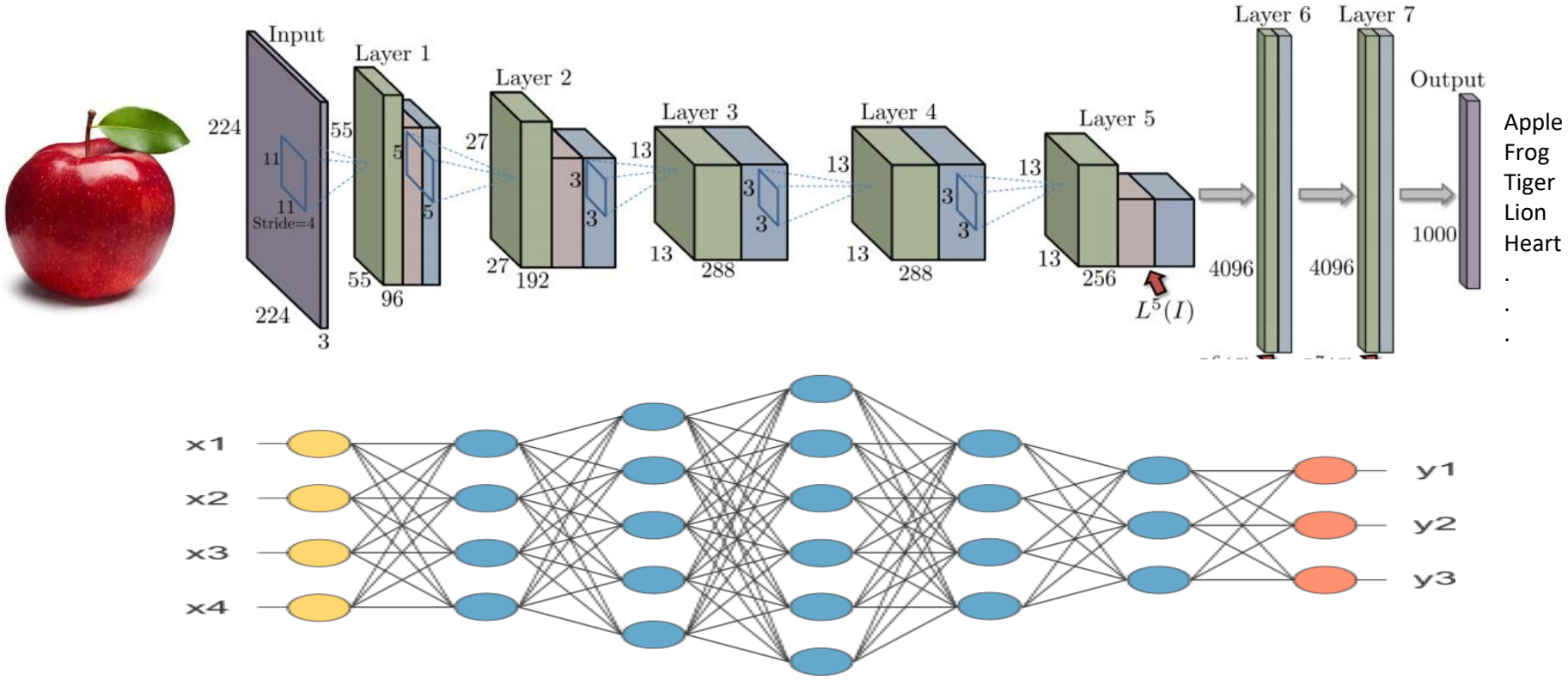
Class 10: Attention and Transformers

Petia Radeva
Universitat de Barcelona

Until now: CNN

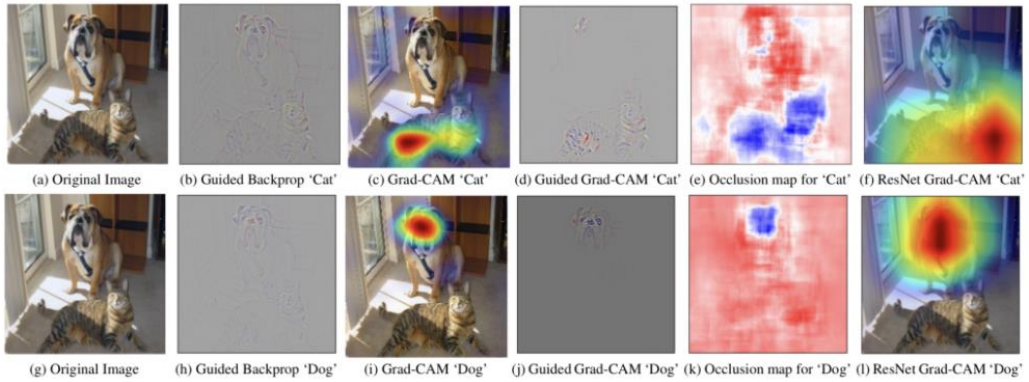


What is the restriction regarding the input?

00:49

Class Activation Maximization Recap

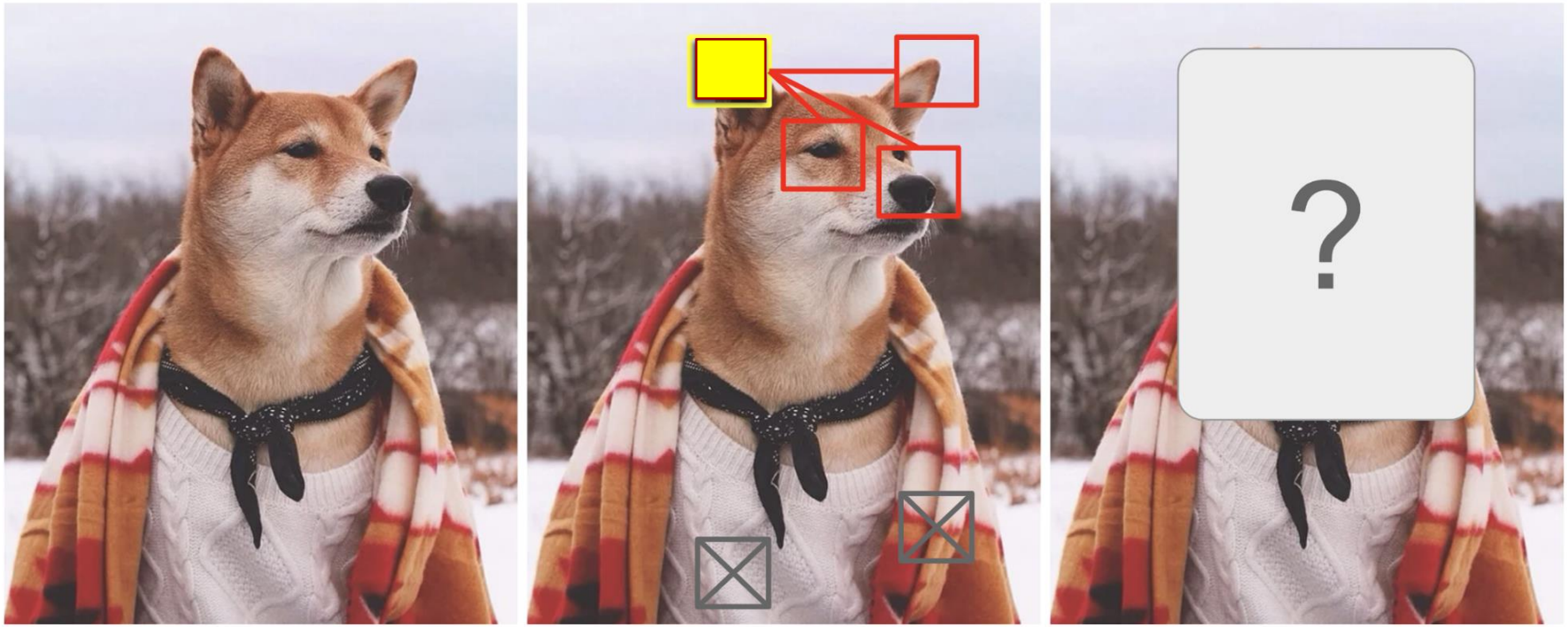
- Both CAM and Grad-CAM are local backpropagation-based interpretation methods.



- They are model-specific as they can be used exclusively for interpretation of convolutional neural networks.
- Are they affecting the training process?

Selective attention

Attention mechanism

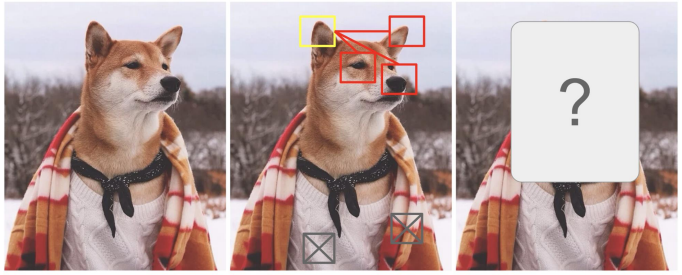


What should come in the yellow box?
What visual cues did you use to guess it?

Attention mechanism

- **Human visual attention allows us to focus on a certain region** with “high resolution” while perceiving the surrounding image in “low resolution” and then adjust the focal point or do the inference accordingly.
- The attention mechanism in Neural Networks tends to **mimic the cognitive attention** possessed by human beings.
- The **main aim** of this function is to emphasize the important parts of the information and try to de-emphasize the non-relevant parts.
- Since working memory in both humans and machines is limited, this process is **key to not overwhelming a system's memory.**
- In deep learning, **attention can be interpreted as a vector of importance weights**.

- When we predict an element, which could be a pixel in an image or a word in a sentence, we use the attention vector to infer **how much is it related to the other elements**.



Attention mechanism

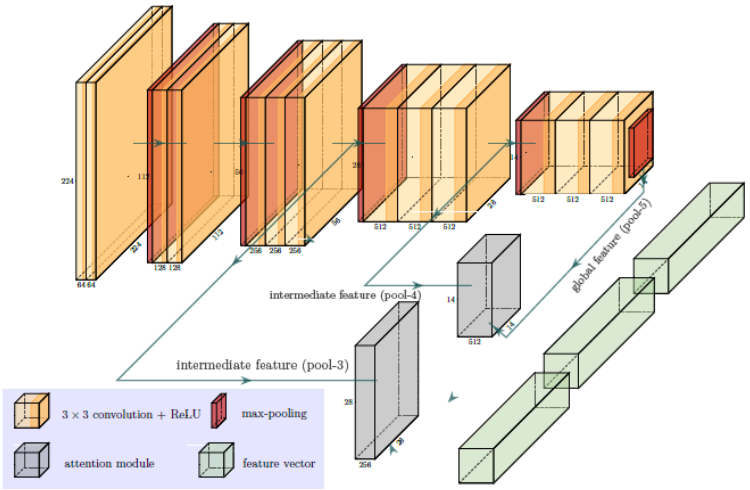
- When training an image model, we want the model to be able to **focus on important parts** of the image.
  - One way of accomplishing this is through **trainable attention** mechanisms.

- It is necessary to be able to **interpret** the model.
  - It is important to understand **which part of the image contributes** more towards the final decision.

- Post-hoc analysis like **Grad-CAM** are **not the same** as attention.
  - They are **not intended to change** the way the **model learns**, or to change what the model learns.
  - They are applied to an **already-trained model** with fixed weights, and
  - are intended solely to **provide insight** into the model's decisions.

Model - VGG16 with Attention

- **VGG16 (**Simonyan'2014**)** is 16 layers deep, and the design **won the ImageNet competition in 2014**.
- They use convolution layers of 3x3 filter size with a stride of 1 and ReLu as its activation function.
- The Maxpooling layer has 2x2 filters with stride 2.
- In the end there are 2 Dense (e.i. fully connected) layers followed by a softmax layer.
- **Two attention modules** are applied (the gray blocks) after pool3 and pool5. Why?

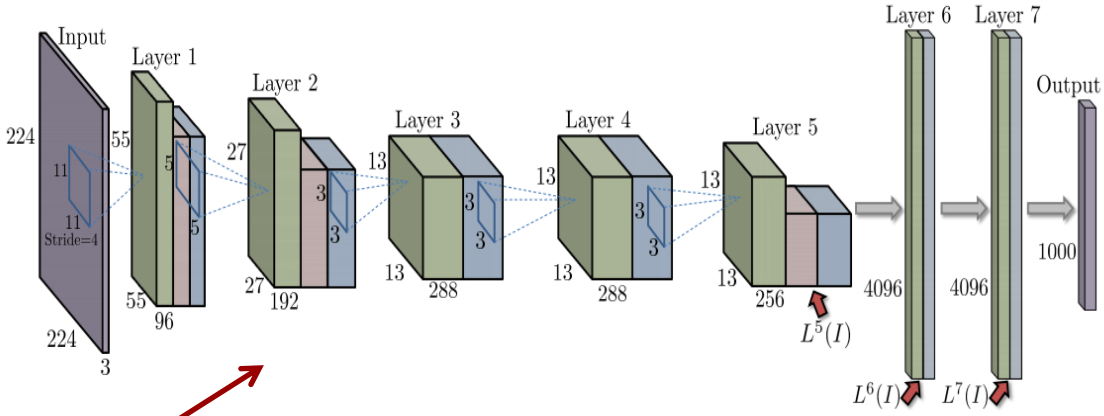


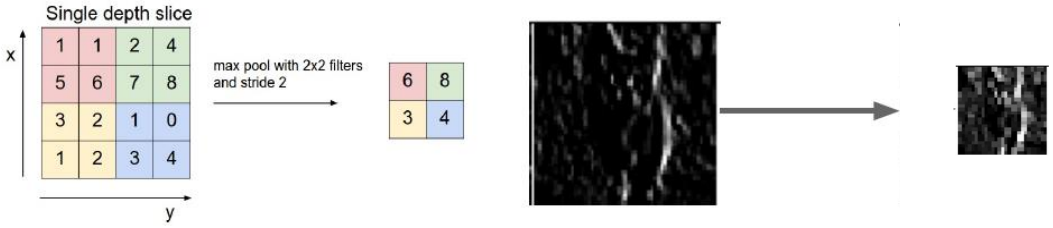
Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. 113364 cites

Max-pooling layer: Recap

Spatial info

No spatial info



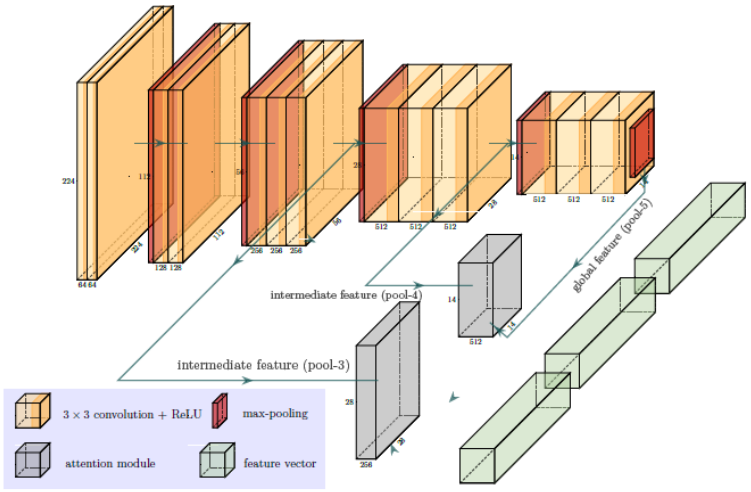


Max-pool layer

Model - VGG16 with Attention

- **VGG16 (**Simonyan'2014**)** is 16 layers deep, and the design **won the ImageNet competition in 2014**.

- **Two attention modules** are applied (the gray blocks).
  - The output of intermediate feature maps (pool-3 and pool-4) are used to infer attention maps.
  - Output of pool-5 serves as a form of global-guidance because the last stage feature contains the most abstract and compressed information over the entire image.



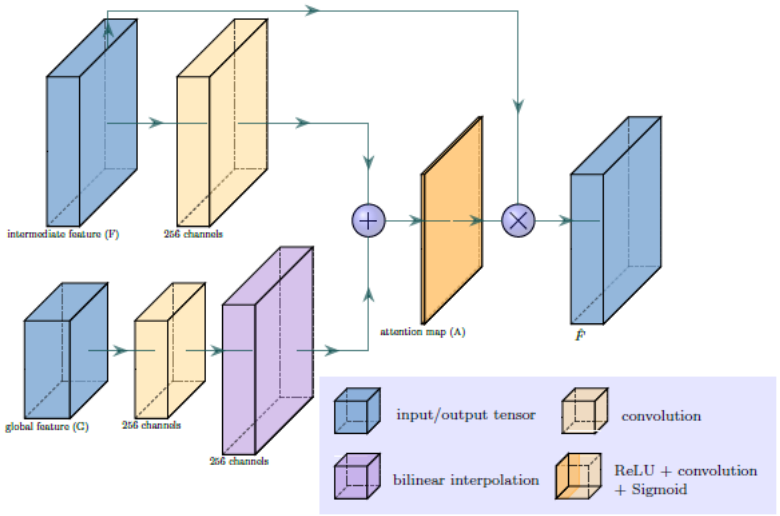
•The three feature vectors (green blocks) are computed via global average pooling and are concatenated together to form the final feature vector, which serves as the input to the classification layer(not shown here).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. 113364 cites

Attention layer

• The intermediate feature vector (F) is the output of pool-3 or pool-4 and the global feature vector (output of pool-5) is fed as input to the attention layer.

• Both the feature vectors pass through a convolution layer.
  • When the spatial size of global and intermediate features are different, feature **upsampling** is done via bilinear interpolation.
  •The *up_factor* determines by **what factor** the convoluted global feature vector has to be upscaled.



•After that an **element wise sum** is done followed by a convolution operation that just reduces the 256 channels to 1.

•This is then fed into a Softmax layer, which gives us a **normalized Attention map (A).**
  •Each scalar element in A represents the degree of attention to the corresponding spatial feature vector in F.

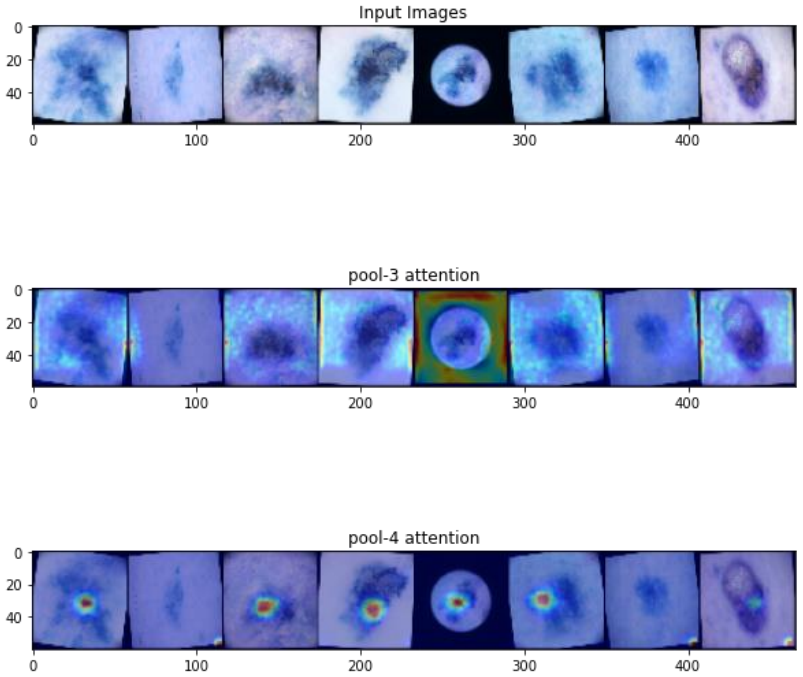
•The new feature vector F' is then computed by ***pixel-wise*** **multiplication**
  •i.e. each feature vector f is multiplied by the attention element a

•So, **the attention map A and the new feature vector F' are the outputs of the Attention Layer.**

Illustration on Melanoma Classification

•The shallower layer (pool-3) tends to focus on more general and diffused areas, while the deeper layer (pool-4) is more concentrated, focusing on the lesion and avoiding irrelevant pixels.

•Since most images are benign, pool-3 tries to learn some areas but pool-4 eventually minimizes the activated regions because the image is benign.

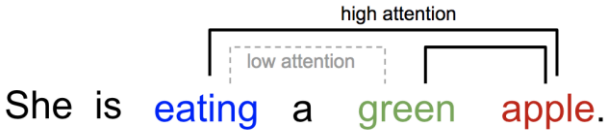


Note!: In this case, due to the elimination of Dense Layers, the number of parameters are greatly reduced and the network is lighter to train.

Yan, Y., Kawahara, J., & Hamarneh, G. (2019). Melanoma recognition via visual attention. In *Information Processing in Medical Imaging: 26th International Conference, IPMI 2019, Hong Kong, China, June 2–7, 2019, Proceedings 26* (pp. 793-804). Springer International Publishing.

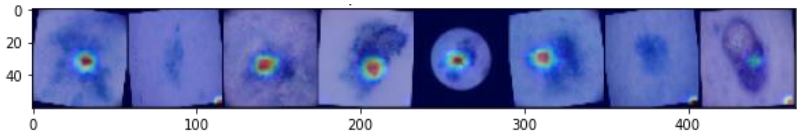
What about text analysis and attention?

- Similarly, we can explain the **relationship between words** in one sentence or close context.
  - When we see “eating”, we expect to encounter a food word very soon.
  - The colour term describes the food, but probably not so much with “eating” directly.

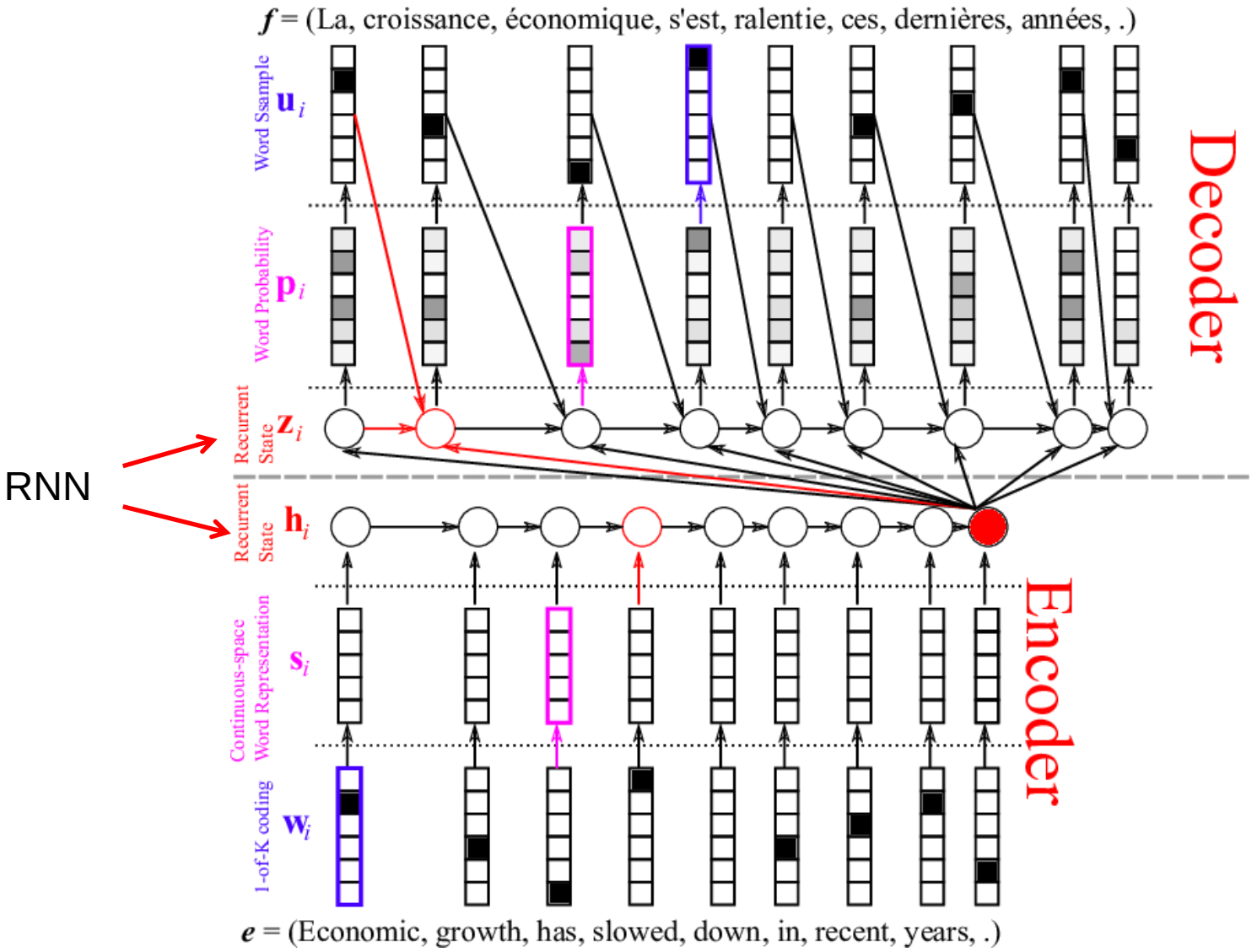


One word "attends" to other words in the same sentence differently.

- **Attention in deep learning** can be broadly interpreted as a vector of **importance weights:**
  - in order to predict or infer one element, such as a pixel in an image or a word in a sentence, we **estimate using the attention vector how strongly it is correlated** with (or “*attends to*”) other elements and
  - take the sum of their values weighted by the attention vector as the approximation of the target.



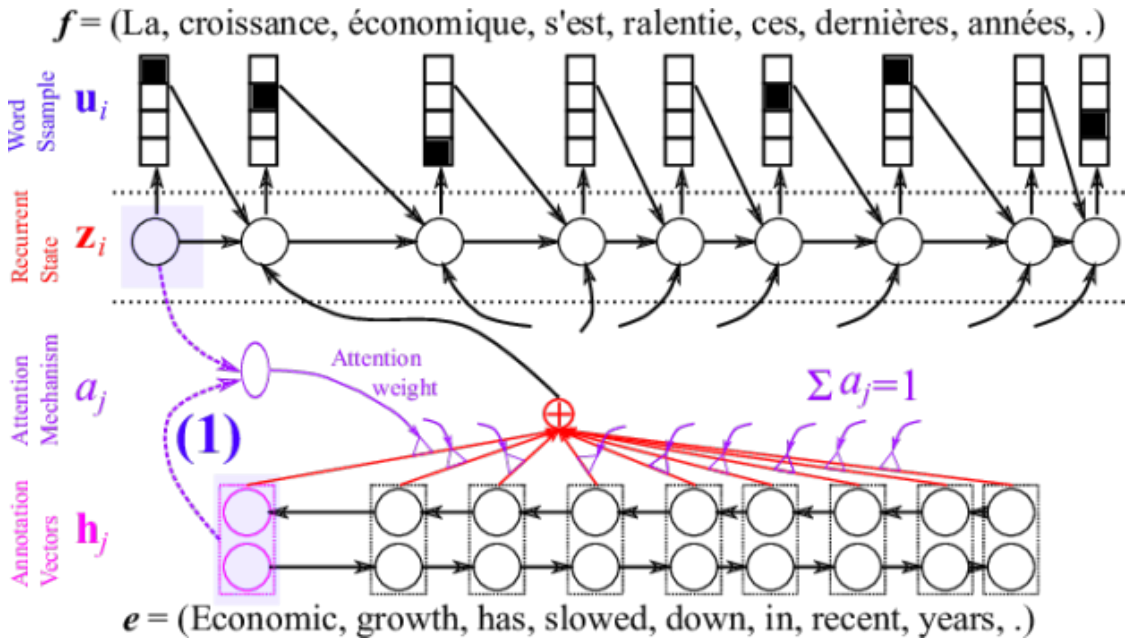
Encoder-Decoder machine translation



Attention

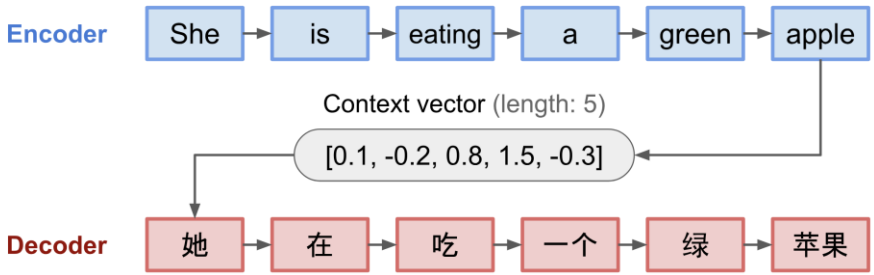
- Attention is that rather than compressing the input, it might be better for the **decoder to revisit the input sequence at every step.**
- Rather than always seeing the same representation of the input, one might imagine that the **decoder should selectively focus on particular parts** of the input sequence at particular decoding steps.
- Attention mechanism provides a simple means by which the decoder could dynamically *attend* to different parts of the input at each decoding step.
  - The high-level idea is that the **encoder could produce a representation** of length equal to the original input sequence.
  - At decoding time, **the decoder can (via some control mechanism) receive as input a context vector consisting of a weighted sum of the representations on the input** at each time step.
  - Intuitively, the **weights determine the extent to which each step's context "focuses**" on each input token,
  - the key is to **make this process for assigning the weights differentiable** so that it can be learned along with all of the other neural network parameters.

Attention


f = (La, croissance, économique, s'est, ralentie, ces, dernières, années, .)
Word Ssample
u_i
Recurrent State
z_i
Attention Mechanism
a_j
Attention weight
Σ a_j=1
(1)
Annotation Vectors
h_j
e = (Economic, growth, has, slowed, down, in, recent, years, .)

Seq2Seq model

- The **seq2seq** model was born in the field of language modelling.
  - It aims **to transform an input sequence (source) to a new one (target)** and both sequences can be of arbitrary lengths.
  - Examples of transformation tasks include machine translation between multiple languages in either text or audio, question-answer dialog generation, or even parsing sentences into grammar trees.
- The seq2seq model normally has an **encoder-decoder architecture**, composed of:
  - An **encoder** processes the input sequence and compresses the information into a context vector (also known as sentence embedding or "thought" vector) of a *fixed length*.
  - This representation is expected to be a good summary of the meaning of the *whole* source sequence.



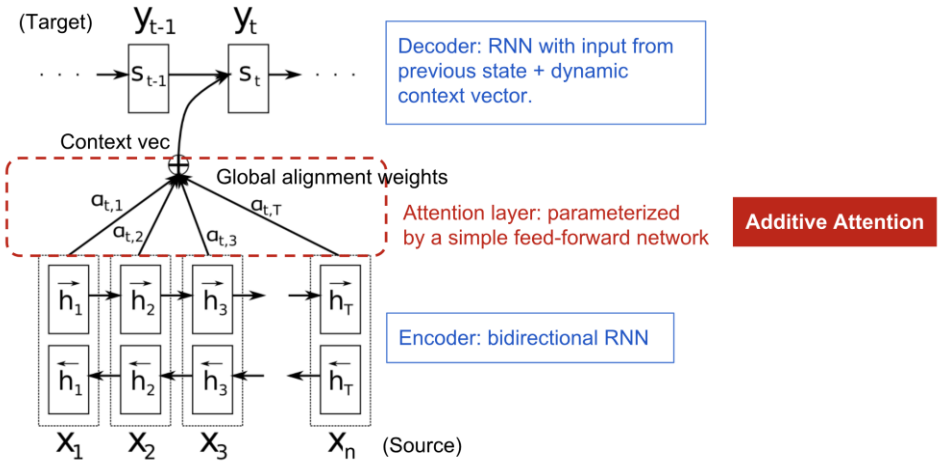
The encoder-decoder model, translating the sentence "she is eating a green apple" to Chinese. The visualization of both encoder and decoder is unrolled in time.

- A **decoder** is initialized with **the context vector to emit the transformed output**.
  - The early work only used the **last state** of the encoder network as the decoder initial state.
  - A **critical and apparent disadvantage** of this fixed-length context vector design is incapability of remembering long sentences.
    - Often it has forgotten the first part once it completes processing the whole input.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." *Advances in neural information processing systems* 27 (2014) 24160 cites.

Seq2Seq model

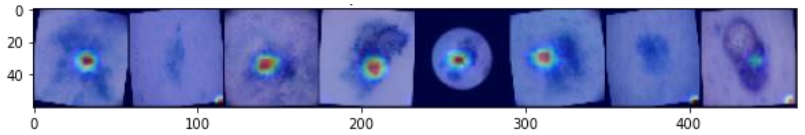
- The attention mechanism was born to help **memorize long source sentences** in neural machine translation (NMT).
  - Rather than building a single context vector out of the encoder's last hidden state**, create shortcuts between the context vector and the entire source input.**
  - The weights of these shortcut connections are customizable for each output element.
- While the context vector has access to the entire input sequence, don't worry about forgetting.
  - The alignment between the source and target is learned and controlled by the context vector.



The encoder-decoder model with additive attention mechanism

The context vector consumes three pieces of information:

- **encoder** hidden states;
- **decoder** hidden states;
- **alignment** between source and target.



Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*. 31806 cites

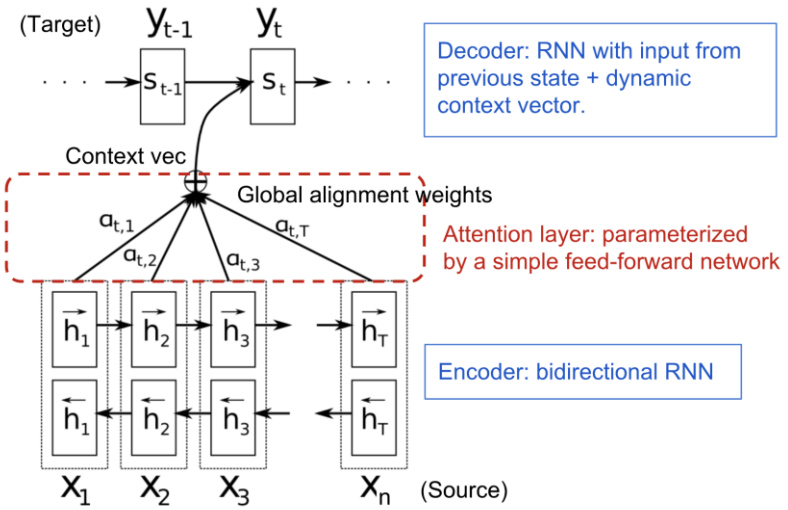
Attention mechanism in NMT

- Given a source sequence x of length n, **output a target sequence** y of length m:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]$$
$$\mathbf{y} = [y_1, y_2, \ldots, y_m]$$

- The encoder is a bidirectional RNN with a forward hidden state $\overrightarrow{\boldsymbol{h}}_i$ and a backward one $\overleftarrow{\boldsymbol{h}}_i$.
- A simple concatenation of two represents the encoder state.
  - The motivation is to include both the preceding and following words in the annotation of one word.

$$\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i^\top; \overleftarrow{\boldsymbol{h}}_i^\top]^\top, i = 1, \ldots, n$$



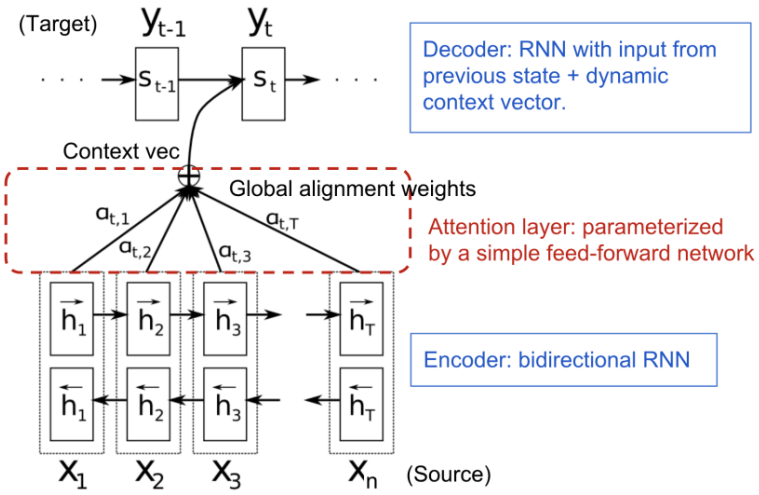
https://lilianweng.github.io/posts/2018-06-24-attention/?ref=blog.paperspace.com

Attention mechanism in NMT

- The decoder network has hidden state:

$$\boldsymbol{s}_t = f(\boldsymbol{s}_{t-1}, y_{t-1}, \mathbf{c}_t)$$

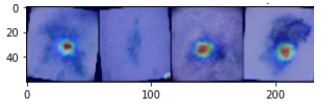
for the output word at position t, t=1,…,m, where the context vector c_t is a sum of hidden states of the input sequence, weighted by alignment scores:



$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i \qquad ; \text{Context vector for output } y_t$$

$$\alpha_{t,i} = \text{align}(y_t, x_i) \qquad ; \text{How well two words } y_t \text{ and } x_i \text{ are aligned.}$$

$$= \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))} \qquad ; \text{Softmax of some predefined alignment score..}$$



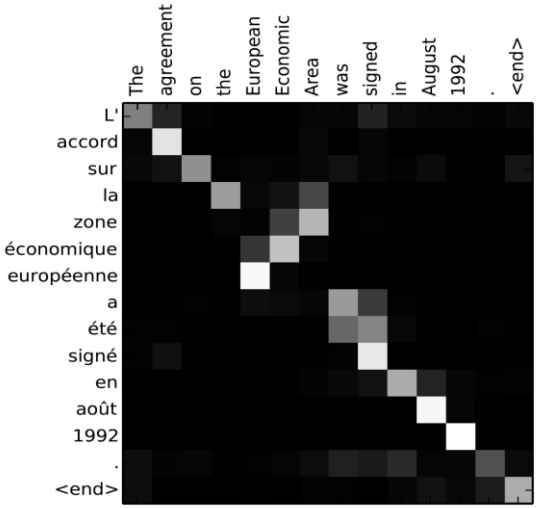
https://lilianweng.github.io/posts/2018-06-24-attention/?ref=blog.paperspace.com

Attention mechanism in NMT

- The alignment model assigns a score $a_{t,i}$ to the pair of input at position i and output at position t, (y_t,x_i), based on how well they match.
- The set of $\{a_{t,i}\}$ are weights defining how much of each source hidden state should be considered for each output.
- The **alignment score** a is parametrized by a **feed-forward network** with a single hidden layer and this network is jointly trained with other parts of the model.
- The **score function** is therefore in the following form, given that tanh is used as the non-linear activation function:

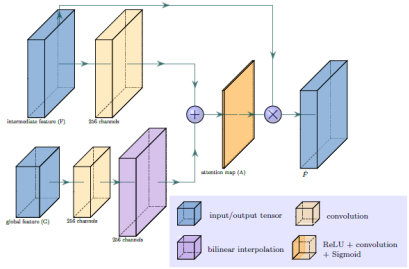
$$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$$

- where both v_a and W_a are weight matrices to be learned in the alignment model.
- The matrix of alignment scores is a nice byproduct to explicitly show the correlation between source and target words.



Alignment matrix of "L'accord sur l'Espace économique européen a été signé en août 1992" (French) and its English translation "The agreement on the European Economic Area was signed in August 1992".

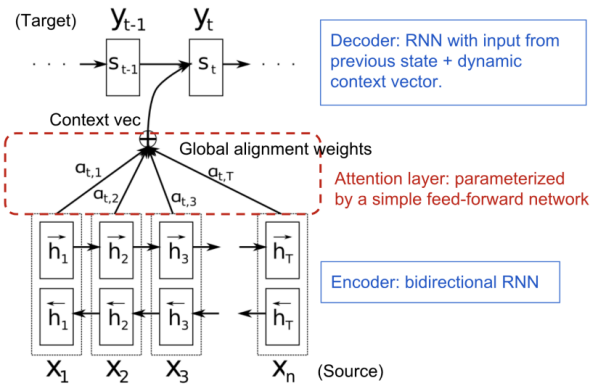
$$\begin{aligned}
\mathbf{c}_t &= \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i && \text{; Context vector for output } y_t \\
\alpha_{t,i} &= \text{align}(y_t, x_i) && \text{; How well two words } y_t \text{ and } x_i \text{ are aligned.} \\
&= \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))} && \text{; Softmax of some predefined alignment score..}
\end{aligned}$$



Popular attention mechanisms and alignment score functions

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_{t-1}; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

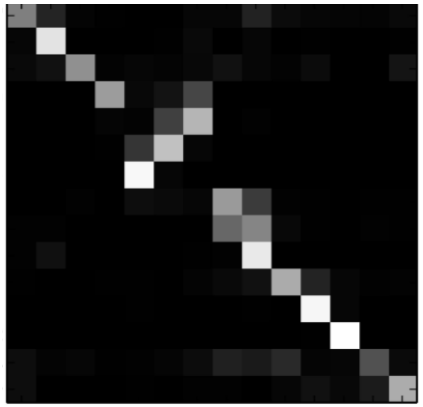
Which of them are learnable?



Popular attention mechanisms

Name	Definition	Citation
Self-Attention(&)	Relating different positions of the same input sequence. Theoretically the self-attention can adopt any score functions above, but just replace the target sequence with the same input sequence.	Cheng2016
Global/Soft	Attending to the entire input state space.	Xu2015
Local/Hard	Attending to the part of input state space; i.e. a patch of the input image.	Xu2015; Luong2015

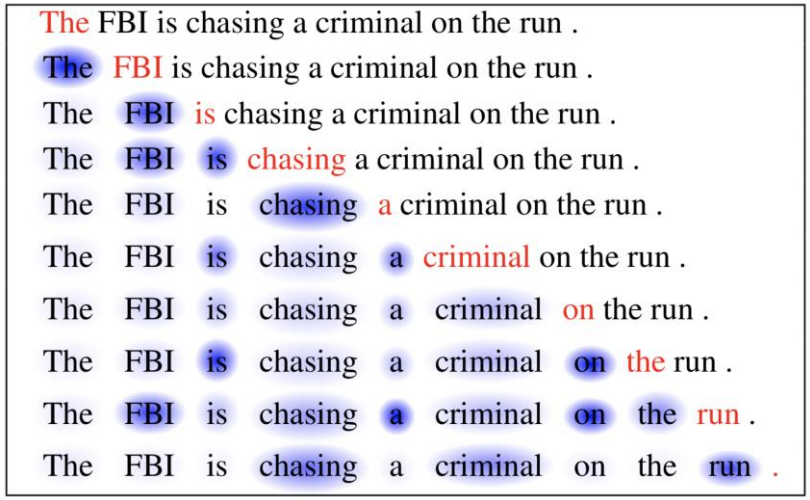
(&) Also, referred to as "intra-attention" in Cheng et al., 2016 and some other papers.



Self-Attention

- **Self-attention**, also known as **intra-attention**, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence.

- It has been shown to be very useful in machine reading, abstractive summarization, or image description generation.



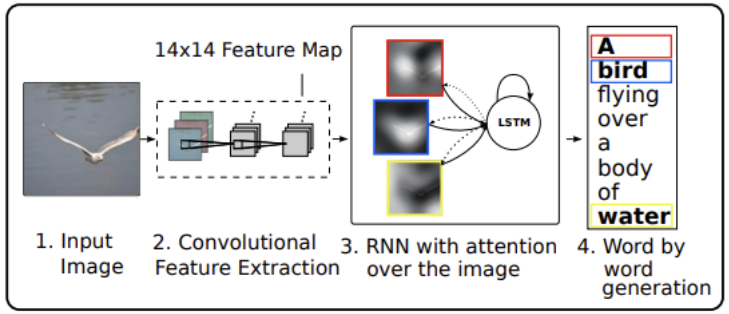
The current word is in red and the size of the blue shade indicates the activation level. (Image source: Cheng et al., 2016)

Soft vs Hard Attention

- In the show, attend and tell paper, attention mechanism is applied to images to generate captions:

1. The image is encoded by a **CNN to extract features.**
2. A **RNN decoder** consumes the convolution features to **produce descriptive words** one by one, where the weights are learned through attention.

- The visualization of the **attention weights clearly demonstrates which regions** of the image the model is paying attention to so as to output a certain word.



The model learns a words/image alignment.

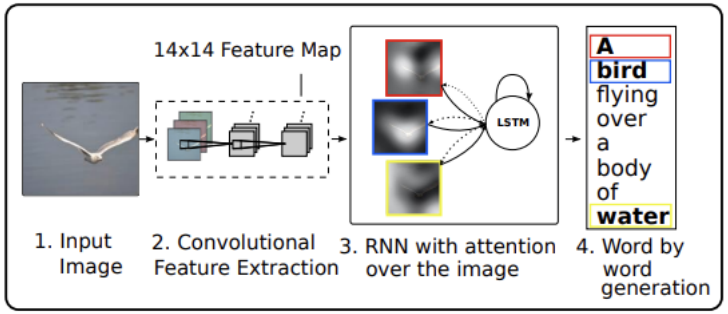
"A woman is throwing a frisbee in a park." (Image source: Fig. 6(b) in Xu et al. 2015)

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048-2057). PMLR. 11465 cites

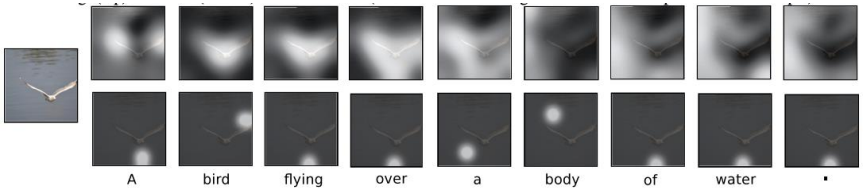
Soft vs Hard Attention

"Soft" vs "hard" attention, based on whether the attention has access to the entire image or only a patch:

- **Soft Attention**: the alignment weights are learned and placed "softly" over all patches in the source image (Bahdanau et al., 2015).
  - *Pro*: the model is smooth and differentiable.
  - *Con*: expensive when the source input is large.
- **Hard Attention**: only selects one patch of the image to attend to at a time.
  - *Pro*: less calculation at the inference time.
  - *Con*: the model is non-differentiable and requires more complicated techniques such as variance reduction or reinforcement learning to train. (Luong, et al., 2015)



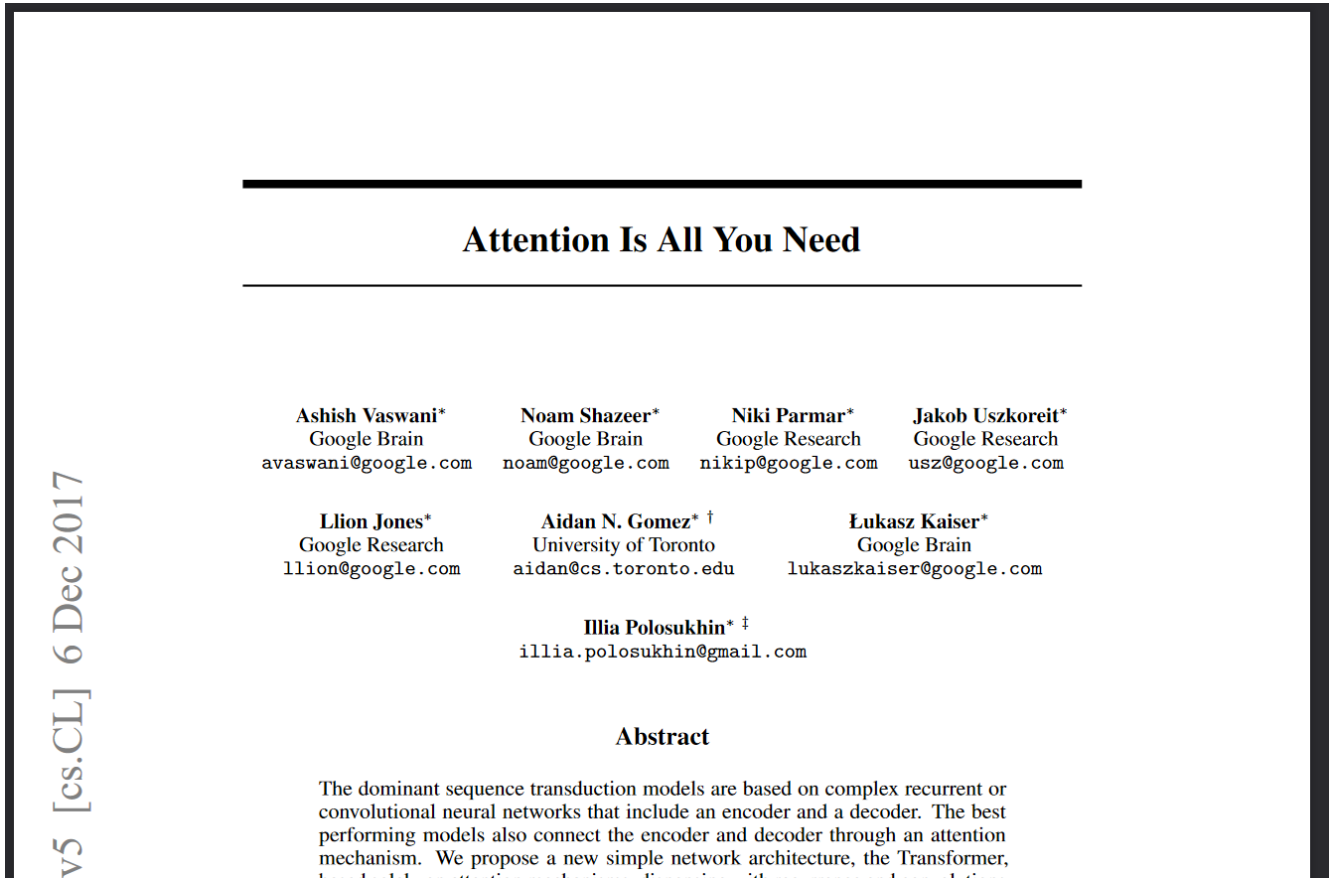
The model learns a words/image alignment.



Visualization of the attention for each generated word. The rough visualizations obtained by upsampling the attention weights and smoothing. (top)"soft" and (bottom) "hard" attention (note that both models generated the same captions in this example).

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048-2057). PMLR. 11465 cites

Transformers: Attention is all you need



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,

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*. 97903 cites

4th most influential google scholar paper in 2020

Attention is all you need

- "Attention is All you Need" is one of the most impactful and interesting papers in 2017.

- It presented a lot of improvements to the soft attention
  - **do seq2seq modeling *without* recurrent network units.**

- The proposed "**transformer**" model is **entirely built on the self-attention** mechanisms **without using sequence-aligned** recurrent architecture.

- The **secret recipe** is carried in its **model architecture**.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.
4th most influential google scholar paper in 2020

Key, Value and Query

- The major component in the transformer is the unit of ***multi-head self-attention mechanism*.**
- The transformer views the encoded representation of the input as a set of **key-value** pairs, (K,V), both of dimension n (input sequence length);
  - in the context of NMT, both **the keys and values** are the **encoder hidden states**.
  - In the **decoder**, the previous output is compressed into a **query** (Q of dimension m) and the next output is produced by **mapping this query and the set of keys and values**.
- The transformer adopts the scaled dot-product attention: the output is a weighted sum of the values, where the weight assigned to each value is determined by the dot-product of the query with all the keys:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}})\mathbf{V}$$

Intuition for Queries, Keys, and Values

- Consider a database. In the simplest form there are collections of keys (k) and values (v).
  - For instance, our database D might consist of tuples {(“Zhang”, “Aston”), (“Lipton”, “Zachary”), (“Li”, “Mu”), (“Smola”, “Alex”), (“Hu”, “Rachel”), (“Werness”, “Brent”)} with the last name being the key and the first name being the value.
  - We can operate on D, for instance with the exact query (q) for “Li” which would return the value “Mu”.
  - If (“Li”, “Mu”) was not a record in D, there would be no valid answer.
  - If we also allowed for approximate matches, we would retrieve (“Lipton”, “Zachary”) instead.

Queries, Keys and Values

- This quite simple and trivial example nonetheless teaches us a number of useful things:

- We can **design queries q that operate on (k,v) pairs** in such a manner as to be valid **regardless of the database size**.

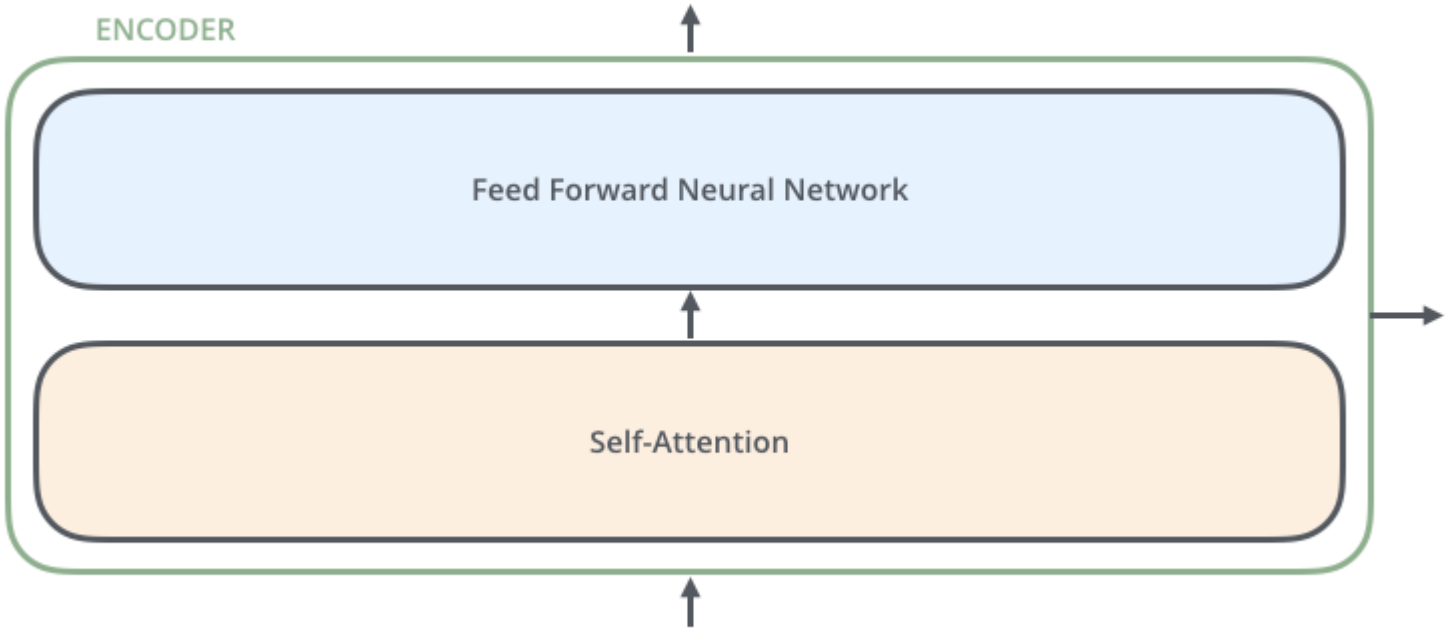
- The **same query can receive different answers**, according to the contents of the database.

- The **“code” being executed for operating on a large state space** (the database) can be quite **simple** (e.g., exact match, approximate match, top-k).

- There is **no need to compress or simplify the database** to make the operations effective.

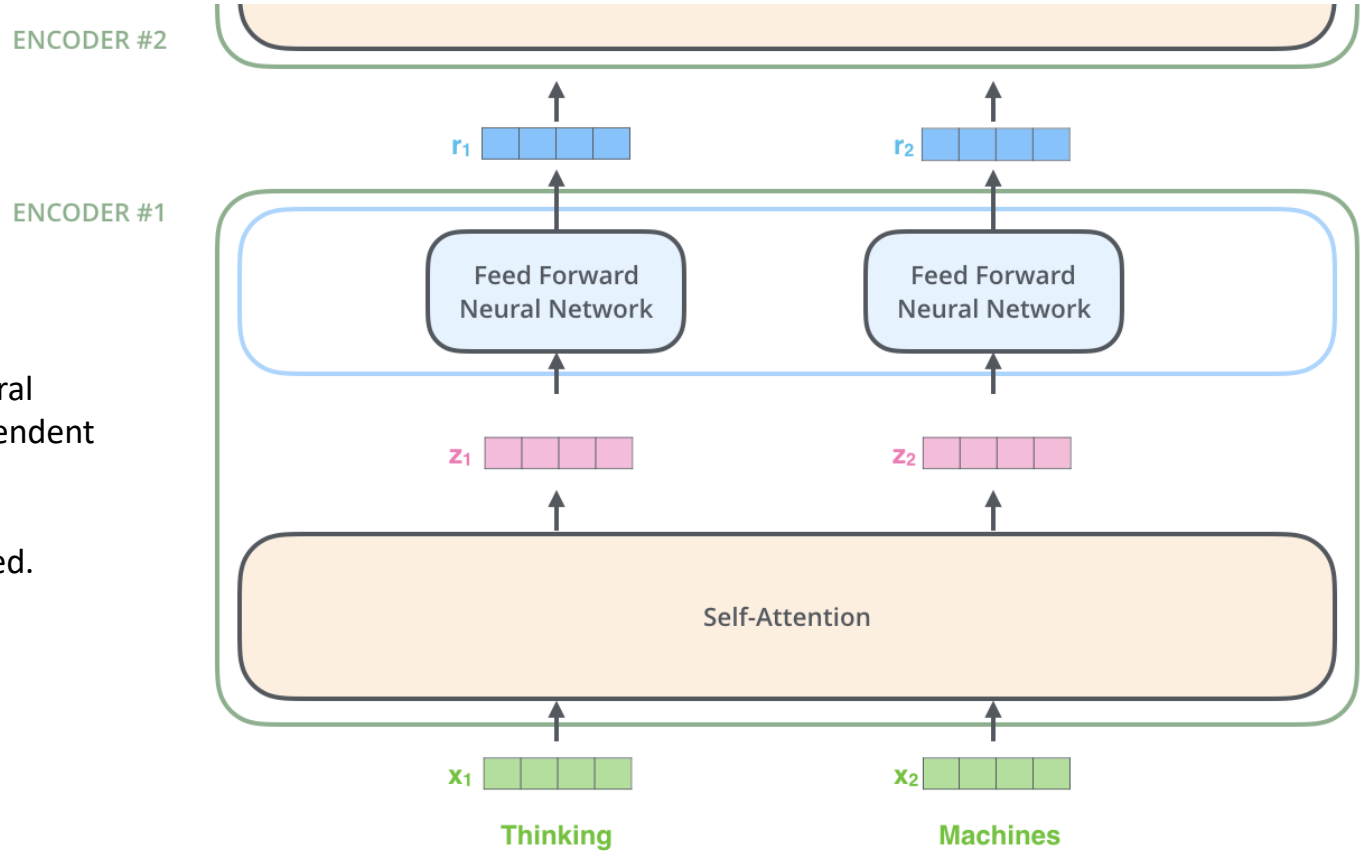
Transformers

An Encoder Block: same structure, different parameters



Transformers

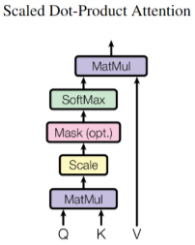
Encoder



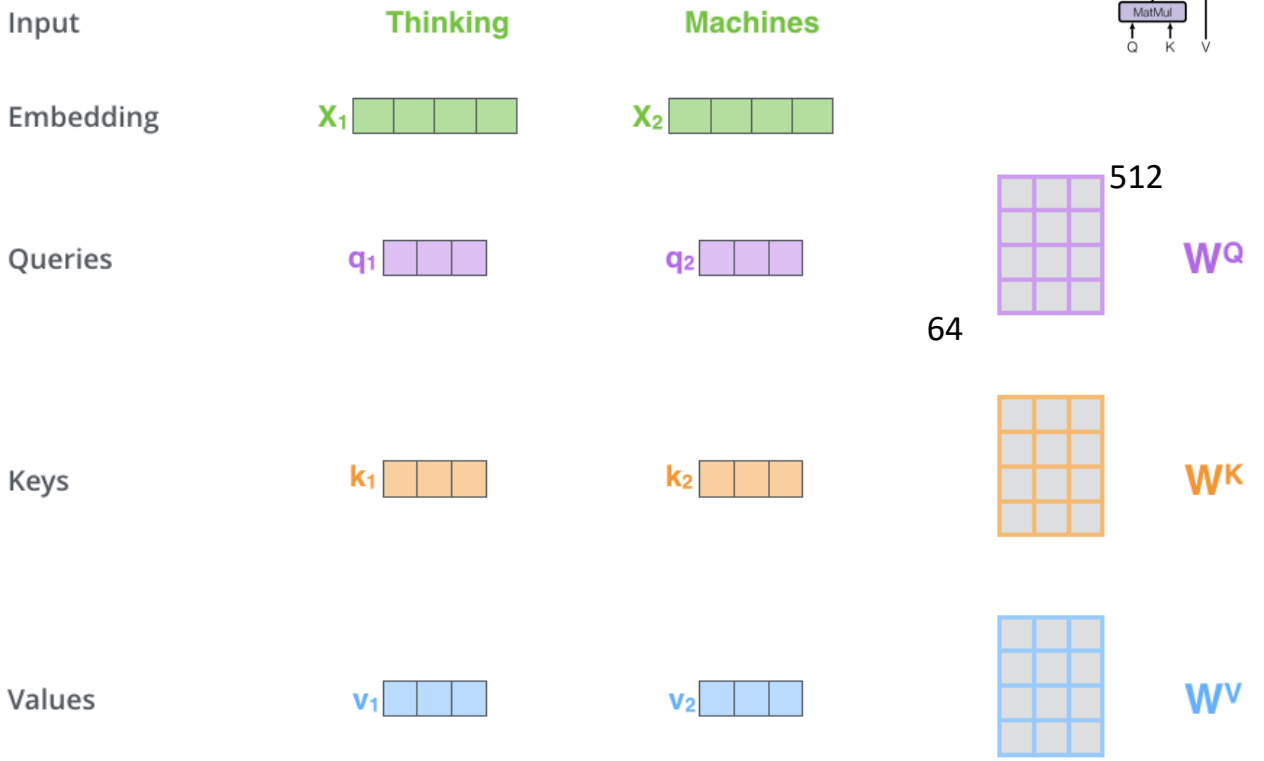
Note: The Forward Neural Network (ffnn) is independent for each word.

Hence can be parallelized.

Transformers



Self-Attention



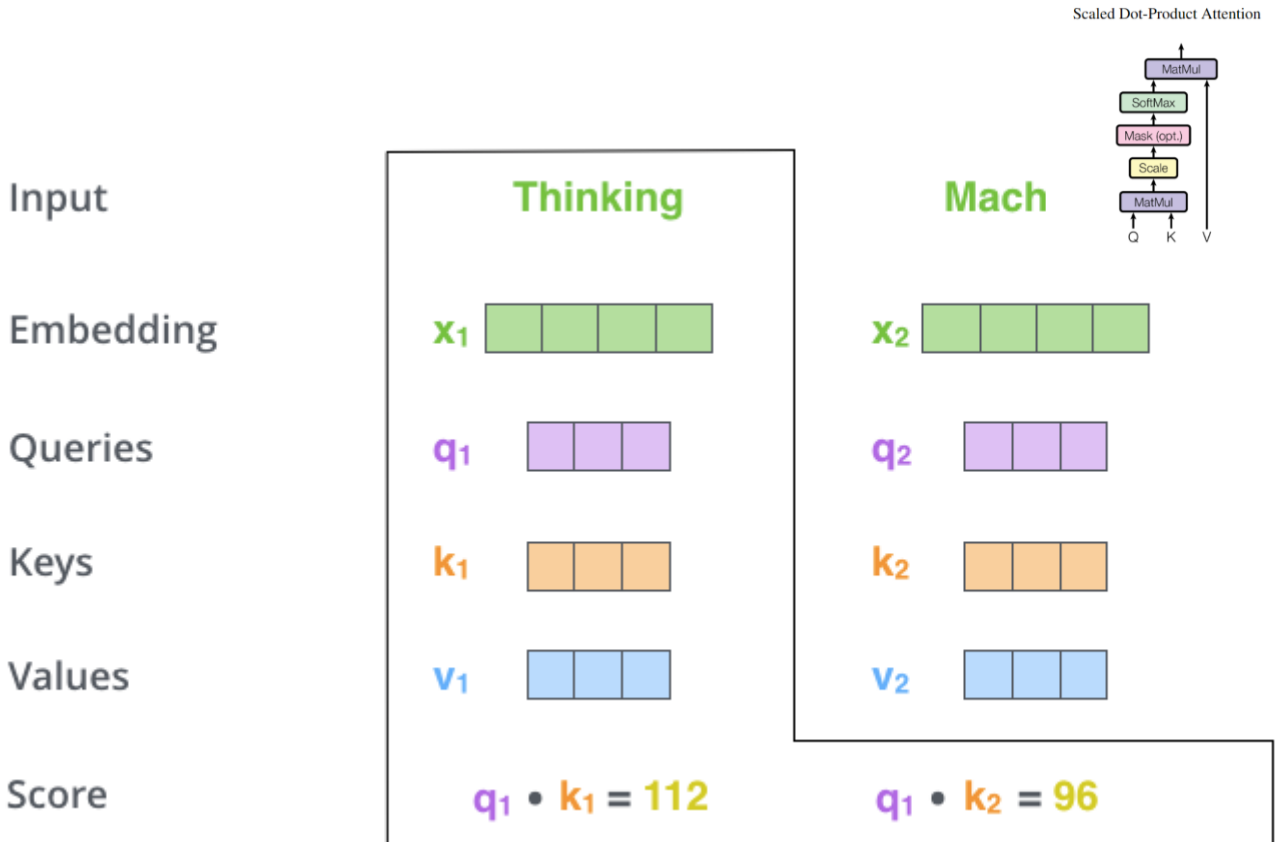
First we create three vectors by multiplying input embedding (512x1) x_i with three matrices (64x512):

$Q_i = W^Q x_i$
$K_i = W^K x_i$
$V_i = W^V x_i$

Attention and Transformers

Self-Attention

Now we need to calculate a score to determine how much focus to place on other parts of the input.



Attention and Transformers

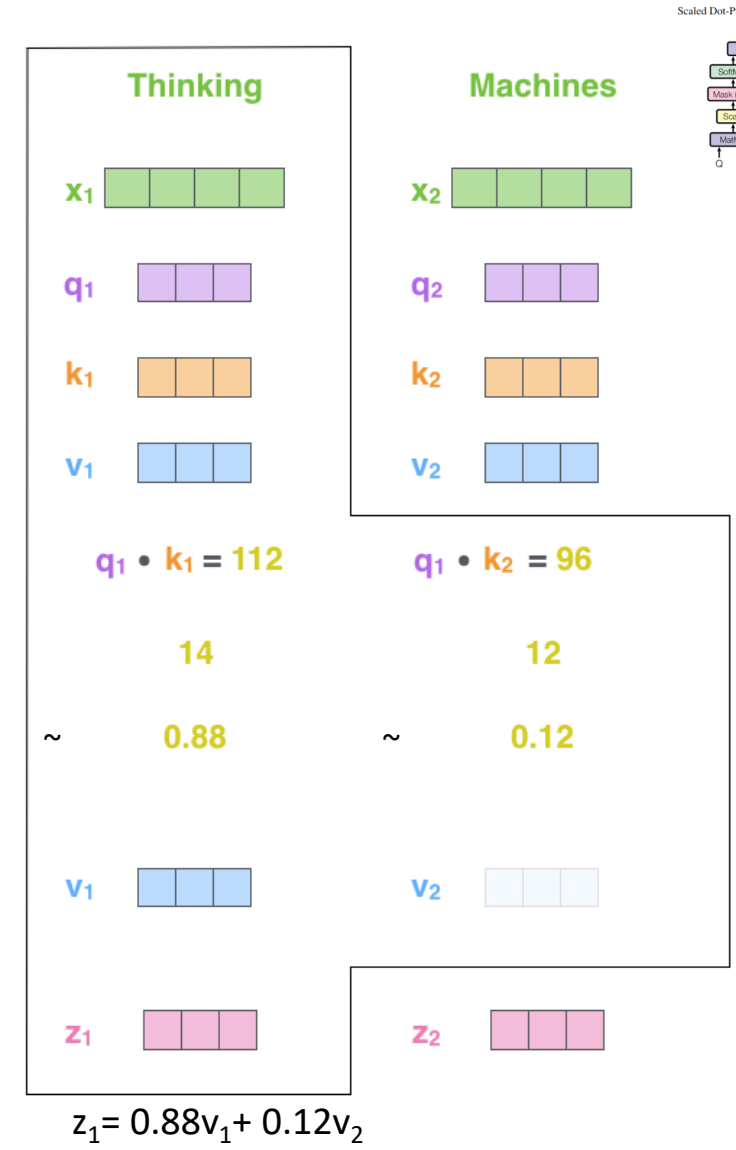
Self-Attention

Formula:

64x64 64x512

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

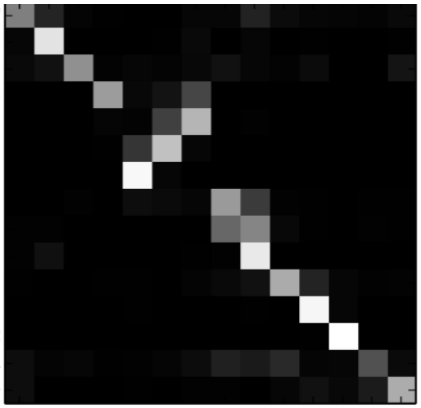
d_k=64 is the dimension of the key vector



$z_1 = 0.88v_1 + 0.12v_2$

Transformers vs Attention

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_{t-1}; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017



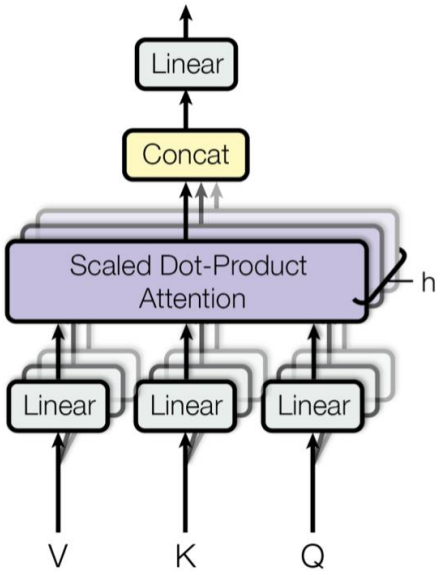
$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}})\mathbf{V}$$

where: $Q = W^Q x$, $K = W^K x$, $V = W^V x$

00:49

Transformers: **Multi-Head Self-Attention**

- Rather than only computing the attention once, the multi-head mechanism runs through the **scaled dot-product attention multiple times in parallel**.
  - The independent attention outputs are simply **concatenated and linearly transformed** into the expected dimensions.
- *Multi-head attention allows the model to jointly attend to information from different representation **subspaces** at different positions.*
  - *With a single attention head, averaging inhibits it.*



Multi-head scaled dot-product attention mechanism. (Vaswani, et al., 2017)

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{head}_1; \ldots; \text{head}_h]\mathbf{W}^O$$
$$\text{where head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

- where W_i^Q, W_i^K, W_i^V and W^O are parameter matrices to be learned.

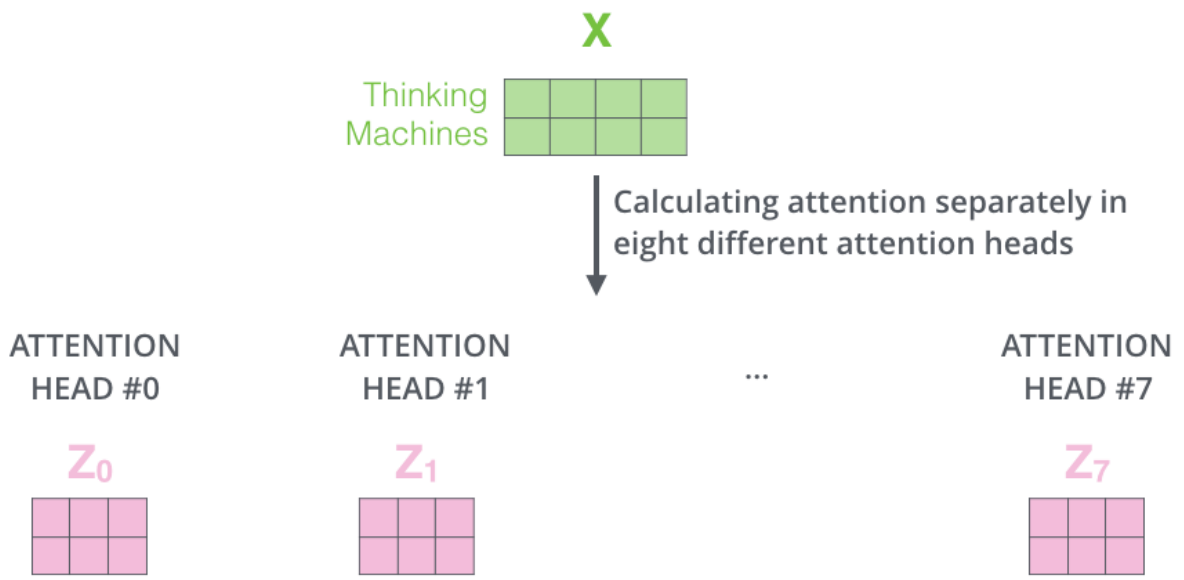
00:49

Attention and Transformers

Multiple heads

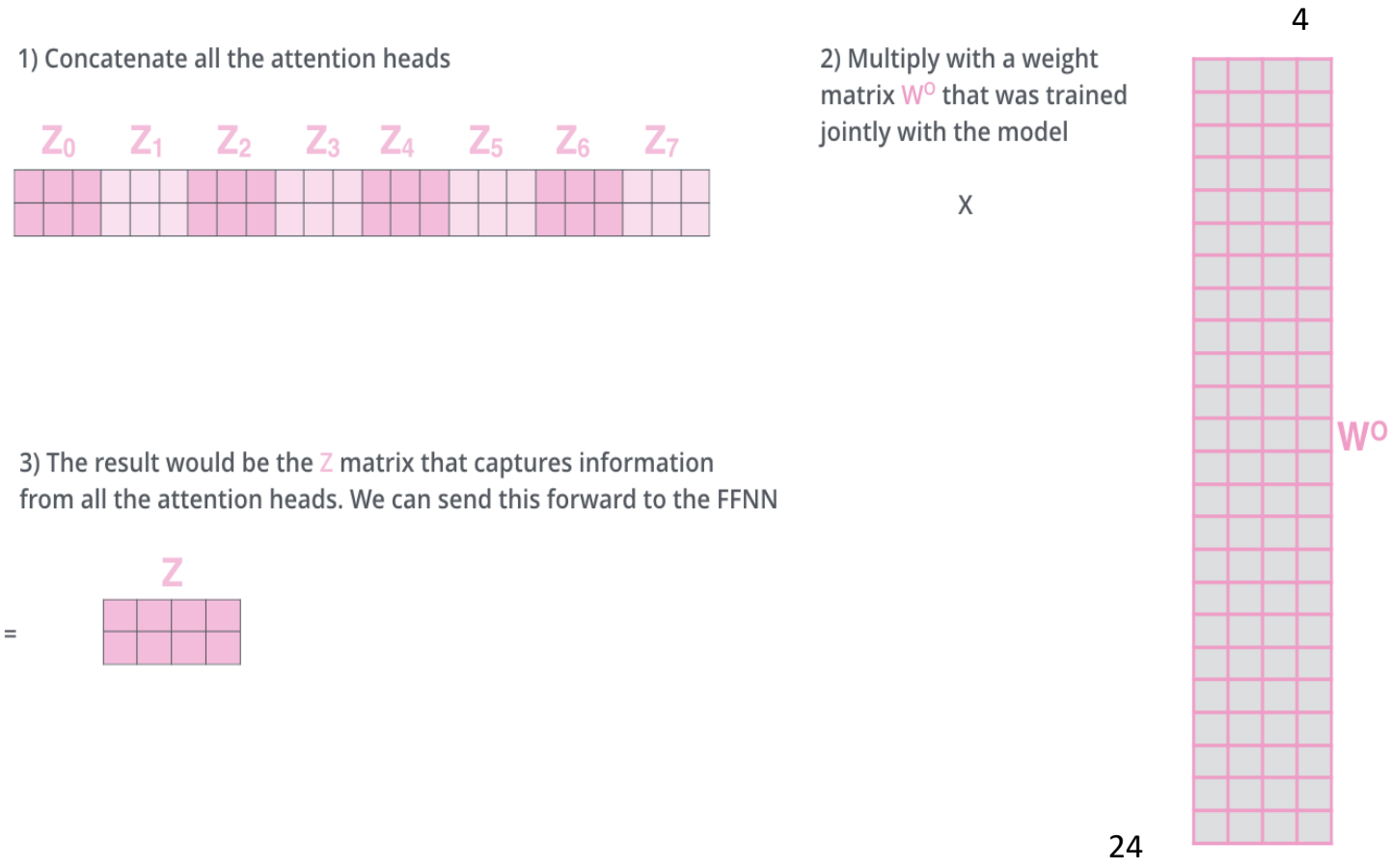
1. It expands the model's ability to focus on different positions.

2. It gives the attention layer multiple "representation subspaces"



Attention and Transformers

The output is expecting only a 2x4 matrix, hence,



https://www.youtube.com/watch?v=-9vVhYEXeyQ

Encoder

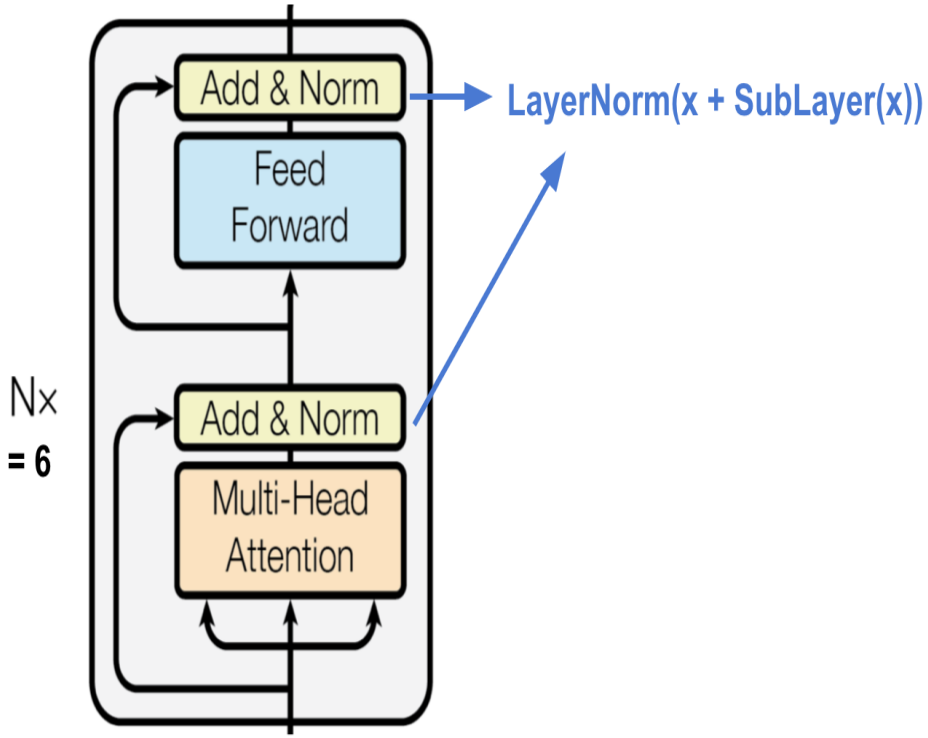
The encoder generates an attention-based representation with capability to locate a specific piece of information from a potentially infinitely-large context.

• A stack of N=6 identical layers.

• Each layer has a **multi-head self-attention layer** and a simple position-wise **fully connected feed-forward network**.

• Each sub-layer adopts a **residual** connection and a layer **normalization**.

- All the sub-layers output data of the same dimension d_{model}=512.

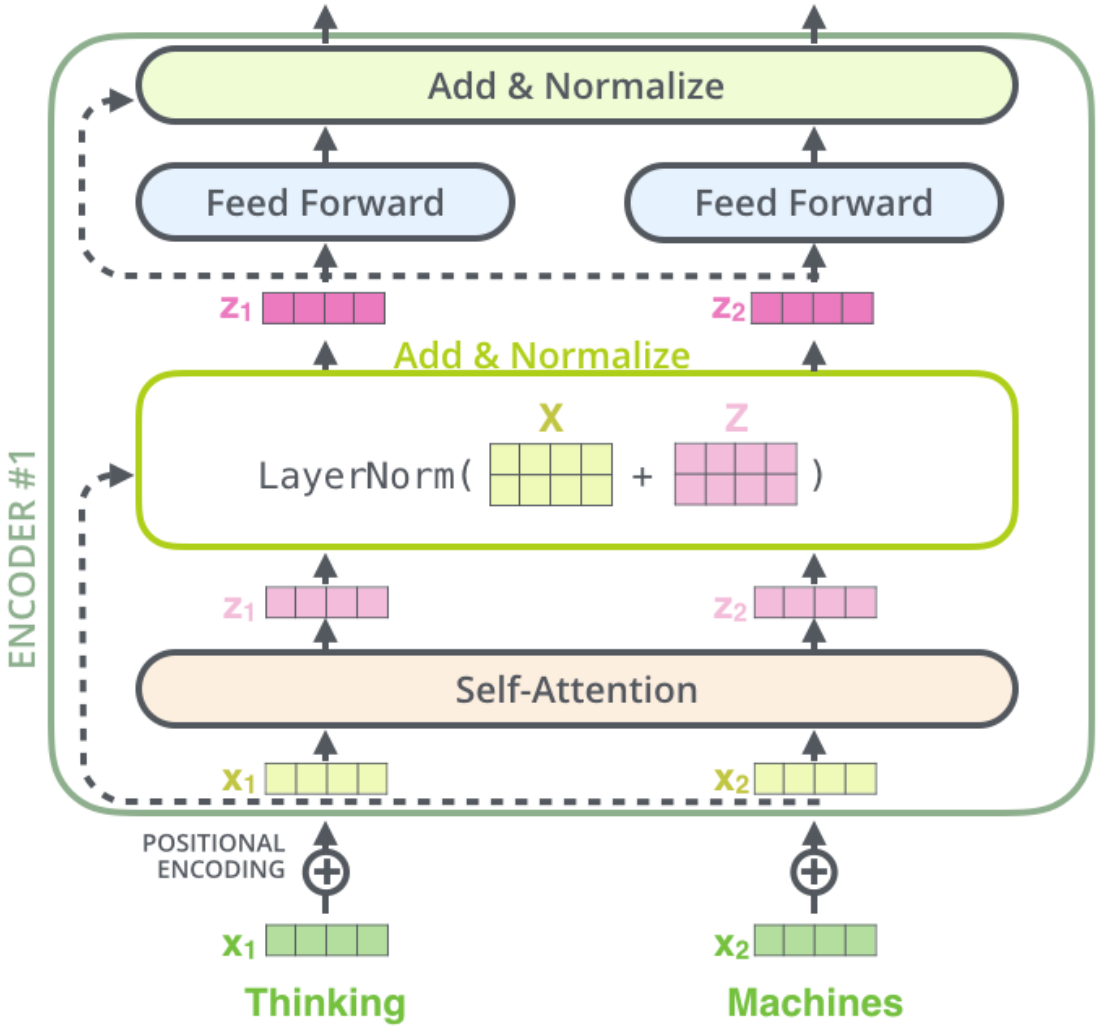


The transformer's encoder. (Image source: Vaswani, et al., 2017)

Attention and Transformers

Add and Normalize

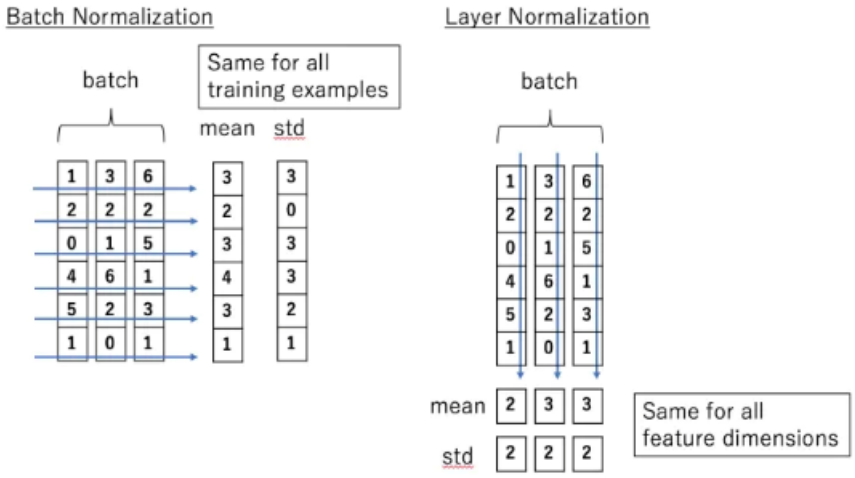
In order to regulate the computation, this is a normalization layer so that each feature (column) have the same average and deviation.

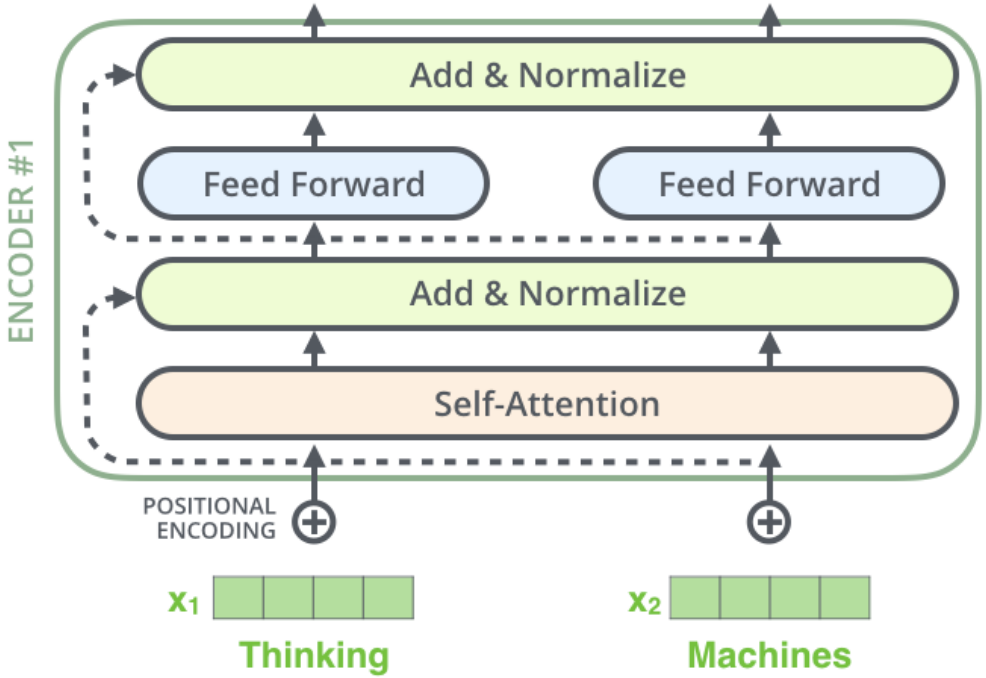


Attention and Transformers

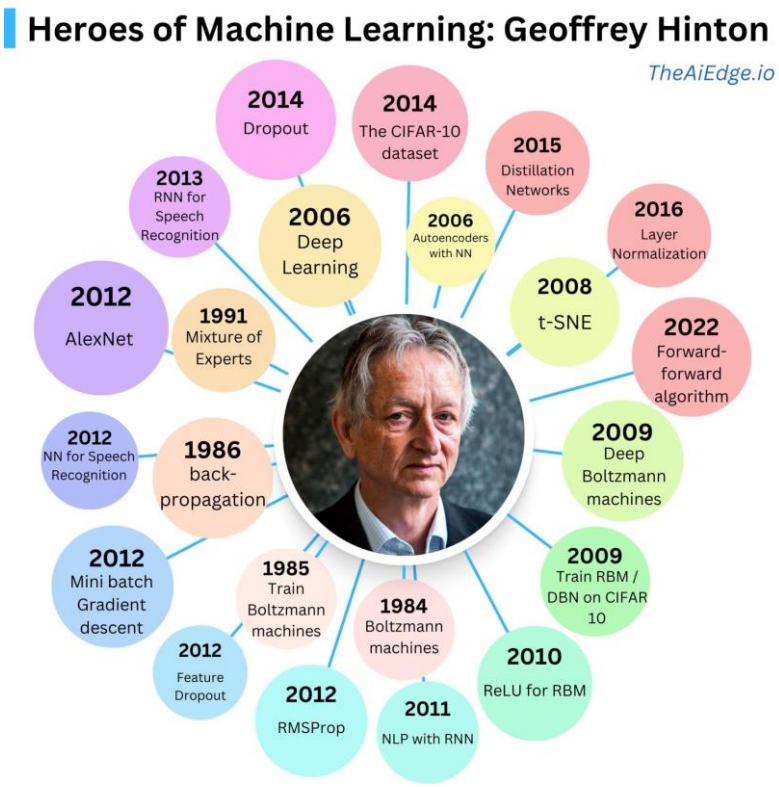
Layer Normalization (Hinton)

Layer normalization normalizes the inputs across the features.





Heroes of Machine learning: Geoffrey Hinton



- 1986 - He invents the Boltzmann machines
- 1985 - He proposes a new learning algorithm for Boltzmann machines
- 1986 - He is credited as one of the inventors of the Back-propagation algorithm
- 1991 - He invents the Mixture of Experts
- 2006 - He proposes an algorithm to train Deep Belief Nets leading to the term "Deep Learning"
- 2006 - He shows how to build Autoencoders with Neural Networks
- 2008 - He invents t-SNE, a new technique for dimension reduction
- 2009 - He presents an algorithm to train Deep Boltzmann machines
- 2009 - Trains Restricted Boltzmann Machines and Deep Belief Networks with the CIFAR-10 dataset
- 2010 - Shows the improved performance of Restricted Boltzmann Machines with ReLU
- 2011 - Shows how to build a generative text model with Recurrent Neural Networks
- 2012 - Invents RMSprop in a course lecture (!!!)
- 2012 - Proposes the Feature Dropout technique to improve networks
- 2012 - Suggests mini-batch gradient descent in a course lecture
- 2012 - Deep Learning for speech recognition
- 2014 - Revolutions CV capabilities with AlexNet (the most cited paper of his career)
- 2014 - He proposes the Dropout technique to reduce overfitting
- 2014 - the CIFAR 10 dataset is made available
- 2015 - He invents the Distillation Network to reduce the size of models
- 2016 - He invents the Layer Normation technique (used in Transformer architecture
- 2017- He proposes CapsNets, or Capsule networks aiming to overcome some limitations of CNNs, particularly in the area of understanding hierarchical relationships between objects and their parts within an image
- 2022 - He presents a new alternative to the Back-propagation algorithm: the Forward-forward algorithm

Decoder

The **decoder** is able to retrieve from the encoded representation.

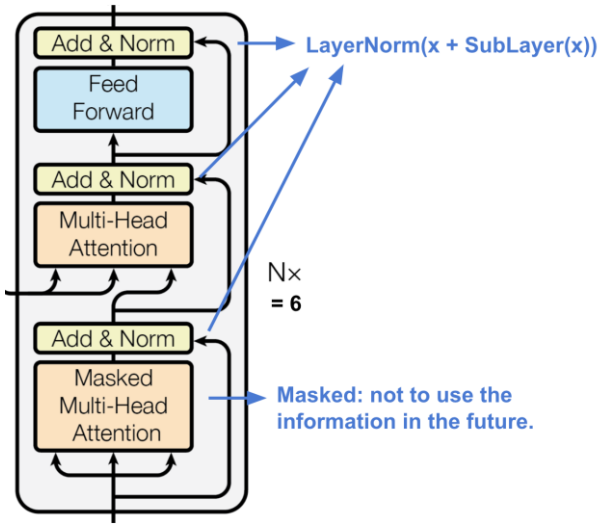
• A stack of N = 6 identical layers

• Each layer has **two sub-layers of multi-head attention** mechanisms and **one sub-layer of fully-connected feed-forward network.**

• Similar to the encoder, each sub-layer adopts a **residual connection and a layer normalization.**

• For the sequence generation, the first multi-head attention sub-layer is **modified** to prevent positions from attending to subsequent positions,

- to avoid looking into the future of the target sequence when predicting the current position.



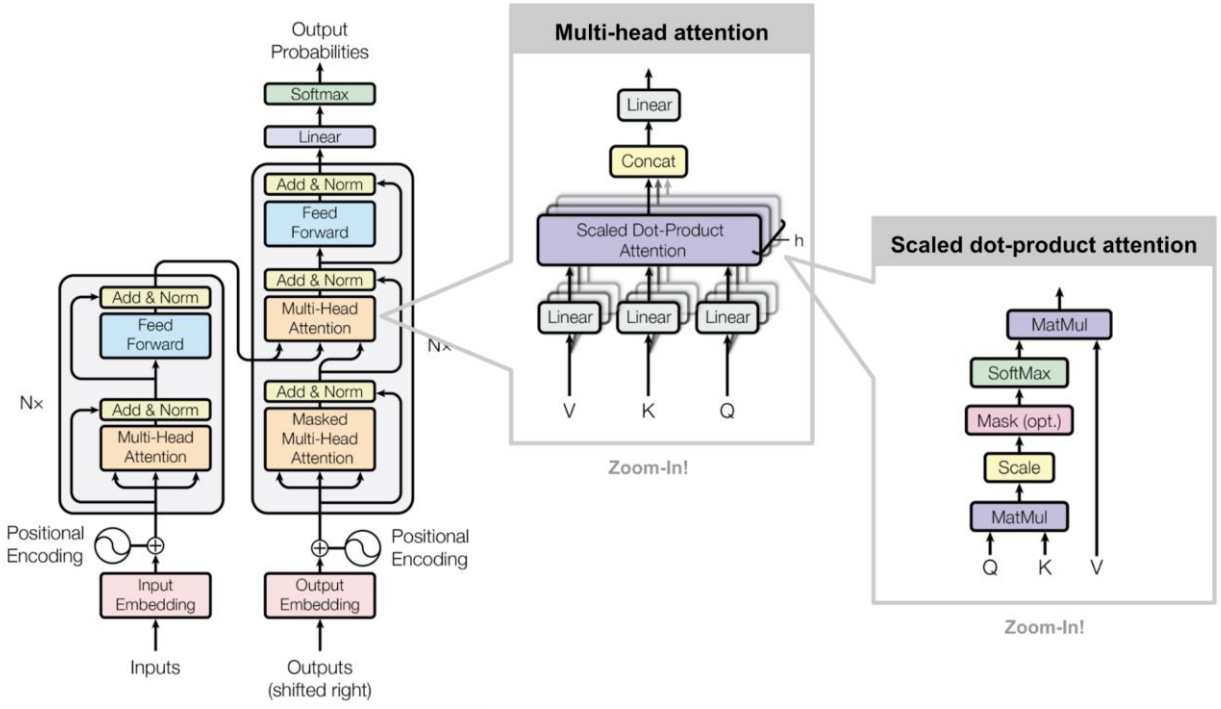
The transformer's decoder. (Image source: Vaswani, et al., 2017)

Full Architecture of Transformers

Complete view of the transformer architecture:

• Both the source and target sequences first go through embedding layers to produce data of the same dimension d_{model}=512.

• To preserve the position information, a sinusoid-wave-based positional encoding is applied and summed with the embedding output.



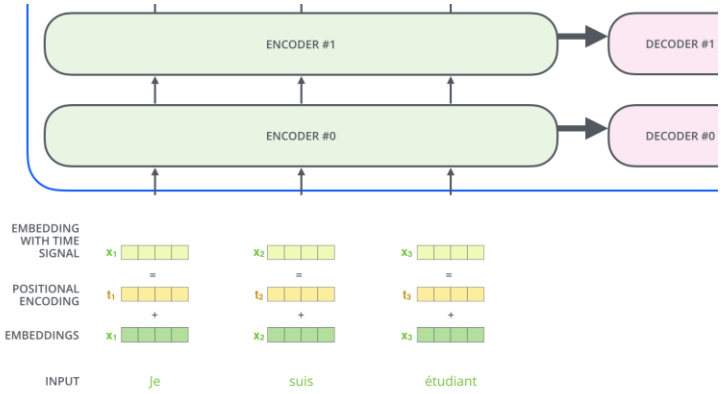
The full model architecture of the transformer.

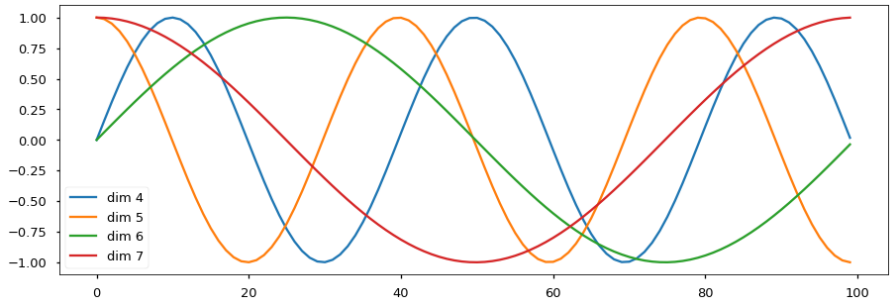
• A softmax and linear layer are added to the final decoder output.

Attention and Transformers

Representing the input order (positional encoding)

The transformer adds a vector to each input embedding. These vectors follow a specific pattern that the model learns, which helps it determine the position of each word, or the distance between different words in the sequence. The intuition here is that adding these values to the embeddings provides meaningful distances between the embedding vectors once they are projected into Q/K/V vectors and during dot-product attention.

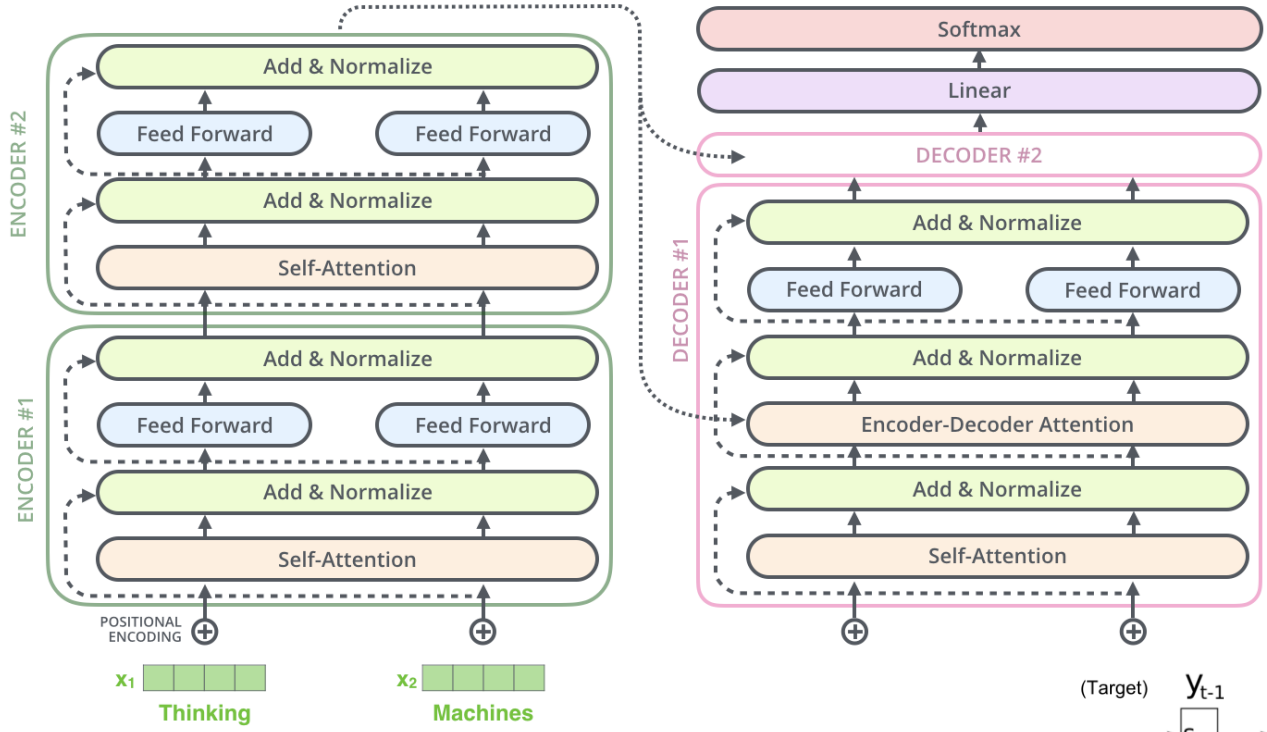




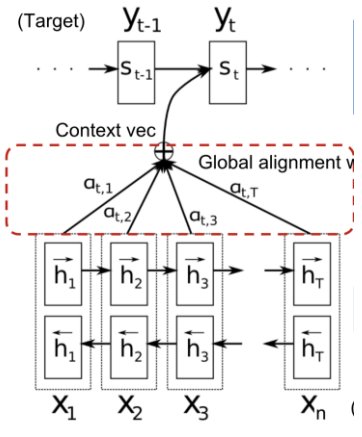
More on positional encoding: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Attention and Transformers

The encoder-decoder attention is just like self attention, except it uses K, V from the top of encoder output, and its own Q

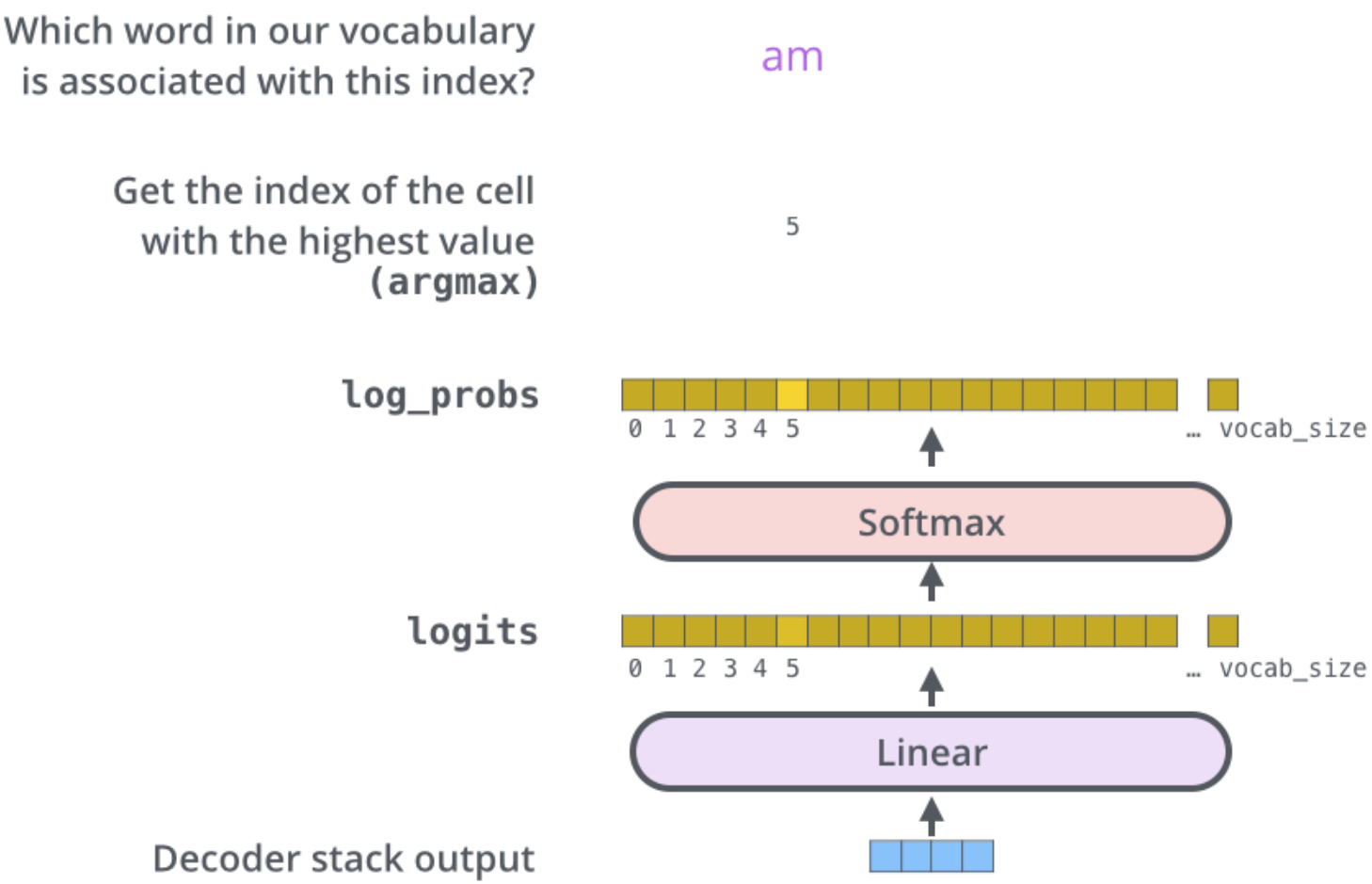


Compare Transformers to Seq2Seq:



Attention and Transformers

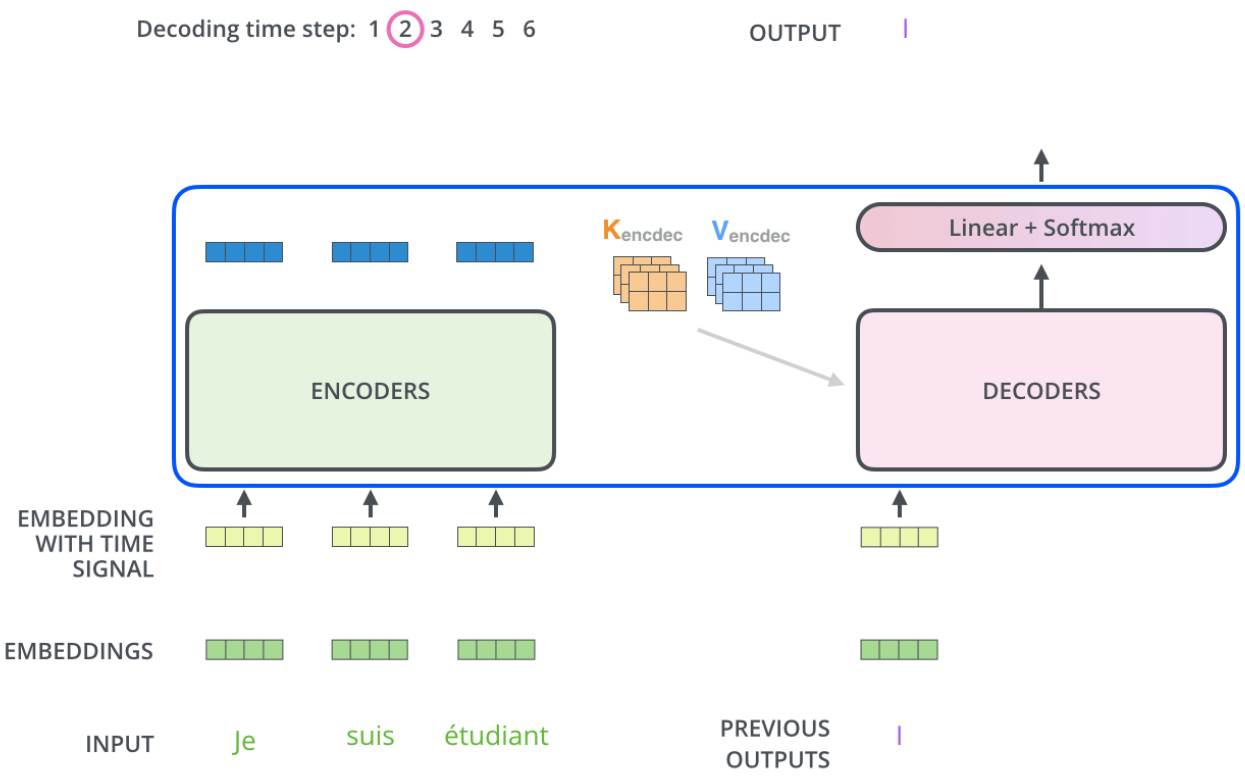
Decoder's Output Linear Layer



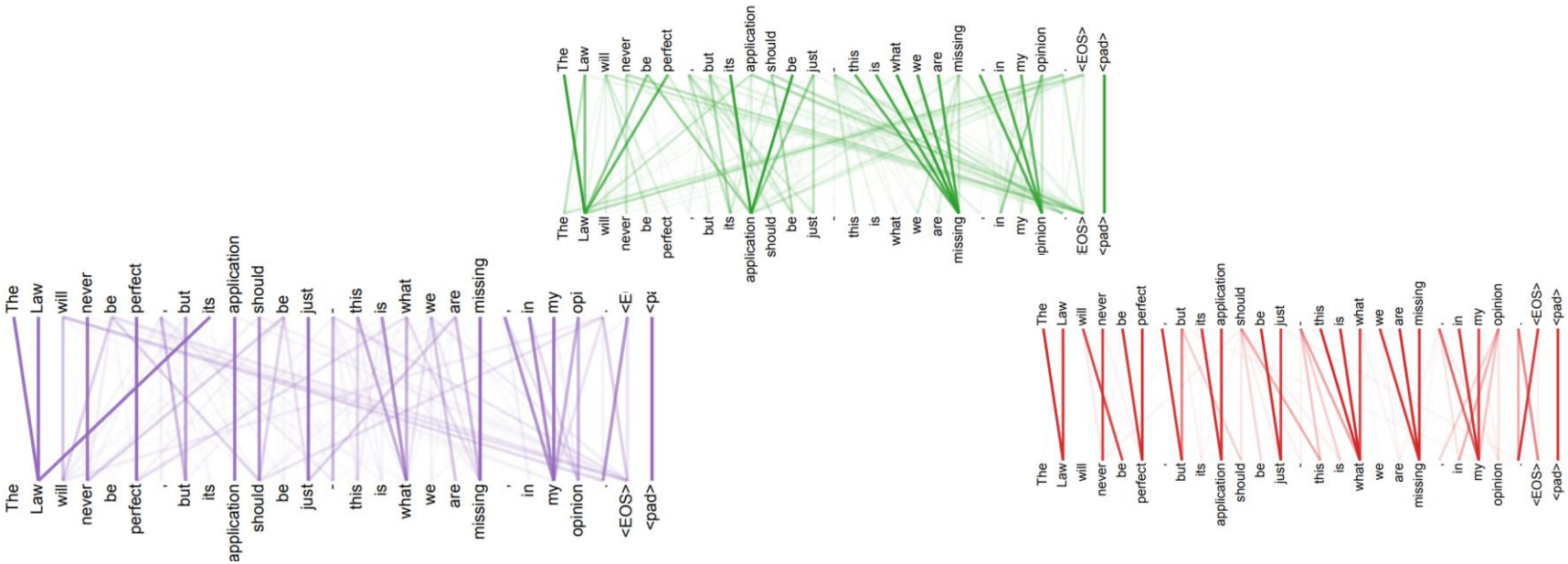
Attention and Transformers

How it works

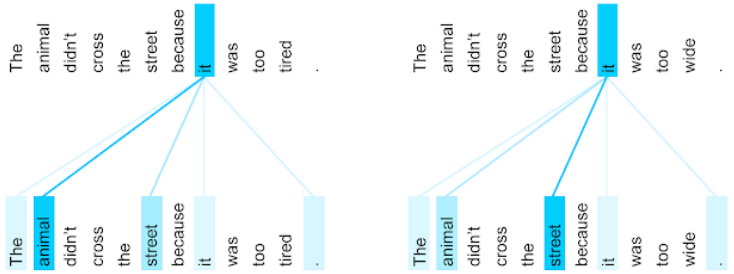
Decoder’s Output Linear Layer



Attention Visualization



The encoder self-attention distribution for the word “it” from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads).



https://arxiv.org/abs/1706.03762 https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

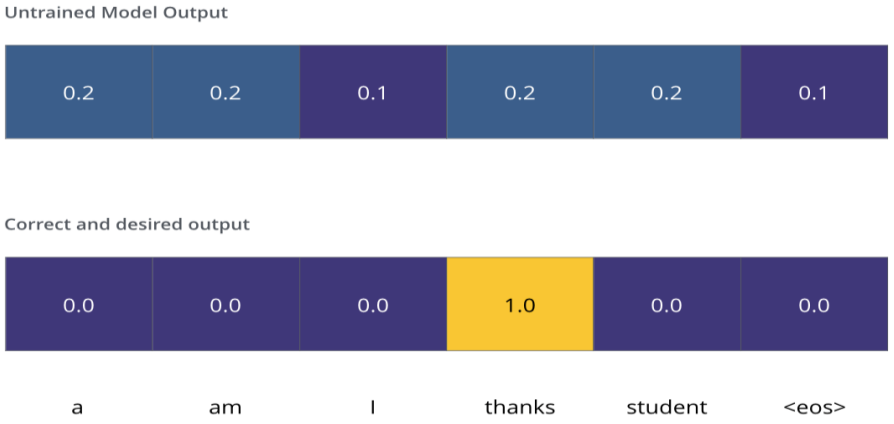
Attention and Transformers

Output Vocabulary

WORD	a	am	I	thanks	student	<eos>
INDEX	0	1	2	3	4	5

One-hot encoding of the word "am"

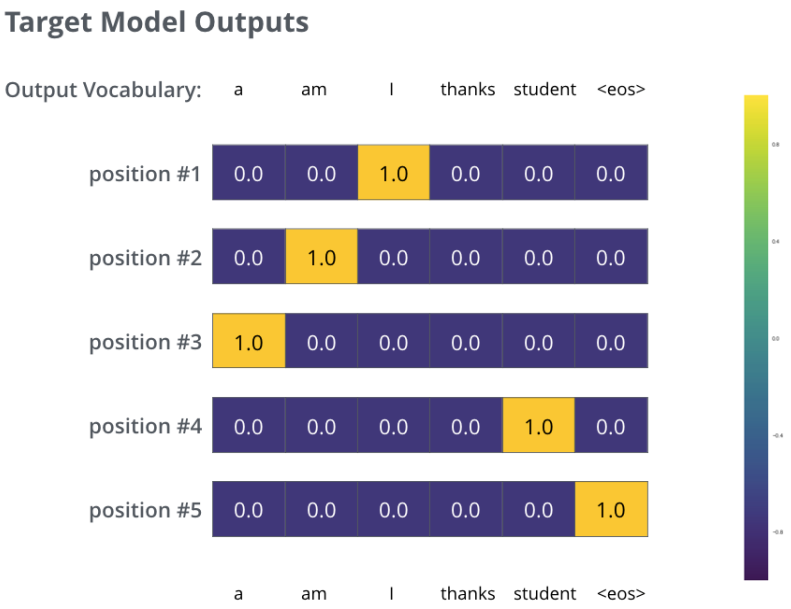
Training and the Loss Function

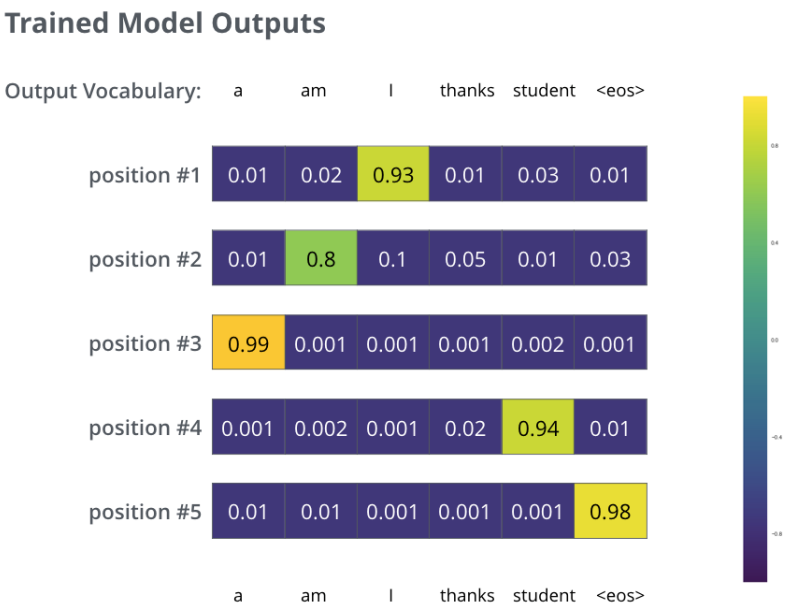


We can use cross Entropy.

We can also optimize two words at a time: using BEAM search: keep a few alternatives for the first word.

The Loss function





Goal: Compare both distributions coming from groundtruth and predictions

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

Attention and Transformers: Loss functions

Cross Entropy and KL (Kullback-Leibler) divergence

- Entropy: $E(P) = -\Sigma_i P(i) \log P(i)$ - expected prefix-free code length (also optimal)
- Cross Entropy: $C(P) = -\Sigma_i P(i) \log Q(i)$ – expected coding length using optimal code for Q
- KL divergence:

  $D_{KL}(P \,||\, Q) = \Sigma_i P(i) \log[P(i)/Q(i)] = \Sigma_i P(i)[\log P(i) - \log Q(i)]$, extra bits to code using Q rather than P

- JSD$(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M)$, $M = \frac{1}{2}(P+Q)$, symmetric KL
- JSD = Jensen-Shannon Divergence

Resources:

https://nlp.seas.harvard.edu/2018/04/03/attention.html (Excellent explanation of transformer model with codes.)

Jay Alammar, The illustrated transformer (from which I borrowed many pictures):

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

What about image analysis?

Preprint. Under review.

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy*,†, Lucas Beyer*, Alexander Kolesnikov*, Dirk Weissenborn*, Xiaohua Zhai*, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby*,†

*equal technical contribution, †equal advising

Google Research, Brain Team

{adosovitskiy, neilhoulsby}@google.com

ABSTRACT

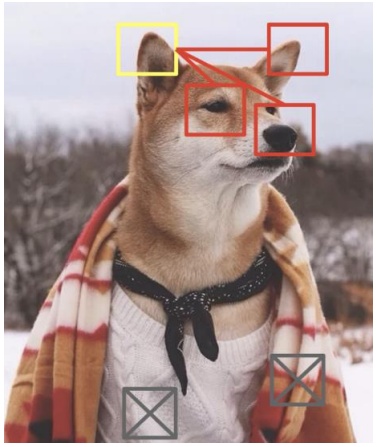
While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train[1].

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. 24851 cites

Attention

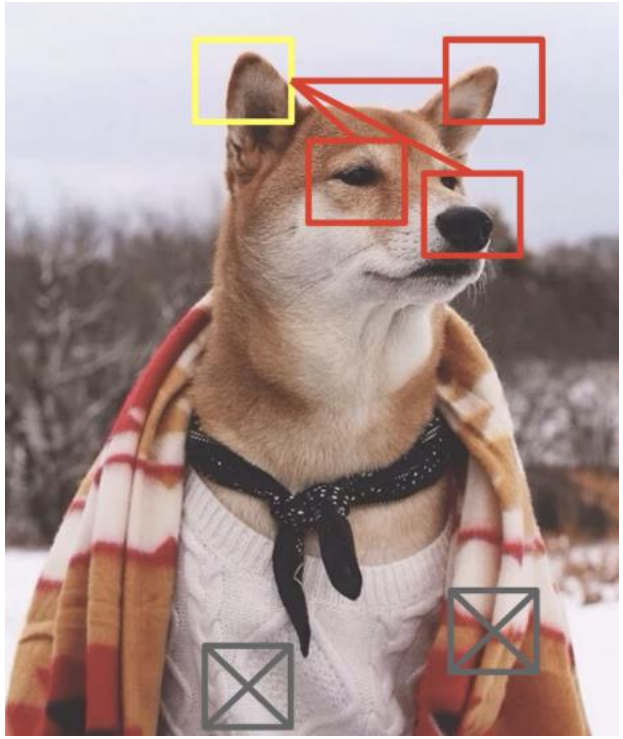
- Denote by $D=\{(k_1,v_1),\ldots(k_m,v_m)\}$ a database of m tuples of *keys* and *values*.
- Denote by q a *query*.
- Then we can define the *attention* over D as

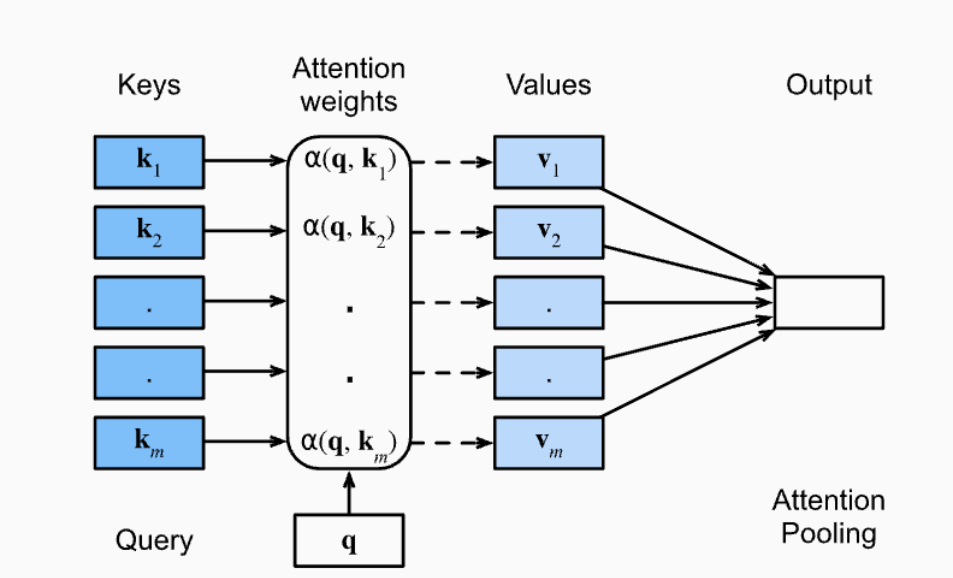
$$\mathrm{Attention}(\mathbf{q},\mathcal{D}) \stackrel{\mathrm{def}}{=} \sum_{i=1}^{m} \alpha(\mathbf{q},\mathbf{k}_i)\mathbf{v}_i,$$



- where $a(q,k_i)\in R$ $(i=1,\ldots,m)$ are **scalar attention weights**.
- The operation itself is typically referred to as ***attention pooling***.
- The name ***attention*** derives from the fact that the operation pays particular attention to the terms for which the **weight a is significant** (i.e., large).
- As such, the **attention over D generates a linear combination of values** contained in the database.

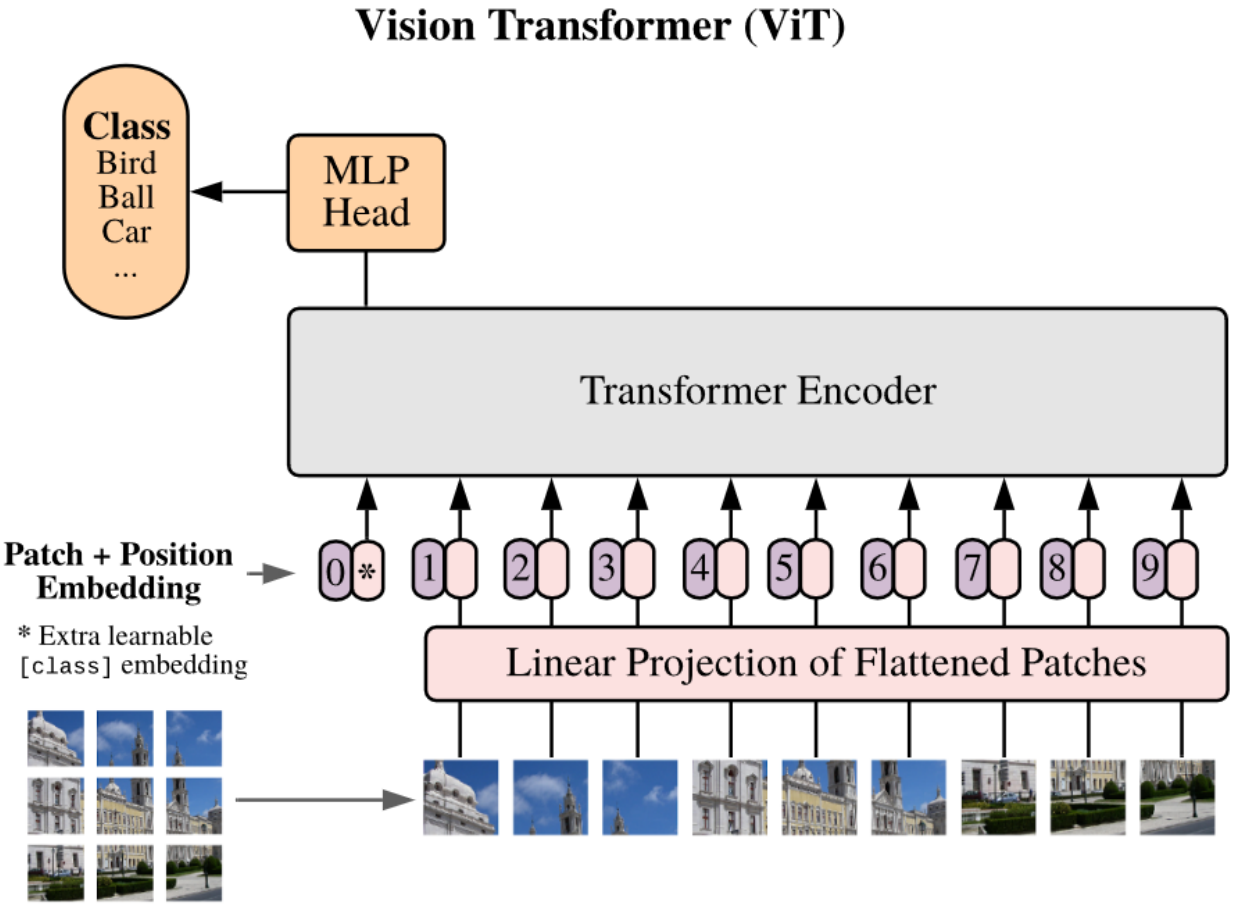
Attention pooling

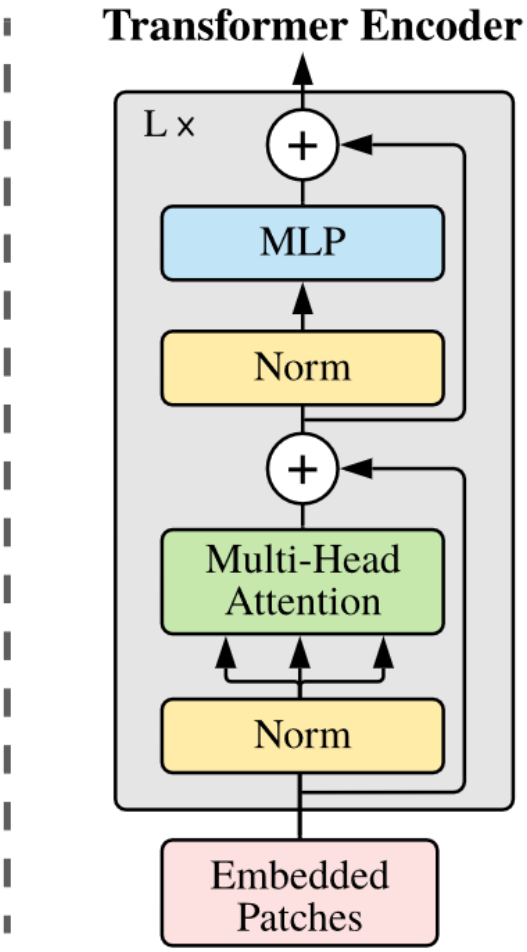




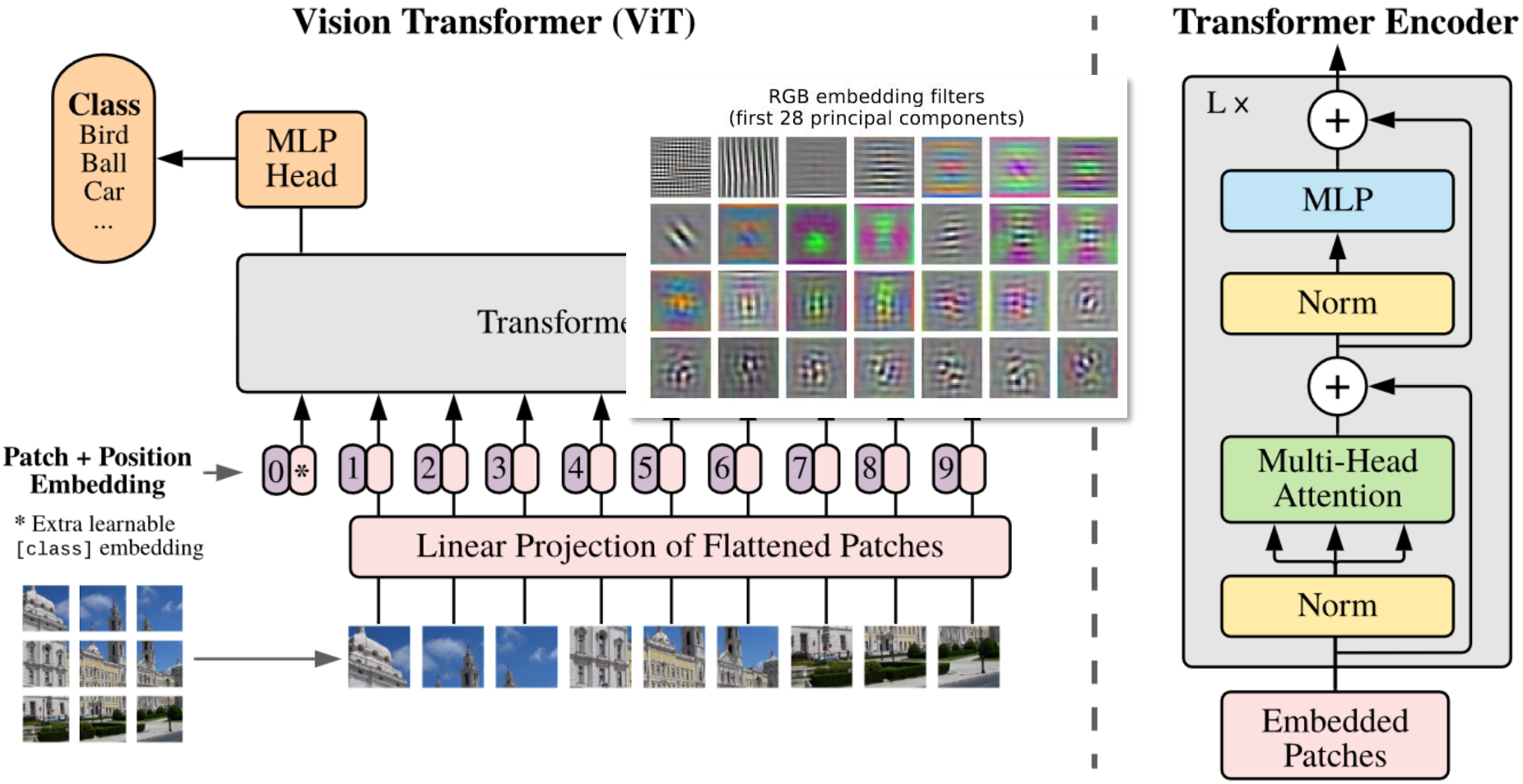
The attention mechanism computes a **linear combination over values $\mathbf{v}_i$ via attention pooling,** where weights are derived according to the compatibility between a query q and keys k_i.

Vision Transformer (ViT)

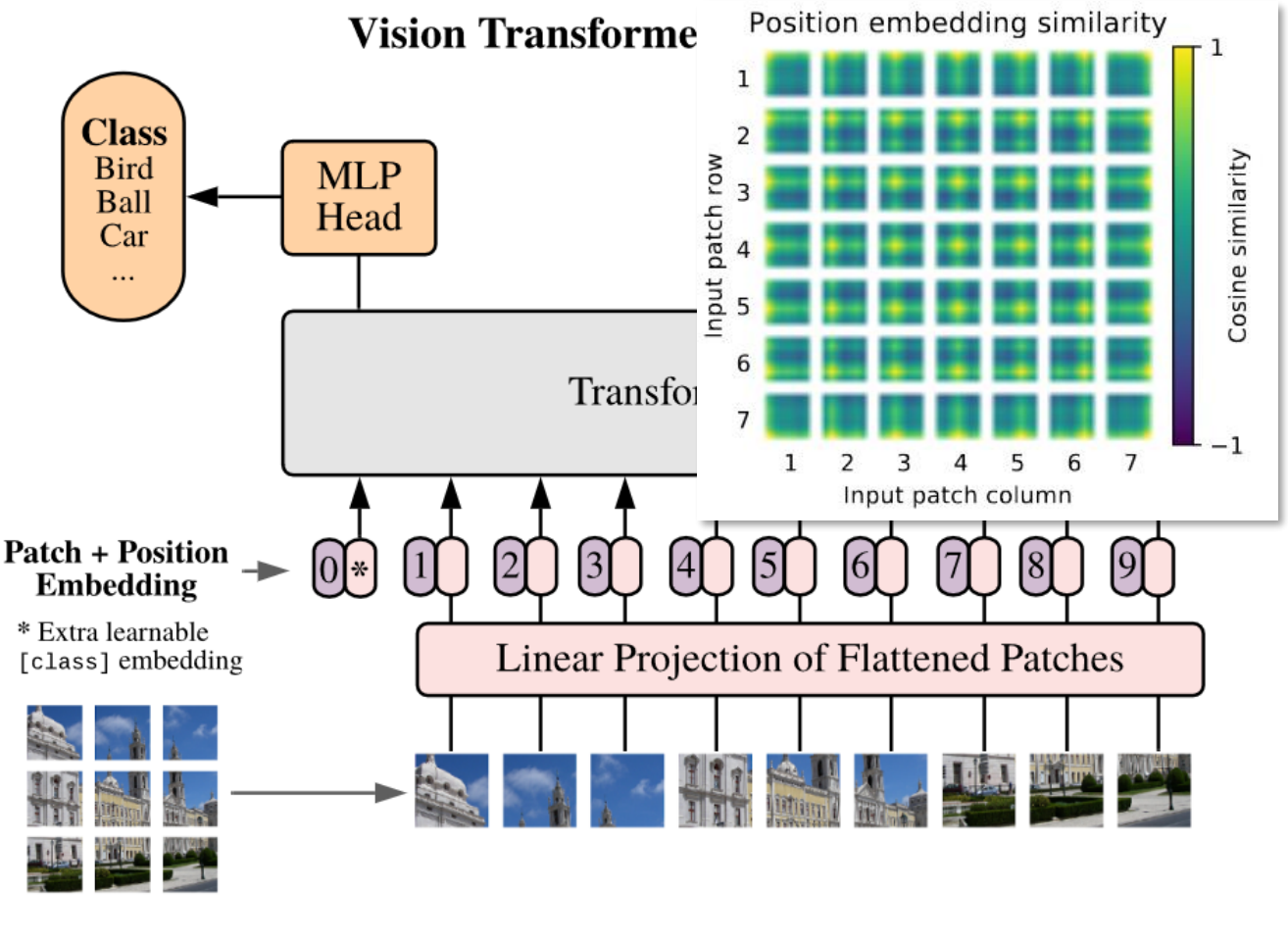


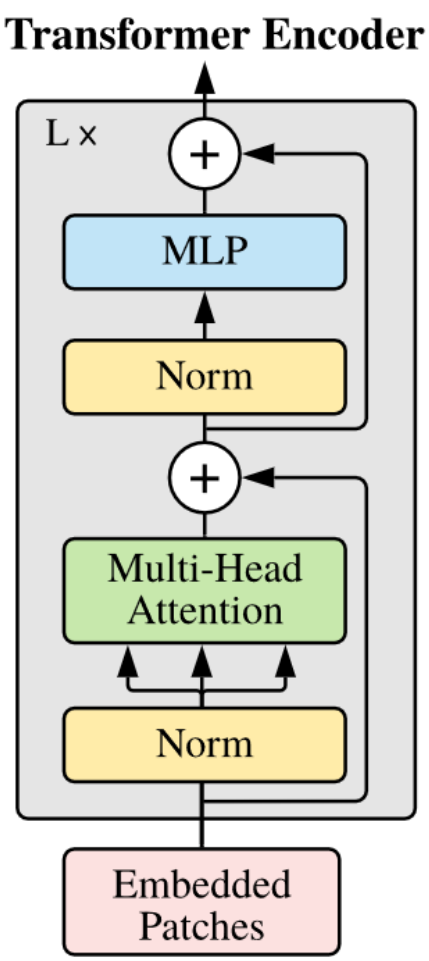


Vision Transformers



Vision Transformers





Results of ViT

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Details of Vision Transformer model variants.

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21K (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 0.06	99.42 $\pm$ 0.03	99.15 $\pm$ 0.03	99.37 $\pm$ 0.06	–
CIFAR-100	**94.55** $\pm$ 0.04	93.90 $\pm$ 0.05	93.25 $\pm$ 0.05	93.51 $\pm$ 0.08	–
Oxford-IIIT Pets	**97.56** $\pm$ 0.03	97.32 $\pm$ 0.11	94.67 $\pm$ 0.15	96.62 $\pm$ 0.23	–
Oxford Flowers-102	99.68 $\pm$ 0.02	**99.74** $\pm$ 0.00	99.61 $\pm$ 0.02	99.63 $\pm$ 0.03	–
VTAB (19 tasks)	**77.63** $\pm$ 0.23	76.28 $\pm$ 0.46	72.72 $\pm$ 0.21	76.29 $\pm$ 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

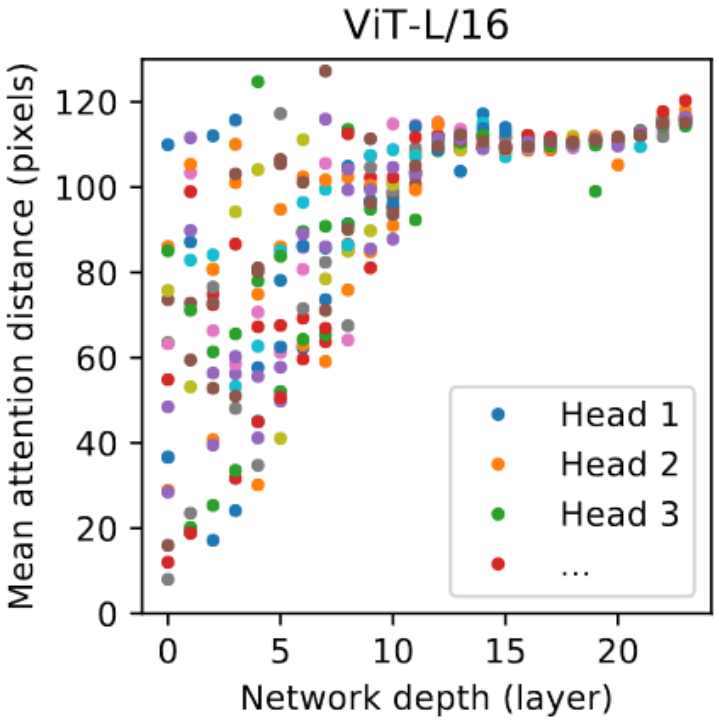
Observations

When trained on mid-sized datasets such as ImageNet, such models yield modest accuracies of a few percentage points below ResNets of comparable size.

- Transformers lack some of the inductive biases inherent to CNNs, such as translation equivariance and locality,
- Do not generalize well when trained on insufficient amounts of data.

However, the picture changes if the models are trained on larger datasets (14M-300M images).

- Large scale training trumps inductive bias.



https://paperswithcode.com/sota/image-classification-on-imagenet

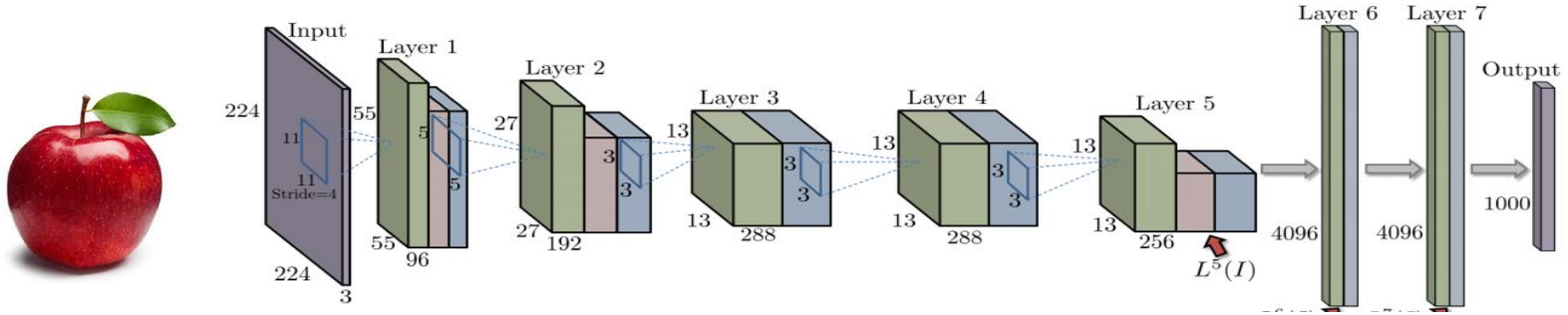
Summary

- “Attention” models outperform recurrent models and convolutional models for sequence processing. They allow long range interactions.
- These models do best with LOTS of training data
- They seem like a good fit for point processing tasks, although maybe there isn’t enough data for them to outperform other methods.
- Surprisingly, they seem to outperform convolutional networks for image processing tasks.
- Naïve attention mechanisms have quadratic complexity with the number of input tokens, but there are often workarounds for this.

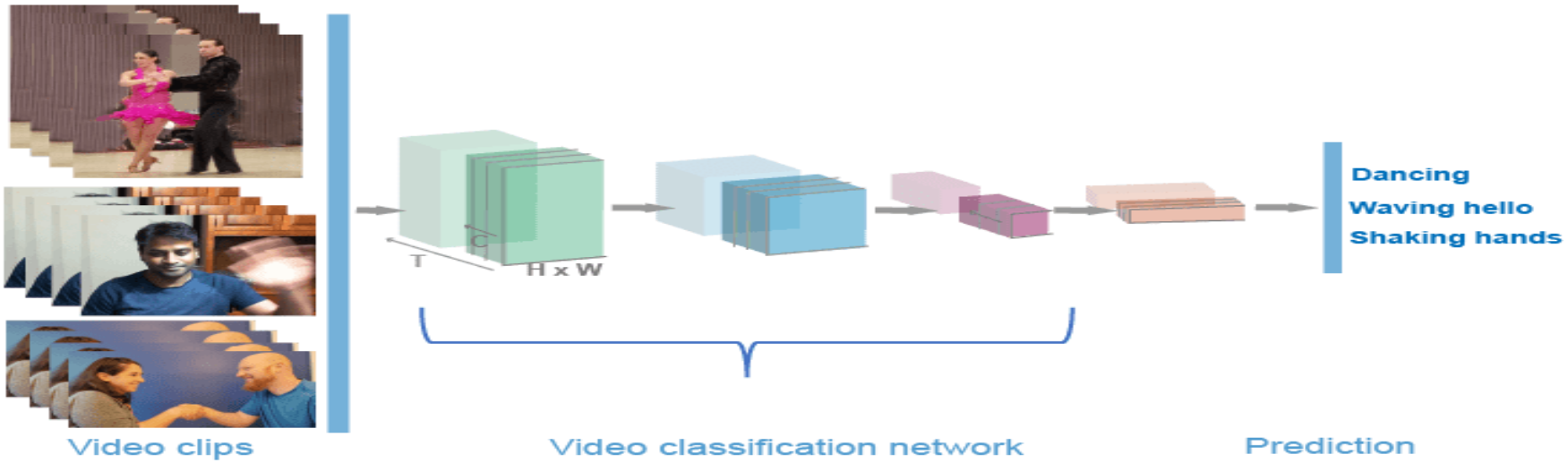
Summary on attention mechanism

- The attention mechanism allows us to aggregate data from many (key, value) pairs.
  - For instance, in a regression setting, the query might correspond to the location where the regression should be carried out.
  - The keys are the locations where past data was observed and the values are the (regression) values themselves.
- By design, the attention mechanism provides a ***differentiable* means** of control by which a neural network can select elements from a set and to construct an associated weighted sum over representations.
- "Attention" models **outperform recurrent models and convolutional models** for sequence processing. They allow long range interactions.
- These models do best **with LOTS of training data**
- They seem like a **good fit for point processing tasks**, although maybe there isn't enough data for them to outperform other methods.

What to do when we have a video?



Apple
Frog
Tiger
Lion
Heart
.
.
.



Questions?