

UNIVERSITY OF OSLO

IN4310 Deep Learning for Image Analysis

Lecture 10: Vision Transformers

March 28, 2025

Vilde Schulerud Bøe

Slides based on Dhananjay Tomar

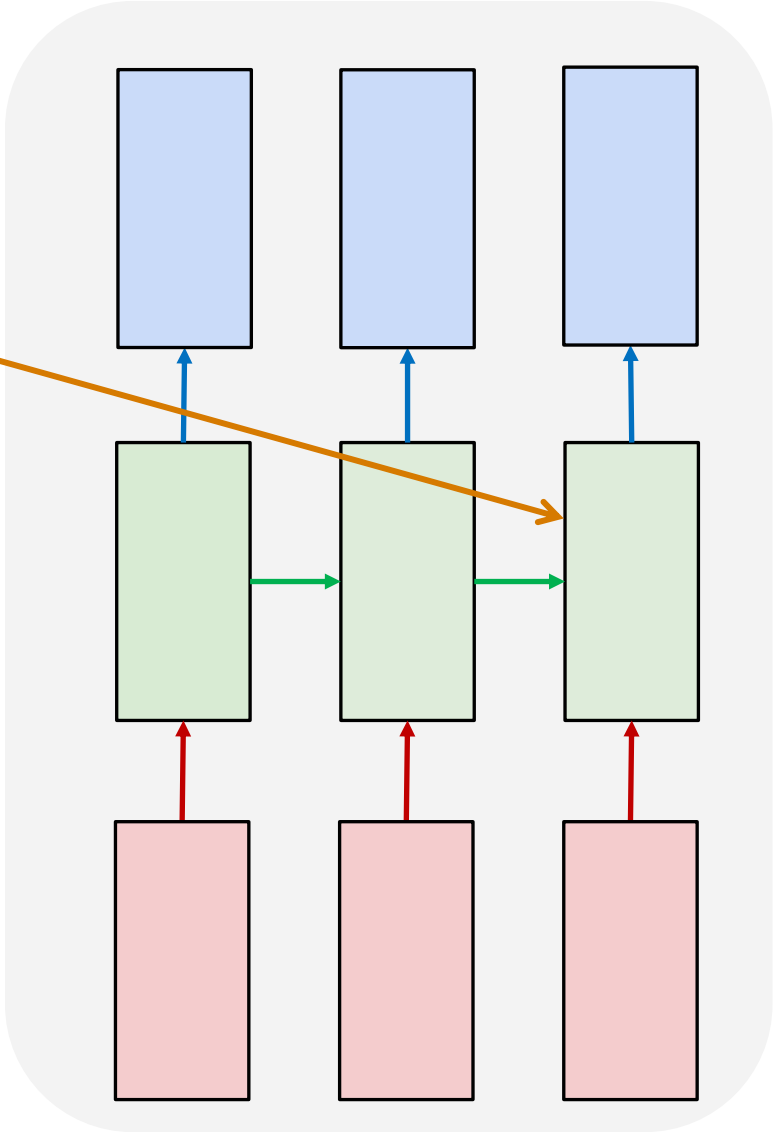
Outline

- **Attention Mechanism**
- Self-attention
- Multi-head self-attention
- Transformer encoder block
- Transformer decoder block
- Vision transformer (ViT)
- Swin transformer
- Suggested resources
- Mandatory Assignment 2
- (DINO)

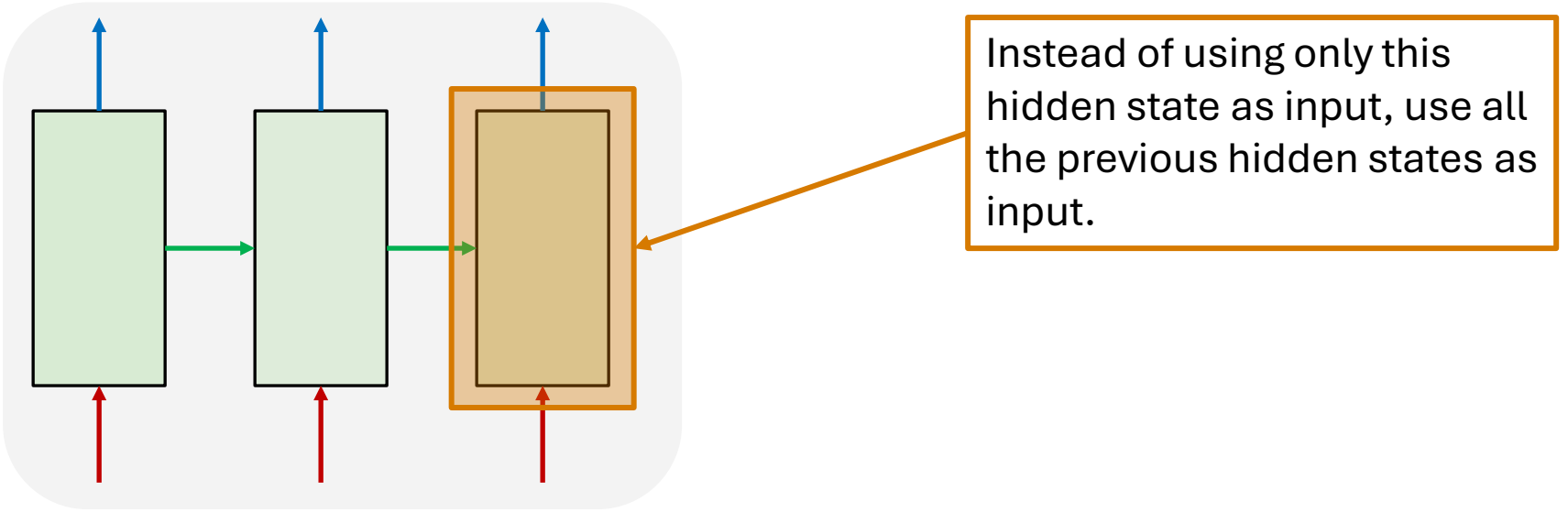
RNNs suffer from a bottleneck problem

This vector must encode all the previous hidden states.

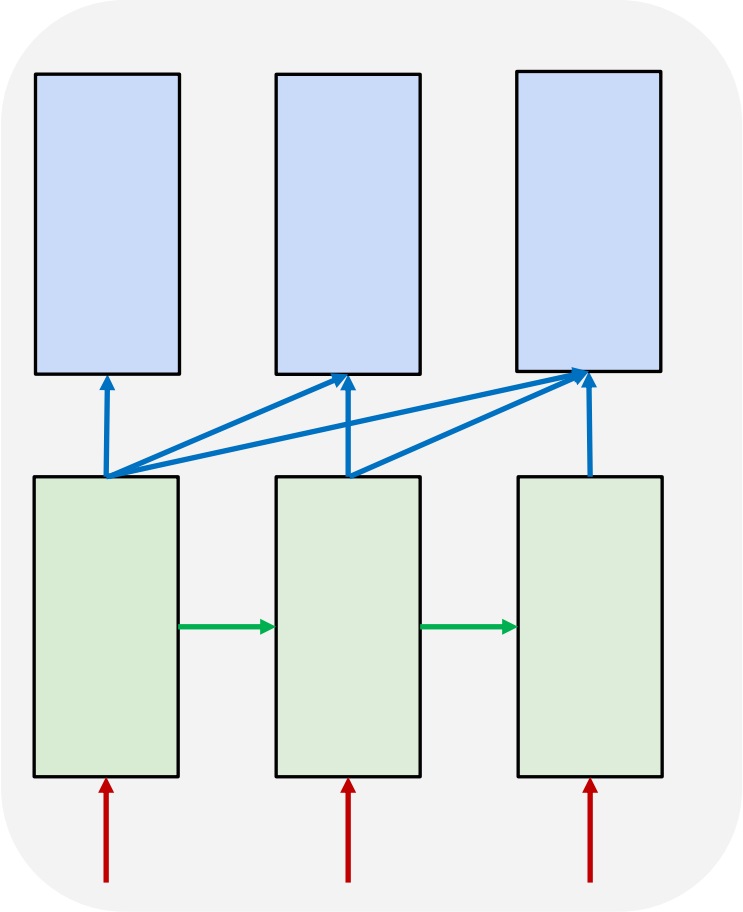
This keeps getting more difficult with longer sequences.



Idea: Use all previous hidden states



Idea: Use all previous hidden states

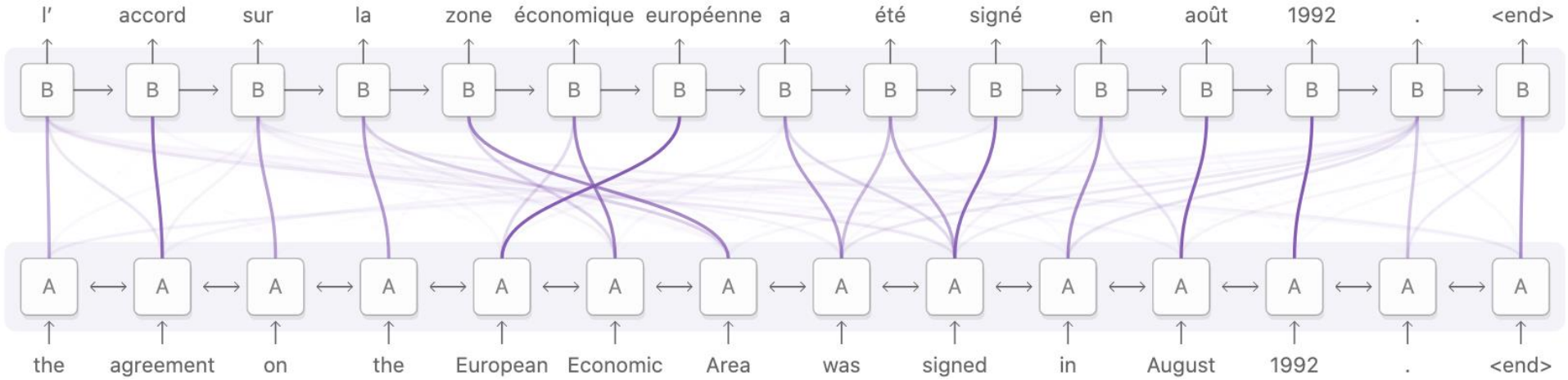


Naïve method: Connect the output layer to all previous hidden states.
It has at least three problems:

- **Additional parameters:** Adding connections to k hidden states instead of just one will increase the parameter count k-times for the layer.
- **Fixed length:** Since the parameters of a network are fixed, we can process hidden states only up to a certain length.
- **Training long sequences:** If the sequence length in the training data varies, some parameters might receive an insufficient amount of gradient updates to learn something useful.

Attention mechanism

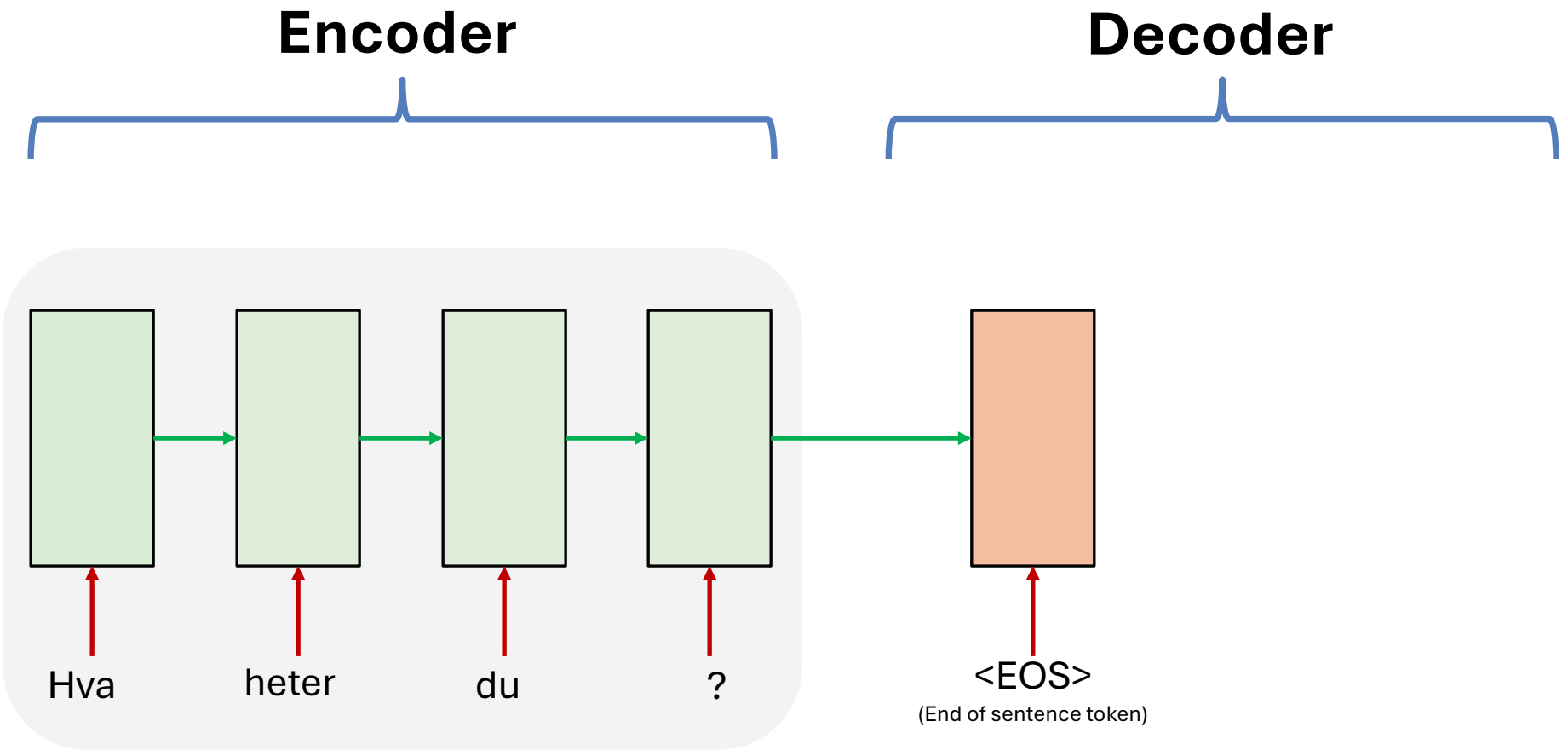
- When people say “attention mechanism”, they usually refer to either the attention mechanism proposed by *Bahdanau et al.* (2014) for Neural Machine Translation (NMT) or one of its derived forms.
- Basic intuition: use all the hidden states of the input sentence but focus more on the relevant ones.



Source: Olah & Carter, "Attention and Augmented Recurrent Neural Networks", Distill, 2016. http://doi.org/10.23915/distill.00001
Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

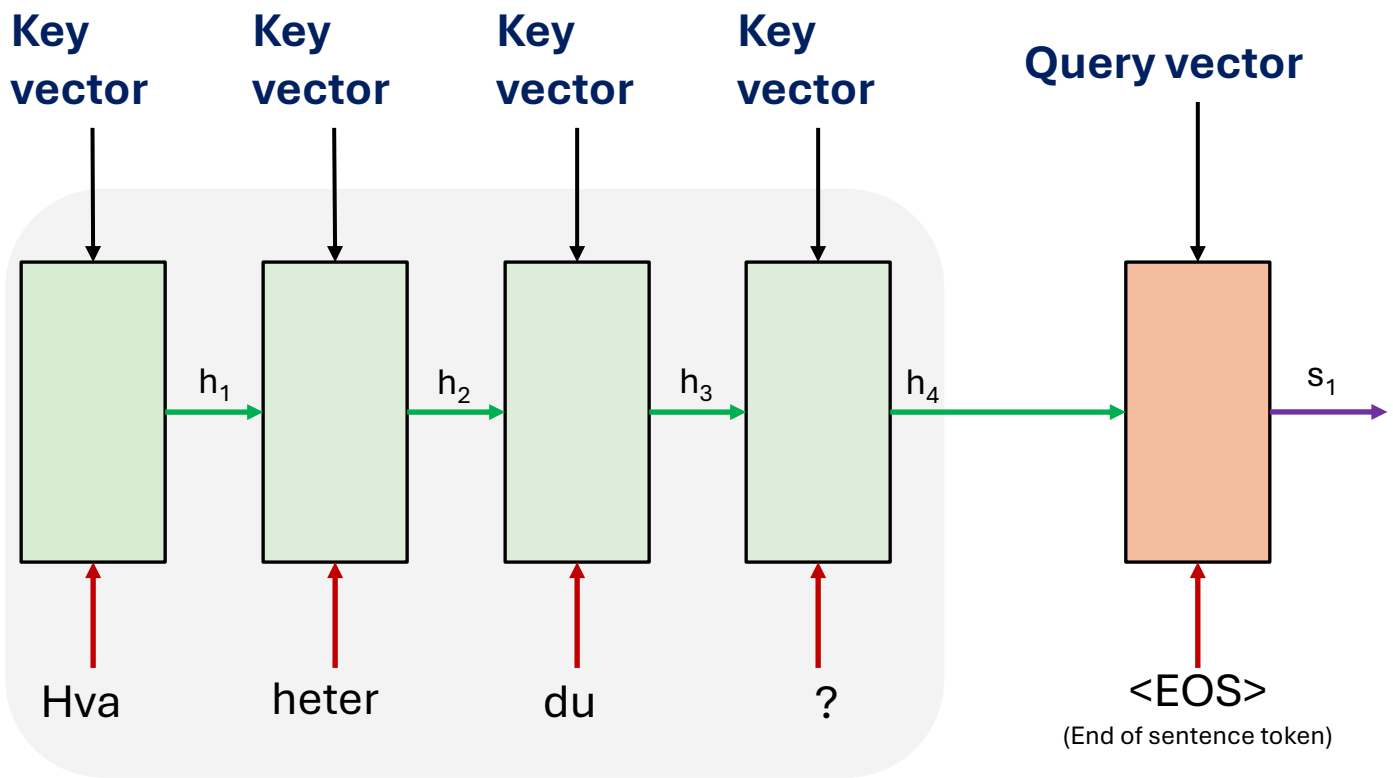
Attention mechanism

- We will look at attention mechanism via Neural Machine Translation (NMT), i.e., translating text from a source language to a target language using neural networks.
- NMT typically uses an encoder-decoder architecture.
- The decoder makes use of the attention mechanism.

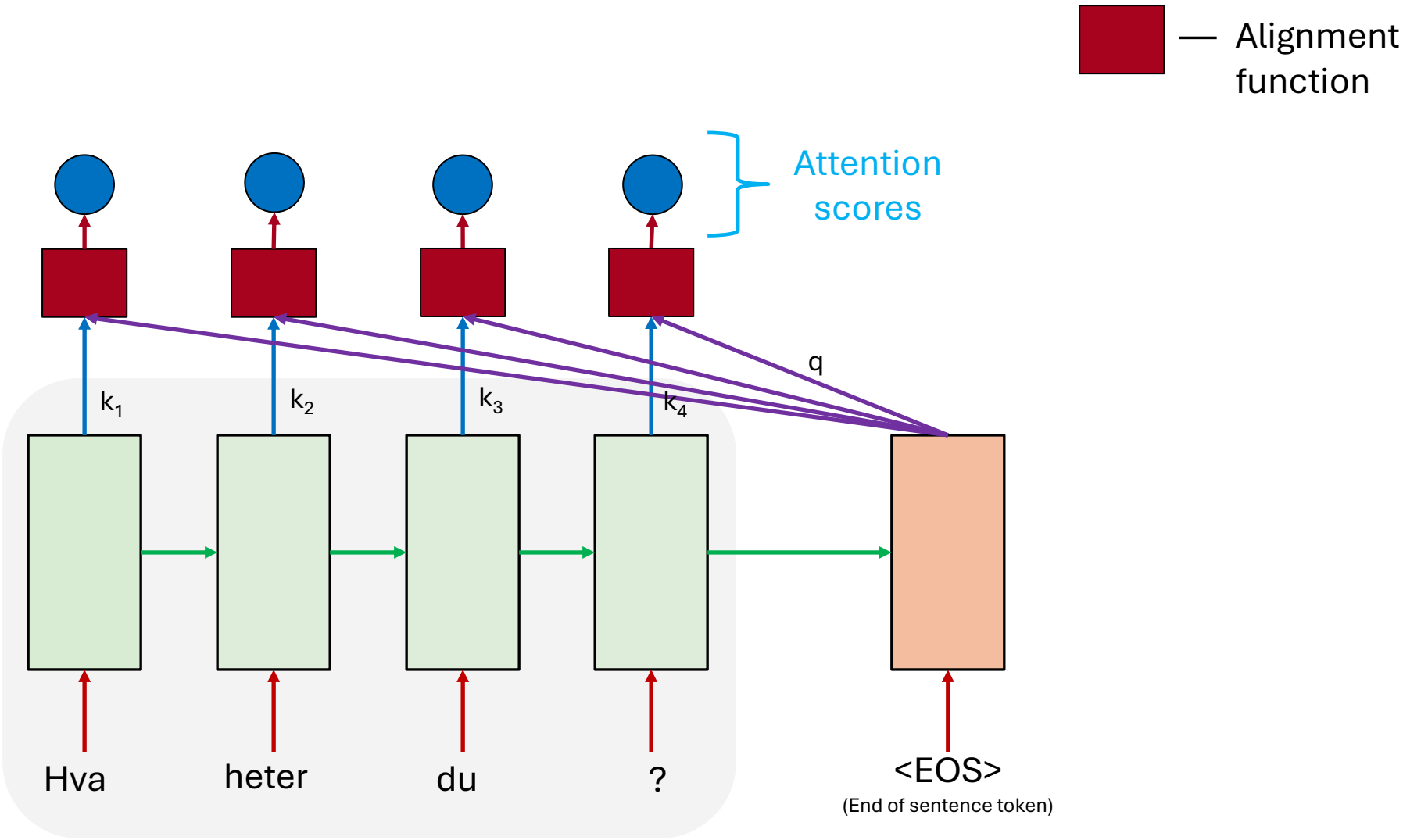


Attention mechanism

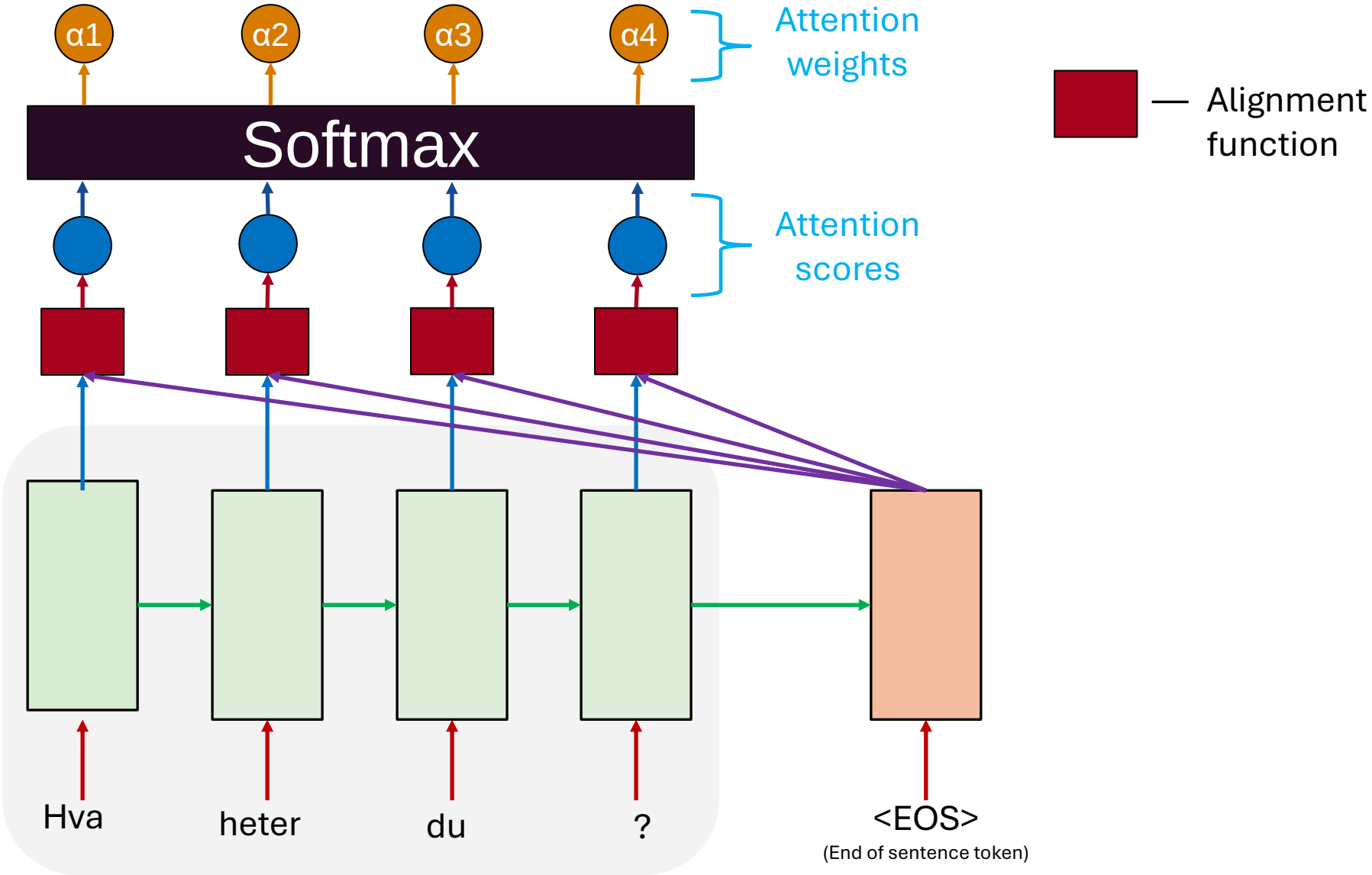
- The input to the attention mechanism is a single query vector and multiple key vectors. Additionally, we can also have value vectors.
- Each query-key pair gets a score which determines how much attention each key vector (or the value vector if present) will get.
- We will use the encoder's hidden states as both key and value vectors.



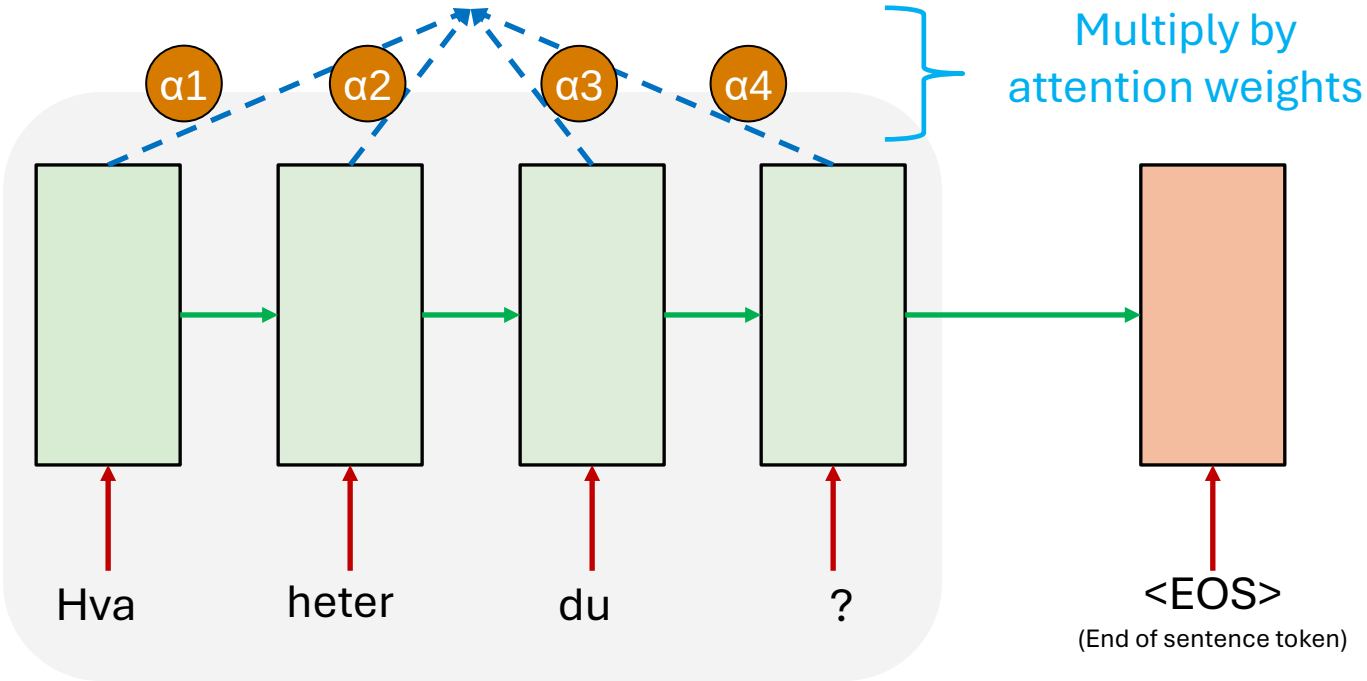
Attention mechanism



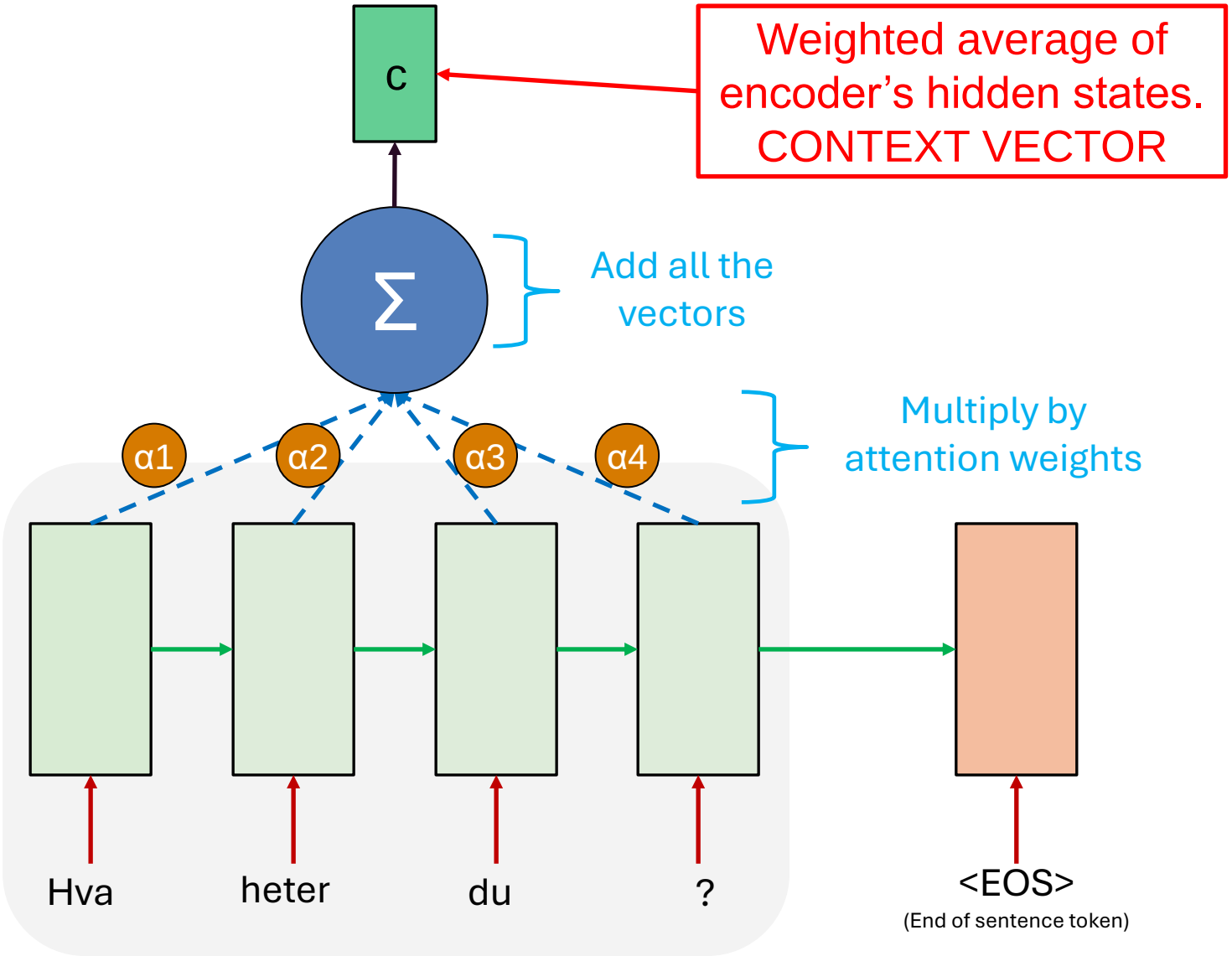
Attention mechanism



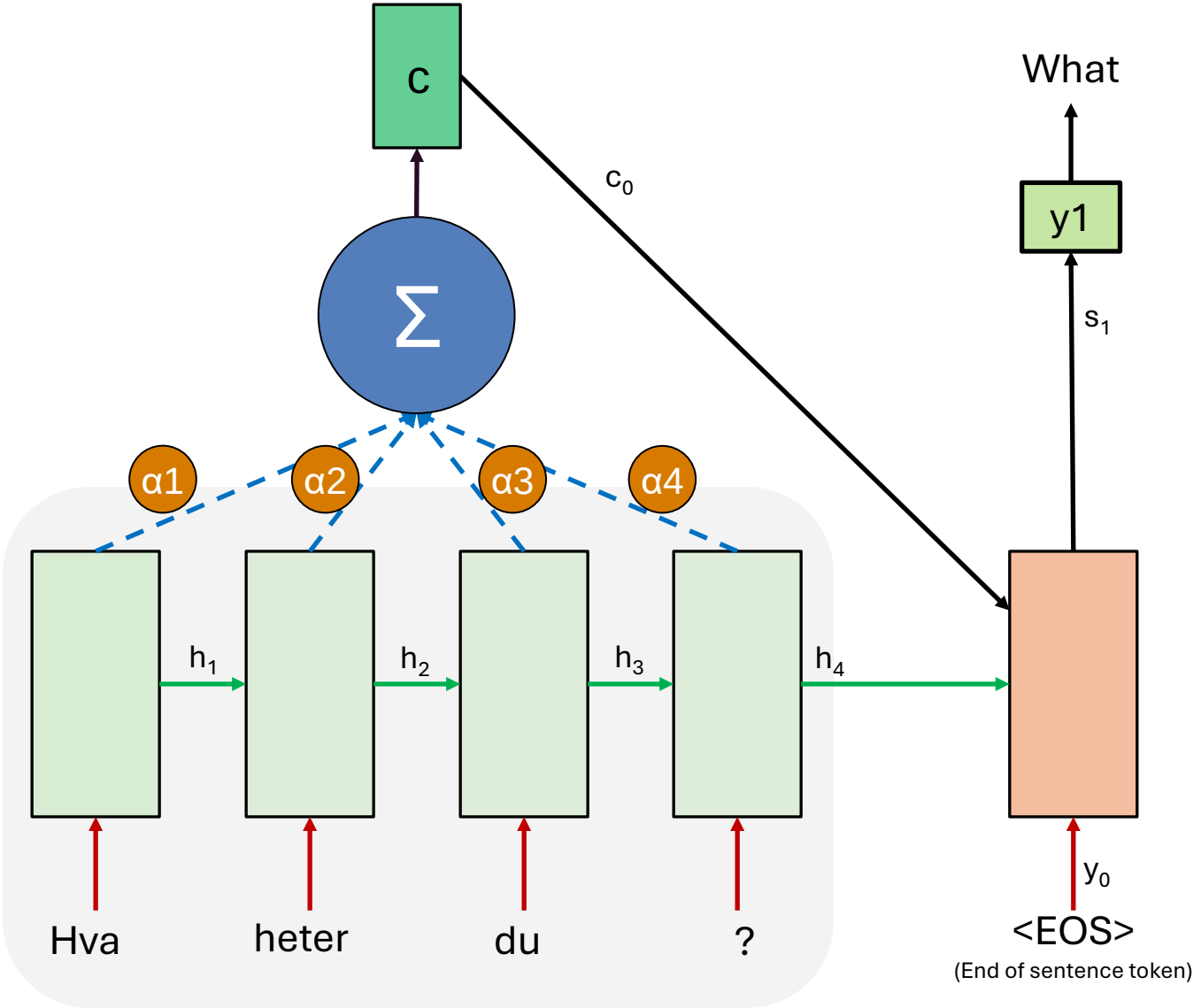
Attention mechanism



Attention mechanism

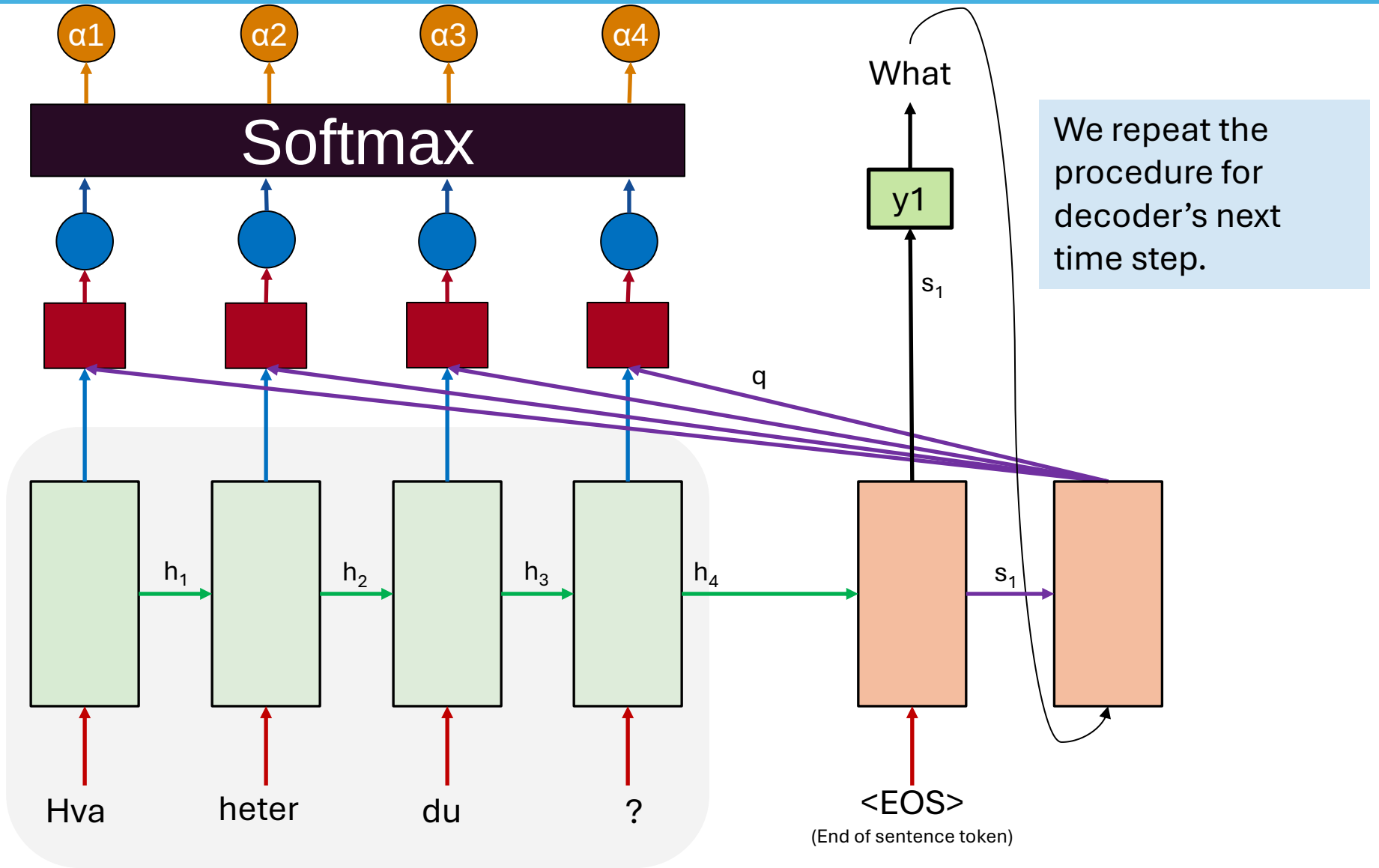


Attention mechanism



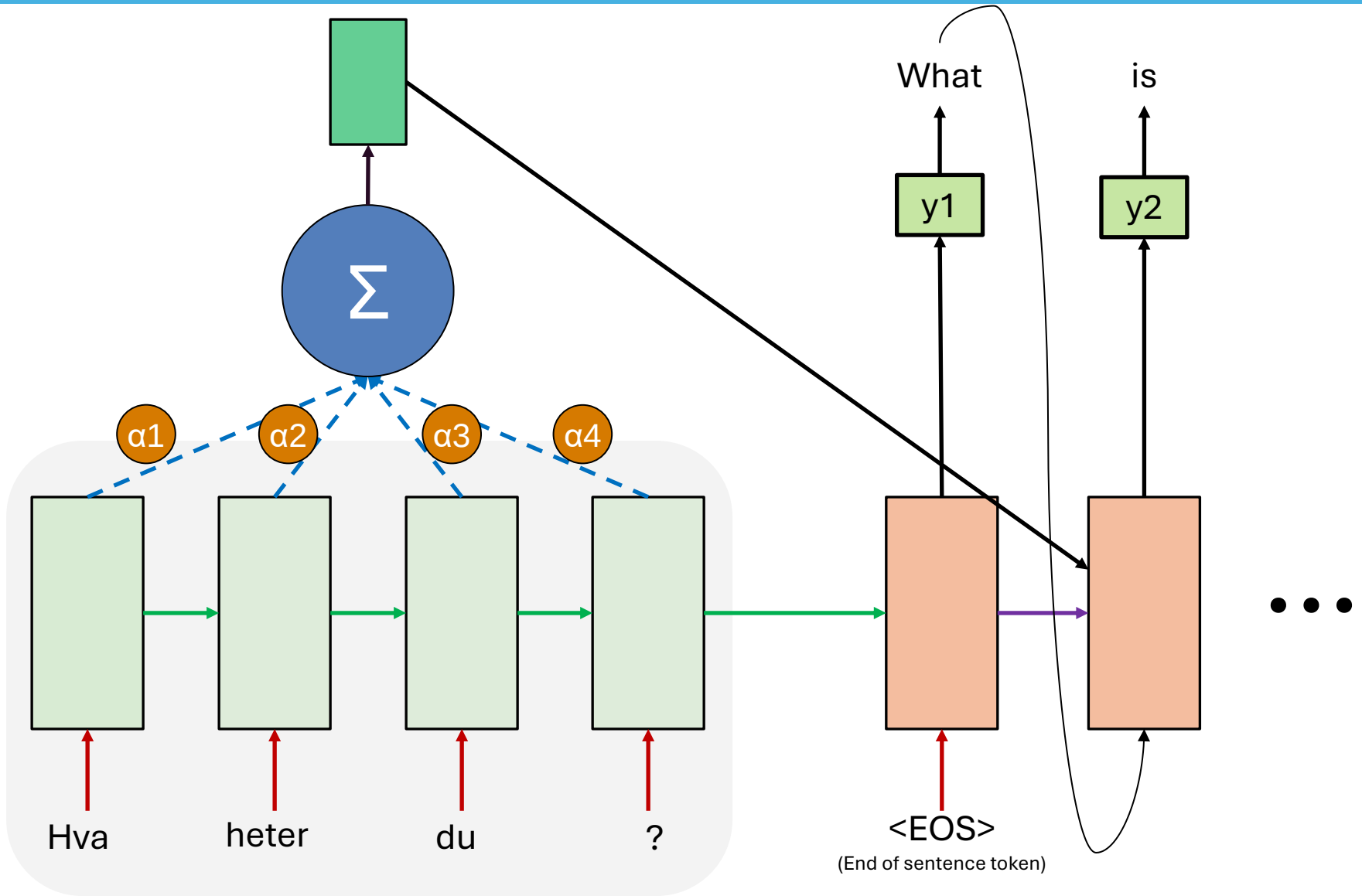
The context vector(c_0) and the hidden state of the decoder (s_0) are used to generate output y1.

Attention mechanism



We repeat the procedure for decoder's next time step.

Attention mechanism



Attention mechanism

- **Alignment function** takes a query (q) and a key (k) vector as input and outputs a score for the query-key pair.
- Popular alignment functions:
  - Additive/concat attention: $a(q,k) = v^T \tanh(W_a[q;k])$ *Bahdanau et al. 2014*
  - General attention: $a(q,k) = q^T W_a k$ *Luong et al. 2015*
  - Dot-product attention: $a(q,k) = q^T k$ *Luong et al. 2015*
  - Scaled dot-product attention: $a(q,k) = \frac{q^T k}{\sqrt{d_k}}$ $where\ d_k = length\ of\ k$ *Vaswani et al. 2017*
- Transformers use scaled dot-product attention in self-attention.

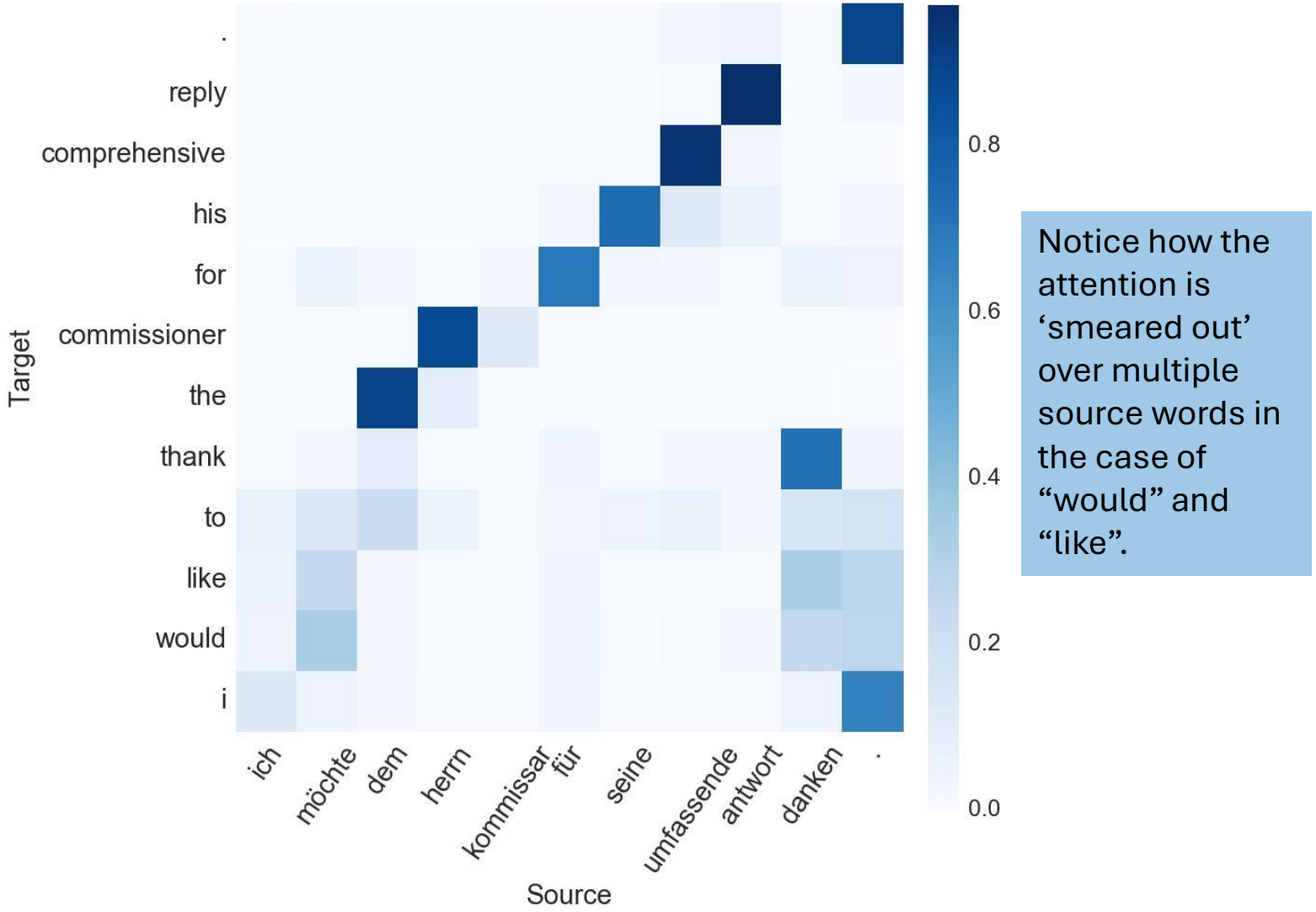
Attention mechanism

- Attention mechanism solves the bottleneck problem.
  - The encoder doesn’t need to store all the information in the last hidden state, as the decoder now looks directly at all the hidden states.

- Attention mechanism helps with the vanishing gradient problem.
  - Gradients can now be transferred directly from the decoder to all the time steps in the encoder.

- Attention mechanism provides some interpretability.
  - By looking at the attention weights, we can determine what the decoder focused on.

Attention mechanism: interpretability example



Notice how the attention is ‘smeared out’ over multiple source words in the case of “would” and “like”.

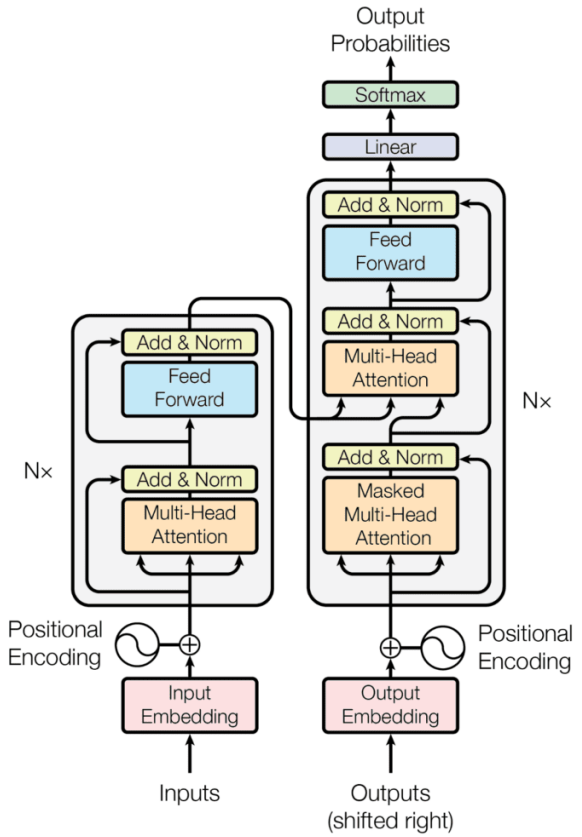
Source: Ghader, Hamidreza, and Christof Monz. "What does Attention in Neural Machine Translation Pay Attention to?." *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2017.

Attention mechanism: interpretability example



Source: Xu, Kelvin, et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention." *arXiv preprint arXiv:1502.03044* (2015).

Transformers



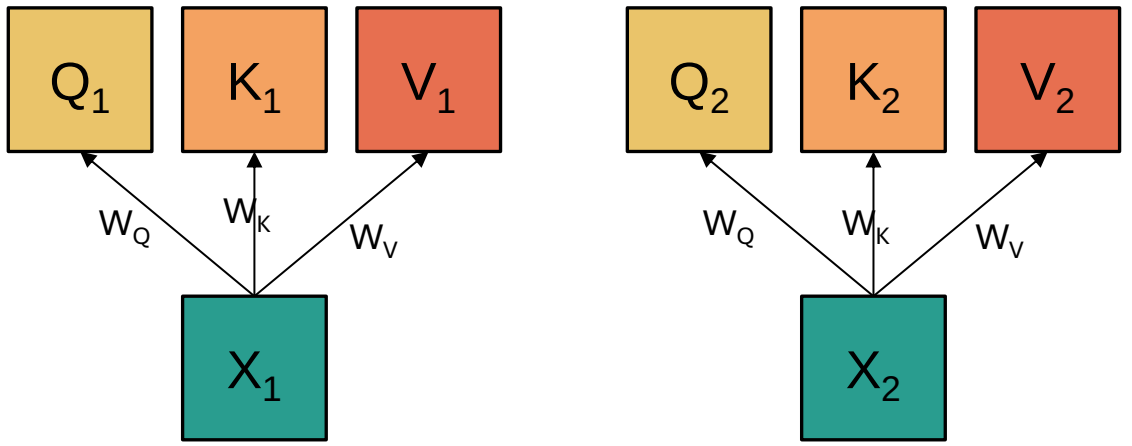
Source: Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017). link

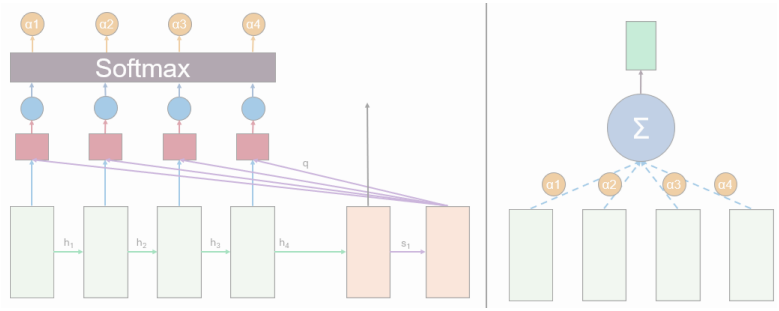
Outline

- Attention Mechanism
- **Self-attention**
  - Positional encoding
  - Masked self-attention
- Multi-head self-attention
- Transformer encoder block
- Transformer decoder block
- Vision transformer (ViT)
- Swin transformer
- Suggested resources
- Mandatory Assignment 2

Self-Attention: Step 1: Compute Query, Key, Value vectors

- Encodes relationships among the input
- In step 1, each input vector X_i is multiplied with 3 different matrices to get query, key, and value vectors.
- You can imagine this as having 3 **`torch.nn.Linear`** layers:
  - `L_Q = torch.nn.Linear(size_of_x_vector, size_of_query_vector)`
  - `L_K = torch.nn.Linear(size_of_x_vector, size_of_key_vector)`
  - `L_V = torch.nn.Linear(size_of_x_vector, size_of_value_vector)`
  - `Q = L_Q(X)K = L_K(X)  V = L_V(X)`
- Query, key, and value vectors typically have the same dimensions.

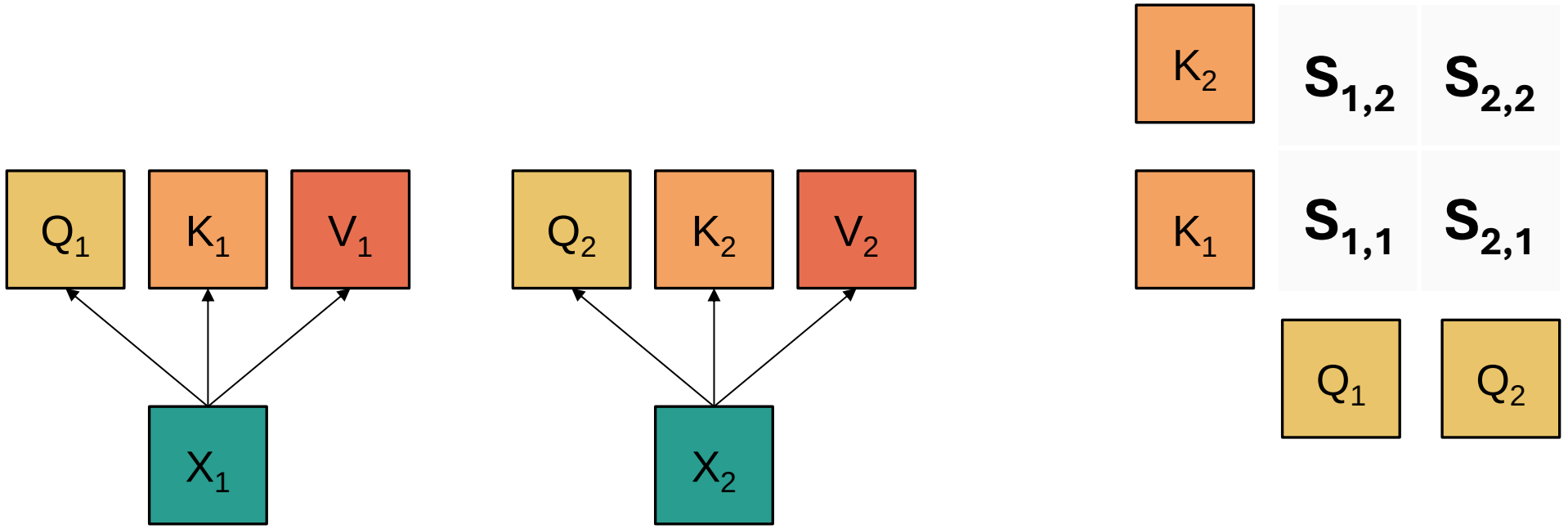




Previously introduced attention

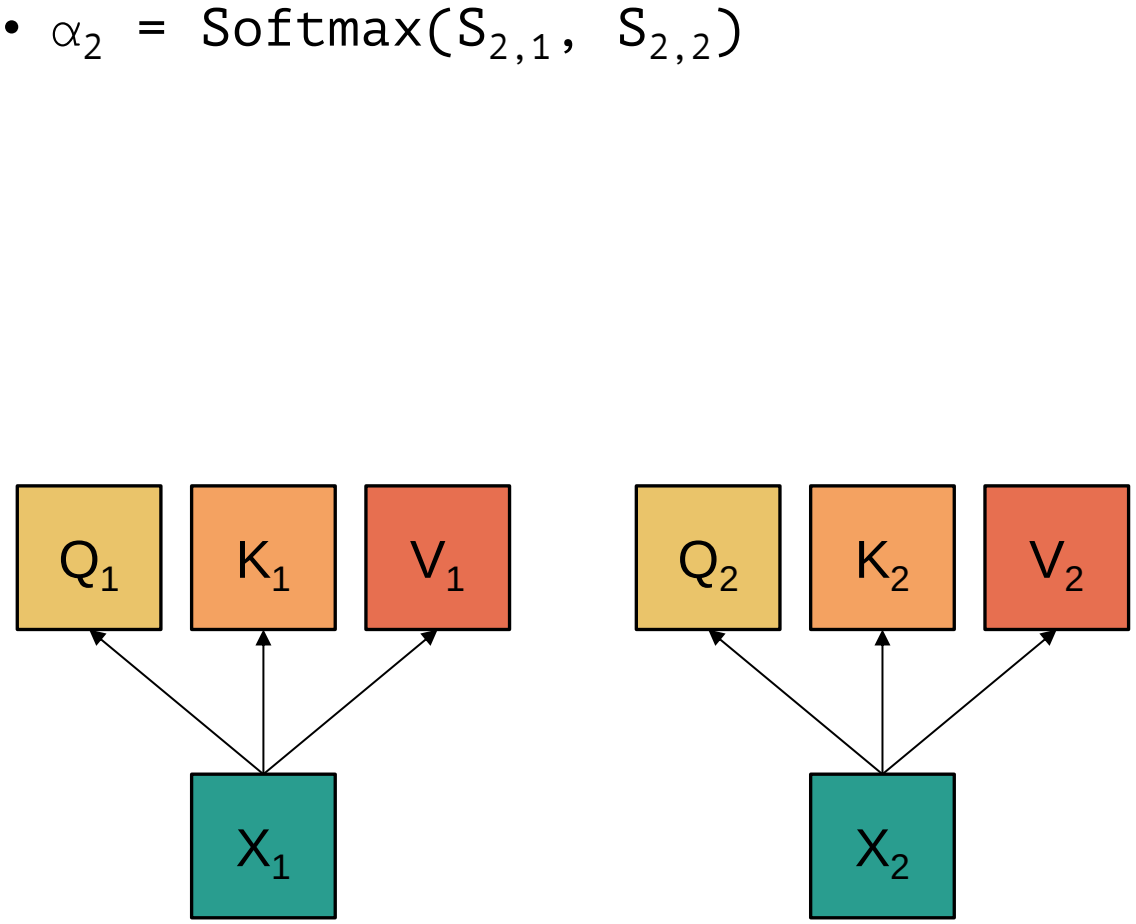
Self-Attention: Step 2a: Compute score

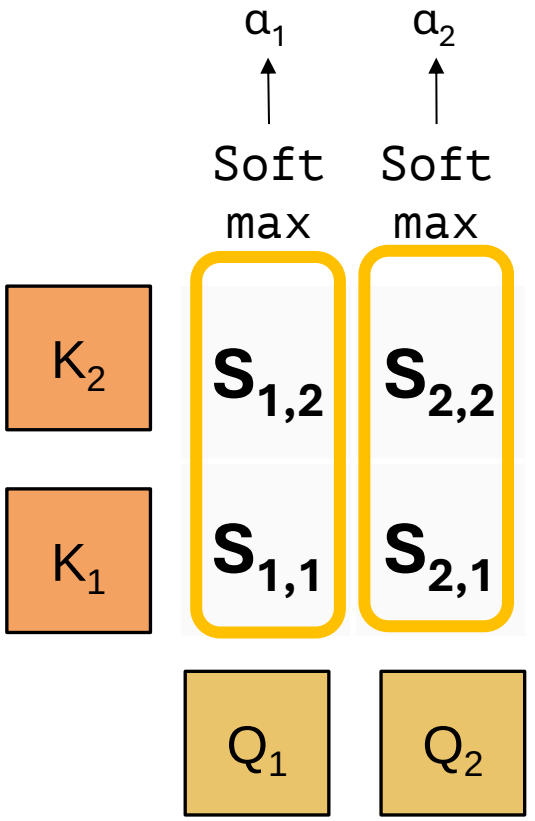
- Apply *scaled dot product attention* using query and key vectors to get scores for each pair.
- Each K_i, Q_i pair generates a score.
  - $S_{1,1}$ = dot_product(Q_1, K_1) $S_{1,2}$ = dot_product(Q_1, K_2) $S_{2,1}$ = dot_product(Q_2, K_1) $S_{2,2}$ = dot_product(Q_2, K_2)
- Each score is divided by the square root of the dimension of the key vector.
  - $S_{i,j}$ = $S_{i,j}$ / square_root(dimension of K_i vectors)
  - This is done to stabilise gradients during backpropagation.



Self-Attention: Step 2a: Compute Softmax

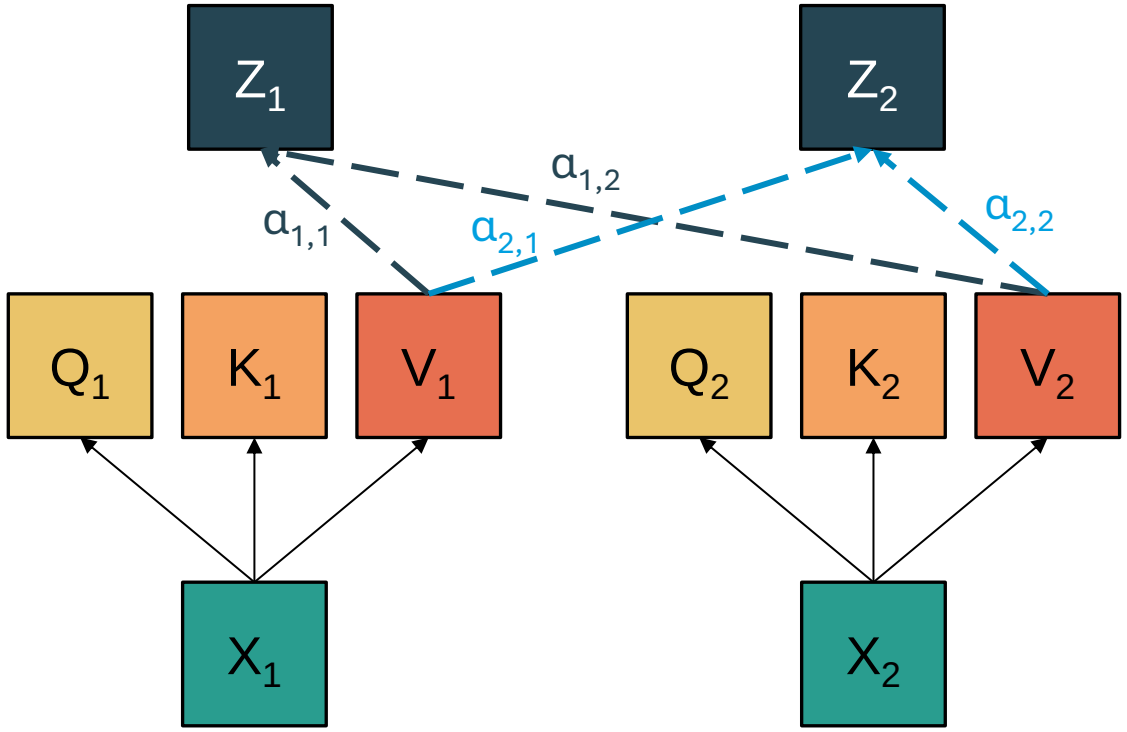
- Next, scores corresponding to each query vector are passed through a softmax layer to get softmax scores.
  - α_1 = Softmax($S_{1,1}$, $S_{1,2}$)
  - α_2 = Softmax($S_{2,1}$, $S_{2,2}$)

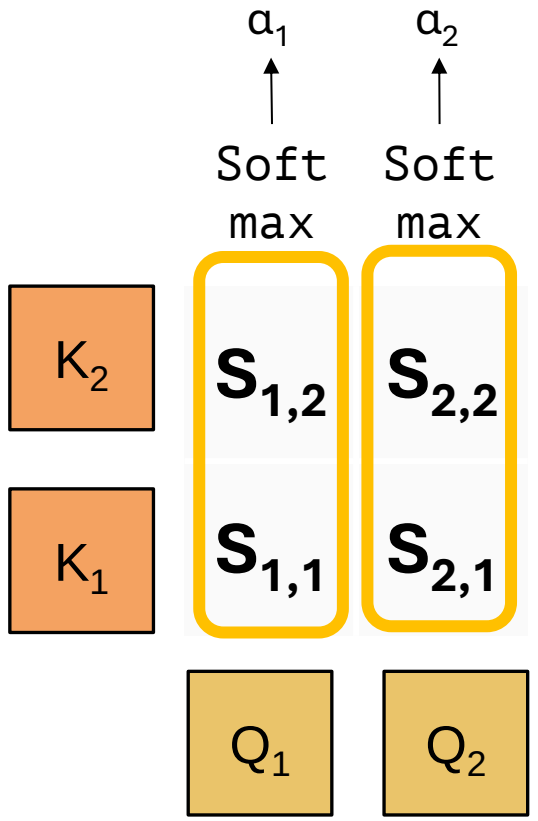




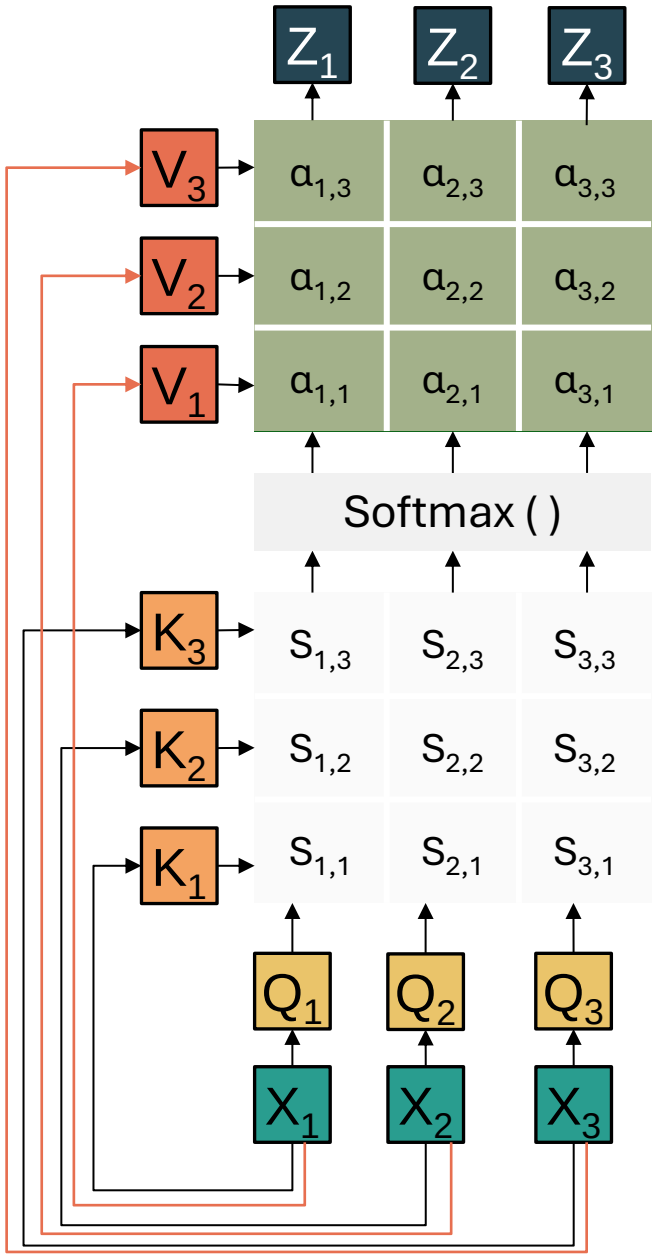
Self-Attention: Step 3: Weighted average of value vectors

- For every X_i, compute the weighted average of the value vectors using the softmax scores (corresponding to X_i) as weights.
  - $Z_1 = (\alpha_{1,1} * V_1) + (\alpha_{1,2} * V_2)$
  - $Z_2 = (\alpha_{2,1} * V_1) + (\alpha_{2,2} * V_2)$





Self-Attention: Overview



$$Z_1 = (\alpha_{1,1} * V_1) + (\alpha_{1,2} * V_2) + (\alpha_{1,3} * V_3)$$

$$Z_2 = (\alpha_{2,1} * V_1) + (\alpha_{2,2} * V_2) + (\alpha_{2,3} * V_3)$$

$$Z_3 = (\alpha_{3,1} * V_1) + (\alpha_{3,2} * V_2) + (\alpha_{3,3} * V_3)$$

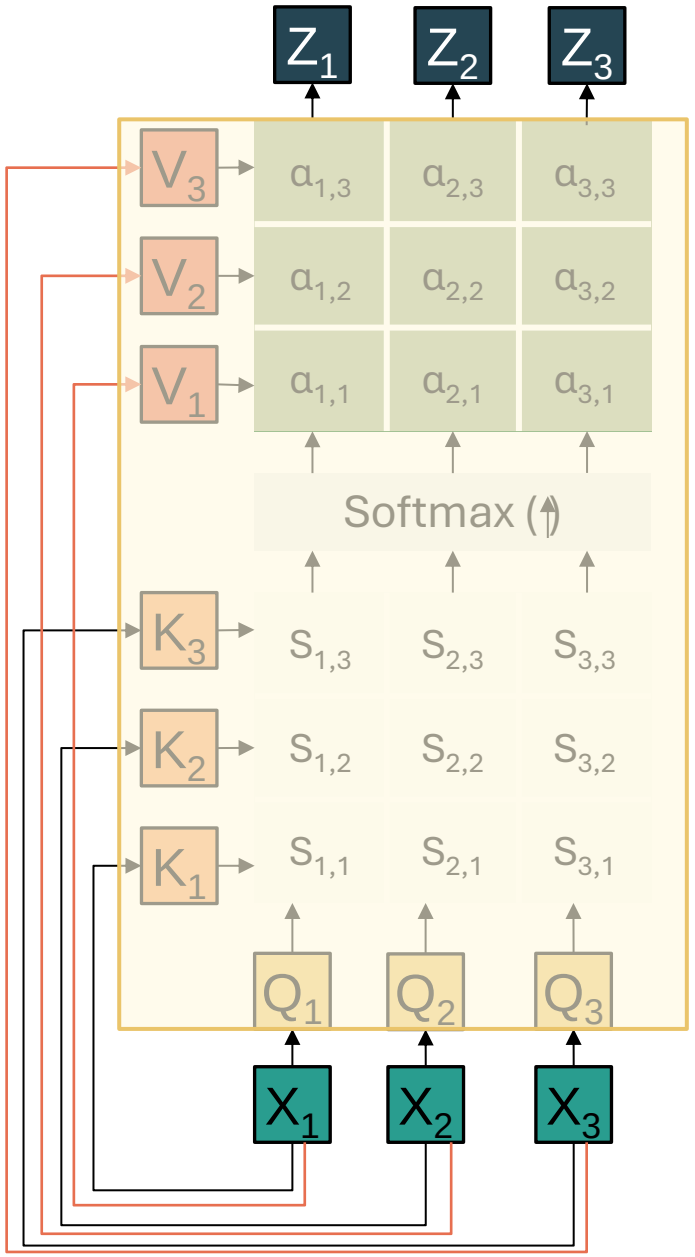
Query vectors: $Q = XW_Q$

Key vectors: $K = XW_K$

Value vectors: $V = XW_V$

Scores: $S_{i,j} = Q_i \bullet K_j / \sqrt{D}$

Self-Attention: Overview

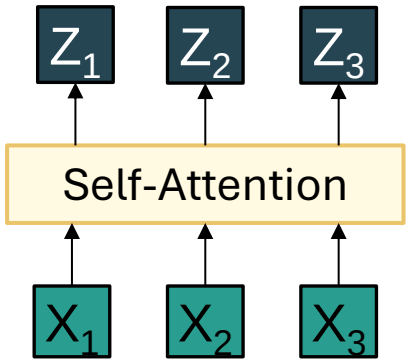


$Z_1 = (\alpha_{1,1} * V_1) + (\alpha_{1,2} * V_2) + (\alpha_{1,3} * V_3)$

$Z_2 = (\alpha_{2,1} * V_1) + (\alpha_{2,2} * V_2) + (\alpha_{2,3} * V_3)$

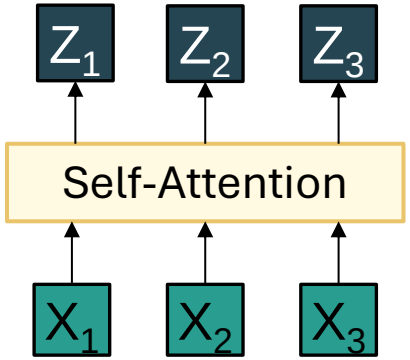
$Z_3 = (\alpha_{3,1} * V_1) + (\alpha_{3,2} * V_2) + (\alpha_{3,3} * V_3)$

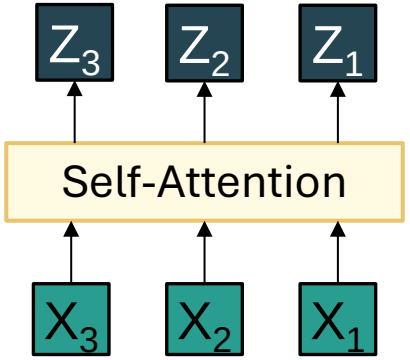
Query vectors: $Q = XW_Q$
Key vectors: $K = XW_K$
Value vectors: $V = XW_V$
Scores: $S_{i,j} = Q_i \bullet K_j / \sqrt{D}$

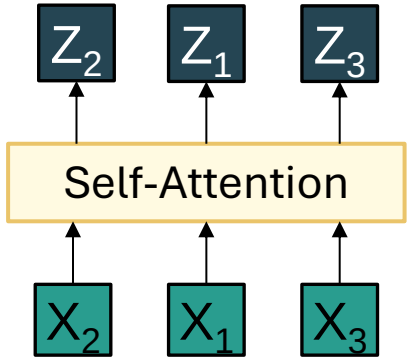


$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Self-Attention: Permutation equivariance





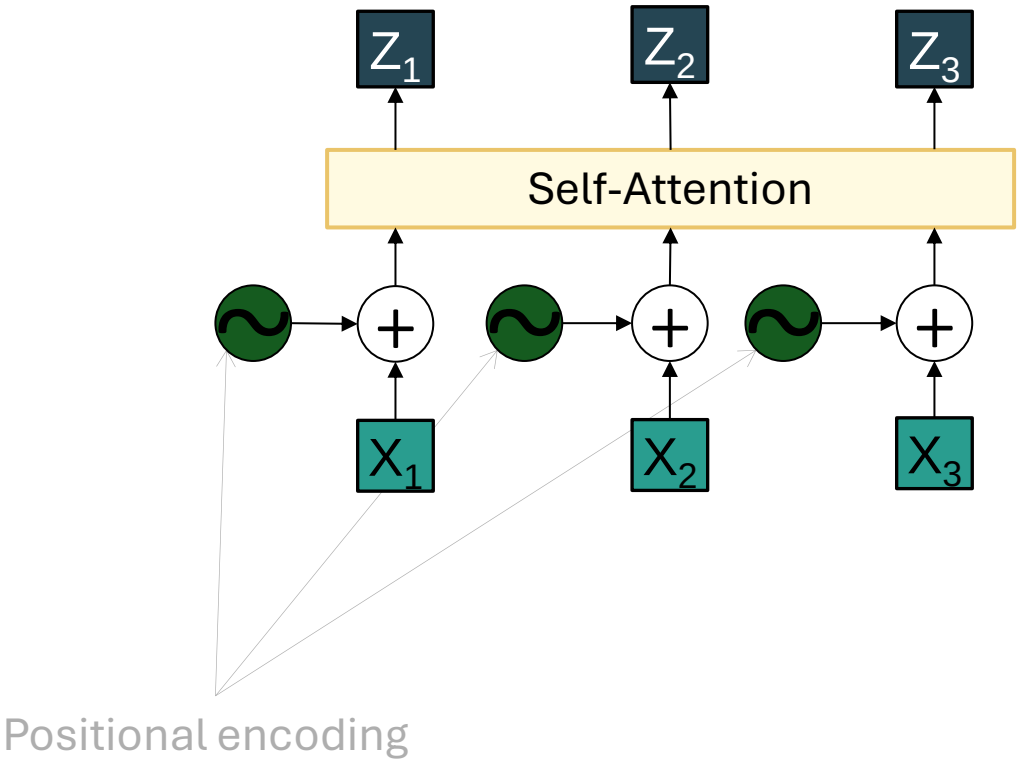


- Self-attention layer is permutation equivariant.
- Self-attention layer doesn't care about the order of the input sequence.

Problem: How to encode the order of input sequence when it matters like in language or images?

Positional encoding

- There are multiple ways of encoding position.
  - Absolute/fixed positional encoding.
  - Relative positional encoding.
  - A hybrid of relative and absolute positional encoding.
  - Learned positional encoding.
  - etc.
- Positional encodings are vectors with the same dimension as the input vectors.
- Positional encodings are usually added to the input vectors.



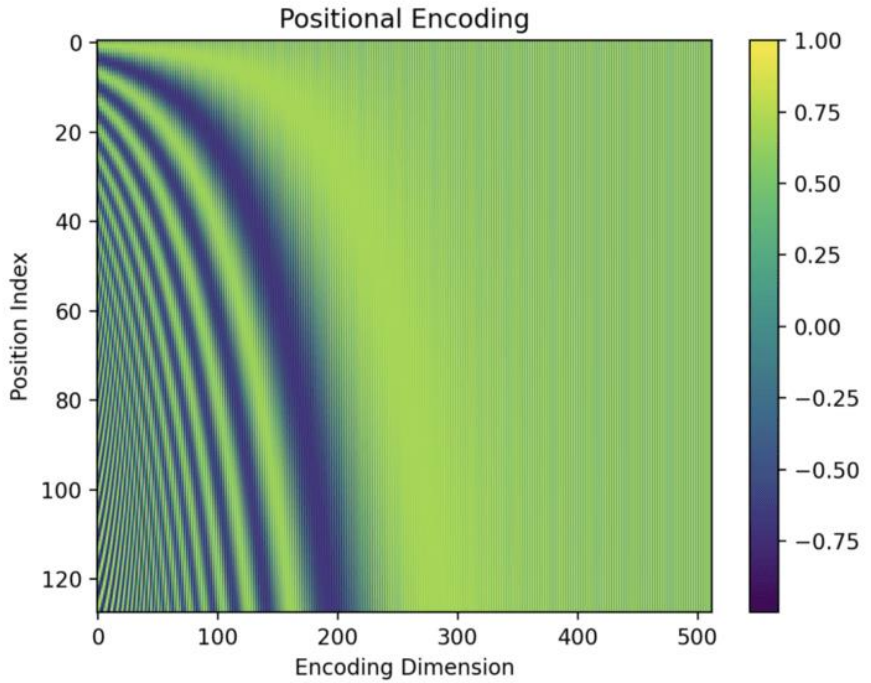
Positional encoding: Absolute positional encoding

- Absolute positional encoding at position ***pos*** for dimension **i = 0…(d/2 - 1)**:

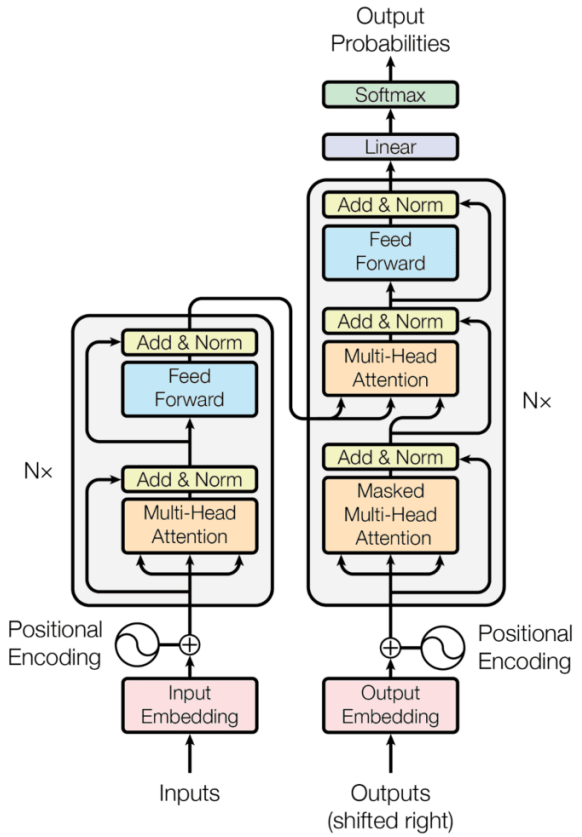
$$PE(pos, 2i) = \sin(pos/10000^{2i/d})$$
$$PE(pos, 2i+1) = \cos(pos/10000^{2i/d})$$

- A 512-dimensional encoding for ***i*** in the sequence:

$\sin(i/10000^{0/512})$	$cos(i/10000^{0/512})$	$\sin(i/10000^{2/512})$	$\cos(i/10000^{2/512})$	…	$\sin(i/10000^{510/512})$	$\cos(i/10000^{510/512})$



Transformers



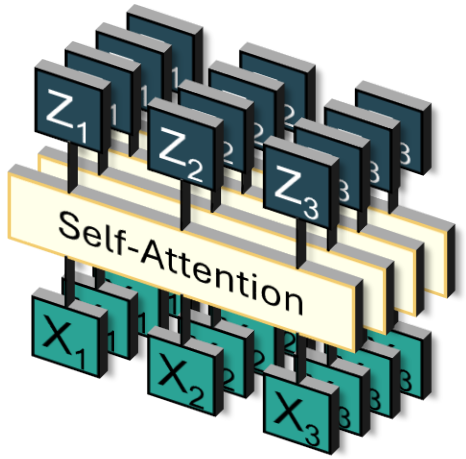
Source: Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017). link

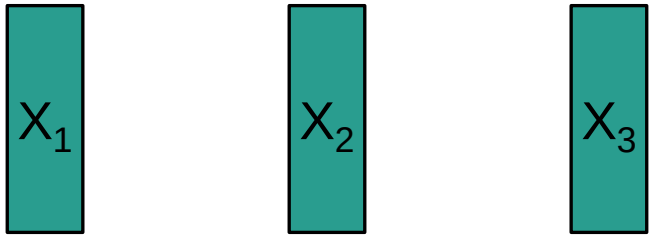
Outline

- Attention Mechanism
- Self-attention
- **Multi-head self-attention**
- Transformer encoder block
- Transformer decoder block
- Vision transformer (ViT)
- Swin transformer
- Suggested resources
- Mandatory Assignment 2

Multi-head Self-Attention

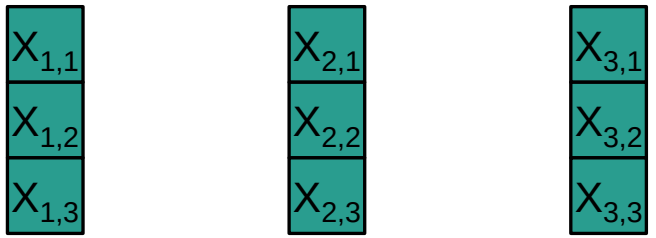
- Use H independent “attention heads” in parallel.
  - Attention heads do NOT share parameters.
- Each Input embedding is split into H parts.
- Each part is then processed by an attention head.



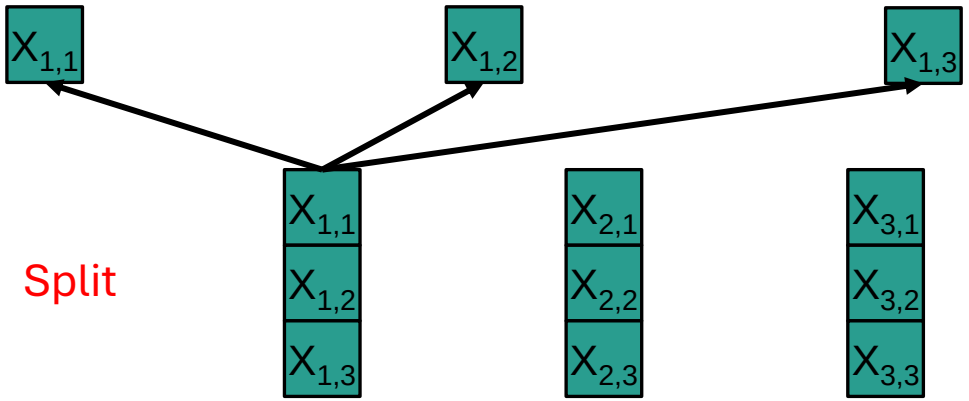


Multi-head Self-Attention

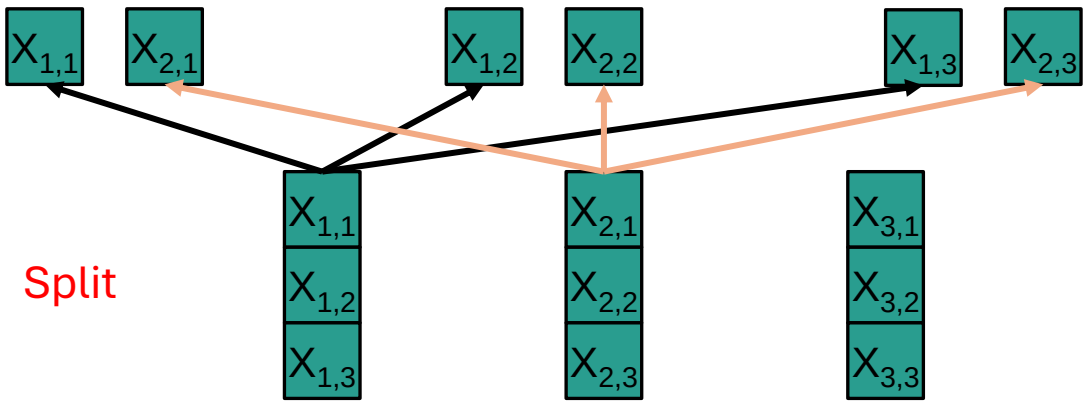
Split



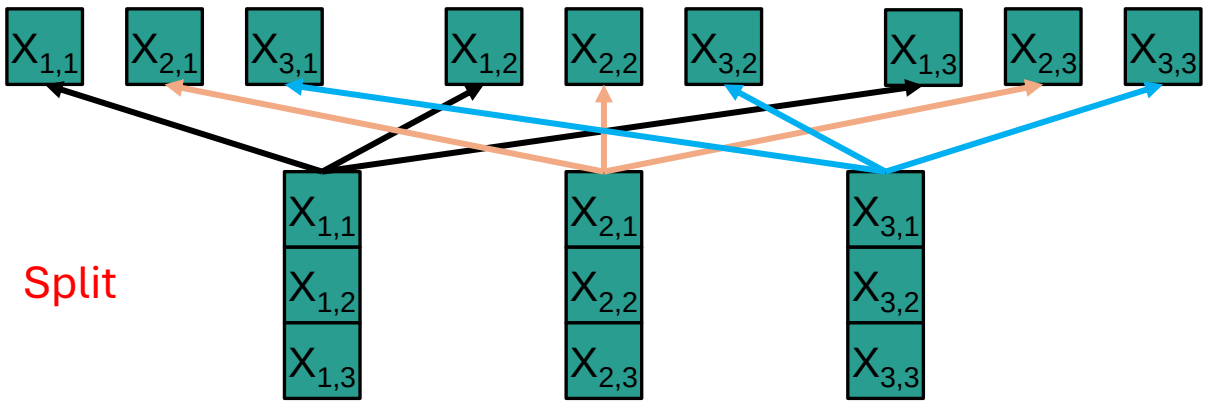
Multi-head Self-Attention



Multi-head Self-Attention

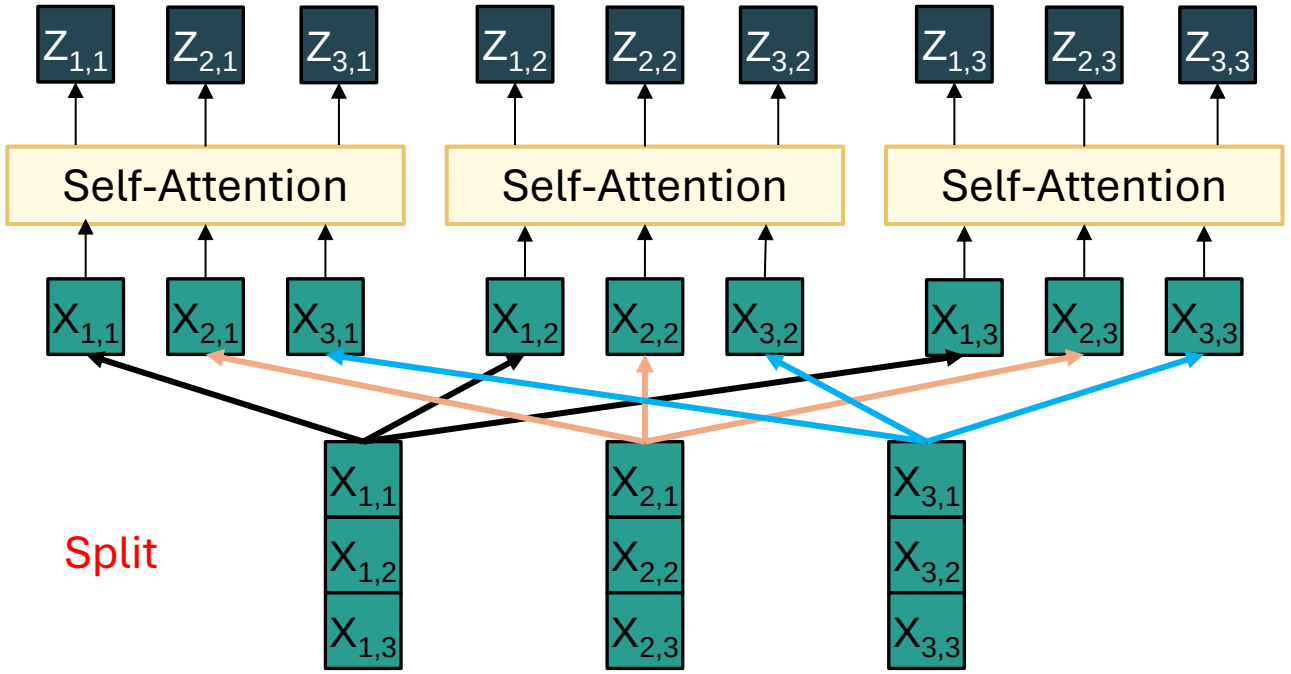


Multi-head Self-Attention

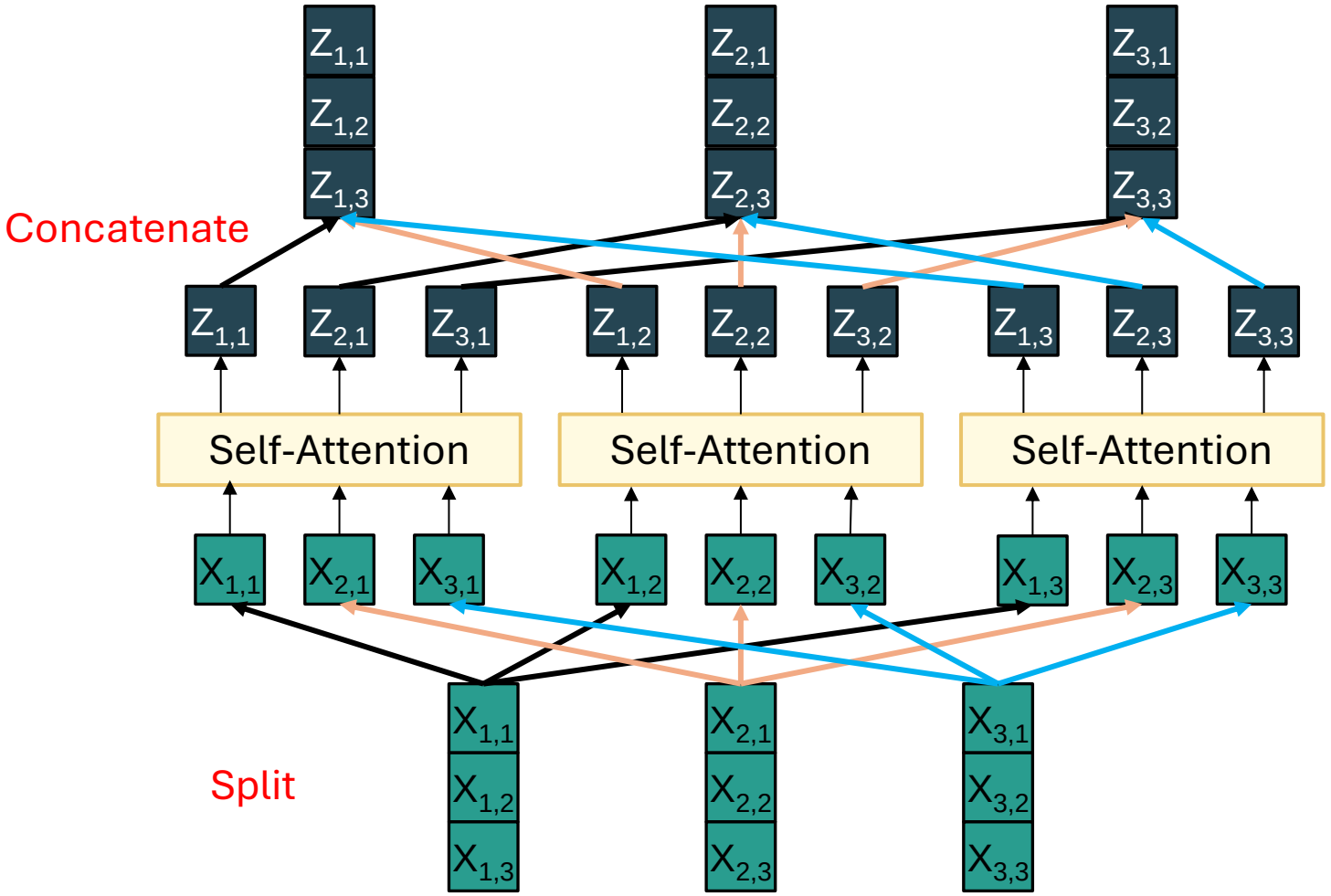


Multi-head Self-Attention

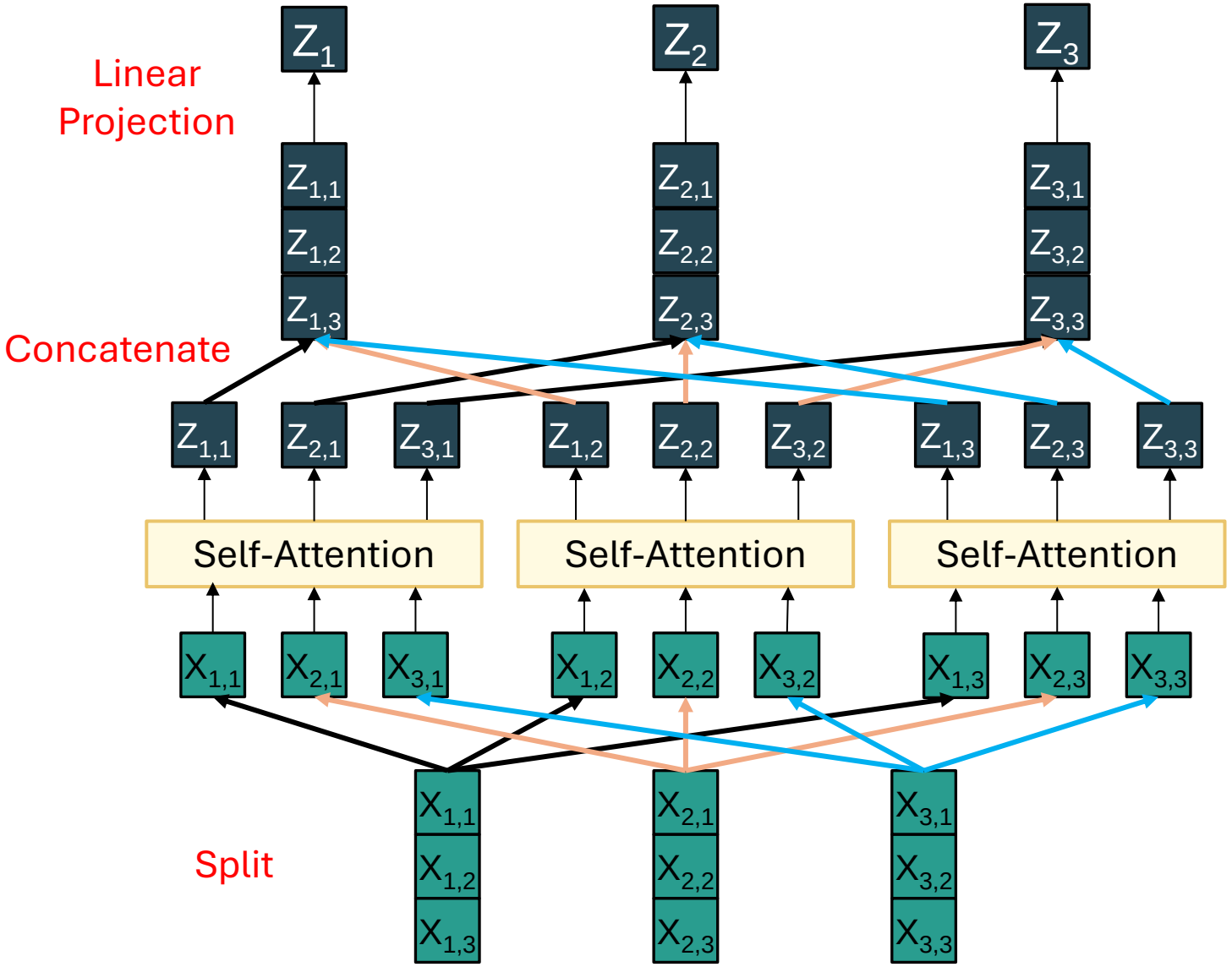
Run self-attention in parallel on each set of input vectors using a different set of parameters for each self-attention head.



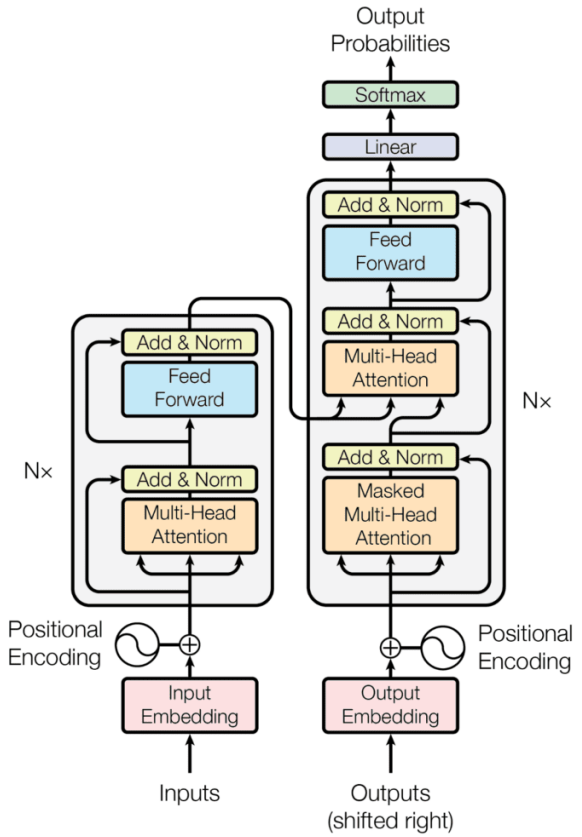
Multi-head Self-Attention



Multi-head Self-Attention

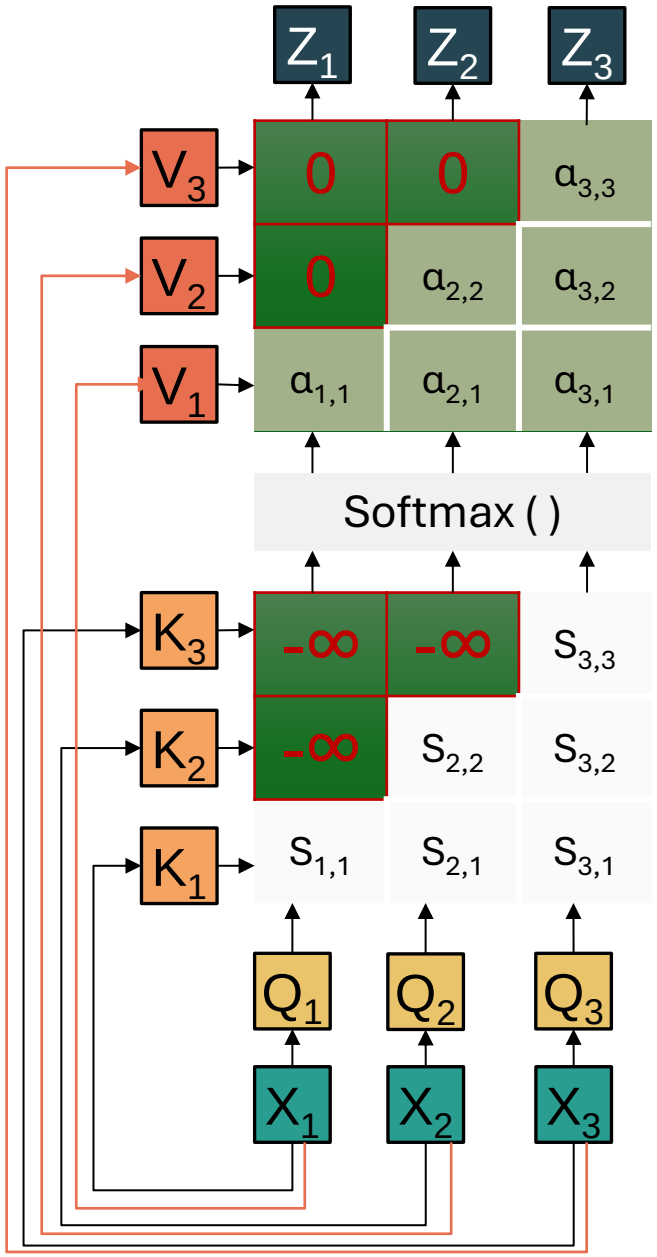


Transformers



Source: Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017). link

Masked Self-Attention



$Z_1 = (\alpha_{1,1} * V_1)$

$Z_2 = (\alpha_{2,1} * V_1) + (\alpha_{2,2} * V_2)$

$Z_3 = (\alpha_{3,1} * V_1) + (\alpha_{3,2} * V_2) + (\alpha_{3,3} * V_3)$

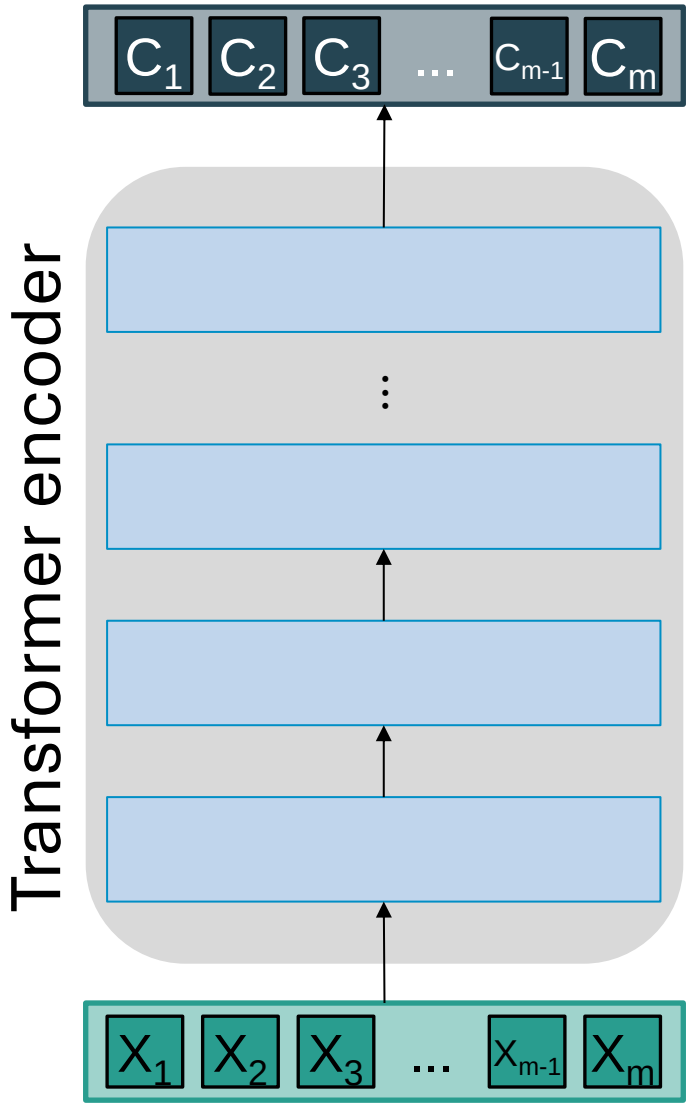
Query vectors: $Q = XW_Q$
Key vectors: $K = XW_K$
Value vectors: $V = XW_V$
Scores: $S_{i,j} = K_i \bullet Q_j / \sqrt{D}$

- Don't let vectors look ahead in the sequence at future vectors.
- Manually set scores to -infinity.
- Especially relevant for Language Modelling

Outline

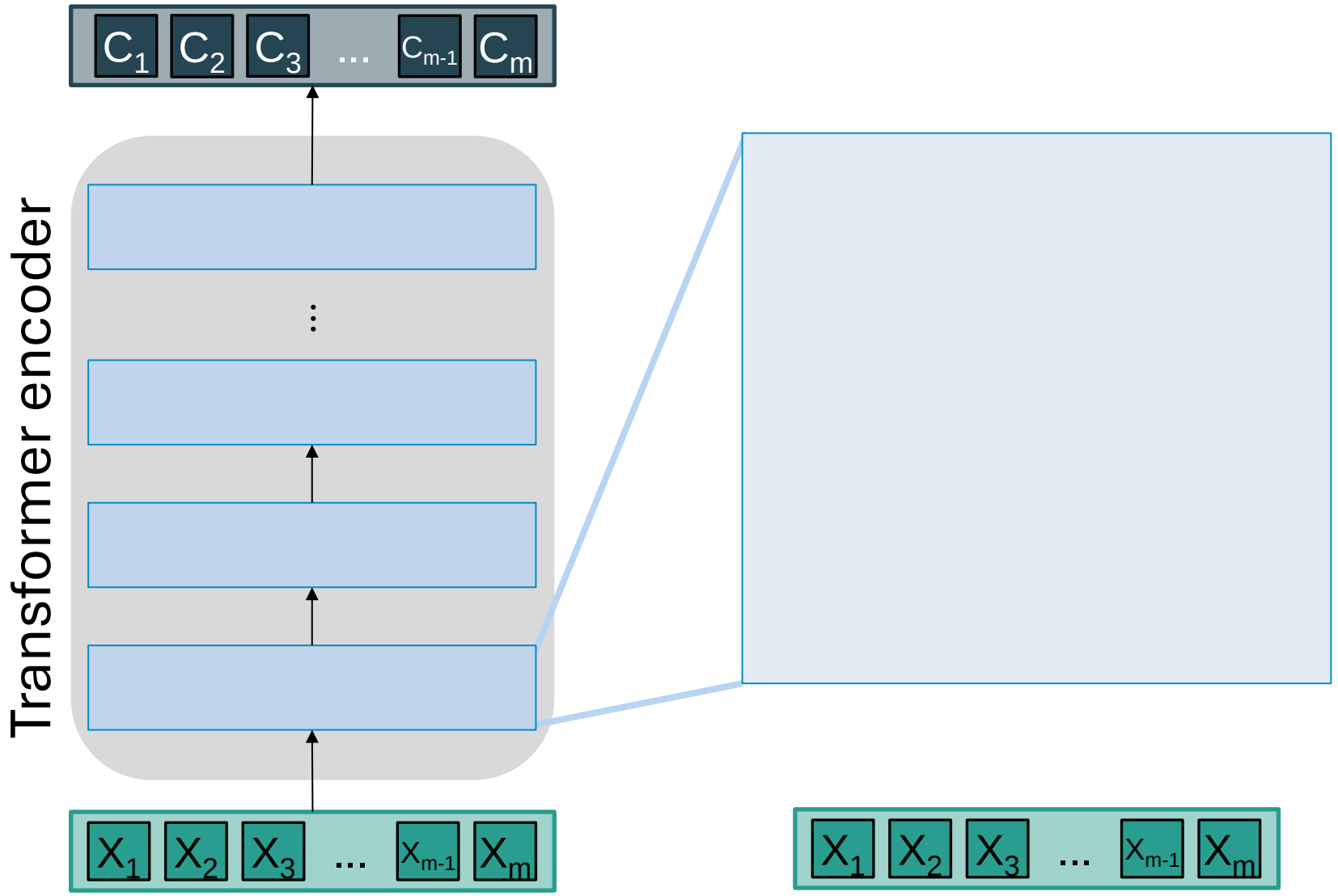
- Self-attention
- Multi-head self-attention
- **Transformer encoder block**
- Transformer decoder block
- Vision transformer (ViT)
- Swin transformer
- Suggested resources
- Mandatory Assignment 2

Transformer encoder block



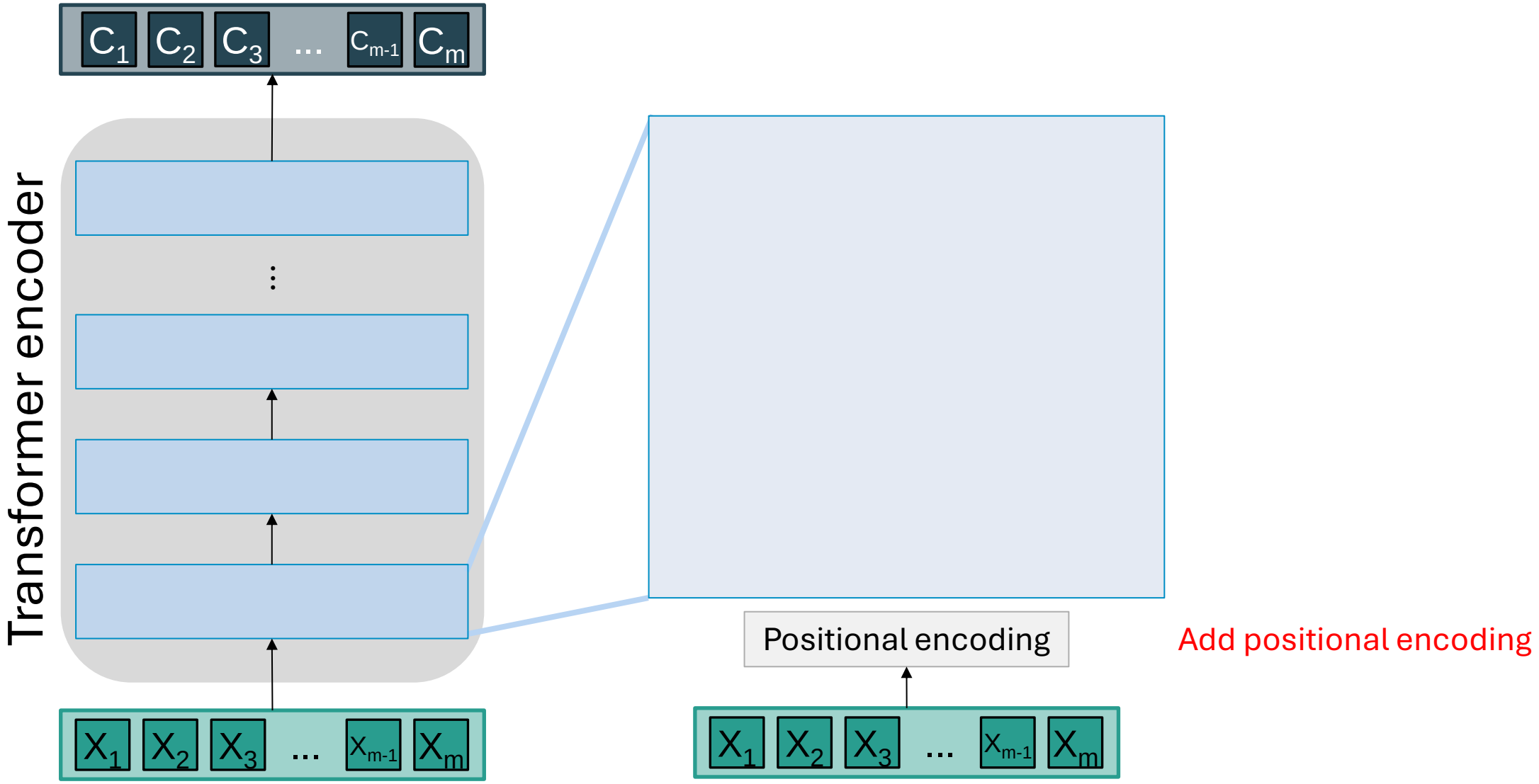
Transformer encoder is made up of a series of N encoder blocks.
In the original model (Vaswani et al.) N=6

Transformer encoder block

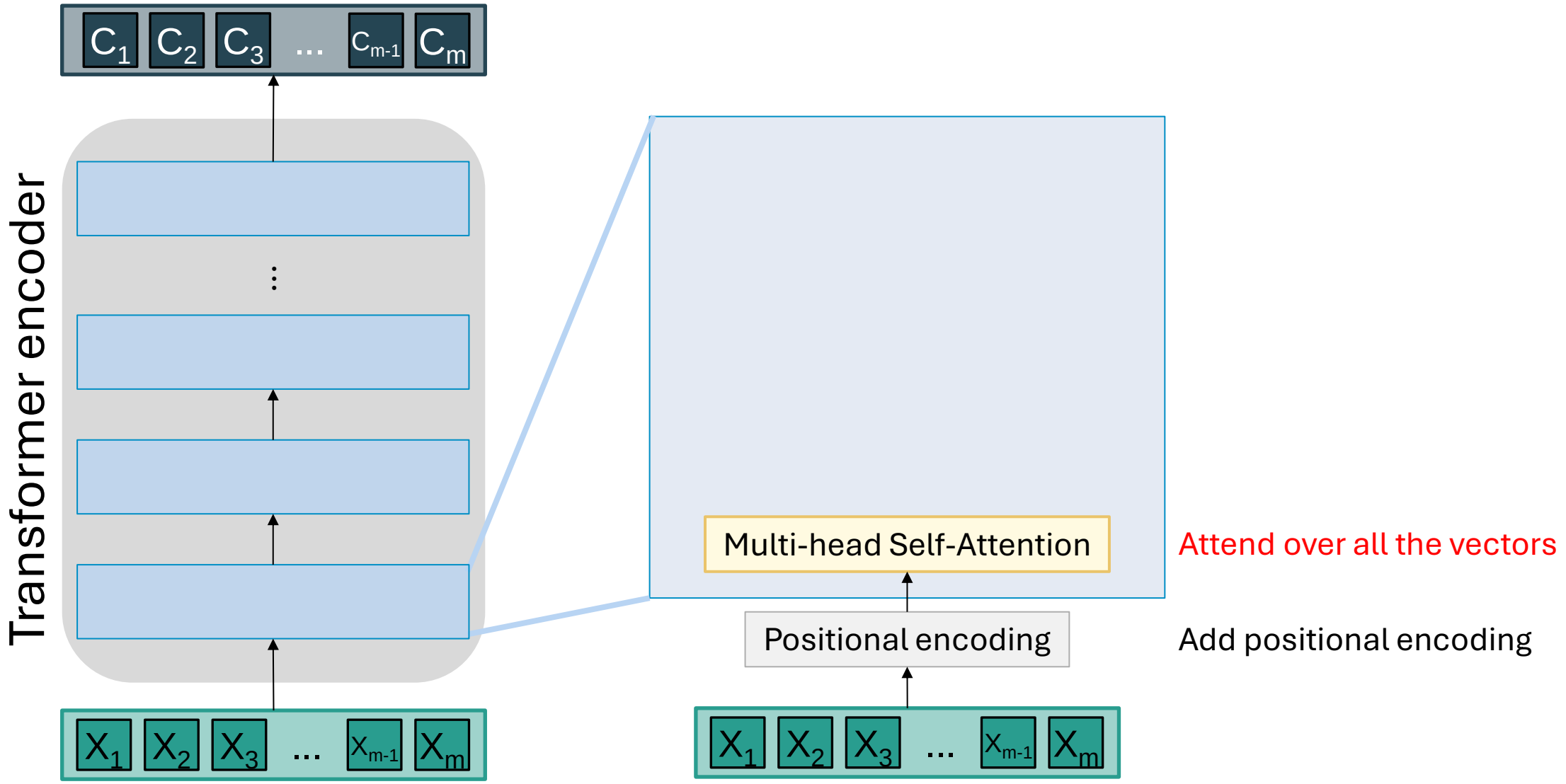


Let's look at the architecture of an encoder block.

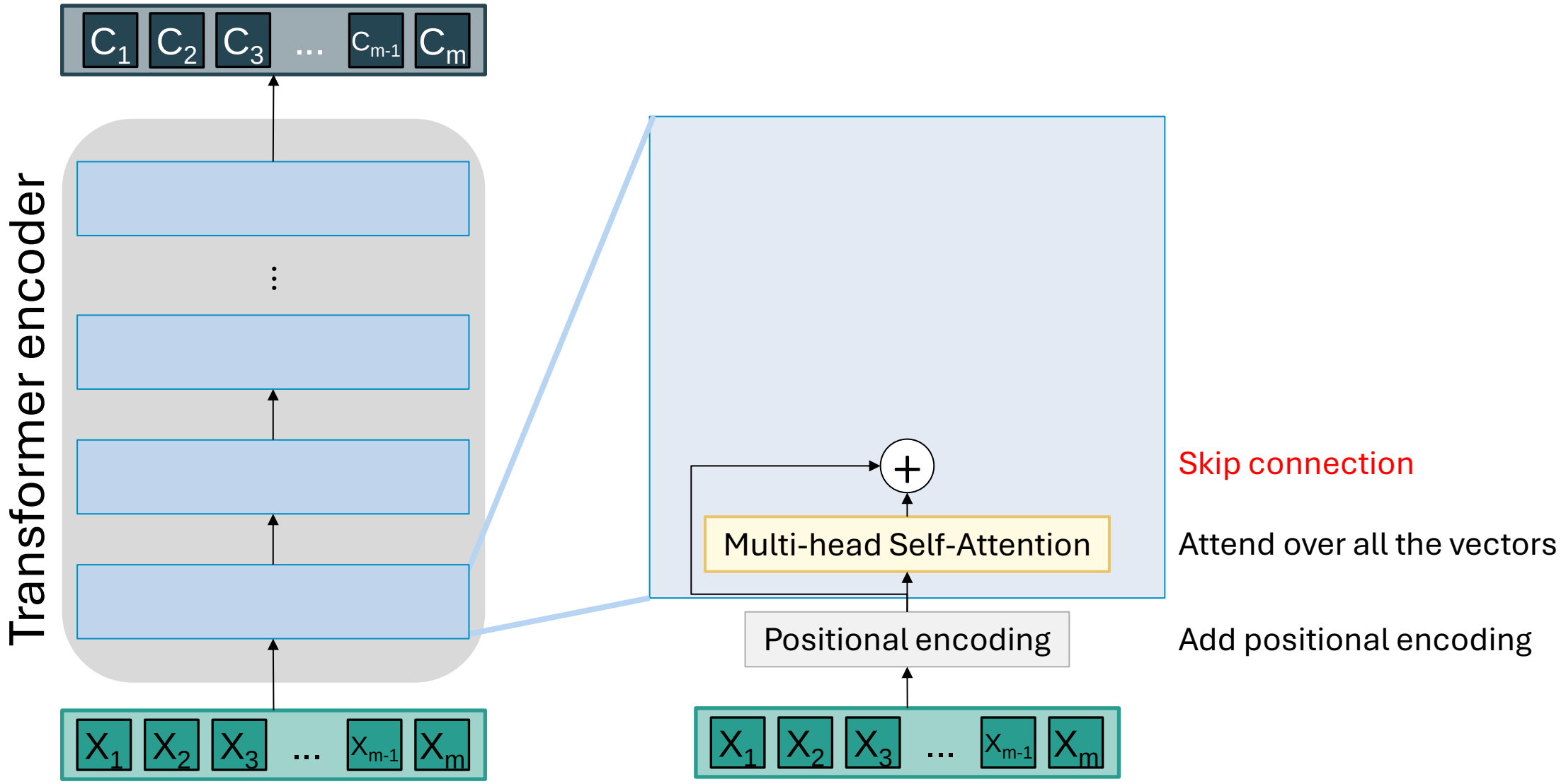
Transformer encoder block



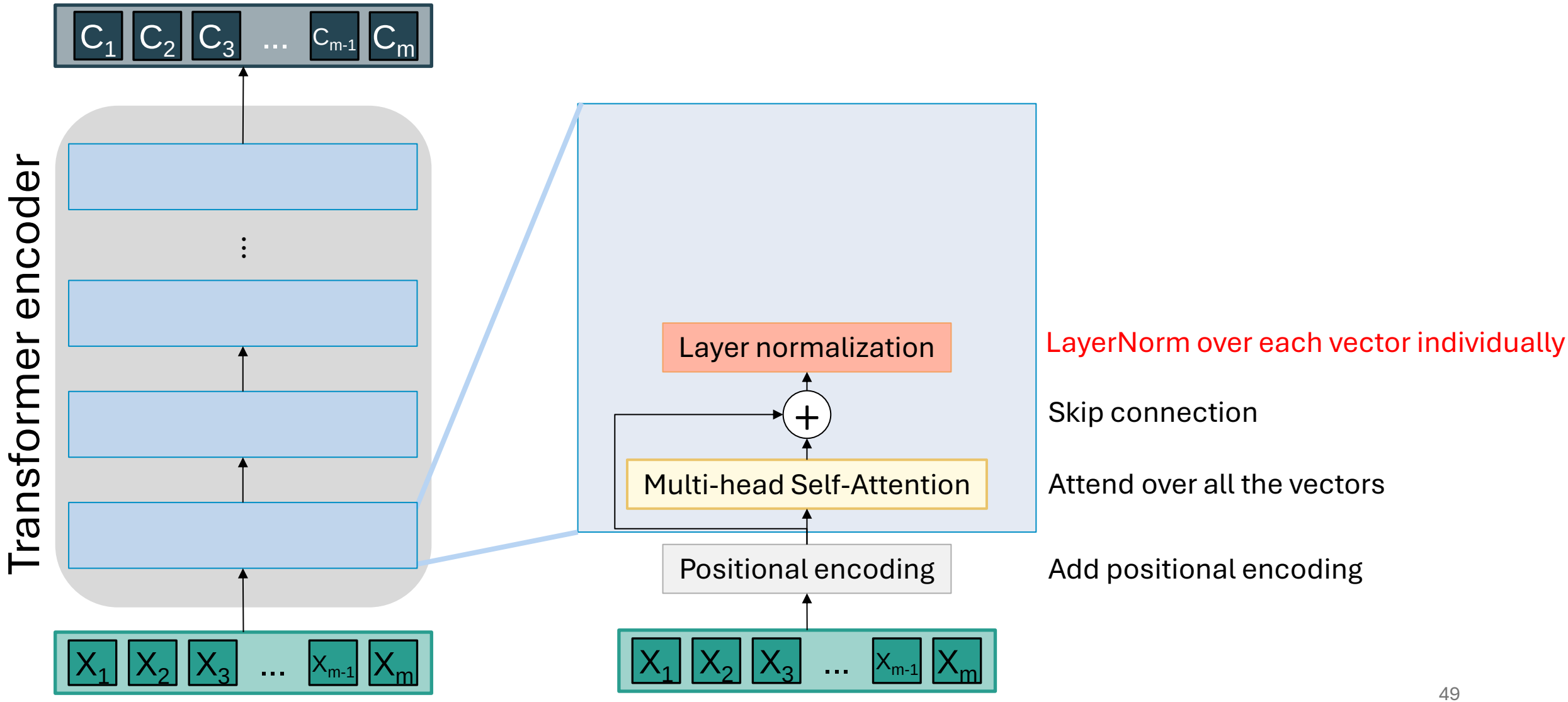
Transformer encoder block



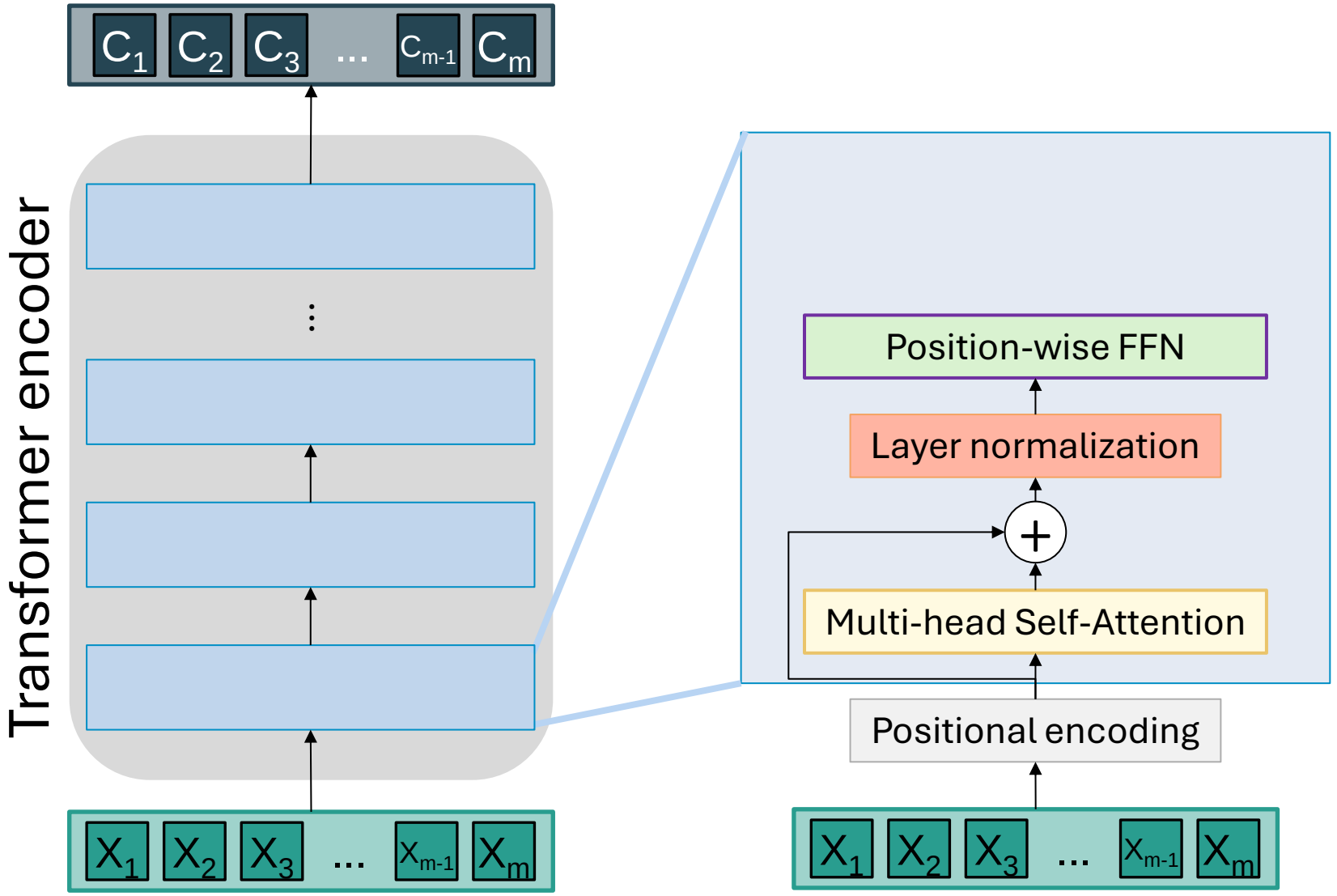
Transformer encoder block



Transformer encoder block



Transformer encoder block



2-layer Feed Forward Network over each vector individually (shared W)

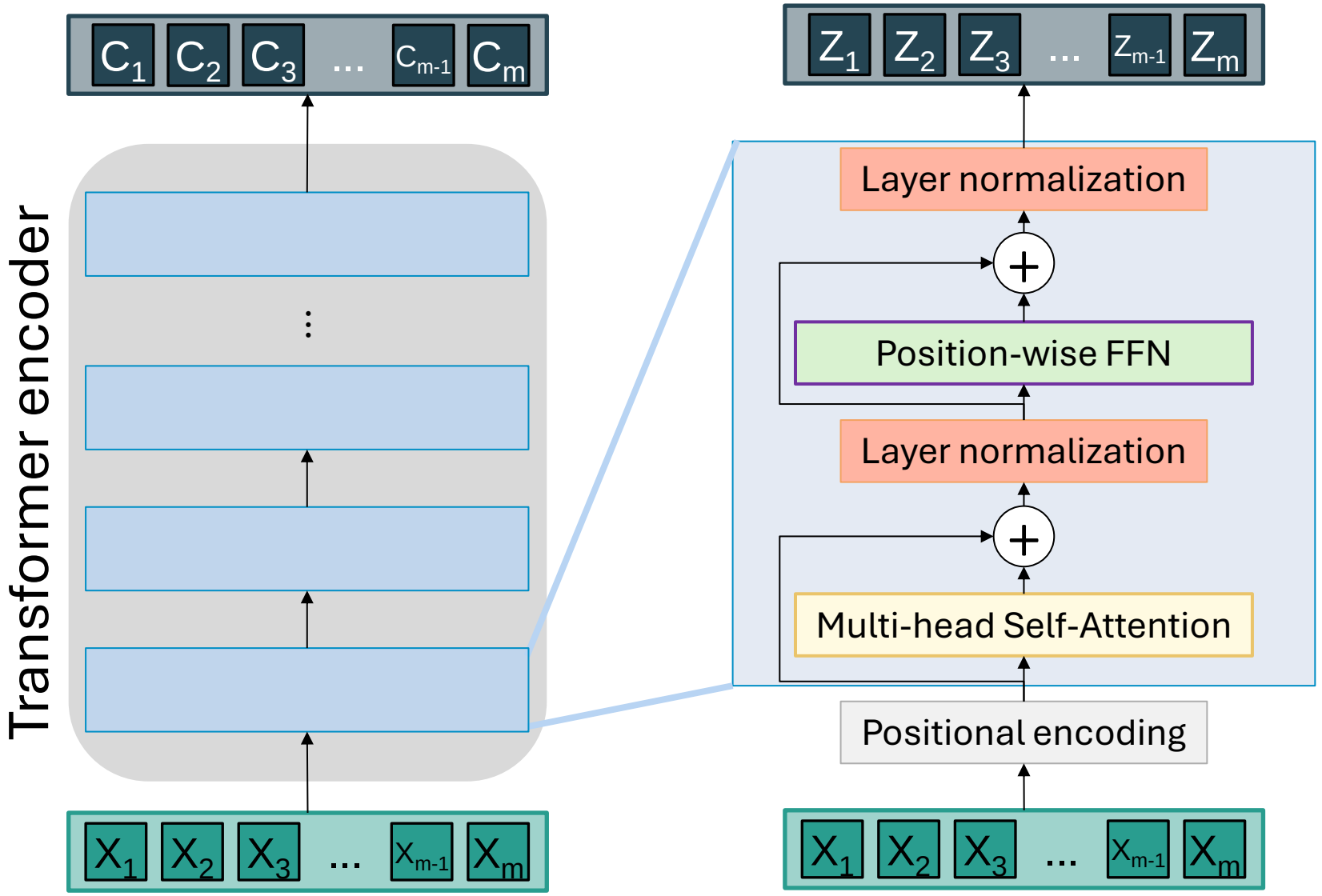
LayerNorm over each vector individually

Skip connection

Attend over all the vectors

Add positional encoding

Transformer encoder block



2-layer Feed Forward Network over each vector individually (shared W)

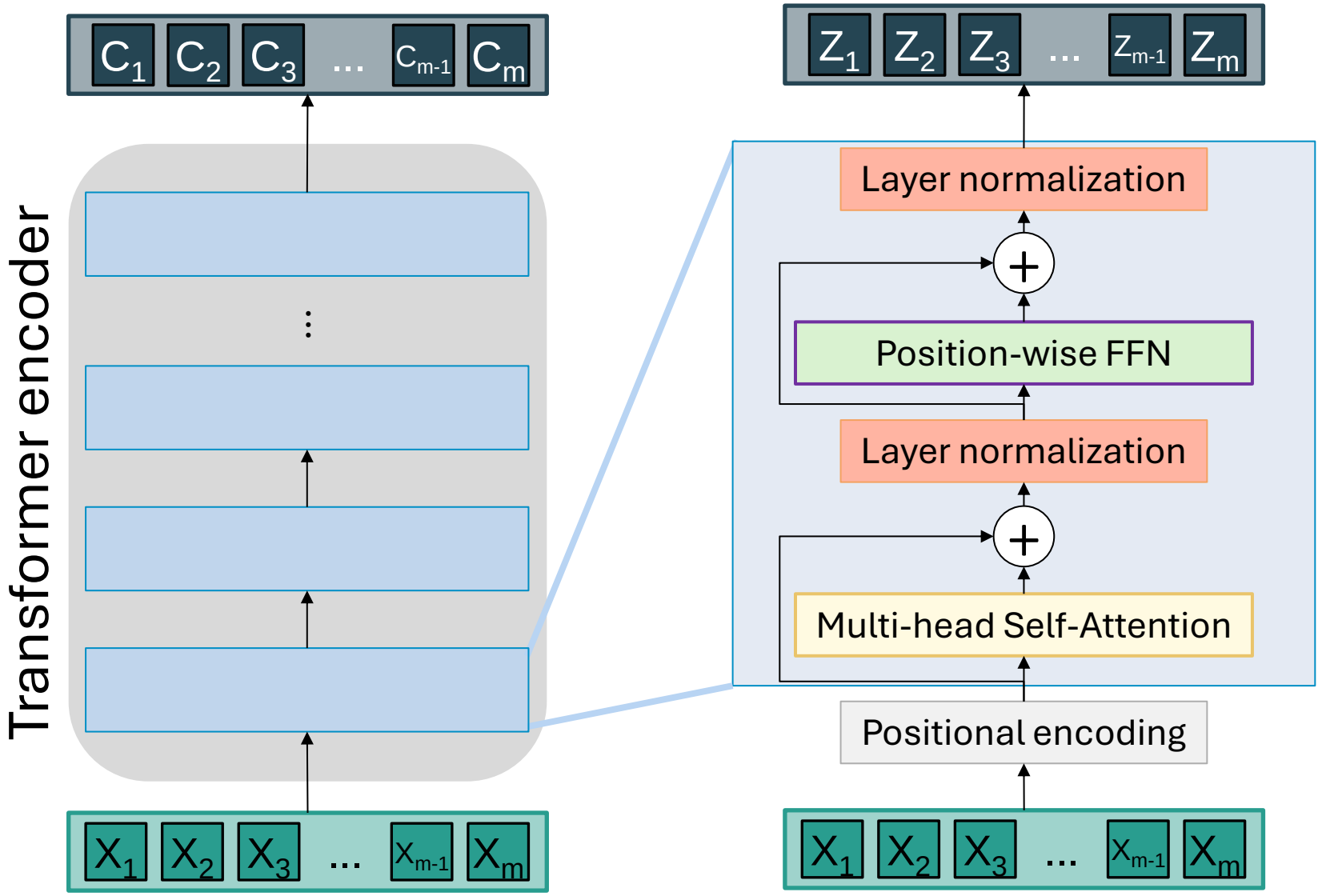
LayerNorm over each vector individually

Skip connection

Attend over all the vectors

Add positional encoding

Transformer encoder block



Transformer Encoder Block:

Input: Set of vectors **X**
Output: Set of vectors **Z**

Self-attention is the only interaction between vectors.

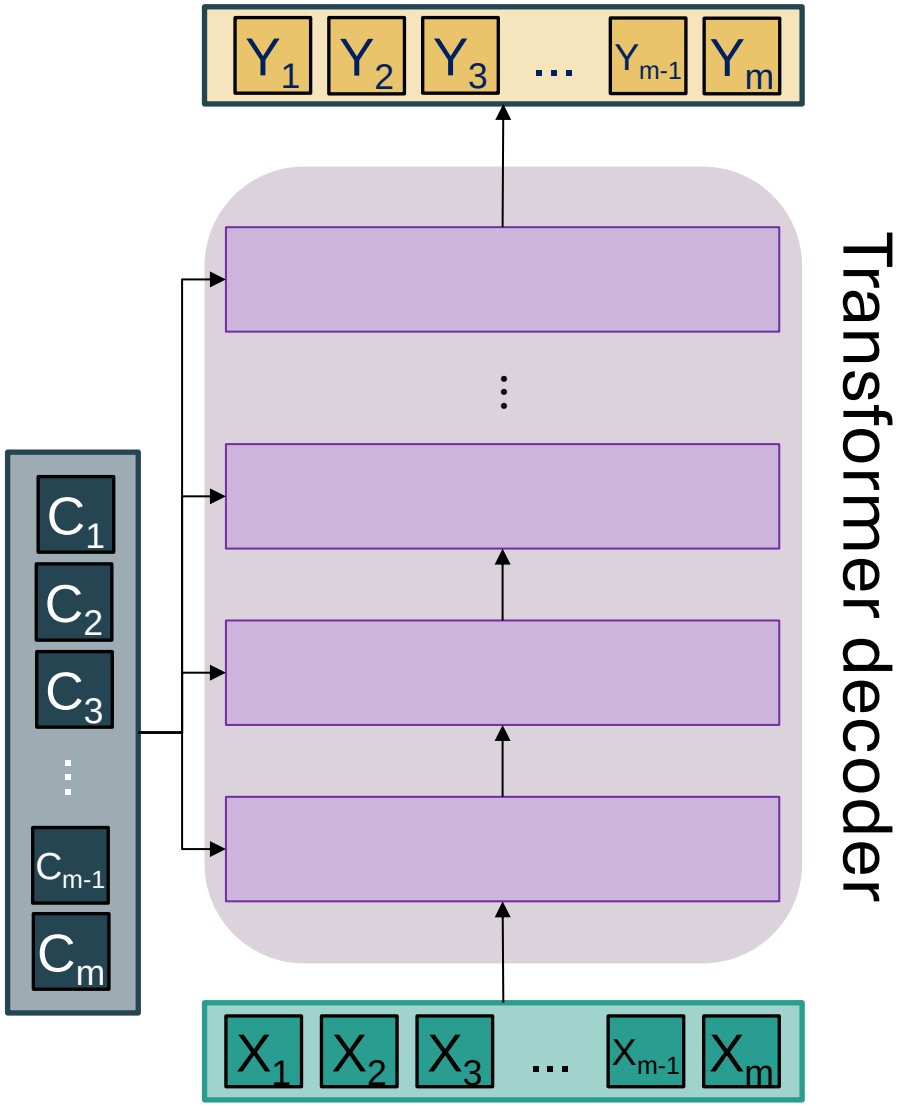
LayerNorm and FFN operate independently per vector.

Highly parallelisable but has high memory usage.

Outline

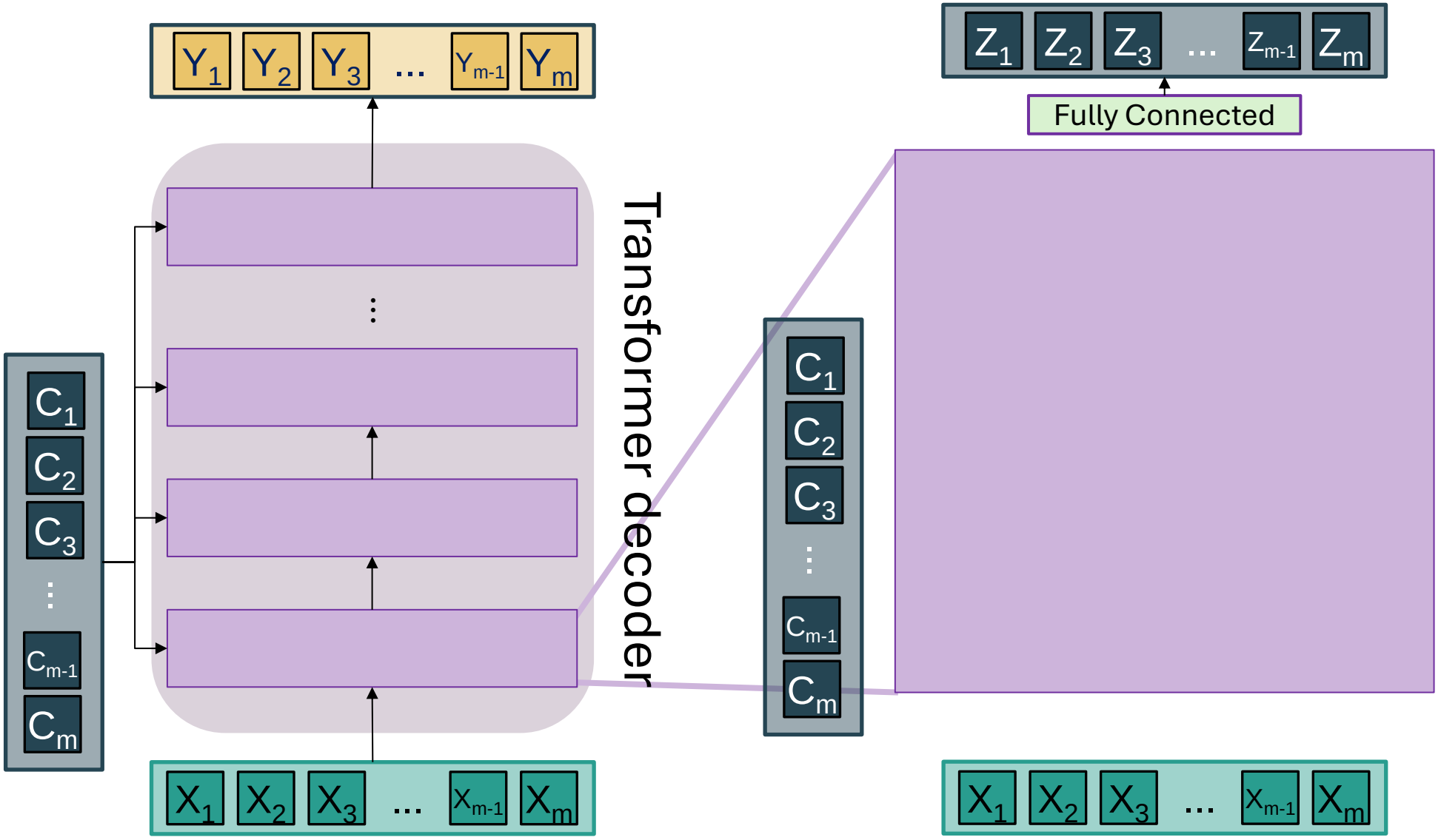
- Attention Mechanism
- Self-attention
- Multi-head self-attention
- Transformer encoder block
- **Transformer decoder block**
- Vision transformer (ViT)
- Swin transformer
- Suggested resources
- Mandatory Assignment 2

Transformer decoder block



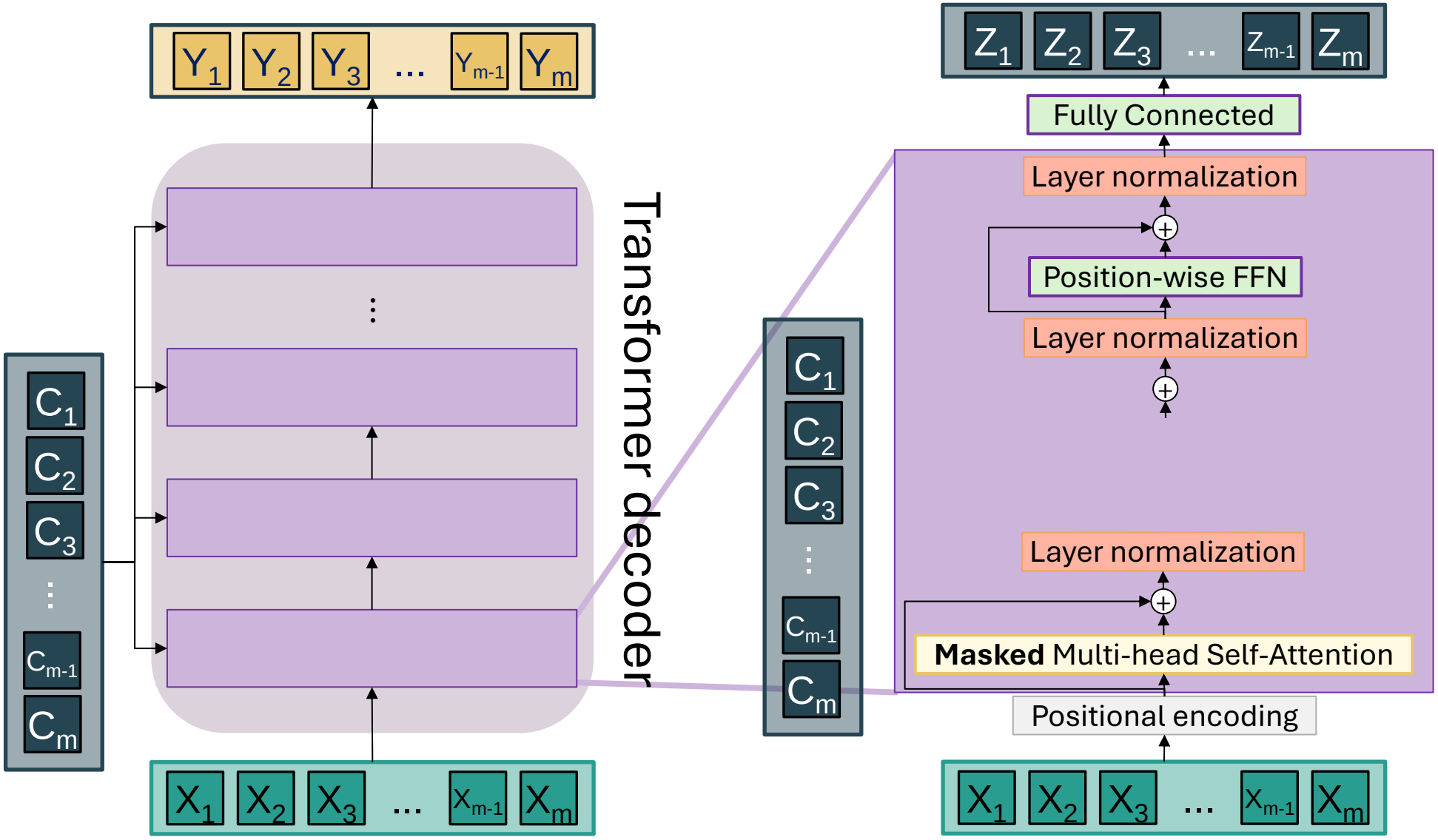
Transformer decoder is made up of a series of N decoder blocks.
In the original model (Vaswani et al.) N=6

Transformer decoder block



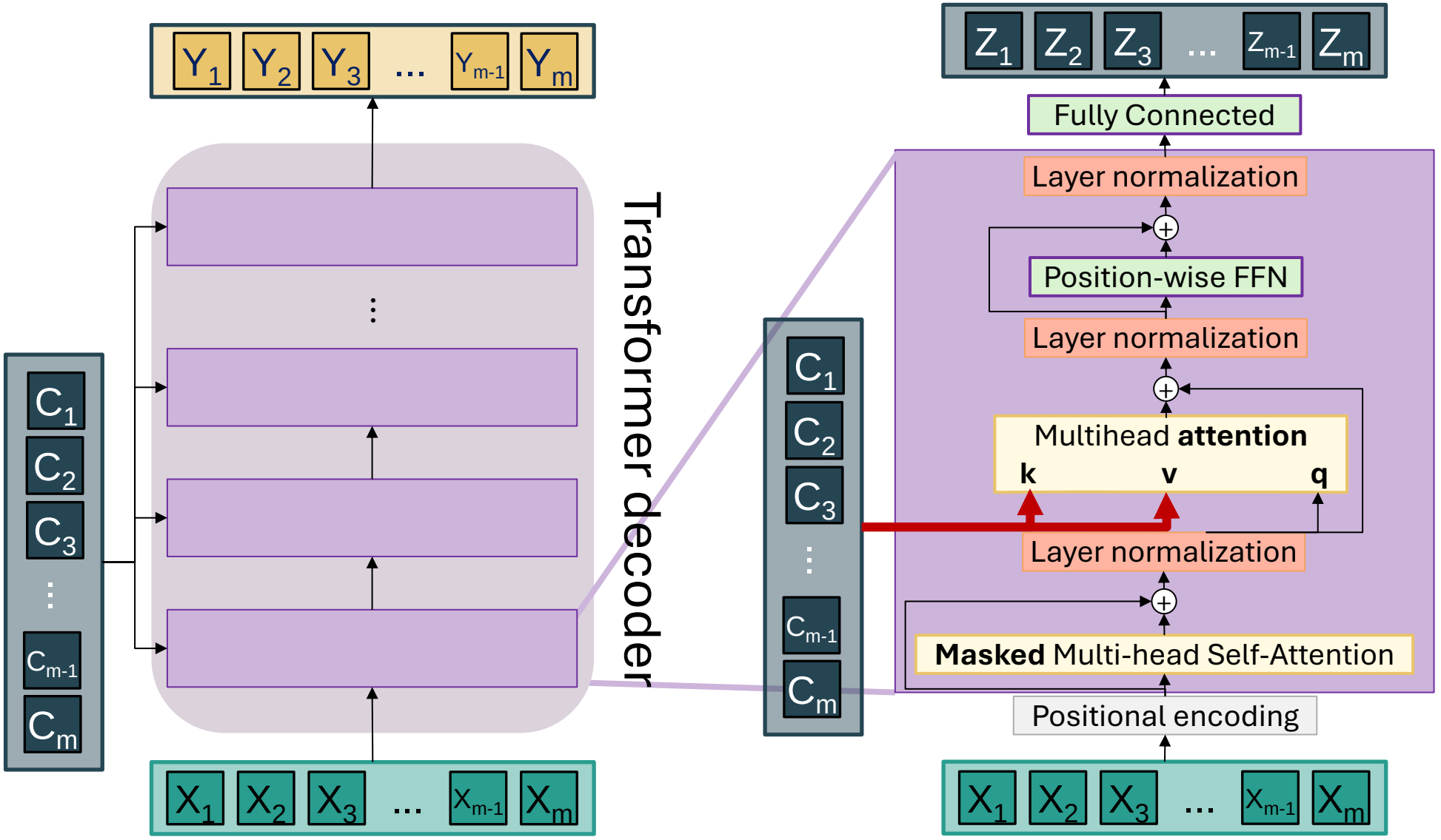
Let's look at the architecture of a decoder block.

Transformer decoder block



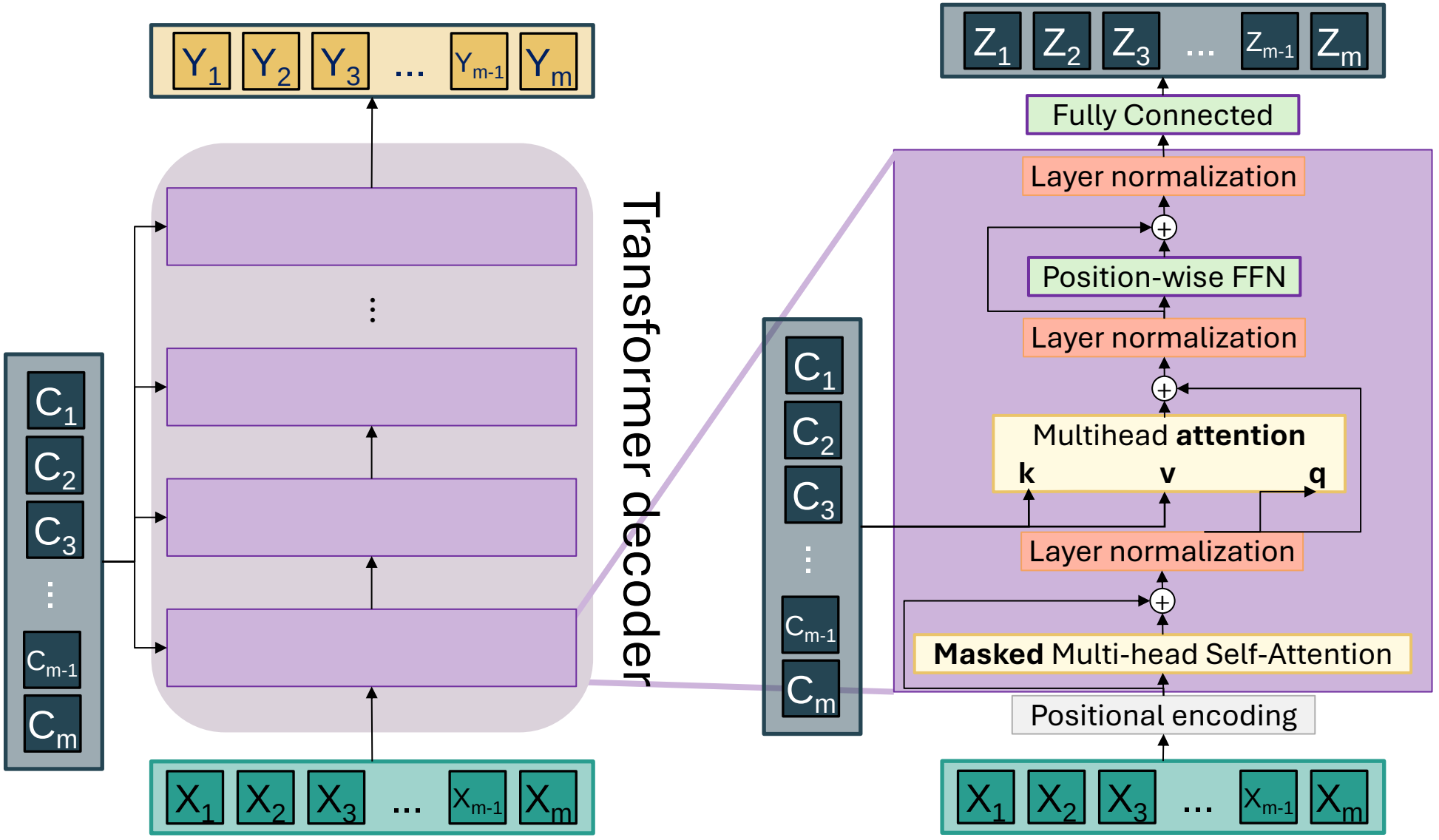
Most of the block is the same as the encoder block.

Transformer decoder block



Multi-head attention block attends over encoder's outputs.

Transformer decoder block



Transformer Decoder Block:

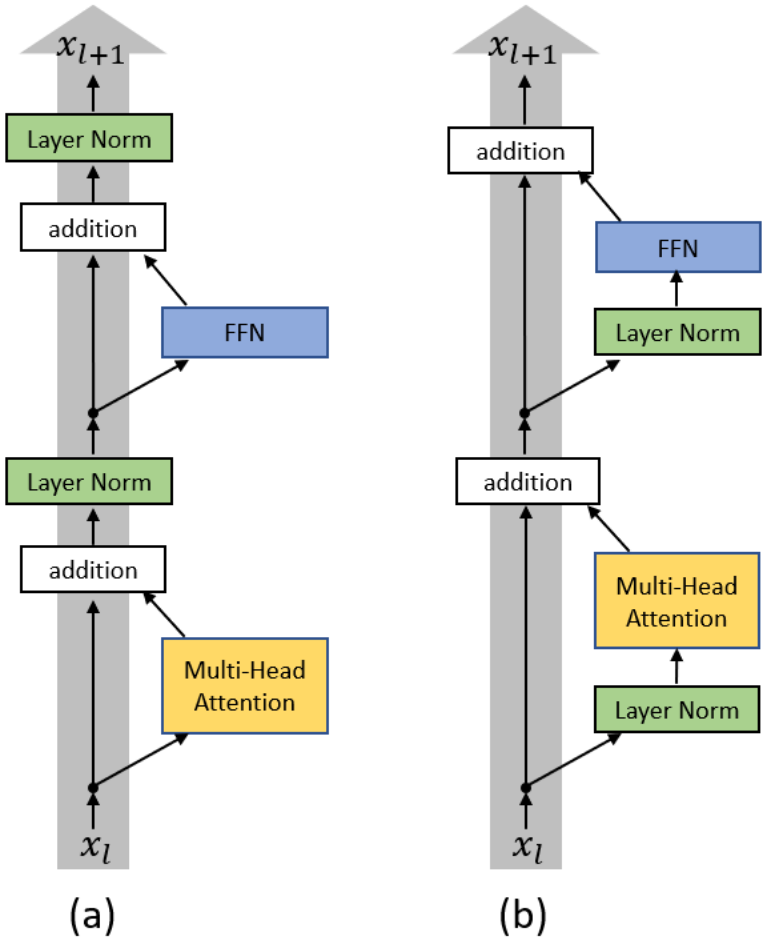
Input: Set of vectors **X** and set of context vectors **C**
Output: Set of vectors **Z**

Masked Self-attention interacts only with past inputs.

Multi-head attention is **NOT** self-attention. It attends over encoder's outputs.

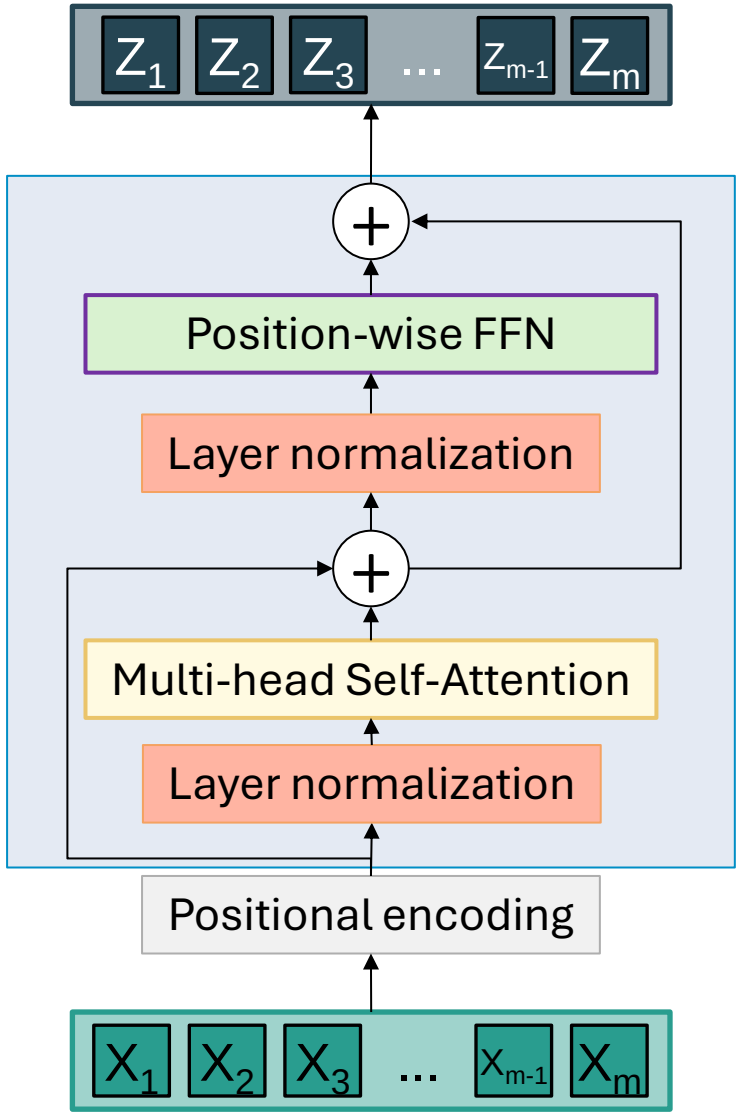
Highly parallelisable but has high memory usage.

Transformer block: Pre-Norm Transformer



(a) Post-norm transformer (b) Pre-norm transformer

Image Source: https://arxiv.org/pdf/2002.04745.pdf



Pre-Norm Transformer:

Layer normalization is inside residual (skip) connections.

Gives more stable training.

Outline

- Attention Mechanism
- Self-attention
- Multi-head self-attention
- Transformer encoder block
- Transformer decoder block
- **Vision transformer (ViT)**
- Swin transformer
- Suggested resources
- Mandatory Assignment 2

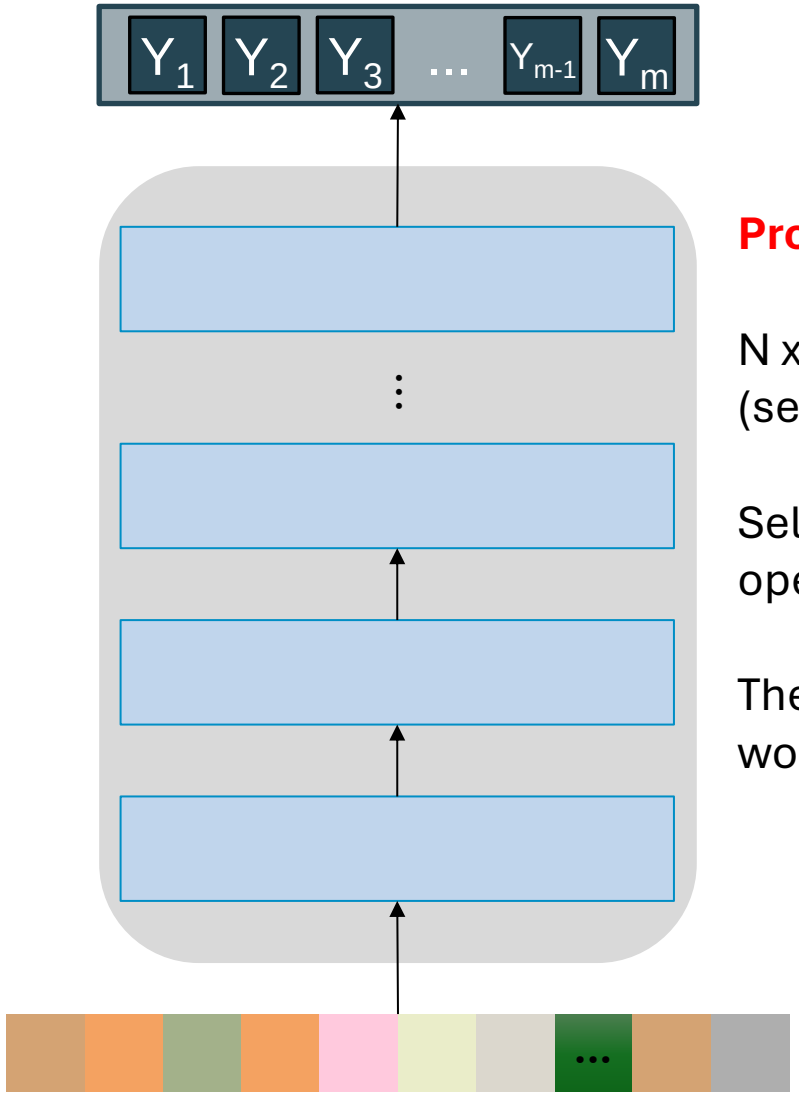
How to use transformers for images?

Idea: Treat an image as a sequence of pixels and use the standard transformer



Photo by Josh Hild on Unsplash

Flatten and feed as input



Problem: Memory usage!

N x N image has N^2 pixels (sequence length).

Self-attention is an $O(N^2)$ operation.

Therefore, self-attention would need N^4 elements!

Idea: Use standard transformer on patches



Photo by Josh Hild on Unsplash

Idea: Use standard transformer on patches

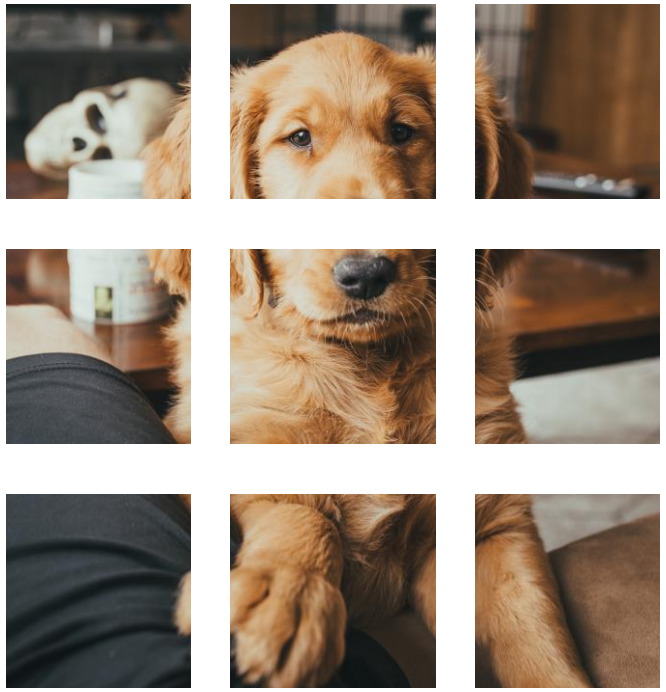


Photo by Josh Hild on Unsplash

Vision Transformer (ViT)

N input patches, each of shape 3x16x16



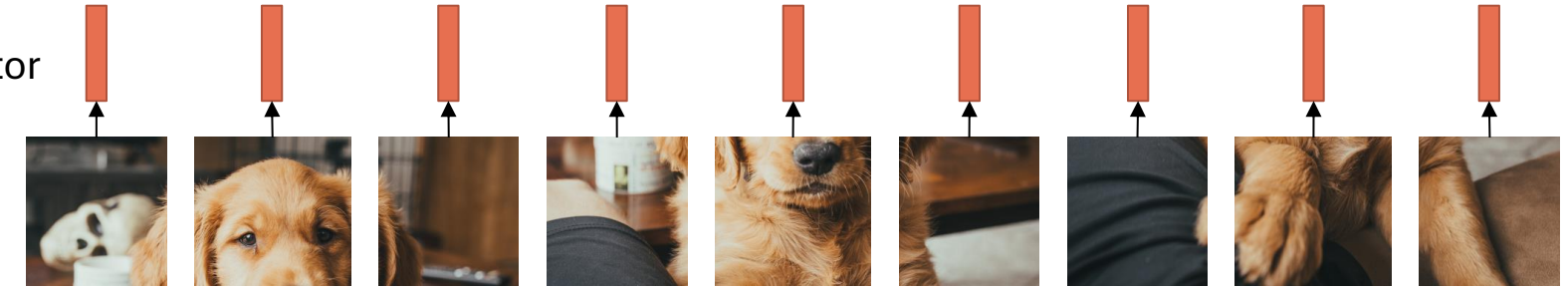
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Photo by Josh Hild on Unsplash

Vision Transformer (ViT)

Linear projection to D-dimensional vector

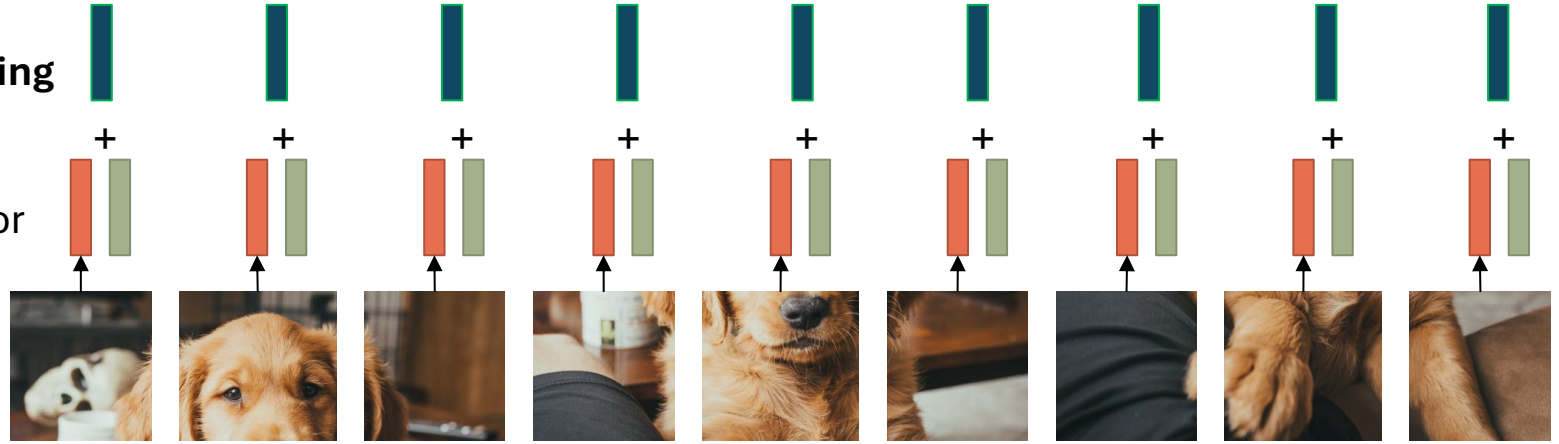
N input patches, each of shape 3x16x16



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

Photo by Josh Hild on Unsplash

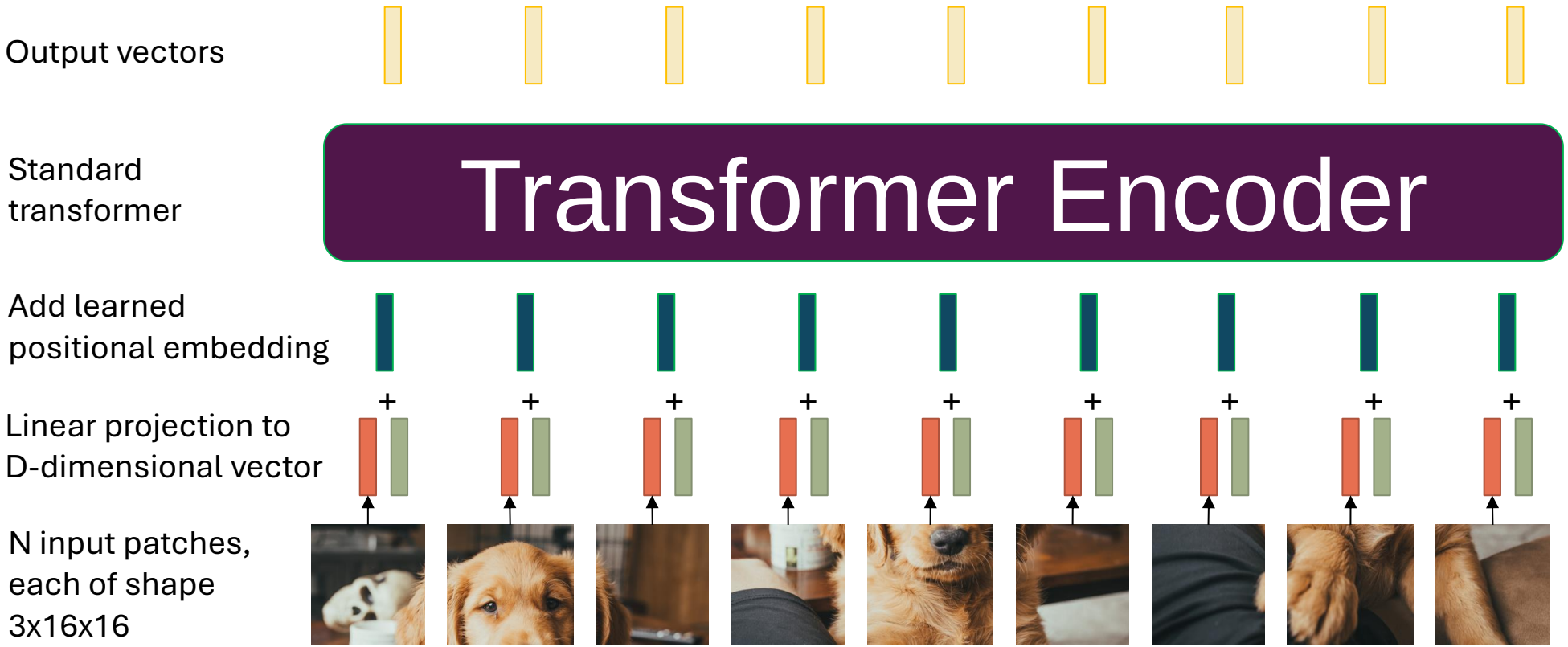
Vision Transformer (ViT)



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

Photo by Josh Hild on Unsplash

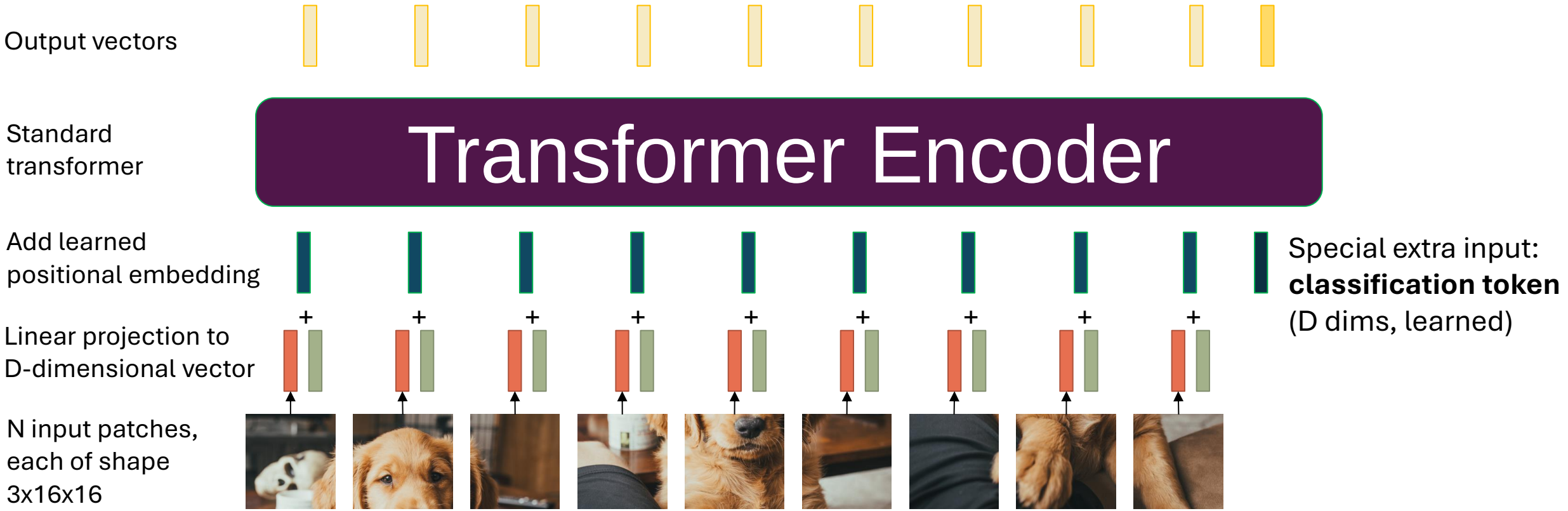
Vision Transformer (ViT)



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

Photo by Josh Hild on Unsplash

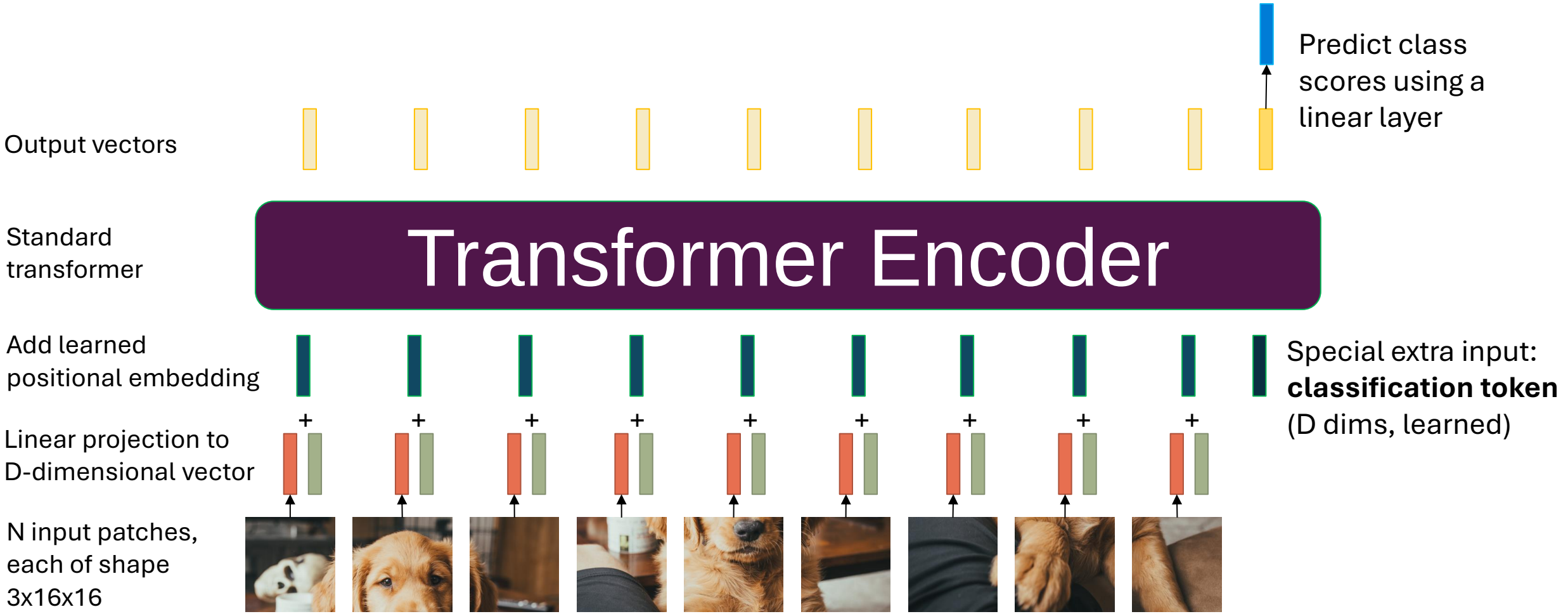
Vision Transformer (ViT)



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

Photo by Josh Hild on Unsplash

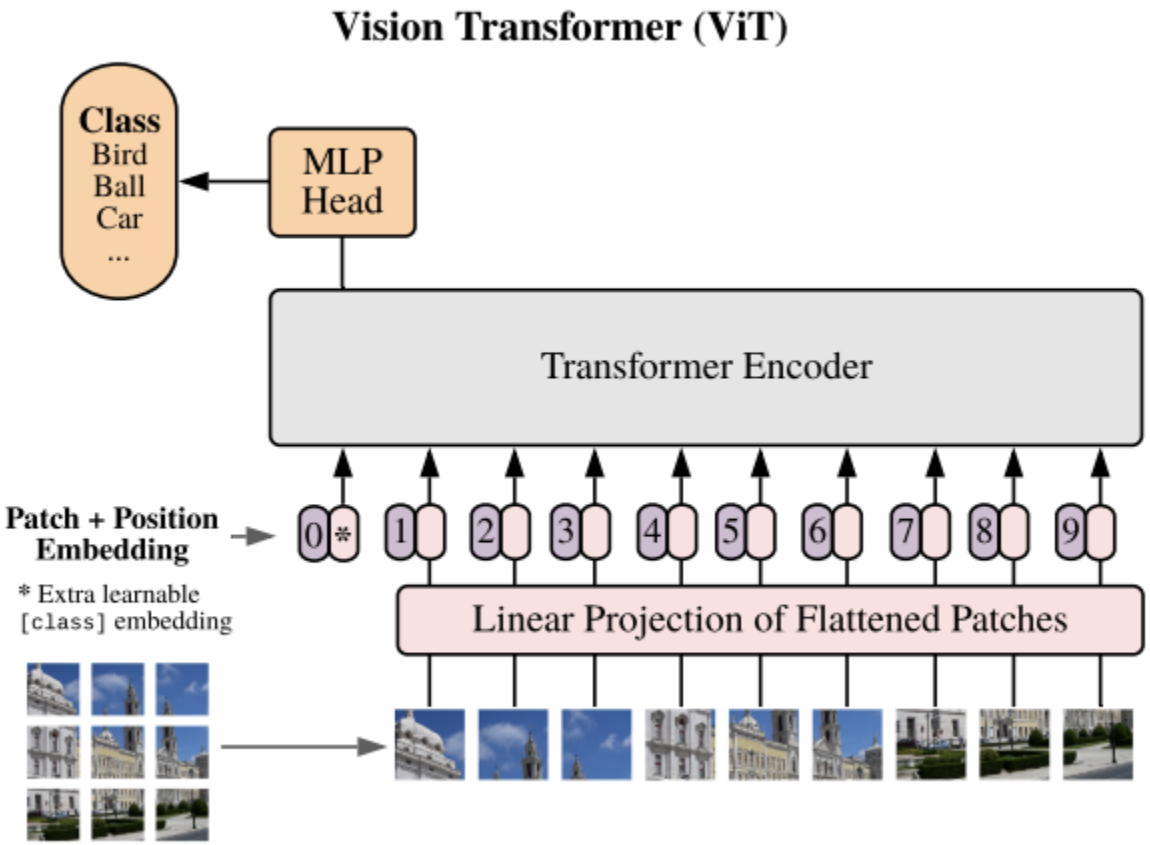
Vision Transformer (ViT)



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

Photo by Josh Hild on Unsplash

Vision Transformer (ViT)



Source: Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020). link

Outline

- Attention Mechanism
- Self-attention
- Multi-head self-attention
- Transformer encoder block
- Transformer decoder block
- Vision transformer (ViT)
- **Swin transformer**
- DEtection TRansformer (DETR) for object detection
- Suggested resources

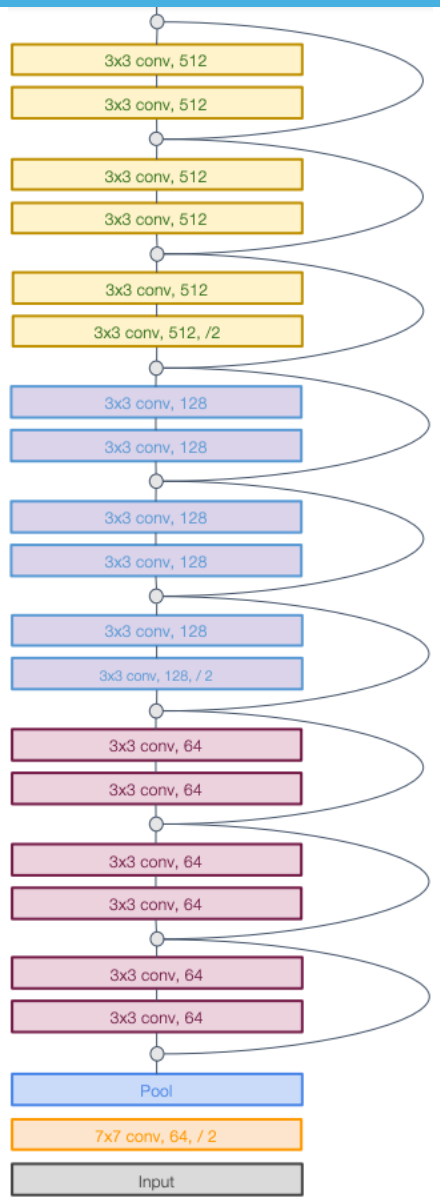
ViT vs ConvNet

Stage 3:
256 x 14 x 14

Stage 2:
128 x 28 x 28

Stage 1:
64 x 56 x 56

Input:
3 x 224 x 224



In most ConvNets, as we go deeper, resolution **decreases** and #channels **increase**. (Hierarchical architecture)

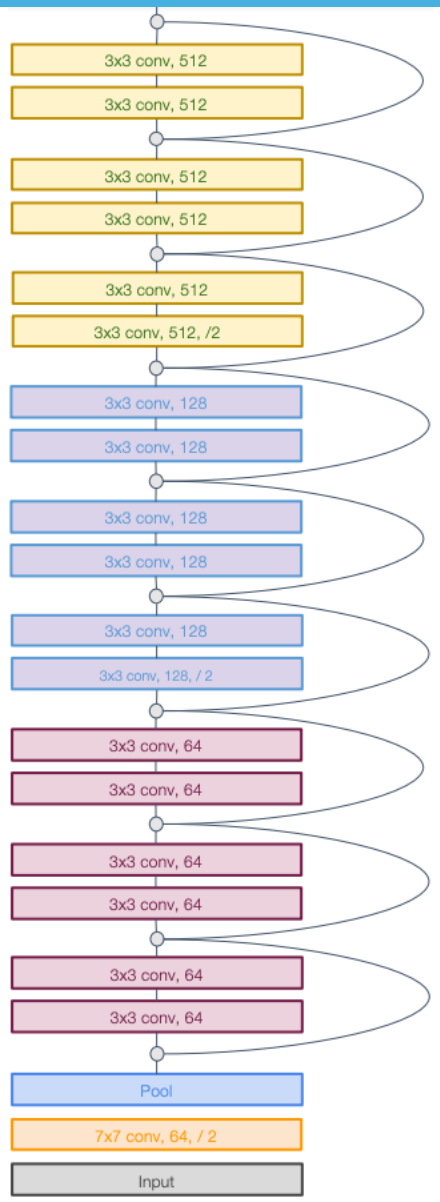
ViT vs ConvNet

Stage 3:
256 x 14 x 14

Stage 2:
128 x 28 x 28

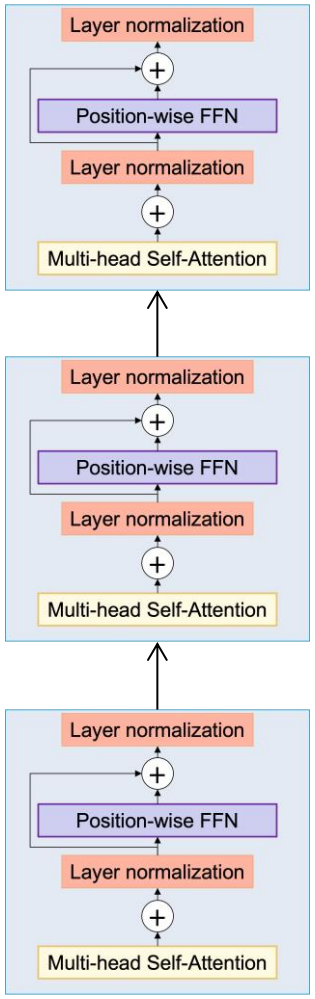
Stage 1:
64 x 56 x 56

Input:
3 x 224 x 224



In most ConvNets, as we go deeper, resolution **decreases** and #channels **increase**. (Hierarchical architecture)

In ViT, all blocks have the same resolution and number of channels



3rd Block:
768 x 14 x 14

2nd Block:
768 x 14 x 14

1st Block:
768 x 14 x 14

Input:
3 x 224 x 224

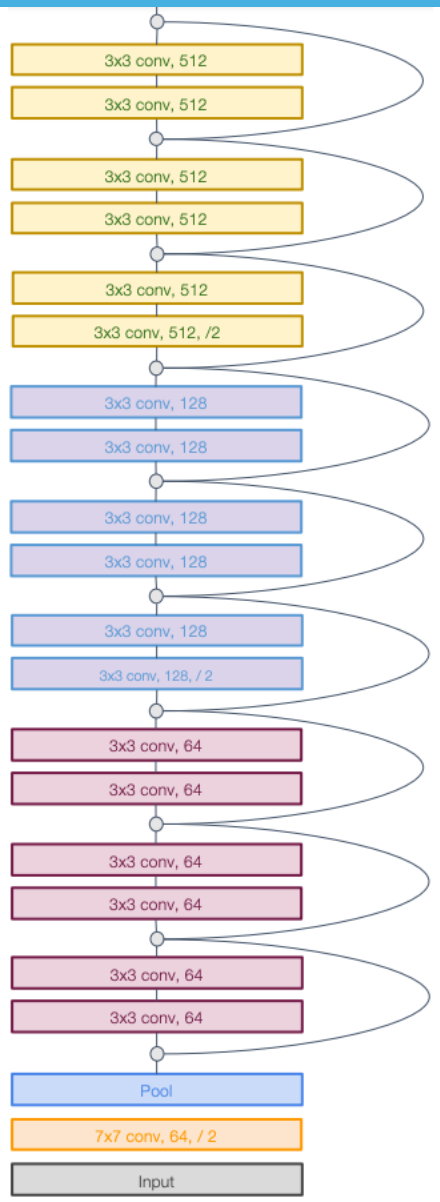
ViT vs ConvNet

Stage 3:
256 x 14 x 14

Stage 2:
128 x 28 x 28

Stage 1:
64 x 56 x 56

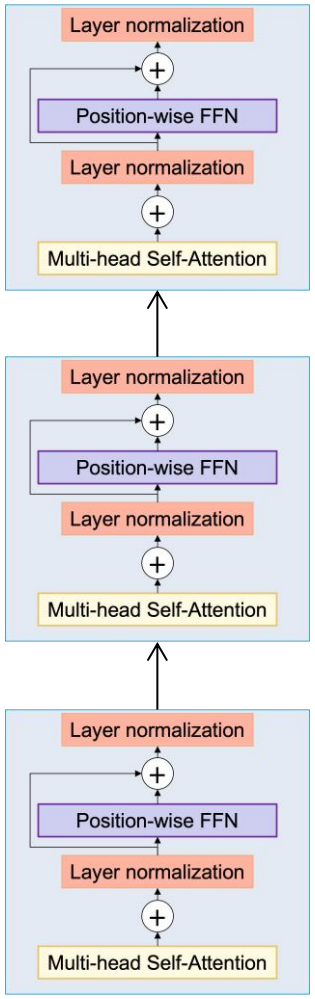
Input:
3 x 224 x 224



In most ConvNets, as we go deeper, resolution **decreases** and #channels **increase**. (Hierarchical architecture)

In ViT, all blocks have the same resolution and number of channels

Can we build a hierarchical ViT model?



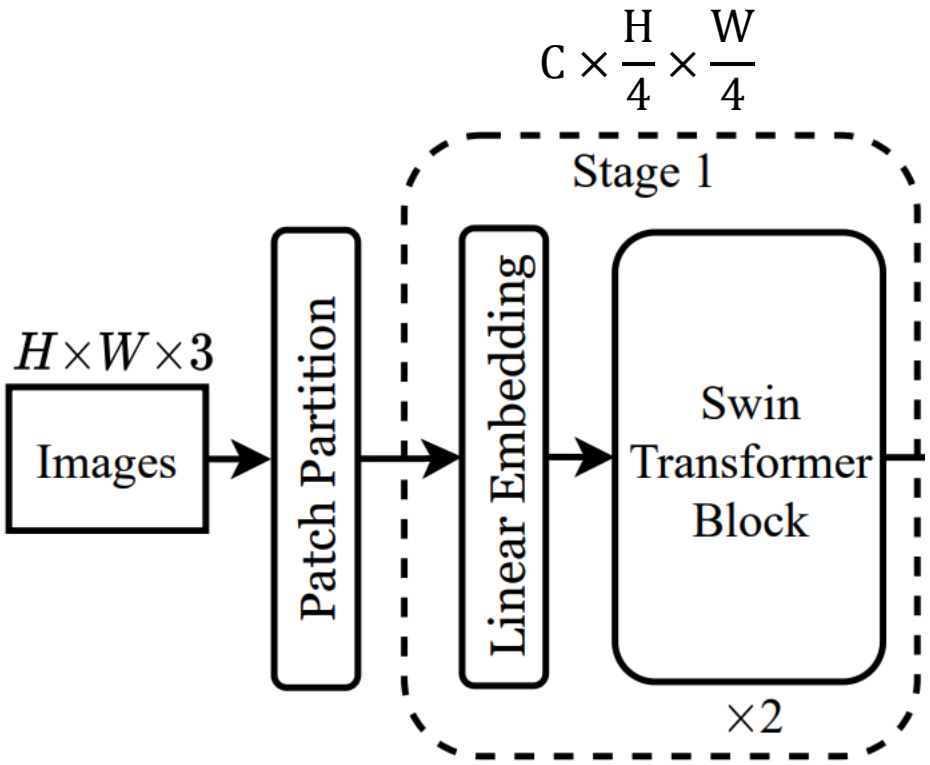
3rd Block:
768 x 14 x 14

2nd Block:
768 x 14 x 14

1st Block:
768 x 14 x 14

Input:
3 x 224 x 224

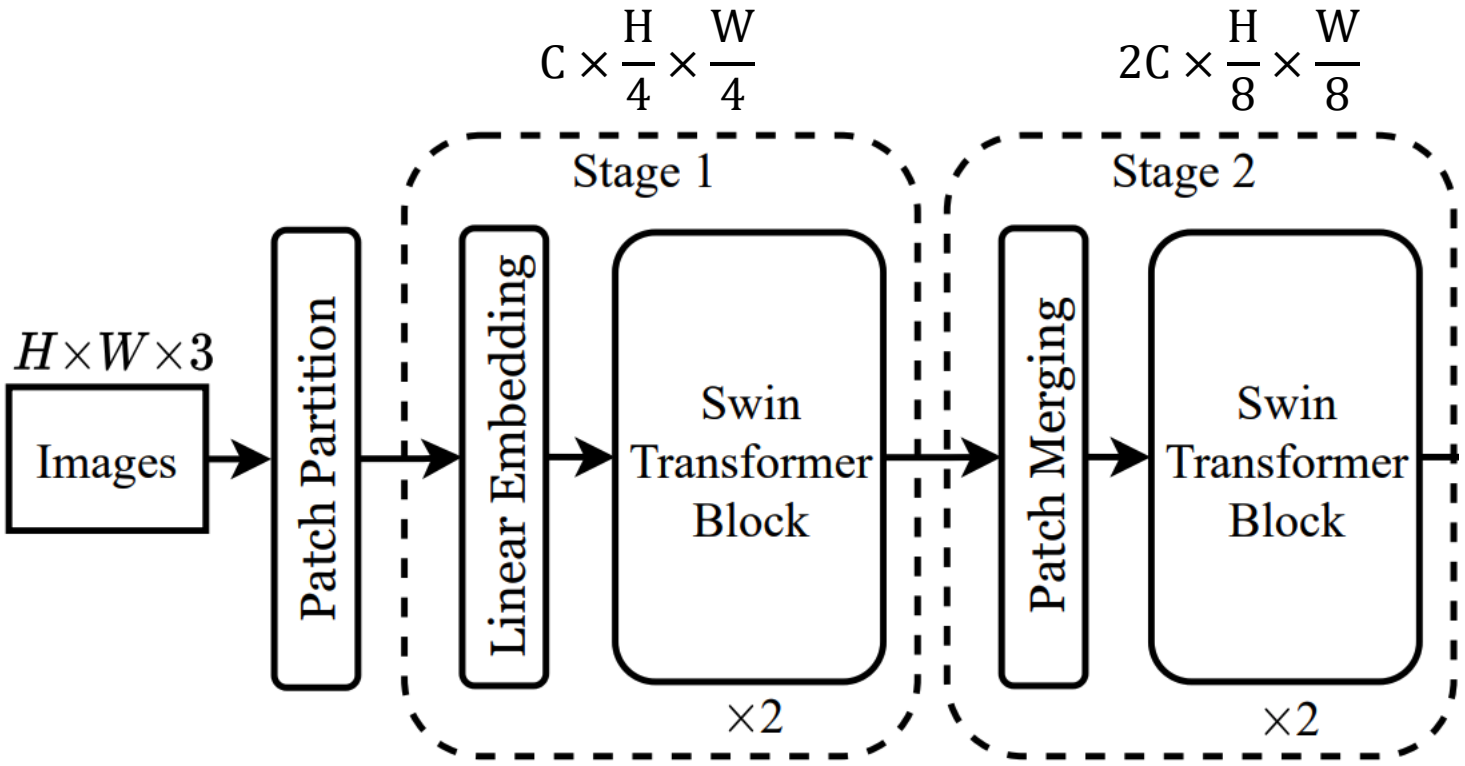
Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighbourhoods. Now patches are (effectively) 8x8

Patch Merging:

1. For every group of 2x2 patches, concatenate the corresponding 4 vectors.
2. Pass the concatenated vectors (of size 4C) through a linear layer to get vectors of size 2C.

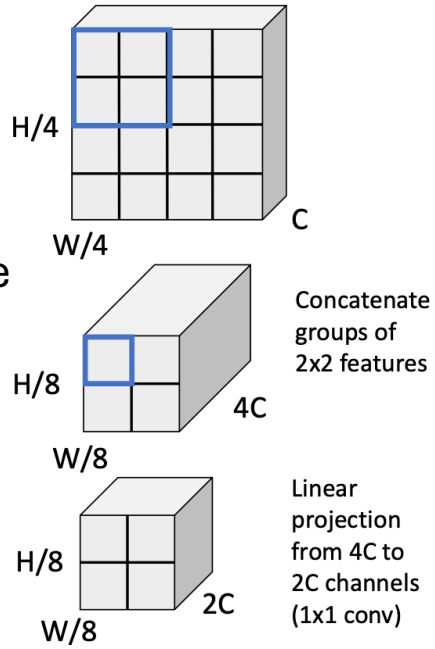
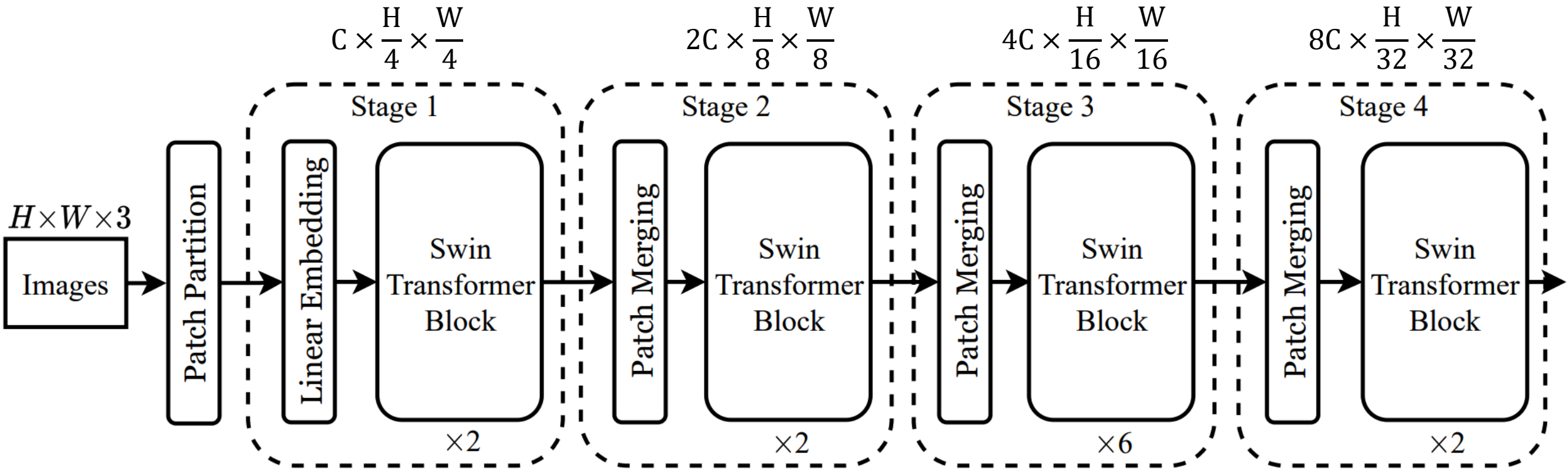


Image source

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighbourhoods. Now patches are (effectively) 8x8

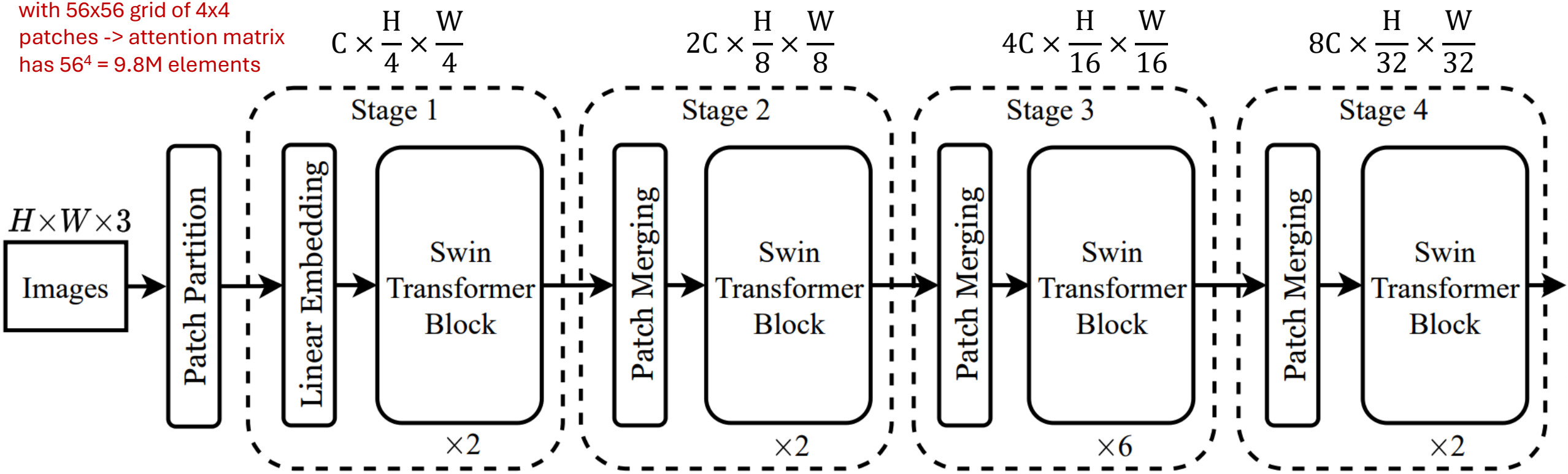
Merge 2x2 neighbourhoods. Now patches are (effectively) 16x16

Merge 2x2 neighbourhoods. Now patches are (effectively) 32x32

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer

Problem: 224x224 image with 56x56 grid of 4x4 patches -> attention matrix has 56^4 = 9.8M elements



Solution: Limit the attention of a patch to its "window".

Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighbourhoods. Now patches are (effectively) 8x8

Merge 2x2 neighbourhoods. Now patches are (effectively) 16x16

Merge 2x2 neighbourhoods. Now patches are (effectively) 32x32

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Window Attention



Instead of allowing each token to attend to all other tokens, divide image into M x M **windows** (here M=4).

Compute attention only within each window.

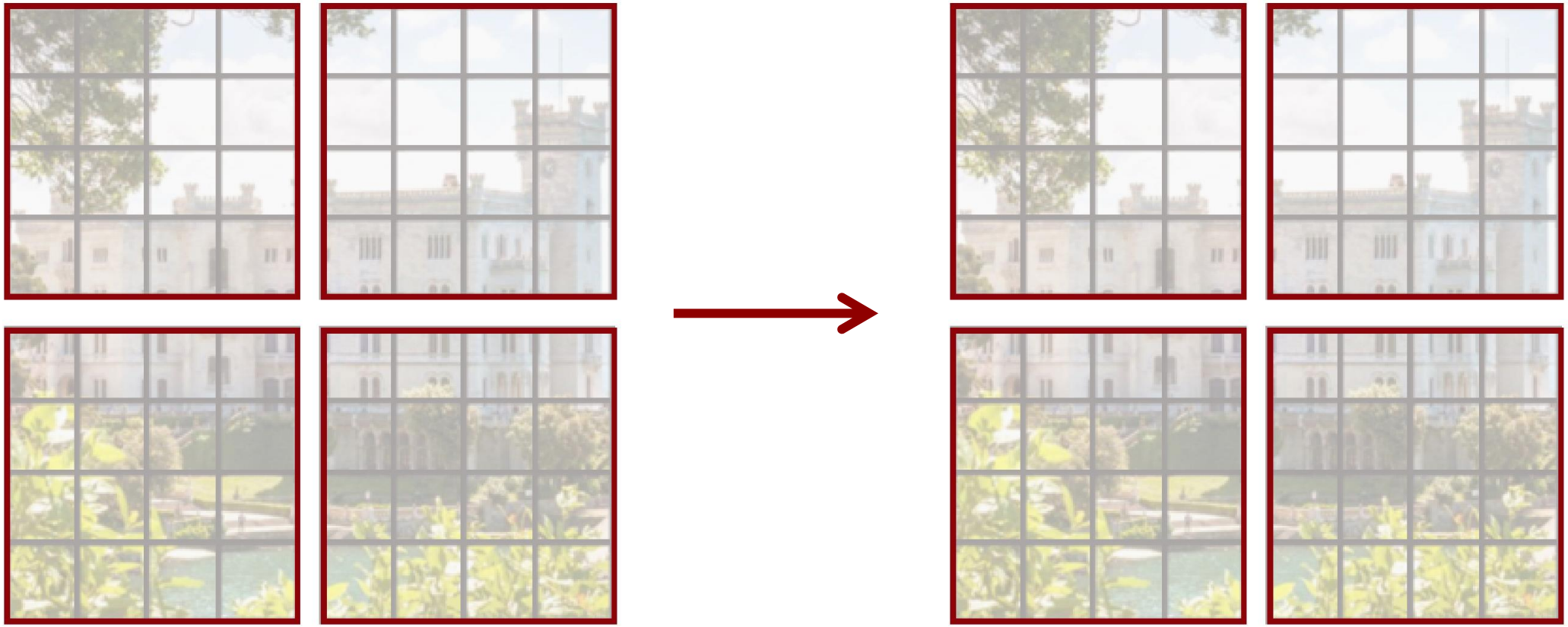
A local window to perform self-attention

A patch

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Window Attention

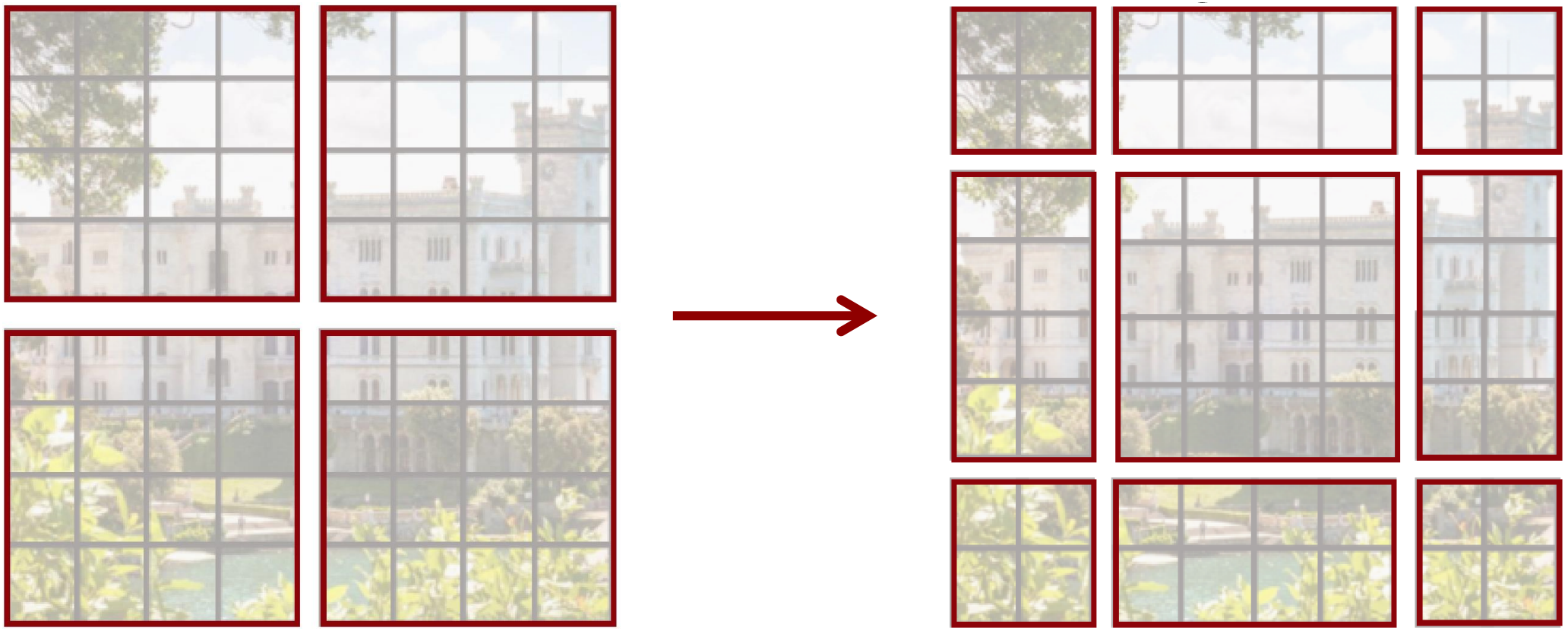
Problem: No communication across windows. Global context?



Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Shifted Window Attention

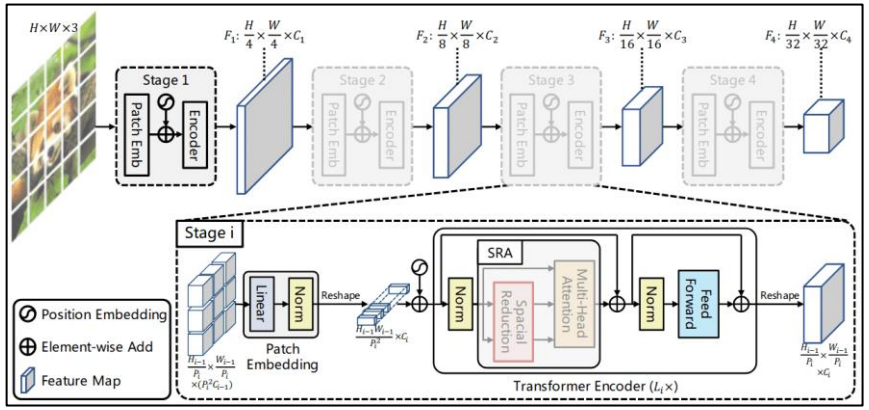
Solution: Alternate between normal windows and shifted windows in successive Transformer blocks.



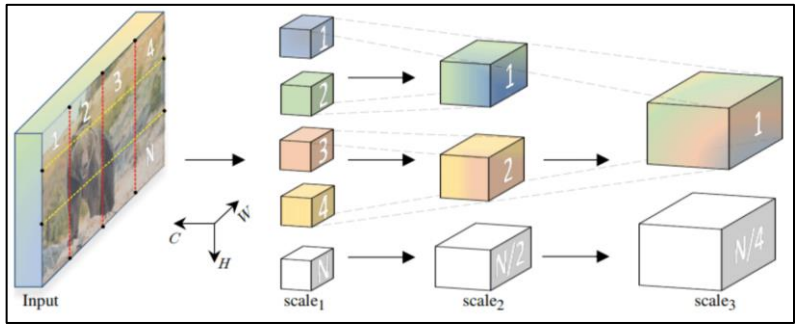
Block L: Normal windows

Block L+1: Shifted windows

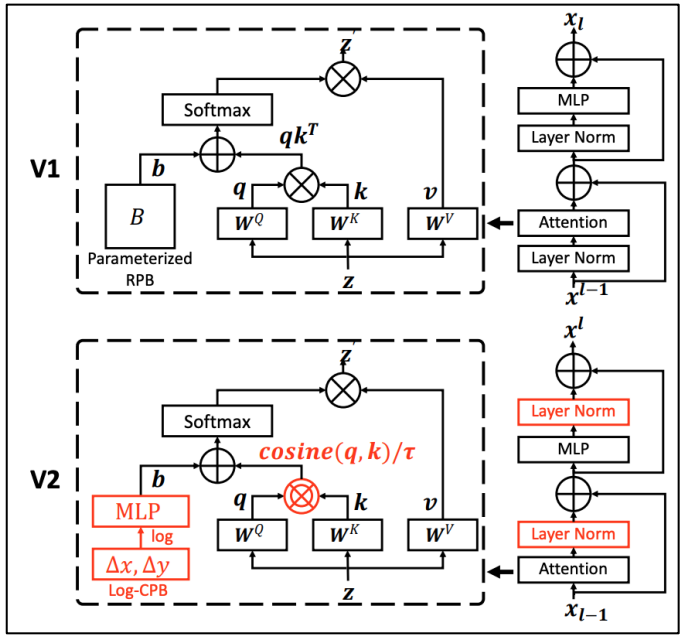
Other hierarchical vision transformers



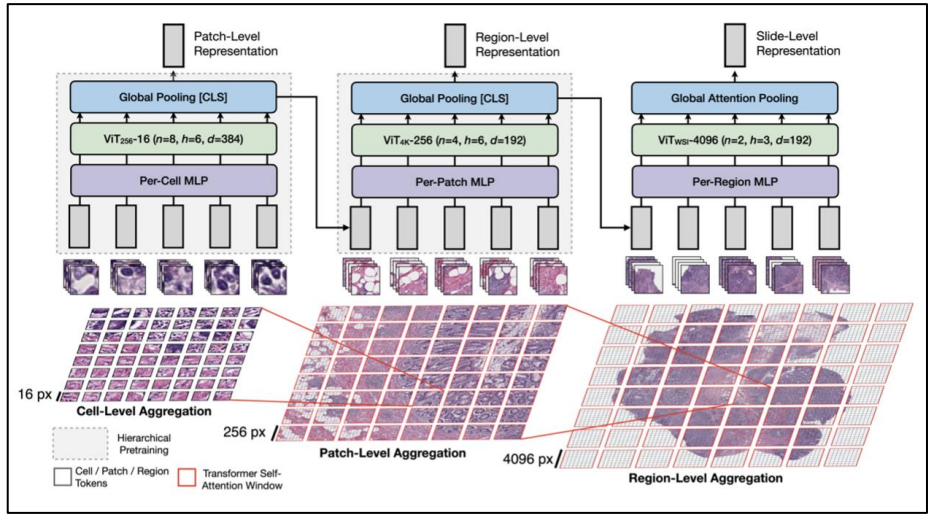
Wang, Wenhai, et al. "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions.", ICCV 2021



Fan, Haoqi, et al. "Multiscale vision transformers.", ICCV 2021



Liu, Ze, et al. "Swin transformer v2: Scaling up capacity and resolution.", arXiv 2021, CVPR 2022



Chen, Richard J., et al. "Scaling vision transformers to gigapixel images via hierarchical self-supervised learning." *CVPR* 2022.

Is self-attention magical and absolutely necessary?

- A lot of works have explored replacing self-attention with **X** and some of them achieve competitive results.
  - Wu et al replace self-attention with convolution.
  - Tay et al replace alignment scores in self-attention with random projection.
  - Tolstikhin et al replace self-attention with an MLP-mixer making the architecture exclusively based on MLPs.
  - Lee-Thorp et al replace self-attention with Fourier Transforms.
  - Tatsunami and Taki replace self-attention with LSTM.

- But, self-attention continues to be the standard choice.

Outline

- Attention Mechanism
- Self-attention
- Multi-head self-attention
- Transformer encoder block
- Transformer decoder block
- Vision transformer (ViT)
- Swin transformer
- **Suggested resources**
- Mandatory Assignment 2

Suggested resources

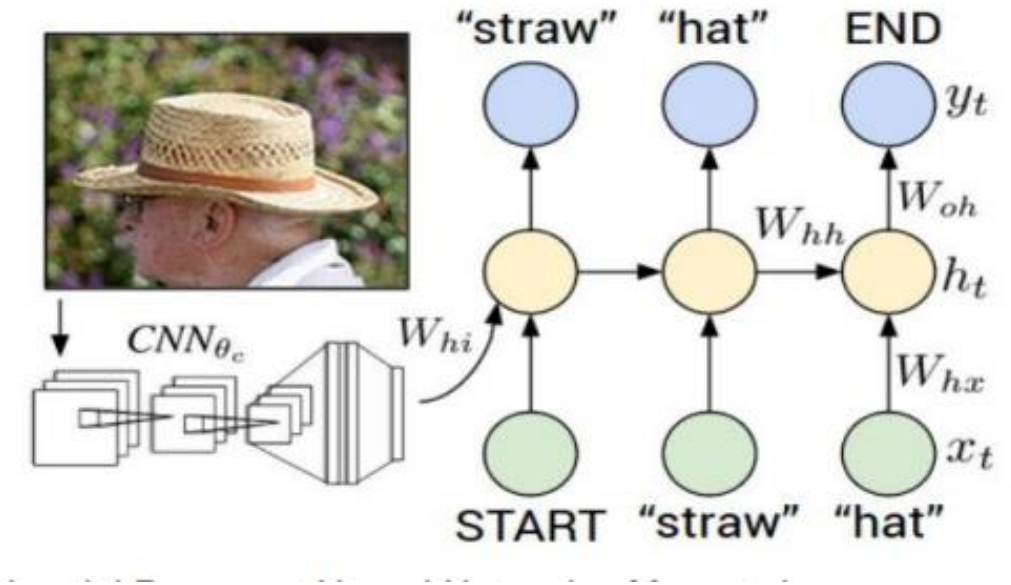
- The Illustrated Transformer: A famous blog post covering the basics of Transformers.
- The Annotated Transformer: A blog post that implements Transformer line by line with explanation.
- StatQuest: YouTube channel with simple explanations
- Attention mechanism:
  - Basic Intuition: https://www.youtube.com/watch?v=SysgYptB198
  - Attention model: https://www.youtube.com/watch?v=quoGRI-1l0A
  - https://machinelearningmastery.com/the-attention-mechanism-from-scratch/
- Additional material for mandatory 2:
  - BLEU score: https://towardsdatascience.com/bleu-bilingual-evaluation-understudy-2b4eab9bcfd1
  - METEOR score: https://machinelearninginterview.com/topics/machine-learning/meteor-for-machine-translation/

Outline

- Attention Mechanism
- Self-attention
- Multi-head self-attention
- Transformer encoder block
- Transformer decoder block
- Vision transformer (ViT)
- Swin transformer
- Suggested resources
- **Mandatory Assignment 2**
- (DINO)

Image Captioning

- Combine a CNN with an RNN to generate descriptive text for an image.
- Train an RNN to interpret high level features of a CNN



Deep Visual-Semantic Alignments for Generating Image Descriptions

Image Captioning

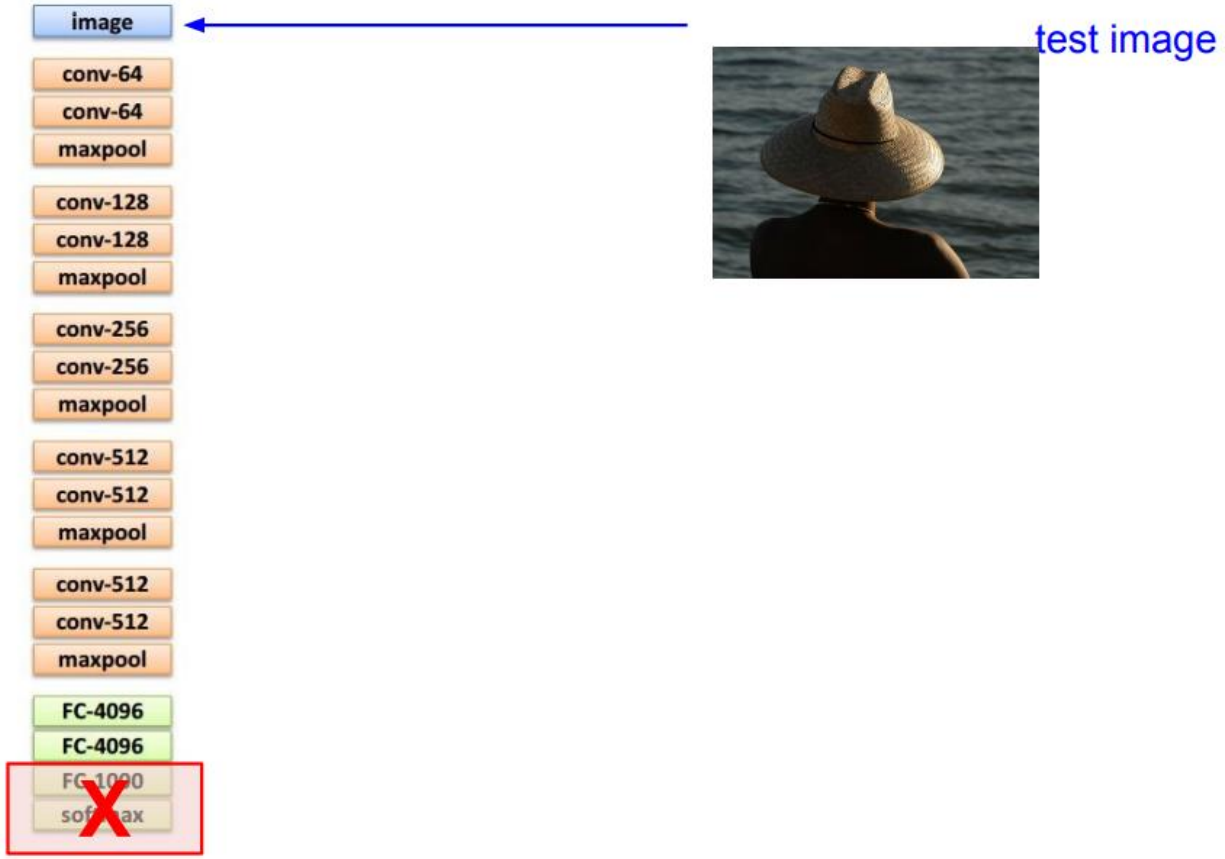


Image Captioning

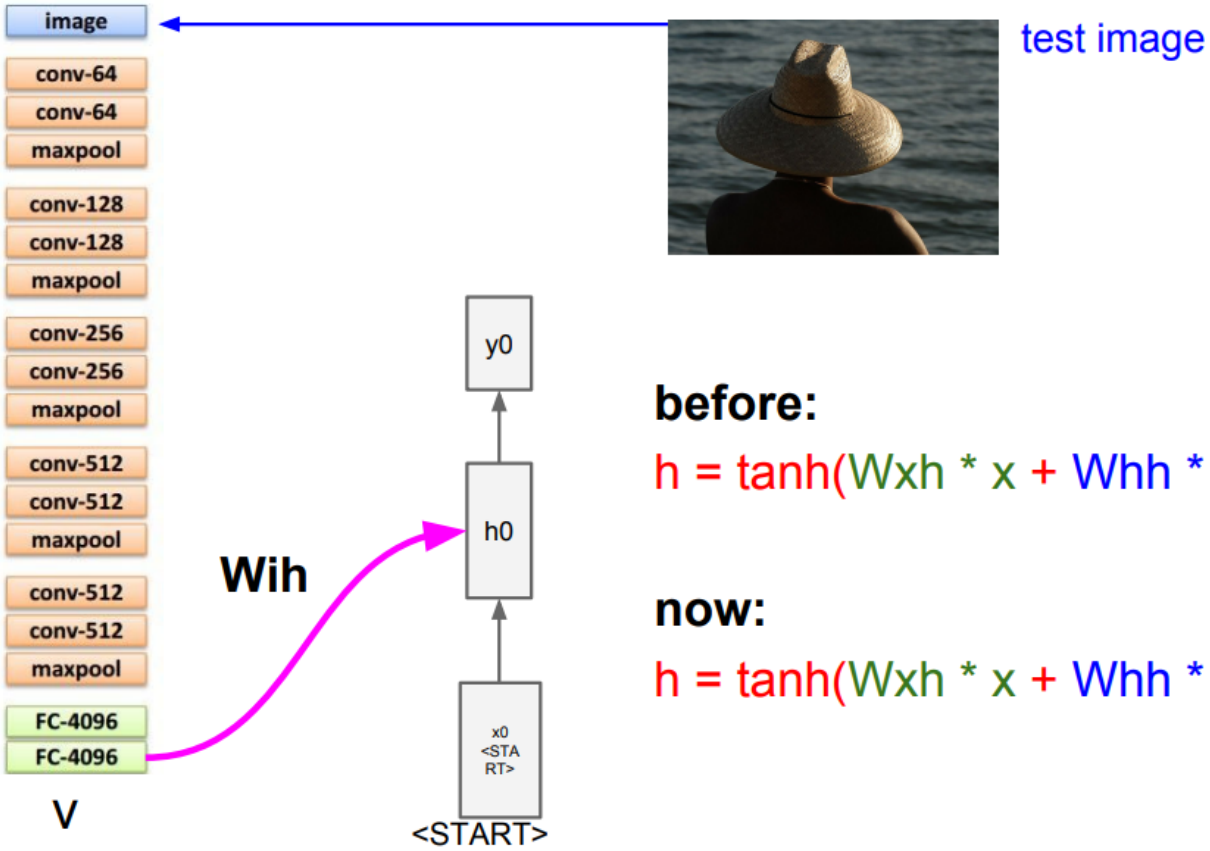


Image Captioning

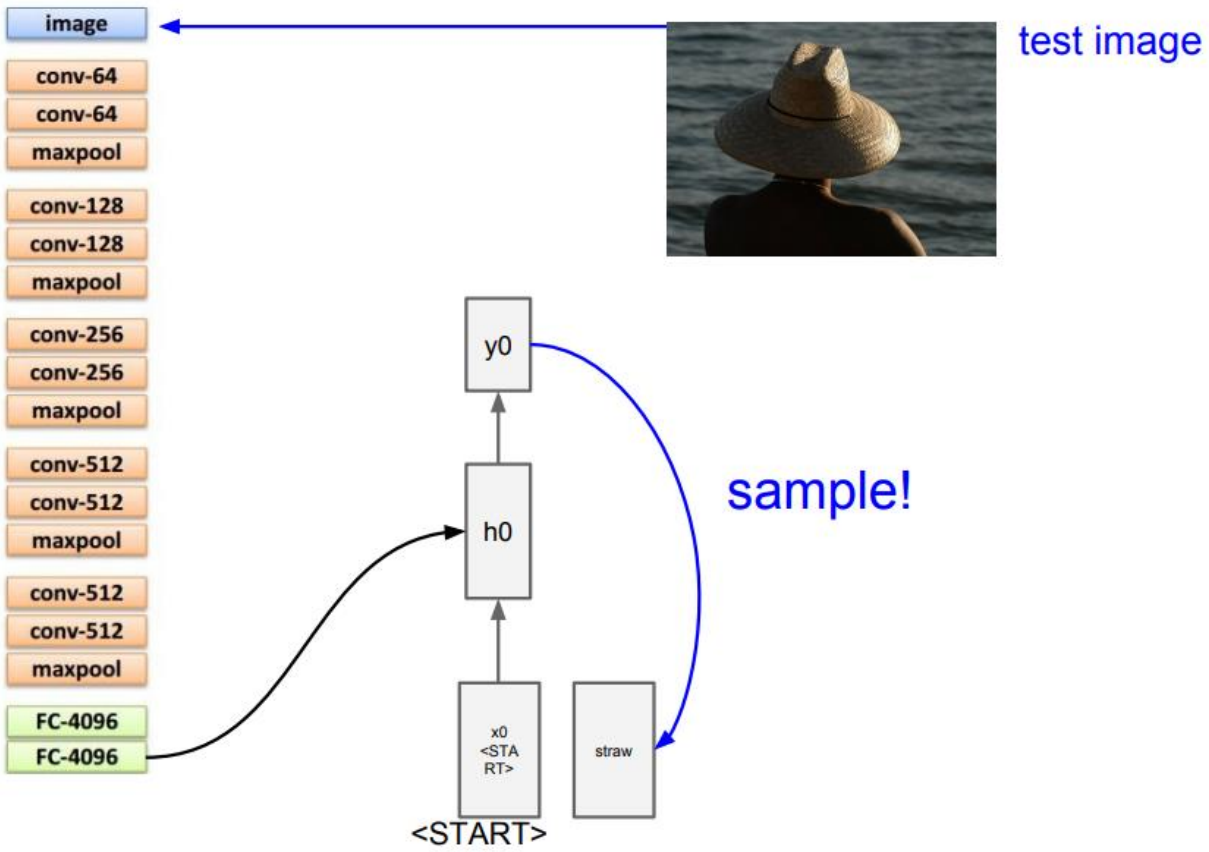


Image Captioning

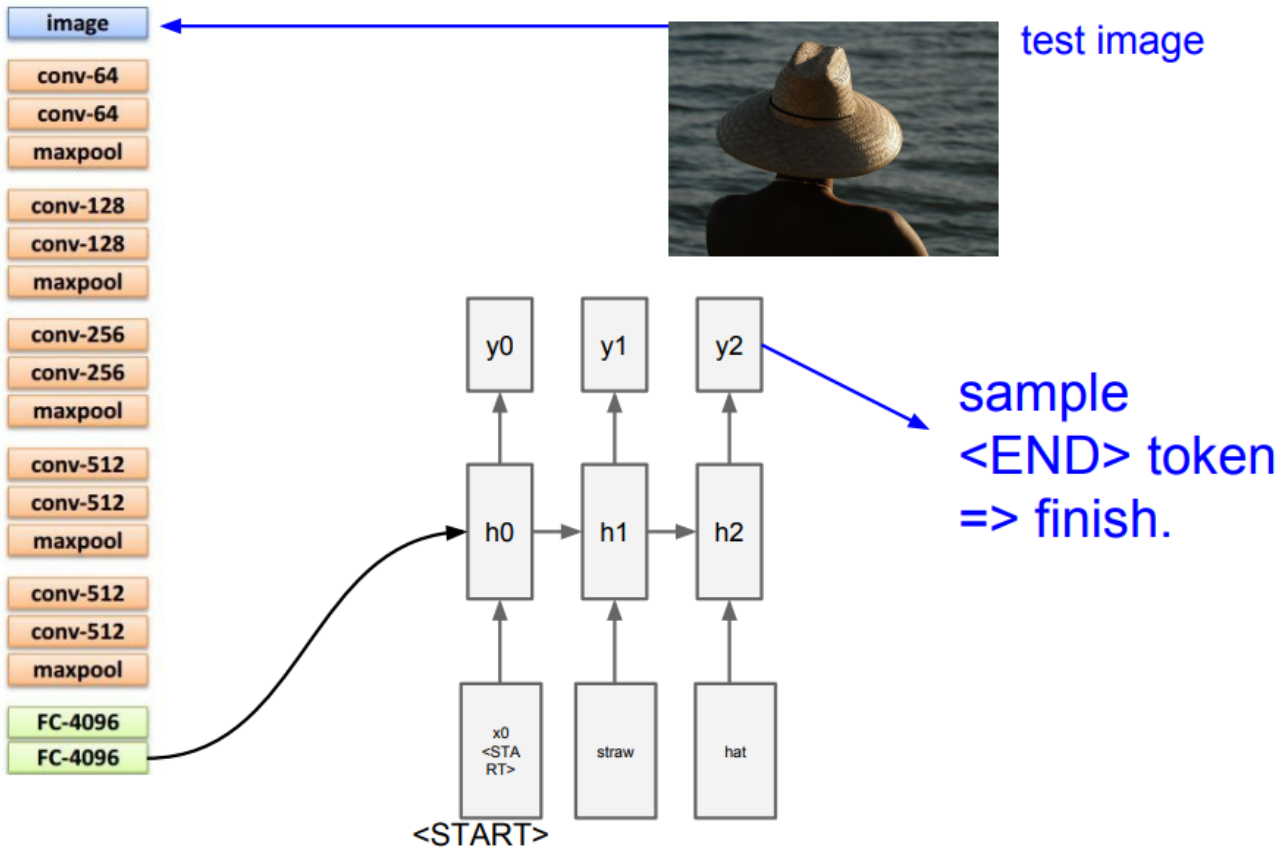


Image Captioning: Results



A cat sitting on a suitcase on the floor



A cat is sitting on a tree branch



A dog is running in the grass with a frisbee



A white teddy bear sitting in the grass



Two people walking on the beach with surfboards



A tennis player in action on the court



Two giraffes standing in a grassy field



A man riding a dirt bike on a dirt track

NeuralTalk2

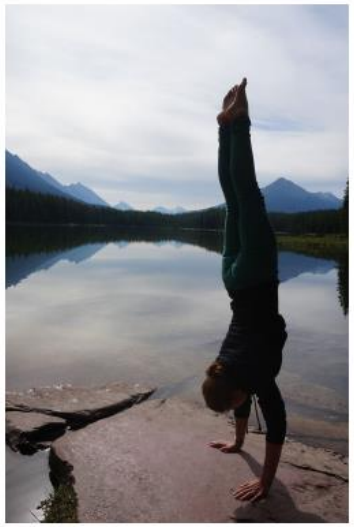
Image Captioning: Failures



A woman is holding a cat in her hand



A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard



A bird is perched on a tree branch



A man in a baseball uniform throwing a ball

NeuralTalk2

Image Captioning: Measuring Quality

- We can measure the performance of the model by using a criterion for similarity between the true caption and the generated caption.
- One such measure is the BLEU score (**Bi**l**e**ingual **e**valuation **u**nderstudy) see .e.g. https://en.wikipedia.org/wiki/BLEU.
- BLEU is a modified precision measure that handles sequences of different length. It combines counts of unigrams, bigrams, and n-grams into one score 0-100%.
- BLEU will basically compare n-grams (n=1-4 for instance) of the generated caption with the n-grams of the target captions and give a score.
- Read more here: https://towardsdatascience.com/bleu-bilingual-evaluation-understudy-2b4eab9bcfd1

General tips

- Complete report

- Code

Outline

- Attention Mechanism
- Self-attention
- Multi-head self-attention
- Transformer encoder block
- Transformer decoder block
- Vision transformer (ViT)
- Swin transformer
- DEtection TRansformer (DETR) for object detection
- Suggested resources
- Mandatory Assignment 2
- **DINO**

Trending transformers: Foundation models

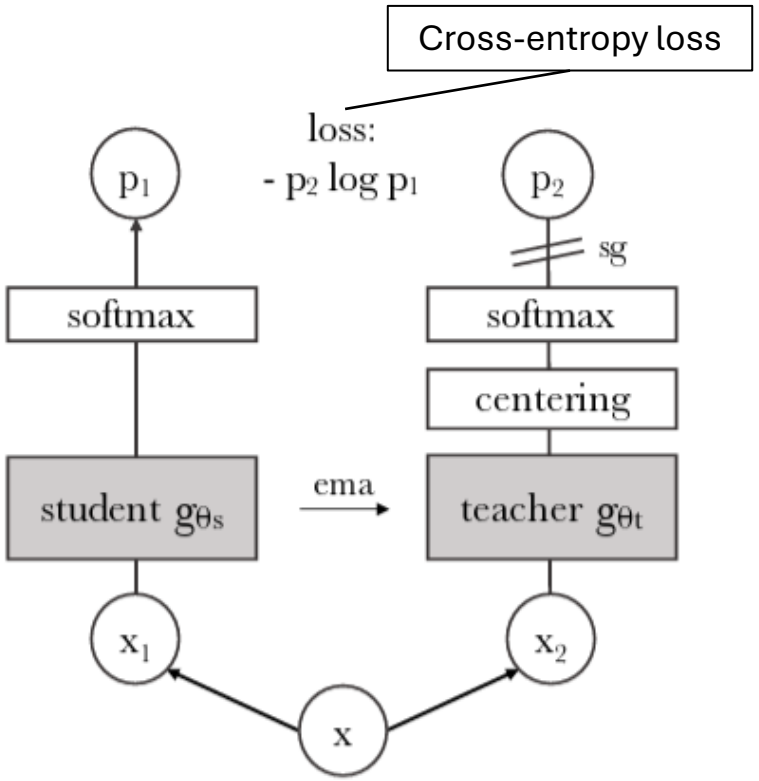
- Large models trained on vast amounts of data
- GPT Achiam, Josh, et al. "Gpt-4 technical report." arXiv preprint arXiv:2303.08774 (2023).
- StarCoder Li, Raymond, et al. "Starcoder: may the source be with you!." arXiv preprint arXiv:2305.06161 (2023).
- Virchow2 Zimmermann, Eric, et al. "Virchow2: Scaling self-supervised mixed magnification models in pathology." arXiv preprint arXiv:2408.00738 (2024).

DINO

- Contrastive learning
- Data augmentations
- Self-supervised learning

- Caron, Mathilde, et al. "Emerging properties in self-supervised vision transformers." Proceedings of the IEEE/CVF international conference on computer vision. 2021.
- Oquab, Maxime, et al. "Dinov2: Learning robust visual features without supervision." arXiv preprint arXiv:2304.07193 (2023).

DINO





Figures from: Caron, Mathilde, et al. "Emerging properties in self-supervised vision transformers." Proceedings of the IEEE/CVF international conference on computer vision. 2021.

DINO

