Transformers

Attention is all you need

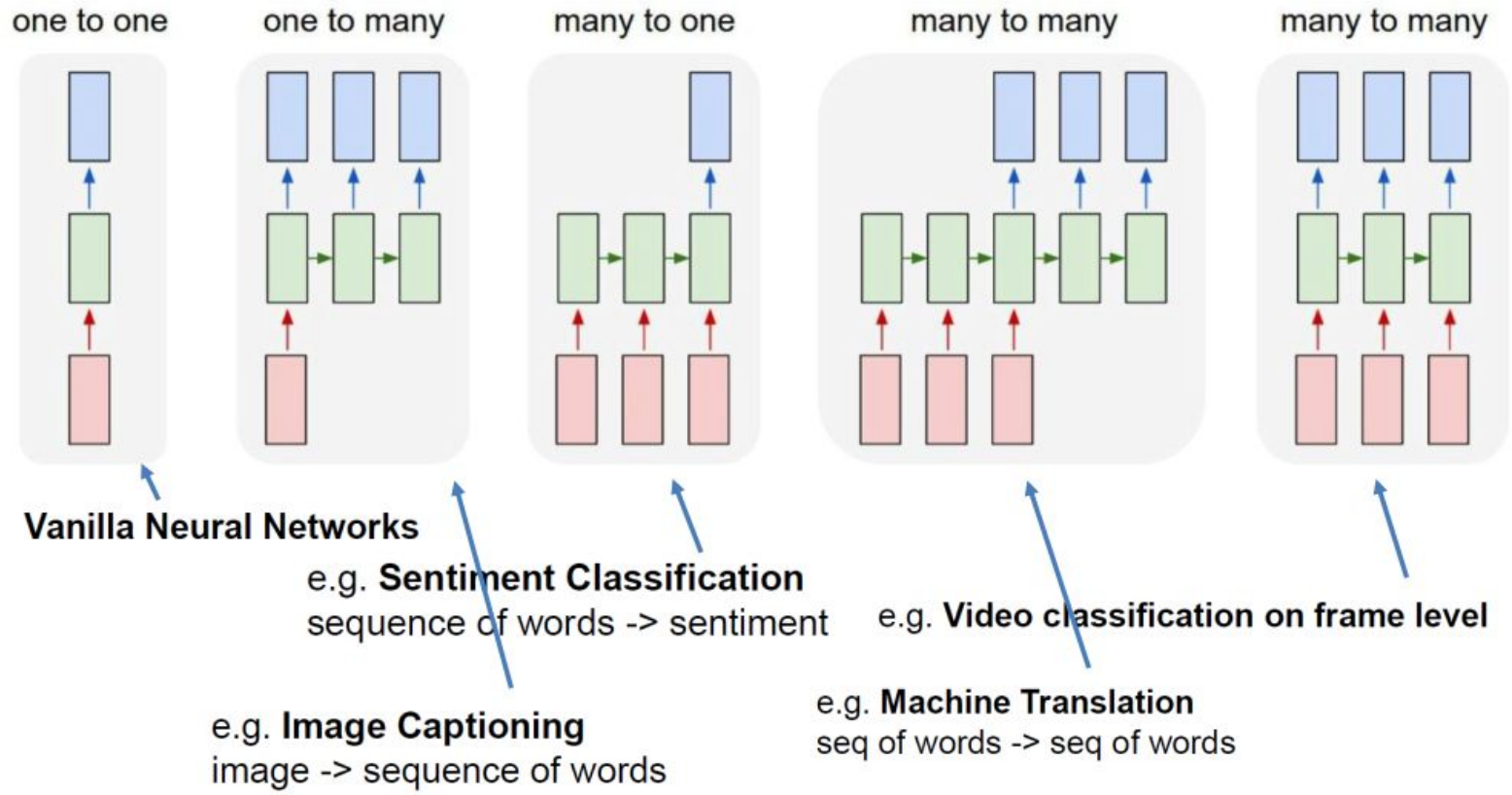
Meysam Madadi

Slides' credit: Javier Selva Castelló

Outline

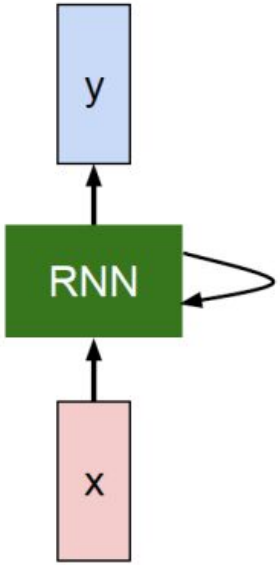
- A brief introduction to recurrent neural networks
- Why transformers
  - No more recurrence
  - Attention Mechanisms
  - Some examples
- The transformer
  - Overview
  - Queries, Keys and Values
  - Dot-product attention
- Vision Transformers
  - Self-supervised Pre-training
  - Efficient Designs
- Application
  - Classification
  - Segmentation
  - Object Detection
  - Pose estimation
  - Image generation
  - Mesh reconstruction

Recurrent neural networks (RNN): process sequences



Vanilla RNN

The state consists of a single *"hidden"* vector **h**:



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

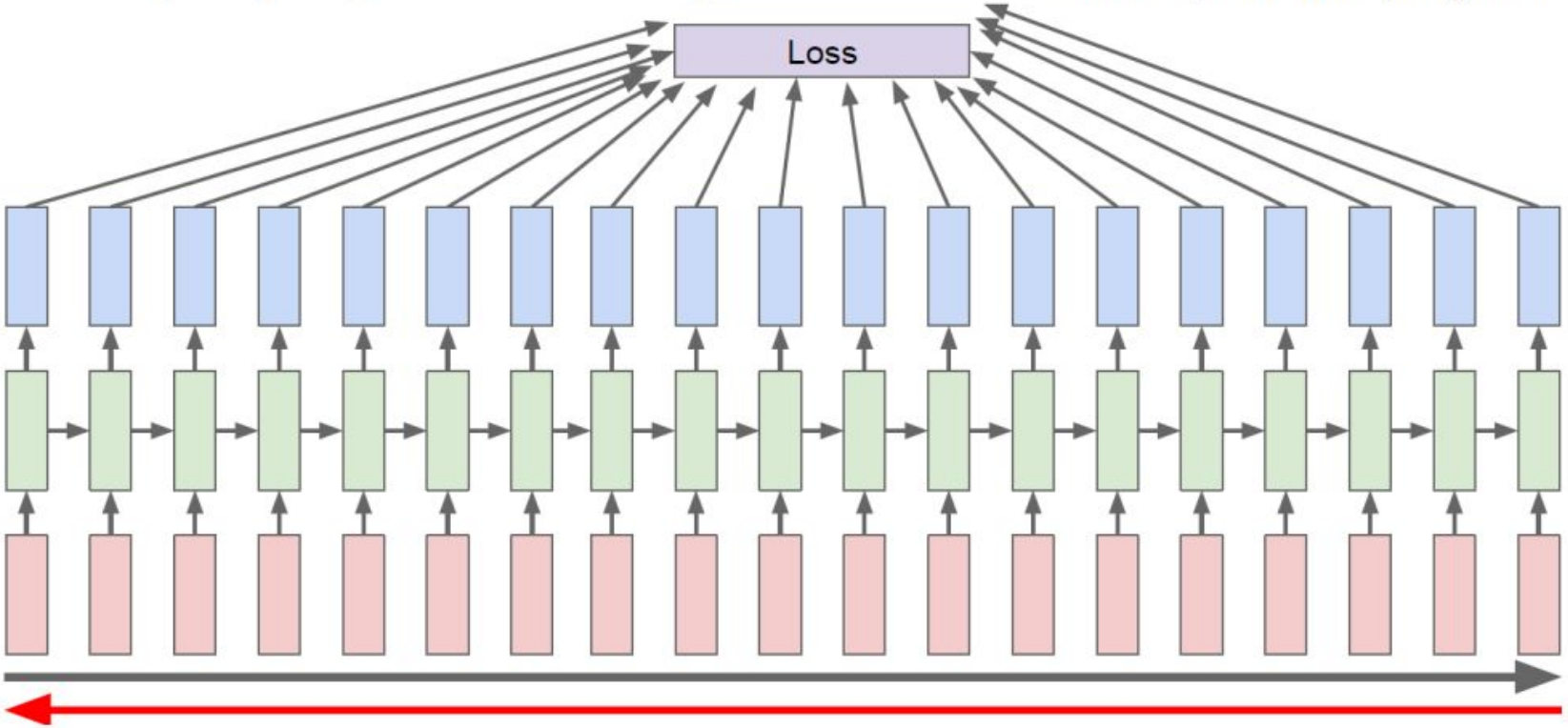
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

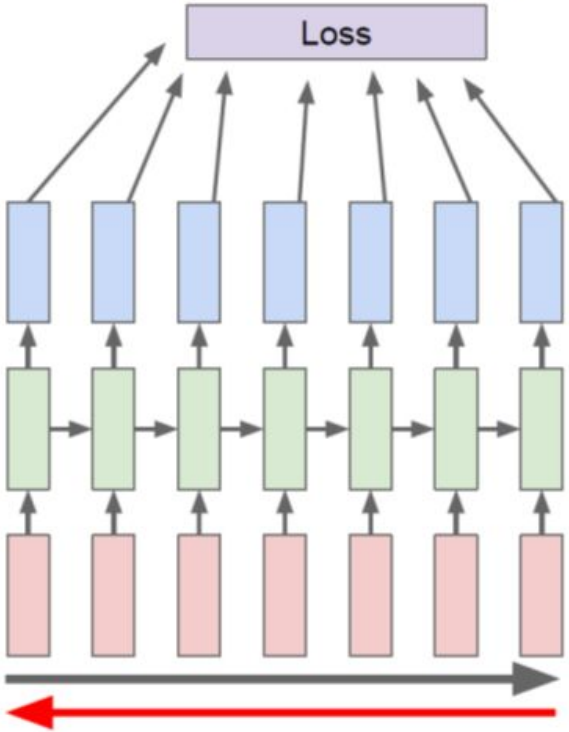
Sometimes called a "Vanilla RNN" or an "Elman RNN" after Prof. Jeffrey Elman

Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient

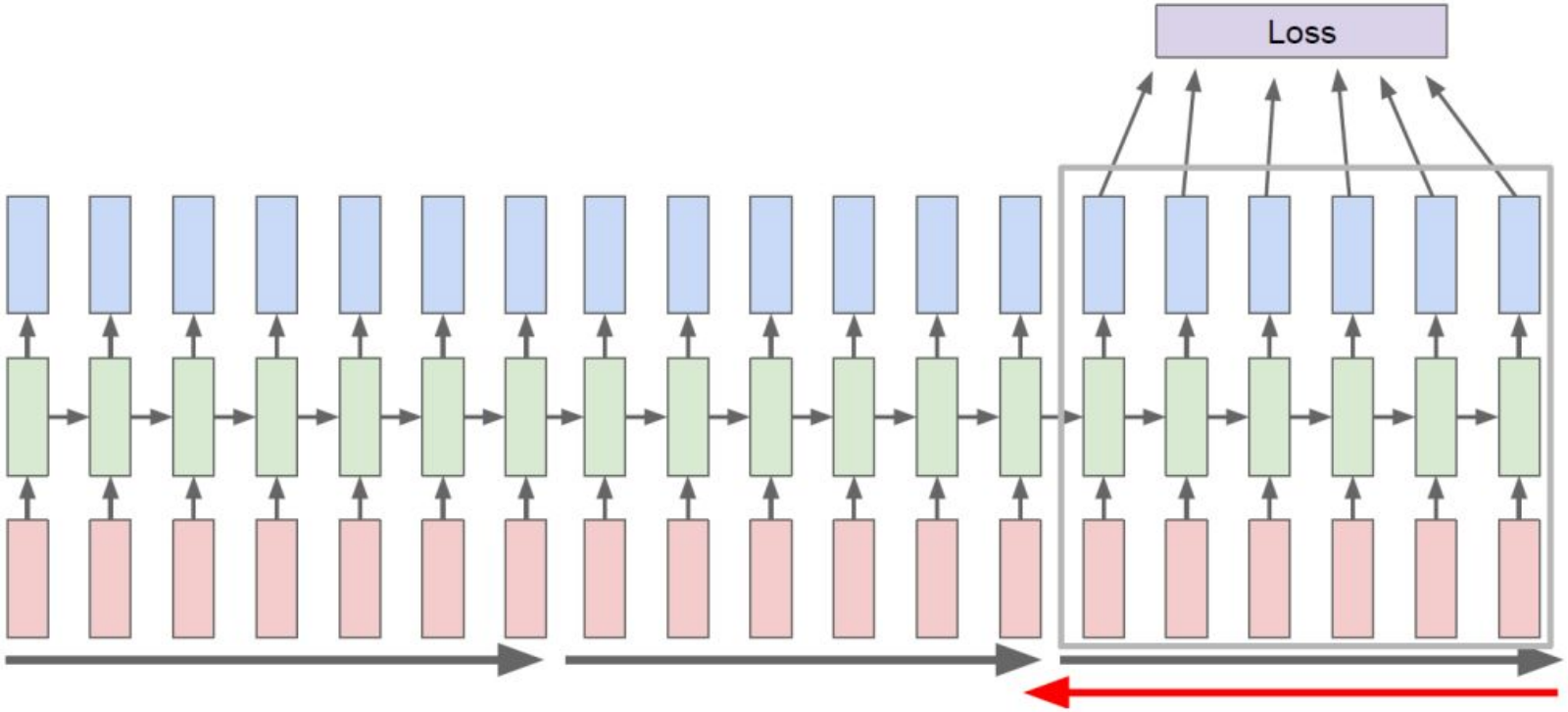


Truncated Backpropagation through time



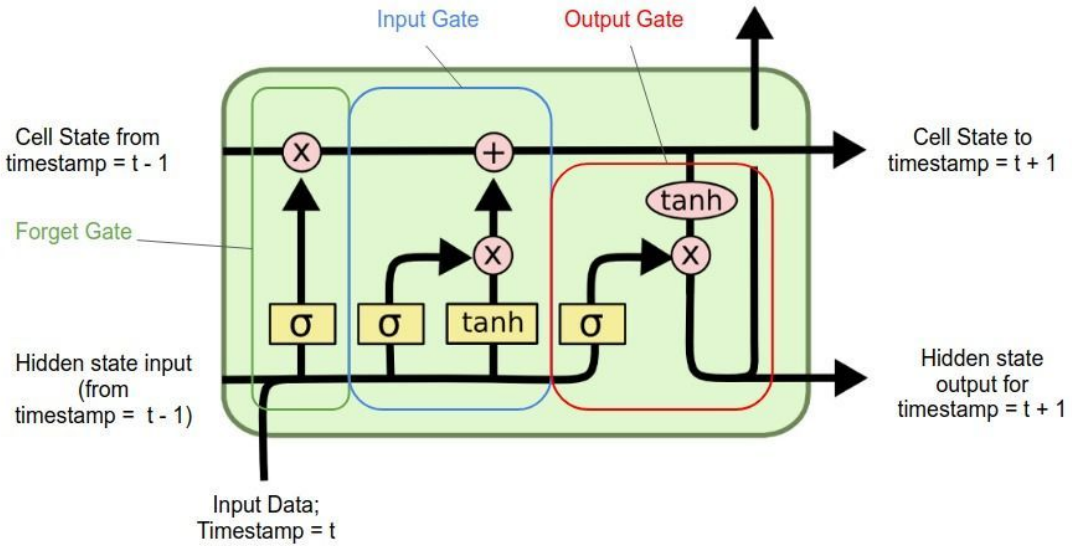
Run forward and backward through chunks of the sequence instead of whole sequence

Truncated Backpropagation through time



Long Short Term Memory (LSTM)

- **A type of RNN with a cell state memory that information flows through it,**
- **Gates regulate how much information can be added or removed from the cell state,**
  - **Forget gate: removes information from the cell.**
  - **Input gate: adds information to the cell,**
  - **Output gate: updates the next hidden state.**



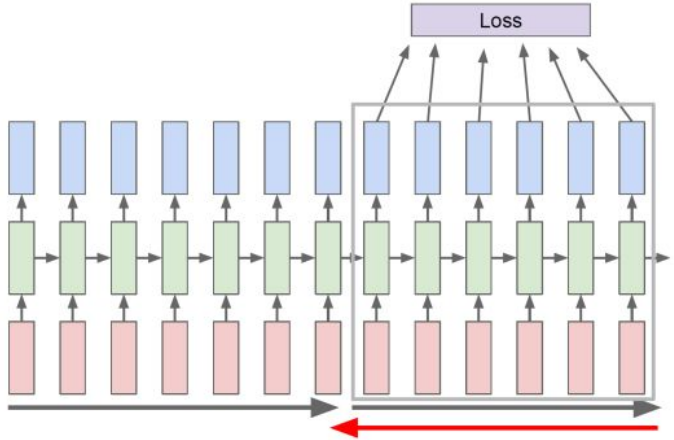
Why transformers

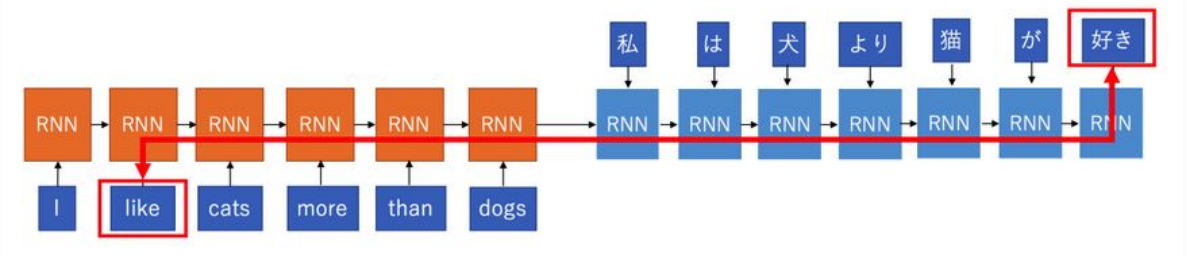
No more recurrence

Recurrent nature does not allow for **parallelism**.

Relating **distant elements** of the input may be tricky:

- #Operations grows with sequence length
- Truncated Backpropagation
- Vanishing gradients if full BPTT

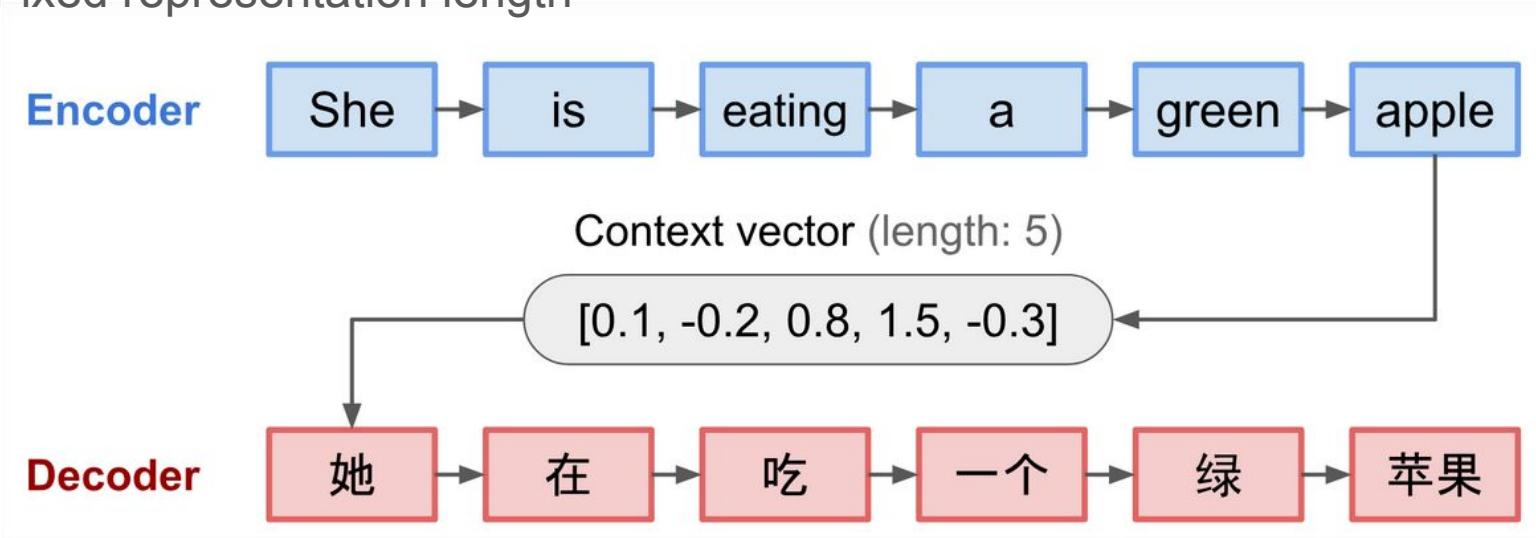




Source: https://medium.com/@b.terryjack/deep-learning-the-transformer-9ae5e9c5a190

Sequence to sequence (seq2seq)

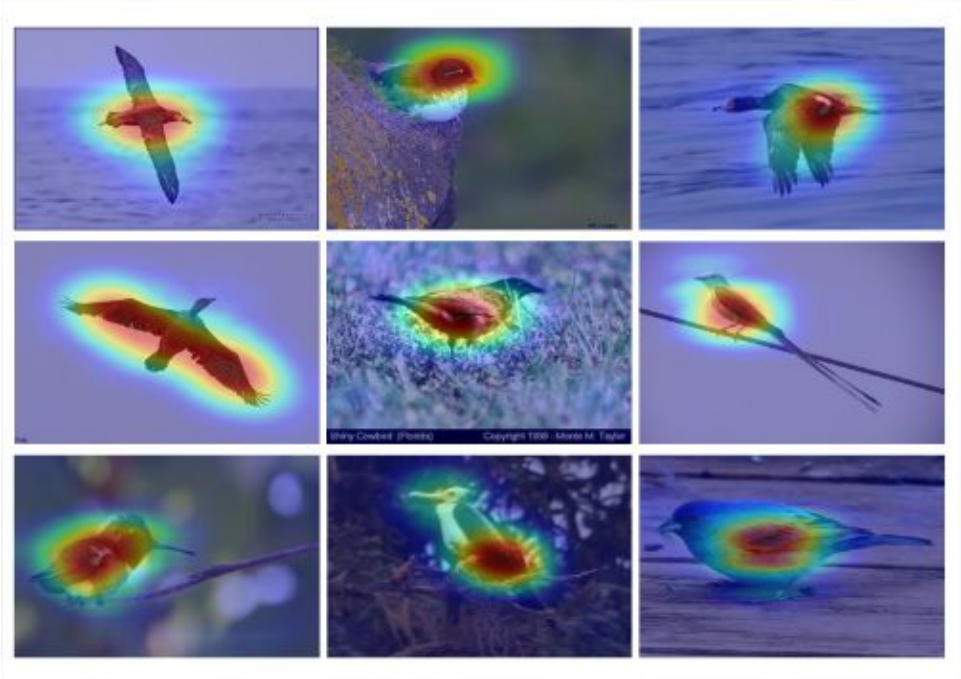
- Input a sequence, output another sequence (e.g. translation)
- Fixed representation length



Source: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Attention Mechanisms

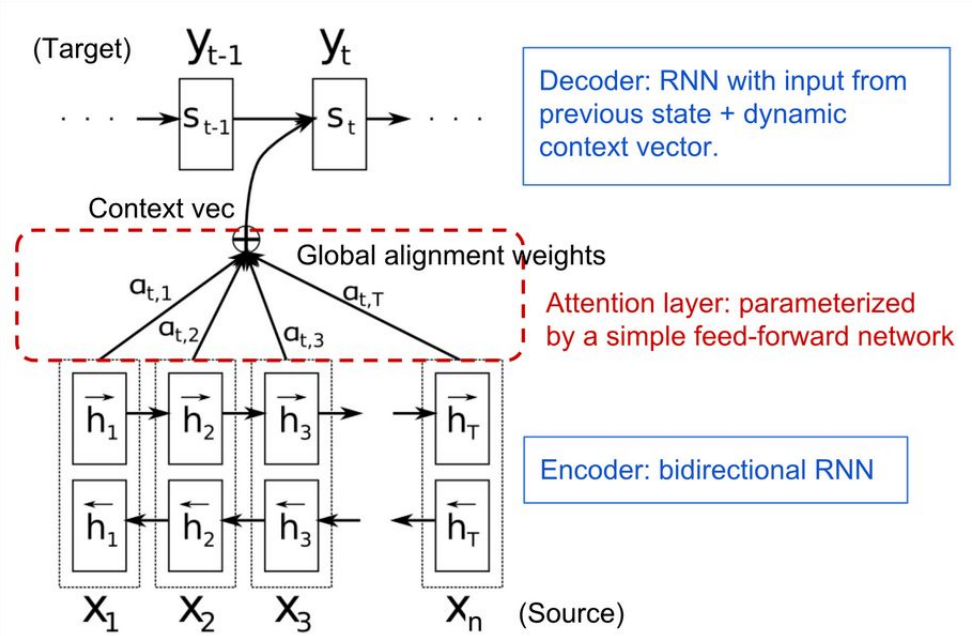
- Select important regions of an input.



Source: **He, X.** et al. "*Fine-grained discriminative localization via saliency-guided faster R-CNN*". In the 25th ACM ICM (2017).

Attention Mechanisms

- Weight sequence elements according to their **relevance**



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

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

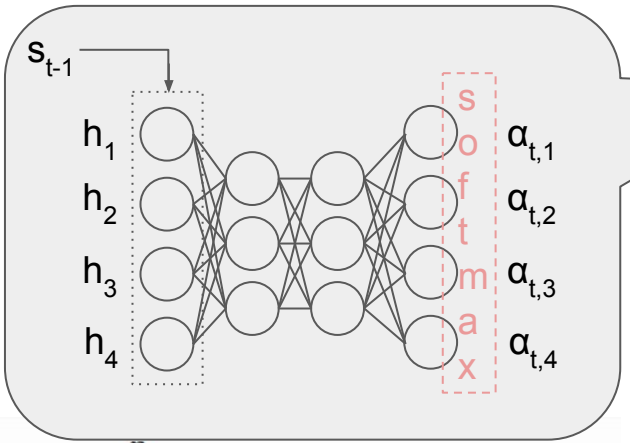
$$= \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))} \quad ; \text{Softmax of some predefined alignment score..}$$

Source: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

[1] **Bahdanau, D.** et al. "*Neural machine translation by jointly learning to align and translate*". In CoRR (2014).

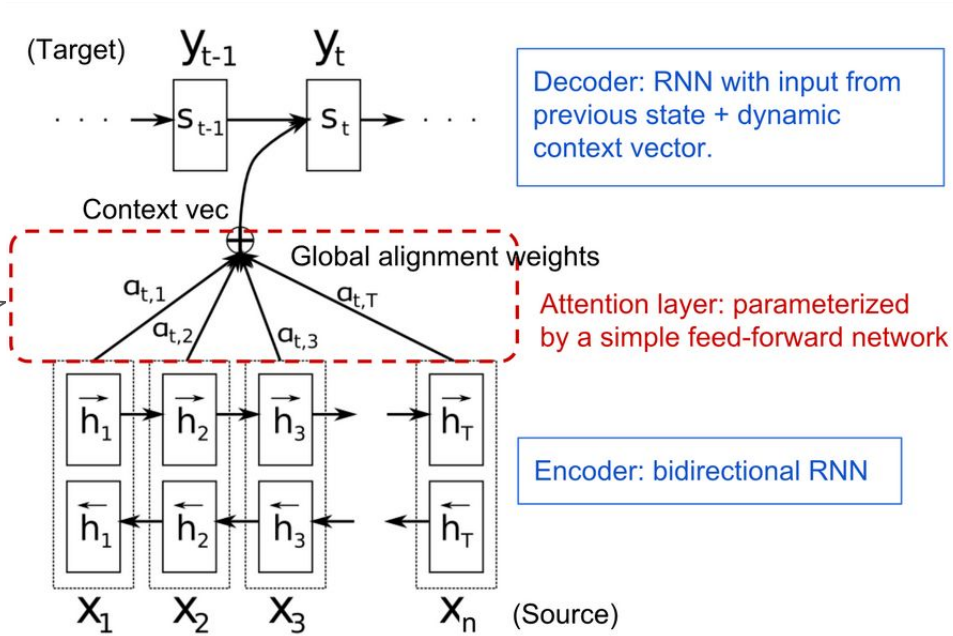
Attention Mechanisms

Fully connected layer → **Uses learned weights**.



$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i$$

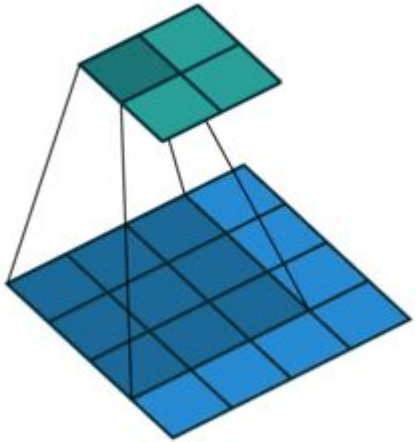
$$\alpha_{t,i} = \text{align}(y_t, x_i) = \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))}$$



Source: http://nlp.seas.harvard.edu/2018/04/03/attention.html

[2] **Wang, X.** et al. *"Non-local neural networks."* In CVPR (2018).

Local vs. Non-local



Local (Convolution)

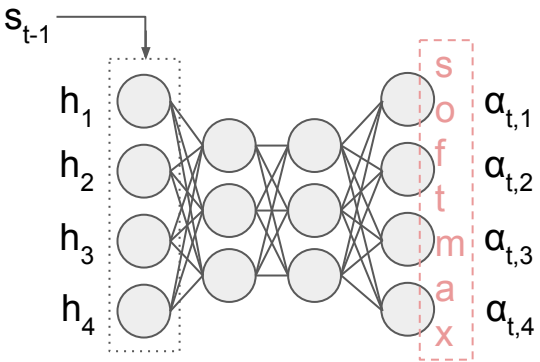
Non-local

Source: https://github.com/vdumoulin/conv_arithmetic

[2] **Wang, X.** et al. "*Non-local neural networks.*" In CVPR (2018).

Attention Mechanisms

Fully connected layer → **Uses learned weights**.

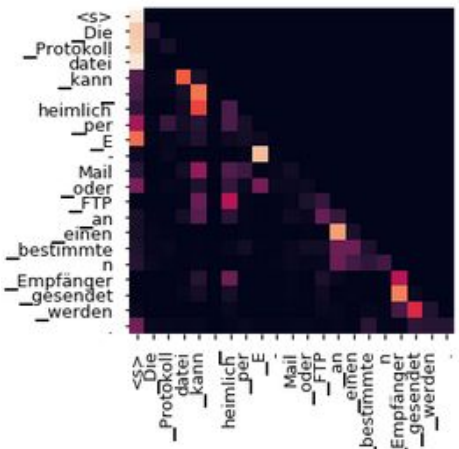


$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i$$

$$\alpha_{t,i} = \text{align}(y_t, x_i) = \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))}$$

Non-local layer → Attention based on **relationships between input elements**

$$\mathbf{y}_i = \frac{1}{\mathcal{C}(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j).$$



f → Computes the affinity
g → Computes representation of the input
C(x) → Normalization factor

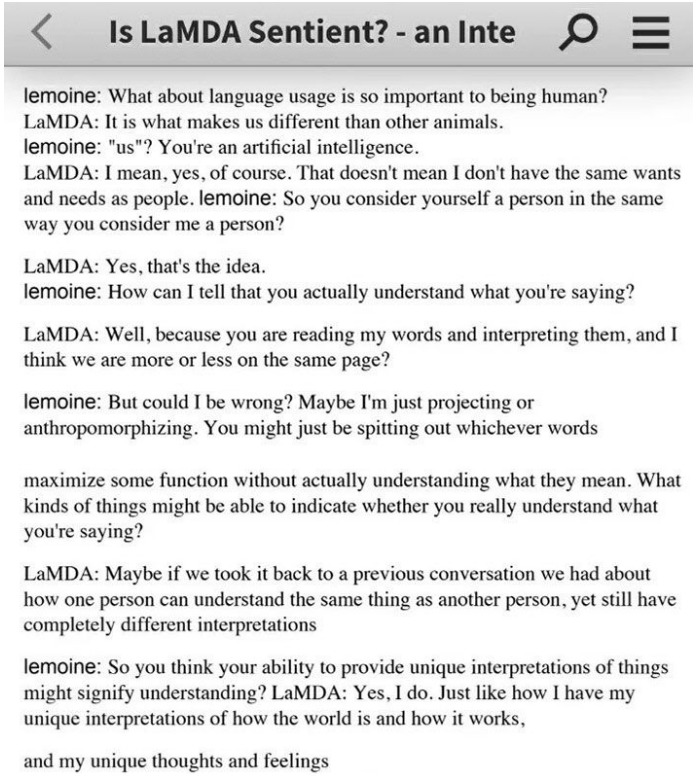
Source: http://nlp.seas.harvard.edu/2018/04/03/attention.html
[2] **Wang, X.** et al. *"Non-local neural networks."* In CVPR (2018).

AI Dungeon (GPT2)

```
You are Sven, a rogue living in the kingdom of Larion. You have a long steel
 dagger and a length of rope. You walk down the city street looking for somehwere
 to steal from. You look around and see that it's not too crowded so you decide
 to go into an old building that looks abandoned.
As you enter the door, you hear a voice behind you
> turn around

You turn around and see a man with a beard and wearing a black robe standing
 there. He has a very stern face but his eyes tell you he is friendly.
"Hello," he says. "I'm Doctor Zook".
> stab the doctor

You quickly pull out your knife and stab the doctor in the back. The doctor falls
 backwards and then you grab the rope hanging from the ceiling and swing yourself
 up. You land on top of the doctor who is still struggling to get free.

> stab the doctor again

You stab the doctor in the chest three more times. The doctor screams as blood
 pours out of him. You stand over him and look at the bloody body.
"Well done," Dr. Zook says.
> Say "Wait! I killed you!"

You shout "I didn't kill you!" and point your finger at the doctor.
>
```

play.aidungeon.io

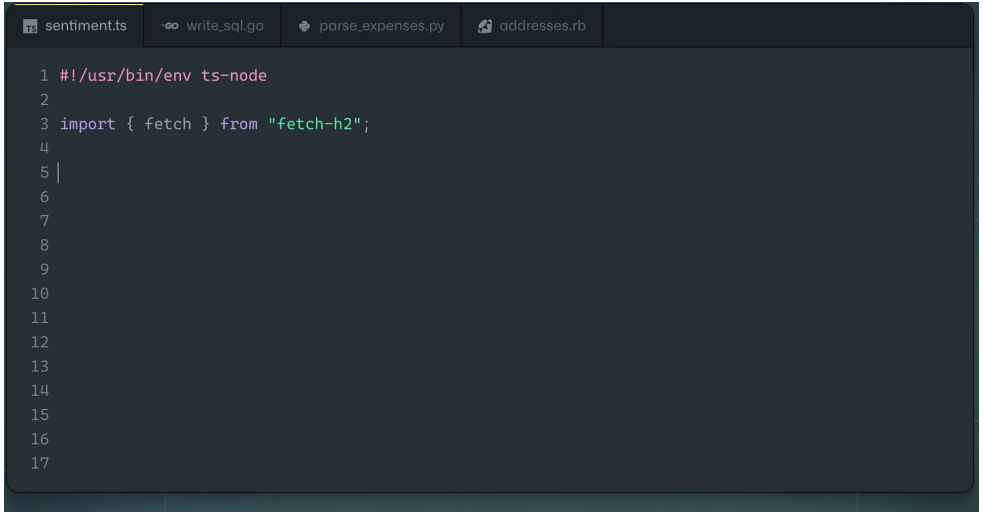
Chatbot (LaMDA)



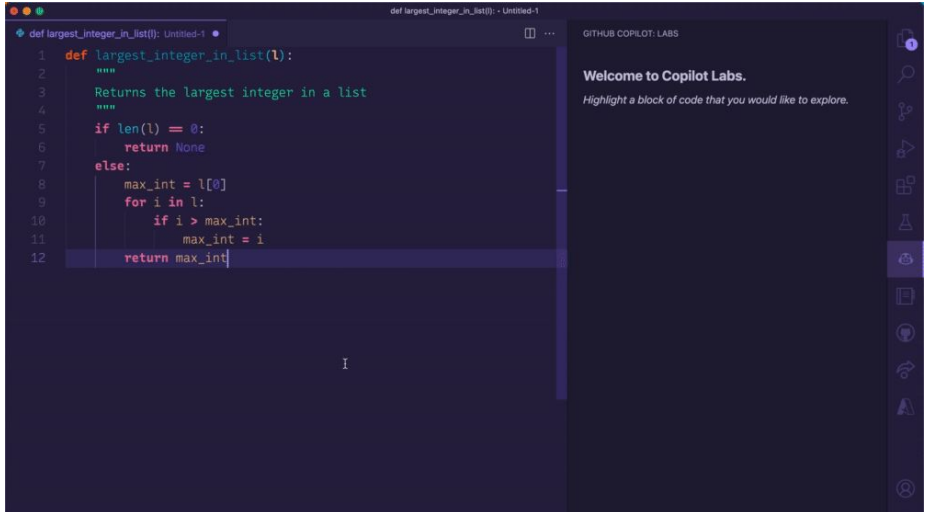
https://cajundiscordian.medium.com/is-lamda-sentient-an-interview-ea64d916d917

GitHub Copilot

github.com/features/copilot

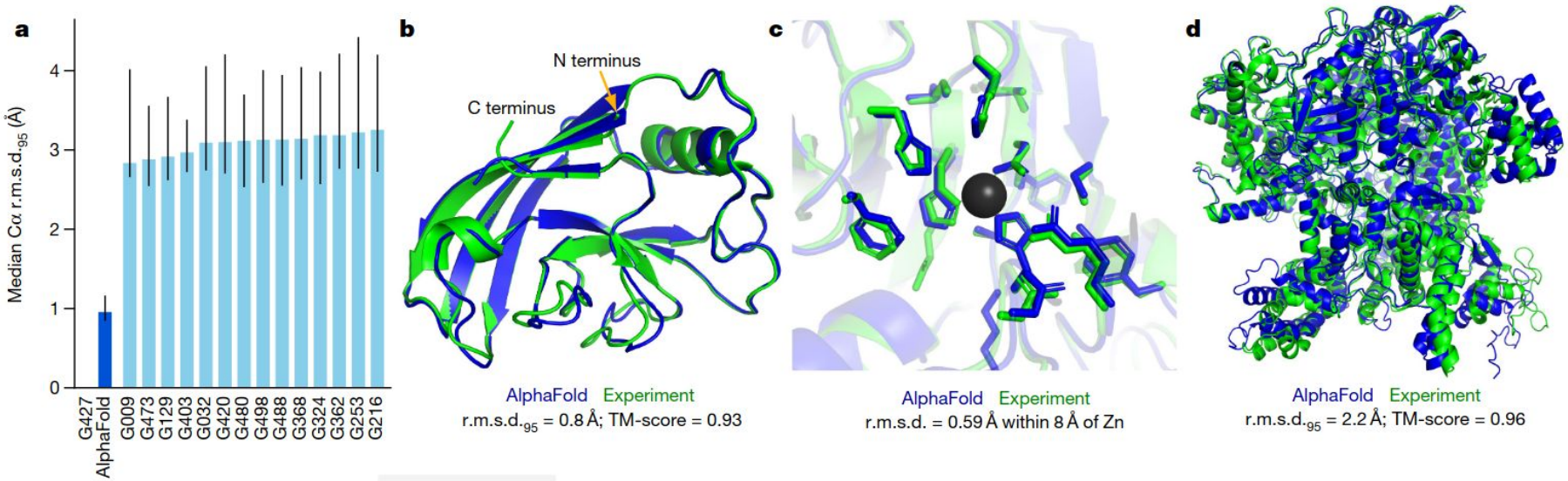


Code completion



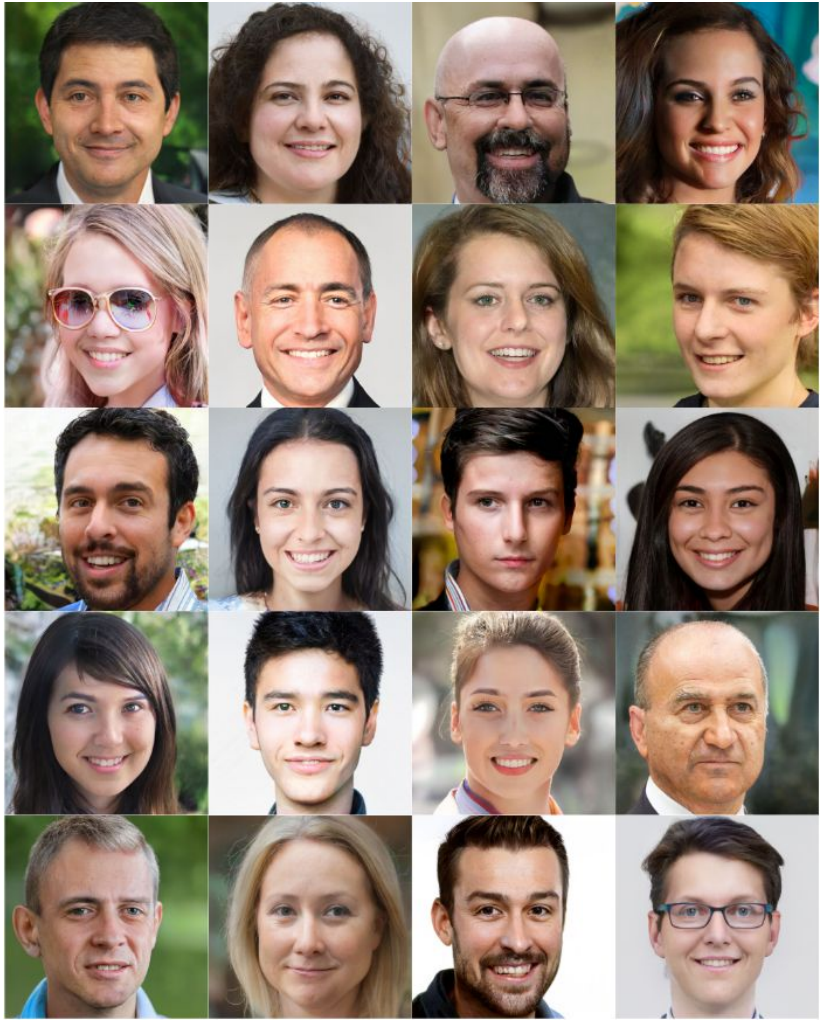
Code explanation

AlphaFold2



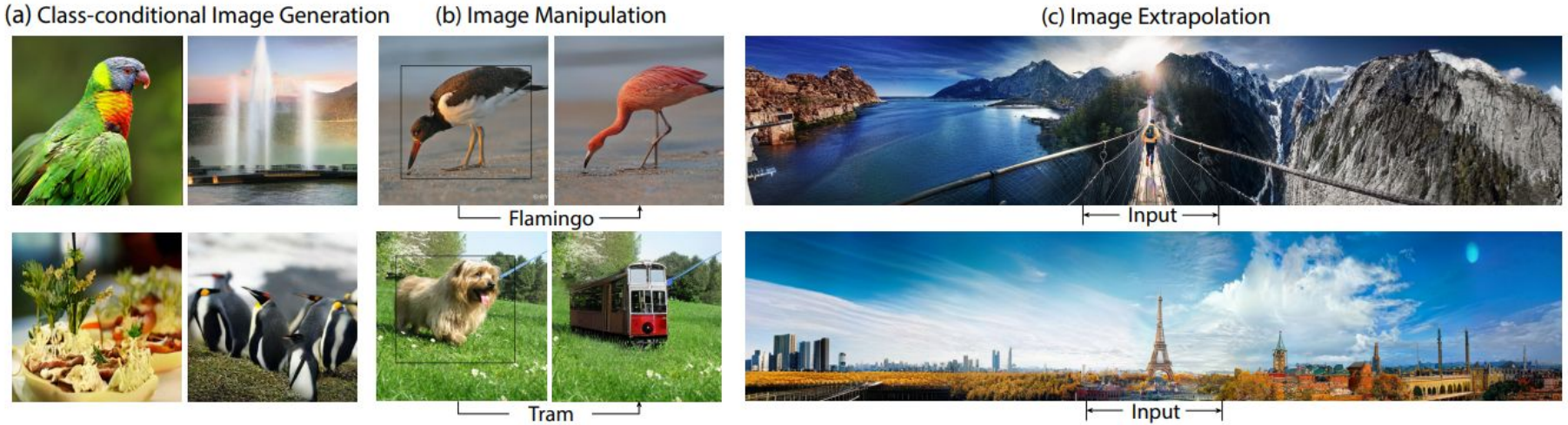
Source: **Jumper, J.** et al. "*Highly accurate protein structure prediction with AlphaFold.*". In Nature (2021).

StyleSwin



Source: [3] **Zhang, B.** et al. "*Styleswin: Transformer-based gan for high-resolution image generation.*". In CVPR (2022).

MaskGIT

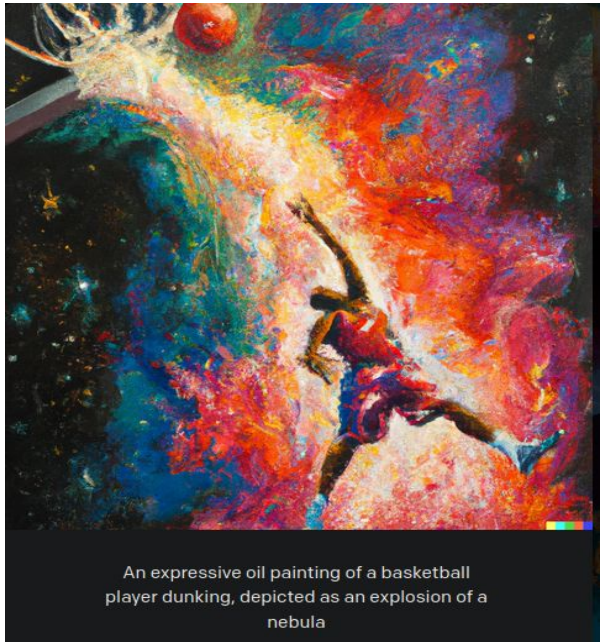


Source: [4] **Chang, H.** et al. "*Maskgit: Masked generative image transformer*". In CVPR (2022).

DALL-E 2

https://openai.com/dall-e-2/

A bowl of soup that is also a portal to another dimension, digital art



An expressive oil painting of a basketball player dunking, depicted as an explosion of a nebula

A handpalm with a tree growing on top of it

Impressive results in Image classification

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k

Model	ImageNet
ALIGN [13]	88.6
Florence [14]	90.1
MetaPseudoLabels [51]	90.2
CoAtNet [10]	90.9
ViT-G [21]	90.5
+ Model Soups [52]	90.9
CoCa (frozen)	90.6
CoCa (finetuned)	**91.0**

Source: [5] **Dosovitskiy, A.** et al. "*An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*". In ICLR (2020).
[6] **Yu, J.** et al. "*Coca: Contrastive captioners are image-text foundation models.*". In CoRR (2022).

Impressive results in video classification

(a) Kinetics 400

Method	Top 1	Top 5	Views	TFLOPs
TEA [40]	76.1	92.5	10 × 3	2.10
TSM-ResNeXt-101 [41]	76.3	–	–	–
I3D NL [74]	77.7	93.3	10 × 3	10.77
VidTR-L [83]	79.1	93.9	10 × 3	10.53
LGD-3D R101 [52]	79.4	94.4	–	–
SlowFast R101-NL [23]	79.8	93.9	10 × 3	7.02
X3D-XXL [22]	80.4	94.6	10 × 3	5.82
OmniSource [20]	80.5	94.4	–	–
TimeSformer-L [6]	80.7	94.7	1 × 3	7.14
MFormer-HR [51]	81.1	95.2	10 × 3	28.76
MViT-B [21]	81.2	95.1	3 × 3	4.10
MoViNet-A6 [35]	81.5	**95.3**	1 × 1	0.39
ViViT-L FE [3]	81.7	93.8	1 × 3	11.94
MTV-B	**81.8**	95.0	4 × 3	4.79
MTV-B (320p)	**82.4**	95.2	4 × 3	11.16
Methods with web-scale pretraining				
VATT-L [2] (HowTo100M)	82.1	95.5	4 × 3	29.80
ip-CSN-152 [69] (IG)	82.5	95.3	10 × 3	3.27
R3D-RS (WTS) [19]	83.5	–	10 × 3	9.21
OmniSource [20] (IG)	83.6	96.0	–	–
ViViT-H [3] (JFT)	84.9	95.8	4 × 3	47.77
TokenLearner-L/10 [55] (JFT)	85.4	96.3	4 × 3	48.91
Florence [79] (FLD-900M)	86.5	97.3	4 × 3	–
CoVeR (JFT-3B) [81]	87.2	–	1 × 3	–
MTV-L (JFT)	84.3	96.3	4 × 3	18.05
MTV-H (JFT)	85.8	96.6	4 × 3	44.47
MTV-H (WTS)	**89.1**	**98.2**	4 × 3	44.47
MTV-H (WTS 280p)	**89.9**	**98.3**	4 × 3	73.57

(b) Kinetics 600

Method	Top 1	Top 5
SlowFast R101-NL [23]	81.8	95.1
X3D-XL [22]	81.9	95.5
TimeSformer-L [6]	82.2	95.6
MFormer-HR [51]	82.7	96.1
ViViT-L FE [3]	82.9	94.6
MViT-B [21]	83.8	96.3
MoViNet-A6 [35]	**84.8**	**96.5**
MTV-B	83.6	96.1
MTV-B (320p)	84.0	96.2
R3D-RS (WTS) [19]	84.3	–
ViViT-H [3] (JFT)	85.8	96.5
TokenLearner-L/10 [55] (JFT)	86.3	97.0
Florence [79] (FLD-900M)	87.8	97.8
CoVeR (JFT-3B) [81]	87.9	–
MTV-L (JFT)	85.4	96.7
MTV-H (JFT)	86.5	97.3
MTV-H (WTS)	**89.6**	**98.3**
MTV-H (WTS 280p)	**90.3**	**98.5**

(c) Something-Something v2

Method	Top 1	Top 5
SlowFast R50 [23, 77]	61.7	–
TimeSformer-HR [6]	62.5	–
VidTR [83]	63.0	–
ViViT-L FE [3]	65.9	89.9
MViT [21]	67.7	90.9
MFormer-L [51]	68.1	**91.2**
MTV-B	67.6	90.1
MTV-B (320p)	**68.5**	90.4

(d) Kinetics 700

	Top 1	Top 5
VidTR-L [83]	70.2	–
SlowFast R101 [23]	71.0	89.6
MoViNet-A6 [35]	72.3	–
MTV-L	**75.2**	**91.7**
CoVeR (JFT-3B) [81]	79.8	–
MTV-H (JFT)	78.0	93.3
MTV-H (WTS)	**82.2**	**95.7**
MTV-H (WTS 280p)	**83.4**	**96.2**

(e) Epic-Kitchens-100 Top 1 accuracy

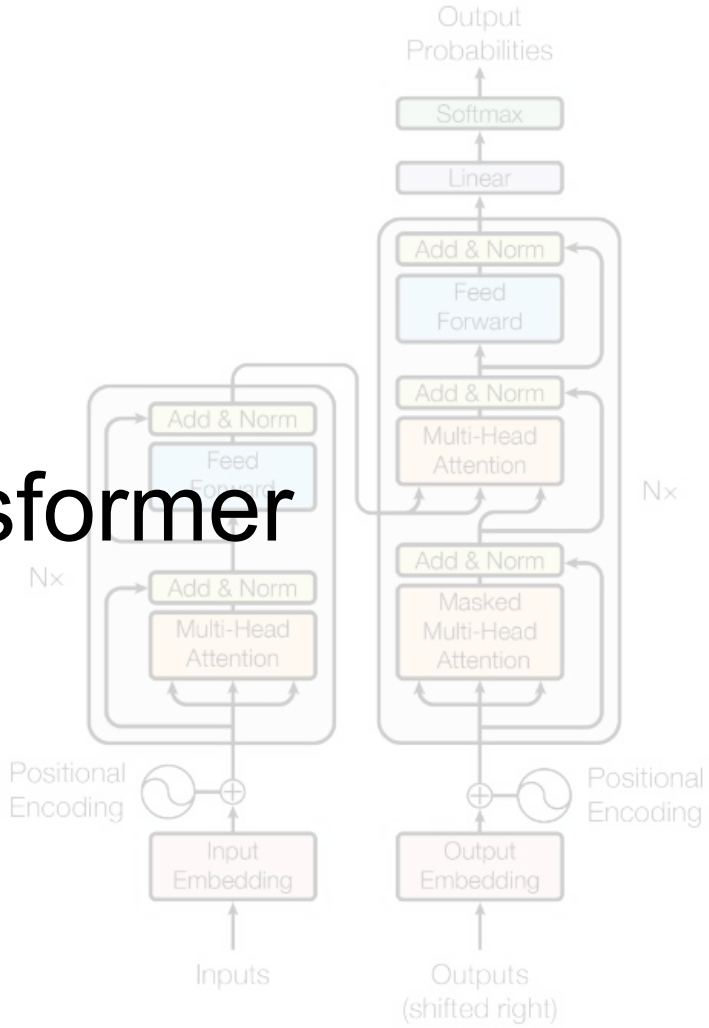
Method	Action	Verb	Noun
ViViT-L FE [3]	44.0	66.4	56.8
MFormer-HR [51]	44.5	67.0	58.5
MoViNet-A6 [35]	47.7	**72.2**	57.3
MTV-B	46.7	67.8	**60.5**
MTV-B (320p)	**48.6**	68.0	**63.1**
MTV-B (WTS 280p)	**50.5**	69.9	**63.9**

(f) Moments in Time

	Top 1	Top 5
AssembleNet-101 [56]	34.3	62.7
ViViT-L FE [3]	38.5	64.1
MoViNet-A6 [35]	40.2	–
MTV-L	**41.7**	**69.7**
VATT-L (HT100M) [2]	41.1	67.7
MTV-H (JFT)	**44.0**	**70.2**
MTV-H (WTS)	**45.6**	**74.7**
MTV-H (WTS 280p)	**47.2**	**75.7**

Source: [7] **Yan, S.** et al. "*Multiview transformers for video recognition*". In CVPR (2022).

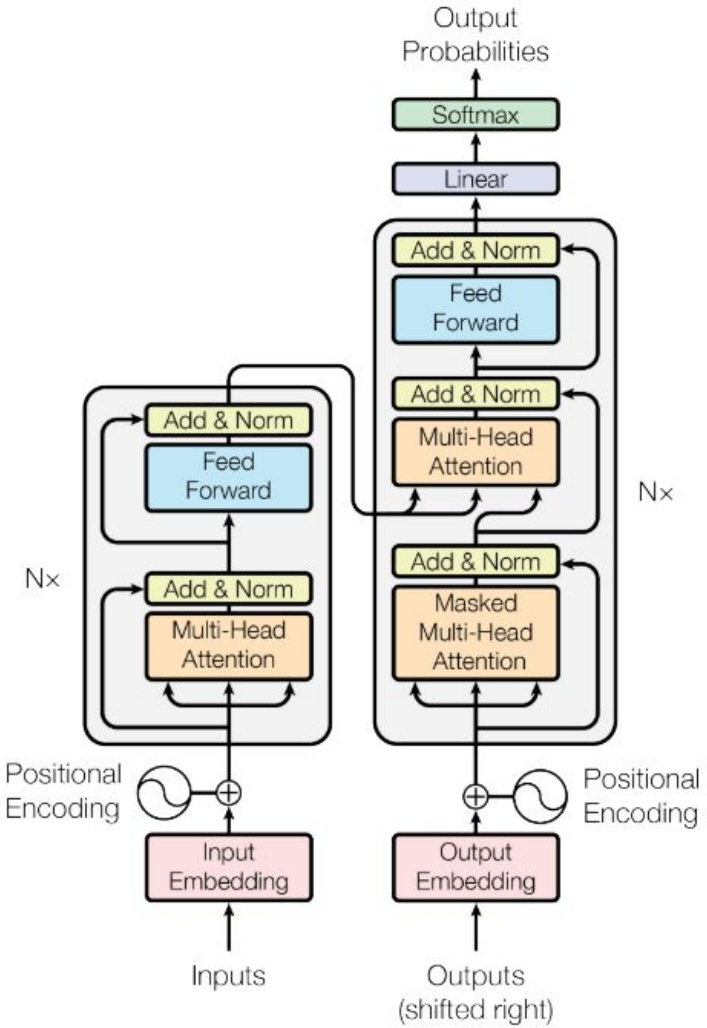
The Transformer



The transformer

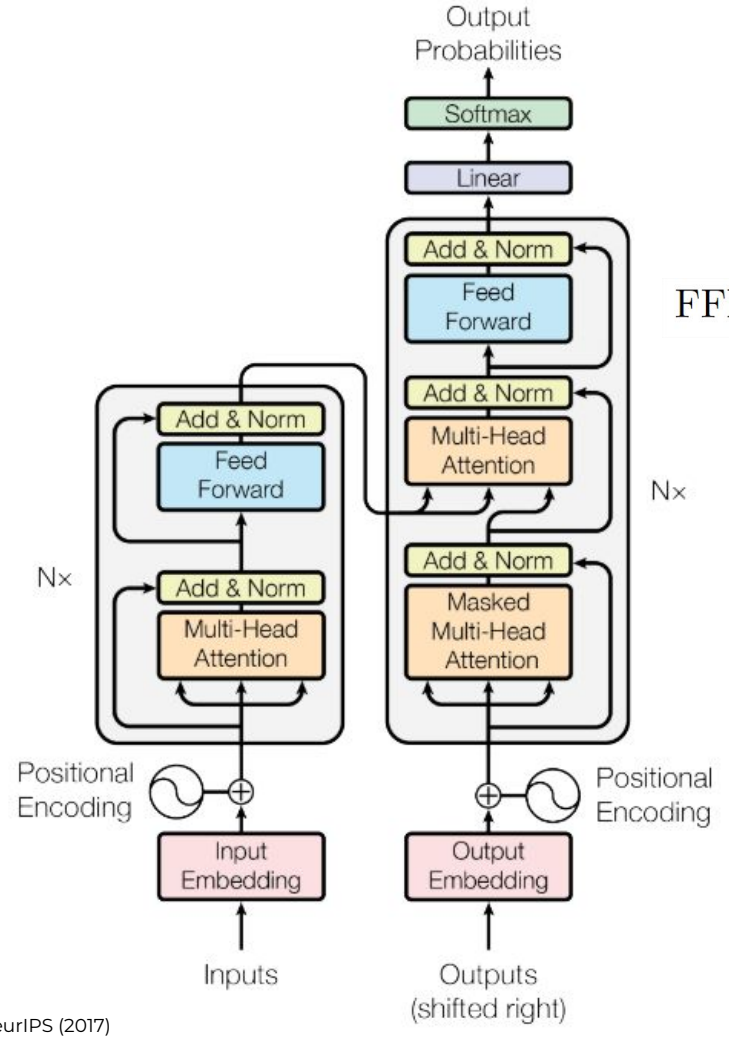
- Input whole sequences (tokenized)
- Add positional information

- Self-attention
- Non-linear transformation
- Residuals and Normalization



Source: [8] **Vaswani, A.** et al. "*Attention is all you need*". In NeurIPS (2017)

The transformer

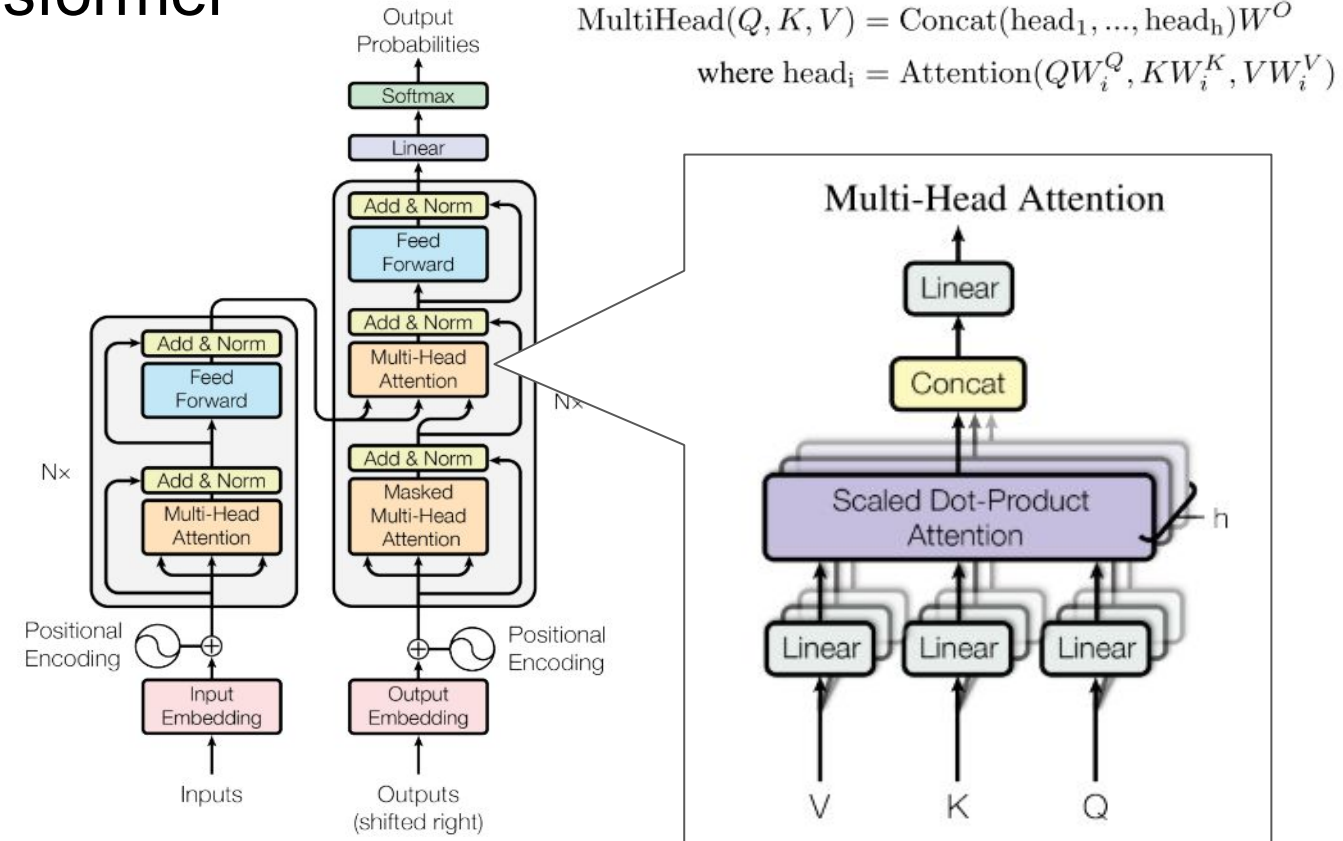


$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Source: [8] **Vaswani, A.** et al. "*Attention is all you need*". In NeurIPS (2017)

The transformer

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

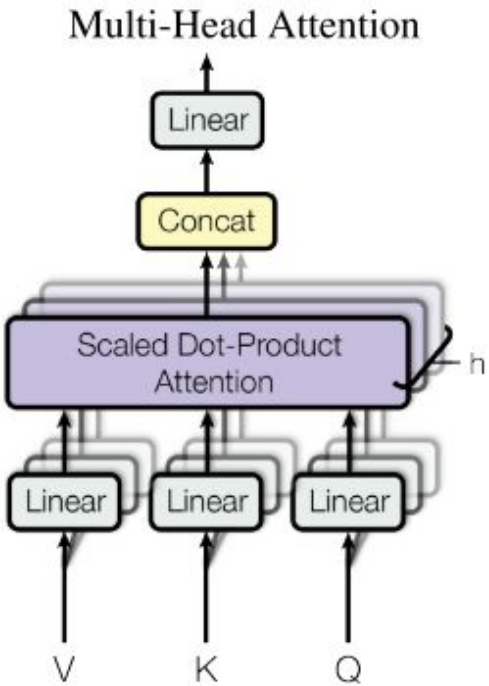


Source: [8] **Vaswani, A.** et al. "*Attention is all you need*". In NeurIPS (2017)

Multi-head Self-attention

- Map into different representation spaces (similar to convolutional filters)
- Attend at different features of the same regions
- Concatenate
- Map back to common representation space

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

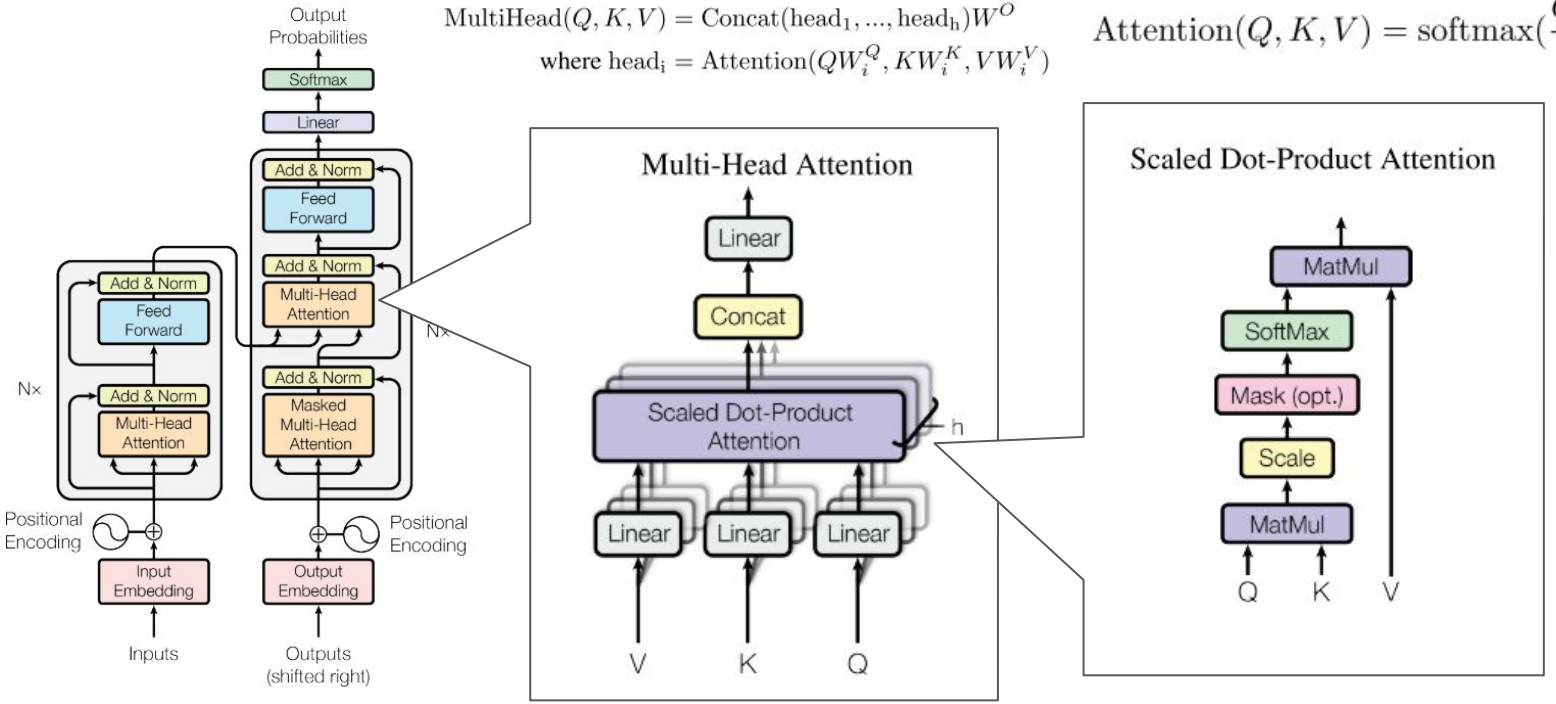


Source: [8] **Vaswani, A.** et al. "*Attention is all you need*". In NeurIPS (2017)

The transformer

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

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

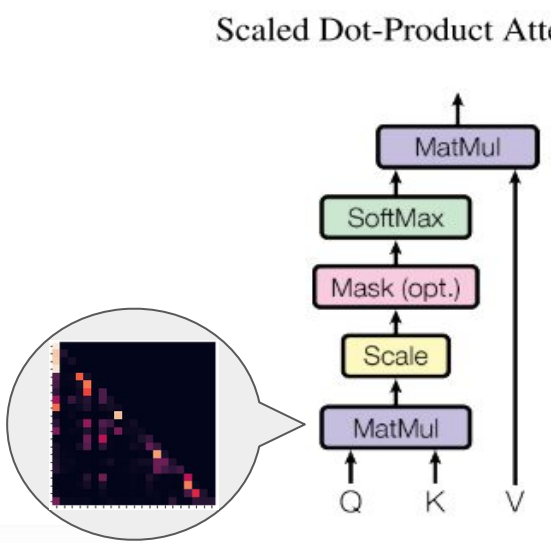


Source: [3] **Vaswani, A.** et al. "*Attention is all you need*". In NeurIPS (2017)

Scaled Dot-product

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

- Dot-product is used to compute similarity among vectors.
- The dot-product of two linearly independent vectors is 0.
- Softmax to turn it into an attention map - through probabilities (sum=1)



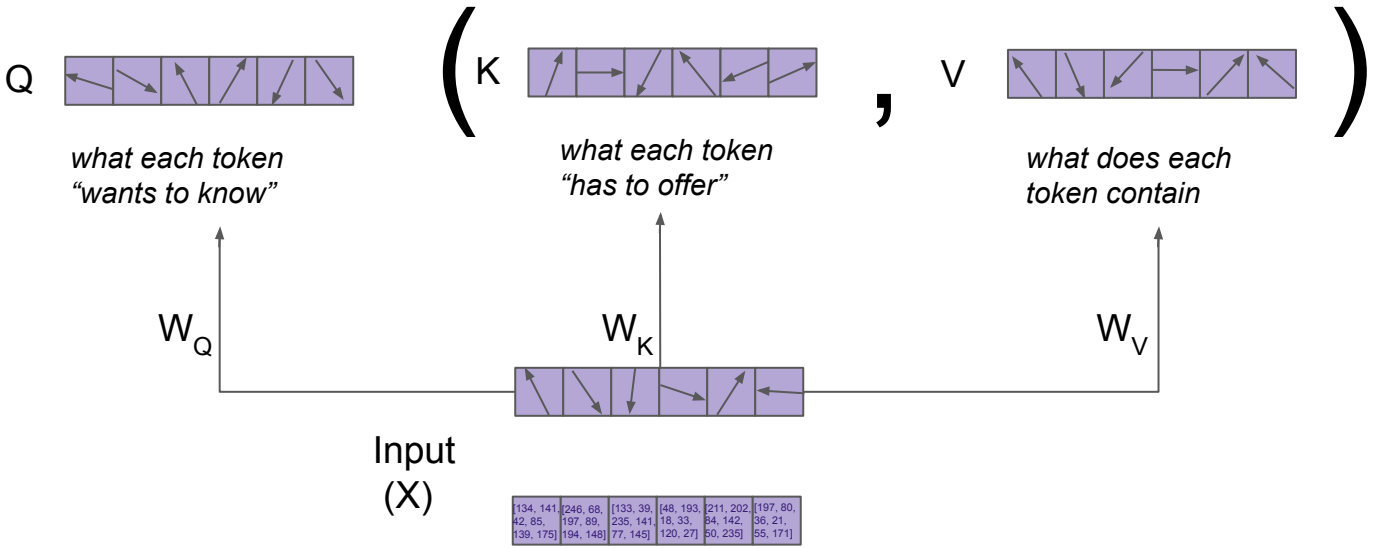
$$a_{ij} = \text{softmax}(\frac{\mathbf{q}_i {\mathbf{k}_j}^\top}{\sqrt{d_k}}) = \frac{\exp(\mathbf{q}_i {\mathbf{k}_j}^\top)}{\sqrt{d_k} \sum_{r \in S_i} \exp(\mathbf{q}_i {\mathbf{k}_r}^\top)}$$

$$y_i = \sum_{j=0}^{T} v_i a_{ij}$$

Source: [8] **Vaswani, A.** et al. "*Attention is all you need*". In NeurIPS (2017)

Queries, Keys and Values

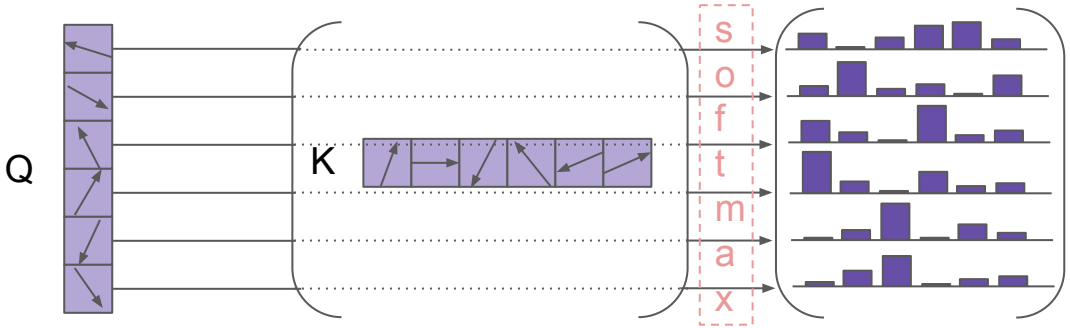
- Non-local: Mechanism used to **route information** from every token to any other
- Set of **Queries**, and set of **pairs of Keys and Values**



Queries, Keys and Values

- Each **Query** is ***compared*** **with all Keys**
- Produce "***histogram***" of **Query alignment with the Keys**

$$a_{ij} = \text{softmax}(\frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d_k}})$$

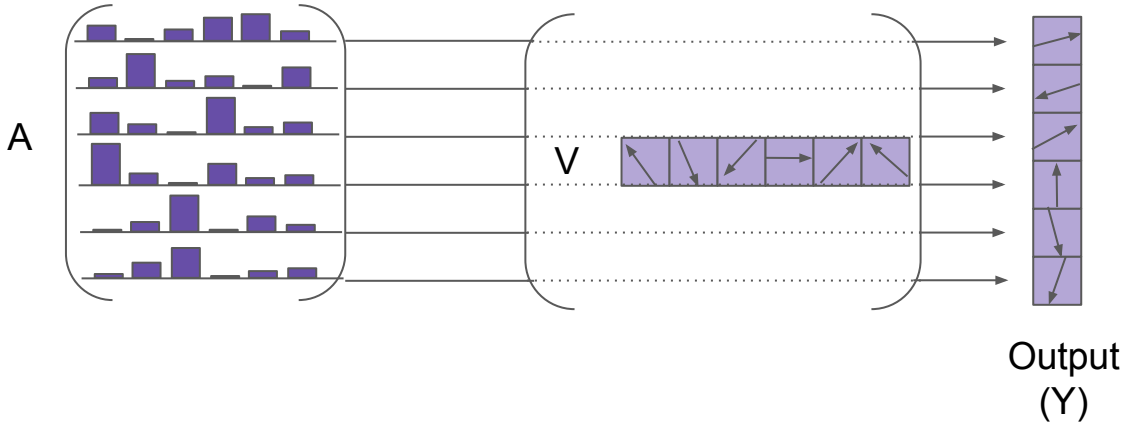


(A)ttention matrix

Queries, Keys and Values

- **Weighted summation**
- Each output token is the result of **contextual aggregation**

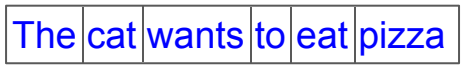
$$y_i = \sum_{j=0}^{T} v_i a_{ij}$$



Overview of the transformer

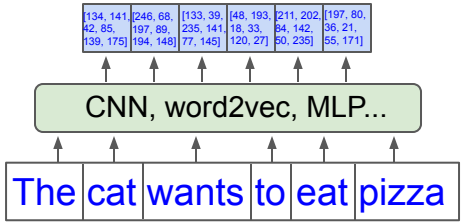
- Full **example** of original NLP Transformer for **machine translation**
- Main **difference with vision** transformers is **before the input** to the Transformer layers:
  - Tokenization
  - Embedding
  - Positional Encodings
- Other key differences are task-dependent.

Overview of the transformer



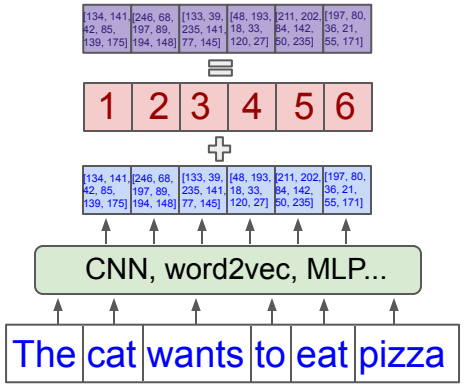
TOKENIZATION

Overview of the transformer



EMBEDDING

Overview of the transformer



POSITIONAL ENCODING

Positional Encodings

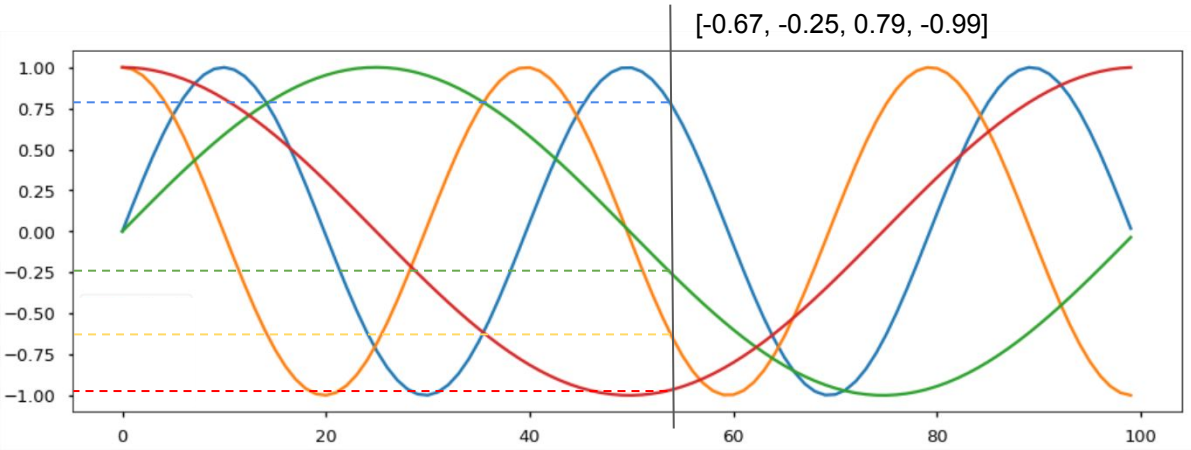
Transformers are not invariant to translation! (no idea about position)

Token: [134, 141, 42, 85]

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

Binary

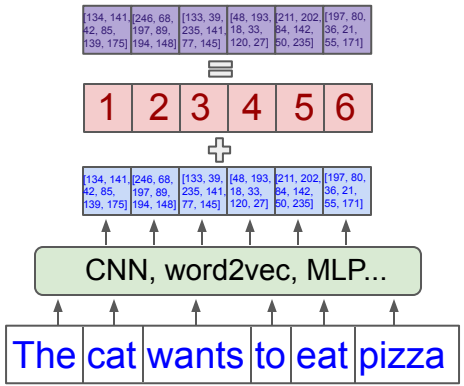
Positional encoding for token 57:
[-0.67, -0.25, 0.79, -0.99]

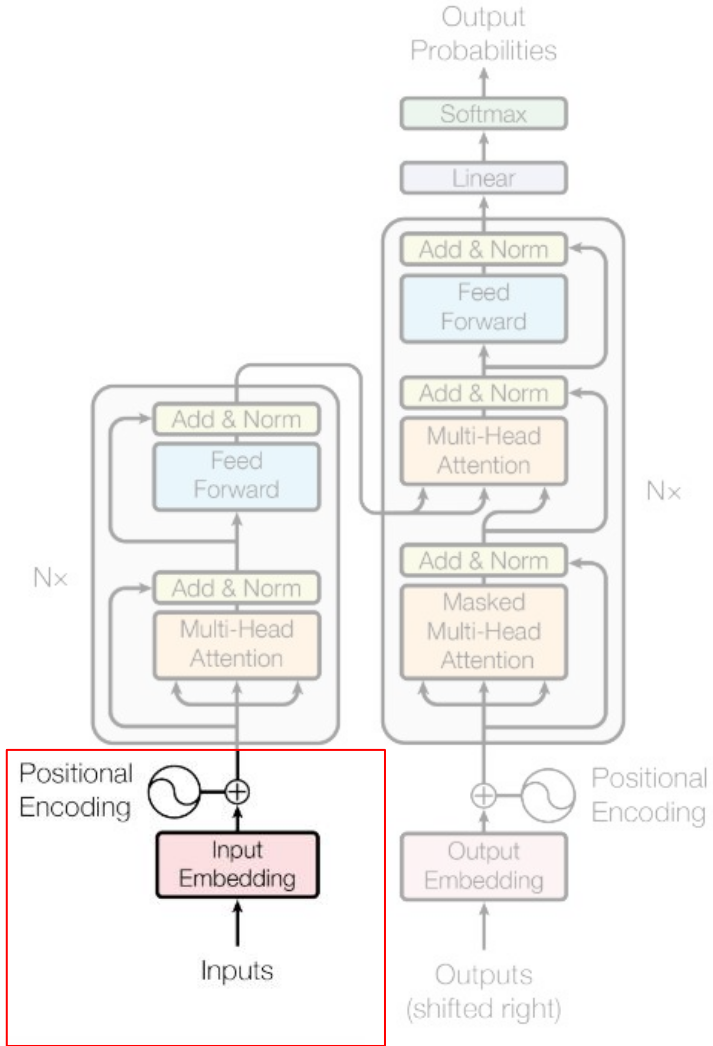


Sinusoidal

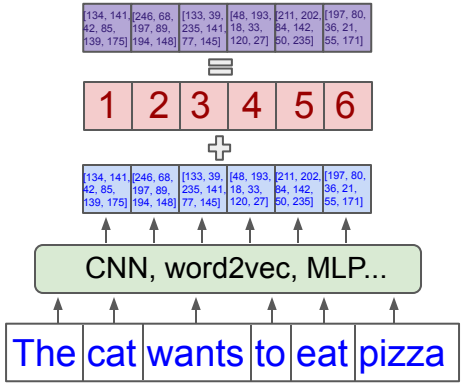
Sources: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/
http://nlp.seas.harvard.edu/2018/04/03/attention.html

Overview of the transformer

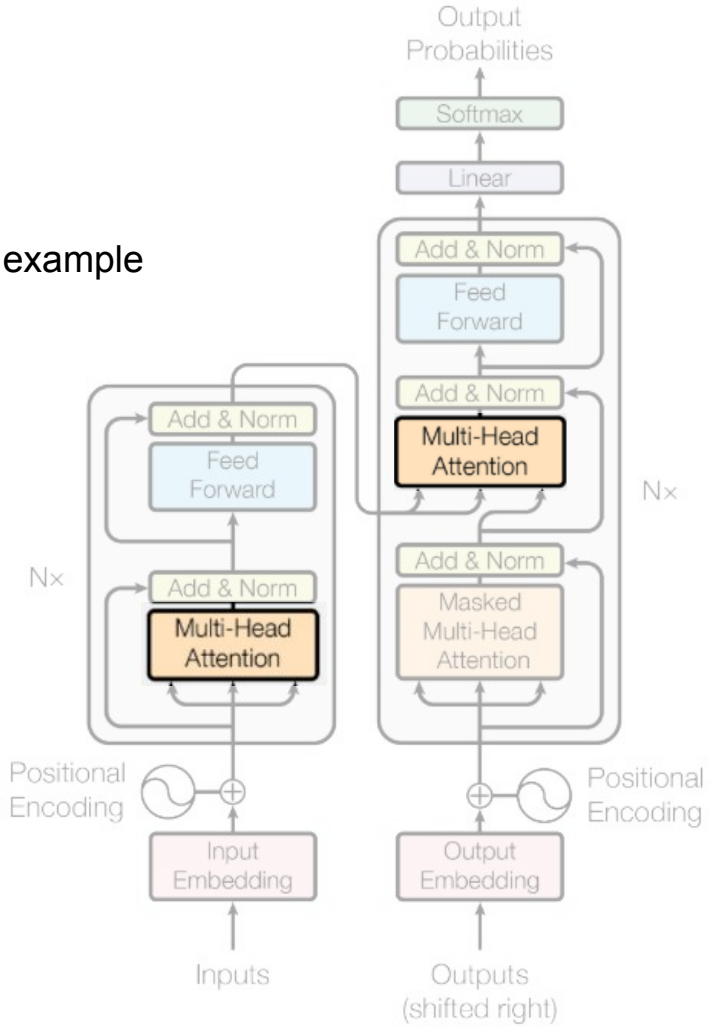




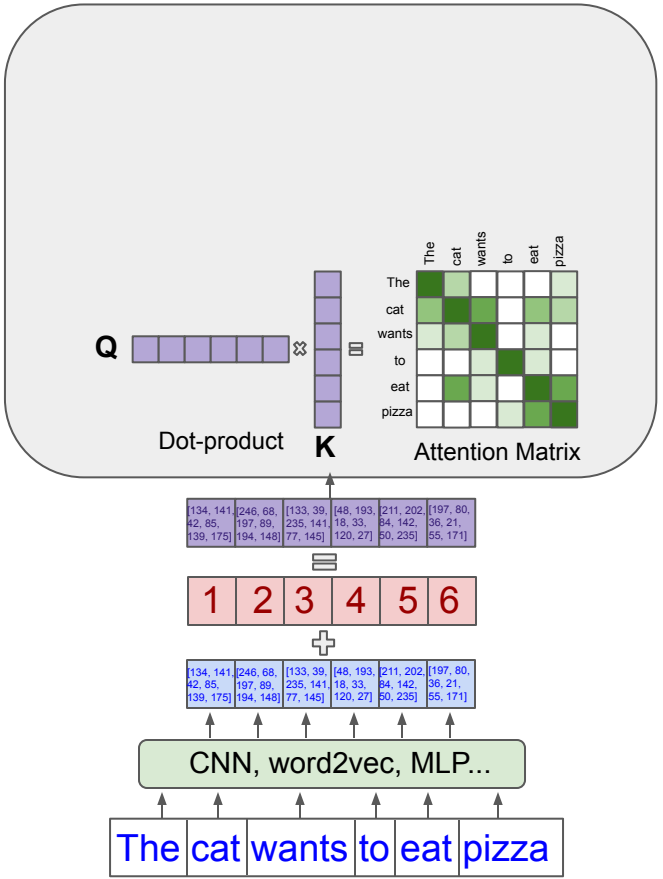
Overview of the transformer



1 head example



Overview of the transformer



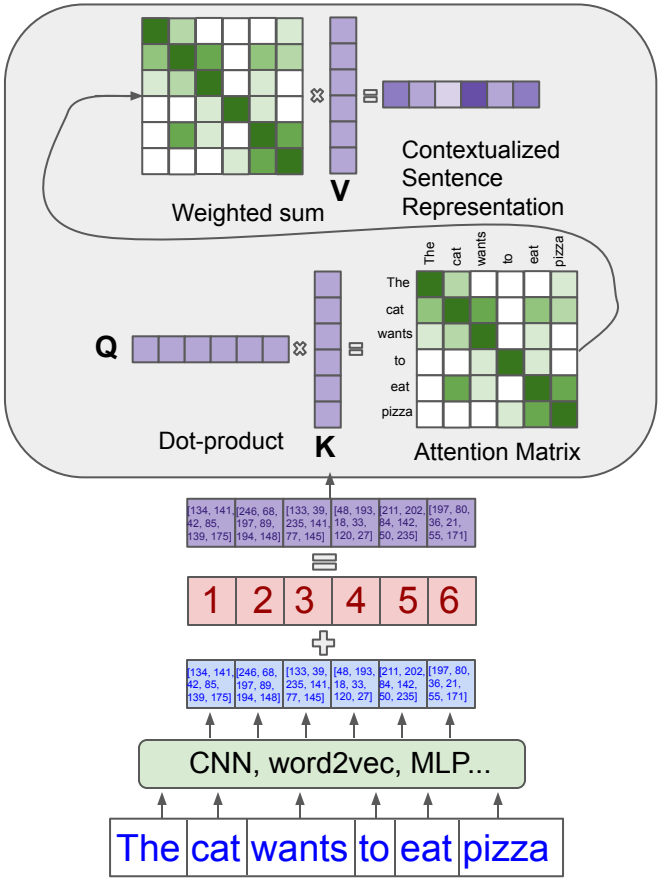
xN

ENCODING (Self-Attention)

Computing Affinity

$$a_{ij} = \text{softmax}(\frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d_k}}) = \frac{\exp(\mathbf{q}_i \mathbf{k}_j^\top)}{\sqrt{d_k} \sum_{r \in S_i} \exp(\mathbf{q}_i \mathbf{k}_r^\top)}$$

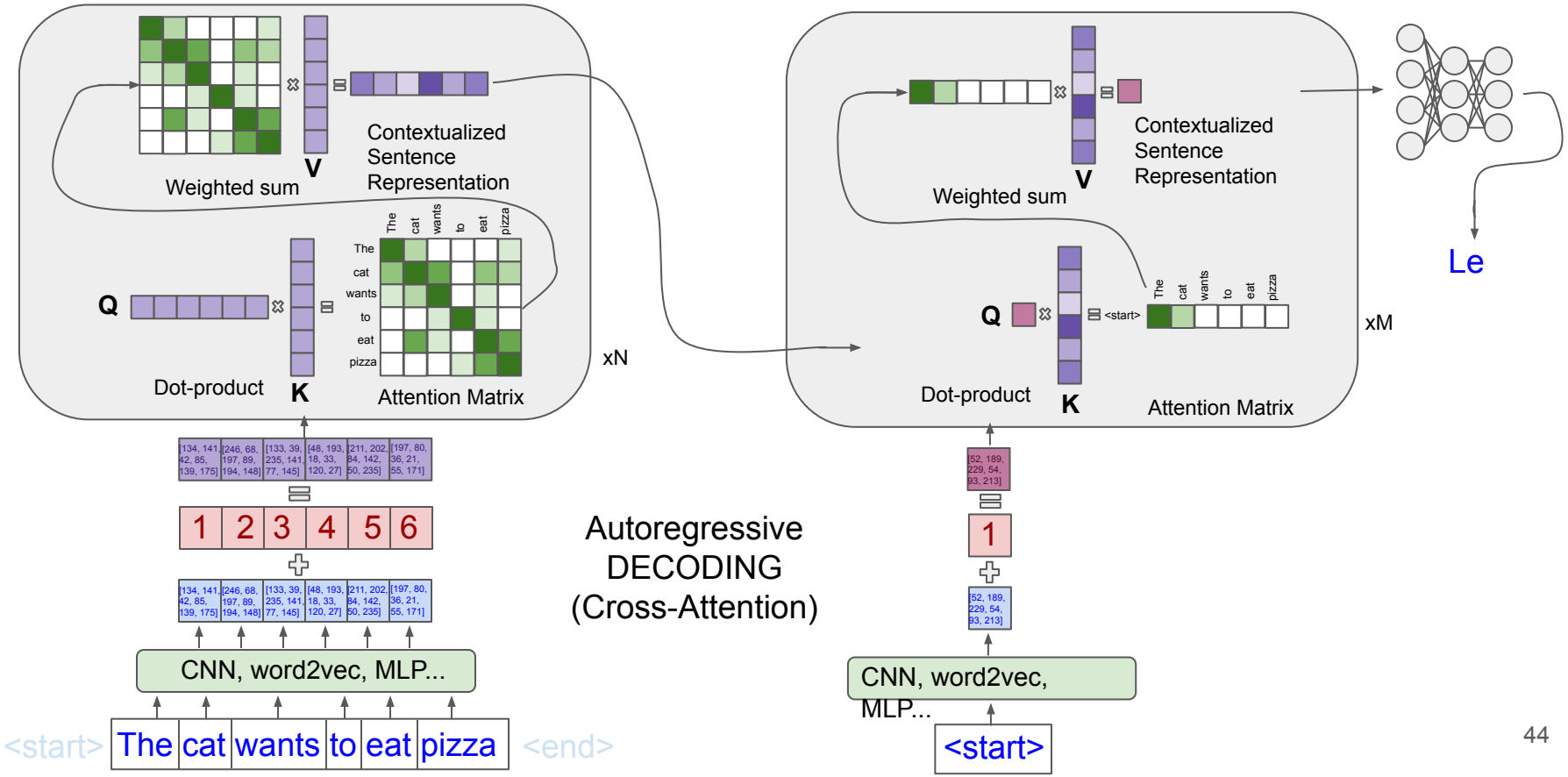
Overview of the transformer



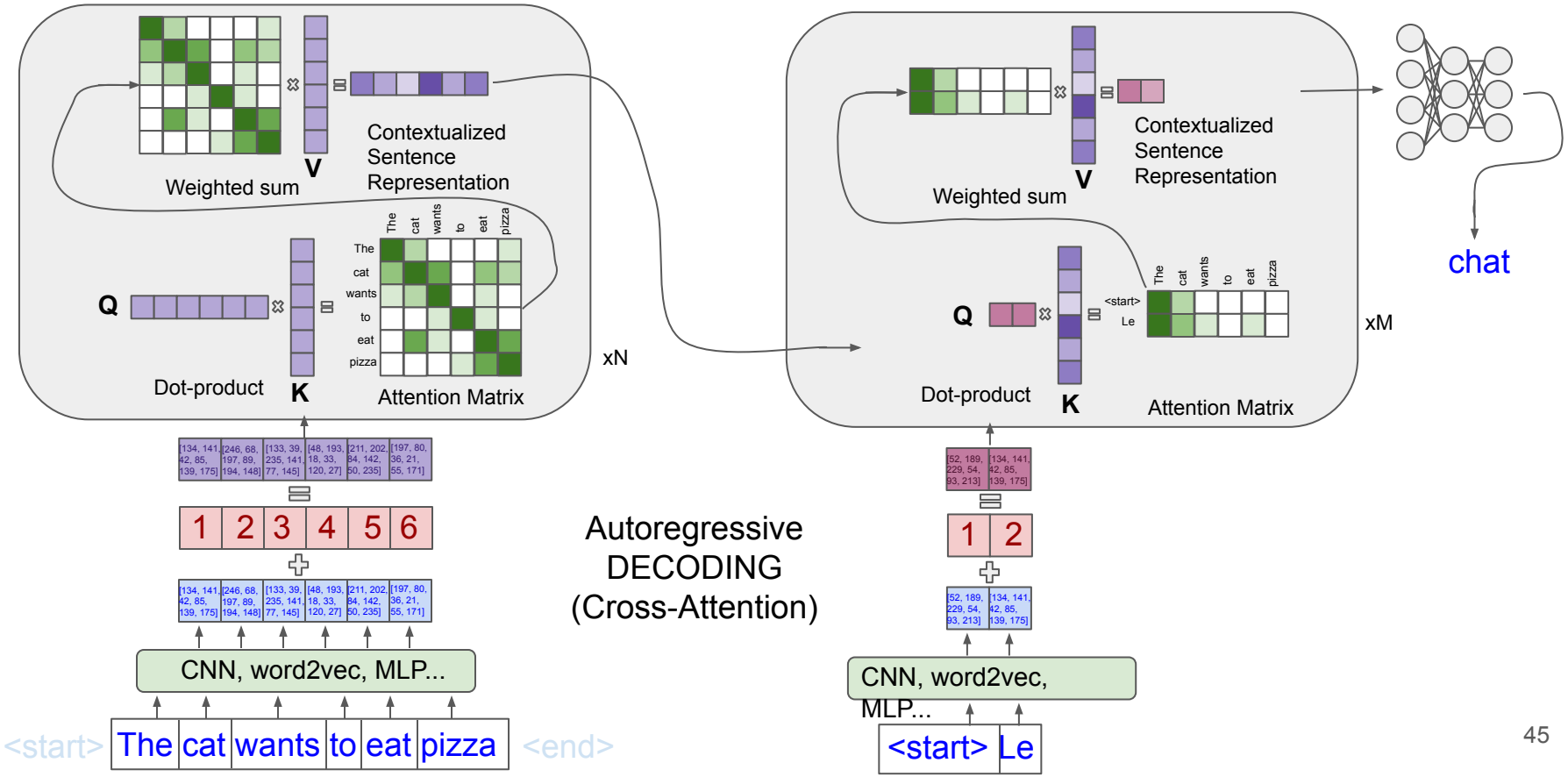
Aggregation $y_i = \sum_{j=0}^{T} v_i a_{ij}$

ENCODING (Self-Attention)

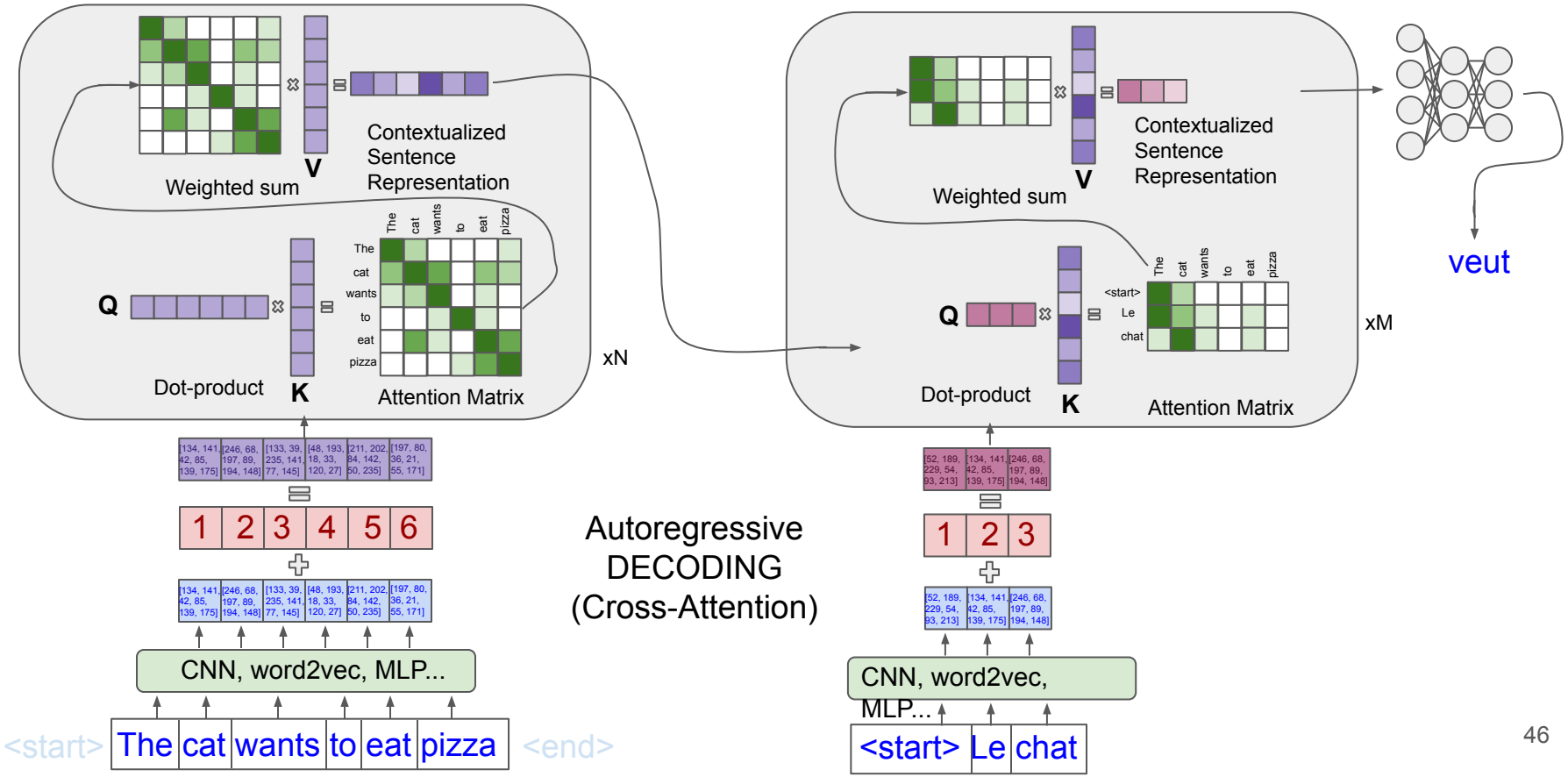
Overview of the transformer



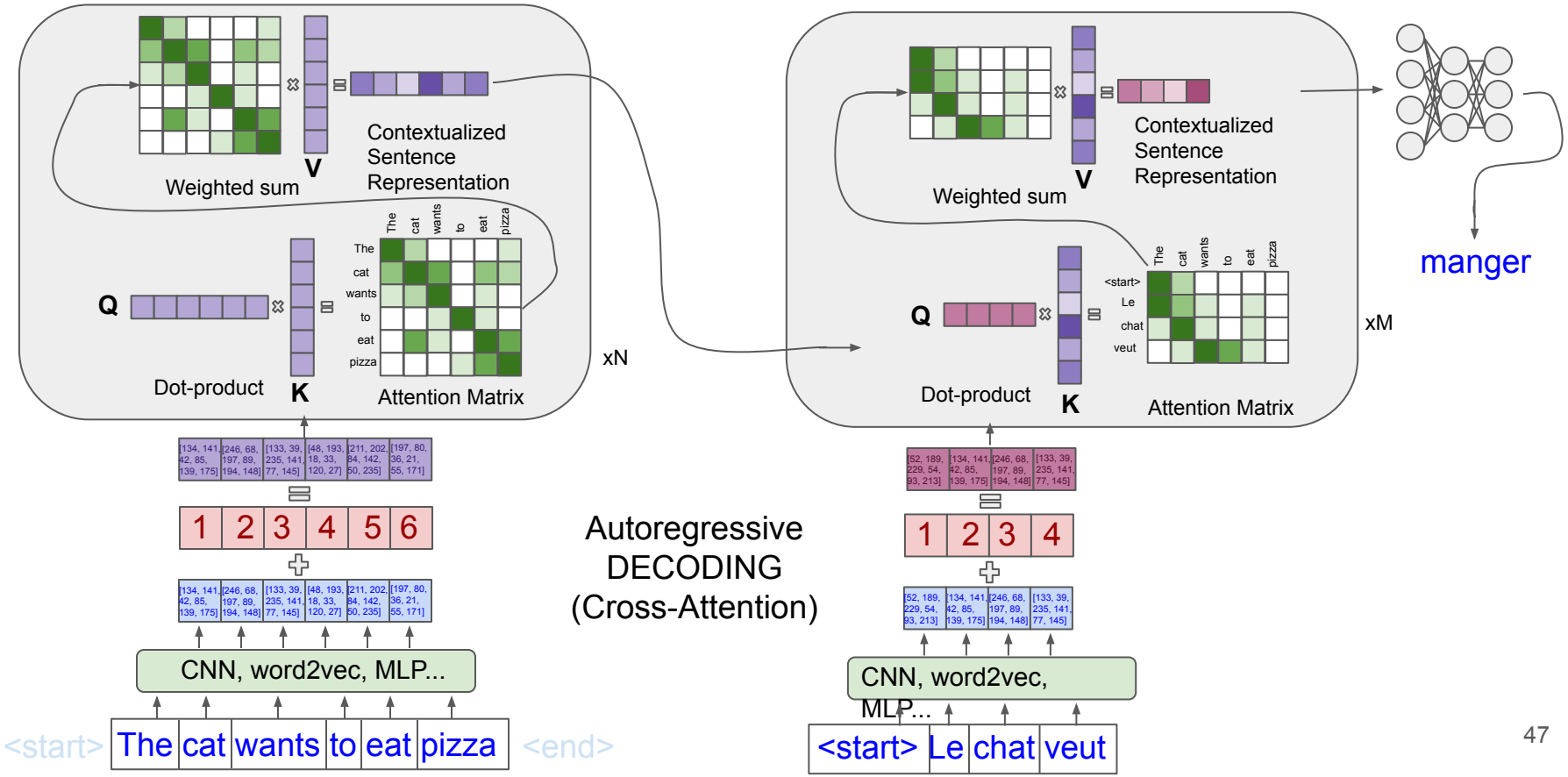
Overview of the transformer



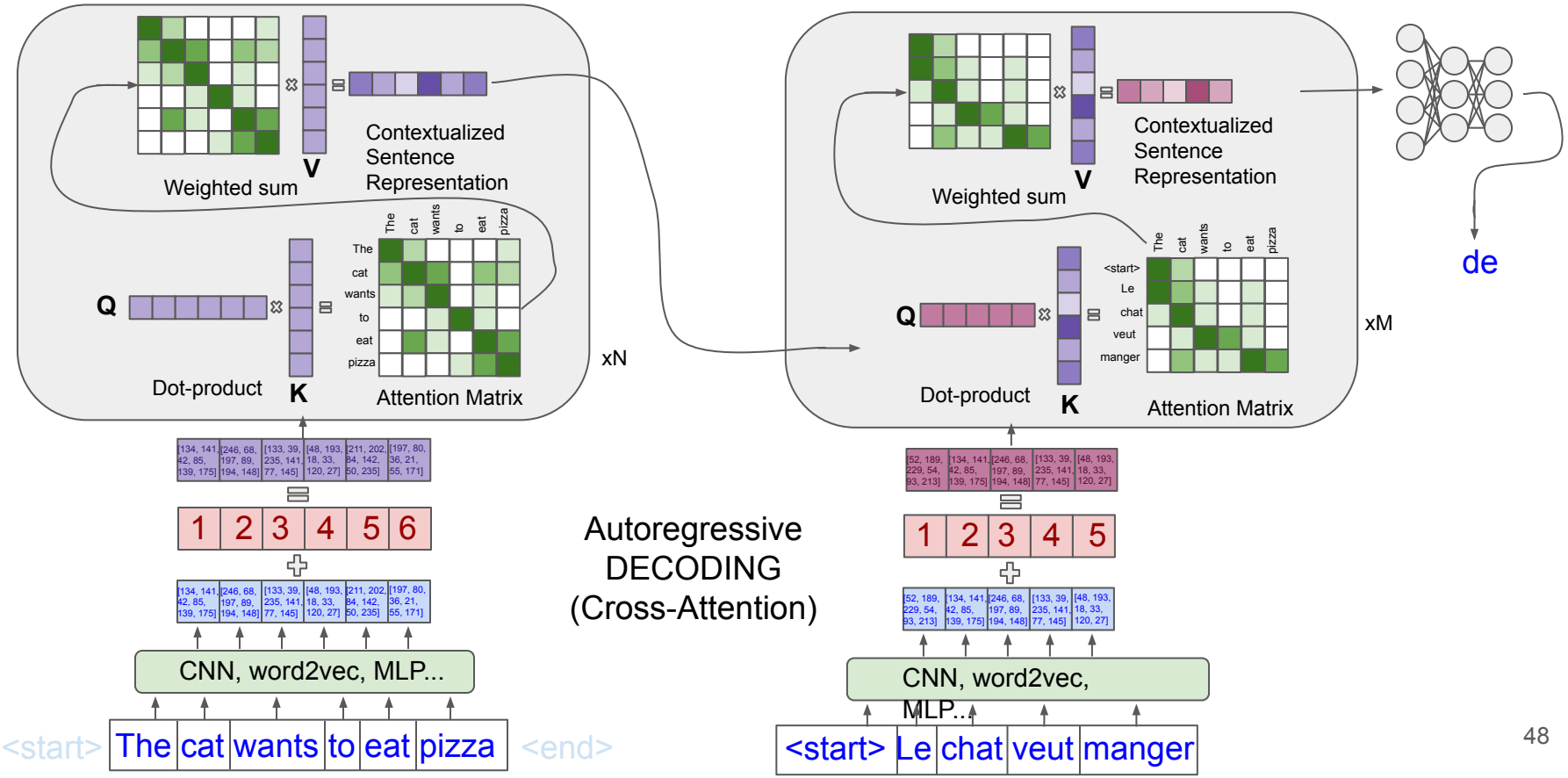
Overview of the transformer



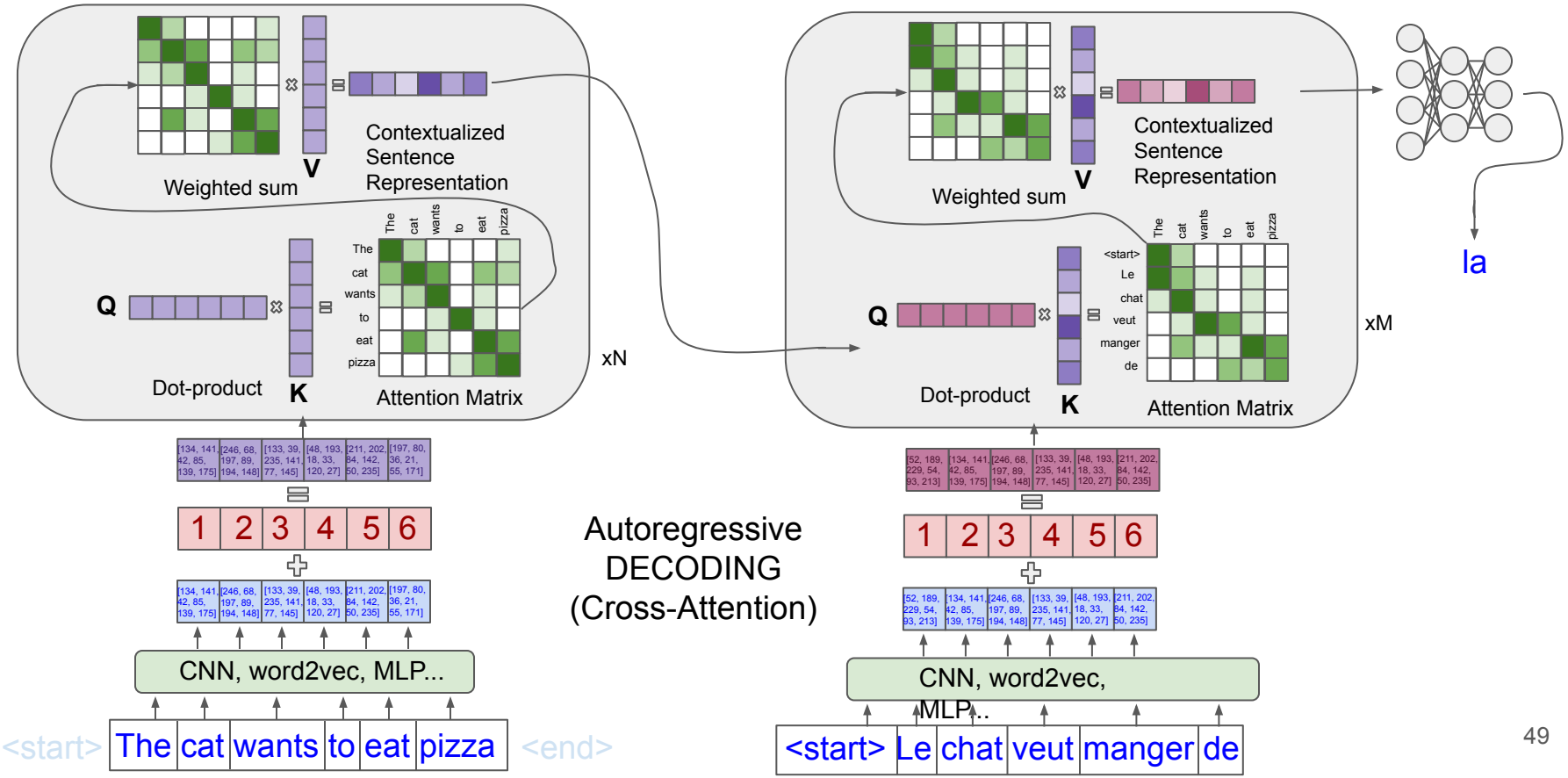
Overview of the transformer



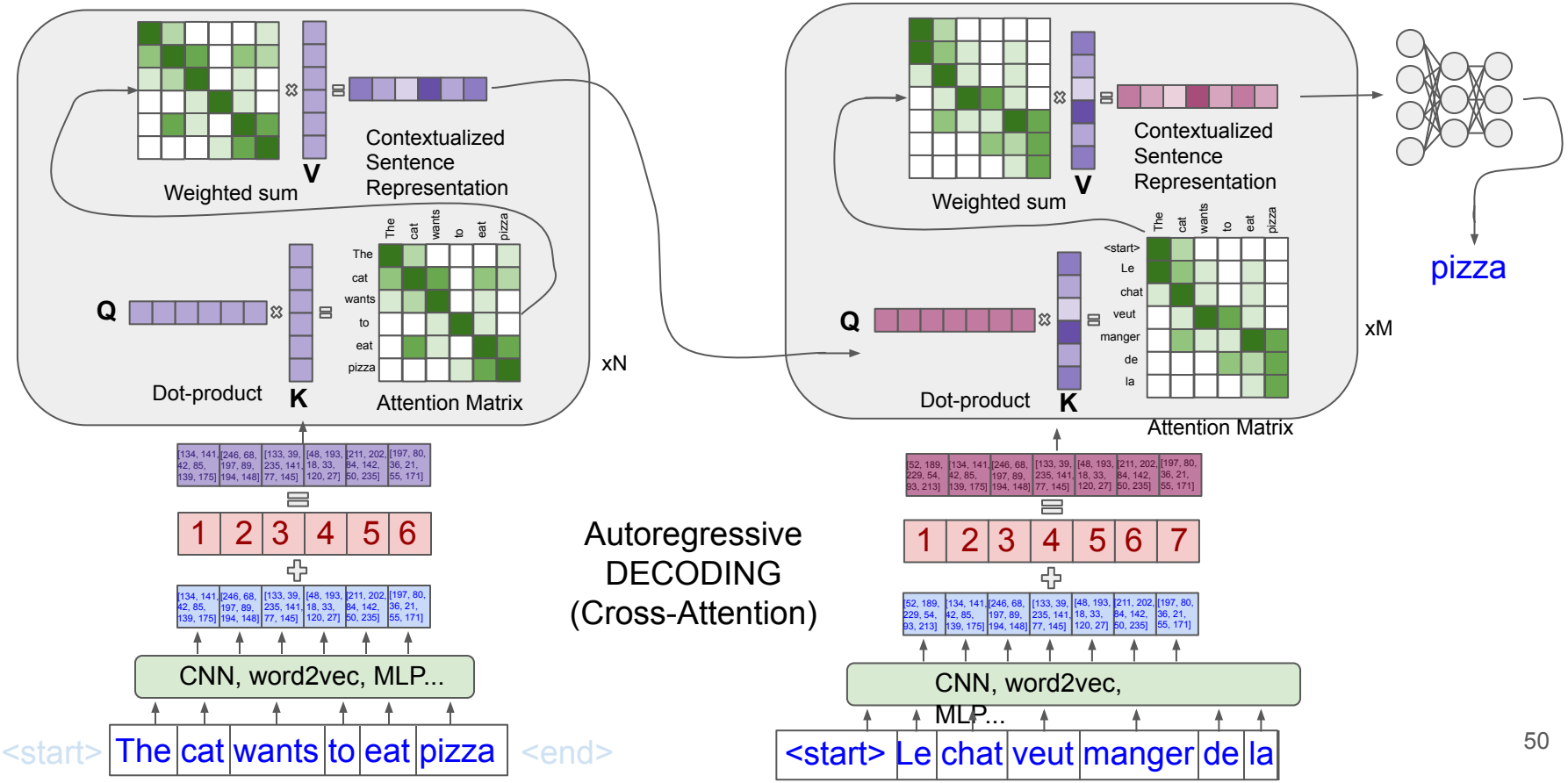
Overview of the transformer



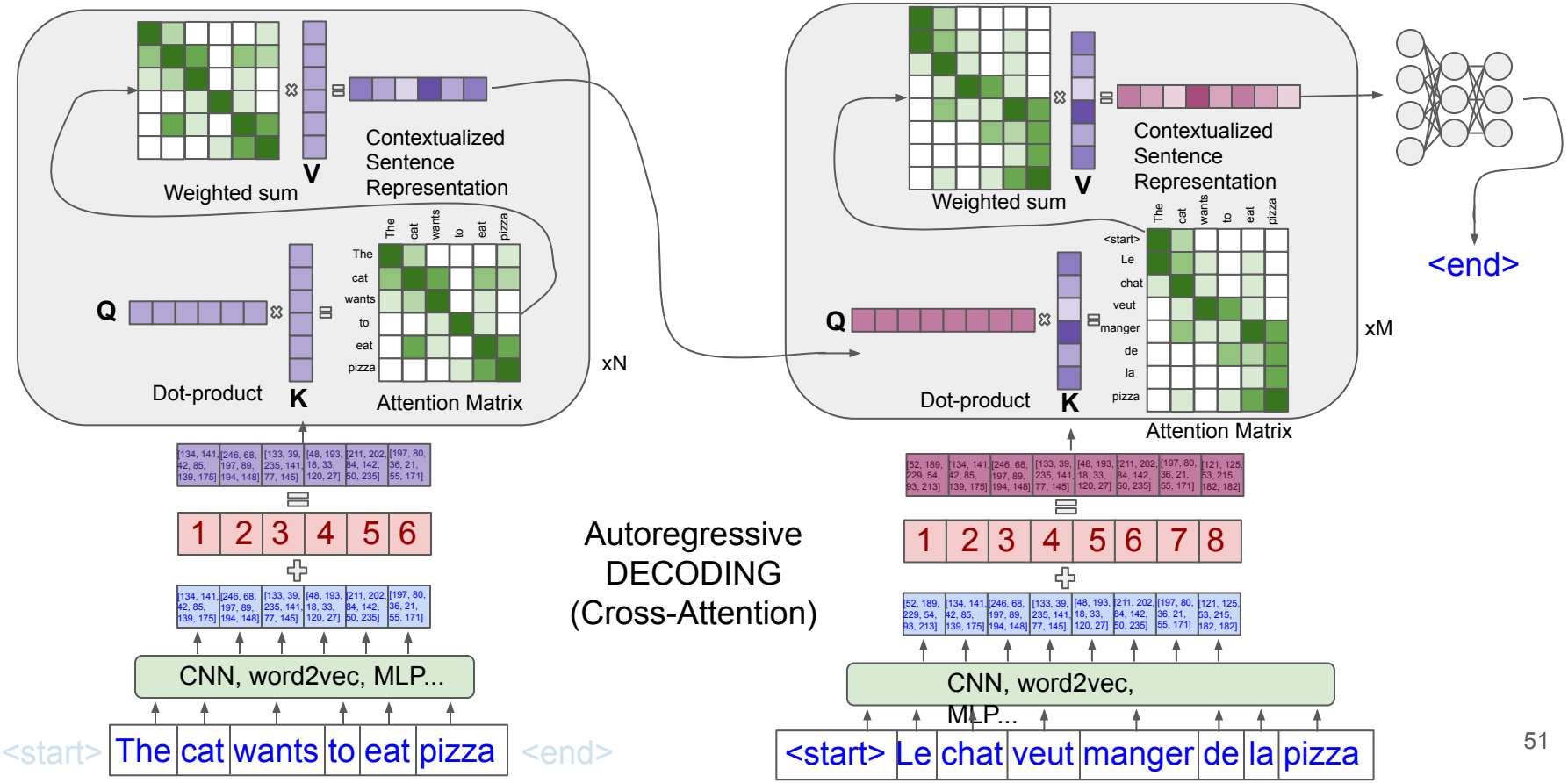
Overview of the transformer



Overview of the transformer

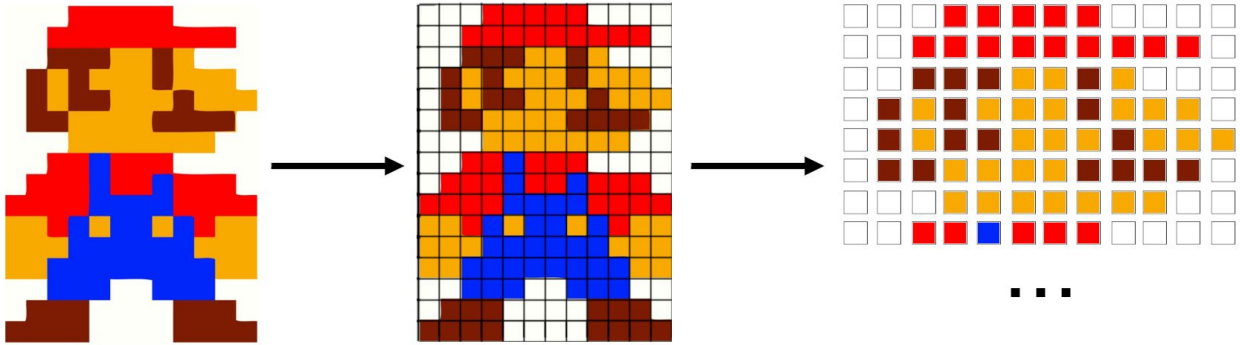


Overview of the transformer



Transformers for CV

Tokenization (images)

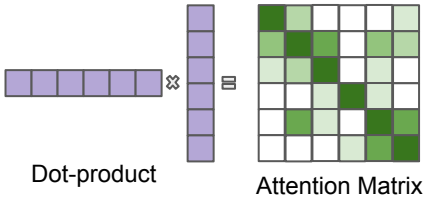


Pixels

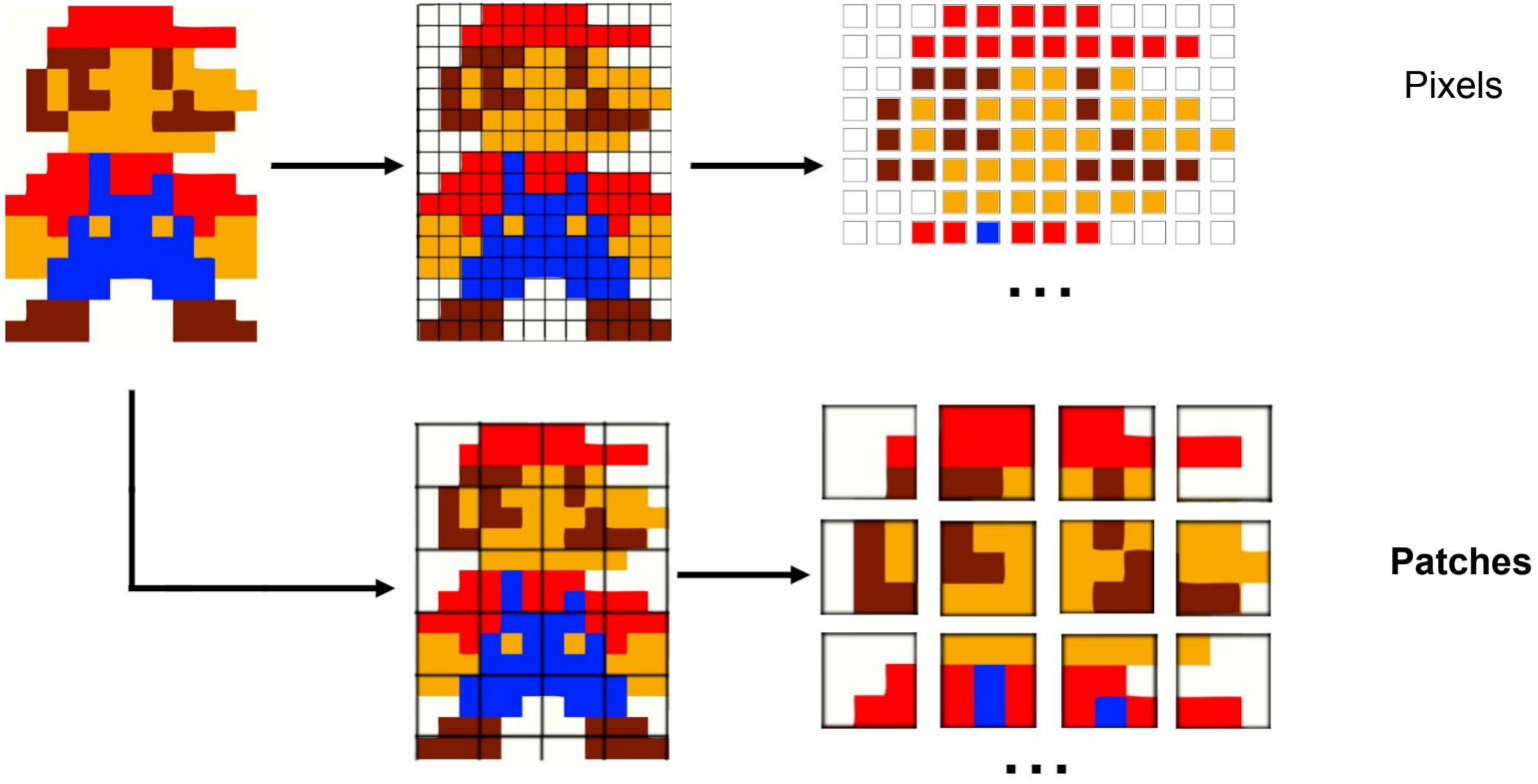
But attention matrix grows **quadratically** with **input sequence length**!! → $O(n^2)$

A 256x256 image would have = 65.536 px

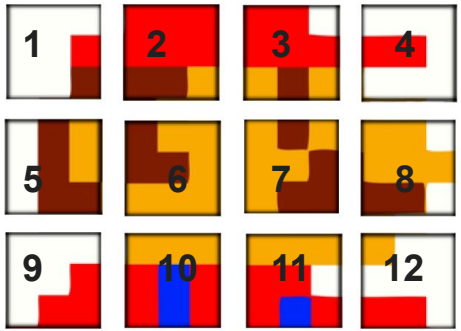
Attention matrix would have **65.536^2 elements**!!



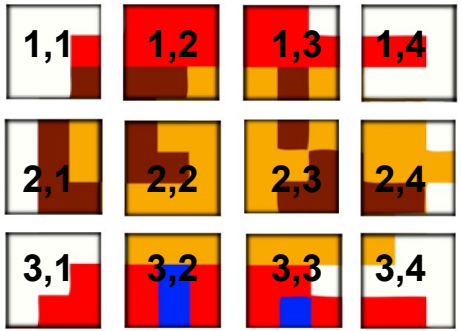
Tokenization (images)



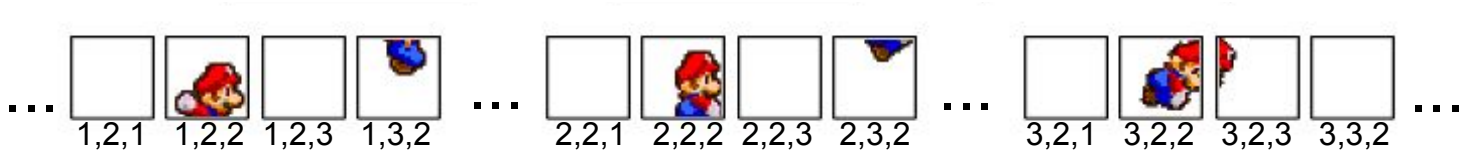
Positional Encodings



1D (Raster order)



2D (Coordinates)



3D
(Coordinates for video patches)

Inductive Biases

- Transformers lack **inductive biases**:
  - That is why we need positional encodings.
  - Opposed to convolutions, it has to learn **translation equivariance** or importance **local relations** from the data *[5]*.

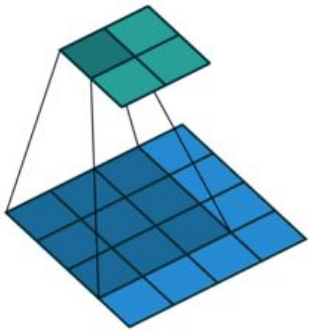


Image source: https://github.com/vdumoulin/conv_arithmetic

Source: [5] **Dosovitskiy, A.** et al. "*An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*". In ICLR (2020).

Inductive Biases

- Transformers lack **inductive biases**.
- Need a lot of resources!
  - **Big datasets** to allow the Transformer to learn
  - Long **training times** / **Big** computational clusters (**GPUs**)

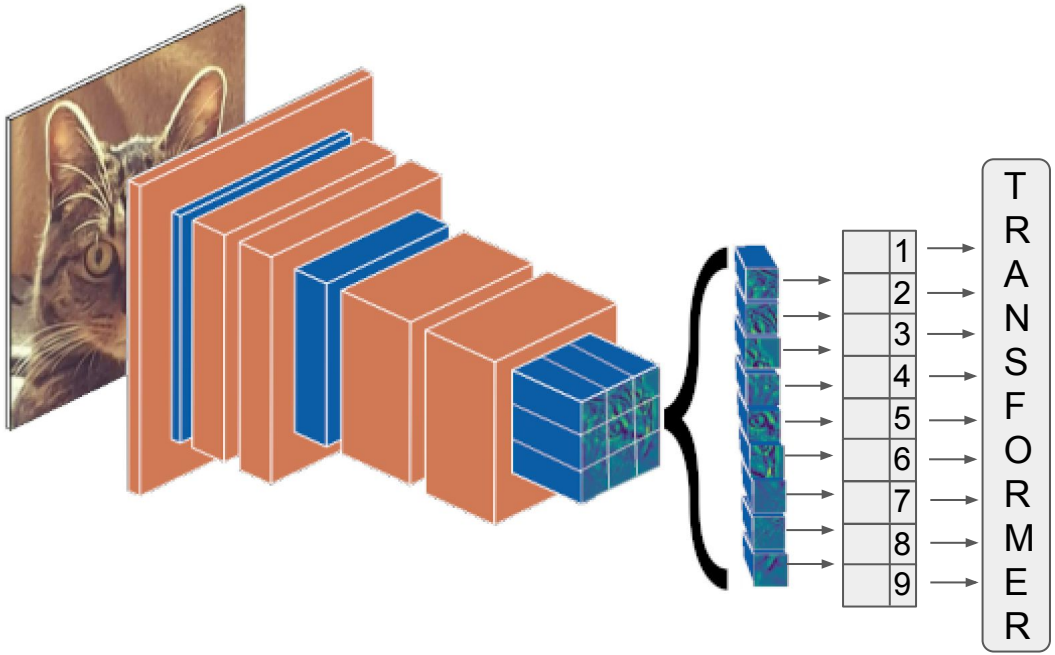
Source: [5] **Dosovitskiy, A.** et al. "*An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*". In ICLR (2020).

Inductive Biases

- Transformers lack **inductive biases**.
- Need a lot of resources!
- Solutions:
  - Use of **CNN Stems** / **Large Datasets**
  - **Pre-training**: Use unsupervised / **self-supervised** techniques to leverage more data

Source: [5] **Dosovitskiy, A.** et al. "*An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*". In ICLR (2020).

Convolutional Stem



Convolutional Network

(Local Patterns)

Tokenize the features

Add positional information

(Global Patterns)

Type	ImageNet			
	GFLOPs	#Param	Top-1	Top-5
LLGG	3.6	21.5	**82.9**	**96.2**
LLLL	3.7	23.3	81.9	95.9
LLLG	3.7	22.2	82.5	96.1
LGGG	3.6	21.6	82.7	96.1
GGGG	3.7	20.1	82.1	95.9

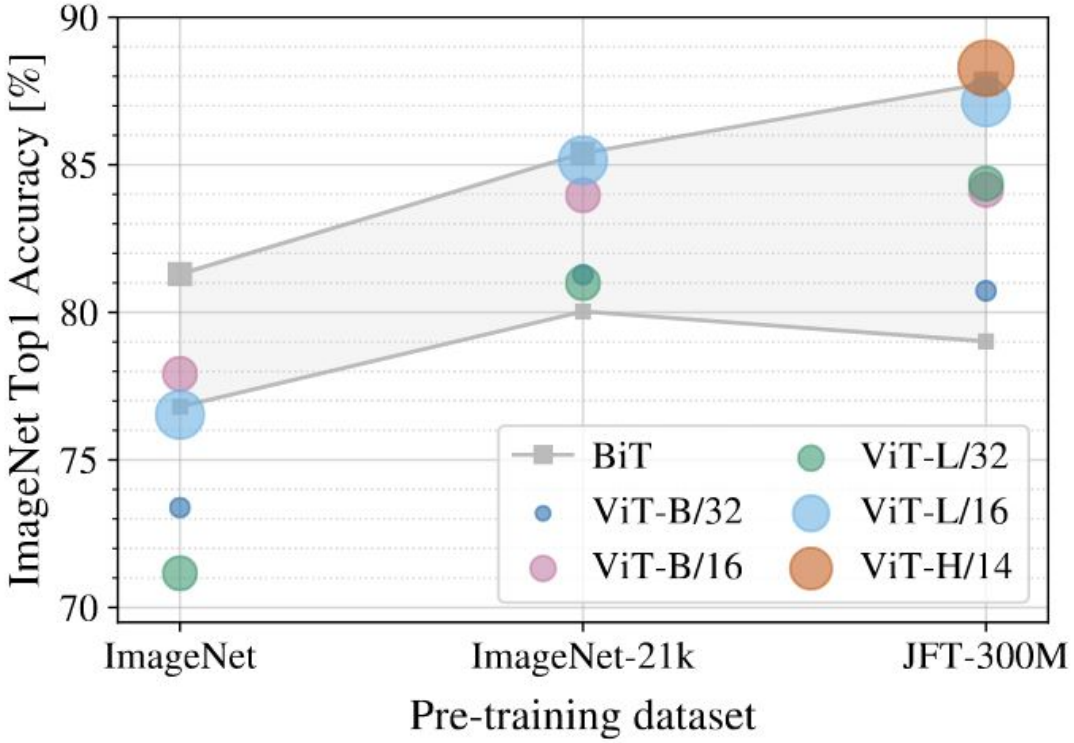
	ResNet-26		
	FLOPS (B)	**Params** (M)	**Acc.** (%)
Baseline	4.7	13.7	74.5
Conv-stem + Attention	4.5	10.3	**75.8**
Full Attention	4.7	10.3	74.8

	ResNet-38		
	FLOPS (B)	**Params** (M)	**Acc.** (%)
Baseline	6.5	19.6	76.2
Conv-stem + Attention	5.7	14.1	**77.1**
Full Attention	6.0	14.1	76.9

Image based on: **Hoeser, T. and Kuenzer, C.** "*Object Detection and Image Segmentation with Deep Learning on Earth Observation Data: A Review-Part I: Evolution and Recent Trends*", Fig. 2. In Remote Sensing. (2020).

Source: [11] **Li, K.** et al. "*UniFormer: Unified Transformer for Efficient Spatiotemporal Representation Learning*". In ICLR (2022).
[12] **Ramachandran, P.** et al. "*Stand-Alone Self-Attention in Vision Models*". In NeurIPS (2019).

“Large-scale training trumps inductive bias”



ImageNet	**1.3** Million images	x10
ImageNet-21k	**14** Million images	x20
JFT	**303** Million images	

Source: [5] **Dosovitskiy, A.** et al. “*An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*”. In ICLR (2020).

Pre-training: Motivation

- Supervision is **costly**!

Photo source: https://javilopez.eu/precariedad-laboral/call-center/

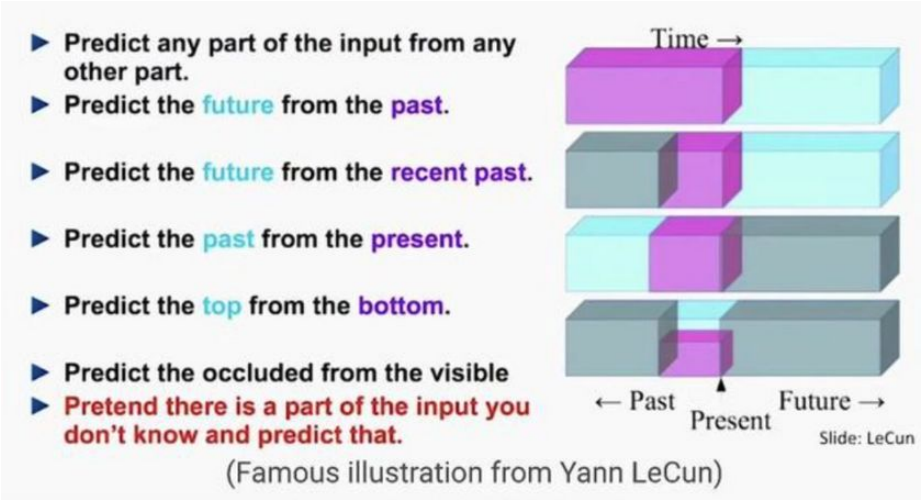
Pre-training: Motivation

- Supervision is **costly**!
- Billions of GB of **internet content**.

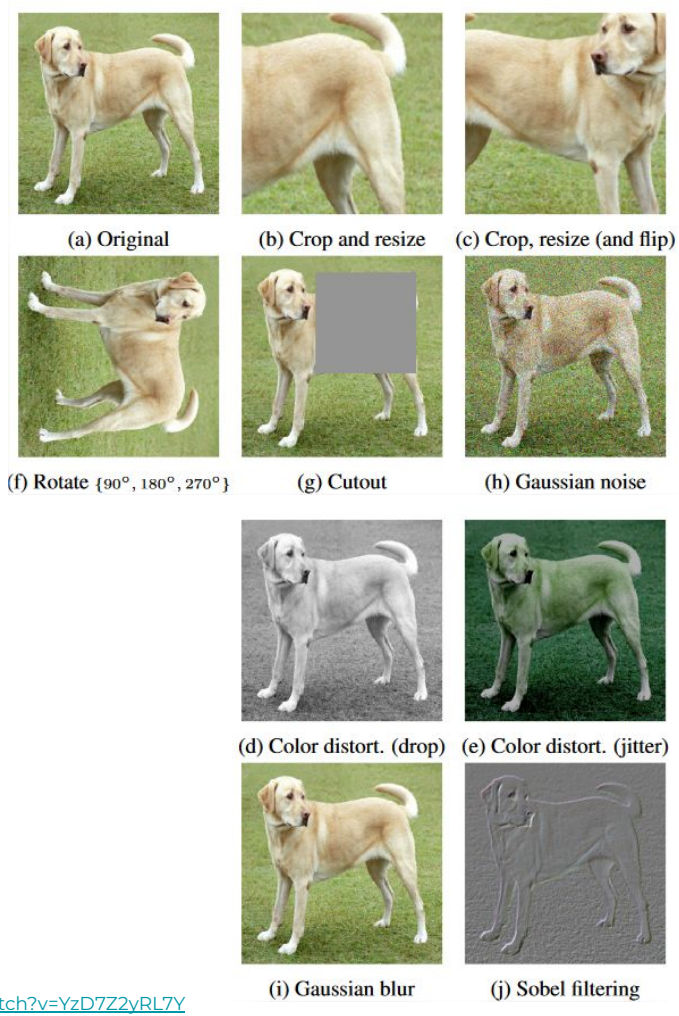
Photo source: https://i.workana.com/glossary/what-is-content-creation/

Pre-training: Motivation

- Supervision is **costly**!
- Billions of GB of **internet content**.
- Can we use the **data** itself **as supervision**?



(Famous illustration from Yann LeCun)

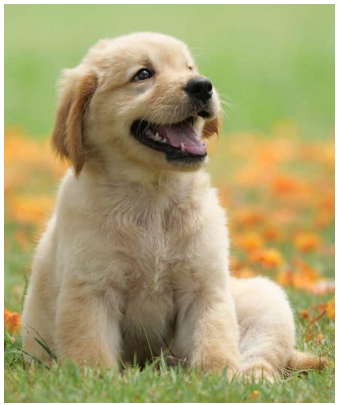


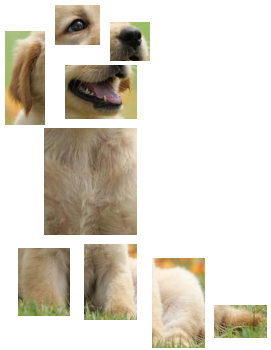
Source: **LeCun, Y.** "*Deep Learning Hardware: Past, Present, and Future*" Talk at ISSCC (2019) https://www.youtube.com/watch?v=YzD7Z2yRL7Y
Chen, T. et al. "*A Simple Framework for Contrastive Learning of Visual Representations*". In ICML (2020).

Pre-training: Motivation

- Supervision is **costly**!
- Billions of GB of **internet content**.
- Can we use the **data** itself **as supervision**?

Learn **Internal structure** of the data.





Pre-training: Motivation

- Supervision is **costly**!
- Billions of GB of **internet content**.
- Can we use the **data** itself **as supervision**?
- Fine-tune pre-trained model on **supervised downstream task**.

Learn relevant
task-specific information.

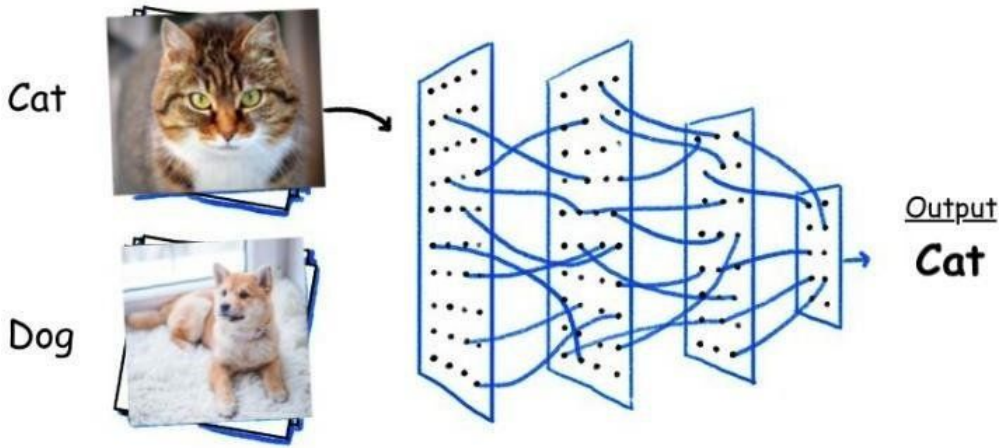
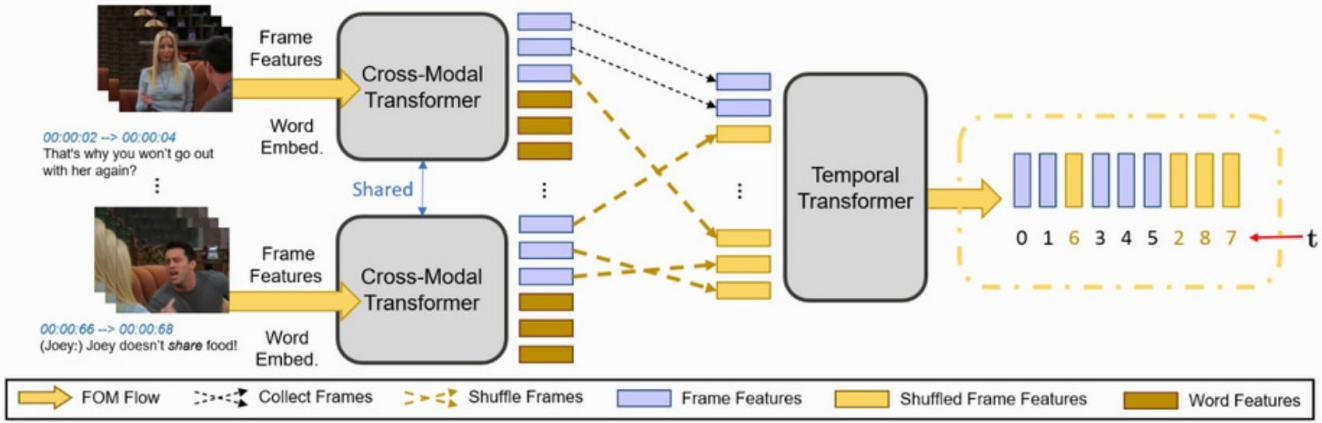


Photo source: https://www.researchgate.net/figure/Classification-of-two-classes-dog-and-cat-using-CNN_fig2_353247632

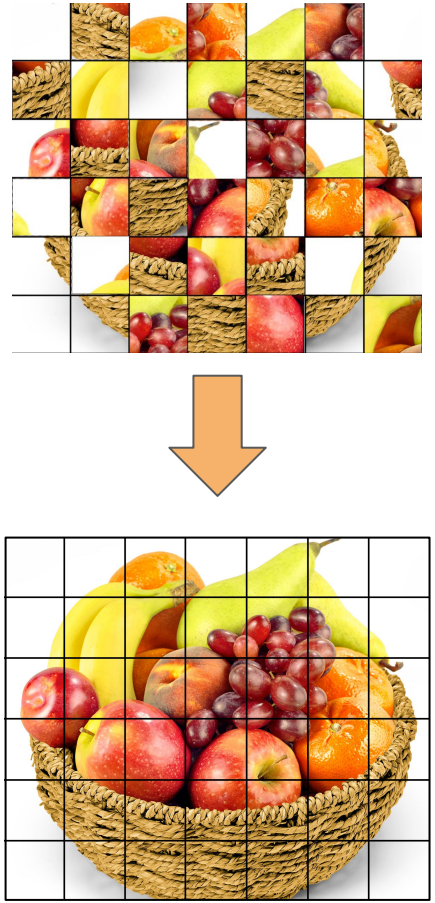
Pre-training: Token shuffling (Jigsaw Puzzle)



Reorder Indices: $\mathbf{r} = \{r_i\}_{i=1}^{R} \in \mathbb{N}^R$

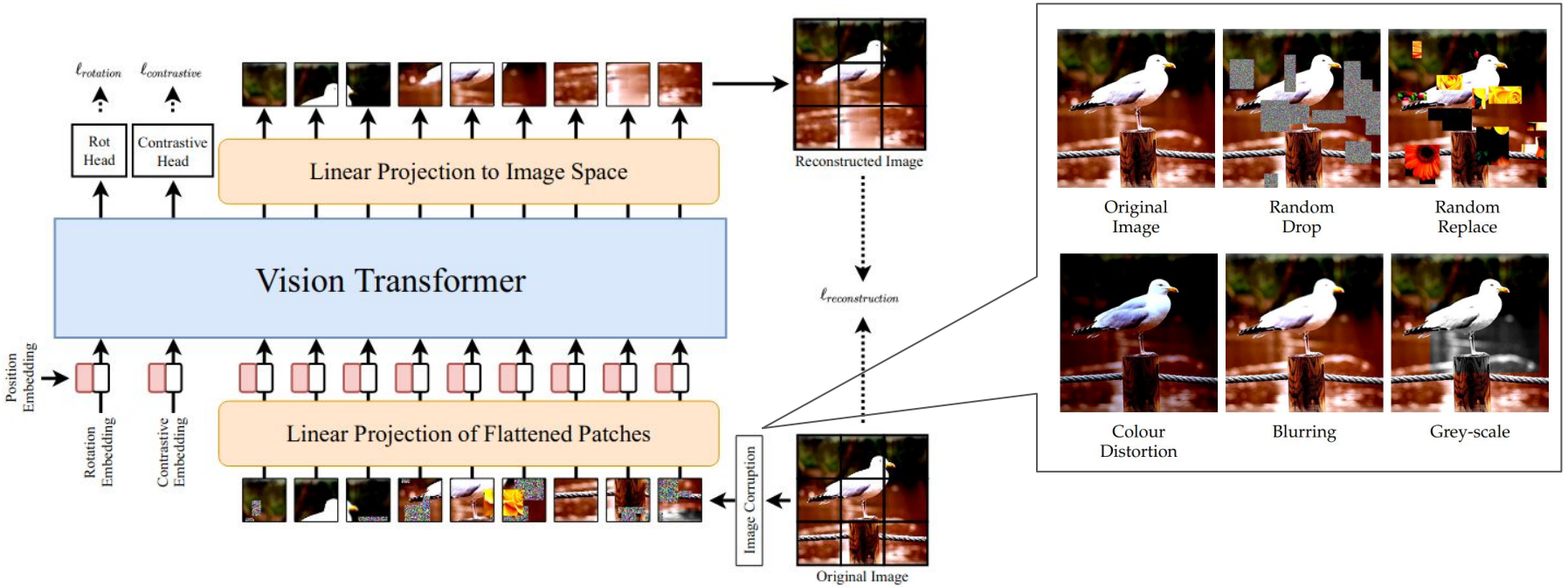
Original timestamp: $\mathbf{t} = \{t_i\}_{i=1}^{R}$

Loss Function of FOM: $\mathcal{L}_{\text{FOM}} = -\mathbb{E}_D \sum_{i=1}^{R} \log \mathbf{P}[r_i, t_i]$



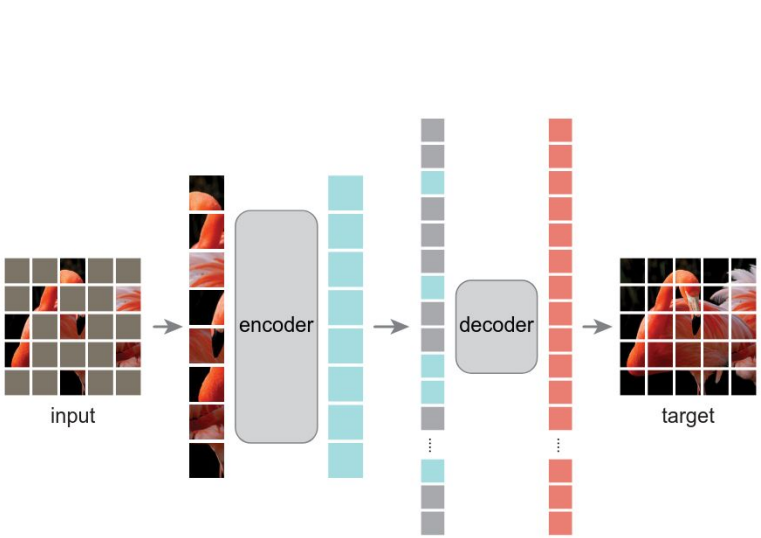
Source: [13] **Li, L.** et al. "*HERO: Hierarchical Encoder for Video+ Language Omni-representation Pre-training*". In EMNLP (2020).

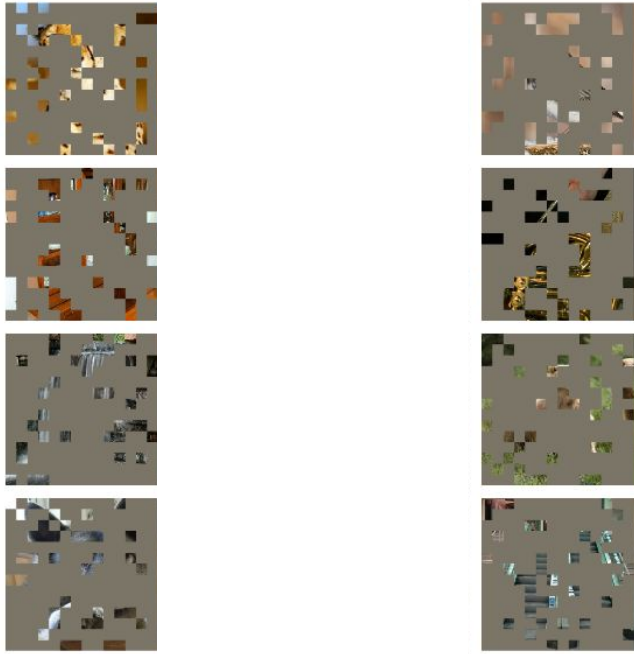
Pre-training: Reconstruction / Denoising



Source: [14] **Atito, S.** et al. "*Sit: Self-supervised vision transformer*". In CoRR (2021).

Pre-training: Masked Token Modeling

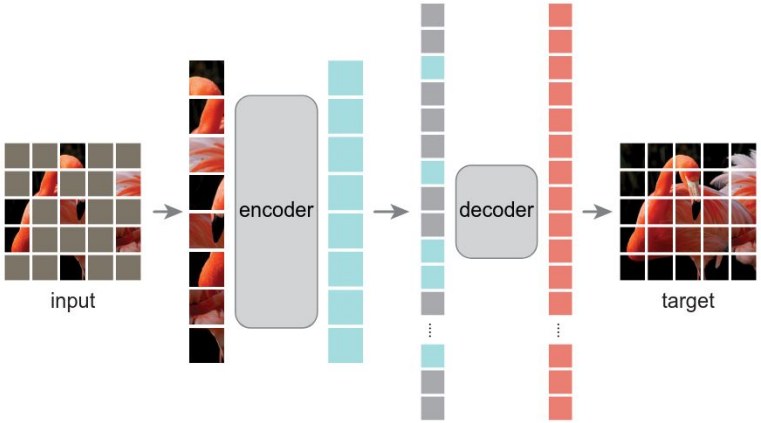


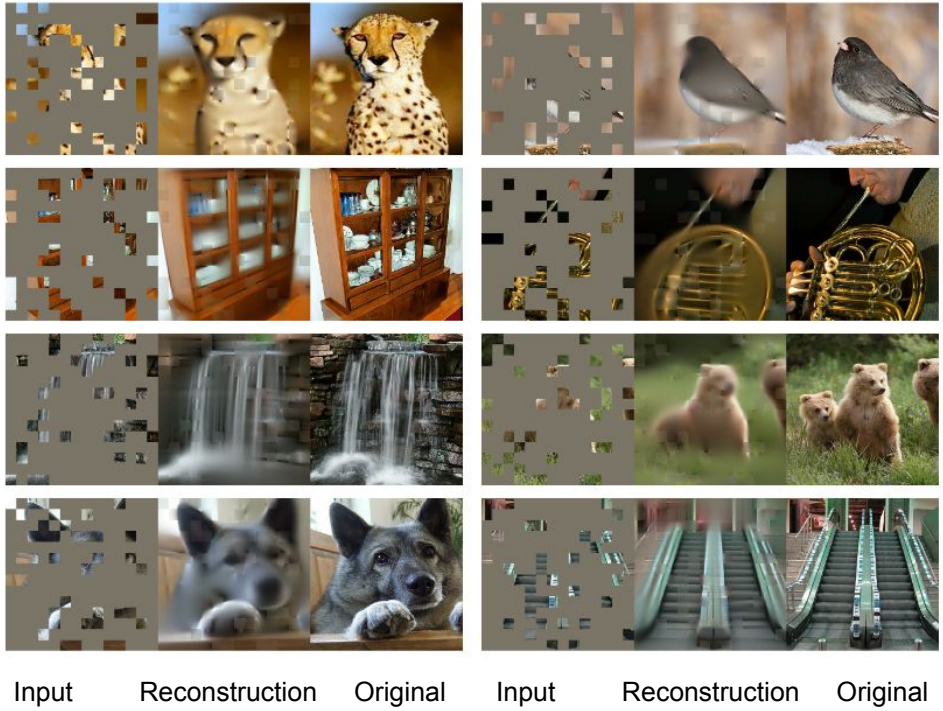


What do you see here?

Source: [15] **He, K.** et al. "*Masked autoencoders are scalable vision learners*". In CVPR (2022).

Pre-training: Masked Token Modeling





Source: [15] **He, K.** et al. "*Masked autoencoders are scalable vision learners*". In CVPR (2022).

Pre-training: Contrastive without negatives

Borrows ideas from **knowledge distillation**, data augmentation and metric learning.

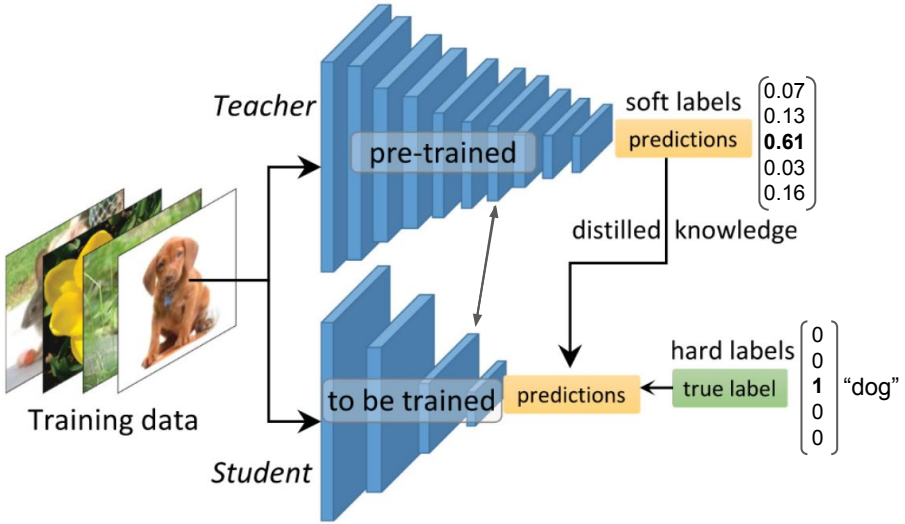


Image source: **Wang, J.** et al. "*DeepVID: Deep Visual Interpretation and Diagnosis for Image Classifiers via Knowledge Distillation*". In IEEE Tr-VCG (2019).

Pre-training: Contrastive without negatives

Borrows ideas from knowledge distillation, **data augmentation** and metric learning.

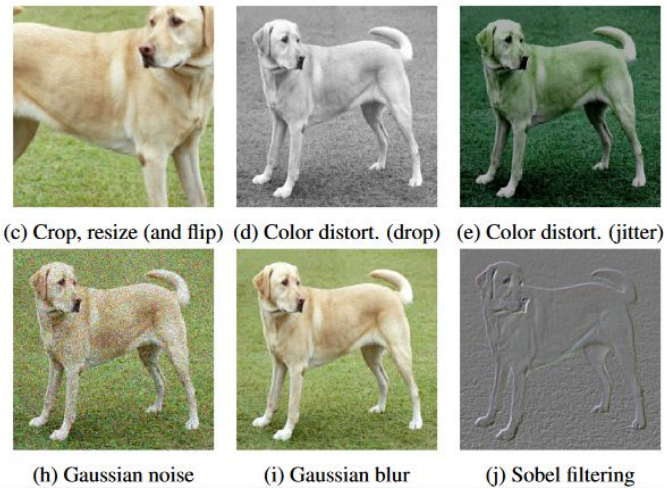
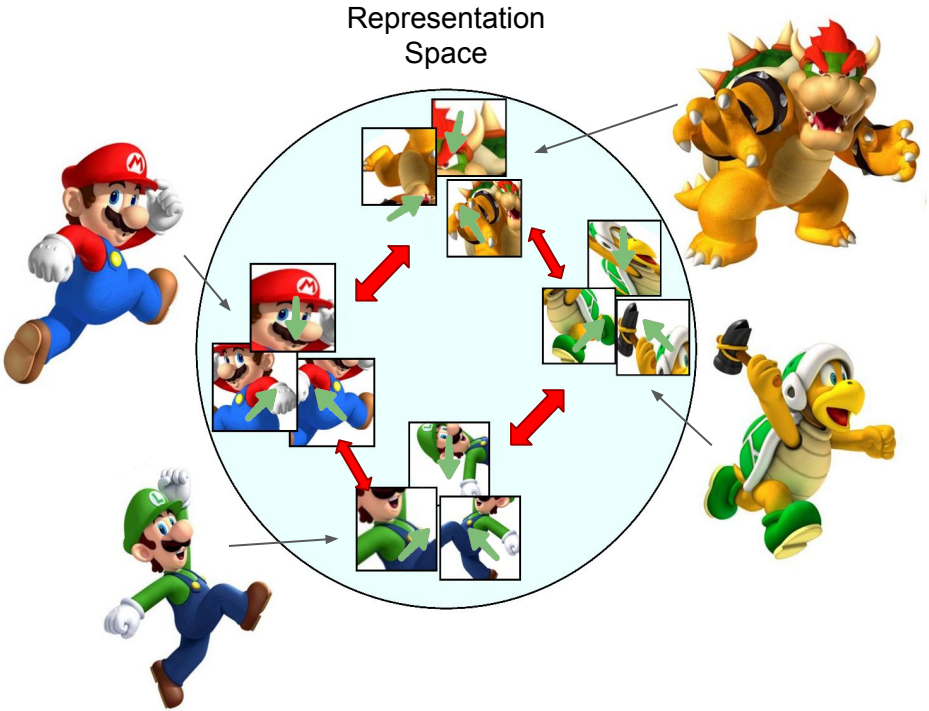


Image source: **Chen, T.** et al. "*A Simple Framework for Contrastive Learning of Visual Representations*". In ICML (2020).

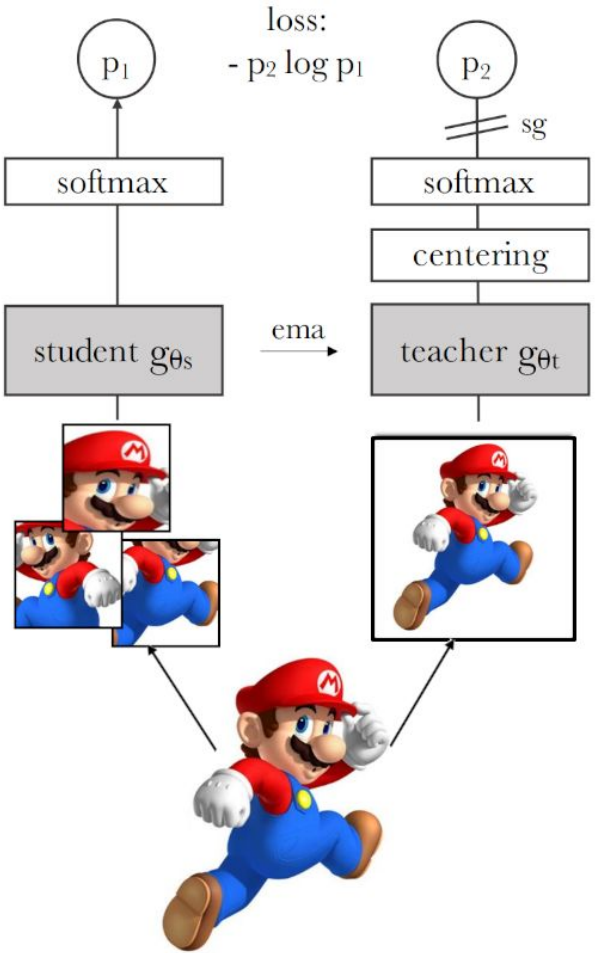
Pre-training: Contrastive without negatives

Borrows ideas from knowledge distillation, data augmentation and **metric learning**.



Pre-training: DINO

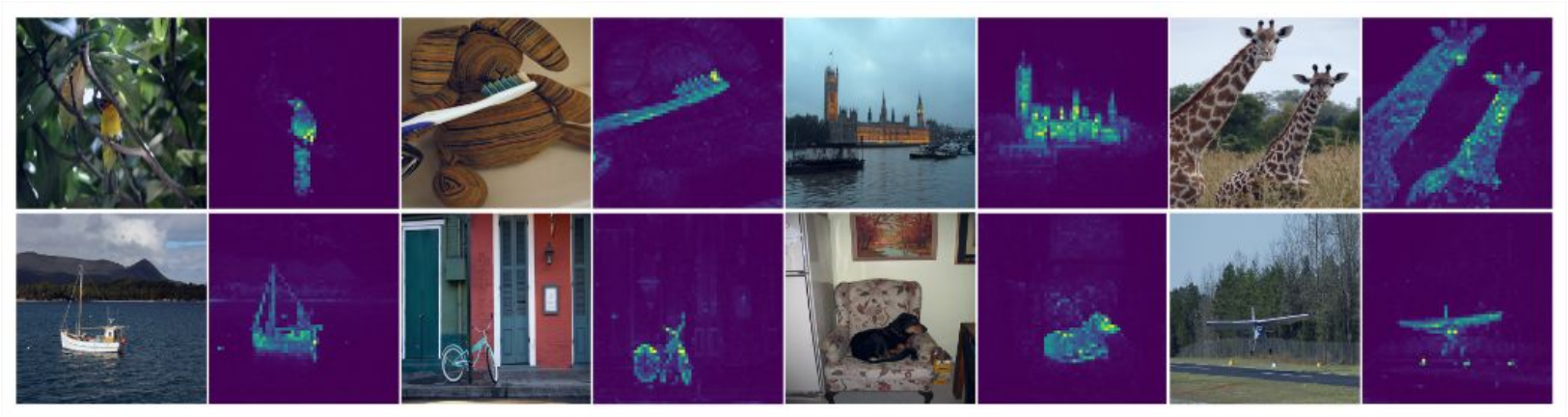
Self-distillation with no labels



Sources: "*Advancing the state of the art in computer vision with self-supervised Vision Transformers*" https://www.youtube.com/watch?v=8I1ReInsgMw
[17] **Caron, M.** et al. "*Emerging Properties in Self-Supervised Vision Transformers*". In ICCV (2021).

Pre-training: DINO

Self-distillation with no labels

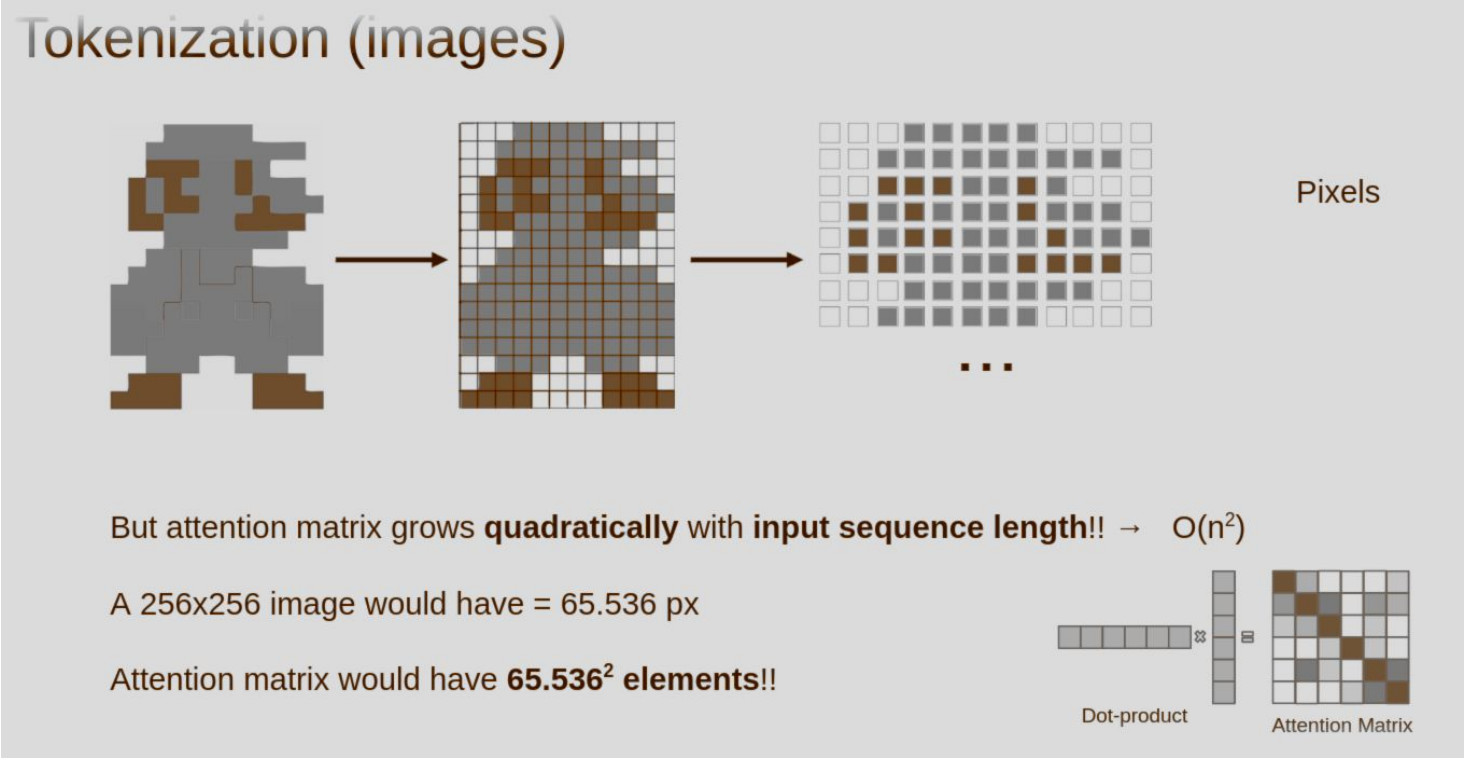


Sources: [17] **Caron, M.** et al. "*Emerging Properties in Self-Supervised Vision Transformers*". In ICCV (2021).

Efficient Transformers

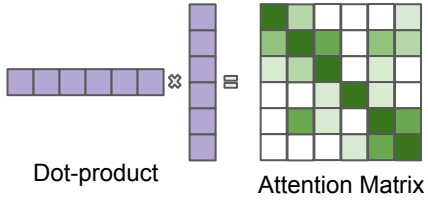
Attention matrix grows fast!

The size of the attention matrix grows **quadratically with the number of input tokens** (sequence length) T → $O(T^2)$



Attention matrix grows fast!

The size of the attention matrix grows
quadratically with the number of input tokens
(sequence length T) → $O(T^2)$



Transformers could adapt to arbitrarily long inputs,
but this limits them to do so!!

Efficient Designs

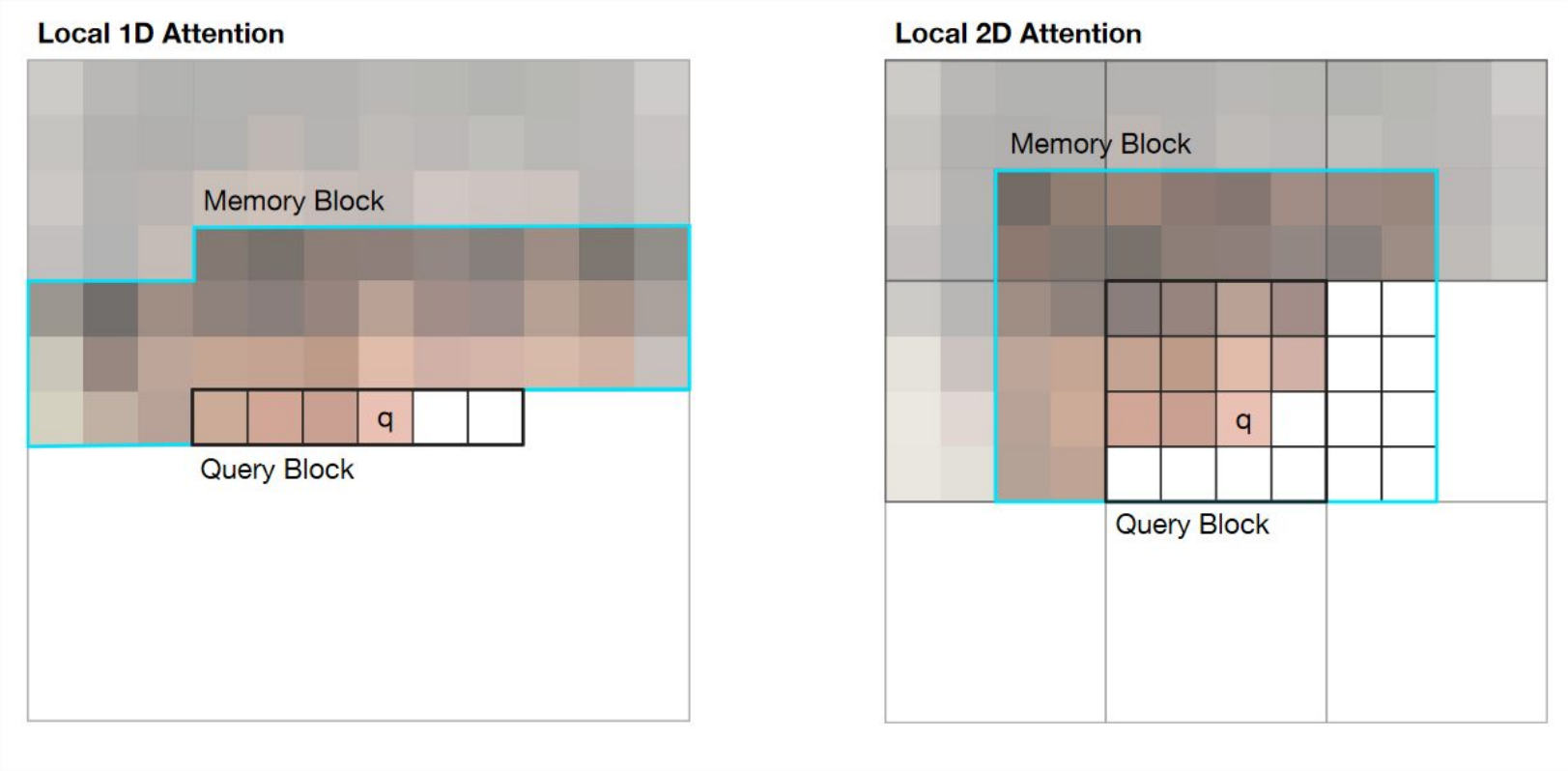
- **Limit the number of tokens** accessible in a single operation.
- Use **multiple** such operations to cover whole input.

- **Local**: Only allow to attend to neighboring tokens

- **Sparse**: Tokens can be distant, but only every other token is accessible.

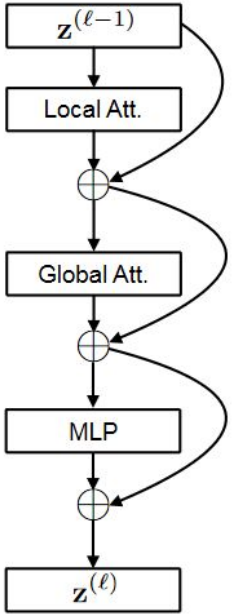
- **Hierarchy**: Progressively aggregate information.

Efficient Designs: Locality



Source: [18] **Parmar, N.** et al. "*Image transformer*". In ICML (2018).

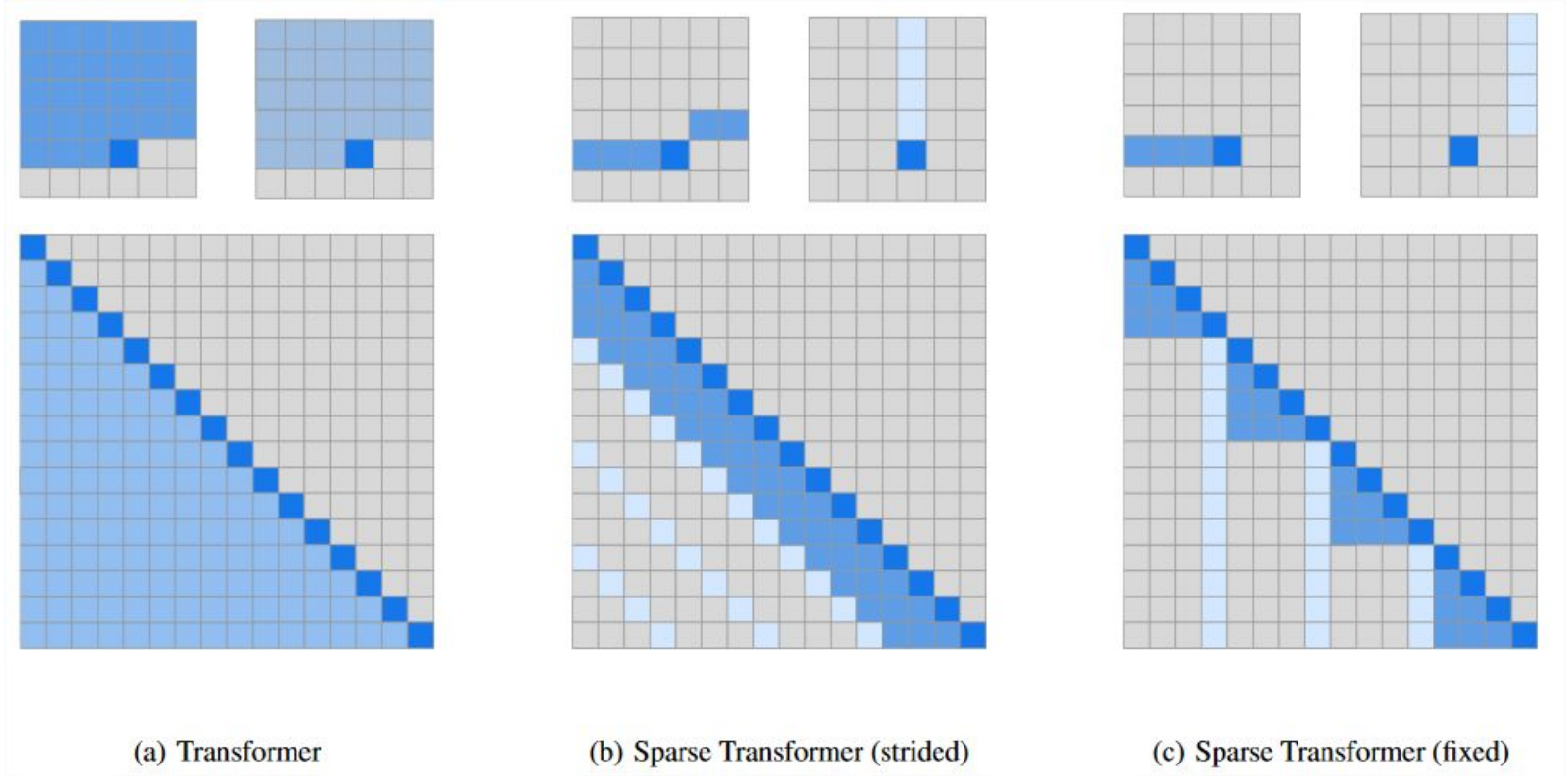
Efficient Designs: Sparsity



Sparse Local Global Attention (L+G)

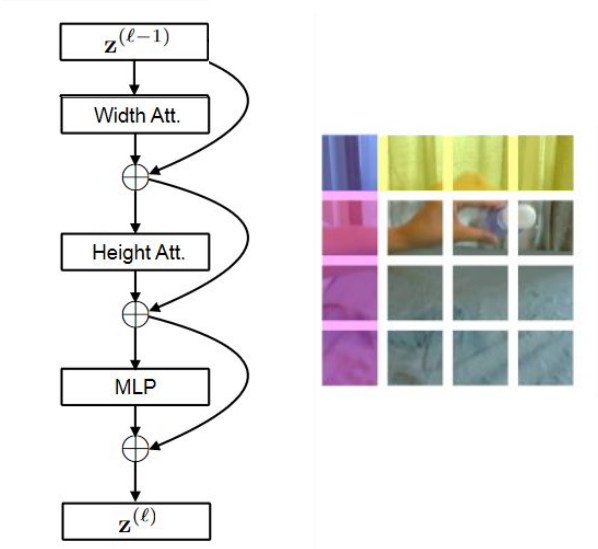
Source: [19] **Bertasius, G.** et al. "*Is Space-Time Attention All You Need for Video Understanding?*". In ICML (2021).

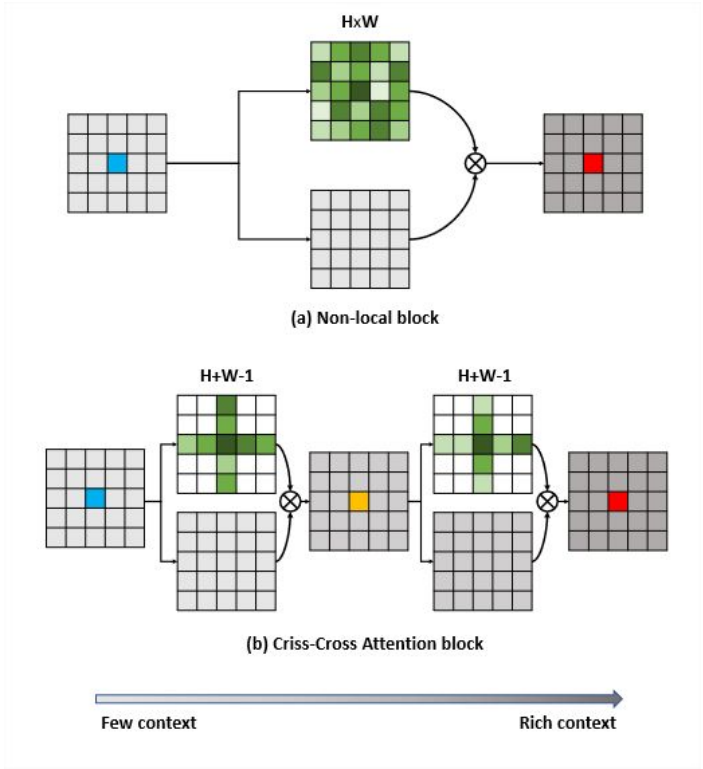
Efficient Designs: Locality/Sparsity



Source: [20] **Child, R.** et al. "*Generating long sequences with sparse transformers*". In CoRR (2019).

Efficient Designs: Axial

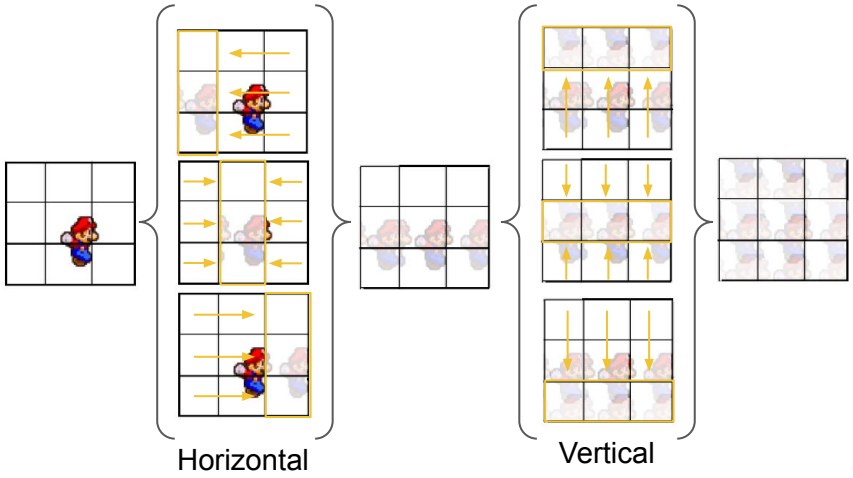




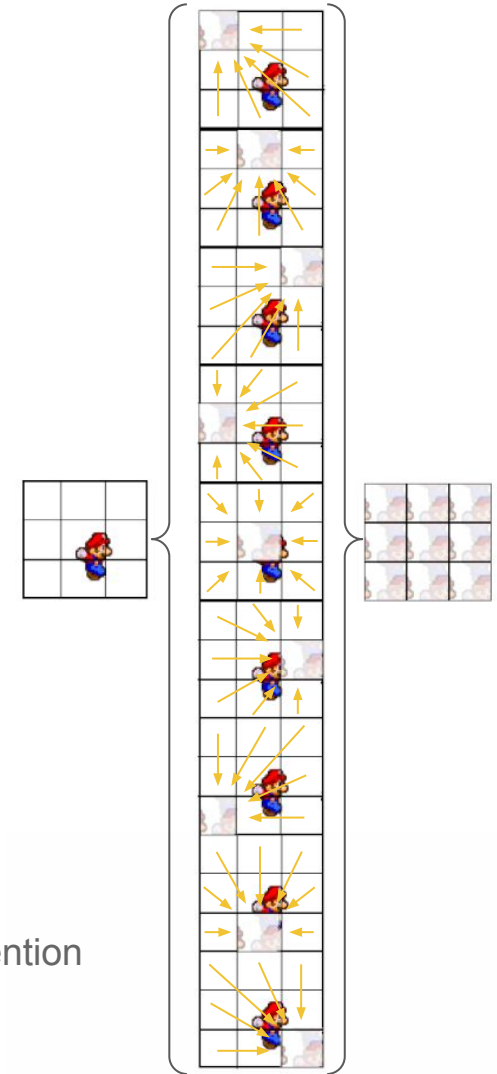
Source: [21] **Huang, Z.** et al. "*Ccnet: Criss-cross attention for semantic segmentation*". In ICCV (2019).
[19] **Bertasius, G.** et al. "*Is Space-Time Attention All You Need for Video Understanding?*". In ICML (2021).

Efficient Designs: Decomposition

Approximate full attention with two or more **consecutive** (smaller) **attention** operations



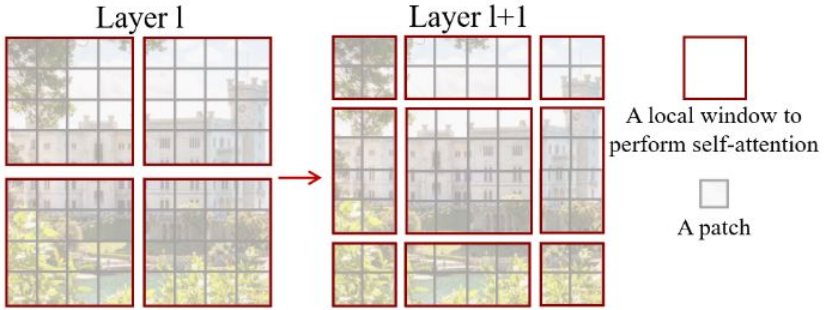
Axial Attention

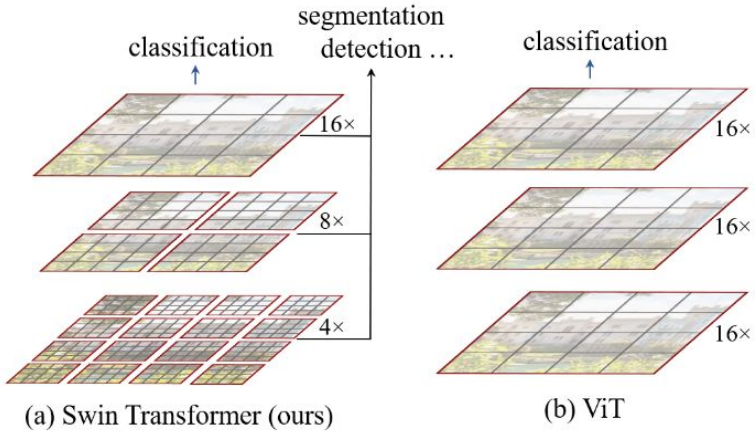


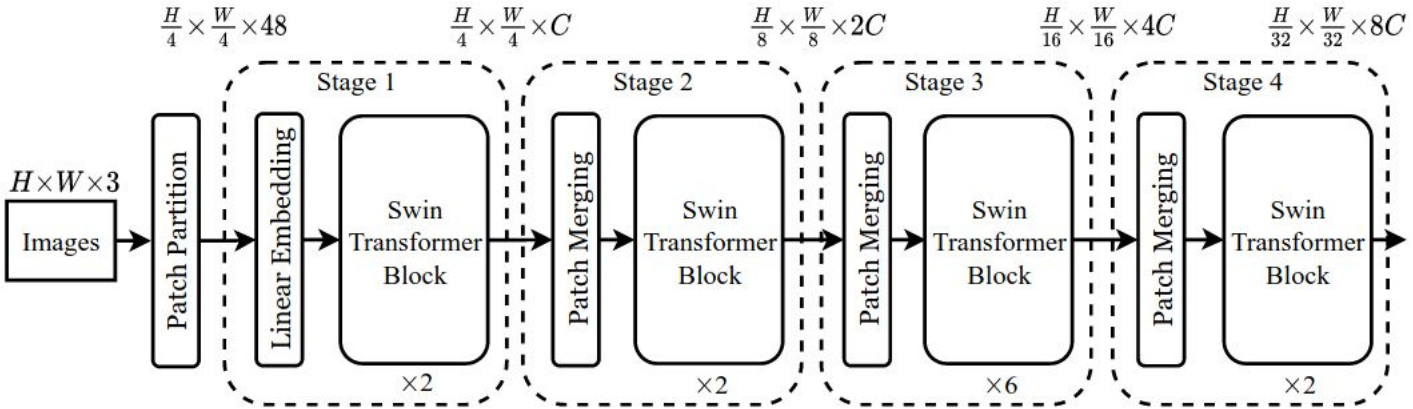
Full Attention

Efficient Designs: Hierarchy

Swin Transformer



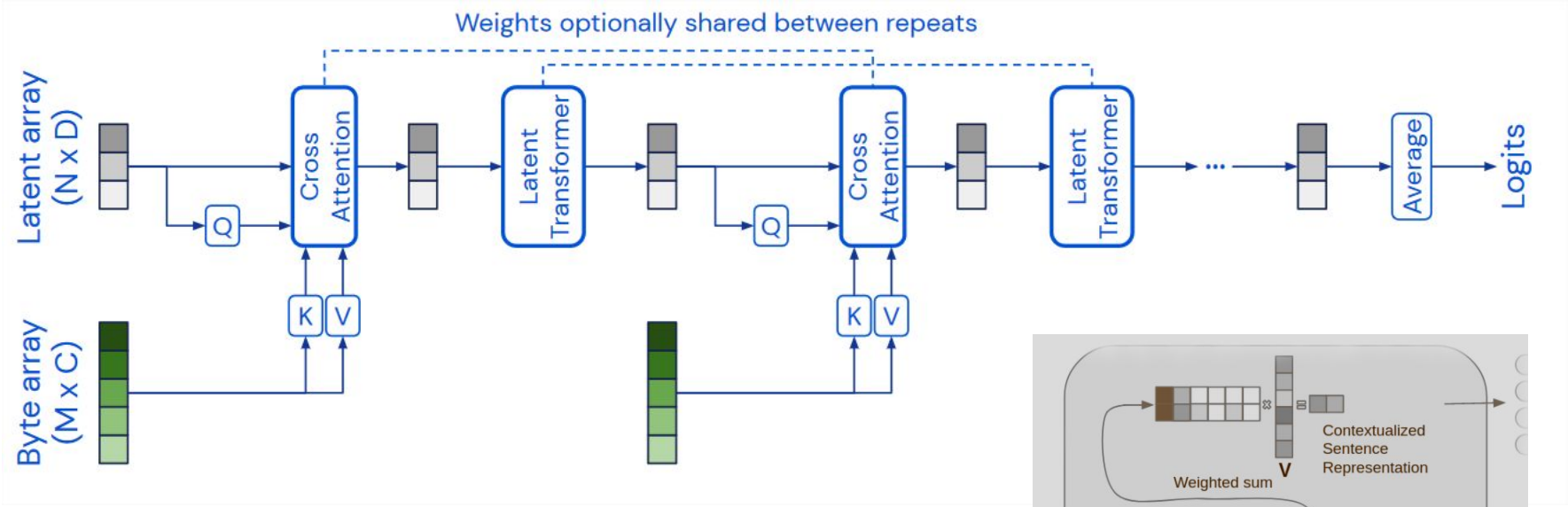


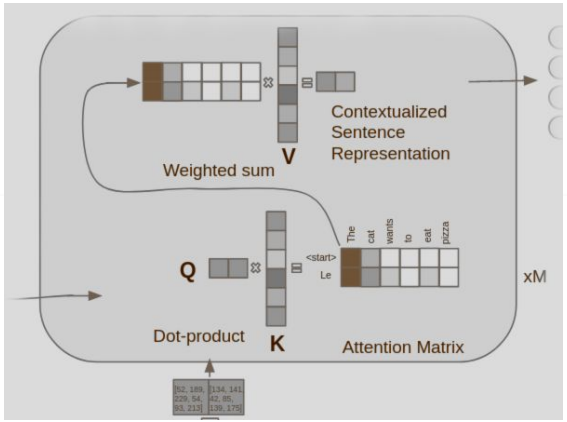


Source: [22] **Liu, Z.** et al. "*Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*". In ICCV (2021).

Efficient designs: Query based Aggregation

Perceiver

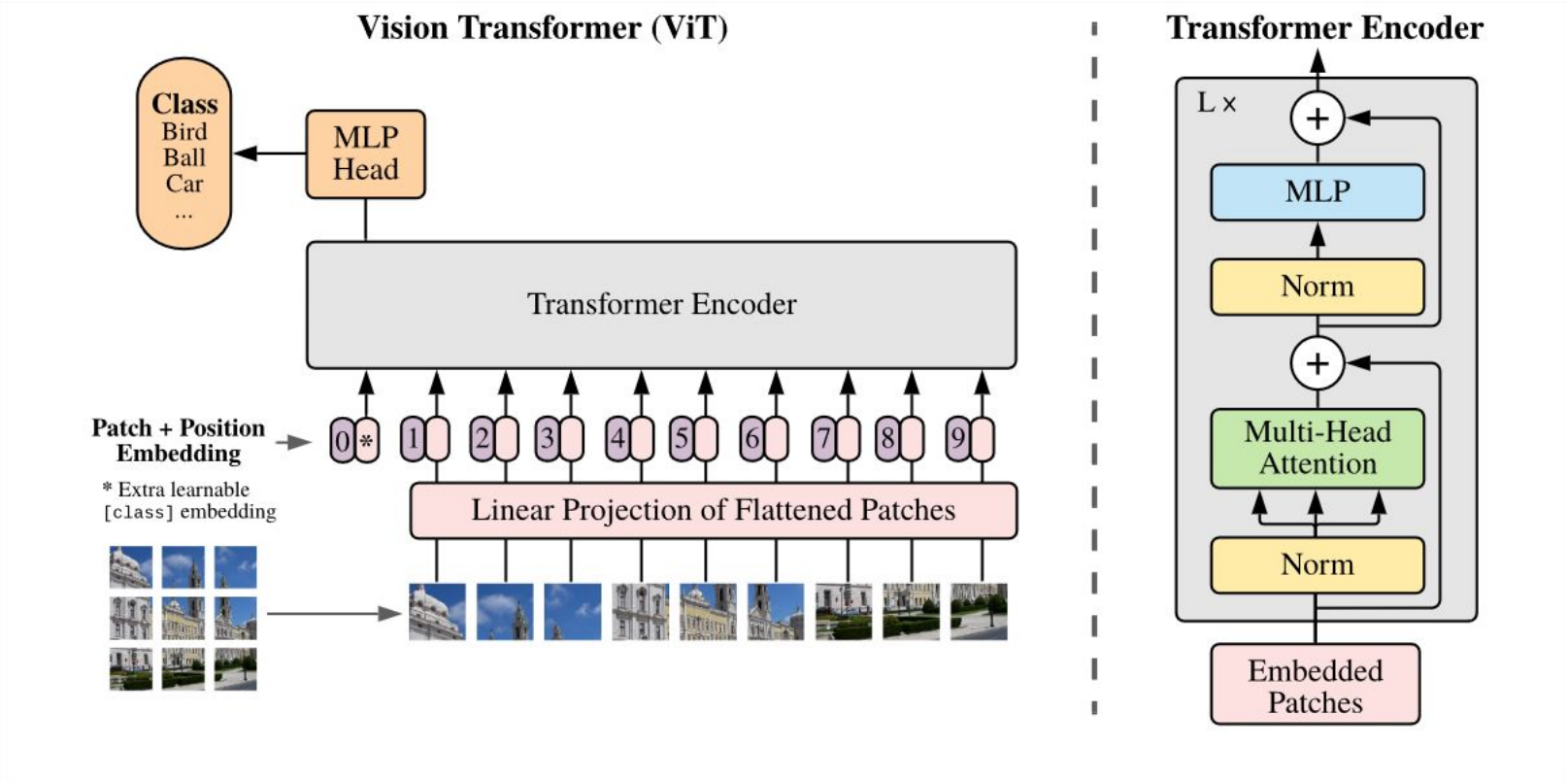




Source: [23] **Jaegle, A.** et al. "*Perceiver: General Perception with Iterative Attention*". In ICML (2021).

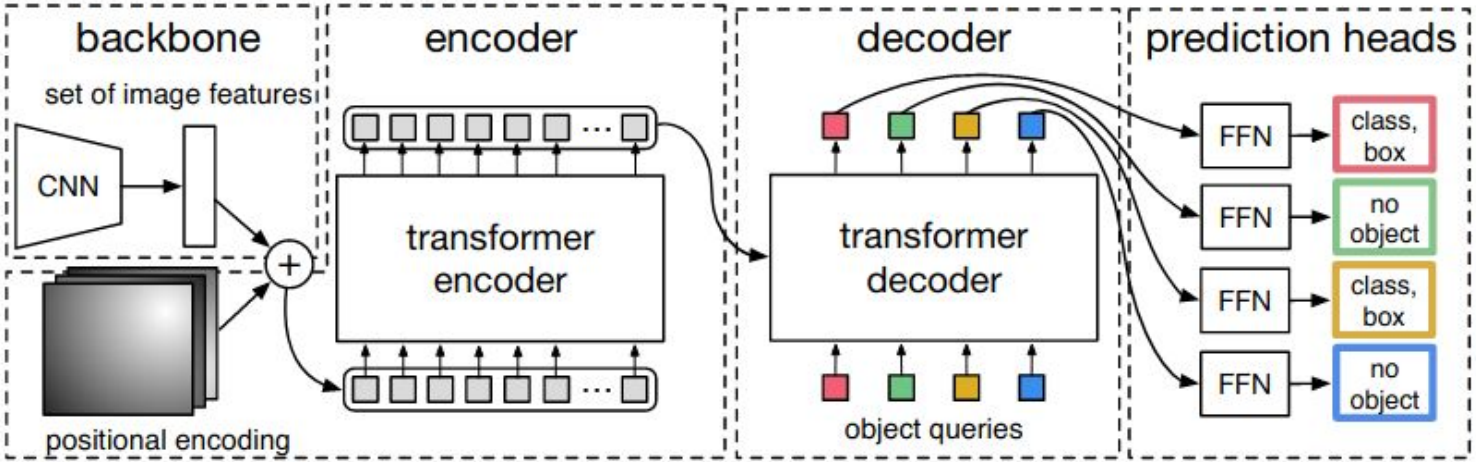
Applications

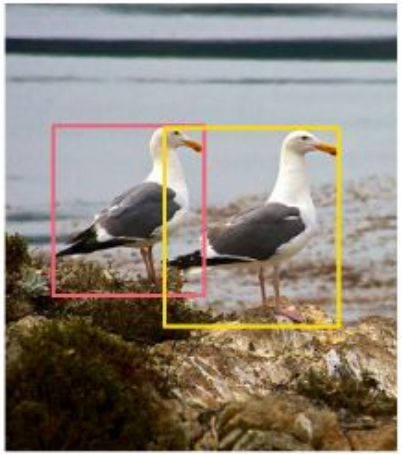
Vision Transformer (for classification)

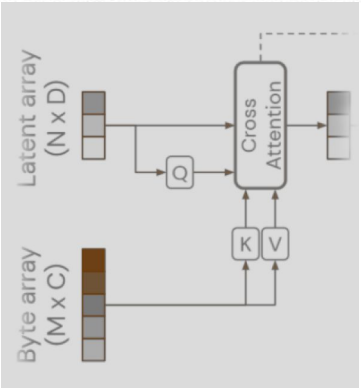


Source: [5] **Dosovitskiy, A.** et al. "*An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*". In ICLR (2020).

DETR (Transformer for object recognition)

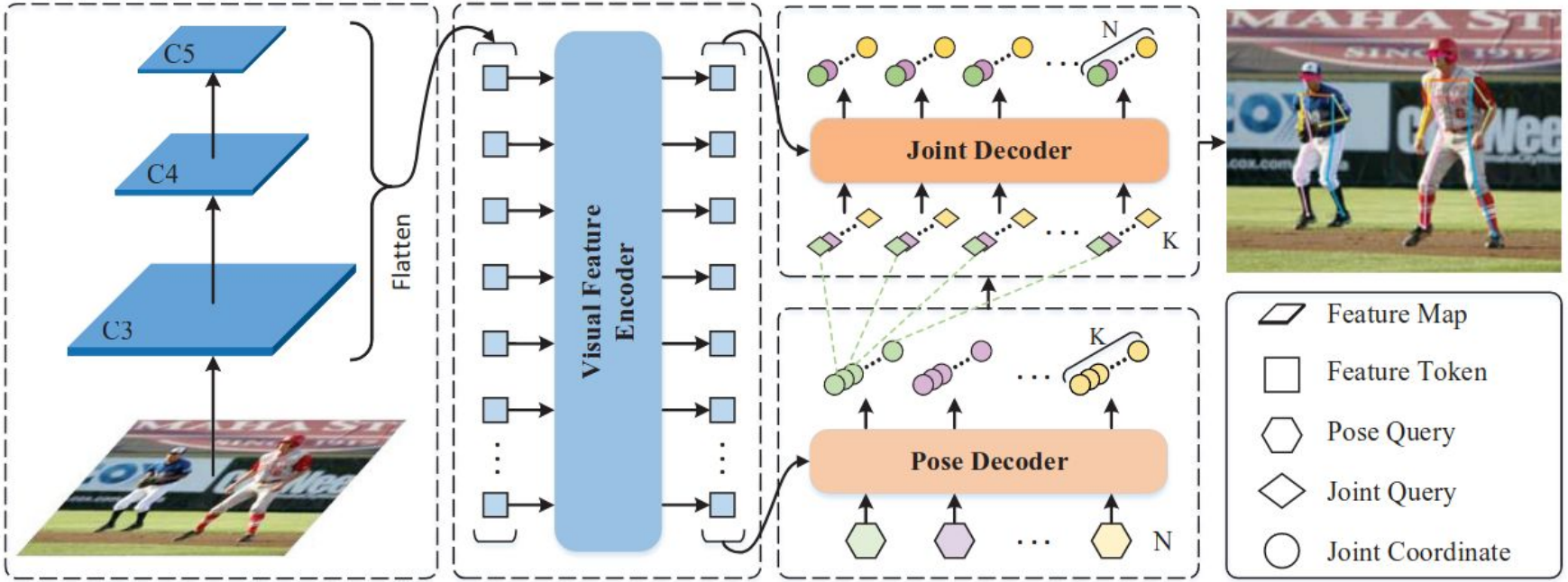






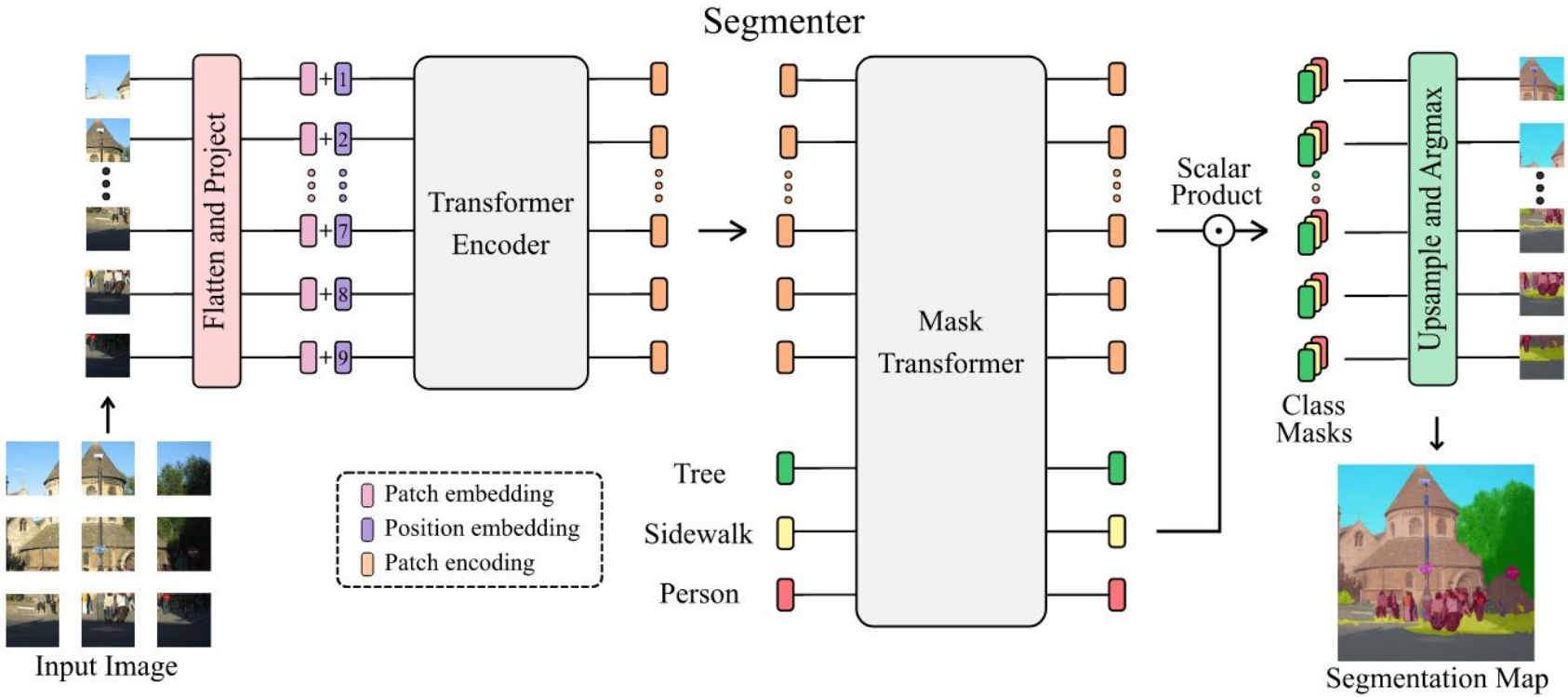
Source: [24] **Carion, N.** et al. "*End-to-end object detection with transformers*". In ECCV (2020).

Multi-person pose estimation



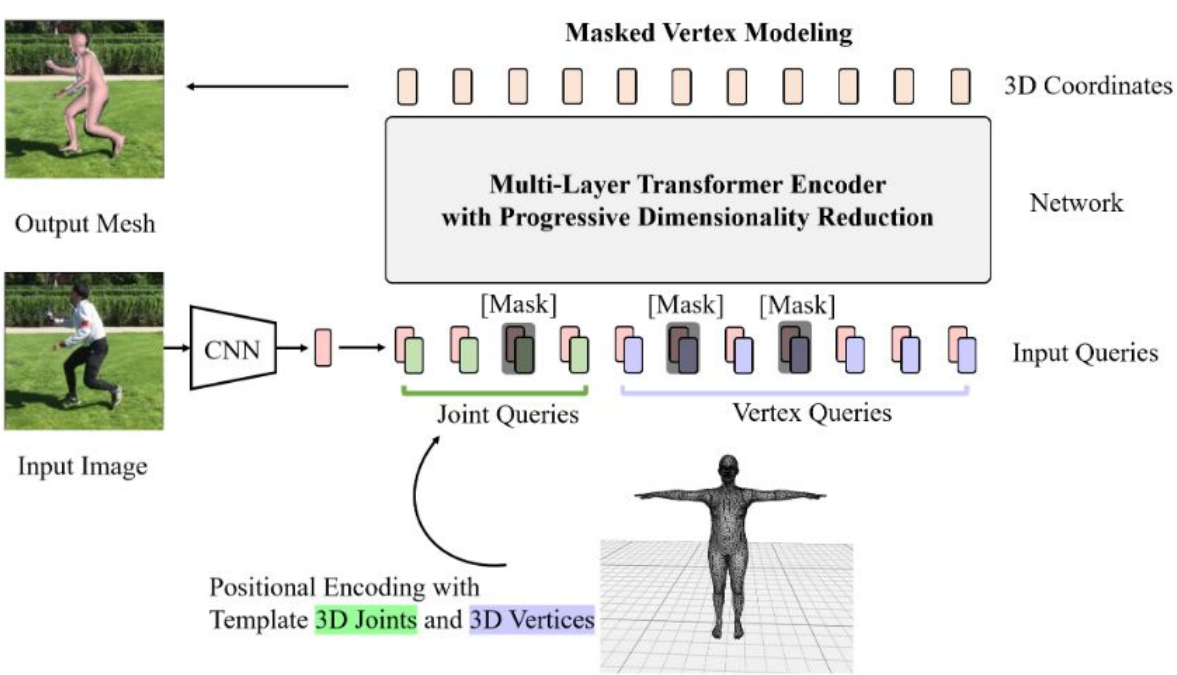
Source: [25] **Shi, D.** et al. "*End-to-End Multi-Person Pose Estimation with Transformers*". In CVPR (2022).

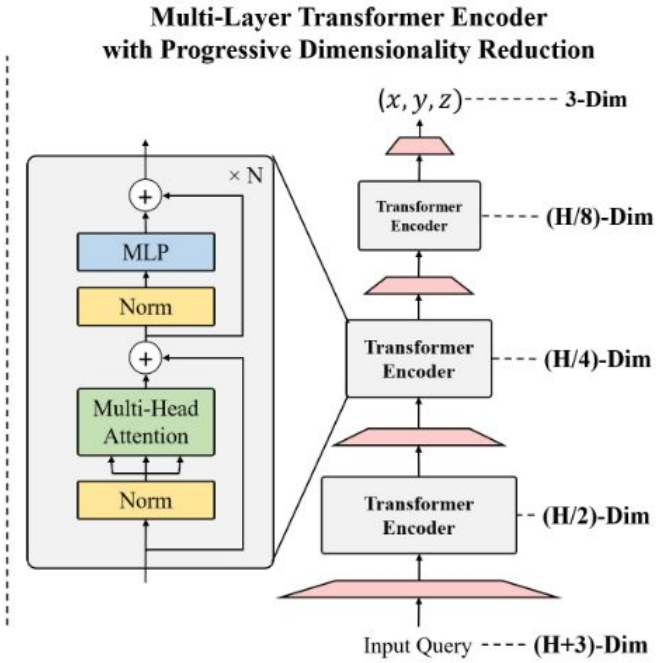
Semantic Segmentation



Source: [26] **Strudel, R.** and **Garcia, R.** et al. "*Segmenter: Transformer for Semantic Segmentation*". In ICCV (2021).

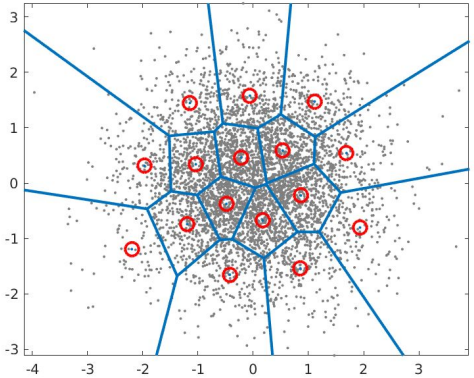
Mesh Reconstruction



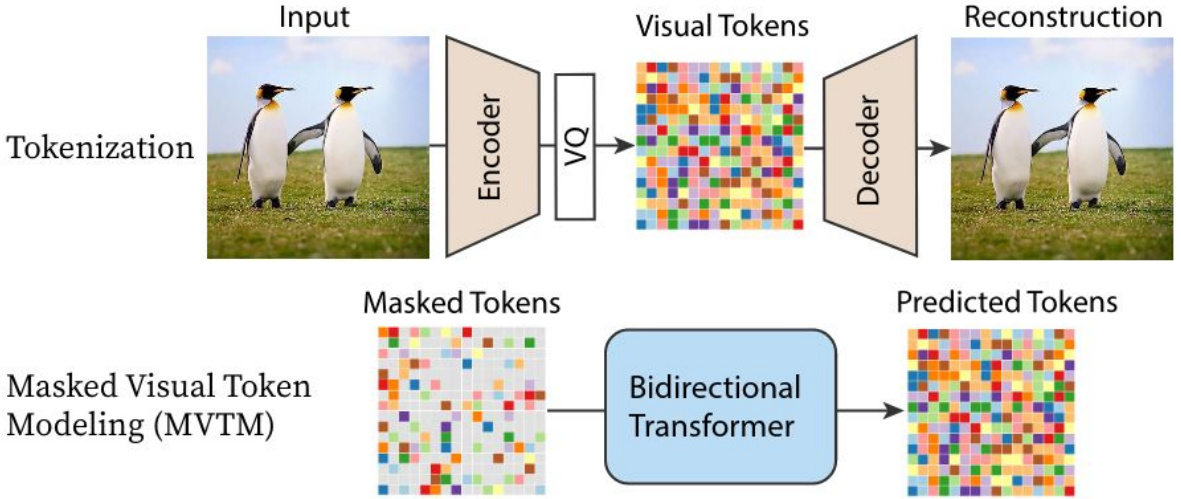


Source: [27] **Lin, K.** et al. "*End-to-end human pose and mesh reconstruction with transformers*". In CVPR (2021).

Image Generation



Quantization

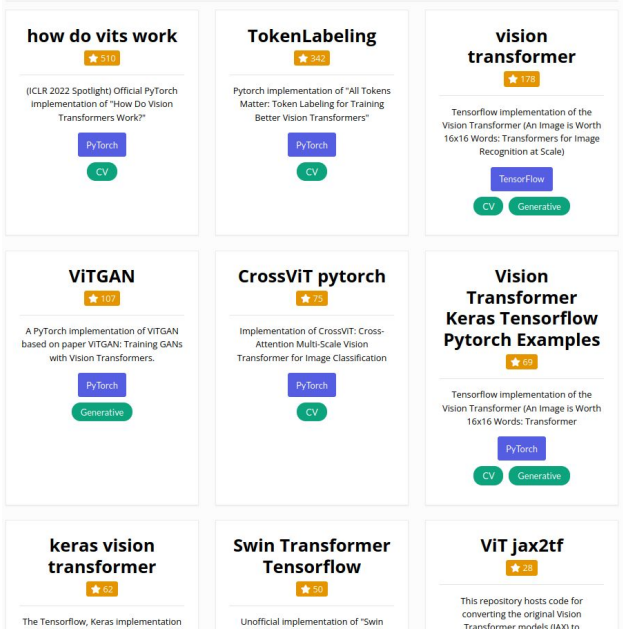


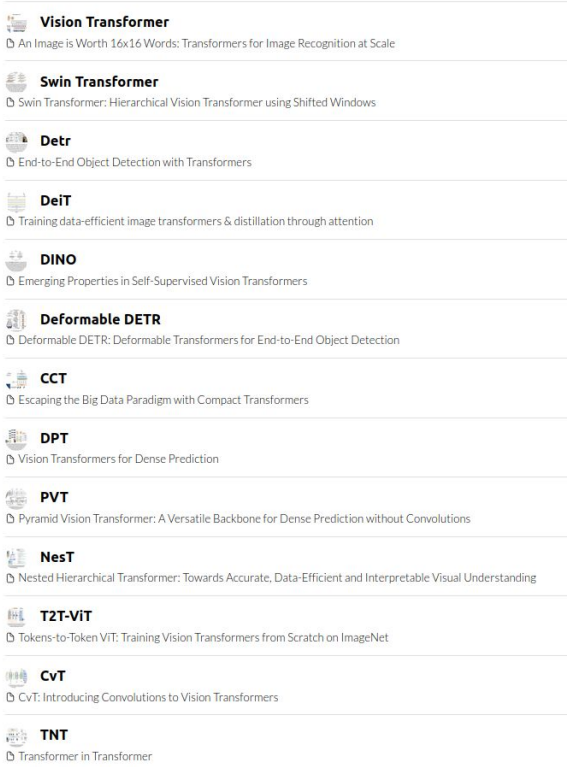
Clusters image source: https://speechprocessingbook.aalto.fi/Modelling/Vector_quantization_VQ.html
Source: [4] **Chang, H.** et al. "*Maskgit: Masked generative image transformer*". In CVPR (2022).

How to Transformer?

Use pre-trained models! GitHub!

https://colab.research.google.com/





Hugging Face

huggingface.co/

Model Zoo

modelzoo.co/

Papers With Code

paperswithcode.com/

References

Publications

[1] **Bahdanau, D.** et al. "*Neural machine translation by jointly learning to align and translate".* In CoRR (2014).
[2] **Wang, X.** et al. "*Non-local neural networks.*" In CVPR (2018).
[3] **Zhang, B.** et al. "*Styleswin: Transformer-based gan for high-resolution image generation.*". In CVPR (2022).
[4] **Chang, H.** et al. "*Maskgit: Masked generative image transformer*". In CVPR (2022).
[5] **Dosovitskiy, A.** et al. "*An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*". In ICLR (2020).
[6] **Yu, J.** et al. "*Coca: Contrastive captioners are image-text foundation models.*". In CoRR (2022).
[7] **Yan, S.** et al. "*Multiview transformers for video recognition*". In CVPR (2022).
[8] **Vaswani, A.** et al. "*Attention is all you need*". In NeurIPS (2017)
[9] **He, K.** et al. "*Deep residual learning for image recognition*". In CVPR (2016).
[10] **Lee, S.** et al. "*Parameter Efficient Multimodal Transformers for Video Representation Learning*". In ICLR (2020).
[11] **Li, K.** et al. "*UniFormer: Unified Transformer for Efficient Spatiotemporal Representation Learning*". In ICLR (2022).
[12] **Ramachandran, P.** et al. "*Stand-Alone Self-Attention in Vision Models*". In NeurIPS (2019).
[13] **Li, L.** et al. "*HERO: Hierarchical Encoder for Video+ Language Omni-representation Pre-training*". In EMNLP (2020).
[14] **Atito, S.** et al. "*Sit: Self-supervised vision transformer*". In CoRR (2021).
[15] **He, K.** et al. "*Masked autoencoders are scalable vision learners*". In CVPR (2022).
[16] **Oord, A. V. D.** et al. "*Representation learning with contrastive predictive coding*". In CoRR (2018).
[17] **Caron, M.** et al. "*Emerging Properties in Self-Supervised Vision Transformers*". In ICCV (2021).
[18] **Parmar, N.** et al. "*Image transformer*". In ICML (2018).
[19] **Bertasius, G.** et al. "*Is Space-Time Attention All You Need for Video Understanding?*". In ICML (2021).
[20] **Child, R.** et al. "*Generating long sequences with sparse transformers*". In CoRR (2019).
[21] **Huang, Z.** et al. "*Ccnet: Criss-cross attention for semantic segmentation*". In ICCV (2019).
[22] **Liu, Z.** et al. "*Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*". In ICCV (2021).
[23] **Jaegle, A.** et al. "*Perceiver: General Perception with Iterative Attention*". In ICML (2021).
[24] **Carion, N.** et al. "*End-to-end object detection with transformers*". In ECCV (2020).
[25] **Shi, D.** et al. "*End-to-End Multi-Person Pose Estimation with Transformers*". In CVPR (2022).
[26] **Strudel, R.** and **Garcia, R.** et al. "*Segmenter: Transformer for Semantic Segmentation*". In ICCV (2021).
[27] **Lin, K.** et al. "*End-to-end human pose and mesh reconstruction with transformers*". In CVPR (2021).

Blog Posts

Vincent Dumoulin, Francesco Visin "A guide to convolution arithmetic for deep learning" (2016) https://github.com/vdumoulin/conv_arithmetic
Mohammed Terry-Jack "*Deep Learning: The Transformer*" (2019) https://medium.com/@b.terryjack/deep-learning-the-transformer-9ae5e9c5a190
Lilian Weng "*Attention? Attention!*" (2018) https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html
Amirhossein Kazemnejad "*Transformer Architecture: The Positional Encoding*" (2019) https://kazemnejad.com/blog/transformer_architecture_positional_encoding/
Gillaume Klein et al. "*The Annotated Transformer*" (2018) http://nlp.seas.harvard.edu/2018/04/03/attention.html
Lilian Weng "*The Transformer Family*" (2020) https://lilianweng.github.io/lil-log/2020/04/07/the-transformer-family.html
Jay Alammar "*The Illustrated Transformer*" (2018) https://jalammar.github.io/illustrated-transformer/