Transformers

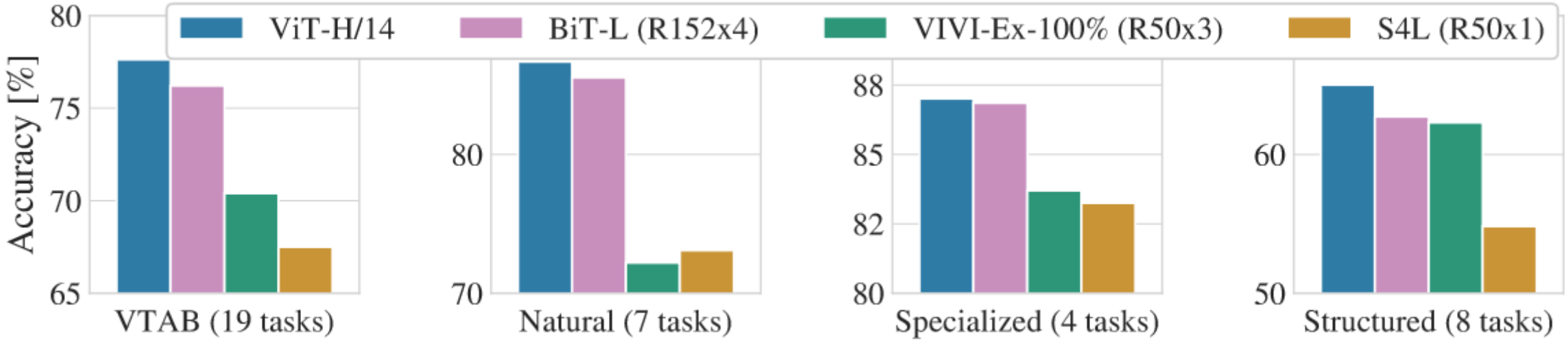
Computer Vision – Lecture 08

Further Reading

- Slides from F Li
- Slides from J Johnson
- Slides from M Niessner
- ~~None of the books I know contains transformers.~~ Foundations of Computer Vision covers transformers

CNNs to Transformers

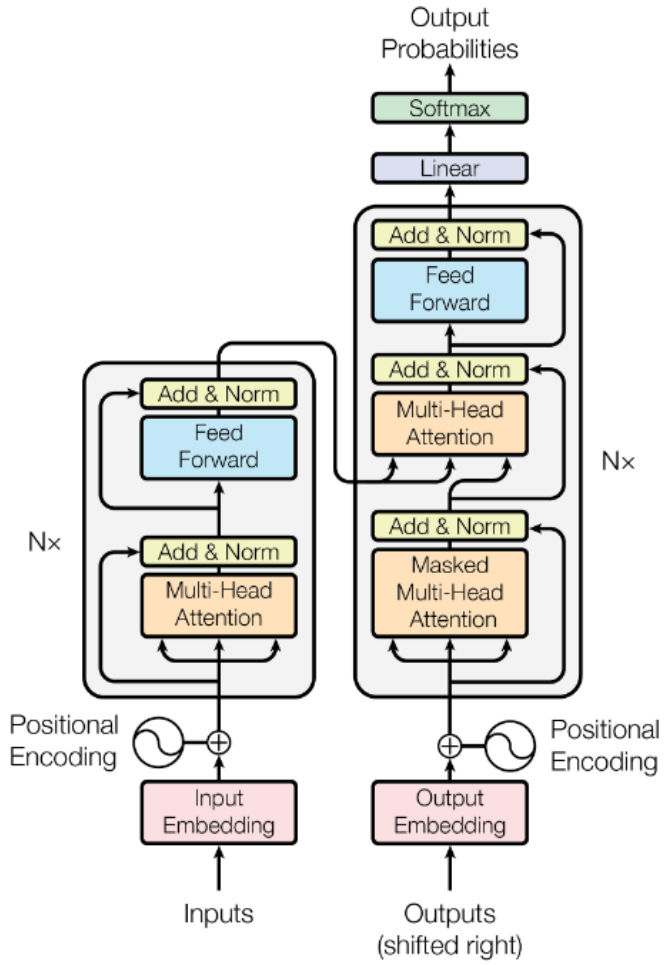
An image is worth 16x16 words: Transformers for image recognition at scale, A Dosovitskiy et al., 2021



Attention

Attention Is All You Need, A Vaswani et al., 2017 (100k citations now)

- Before: main building block is a convolution.
- After: main building block is *multi-head attention*.



1D Attention

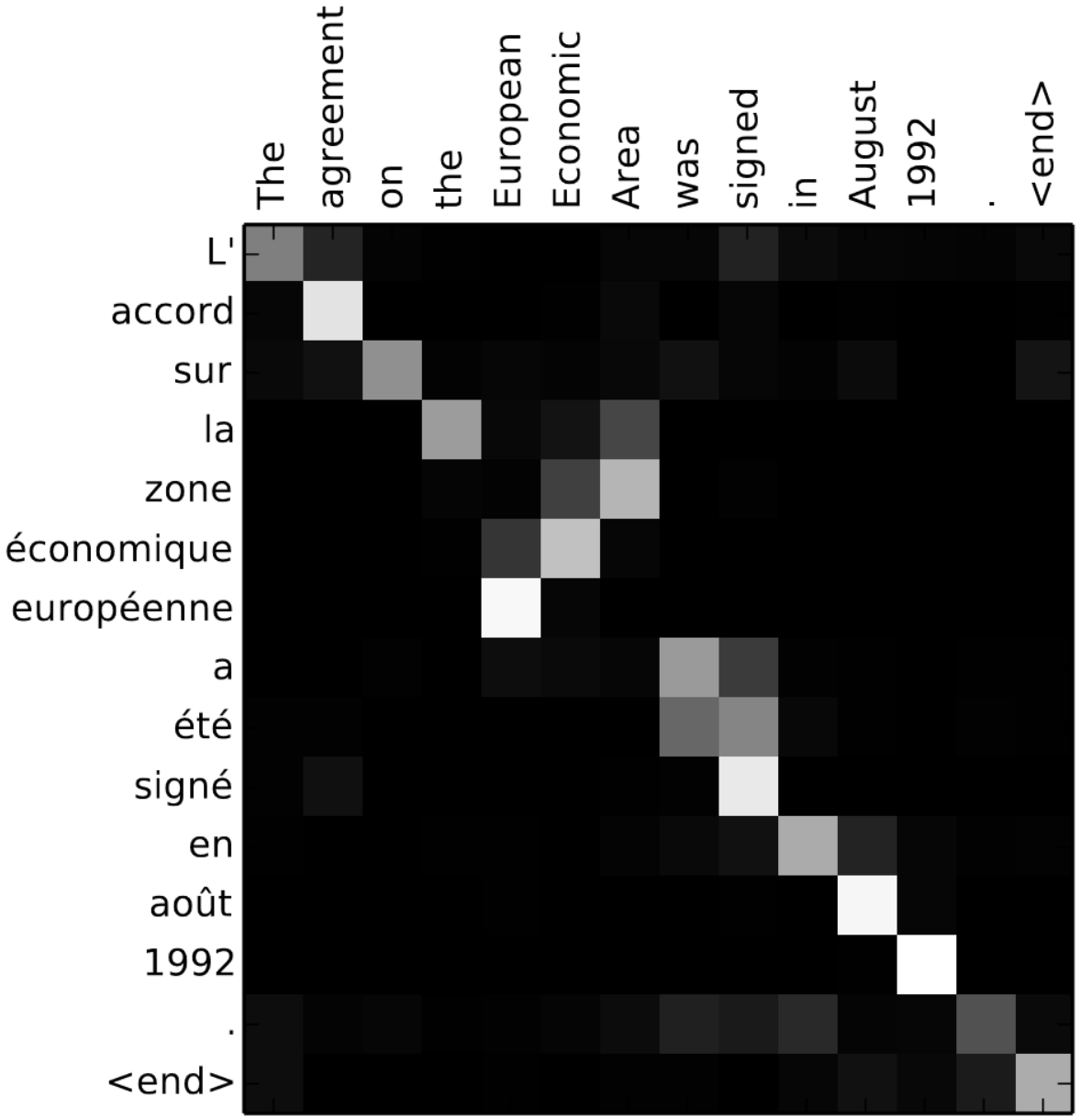
- Attention was first developed for natural language processing.
- Used to process a sequence of *tokens* (e.g. words).
- Each token is embedded into a vector space.
- Idea: long-range context is important.

1D Attention

Example machine translation:

- Languages tend to vary in word order.

- The model cannot translate word-by-word but must be able to look at all words.



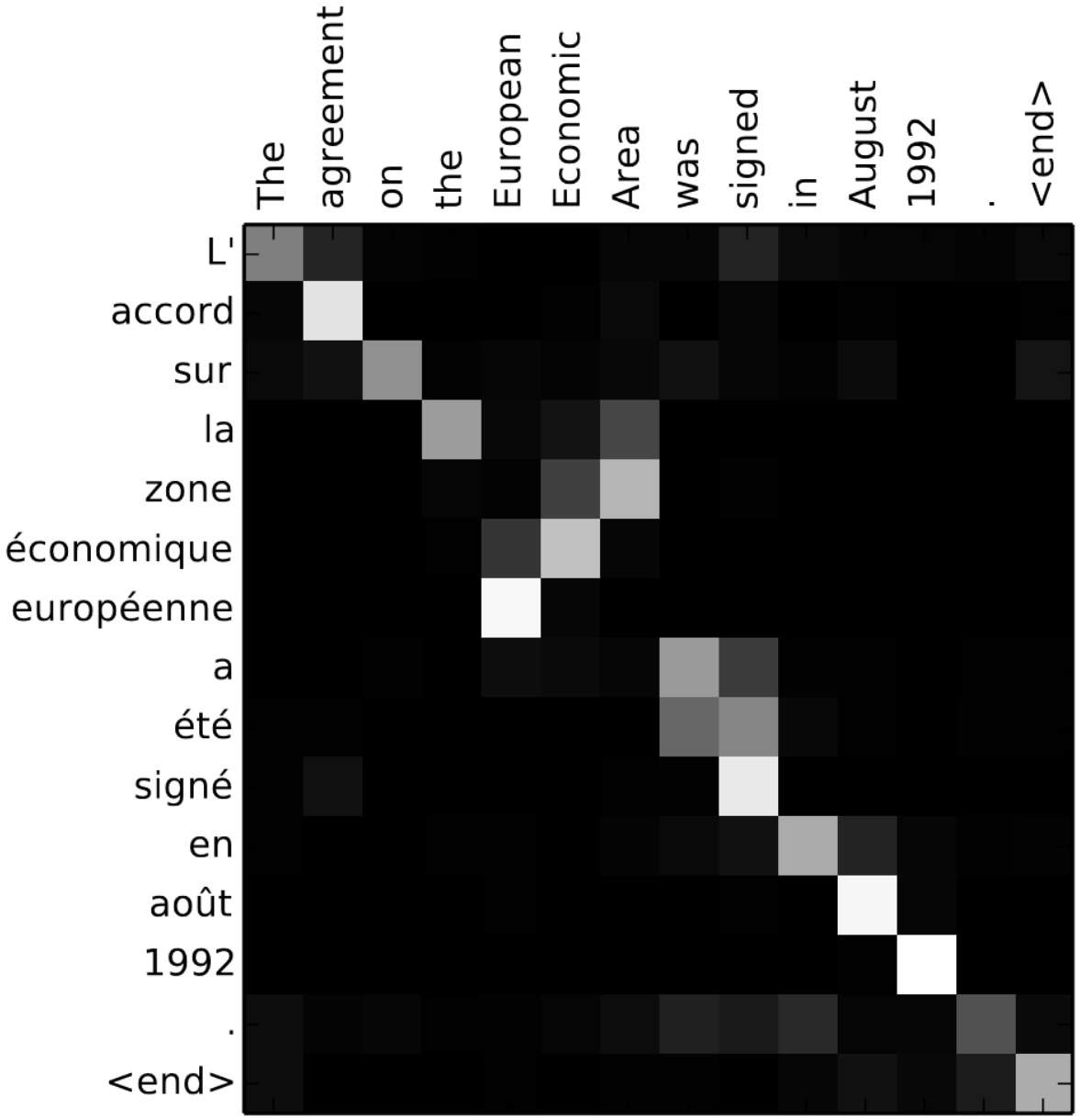
Bahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015

Slide adapted from J Johnson

1D Attention

Example machine translation:

- English: “The agreement on the European Economic Area was signed in August 1992.”
- French: “L’accord sur la zone économique européenne a été signé en août 1992. ”



Bahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015

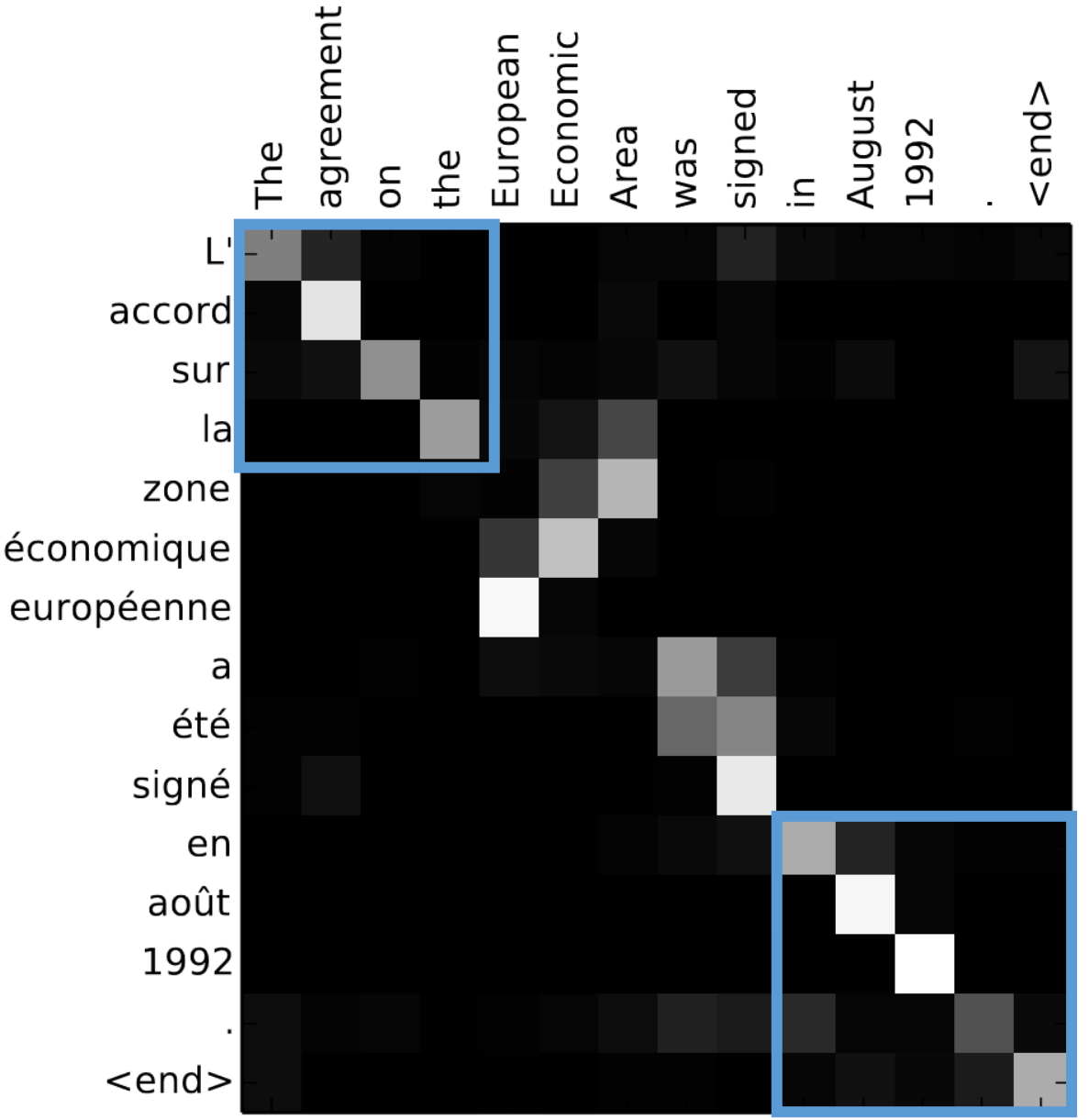
Slide adapted from J Johnson

1-to-1 matching in word order

1D Attention

Example machine translation:

- English: “The agreement on the European Economic Area was signed in August 1992.”
- French: “L’accord sur la zone économique européenne a été signé en août 1992. ”



Bahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015

Slide adapted from J Johnson

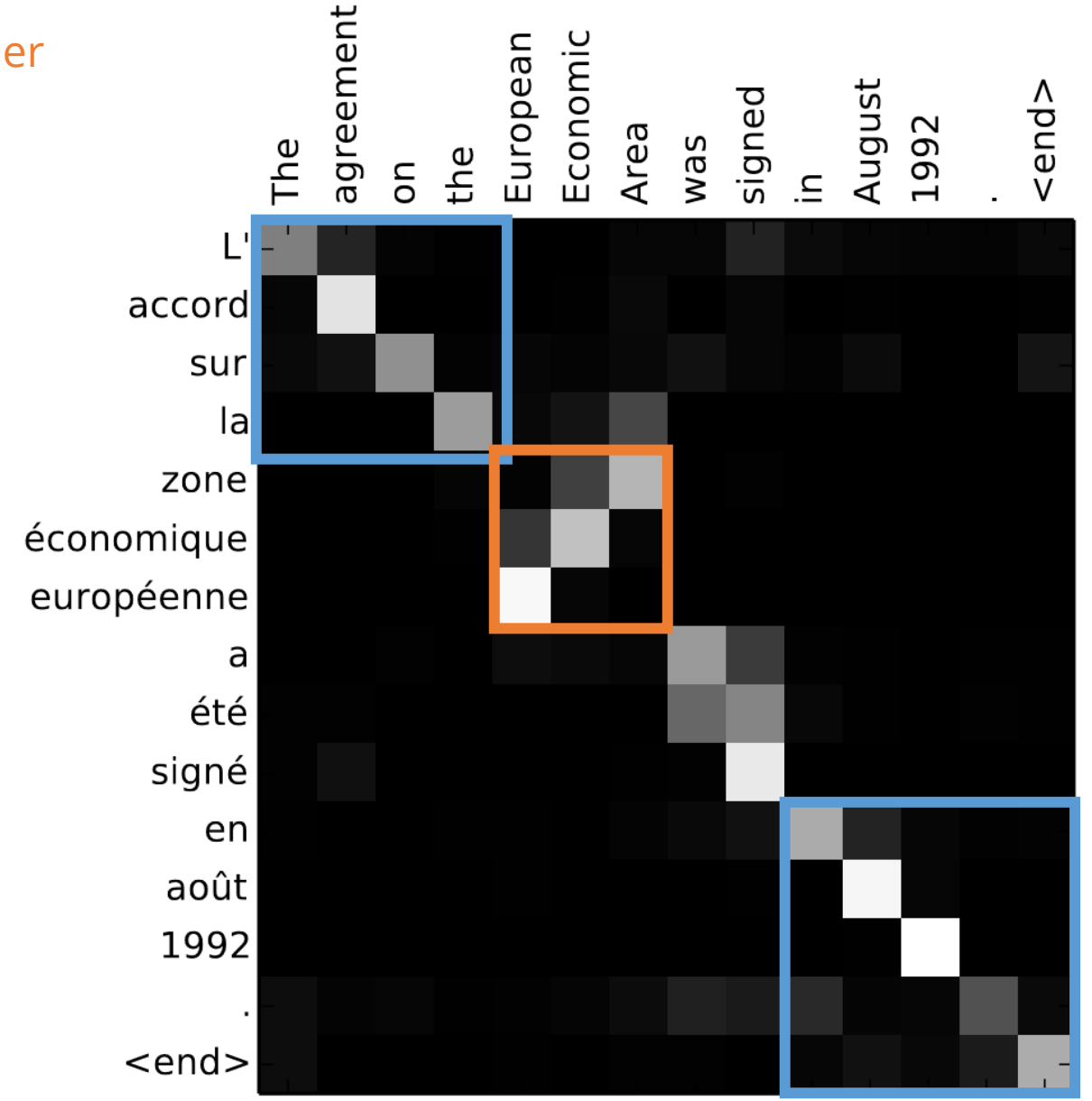
1D Attention

1-to-1 matching in word order

reverse word order

Example machine translation:

- English: “The agreement on the European Economic Area was signed in August 1992.”
- French: “L’accord sur la zone économique européenne a été signé en août 1992. ”



Bahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015

Slide adapted from J Johnson

1D Attention

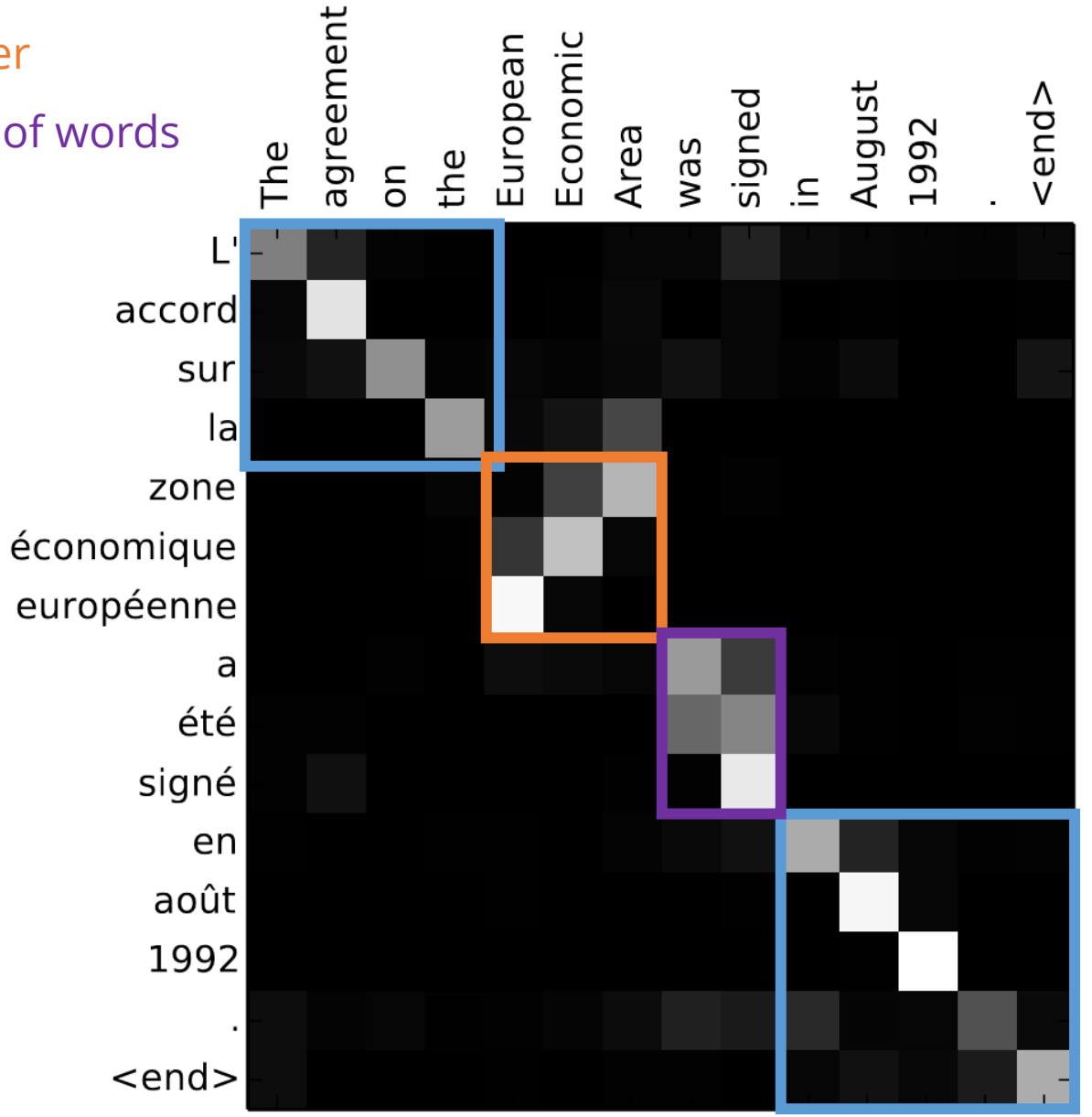
1-to-1 matching in word order

reverse word order

different number of words

Example machine translation:

- English: “The agreement on the European Economic Area was signed in August 1992.”
- French: “L’accord sur la zone économique européenne a été signé en août 1992. ”



Attention

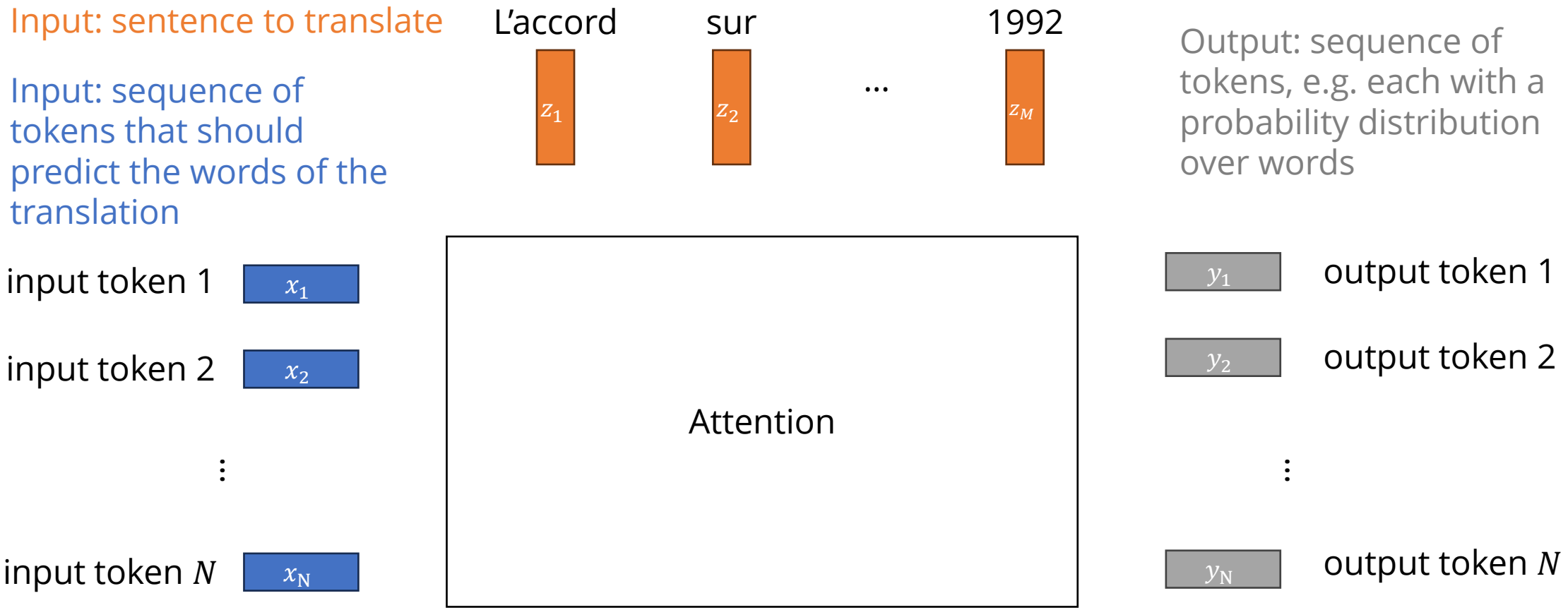
Bahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015

Slide adapted from J Johnson

Setup for attention

Input: sentence to translate

Input: sequence of tokens that should predict the words of the translation



Output: sequence of tokens, e.g. each with a probability distribution over words

Input/Output Behaviour

- N input tokens $x_i \in \mathbb{R}^d$
- M "condition" tokens $z_j \in \mathbb{R}^d$
- Predict: N output tokens $y_i \in \mathbb{R}^d$

- We assume all vectors are in $\mathbb{R}^d$, if not a linear layer can fix it.
- Similar to convolution, many of these layers can be stacked.

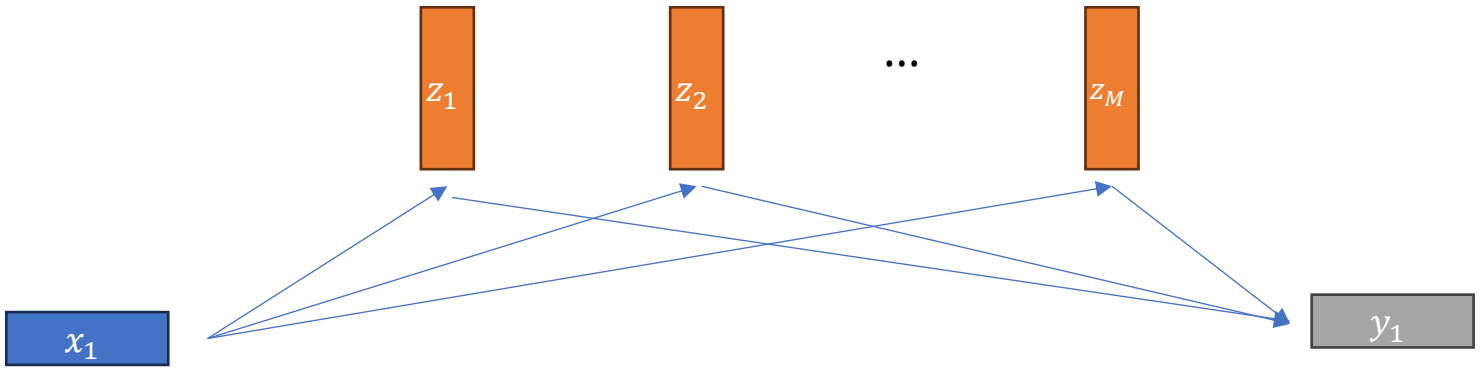
- Example translation: we can add a final linear layer that maps each output token to word probabilities (use softmax)

Attention

- Each input token is processed in the same way:

  $y_i = Attn(x_i, Z)$ – analyse only one for now.

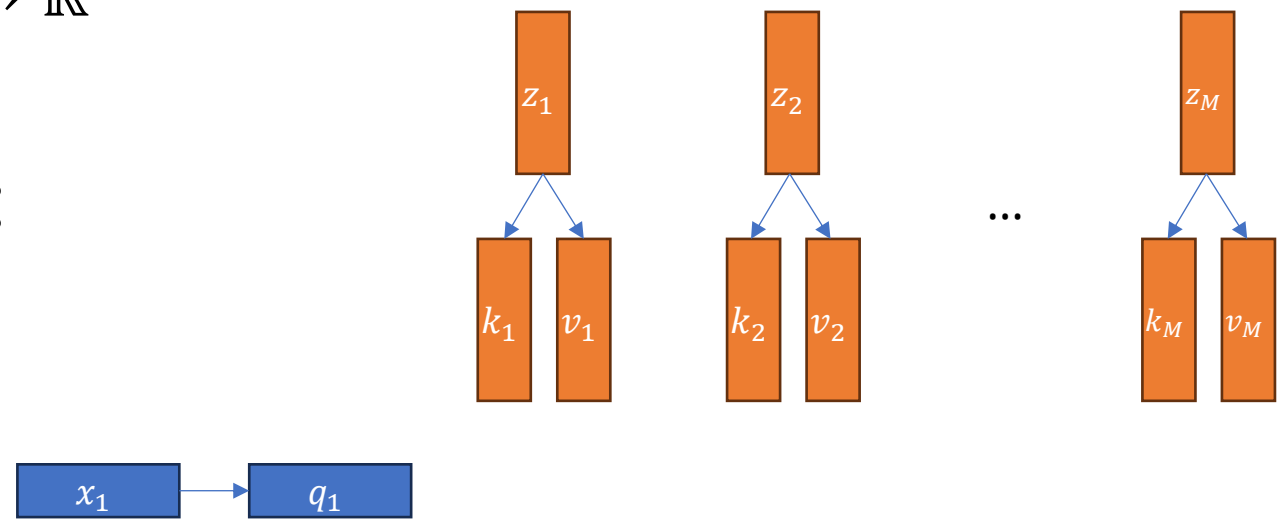
- Intuition: x_i looks at all z_j and decide what is relevant.



Attention

- We do this by introducing three functions:
  - A *query* function $Q: \mathbb{R}^d \to \mathbb{R}^d$
  - A *key* function $\mathrm{K}: \mathbb{R}^d \to \mathbb{R}^d$
  - A *value* function $\mathrm{V}: \mathbb{R}^d \to \mathbb{R}^d$

- They are used like this:
  - $q_i = Q(x_i)$
  - $k_j = K(z_i)$
  - $v_j = V(z_i)$

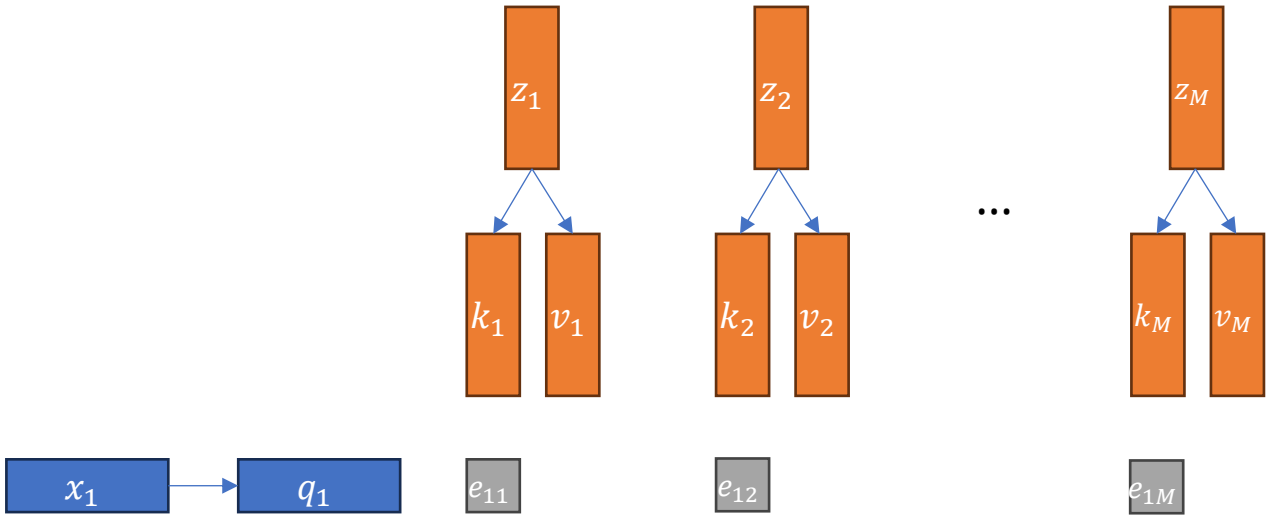


- Naming: we will use queries to find keys and retrieve values.

Attention

- Compute similarities between keys and values using a dot product.
- Intuition: large if key is similar to query.

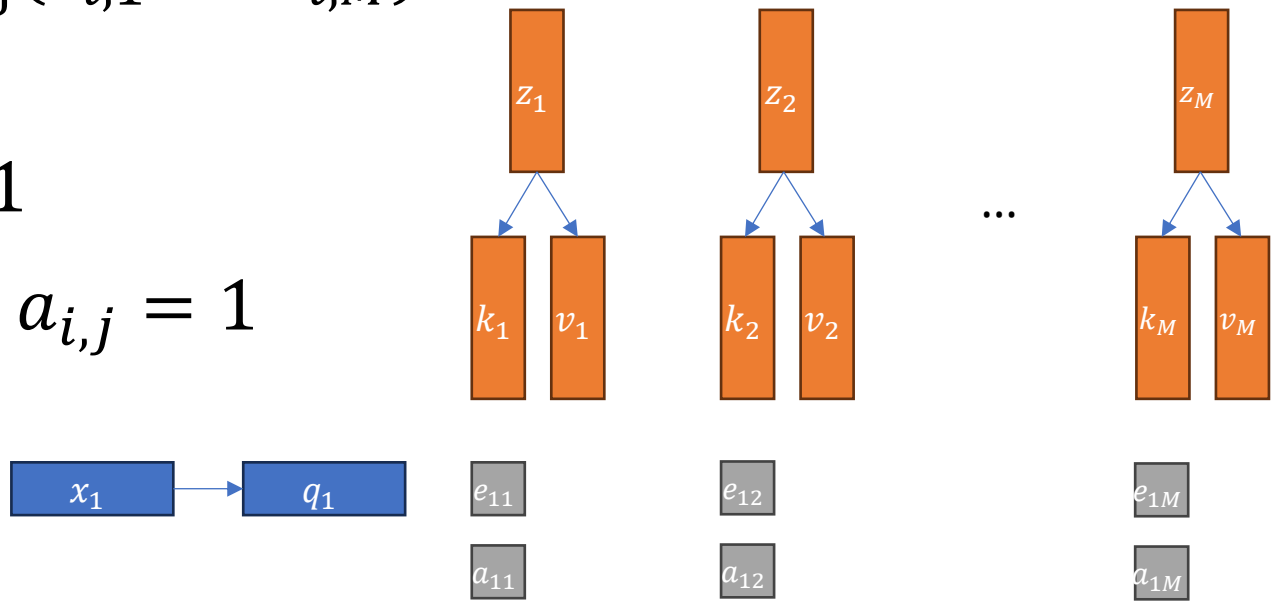
$$e_{i,j} = \frac{1}{\sqrt{d}} q_i^T k_j \in \mathbb{R}$$



Attention

- Intuition: interpret $e_{i,j}$ as weights to combine their values v_j.
- Problem: $e_{i,j} \in \mathbb{R}$, unbounded
- Solution: $a_{i,j} = \mathrm{softmax_j}(e_{i,1}, \ldots, e_{i,M})$

- Normalises: $0 \leq a_{i,j} \leq 1$
- Weights sum to 1: $\sum_{j=1}^{M} a_{i,j} = 1$

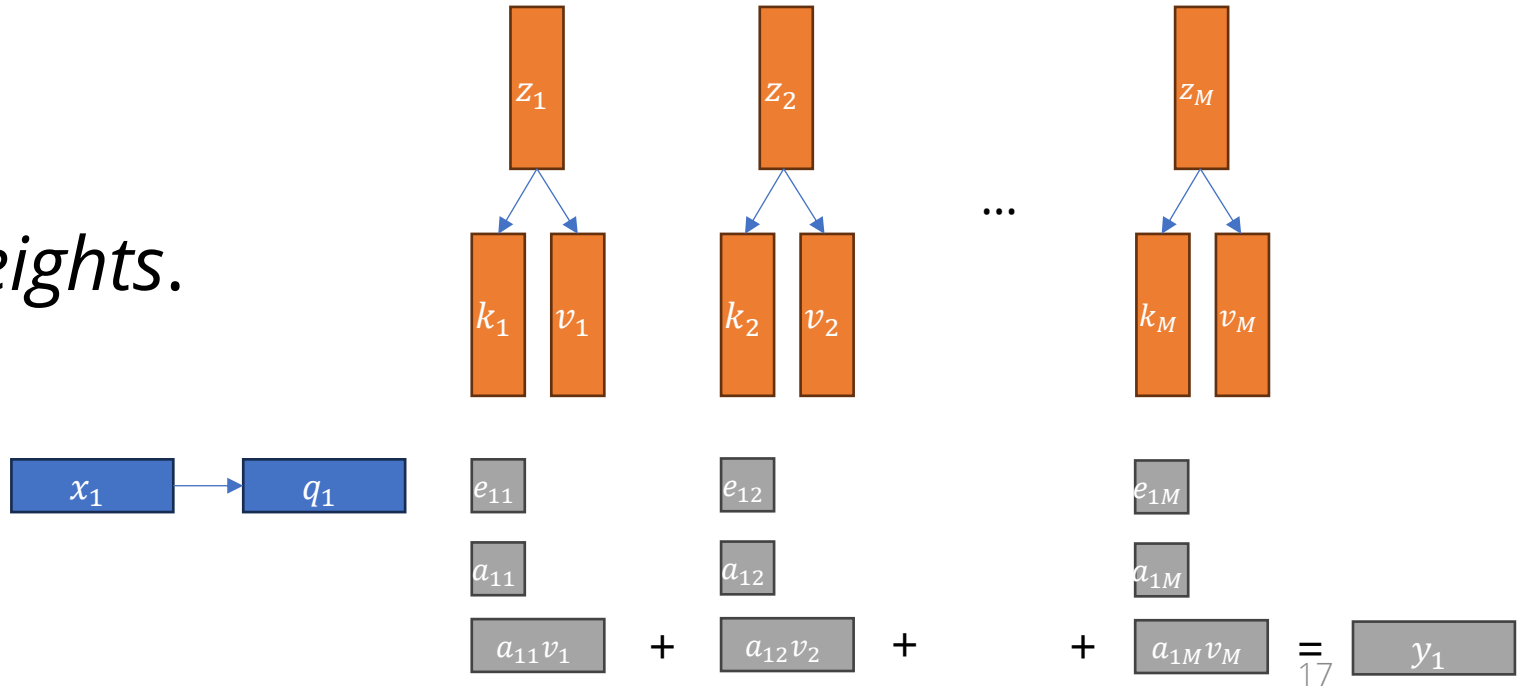


Attention

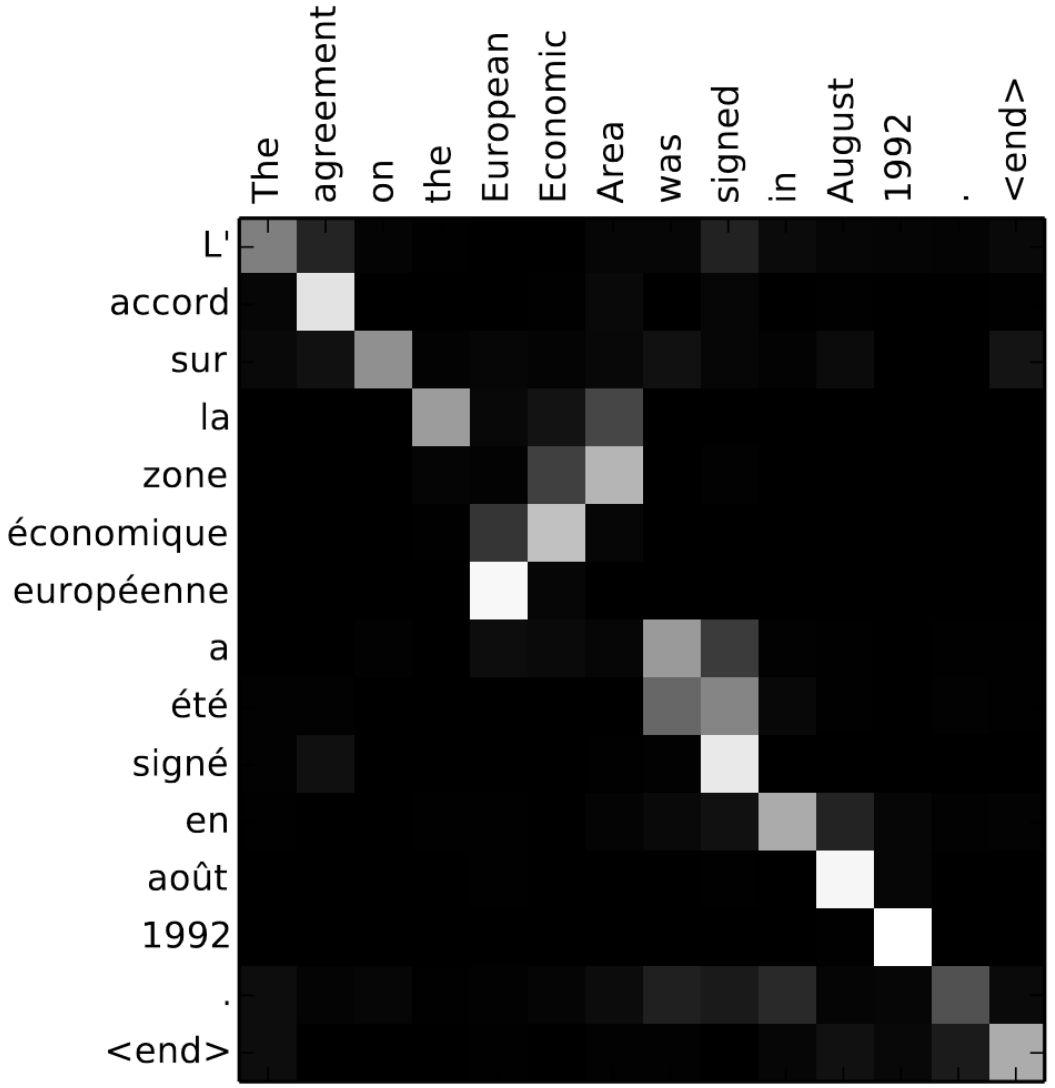
We can now compute the output as the weighted sum of values:

$$y_i = \sum_{j=1}^{M} a_{i,j} v_j$$

- $a_{i,j}$ are the *attention weights*.



Attention Weights



Bahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015

Attention (to) Details

- The three functions for query, key, and value are linear functions, of appropriately sized matrices.
  - *Query* function $Q(x) = \boldsymbol{Q}x$
  - *Key* function $\mathrm{K}(z) = \boldsymbol{K}z$
  - *Value* function $\mathrm{V}(z) = \boldsymbol{V}z$
- All operations (except softmax) can be implemented by matrix-vector (or tensor) products: fast.

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

- Attention is permutation-equivariant: changing the order of inputs x_i will simply change the order of outputs y_i.

Attention (to) Details

- Permutation equivariance is sometimes unwanted: e.g. word-order matters in NLP.

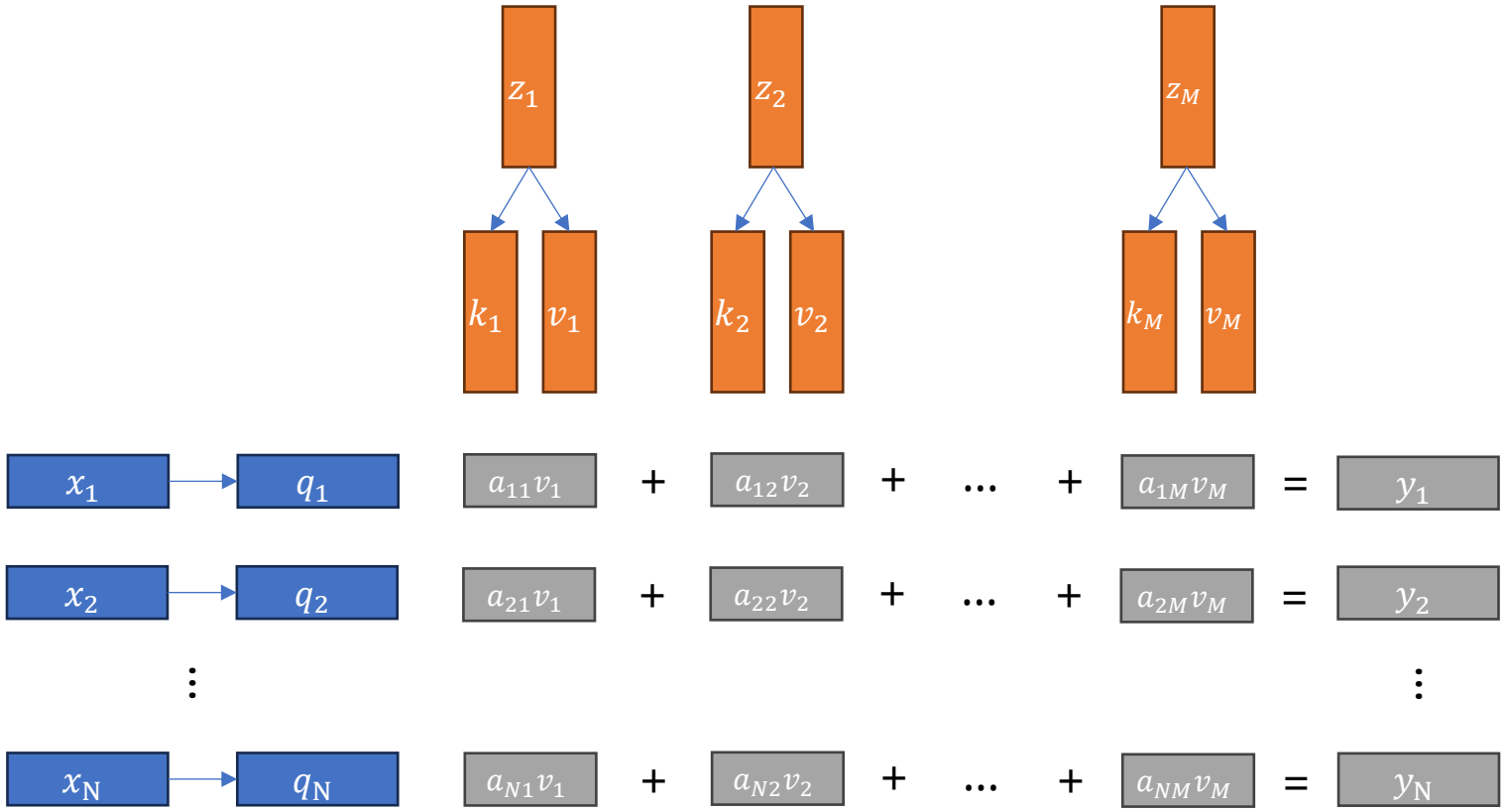
- Idea: add a signal to the inputs, that identifies the order:

$$x_i' = x_i + P(i)$$

- $P: \mathbb{N} \to \mathbb{R}^d$ can be simply learned by the network, or handcrafted:
  e.g. combining sin and cos frequencies sampled at i.

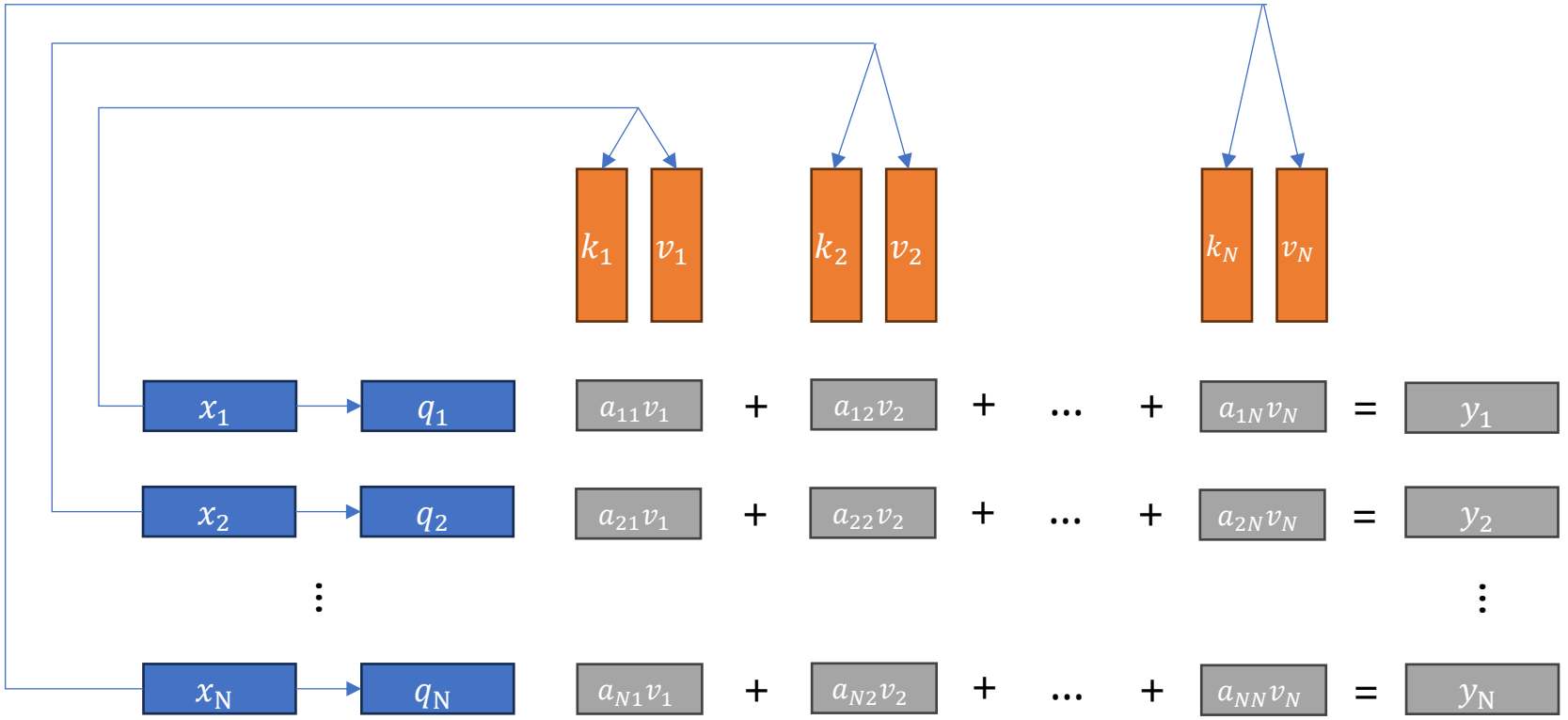
Attention

What if we do not have a conditional task?



Self-Attention

We simply predict the keys and values also from the input.



(Self-)Attention

- Without positional encoding: permutation invariant.
- Complexity: $\mathcal{O}(N^2)$ in the number of input tokens.
- What to pay attention to is learned automatically.
- Attention is thus somewhat interpretable (Lecture 09).

Multi-Head (Self-)Attention

Similar to convolutions, by applying several different filters per layer, we can use multiple SA *heads* simultaneously.

$$y^{(h)} = \mathrm{SAttn}\left(\boldsymbol{K}^{(h)}x, \boldsymbol{V}^{(h)}x, \boldsymbol{Q}^{(h)}x\right), \qquad 1 \leq h \leq H$$

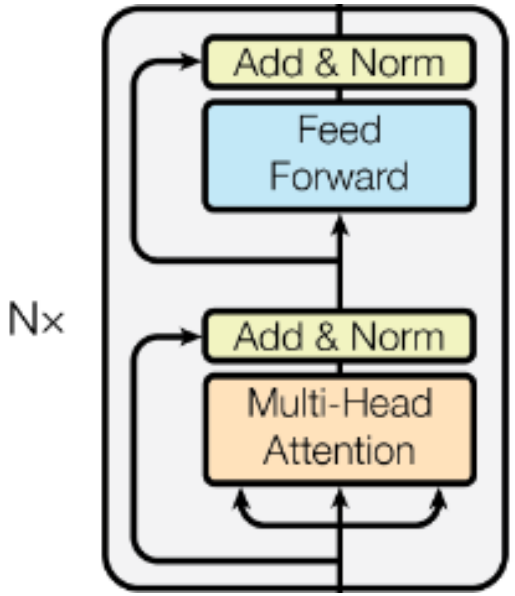
- Each attention head has its own weights $\boldsymbol{K}^{(h)}$, $\boldsymbol{V}^{(h)}$, $\boldsymbol{Q}^{(h)}$.
- Results are combined with a linear projection $\boldsymbol{W} \in \mathbb{R}^{Hd \times d}$.
- By stacking all outputs: $y = \boldsymbol{W}\left[y^{(1)}\ y^{(2)}\ \ldots y^{(H)}\right]^T$

Transformer

- Built from several *transformer blocks*.

- Each block contains:

  - Multi-Head Attention.
  - Residual connection $y = x + \mathrm{MHA}(x)$.
  - Normalisation.
  - Feed forward (fully connected) layer (operates per token).



Residual Connections

- Idea: $y = f(x)$ means that f needs to learn to pass on all necessary information to the following layer.
- It can be easier to only learn the changes in the representation: $y = f(x) + x$.
- Bonus: gradients flow much easier through the network.
- Residual blocks are now in almost all architectures CNNs and Transformers alike.

Normalisation

- Consider a single linear layer with ReLU

$$y = \max(0, Wx + b)\,.$$

- After initialisation with random (normal) noise, W and b can sometimes be hard to learn.
- For example:
  - If x are very close to 0, W needs to be very large.
  - If x is very negative -> large b to avoid 0 gradient from ReLU.
  - Etc.
- Idea: just normalise x so that it is “well-behaved” for optimisation.

Normalisation

- Typical training happens in *mini-batches* where we pass B samples through the model together.

- Gradients are computed using the full batch.
  - More stable than using only a single sample.
  - Much faster than computing the gradient using the whole dataset.

- Let’s use the batch statistics to normalise outputs of layers!

Batch-Normalisation

Ioffe and Szegedy, 2015

- Input: $x \in \mathbb{R}^{B \times d}$ (a batch of B vectors in $\mathbb{R}^d$)
- Output: $y = \mathrm{BN}(x) \in \mathbb{R}^{B \times d}$

- Mean: $\mu = \frac{1}{B}\sum_{i=1}^{B} x_i \in \mathbb{R}^d$
- Variance: $\sigma^2 = \frac{1}{B}\sum_{i=1}^{B}(x_i - \mu)^2 \in \mathbb{R}^d$
- Normalisation: $x'_{i,j} = \frac{1}{\sigma_j^2 + \epsilon}(x_{i,j} - \mu_j)$
- Learnable ($\gamma, \beta \in \mathbb{R}^d$) rescaling: $y_{i,j} = \gamma_j x'_{i,j} + \beta_j$

Batch-Normalisation

- Learning $\gamma = \sigma^2, \beta = \mu$ turns BN into an identity function.
- Usually requires reasonably sized batches to compute stable statistics.

- Depends on training statistics! For testing: use fixed statistics (exponential moving average) from train-set.

- Usually: after Fully Connected or Convolutional layers, and before nonlinearity.
- Idea: ReLU will 0-out *about* 50% activations.

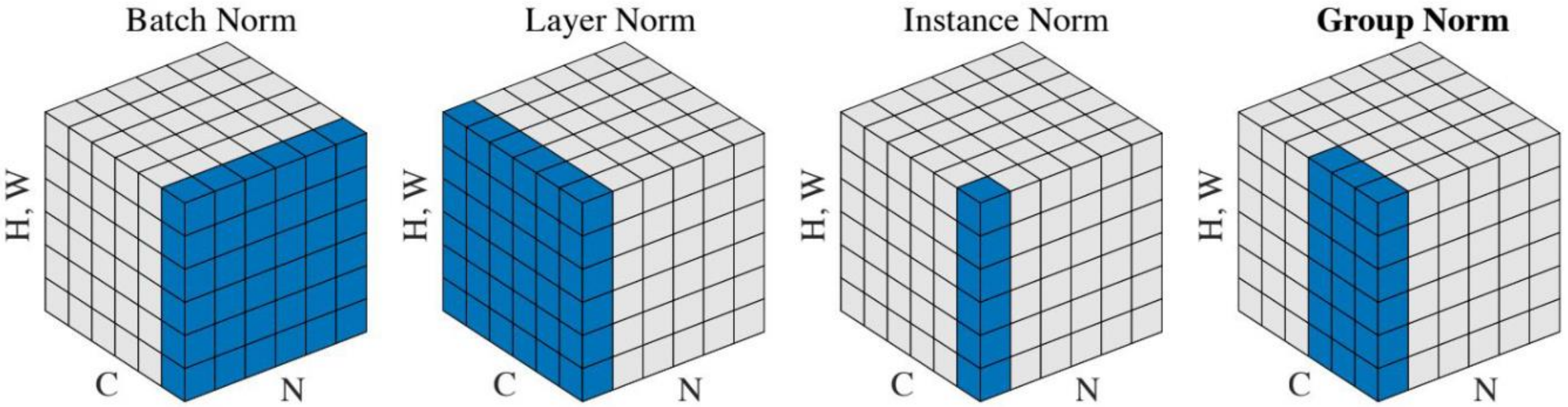
Batch-Normalisation

- Makes deep networks much easier to train.
- Improves gradient flow.
- Allows higher learning rates, faster convergence.
- Networks become more robust to initialization.
- Acts as regularization during training.
- Zero overhead at test-time: statistics are fixed and can be fused with previous layer!

- Behaves differently during training and testing: very common source of bugs!

Slide from F Li

Other Normalisation Layers



Wu and He, “Group Normalization”, ECCV 2018

Creativity of CV Researchers

- "**Show, attend, and tell**" (Xu et al., ICML 2015)
  Look at image, attend to image regions, produce queston
- "**Ask, attend, and answer**" (Xu and Saenko, ECCV 2016)
  "**Show, ask, attend, and answer**" (Kazemi and Elqursh, 2017)
  Read text of question, attend to image regions, produce answer
- "**Listen, attend, and spell**" (Chan et al., ICASSP 2016)
  Process raw audio, attend to audio regions while producing text
- "**Listen, attend, and walk**" (Mei et al., AAAI 2016)
  Process text, attend to text regions, output navigation commands
- "**Show, attend, and read**" (Li et al., AAAI 2019)
  Process image, attend to image regions, output text
- "**Show, attend, and interact**" (Qureshi et al., ICRA 2017)
  Process image, attend to image regions, output robot control commands

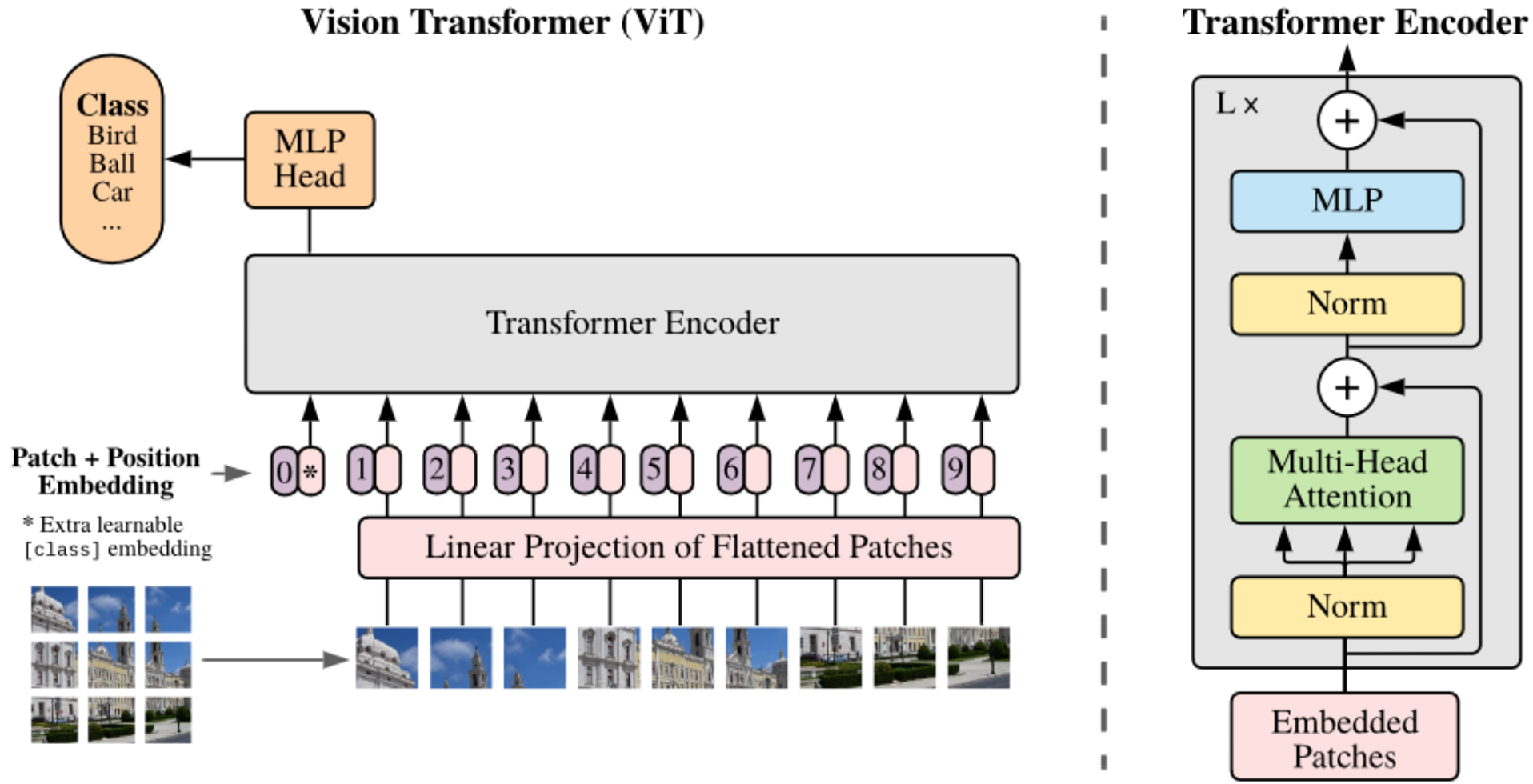
Slide adapted from J Johnson

Vision Transformers

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

- Split image into 16x16 patches.
- A linear layer that maps each patch to a vector (token).
- Add 2D positional encoding.
- Add one extra token to the *sequence* of 256. This token will in the end map to the class distribution.
- Train a transformer on sequences of length 257.

Vision Transformer

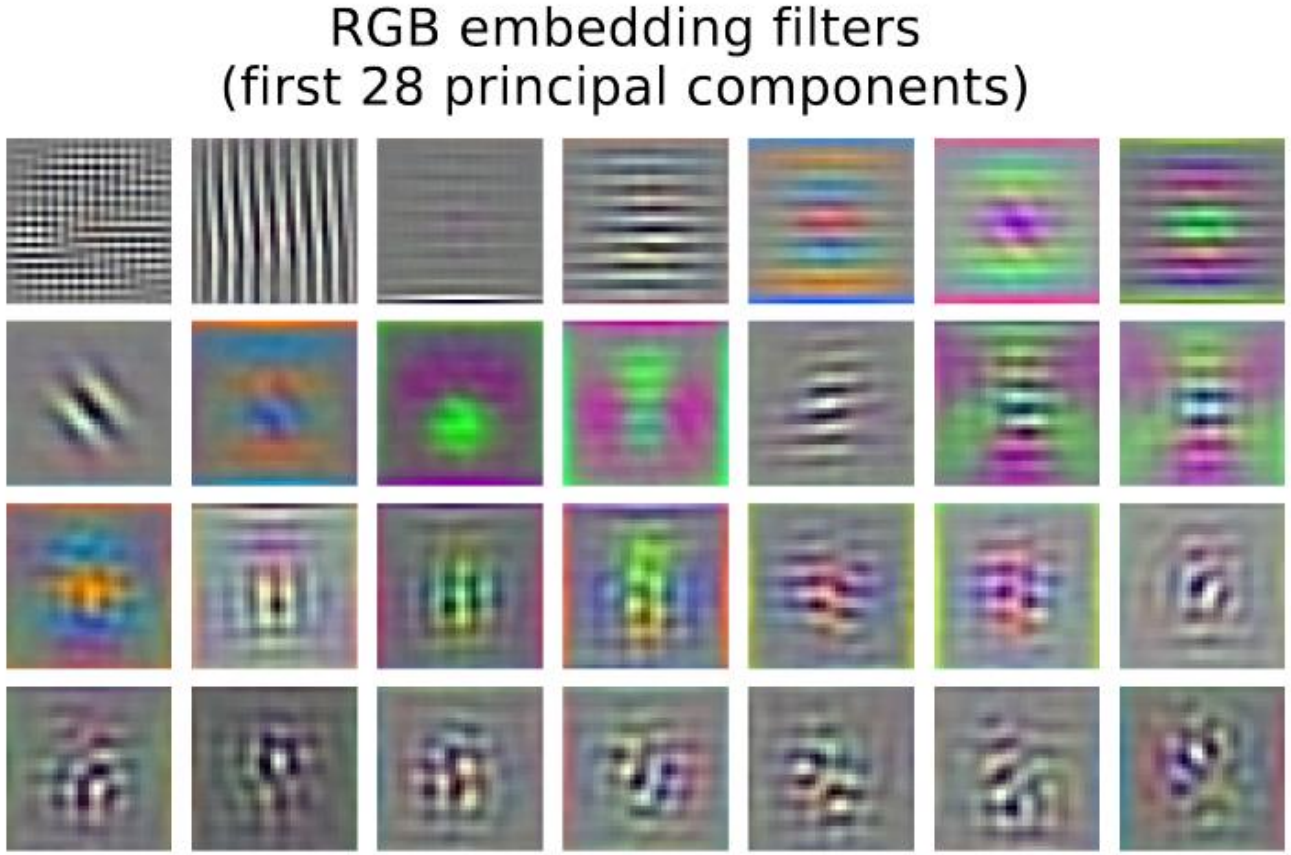


Vision Transformer

- Still seems to learn reasonable “filters” in the first layer.
- Trained with ADAM optimiser.

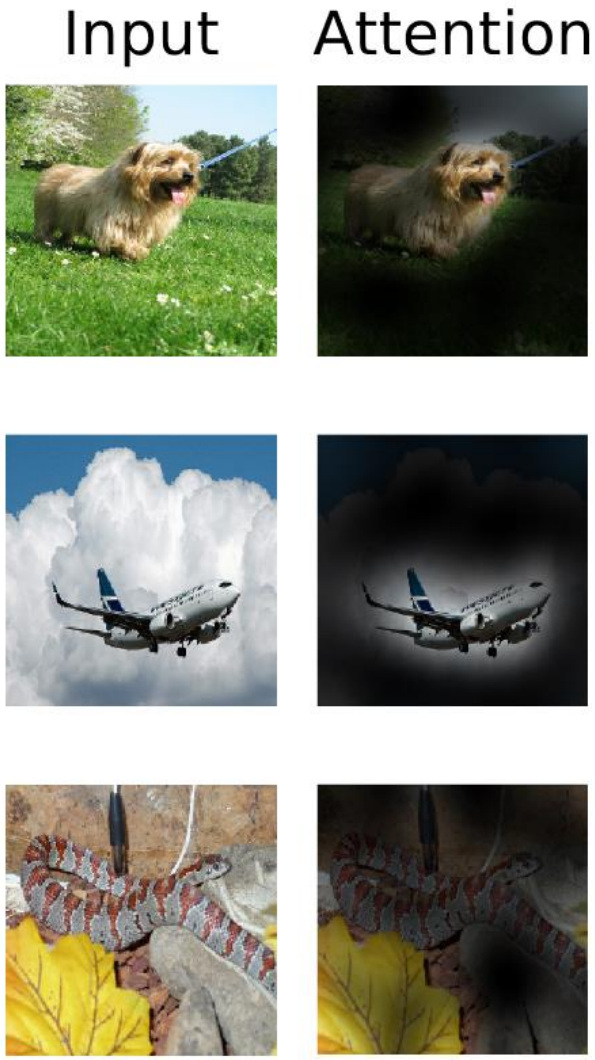
Tricks

- Checkpoint averaging
- Residual dropout
- Label smoothing



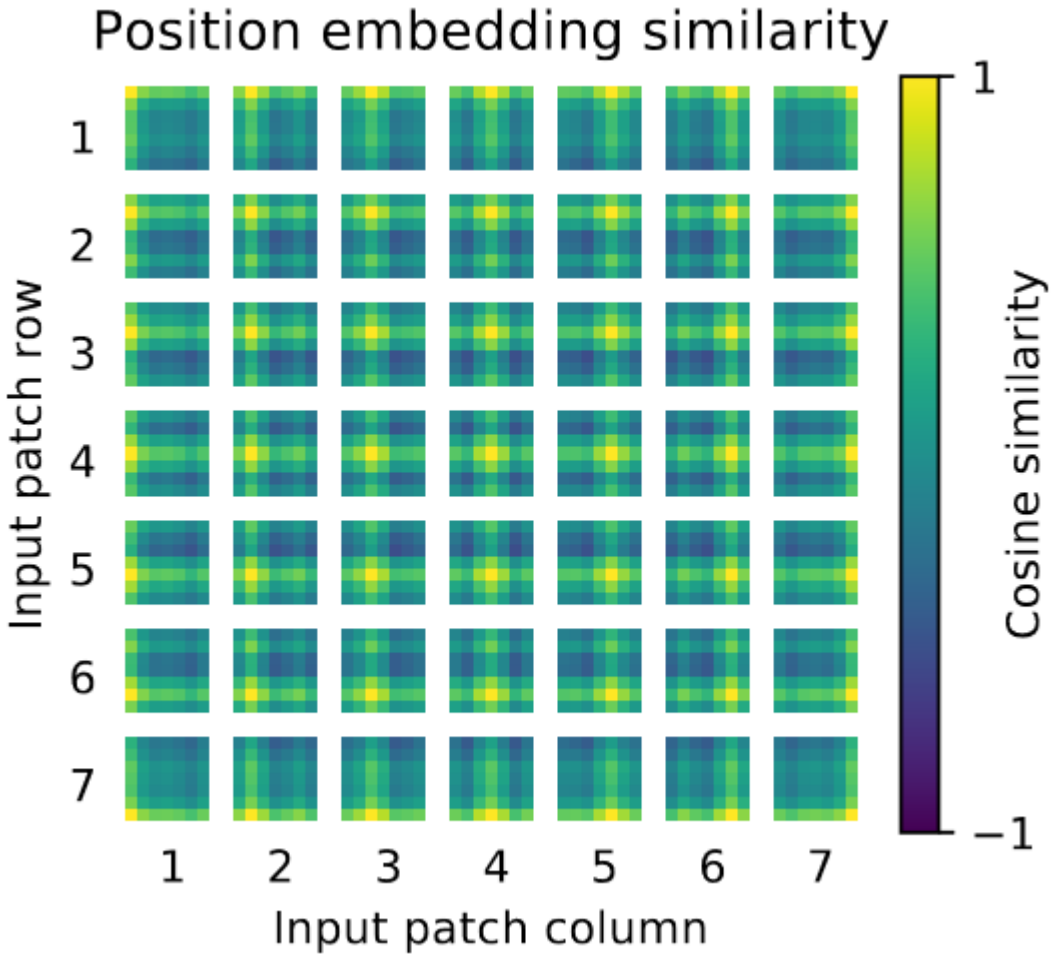
Attention Visualisation

- Attention weights are single scalars per token than sum to 1.
- We can visualise them as a heat-map and overlay them over the image.
- Seems to align with human intuition about what is important in an image.



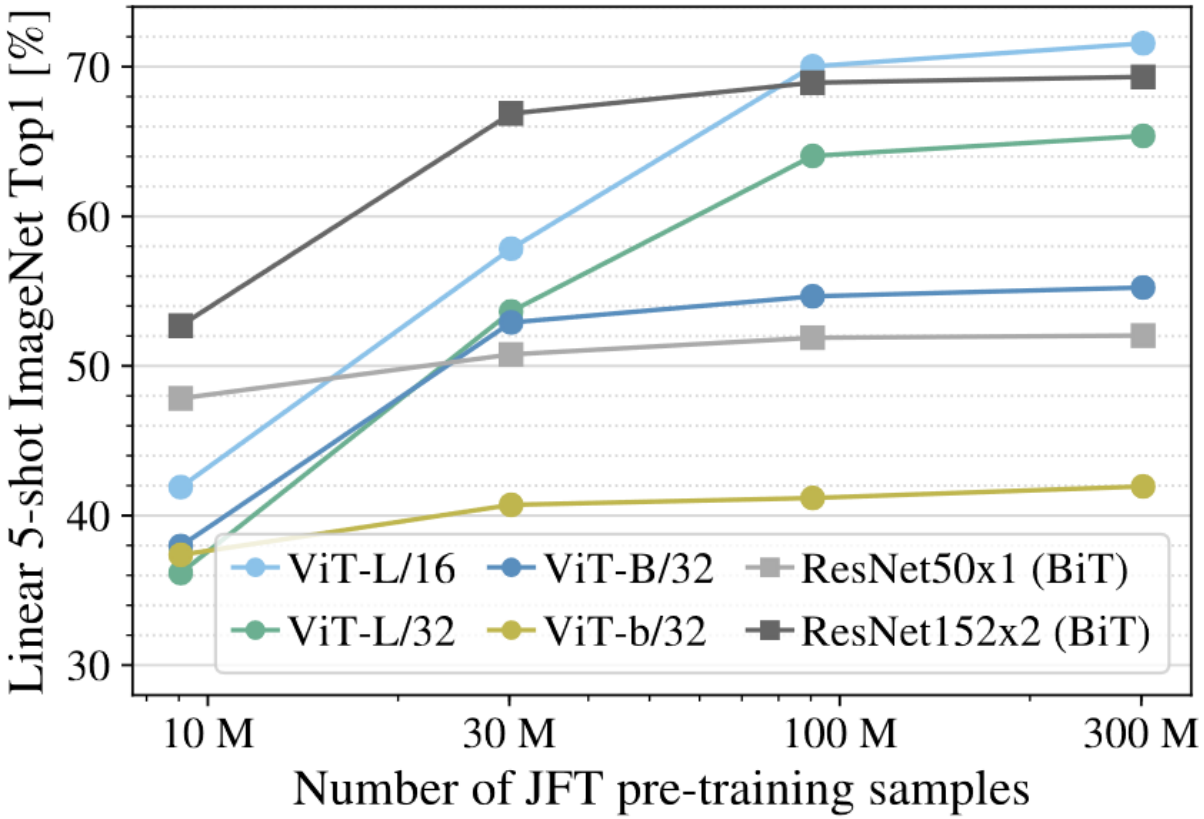
Positional Encoding

- ViT learns the positional encoding from scratch, instead of hand-crafting it.
- The learned embeddings have a strong local similarity.
- They can thus identify where a token comes from in the image.



Data

- ViTs benefit from very large datasets.
- BiT is a CNN (ResNet).
- Transfer learning becomes even more important!



Transfer Learning

- Train on a large dataset for some (related) task.
- Then *fine-tune* on *your* task that has less data.

- Fine-tuning usually with a lower learning rate or some layers frozen.

- Intuition: learned weights for another task often still much better than random initialisation.
- Evidence: first layer filters almost always look the same.

The Encoder-Decoder Transformer

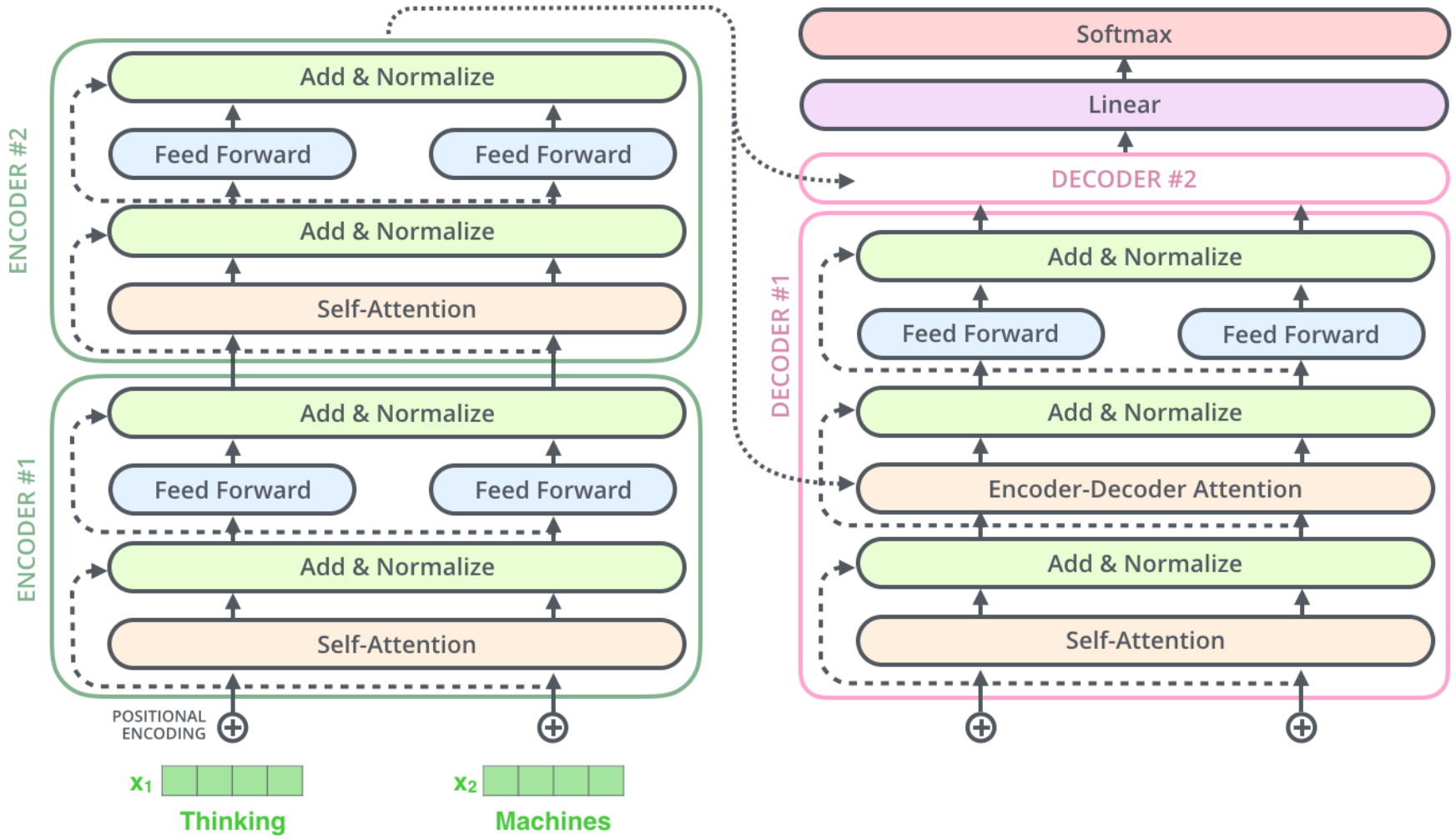


Image source

Encoder block

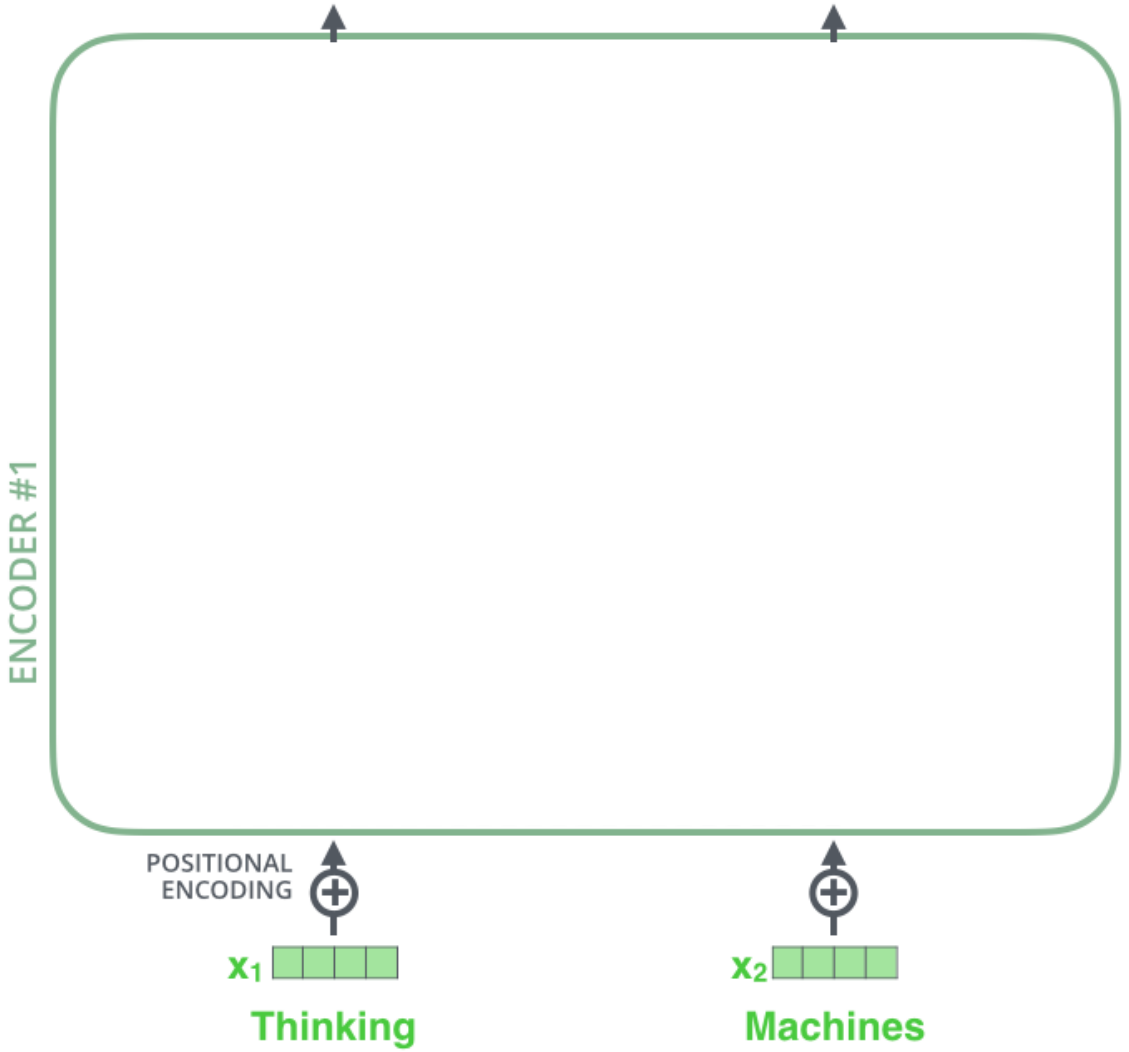


Image source

Encoder block

Encoder self-attention

All tokens interact with each other

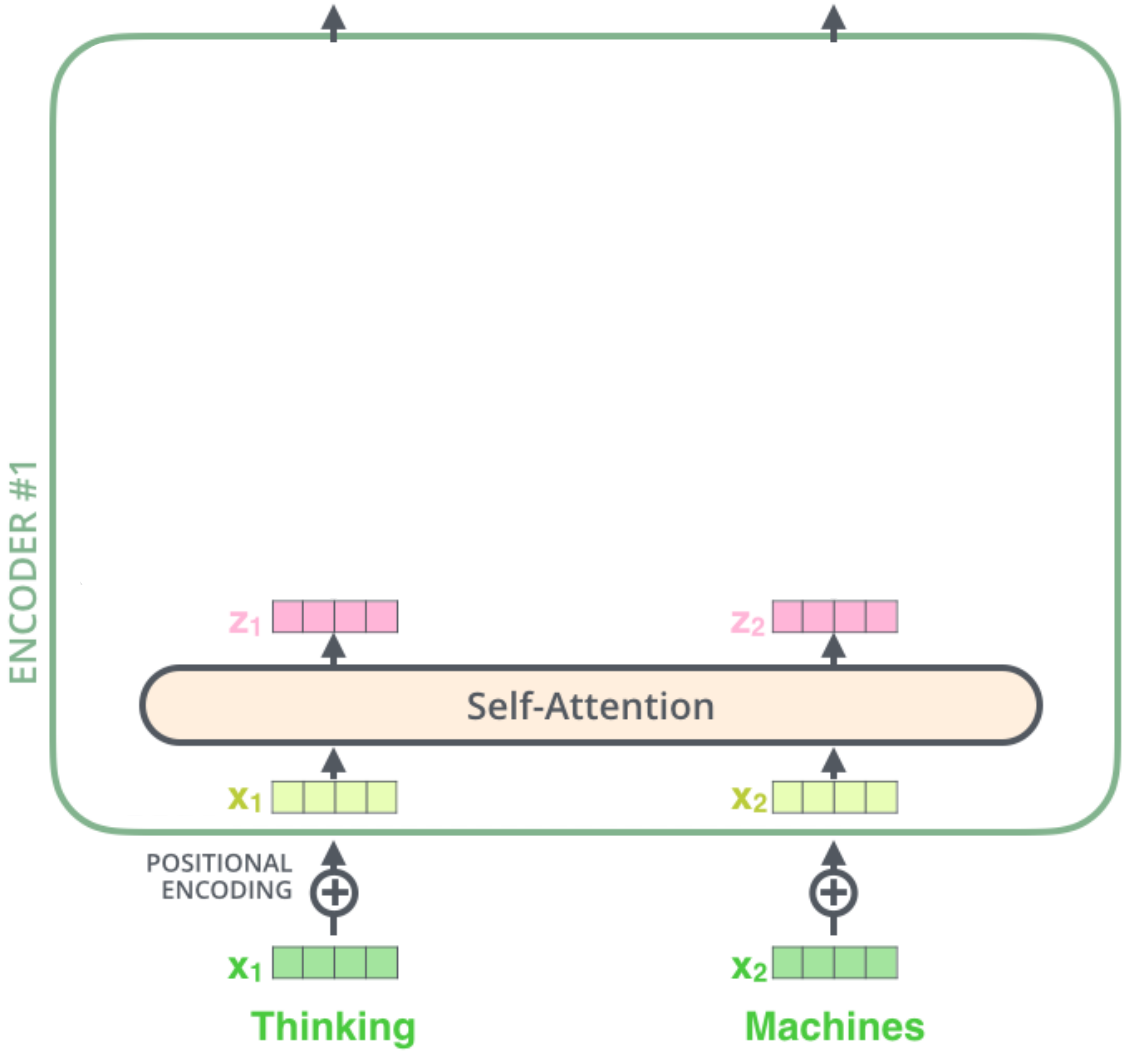


Image source

Encoder block

Residual connection &
Layer normalization

All tokens interact with each other

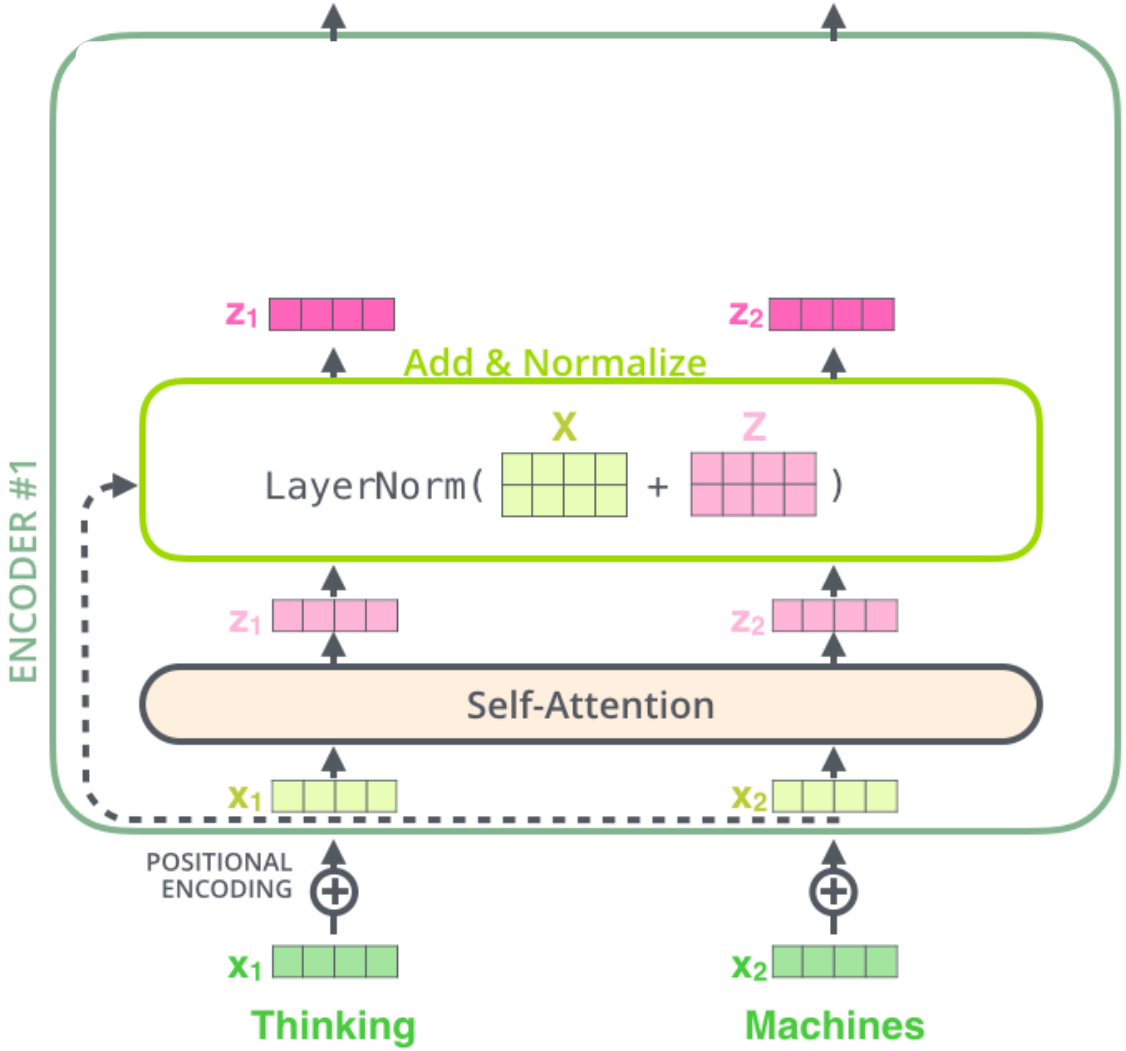


Image source

Encoder block

MLPs independently on each token
(no interaction)

Residual connection &
Layer normalization

All tokens interact with each other

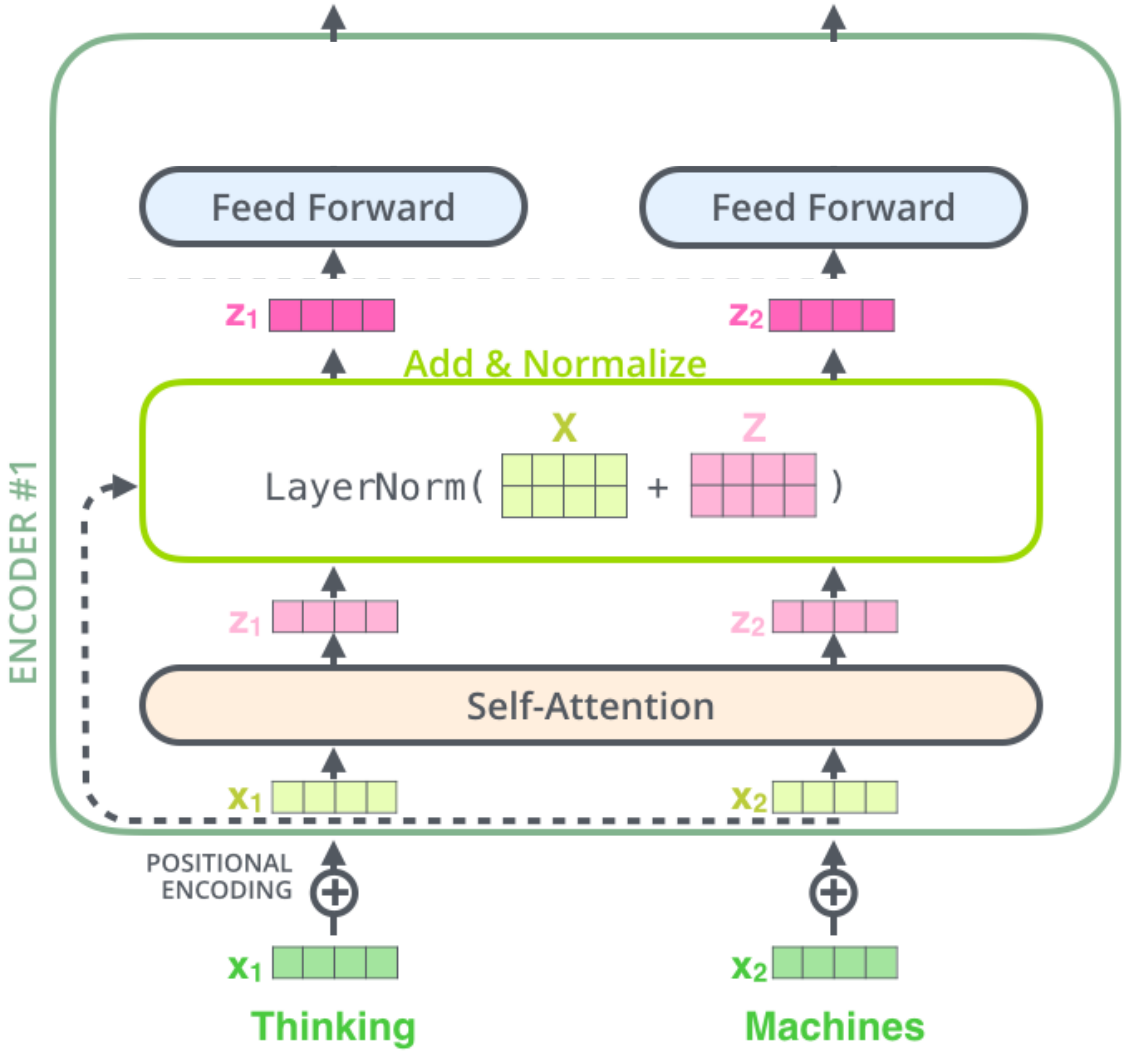


Image source

Encoder block

MLPs independently on each token
(no interaction)

Residual connection &
Layer normalization

All tokens interact with each other

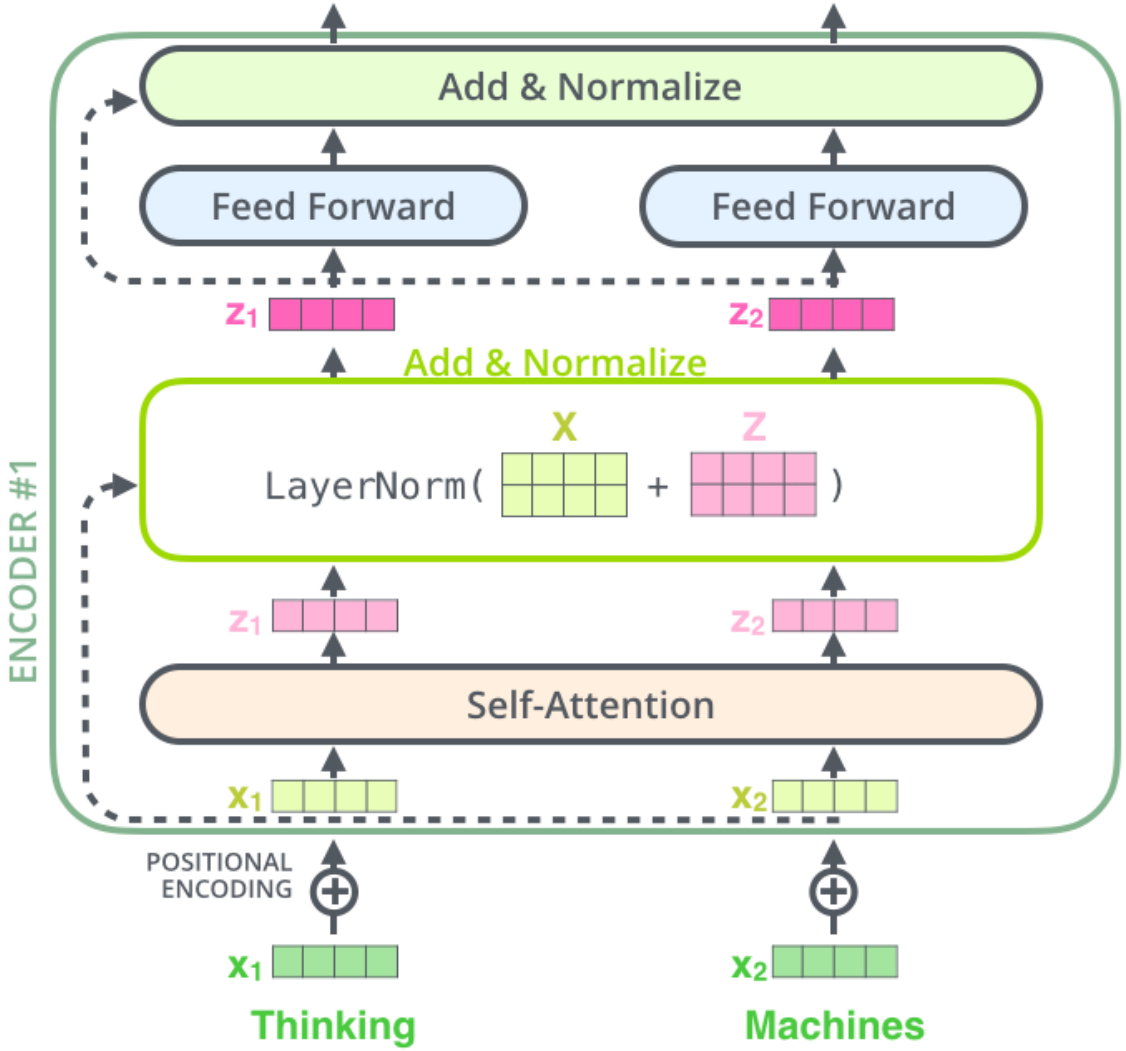
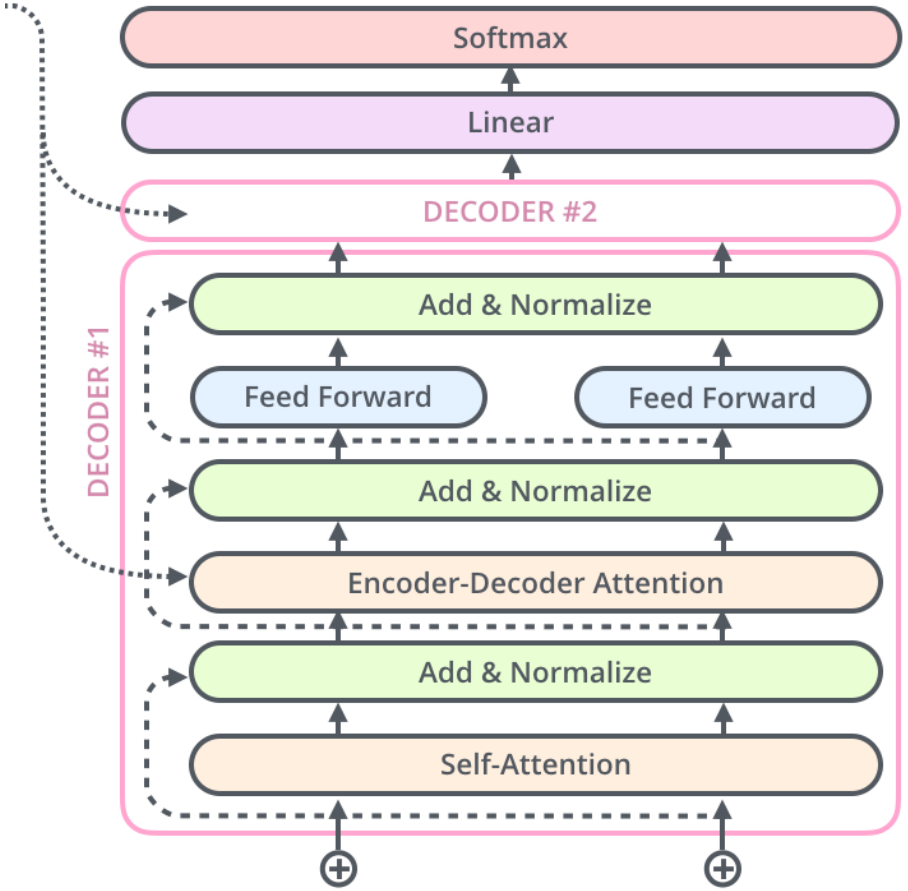


Image source

Decoder block

- Similar to encoder block
- Input tokens are pre-defined/learned
- **Encoder-Decoder attention**
  - Queries $\boldsymbol{Q}$ from the output of the previous layer of the decoder
  - Keys $\boldsymbol{K}$ and values $\boldsymbol{V}$ from the output of the encoder
  - Every position in the decoder attends over all positions in the input sequence
  - Often also called “cross attention”



ConvNeXt? – Probably not

A ConvNet for the 2020s, Liu et al., 2022

- Main changes:
  - Grouped convolutions
  - Wider network
    (more filters per layer)
- So far:
  transformers still dominant.