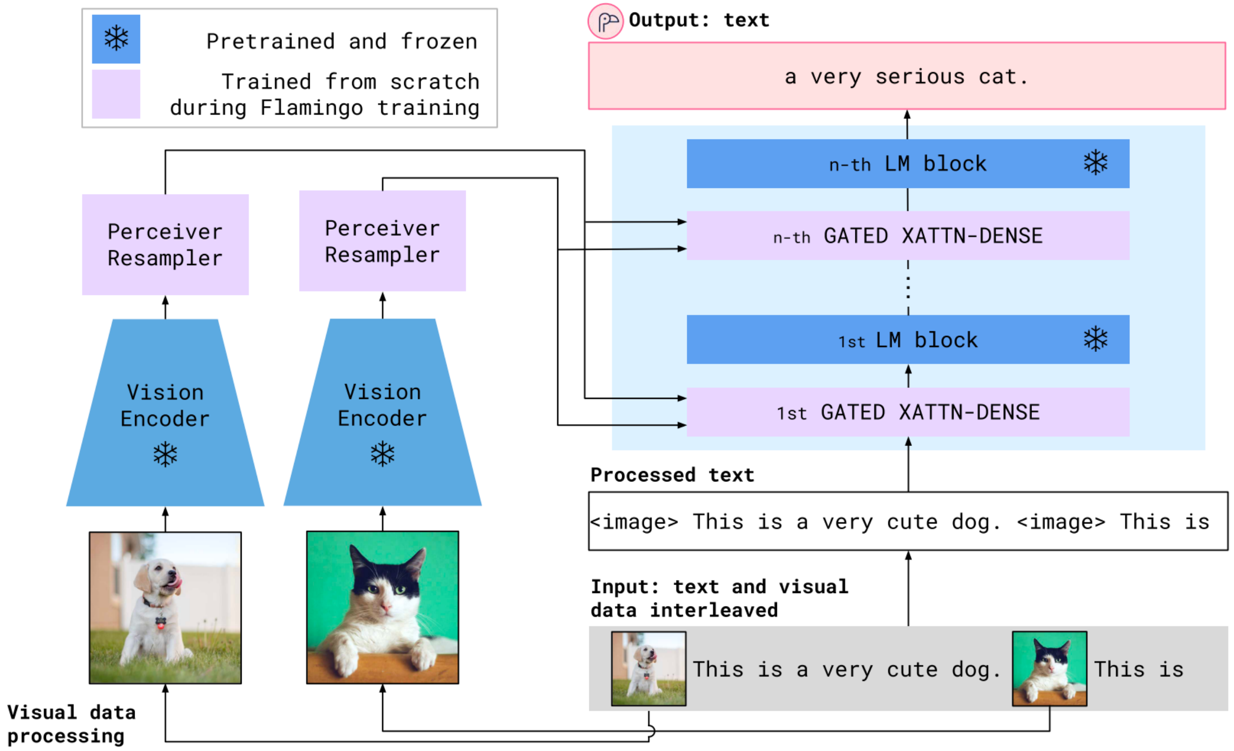
Visual Language Model: Flamingo

- Flamingo is a Transformer-based architecture for multimodal few-shot tasks (image captioning, visual dialogue or visual question answering)
- Able to learn from only a few input/output examples i.e., *in few-shot settings*.
  - It processes arbitrarily interleaved images and text as prompt;
  - And it generates output text in an open-ended manner.
- Basically: it performs in-context learning (like GPT) but with images and text as context (prompt).



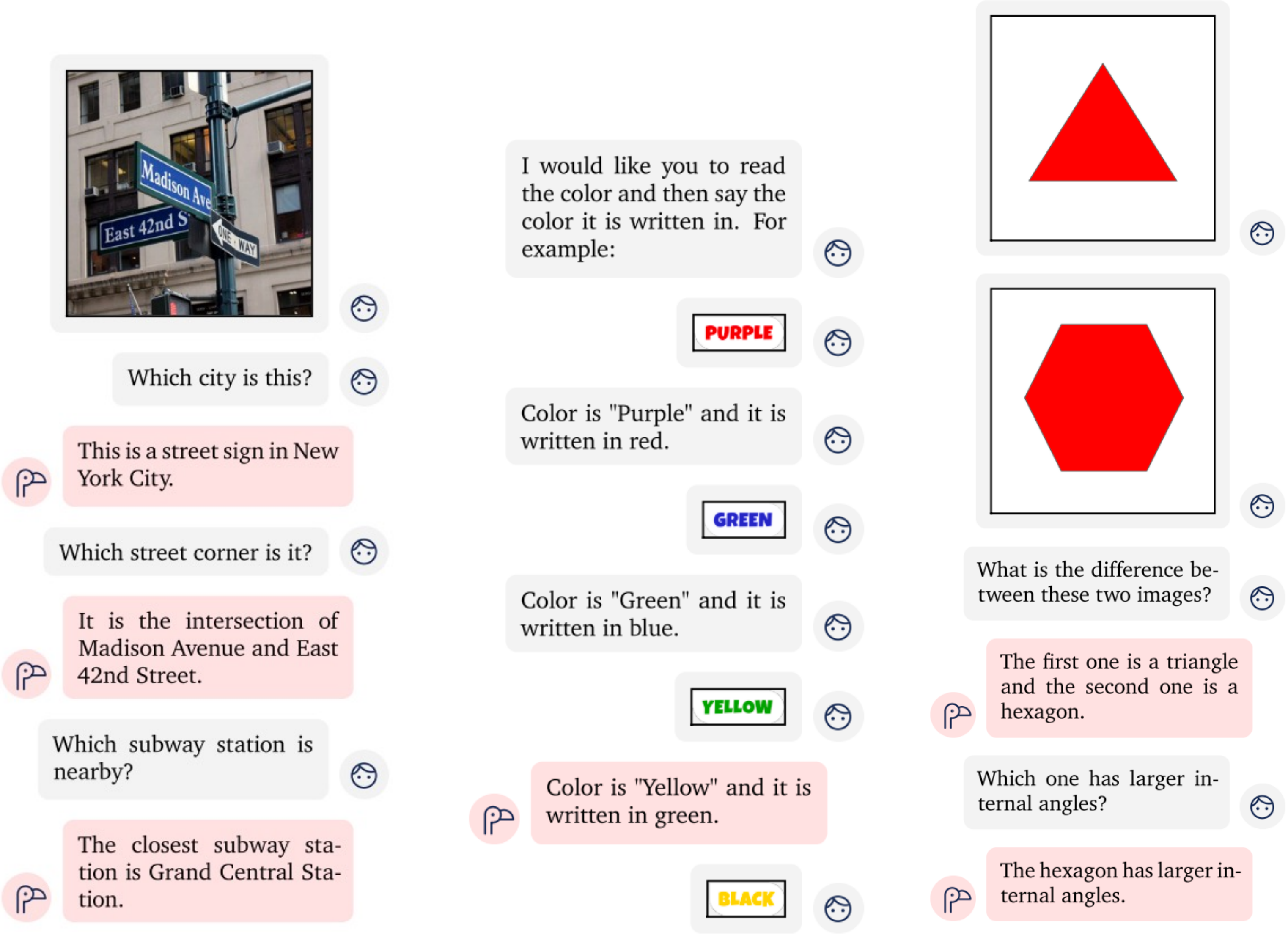
Visual Language Model: Flamingo

- On the vision side: a vision encoder with a contrastive text-image approach, à la CLIP
- On the language side: existing autoregressive LM trained on a large text corpus
- Linked via a learnable attention component (the Perceiver)
  - It outputs a fixed-size set of visual tokens.
  - Which are used to condition the frozen LM, trained to generate text.



Visual Language Model: Flamingo

Visual dialogue examples:



Your feedback

- We want to hear what is going well and what can be improved
- https://forms.gle/w6KZVvwmGtHbZync6
- Takes 2min

Vision Transformer

- Transformer became the de-facto standard for NLP, but was much less used for vision tasks
- The recently introduced **Vision Transformer** (ViT) is a pure Transformer model applied directly *to sequences of image patches*
- It performs well on image tasks without the reliance on convolutional layers

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

Understanding a “Figure 1”

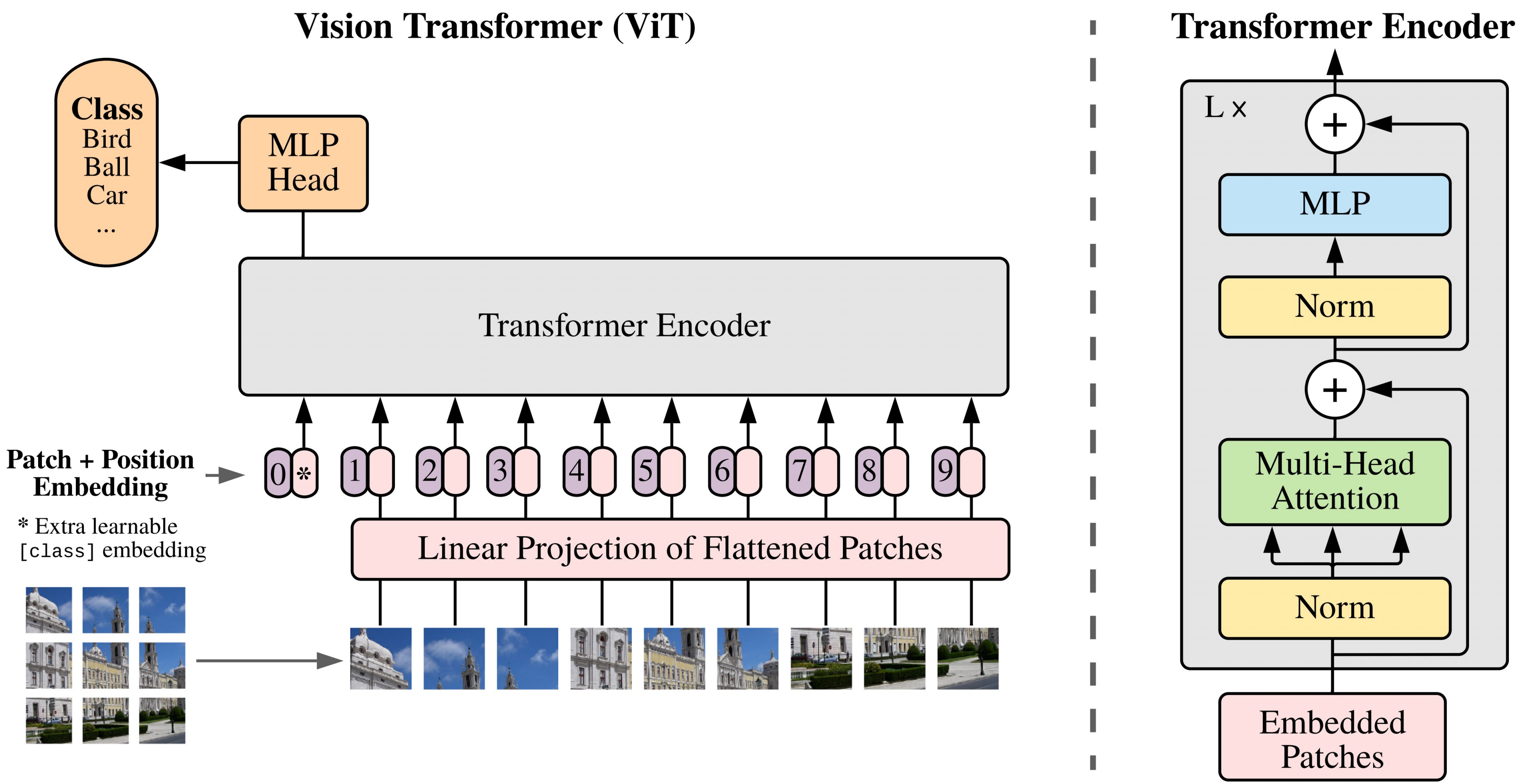


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable “classification token” to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

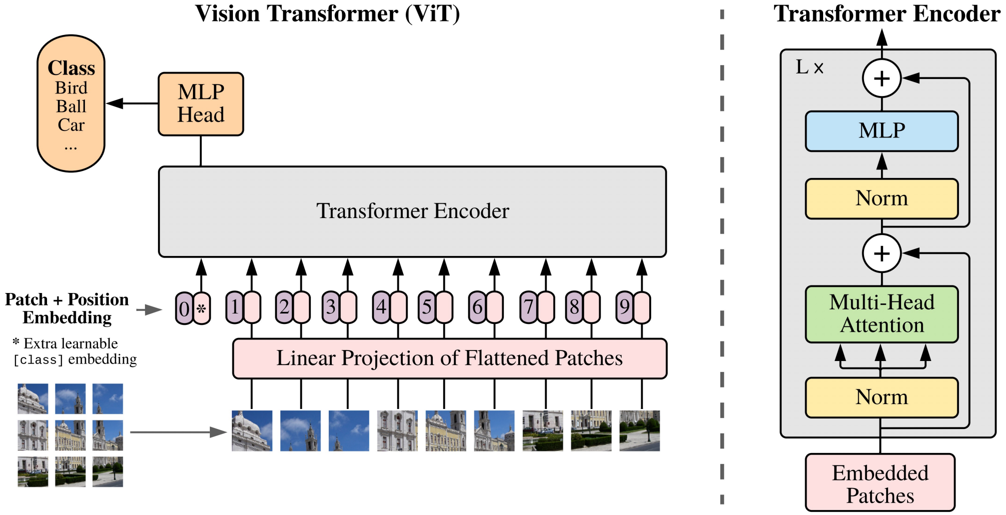
Quiz: From what you now know about attention, what could be an advantage of using attention compared to convolutions?

Discuss with your neighbor for 2min

Vision Transformer

- Like BERT's [CLS] token, a learnable embedding is prepended to the sequence of embedded patches,
  - the classification is done on this token (with an MLP)

- "Position encodings" are added to the patch embeddings to retain positional information. (attention by itself doesn't have any notion of ordering/space)
  - These vectors are also simply learned




```python
# pos_embed has entry for class token, concat then add
if self.cls_token is not None:
    x = torch.cat((self.cls_token.expand(x.shape[0], -1, -1), x), dim=1)
x = x + self.pos_embed
```

https://github.com/rwightman/pytorch-image-models/blob/main/timm/models/vision_transformer.py

Vision Transformer

🤔: why principle components?

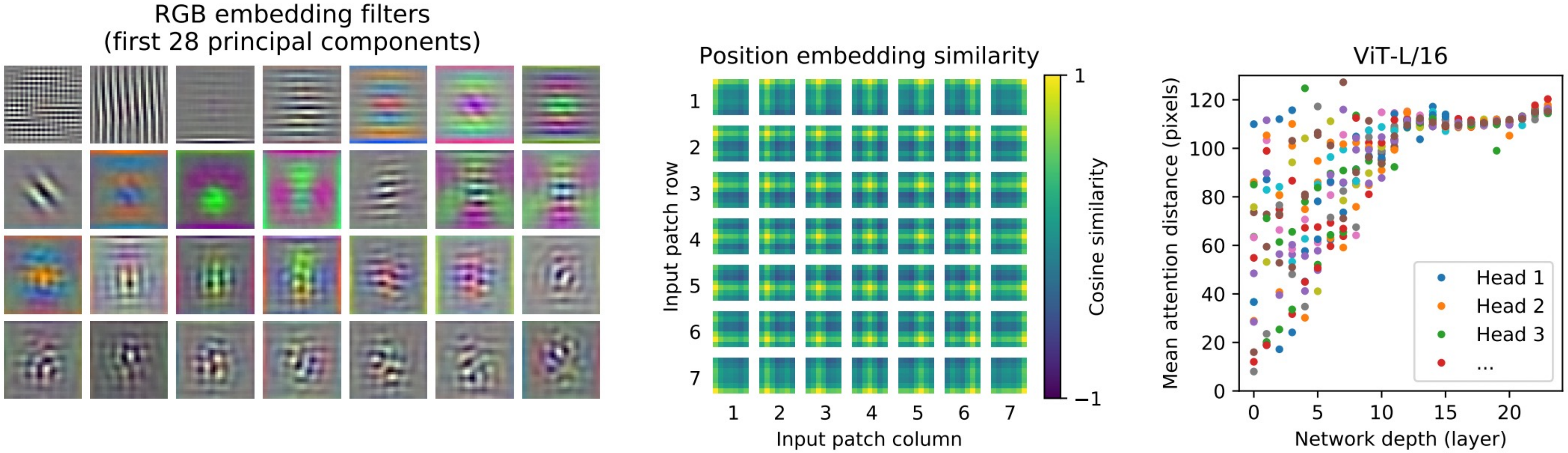


Figure 7: **Left:** Filters of the initial linear embedding of RGB values of ViT-L/32. **Center:** Similarity of position embeddings of ViT-L/32. Tiles show the cosine similarity between the position embedding of the patch with the indicated row and column and the position embeddings of all other patches. **Right:** Size of attended area by head and network depth. Each dot shows the mean attention distance across images for one of 16 heads at one layer. See Appendix D.7 for details.

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

Attention as a superset of convolutions

A **multi-head self-attention layer** with N_h heads of dimension D_h, output dimension D_{out} and a relative positional encoding of dimension $D_p \geq 3$ **can express any convolutional layer** of kernel size $\sqrt{N_h} \times \sqrt{N_h}$ and $\min(D_h, D_{out})$ output channels.

Further reading: *Deformable convolutions*

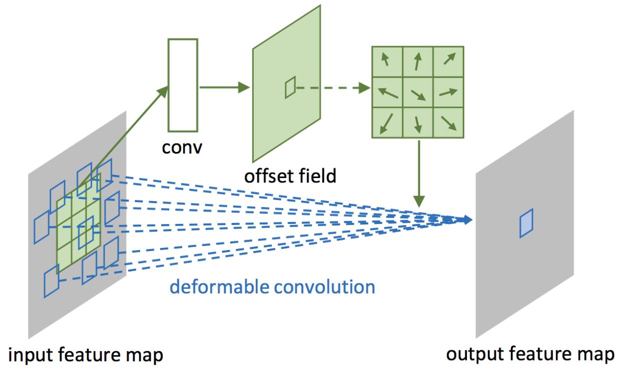
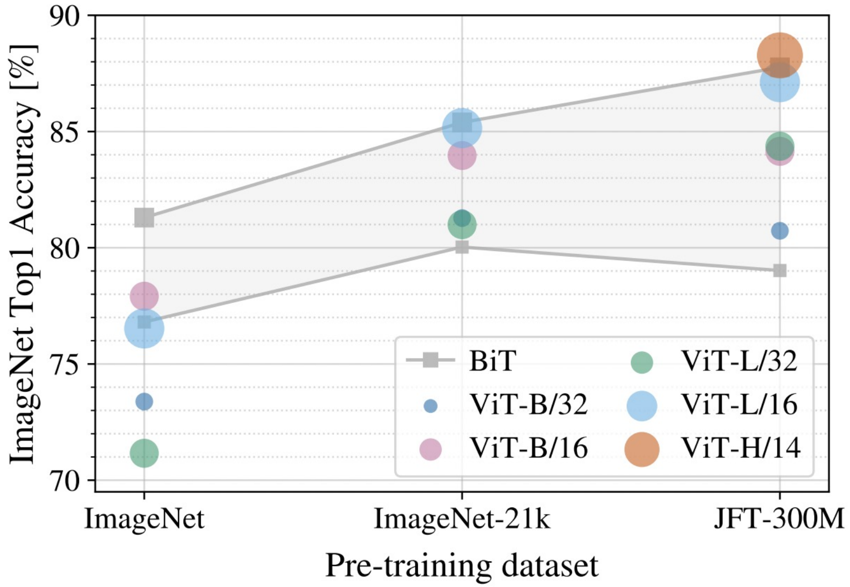


Figure 2: Illustration of 3 × 3 deformable convolution.

On the relationship between self-attention and convolutional layers. Cordonnier et al. ICLR 2020

Training a ViT is more difficult

- Original paper required ImageNet-22k (14M images) to achieve good performances
- DeiT paper showed training with ImageNet-1k possible if more augmentations and regularisations are used

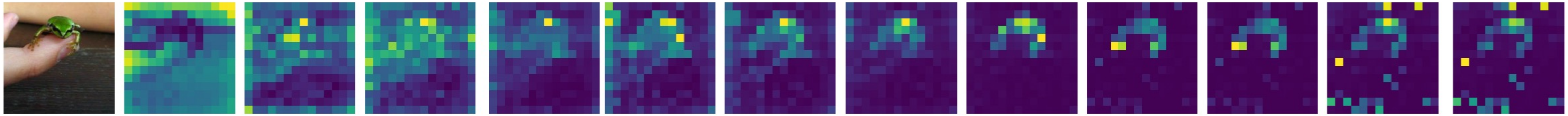


Ablation on ↓	Pre-training	Fine-tuning	Rand-Augment	AutoAug	Mixup	CutMix	Erasing	Stoch. Depth	Repeated Aug.	Dropout	Exp. Moving Avg.	top-1 accuracy: pre-trained 224^2	top-1 accuracy: fine-tuned 384^2
none: DeiT-B	adamw	adamw	✓	✗	✓	✓	✓	✓	✓	✗	✗	81.8 ±0.2	83.1 ±0.1
optimizer	SGD	adamw	✓	✗	✓	✓	✓	✓	✓	✗	✗	74.5	77.3
	adamw	SGD	✓	✗	✓	✓	✓	✓	✓	✗	✗	81.8	83.1
data augmentation	adamw	adamw	✗	✗	✓	✓	✓	✓	✓	✗	✗	79.6	80.4
	adamw	adamw	✗	✓	✓	✓	✓	✓	✓	✗	✗	81.2	81.9
	adamw	adamw	✓	✗	✗	✓	✓	✓	✓	✗	✗	78.7	79.8
	adamw	adamw	✓	✗	✓	✗	✓	✓	✓	✗	✗	80.0	80.6
	adamw	adamw	✓	✗	✗	✗	✓	✓	✓	✗	✗	75.8	76.7
regularization	adamw	adamw	✓	✗	✓	✓	✗	✓	✓	✗	✗	4.3*	0.1
	adamw	adamw	✓	✗	✓	✓	✓	✗	✓	✗	✗	3.4*	0.1
	adamw	adamw	✓	✗	✓	✓	✓	✓	✗	✗	✗	76.5	77.4
	adamw	adamw	✓	✗	✓	✓	✓	✓	✓	✓	✗	81.3	83.1
	adamw	adamw	✓	✗	✓	✓	✓	✓	✓	✗	✓	81.9	83.1

Training data-efficient image transformers & distillation through attention. Tuvron et al. ICML 2021

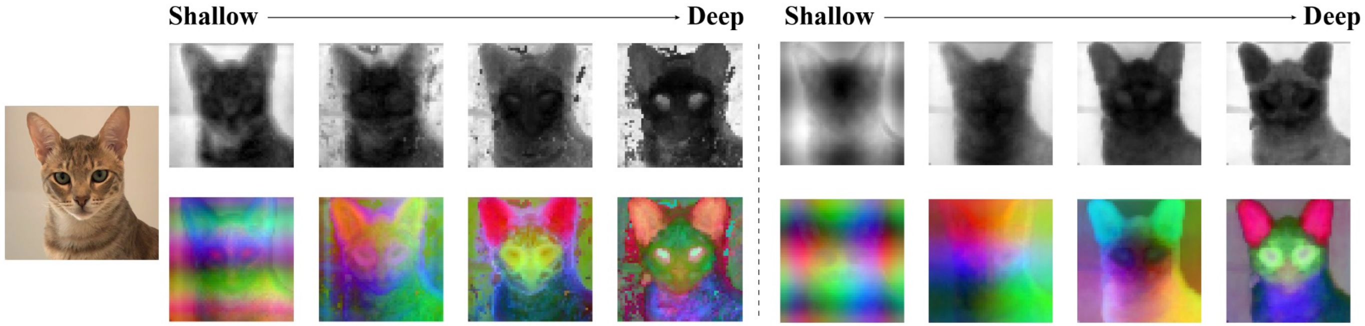
ViT features

Feature maps stay the same size. Easy for computing, optimizations, dense downstream tasks etc.

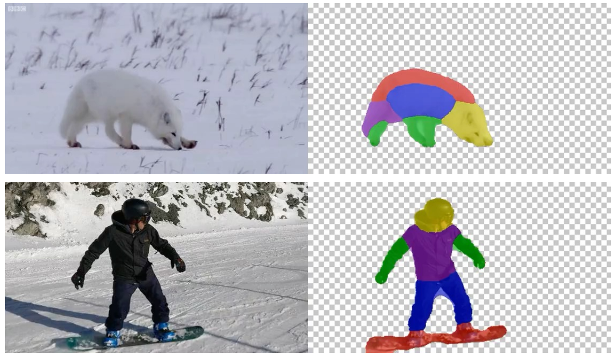


PCA Visualization

We apply principal component analysis (PCA) on spatial descriptors across layers from a supervised ViT and a DINO-trained ViT. We find that early layers contain positionally biased representations, that gradually become more semantic in deeper layers. Both ViTs produce semantic representations with high granularity, that cause semantic object parts to emerge. However, DINO ViT representations are less noisy that supervised ViT representations.



(a) Supervised ViT (b) DINO ViT



Quite useful features!
Especially when combined with self-supervised learning (SSL)

A SSL pretraining method

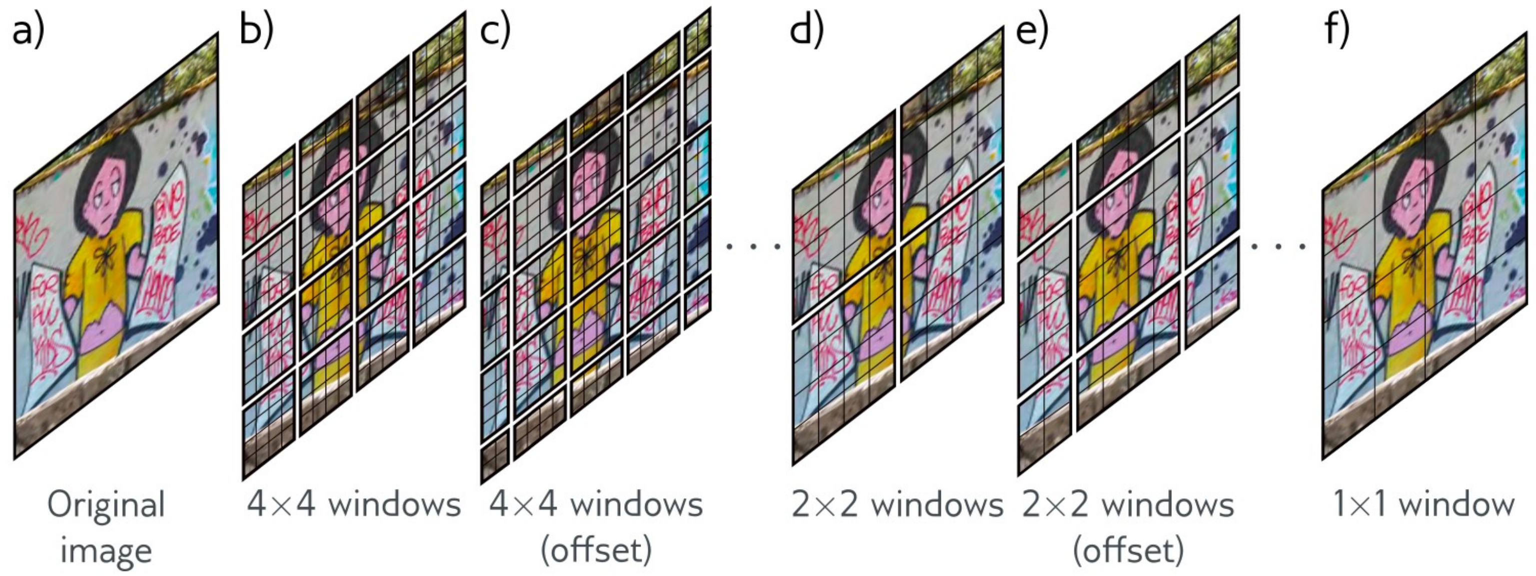
Also here: ImageNet can (more or less) be solved with textures

Pos. Emb.	Default/Stem	Every Layer	Every Layer-Shared
No Pos. Emb.	0.61382	N/A	N/A
1-D Pos. Emb.	0.64206	0.63964	0.64292
2-D Pos. Emb.	0.64001	0.64046	0.64022
Rel. Pos. Emb.	0.64032	N/A	N/A

Table 8: Results of the ablation study on positional embeddings with ViT-B/16 model evaluated on ImageNet 5-shot linear.

Swin Transformer: add hierarchy back in?

- Idea: mostly looking at "local" neighborhood, so can save some computation (remember attention is O(n^2)) or gain some accuracy by modelling this
- Strong performance but slow models



Hybrid Architectures get best performances (atm)

Basic idea/observation

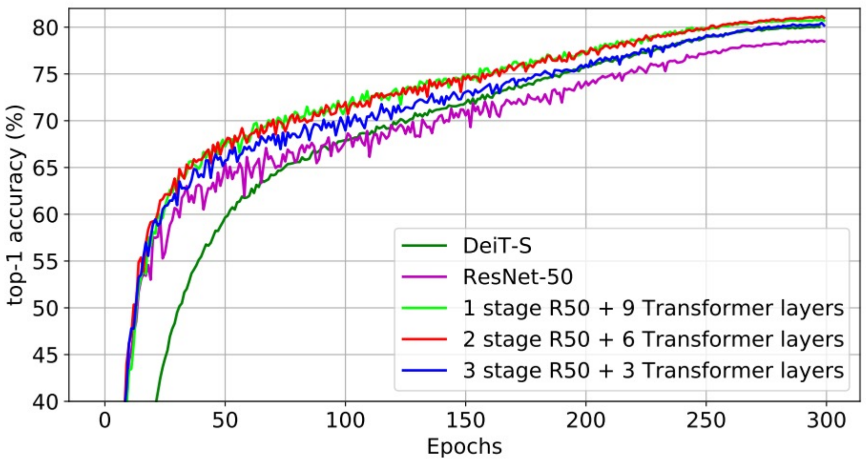
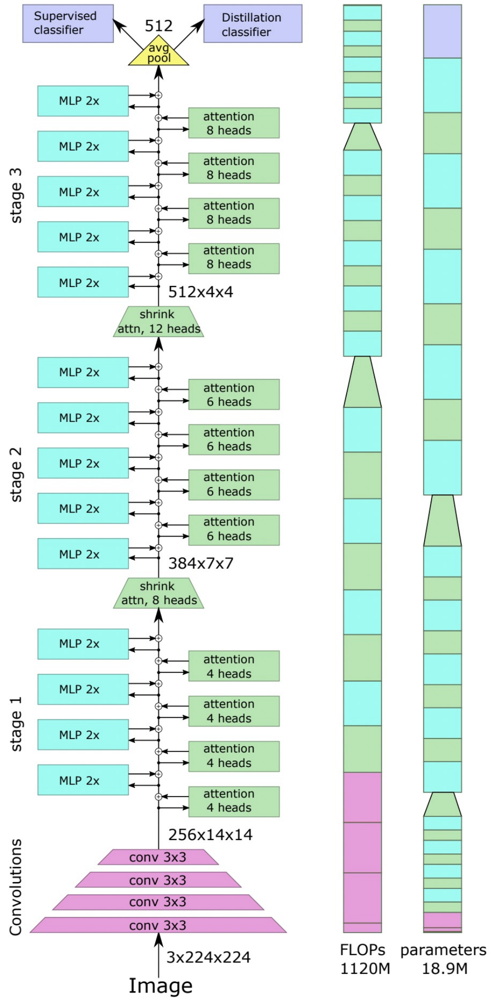


Figure 3: Models with convolutional layers show a faster convergence in the early stages compared to their DeiT counterpart.

That's funny…



LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference. Graham et al. ICCV 2021

The Perceiver

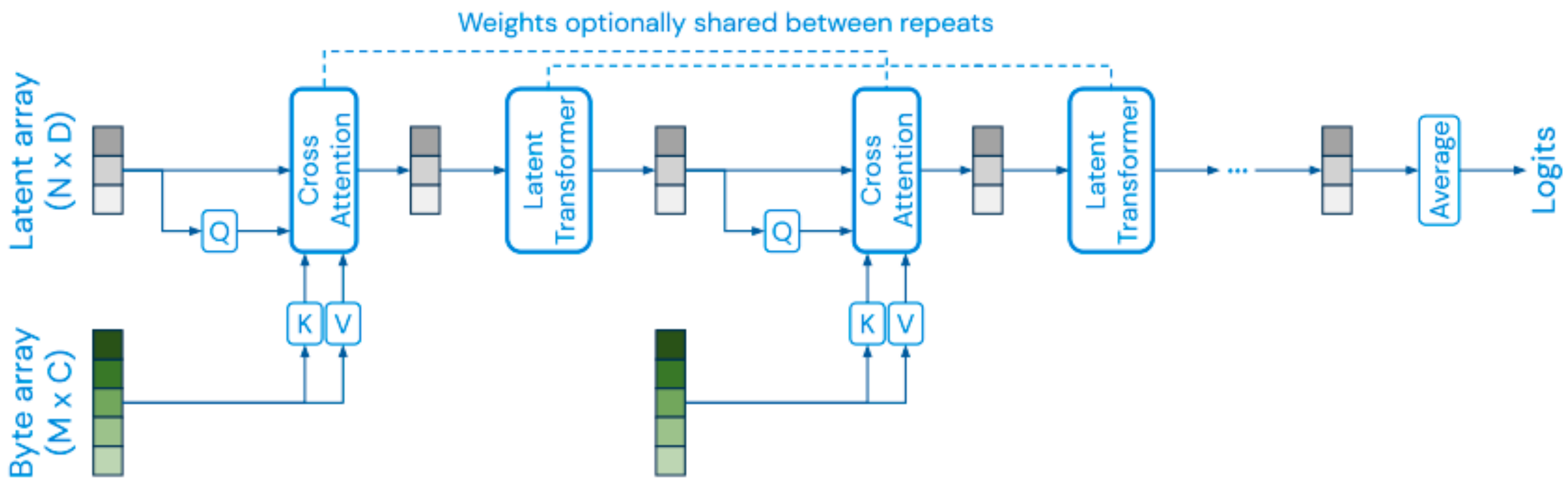
- Biological systems: vision, audition, touch etc.
- Deep learning models currently: designed for individual modalities
  - Architectural priors introduce helpful inductive biases, but also lock models to individual modalities. (CNNs don't work on eg point cloud data)
- **The Perceiver:** designed to handle arbitrary configurations of different modalities using Transformer-based architecture.

Images, video, audio, point clouds

[1] Perceiver: General Perception with Iterative Attention, Jeagle et al. (2021)

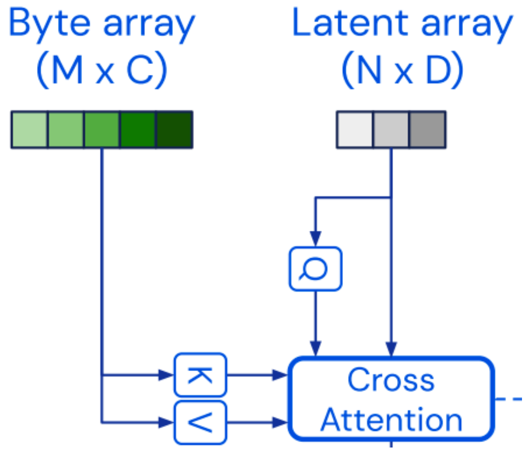
The Perceiver: main idea

- Cross-attention module to project a high-dimensional input byte array to a fixed-dimensional latent bottleneck (M>>N).
- Further processing using a deep stack of Transformer-style self-attention blocks
- Iteratively attends to the input byte array by alternating cross-attention and latent self-attention blocks.



The Perceiver: Taming quadratic complexity

- Challenge addressed: scaling attention architectures to very large and generic inputs.
- Main difficulty: quadratic complexity of QKV self-attention
- The Perceiver: **asymmetry in the attention** mechanism -->
  - K and V are projections of the input byte array and Q is a projection of a **learned latent array** with dimension N << M, where N is a hyperparameter.
- The resulting cross-attention operation has complexity O(MN).



Summary

- Transformer is one of the mostly used deep learning architecture.
- Transformer achieves state-of-the-art results across many tasks, which prevously requred specific datasets and training regimes, such as:
  - Language tasks: BERT, GPT
  - Multimodal tasks & learning: CLIP, Flamingo
  - Vision: Vision Transformer
  - Beyond: Perceiver
- The successful recipe is to do self-supervised pre-training on large datasets and then to fine-tune on specific tasks with a very small architectural change.