1. Recurrent Neural Networks (RNNs) and Their Trade-offs

- **Advantages**:
  - Can process sequences of arbitrary length.
  - Shares weights across time, making it parameter-efficient.
- **Disadvantages**:
  - Sequential processing leads to slow computation.
  - Struggles with long-term dependencies due to vanishing gradients.

2. seq2seq Task

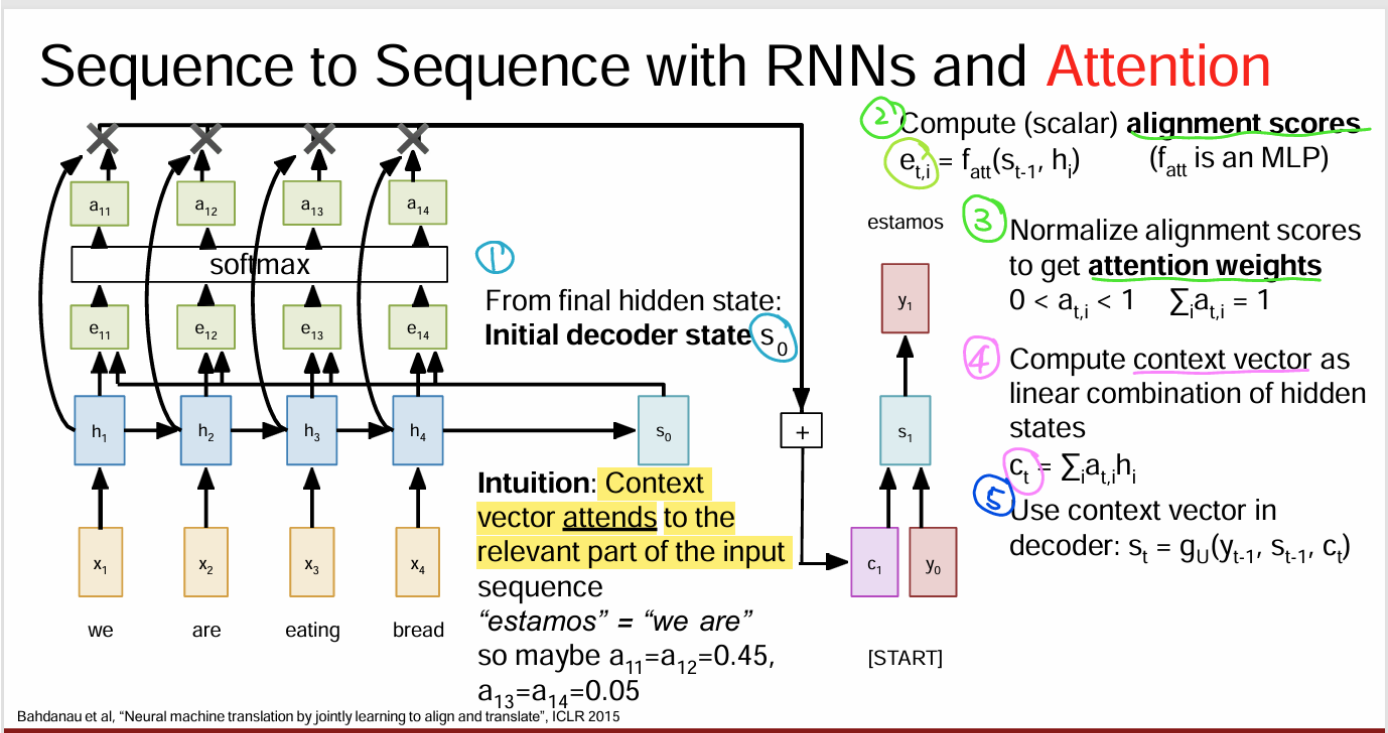
2.1 only RNNs

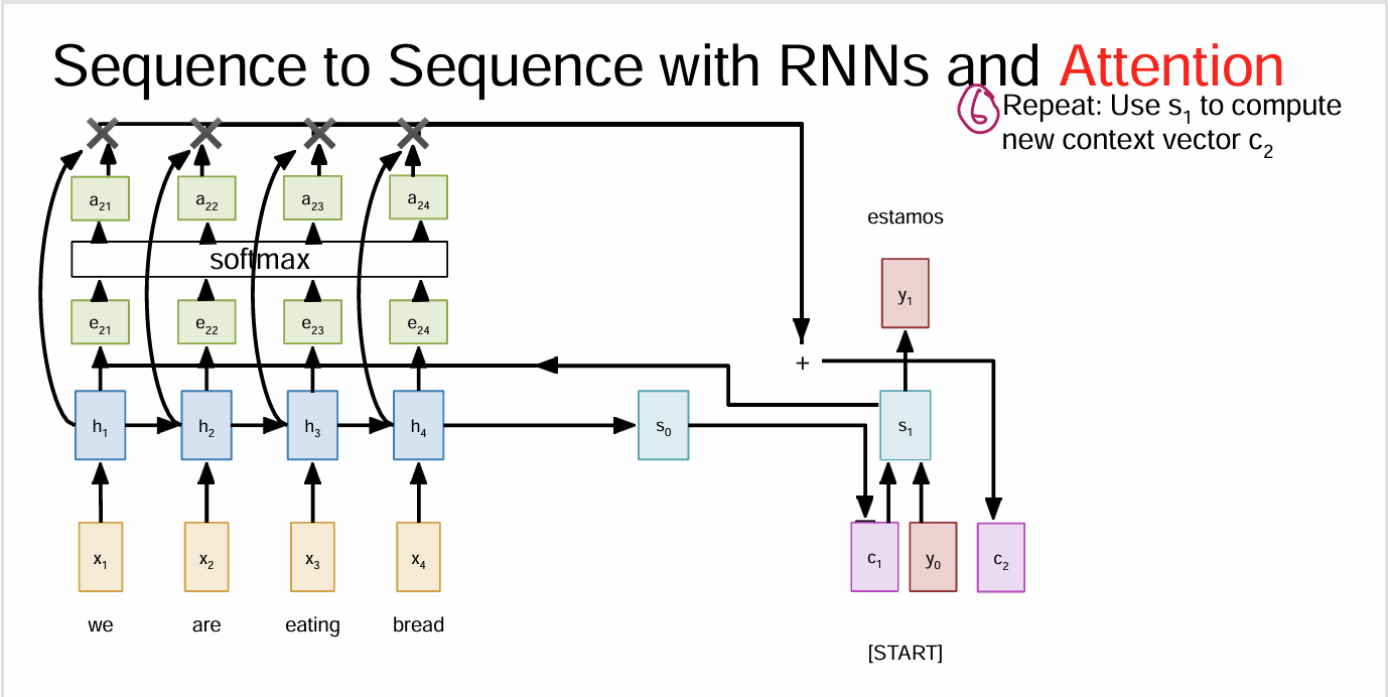
Problem: The encoder compresses the entire input sequence into a single fixed-size vector (the final hidden state of the encoder).
This vector serves as the sole context for the decoder to generate the output sequence.For long input sequences, this fixed-size vector often fails to capture all relevant information, leading to loss of critical details.
Decoders struggle to generate accurate outputs for longer or more complex sequences.

2.2 RNN + Attention

- Attention addresses the bottleneck of fixed-size context vectors by allowing the decoder to focus on specific parts of the input sequence dynamically.
- **Key Features**:
  - **Alignment Scores**

- **Context Vector**



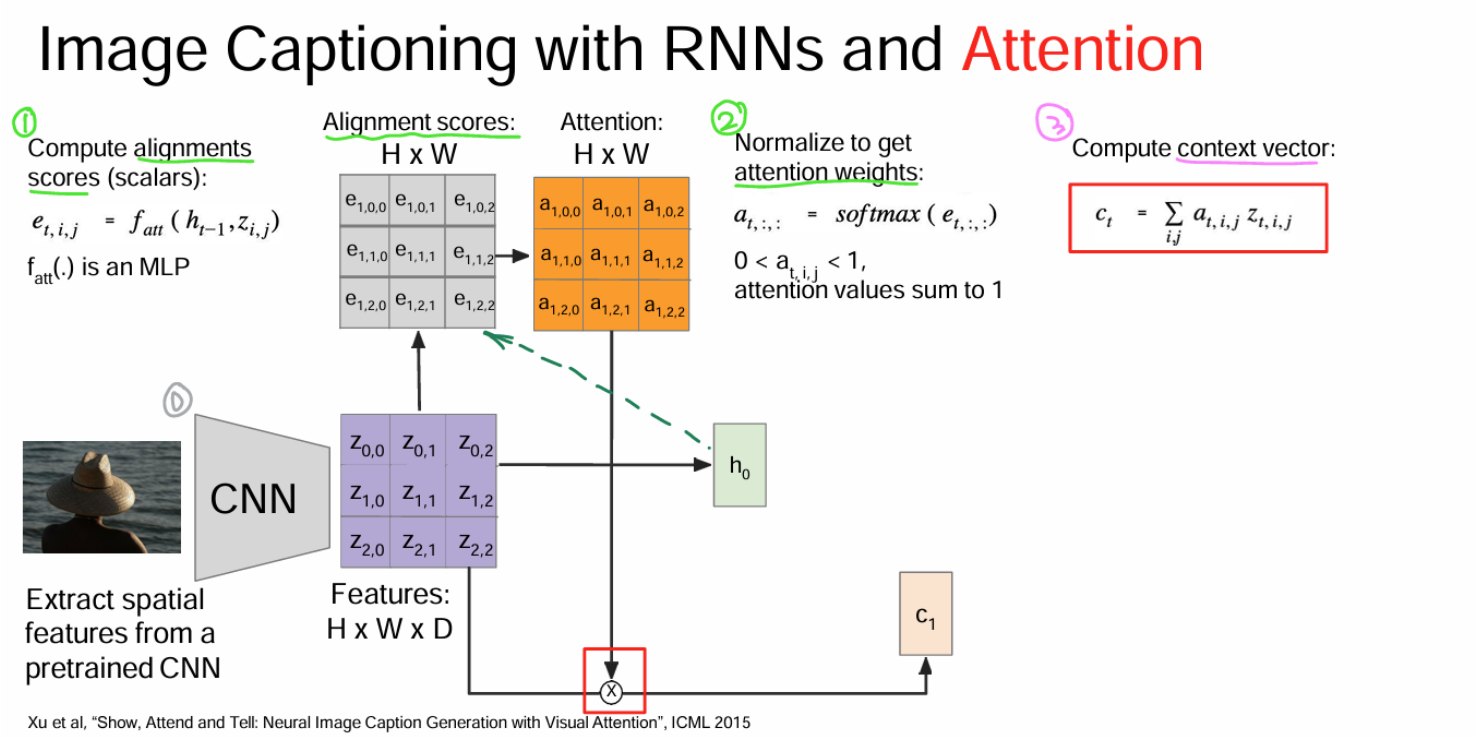


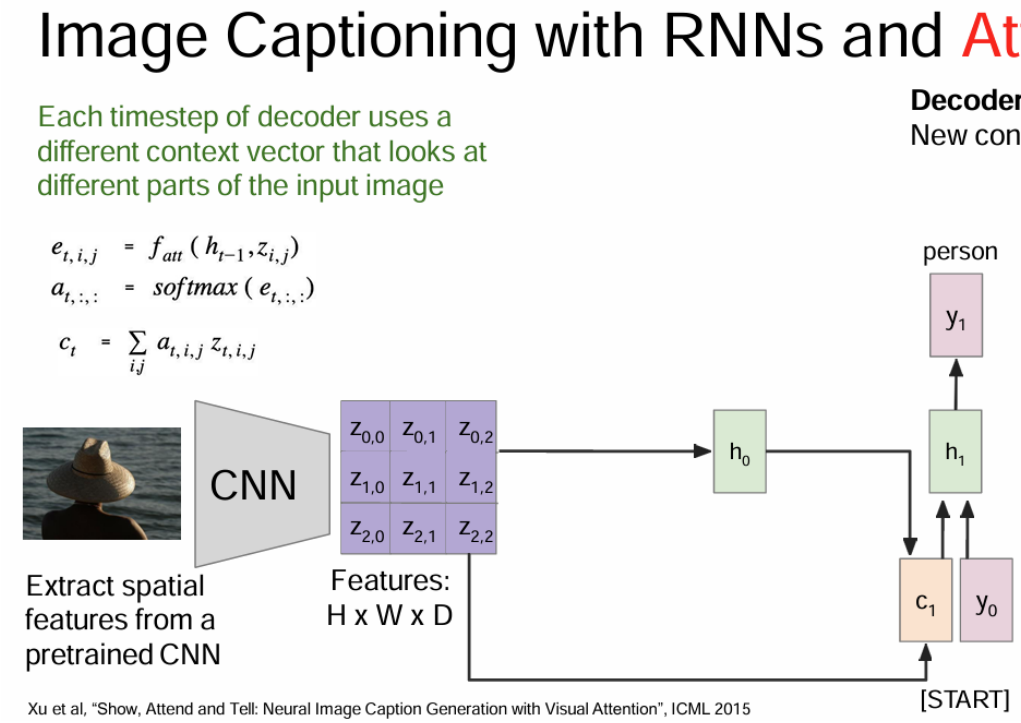
- different context vector in each timestep of decoder
- Input sequence not bottlenecked through single vector
- At each timestep of decoder, context vector "looks at" different parts of the input sequence

3. Image Captioning Task

3.1 RNN + Attention

New context vector at every time step. Each context vector will attend to different image regions.





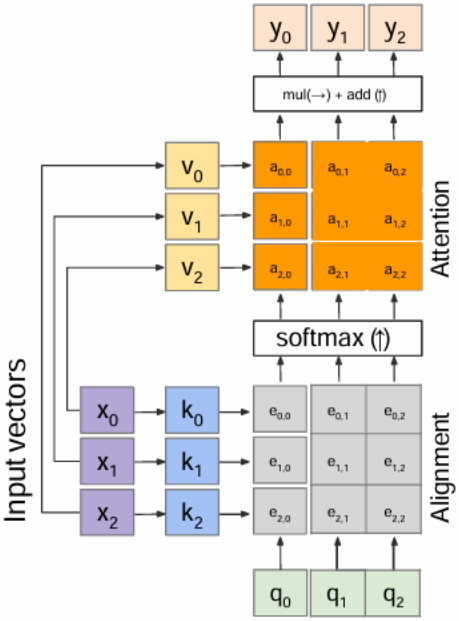
This entire process is differentiable.
Model chooses its own attention weights, no attention supervision is required.

4. General att. V.S. Self att.

Aspect	General Attention Layer	Self-Attention Layer
Purpose,focuses on relationship	between two sets of vectors (e.g., encoder-decoder).	within the same sequence.
Inputs	Query (decoder), Keys and Values (encoder).	Query, Key, and Value are all derived fron

Aspect	General Attention Layer	Self-Attention Layer
Context Vector	information from the external input sequence.	relationships among tokens in the same s
Usage	Used in encoder-decoder attention for seq2seq tasks.	Core of self-contained models like Transf
Applications	Machine translation, image captioning, multimodal tasks.	Language modeling, text generation, ima
Complexity	Depends on the sizes of the query and key-value sets.	Scales quadratically with sequence lengtl

General attention layer



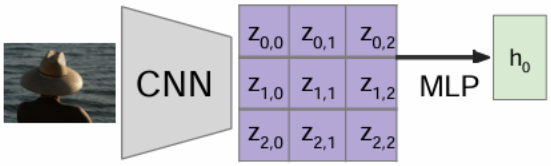
Outputs:
context vectors: **y** (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}W_k$
Value vectors: $\mathbf{v} = \mathbf{x}W_v$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

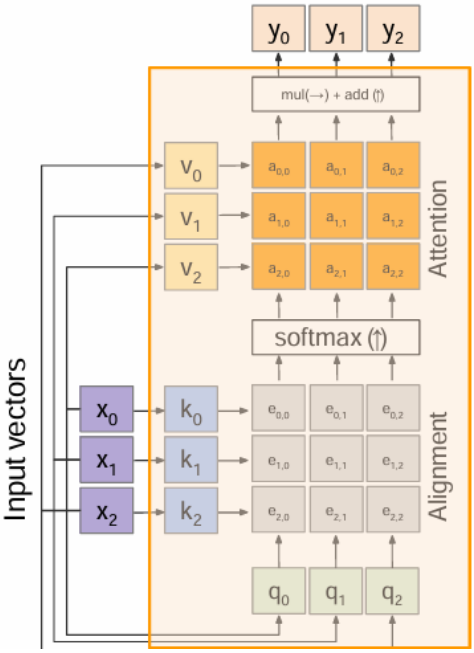
Inputs:
Input vectors: **x** (shape: N x D)
Queries: **q** (shape: M x D_k)

Recall that the query vector was a function of the input vectors

Encoder: $h_0 = f_W(\mathbf{z})$
where **z** is spatial CNN features
$f_W(.)$ is an MLP



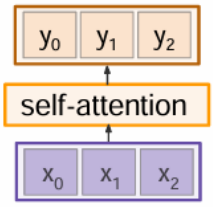
Self attention layer - attends over sets of inputs



Outputs:
context vectors: **y** (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}W_k$
Value vectors: $\mathbf{v} = \mathbf{x}W_v$
Query vectors: $\mathbf{q} = \mathbf{x}W_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

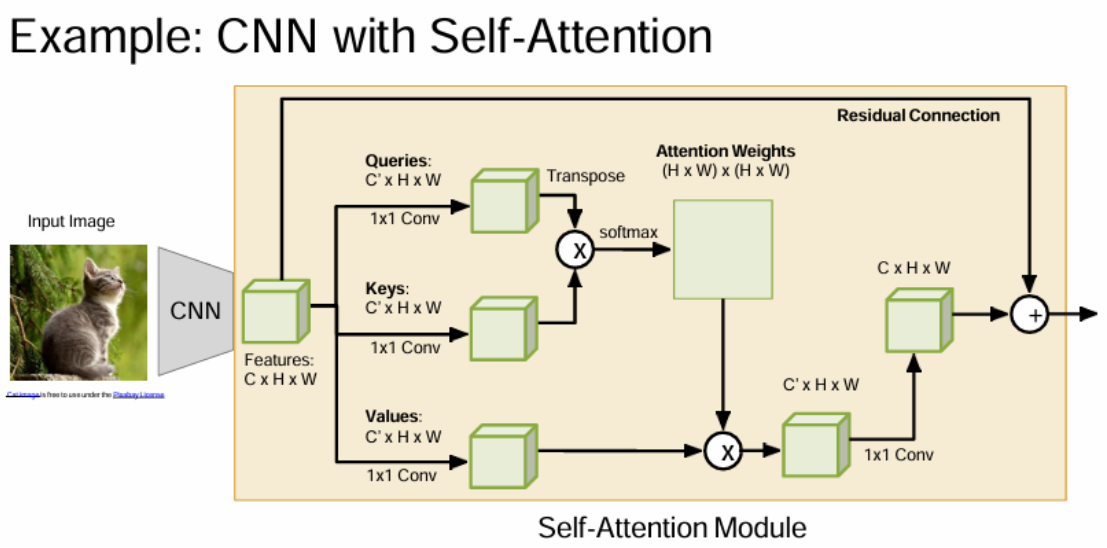
Inputs:
Input vectors: **x** (shape: N x D)



Self-attention layer doesn’ t care about the orders of the inputs!

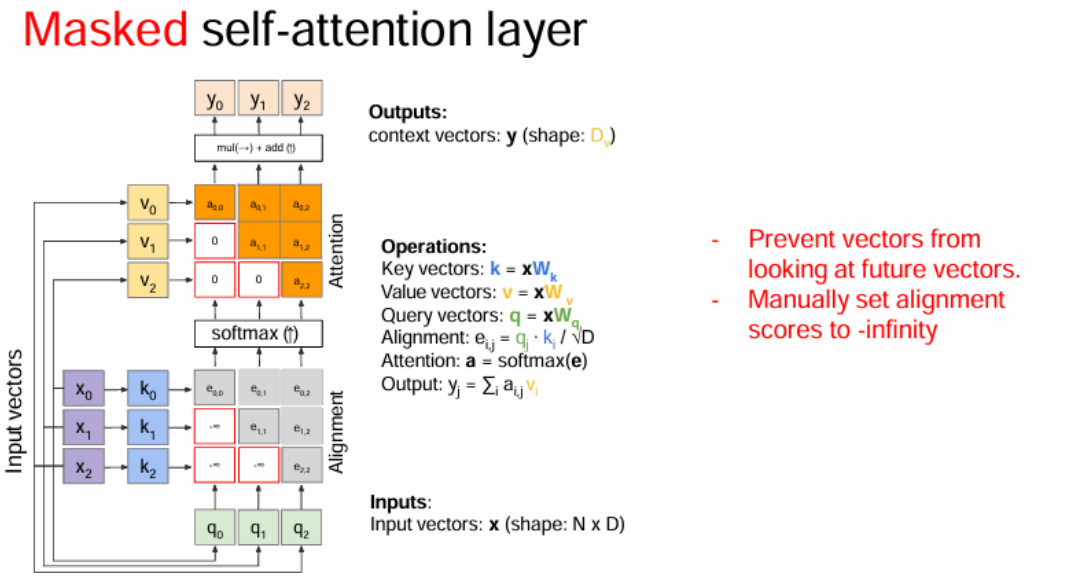
- Problem: How to encode ordered sequences like language adn spatially ordered img features?

- Solution: Positional encoding



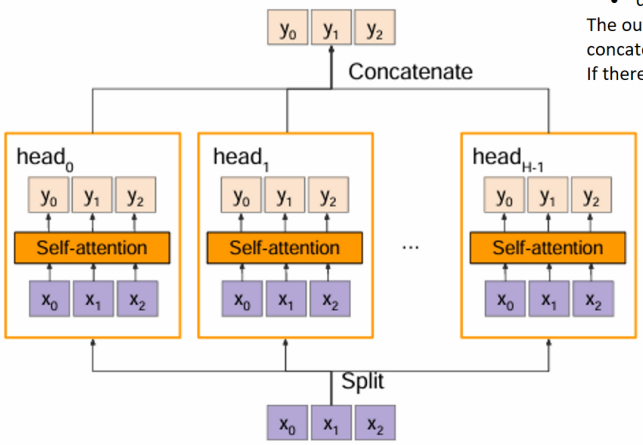
5. Transformers

1. Introduced in **"Attention Is All You Need" (Vaswani et al., 2017)**.
2. Transformers are a type of layer that uses self-attention and layer norm.
   - It is highly scalable and highly parallelizable
   - Faster training, larger models, better performance across vision and language tasks
3. Key Innovations: multi-head att self-att, positional encoding, masked att.



Multi-head self-attention layer

- Multiple self-attention heads in parallel



Each attention head produces an output of shape (N,dk) where:

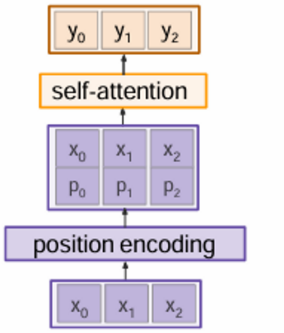
- N: Sequence length (number of tokens).
- dk: Dimension of the head's output.

The outputs of all heads (e.g., head0,head1,…,headH−1) are concatenated along the **feature dimension**.
If there are H heads, the concatenated output has shape (N,H·dk)

Options for *pos*(.)

1. Learn a lookup table:
   - Learn parameters to use for *pos*(t) for t ∈[0, T)
   - Lookup table contains T x d parameters.
2. Design a fixed function with the desiderata

Positional encoding



Concatenate special positional encoding p_j to each input vector x_j

We use a function *pos*: N →R^d to process the position j of the vector into a d-dimensional vector

So, p_j = *pos*(j)

Intuition:

$$p(t) = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \vdots \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_d \quad \text{where } \omega_k = \frac{1}{10000^{2k/d}}$$

0:	0	0	0	0	8:	1	0	0	0
1:	0	0	0	1	9:	1	0	0	1
2:	0	0	1	0	10:	1	0	1	0
3:	0	0	1	1	11:	1	0	1	1
4:	0	1	0	0	12:	1	1	0	0
5:	0	1	0	1	13:	1	1	0	1
6:	0	1	1	0	14:	1	1	1	0
7:	0	1	1	1	15:	1	1	1	1

Vaswani et al, "Attention is all you need", NeurIPS 2017

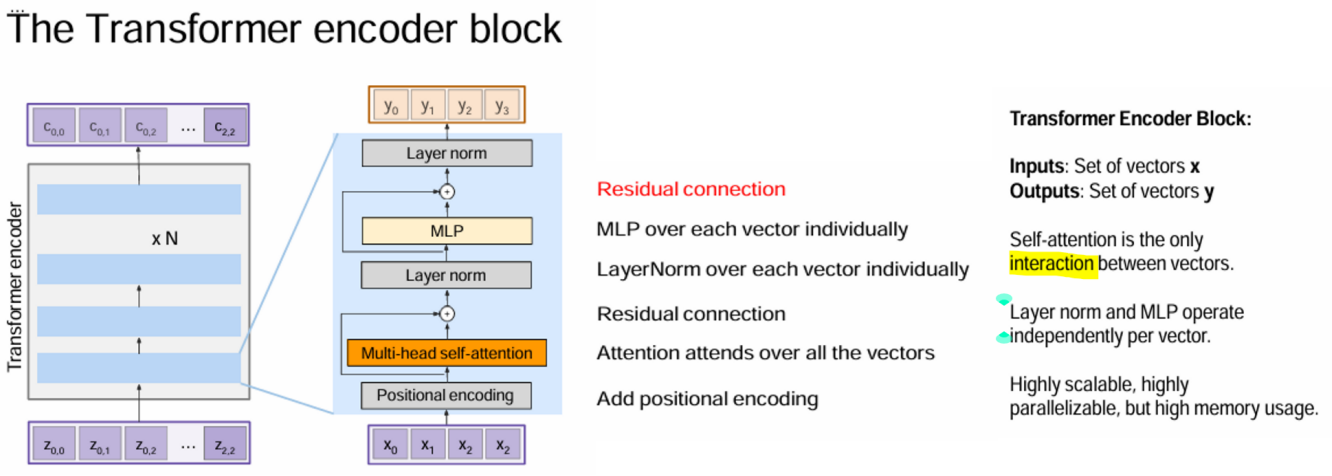
things desired

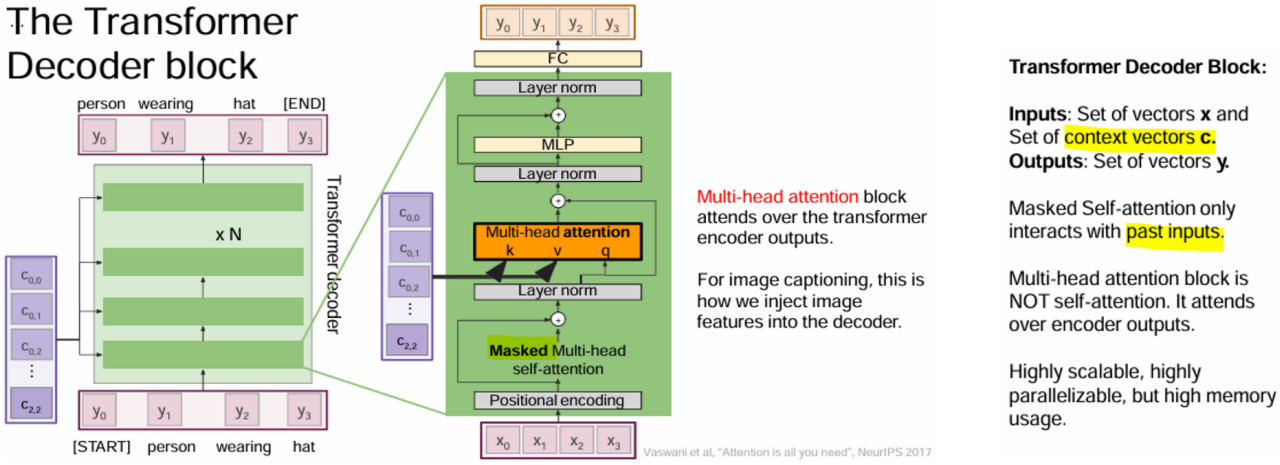
Desiderata of *pos*(.) :

1. It should output a **unique** encoding for each time-step (word's position in a sentence)
2. **Distance** between any two time-steps should be consistent across sentences with different lengths.
3. Our model should generalize to **longer** sentences without any efforts. Its values should be bounded.
4. It must be **deterministic**.

Vaswani et al, "Attention is all you need", NeurIPS 2017

The output or behavior can be exactly predicted based on the input and the defined rules.





Comparing RNNs to Transformer

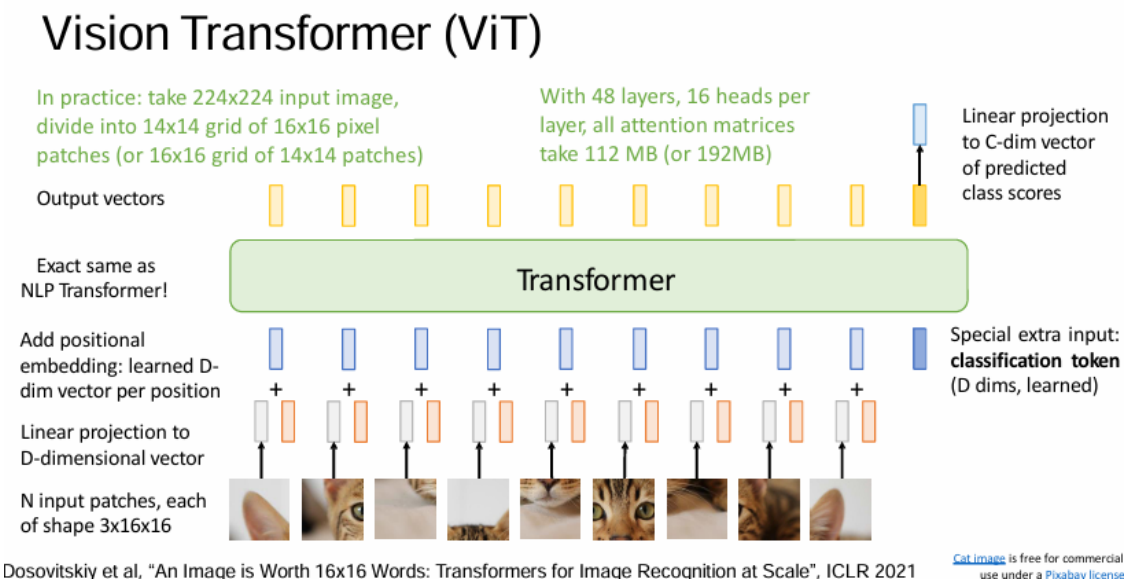
RNNs
(+) LSTMs work reasonably well for long sequences.
(-) Expects an ordered sequences of inputs
(-) Sequential computation: subsequent hidden states can only be computed after the previous ones are done.

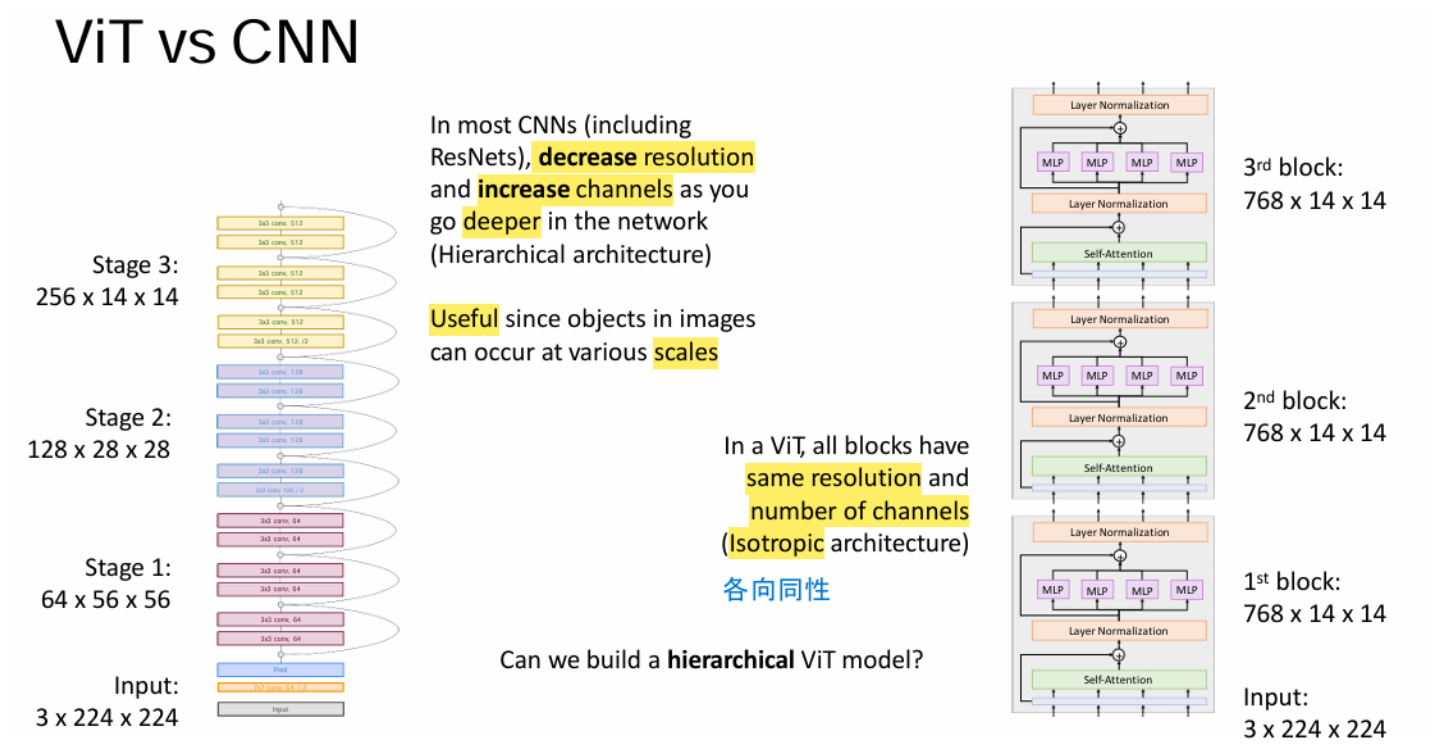
Transformer:
(+) Good at long sequences. Each attention calculation looks at all inputs.
(+) Can operate over unordered sets or ordered sequences with positional encodings.
(+) Parallel computation: All alignment and attention scores for all inputs can be done in parallel.
(-) Requires a lot of memory: N x M alignment and attention scalers need to be calculated and stored for a single self-attention head. (but GPUs are getting bigger and better)

6. Vision Transformers (ViTs)

- Adapt Transformers for image processing.
- Split images into patches (e.g., (16 \times 16)) and treat each patch as a sequence token.

- Pretrained on large-scale image datasets, demonstrating competitive performance with CNNs.





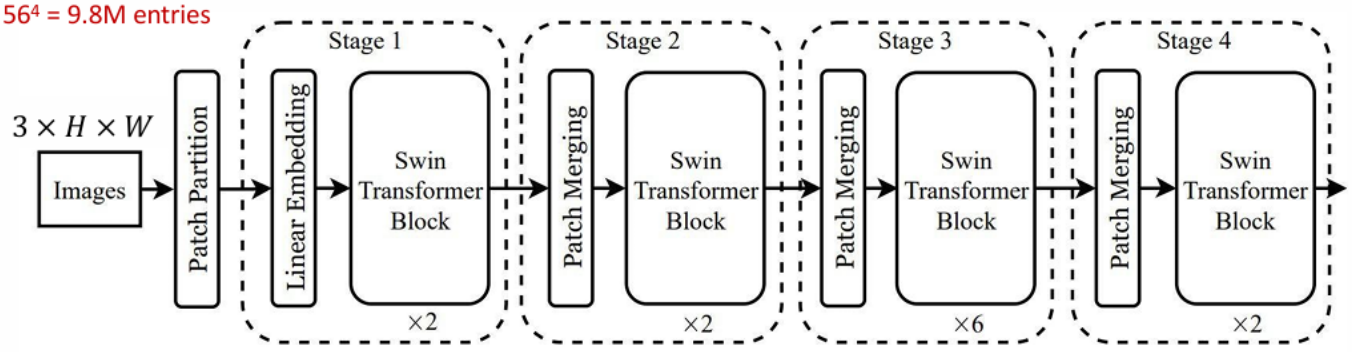
6.1 Hierarchical ViT:

MViT, Improved MViT, Swin (Swin Transformer: shifted window attention)
Swin:

Hierarchical ViT: Swin Transformer

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

$C \times \frac{H}{4} \times \frac{W}{4}$ $\quad 2C \times \frac{H}{8} \times \frac{W}{8}$ $\quad 4C \times \frac{H}{16} \times \frac{W}{16}$ $\quad 8C \times \frac{H}{32} \times \frac{W}{32}$



Solution: don't use full attention, instead use attention over patches

Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

Merge 2x2 neighborhoods; now patches are (effectively) 16x16

Merge 2x2 neighborhoods; now patches are (effectively) 32x32

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

Problem: tokens only interact with other tokens within the same window; no communication across windows

With H x W grid of tokens, each attention matrix is H^2W^2 – **quadratic** in image size

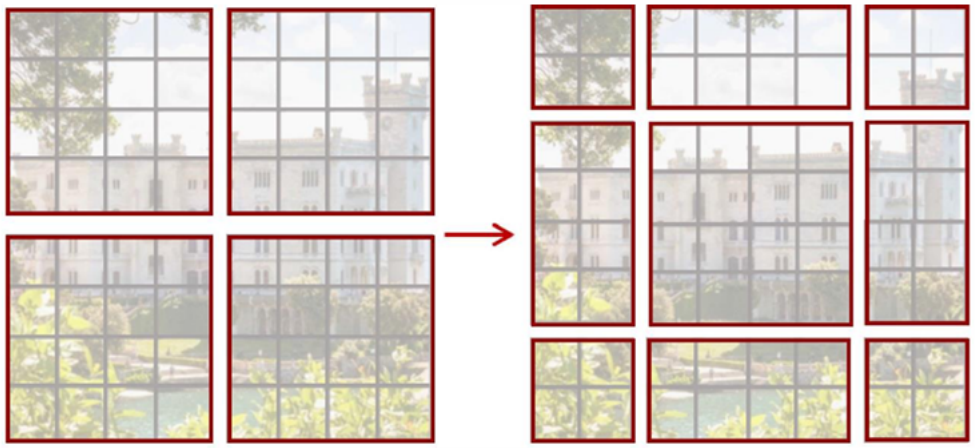
Rather than allowing each token to attend to all other tokens, instead divide into **windows** of M x M tokens (here M=4); only compute attention within each window

Total size of all attention matrices is now: $M^4(H/M)(W/M) = M^2HW$

Linear in image size for fixed M!
Swin uses M=7 throughout the network

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

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



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

Detail: Relative Positional Bias

ViT adds positional embedding to input tokens, encodes *absolute position* of each token in the image

Swin does not use positional embeddings, instead encodes *relative position* between patches when computing attention:

Attention with relative bias:

$$A = Softmax\left(\frac{QK^T}{\sqrt{D}} + B\right)V$$

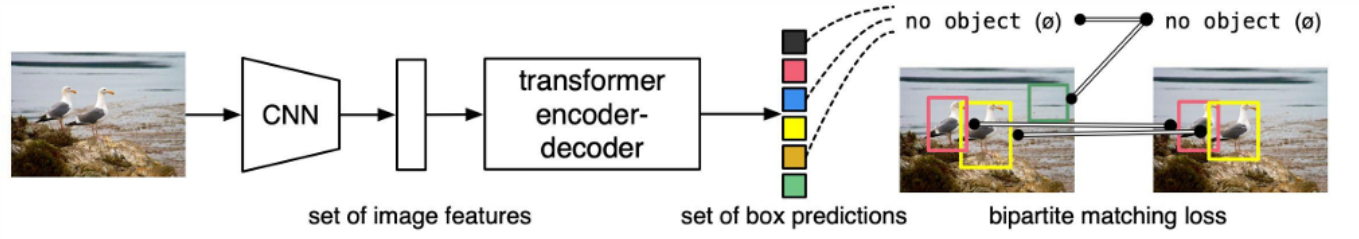
$Q, K, V: M^2 \times D$ (Query, Key, Value)

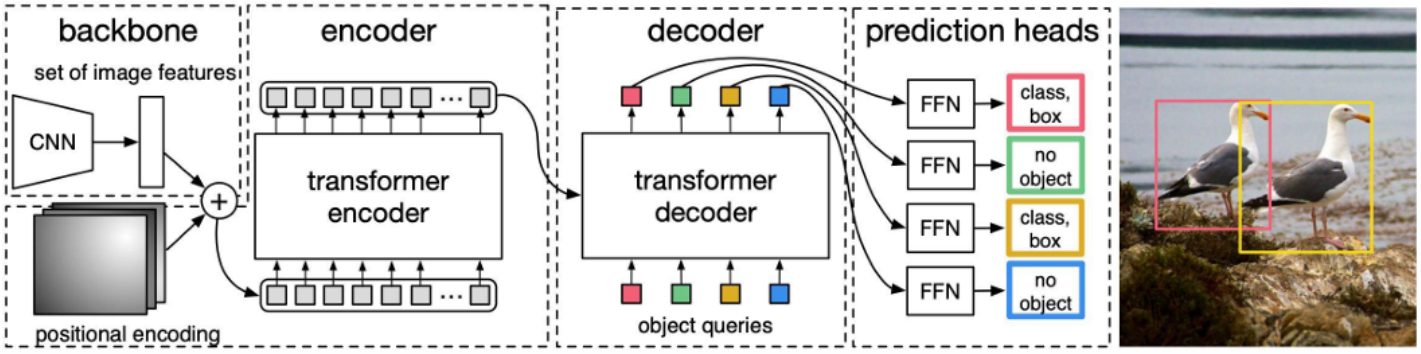
$B: M^2 \times M^2$ (learned biases)

7. Obj detection with Transformers: DETR

Simple object detection pipeline: directly output a set of boxes from a Transformer
Noanchors, no regression of box transforms
Match predicted boxes to GT boxes with bipartite matching; train to regress box coordinates

Object Detection with Transformers: DETR





rion et al, "End-to-End Object Detection with Transformers", ECCV 2020

Architecture of DETR:

1. Backbone (Feature Extraction):
   A CNN (e.g., ResNet) is used as a backbone to extract feature maps from the input image.
2. Transformer Encoder:
   Processes the flattened feature maps and encodes the global context of the image.
3. Transformer Decoder:
   Takes learned positional embeddings (object queries) and interacts with the encoder outputs to predict objects.
4. Prediction Heads:

- Each decoder output predicts:
  - Class label: The category of the object.
  - Bounding box: The coordinates of the object's location, represented as normalized values.

+:

1. End-to-End Training: Removes the need for complex heuristics like anchors, proposals, or NMS.
2. Global context:transformer captures global relationships in the image
3. flexibility: easily extended to other tasks like panoptic (全景式) segmentation.

-:

1. Slow Convergence
2. High Computational Cost
3. Localization Precision: Bounding box regression can sometimes be less precise compared to traditional methods.

References

https://cs231n.stanford.edu/slides/2022/lecture_10_ruohan.pdf
https://cs231n.stanford.edu/slides/2022/lecture_11_ruohan.pdf