ELEG 5491: Introduction to Deep Learning

Attention and Transformer

Prof. LI Hongsheng

Office: SHB 428
e-mail: hsli@ee.cuhk.edu.hk
web: https://dl.ee.cuhk.edu.hk

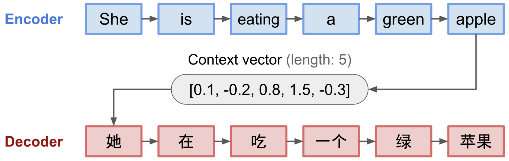
Department of Electronic Engineering
The Chinese University of Hong Kong

March 2023

Outline

Revisit of seq2seq model

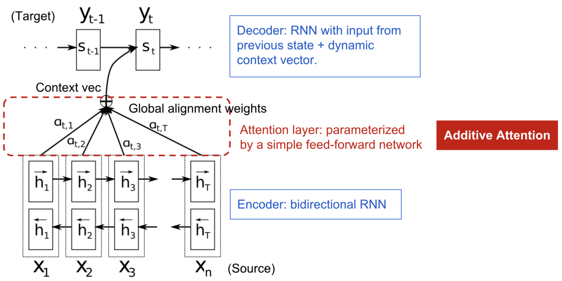
- The conventional seq2seq model is proposed for neural machine translation and generally follows an encoder-decoder architecture
- The **encoder** converts the input sequence into a sentence embedding (or context vector, or "thought" vector) of a fixed length (dimension)
- The embedding vector is expected to be a comprehensive summary of the *whole* input sequence
- The **decoder** takes only the sentence embedding from the encoder as input to emit the output sequence
- Both the encoder and decoder sub-networks can be modeled as recurrent neural networks and can use LSTM or GRU units



- The major **drawback** of the seq2seq model: this fixed-length embedding vector is incapable of remembering the whole long sentences. Often it is more likely to forget the early parts of the input sequence

Attention for NMT

- As the context vector might not be able to capture the information of the whole sentence, the attention mechanism [Bahdanau et al., 2015] explicitly builds word-level alignment between the input and output sequences



- $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ – input (source) sequence of length n
- $\mathbf{y} = [y_1, y_2, \ldots, y_m]$ – output (target) sequence of length m
- The encoder is a **bidirectional RNN** having forward hidden states $\overrightarrow{h}_i$ and backward hidden states $\overleftarrow{h}_i$, and generating the hidden state at position i

$$h_i = [\overrightarrow{h}_i^T ; \overleftarrow{h}_i^T]^T, \quad i = 1, \ldots, n$$

Attention for NMT

- The decoder has hidden state $s_t = f(s_{t-1}, y_{t-1}, c_t)$ for the output word at position t for $t = 1, \dots, m$

$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i$	Context vector for output y_t
$\alpha_{t,i} = \operatorname{align}(y_t, x_i)$	How well two words y_t and x_i are aligned
$= \frac{\exp(\operatorname{score}(s_{t-1}, h_i))}{\sum_{i'=1}^{n} \exp(\operatorname{score}(s_{t-1}, h_{i'}))}$	Softmax of some predefined alignment score

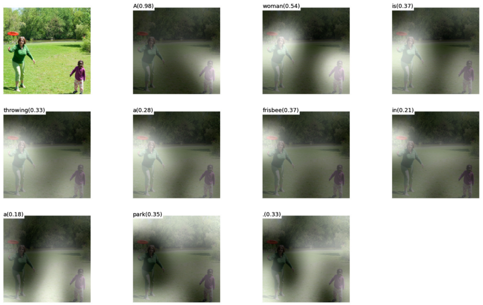
- c_t – the sum of hidden states of the input sequence weighted by the alignment scores, based on which, the class or regression prediction of each output position can be made
- The alignment model assigns a score $\alpha_{t,i}$ to the pair (y_t, x_i) at input position t and output position i
- score – a 2-layer feed-forward network (or MLP) estimating the affinity between s_{t-1} (just before emitting y_t) and h_i

$$\operatorname{score}(s_{t-1}, h_i) = W_1(\tanh(W_2 [s_{t-1}; h_i] + b_2) + b_1)$$

W_1, W_2, b_1, b_2 are weight matrices and biases to be learned

Image Captioning

- By changing the encoder to a CNN model, we can output an image caption according to the contents of an input image
- The above mentioned attention mechanism can also be used to align different image regions with the output words as in the Show, attend and tell paper
- The alignment weights $\alpha_{t,i}$ for each output position t are normalized across the whole 2D spatial image plane. Each input index i indices a pixel (x, y) in the 2D feature maps from the visual CNN



Different Alignment Score Functions

- Different alignment (or similarity measurement) functions

Type	Alignment Score Function
Cosine similarity	$\text{score}(s_t, h_i) = \cos(s_t, h_i)$
Concatenation-based*	$\text{score}(s_t, h_i) = W_1 \tanh(W_2 [s_t; h_i])$
General	$\text{score}(s_t, h_i) = s_t^T W h_i$
Dot-product	$\text{score}(s_t, h_i) = s_t^T h_i$
Scaled dot-product†	$\text{score}(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{\text{dim}}}$

*Bias vectors are not shown. †dim is the dimension of the vectors s_t and h_i.

Transformer and Key, Query, Value Features

- Transformer was proposed in "Attention is All You Need" paper and was one of the most impactful and interesting papers in 2017
- It proposes a series of improvements to the conventional attention and can be used in any sequence modeling tasks
- In the previous attention mechanisms, $\{h_i\}_{i=1}^m$ are used for both affinity/similarity estimation and information aggregation (e.g., $c_t = \sum_{i=1}^n \alpha_{t,i} h_i$) at each output position t
- Given a feature sequence $X \in \mathbb{R}^{n \times k}$ of length n, a Transformer attention layer first converts them into key, query, value features $K \in \mathbb{R}^{n \times d}$, $Q \in \mathbb{R}^{n \times d}$, $V \in \mathbb{R}^{n \times d}$ with linear projections (fully-connected layers)

$$K = W_k X + b_k$$
$$Q = W_q X + b_q$$
$$V = W_v X + b_v$$

  $W^Q, W^K, W^V, b^Q, b^K, b^V$ are learnable parameters
- k_i, q_i, v_i denote the key, query, value feature vectors of the ith input

Transformer and Key, Query, Value Attention

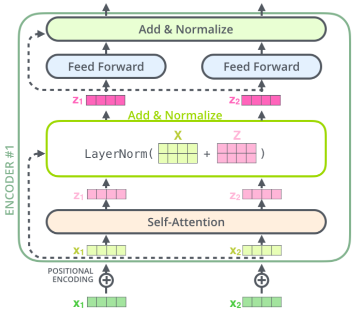
- Unlike conventional attention mechanism, where the hidden states are used for both similarity estimation and information aggregation, a Transformer attention layer calculates pairwise similarities with query and key features and aggregate value features
- The transformer adopts the scaled dot-product attention: the output is a weighted sum of the values, where the weight assigned to each value is determined by the dot-product of the query with all the keys:

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

- $\frac{QK^T}{\sqrt{d}} \in \mathbb{R}^{n\times n}$ stores similarities between every pair of $(q_i,\, k_j)$ at (i, j) of the resulting matrix
- The $\mathrm{softmax}$ normalization is performed for each row
- The Transformer attention result is an $n \times d$ matrix. For the ith row, it weightedly aggregates value features from different positions of the sequence, $v_1, v_2, \dots, v_n$

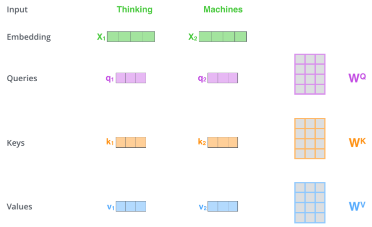
Transformer Encoder

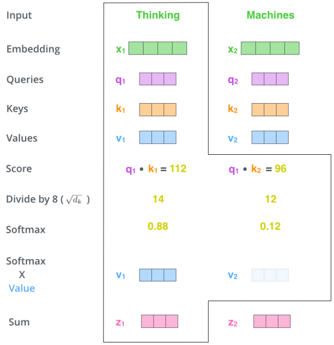
- For both the encoder and decoder of the conventional RNN, hidden states h_i at position i are obtained from its immediate and previous hidden states h_{i-1}
- For Transformer encoder, the representation at position i can receives information from all positions of the input sequence at the same layer
- Self-attention mechanism is adopted: Q, K, V are generated from the same sequence features X at the same layer
- The transformer encoder can consists of multiple transformer blocks



The Illustration of Self-attention in the Encoder

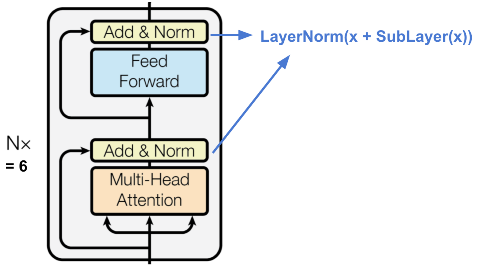
- The self-attention in Transformer encoder





Transformer Block Design

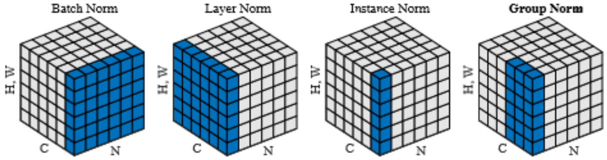
- For conventional RNN with attention mechanism, each time step just has a single FC layer or an 2-layer/3-layer MLP for generating hidden states of each time step
- Transformer adopts a block design



- Each block consists of two sub-layers, a scaled dot-product attention sub-layer and a feed-forward sub-layer, with residual connection and layer normalization (a substitute for BN)
- **Feed-forward network:** Two linear layers with a ReLU activation between them

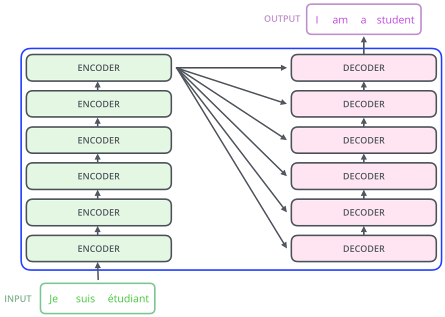
Layer Normalization

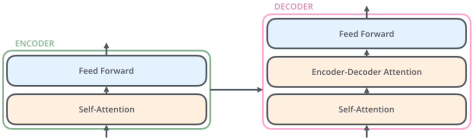
- The effect of batch normalization is dependent on the mini-batch size and it is not obvious how to apply it to recurrent neural networks
- Batch normalization significantly reduces the training time in feed-forward neural networks
- Layer normalization computes the mean and variance used for normalization from all of the neurons in each layer on a single training sample/instance
- A graphical illustration when layer normalization is applied to images



Encoder-decoder with Transformer Block

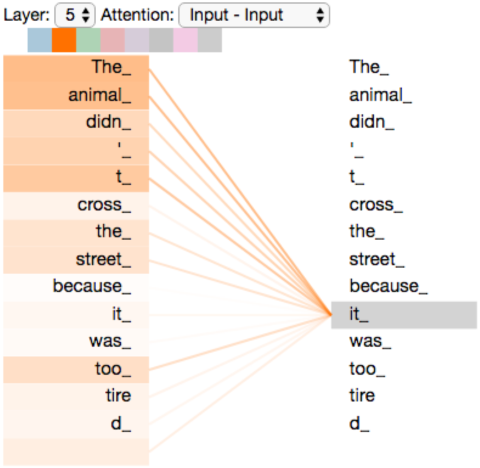
- The encoder and decoder consist of 6 transformer blocks respectively, whose architectures are slightly different





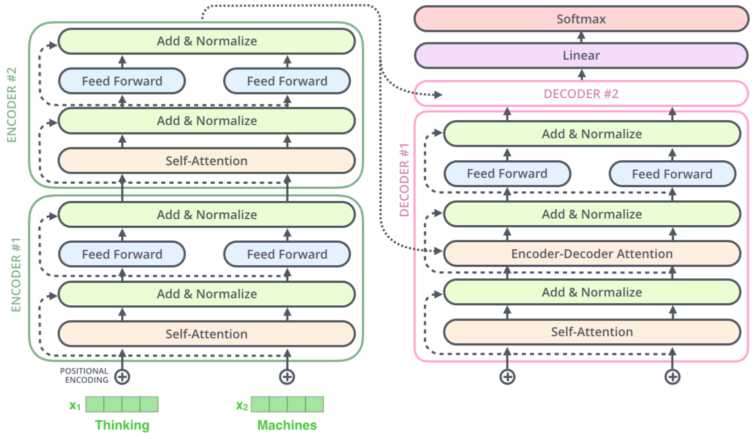
Visualization of self-attention in Transformer Encoder

"The animal didn't cross the street because it was too tired"



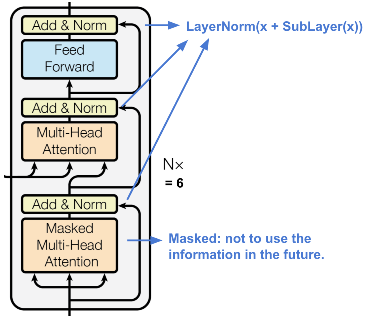
The Encoder-decoder Architecture

- The encoder-decoder architecture



Transformer Decoder

- The original version stacks 6 Transformer decoder layers
- The first multi-head attention sub-layer is modified to prevent positions from attending to subsequent positions, as we don't want to look into the future of the target sequence when predicting the current position
- The first attention sub-layer only attends decoder features. It is therefore self-attention
- The second attention sub-layers attends all final encoder features. It is called *cross-attention*

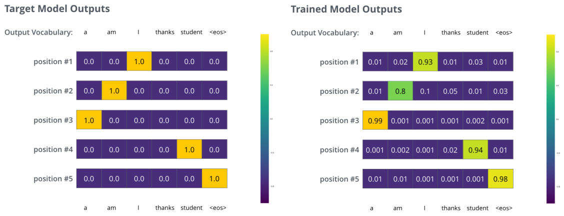

Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
N×
= 6
LayerNorm(x + SubLayer(x))
Masked: not to use the information in the future.

The Animation of Decoding from Transformer

The Animation of Decoding from Transformer

Transformer Output

- Transformer target and actual outputs



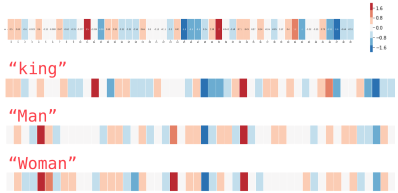
Target Model Outputs

Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.0	0.0	1.0	0.0	0.0	0.0
position #2	0.0	1.0	0.0	0.0	0.0	0.0
position #3	1.0	0.0	0.0	0.0	0.0	0.0
position #4	0.0	0.0	0.0	0.0	1.0	0.0
position #5	0.0	0.0	0.0	0.0	0.0	1.0

Trained Model Outputs

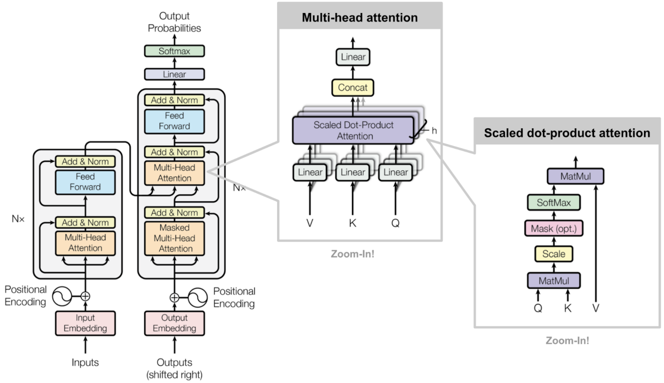
Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.01	0.02	0.93	0.01	0.03	0.01
position #2	0.01	0.8	0.1	0.05	0.01	0.03
position #3	0.99	0.001	0.001	0.001	0.002	0.001
position #4	0.001	0.002	0.001	0.02	0.94	0.01
position #5	0.01	0.01	0.001	0.001	0.001	0.98

- What about transformer inputs? There are pre-trained word embeddings that convert each word into a vector (such as Word2Vec and Glove)



Full Transformer Architecture

- In the decoder, the first masked multi-head attention aggregates information from only previous output words
- The second multi-head attention uses queries generated from the first multi-head attention, computes their similarities to the encoder's key features, and aggregates the encoder's value features



Multi-head Attention

- Instead of performing a single attention function, the authors found it beneficial to linearly project the queries, keys and values for h times with different linear projections (fully-connected layers without sharing parameters)
- On each set of the query, key, value features, the attention functions are performed in parallel with separate sets of parameters
- The obtained features are concatenated, resulting the final values

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}\left(\mathrm{head}_1,\ldots,\mathrm{head}_h\right)W^O,$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

- We can set $h=8$. Then the projections have learnable parameter matrices $W_i^Q \in \mathbb{R}^{\frac{d}{8}\times d}$, $W_i^K \in \mathbb{R}^{\frac{d}{8}\times d}$, $W_i^V \in \mathbb{R}^{\frac{d}{8}\times d}$, $W^O \in \mathbb{R}^{(\frac{d}{8}\times 8)\times d}$ (we didn't show biases here)

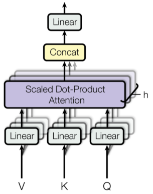
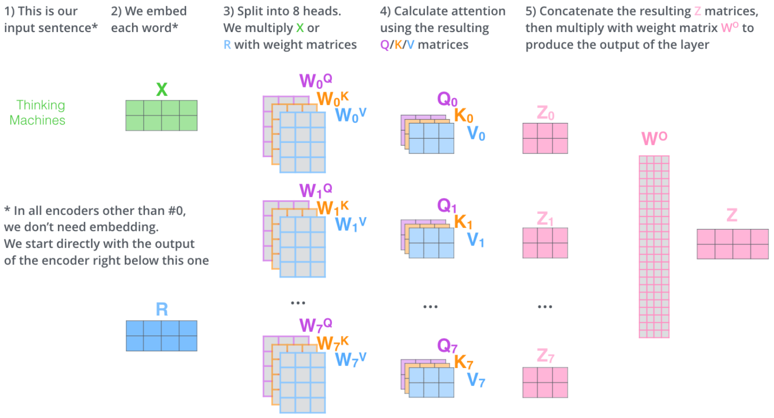


Illustration of Computation of Multi-head Attention

- The actual computation of multi-head attention


1) This is our input sentence*
2) We embed each word*
3) Split into 8 heads. We multiply X or R with weight matrices
4) Calculate attention using the resulting Q/K/V matrices
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer
Thinking Machines
X
R
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one
W0Q W0K W0V
W1Q W1K W1V
W7Q W7K W7V
Q0 K0 V0
Q1 K1 V1
Q7 K7 V7
Z0
Z1
Z7
WO
Z

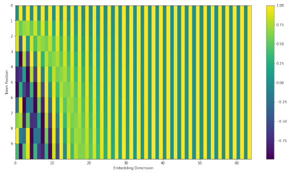
Positional Encoding

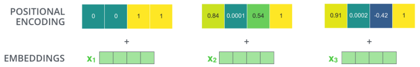
- If the Q, K, V features are just encoded from the sequence contents, we cannot capture their positional information
- For instance, “Do you like apple” and “You do like apple” have totally different meanings
- The Transformer adds a positional encoding vector to each input embedding

$$\mathrm{PE}_{(\mathrm{pos},2i)} = \sin(\mathrm{pos}/10000^{2i/d})$$

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

PE is an d-dimensional feature vector, i denotes the i-th feature dimension, pos is the position of the sequence



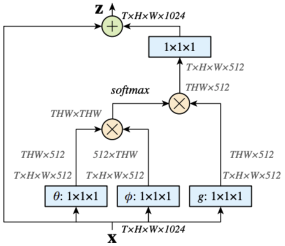


Non-local Networks

- The non-local network [Want et al. 2018] is a direct extension of Transformer attention to image and video understanding
- It was originally proposed for video classification. The non-local operation is formulated as
$$\mathbf{y}_i = \frac{1}{\mathcal{C}(\mathbf{x})} \sum_{\forall j} f\left(\mathbf{x}_i, \mathbf{x}_j\right) g\left(\mathbf{x}_j\right)$$
- $\mathbf{x}$ – the input signal (image, sequence, video; often their features)
- $\mathbf{y}$ – the output features of the same size (might have different channels) of $\mathbf{x}$
- i and j – the index of the output and input positions (in space, time, or spacetime), respectively
- $g(\mathbf{x}_j)$ – a feature representation of the input signal at position j
- $f(\mathbf{x}_i, \mathbf{x}_j)$ – a pairwise function f computing a scalar between i and all j
- $\mathcal{C}$ – normalization term

Similarity Functions

- Similar to different alignment (similarity measurement) functions in the attention mechanism, there are different choices of the f function
- **Gaussian** – $f\left(\mathbf{x}_i, \mathbf{x}_j\right) = e^{\mathbf{x}_i^T \mathbf{x}_j}$, where $\mathcal{C}(\mathbf{x}) = \sum_{\forall j} f\left(\mathbf{x}_i, \mathbf{x}_j\right)$
- **Embedded Gaussian** – $f\left(\mathbf{x}_i, \mathbf{x}_j\right) = e^{\theta(\mathbf{x}_i)^T \phi(\mathbf{x}_j)}$, where θ and ϕ are two learnable linear projections
- **Dot product** – $f\left(\mathbf{x}_i, \mathbf{x}_j\right) = \theta\left(\mathbf{x}_i\right)^T \phi\left(\mathbf{x}_j\right)$. Similar to the dot-product used in Transformer attention
- **Concatenation** – $f\left(\mathbf{x}_i, \mathbf{x}_j\right) = \mathrm{ReLU}\left(\mathbf{w}_f^T \left[\theta\left(\mathbf{x}_i\right), \phi\left(\mathbf{x}_j\right)\right]\right)$
- g can be considered as the value features, while θ and ϕ can be considered as query and key features



Non-local Block

- Non-local operation

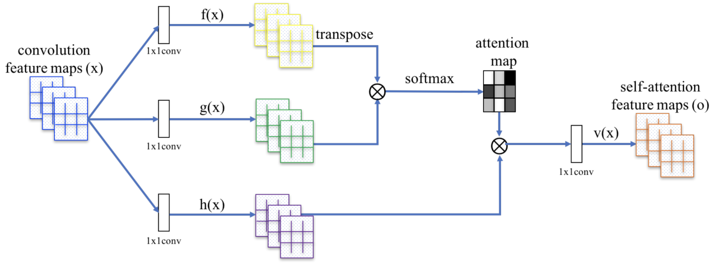


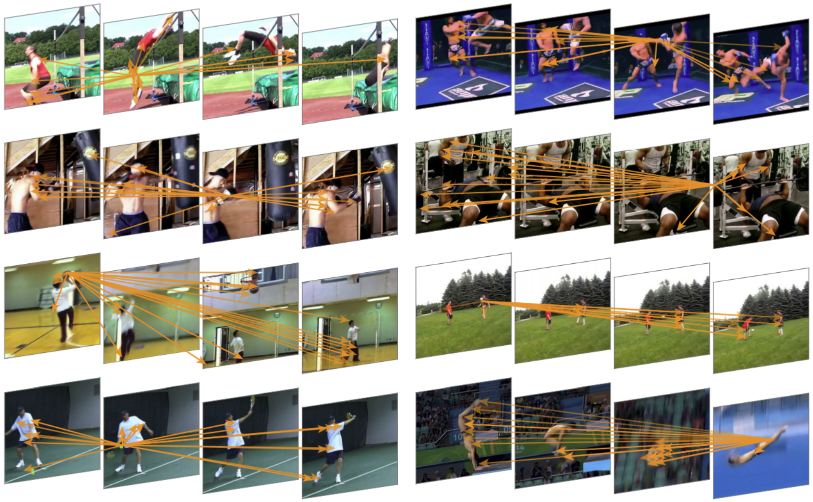
Figure: Here $h(x)$ denotes the value features, and $f(x)$ and $g(x)$ denote query and key features.

- Non-local block

$$\mathbf{z}_i = W_z \mathbf{y}_i + \mathbf{x}_i$$

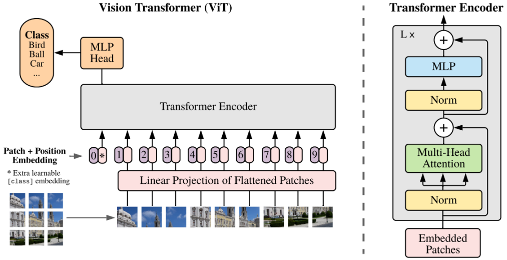
"$+\mathbf{x}_i$" denotes a residual connection

Visualization of the non-local operations



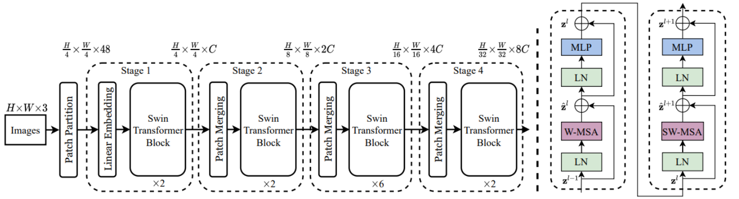
Vision Transformer

- There are also emerging visual neural networks that are purely built based on Transformer and the query-key-value attention mechanism
- (Dosovitskiy et al. 2020) proposed the Vision Transformer (ViT) without using any convolutional operations
- 1) ViT separates an input image into 16×16 image patches, extracts each of their features, and then feeds them into an multi-layer Transformer. 2) The input of the first position of the Transformer encoder has a `cls-token` vector as input. 3) A classification MLP head is appended to the first position's output features to make the final prediction
- It can stack up to 50 attention layers (by now) and can achieve comparable performance on ImageNet classification with visual CNNs



Swin Transformer

- ViT produces feature maps of a single low resolution and have quadratic computation complexity to input image size
- Swin Transformer builds hierarchical feature maps by merging image patches in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window



(a) Architecture (b) Two Successive Swin Transformer Blocks

Swin Transformer

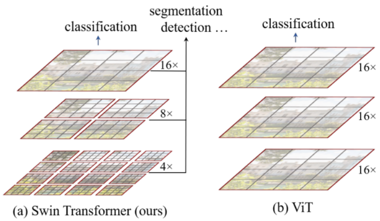


Figure: (Left) Swin Transformer generate multi-scale feature pyramid. (Right) ViT produce a single-scale feature map.

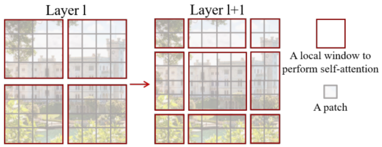
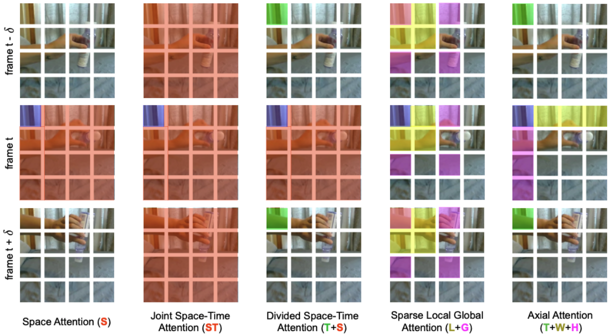


Figure: In layer l, a regular window partition is used and self-attention is computed within each window. In the next layer $l+1$, the window partition is shifted. The self-attention computation in the new windows crosses the boundaries of the previous windows in layer l.

TimeSformer for Video Understanding

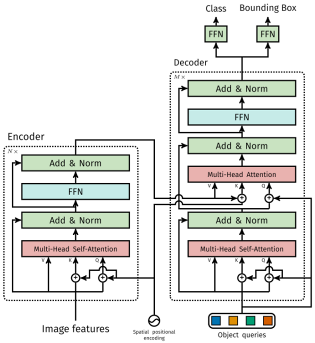
- Following ViT, each frame is decomposed into non-overlapping patches
- Each patch is first linearly embedded into a vector. The patch features are then input into the Transformer
- Multiple attention patterns are investigated

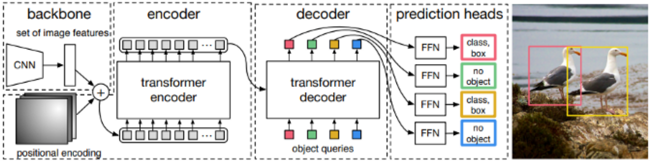

frame t - δ
frame t
frame t + δ
Space Attention (S)
Joint Space-Time Attention (ST)
Divided Space-Time Attention (T+S)
Sparse Local Global Attention (L+G)
Axial Attention (T+W+H)

- Although we only show you three frames, the attention extends to the entire clip (8, 16, or 96 frames investigated)
- 12 Transformer attention layers are stacked following ViT

DETR: End-to-End Object Detection with Transformers

- DETR adopts an encoder-decoder architecture and aims at abandoning the post-processing NMS
- The encoder further transforms the visual features from a visual backbone
- A series of learnable "object" queries to attend the visual feature map via the cross-attention mechanism





DETR: End-to-End Object Detection with Transformers

- Let y be the ground truth set of objects, and $\hat{y} = \{\hat{y}_i\}_{i=1}^N$ the set of N predictions
- As N is generally larger than $|y|$, we pad y with $\varnothing$ to have size N
- We find a bipartite matching between the two sets by searching for a permutation $\sigma \in \mathfrak{S}_N$ of N elements with the lowest cost $\hat{\sigma} = \underset{\sigma \in \widetilde{S}_N}{\arg\min} \sum_i^N \mathcal{L}_{\text{match}}\left(y_i, \hat{y}_{\sigma(i)}\right)$ where $\mathcal{L}_{\text{match}}\left(y_i, \hat{y}_{\sigma(i)}\right)$is a pair-wise matching cost between ground truth y_i and a prediction with index $\sigma(i)$
- Each ground truth $y_i = (c_i, b_i)$ has a class label c_i and box coordinates $b_i \in [0,1]^4$. For the prediction with index $\sigma(i)$, we define its class c_i probability as $\hat{p}_{\sigma(i)}(c_i)$
- The matching loss is defined as

$$\mathcal{L}_{\text{match}} = -\mathbf{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + \mathbf{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}\left(b_i, \hat{b}_{\sigma(i)}\right)$$

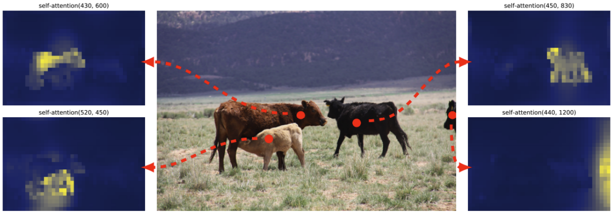
- The box loss is defined as

$$\lambda_{\text{iou}} \mathcal{L}_{\text{iou}}\left(b_i, \hat{b}_{\sigma(i)}\right) + \lambda_{\text{L1}} \left\| b_i - \hat{b}_{\sigma(i)} \right\|_1$$

  where $\mathcal{L}_{\text{iou}}$ is the generalized IoU loss
- This optimal assignment is computed efficiently with the Hungarian algorithm

Visualization of What did DETR Learn

- Visualization of self-attention in the Transformer encoder



- Visualization of box predictions from 20 prediction slots (object queries) in the DETR decoder

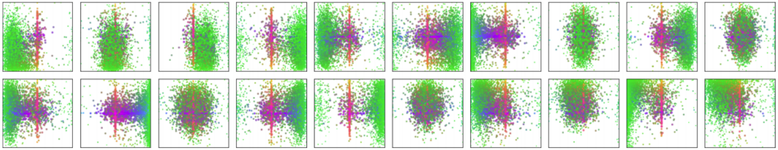


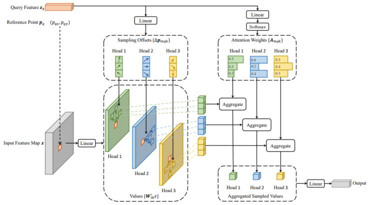
Figure: All box predictions in the COCO val set. Each box center is represented by one dot. Red, blue, and green dots represent large, medium, and small-scale object boxes.

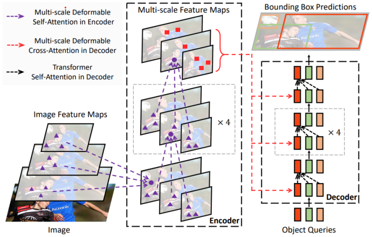
Deformable DETR

- DETR suffers from slow convergence because of the zero-initialized queries need to gradually learn which region it needs to be responsible for
- Deformable DETR introduces the deformable attention mechanism. For each object query $\mathbf{z}_q$ and its associated reference point $\mathbf{p}_q$, it learns to attend to a sparse sub-set of positions in the multiple scale feature maps
- Given a feature map $\mathbf{x} \in \mathbb{R}^{C\times H\times W}$, the deformable attention feature is calculated as

$$\text{DeformAttn}\left(\boldsymbol{z}_q, \boldsymbol{p}_q, \boldsymbol{x}\right) = \sum_{m=1}^{M} \boldsymbol{W}_m \left[\sum_{k=1}^{K} A_{mqk} \cdot \boldsymbol{W}'_m \boldsymbol{x}\left(\boldsymbol{p}_q + \Delta \boldsymbol{p}_{mqk}\right)\right],$$

W_m and W'_m represent multi-head linear projections. A_{mqk} are the normalized attention on the sparse deformed points $\{\mathbf{p}_q + \Delta \boldsymbol{p}_{mqk}\}$


Query Feature $\boldsymbol{z}_q$
Reference Point $\boldsymbol{p}_q$ (p_{qx}, p_{qy})
Linear
Linear
Softmax
Sampling Offsets $\{\Delta\boldsymbol{p}_{mqk}\}$
Attention Weights $\{\boldsymbol{A}_{mqk}\}$
Head 1
Head 2
Head 3
0.5
0.3
0.2
0.4
0.2
0.4
0.3
0.4
0.3
Input Feature Map $\boldsymbol{x}$
Linear
Values $\{\boldsymbol{W}'_m x\}$
Aggregate
Aggregate
Aggregate
Aggregated Sampled Values
Linear
Output


Multi-scale Deformable Self-Attention in Encoder
Multi-scale Deformable Cross-Attention in Decoder
Transformer Self-Attention in Decoder
Multi-scale Feature Maps
Bounding Box Predictions
Image Feature Maps
× 4
× 4
Encoder
Decoder
Image
Object Queries

MaskFormer: Per-Pixel Classification is Not All You Need for Semantic Segmentation

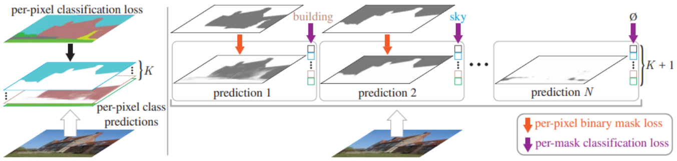

per-pixel classification loss
K
per-pixel class predictions
building
sky
Ø
prediction 1
prediction 2
prediction N
K + 1
per-pixel binary mask loss
per-mask classification loss

Figure: Per-pixel classification vs. per-mask classification.

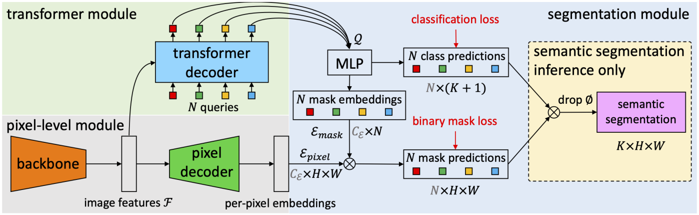


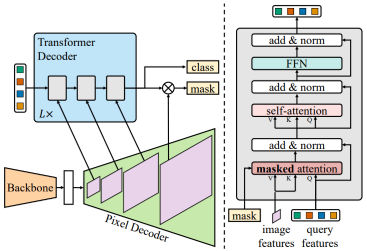
Figure: A set of N queries will be used to attend to the visual feature map. They generate N binary maps and N multi-class confidence score vectors.

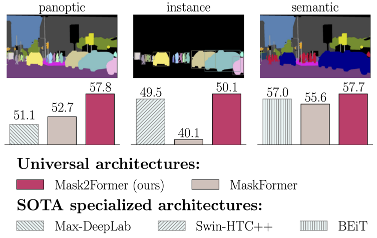
MaskFormer: Per-Pixel Classification is Not All You Need for Semantic Segmentation

- The set of N^{gt} ground truth segments $z^{\text{gt}} = \left\{\left(c_i^{\text{gt}}, m_i^{\text{gt}}\right) \mid c_i^{\text{gt}} \in \{1, \ldots, K\}, m_i^{\text{gt}} \in \{0,1\}^{H\times W}\right\}_{i=1}^{N^{\text{gt}}}$
- c_i^{gt} is the ground truth class of the ith segment
- m_i^{gt} is the ith segment's **binary** mask
- A set of N ($N > N^{gt}$) is used. The set of ground truth is pad with "no object" tokens $\emptyset$ to allow one-to-one matching
- If a query predicts "no object", its mask prediction is ignored
- The matching is directly conducted between the predicted segments and the GT segments

$$\mathcal{L}_{\text{mask-cls}}\left(z, z^{\text{gt}}\right) = \sum_{j=1}^{N}\left[-\log p_{\sigma(j)}\left(c_j^{\text{gt}}\right) + \mathbf{1}_{c_j^{\text{gt}} \neq \varnothing} \mathcal{L}_{\text{mask}}\left(m_{\sigma(j)}, m_j^{\text{gt}}\right)\right]$$

Mask2Former





- Mask2Former is currently the state-of-the-art architecture on various image segmentation tasks
- Its queries consecutively attend multi-scale visual feature pyramid from small to large scales

References

- Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." ICLR 2015.
- Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. "Show, attend and tell: Neural image caption generation with visual attention." ICML 2015.
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." NeurIPS 2017.
- Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization." NeurIPS 2016.
- `https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html`
- Wang, Xiaolong, Ross Girshick, Abhinav Gupta, and Kaiming He. "Non-local neural networks." CVPR 2018.
- Wu, Bichen, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Masayoshi Tomizuka, Kurt Keutzer, and Peter Vajda. "Visual transformers: Token-based image representation and processing for computer vision." arXiv:2006.03677 2020.
- Yuan, Li, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. "Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet." arXiv:2101.11986 2021.
- Gedas Bertasius, Heng Wang, Lorenzo Torresani, "Is Space-Time Attention All You Need for Video Understanding?", arXiv:2102.05095.

References

- Bowen Cheng, Alexander G. Schwing, Alexander Kirillov . "Per-Pixel Classification is Not All You Need for Semantic Segmentation ." NeurIPS 2021.
- Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, Rohit Girdhar. "Masked-attention Mask Transformer for Universal Image Segmentation." CVPR 2022.