Attention and Transformer Networks in Computer Vision

N. M. Militsis, Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr
Version 3.0

Artificial Intelligence & Information Analysis Lab

Transformers in Computer Vision

- **Motivation and related work**
- Transformers
- Vision Transformer (ViT)
- Swin Transformer
- DEtection Transformer (DETR)
- SEgmentation TRansformer (SETR)
- Segment Anything Model (SAM)
- DINO
- Video ViT (ViViT)

Motivation

Convolutional Neural Network limitations

- Convolutional Neural Networks (***CNNs***) at each layer employ moving convolution kernel (2D filter) windows.
- 2D convolution kernels are learned feature detectors.
- They are local operators.

- CNNs cannot benefit from distant image patch correlations.

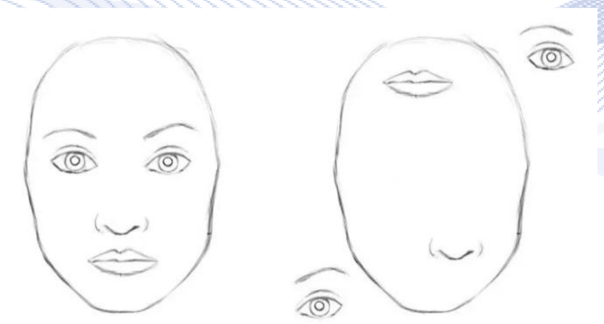
- CNN kernels are ***inefficient*** at modeling visual elements with ***varying spatial distributions.***

Motivation

CNN limitations

- CNN features do not consider important spatial hierarchies between objects.
- Only ***local interactions*** are considered in each convolutional layer.

- CNNs detect parts with no sense of the whole (***Picasso problem***).



Motivation

CNN limitations

- Most CNN architectures leverage ***pooling*** for increasing the ***receptive field*** of higher-level layers kernels, allowing them to capture higher-level features on large image regions.

- ***Pooling*** may lead to severe ***information loss***.

- Only ***local interactions*** are considered in each convolutional layer.

Motivation

CNN inductive bias

- ***CNNs*** rely on the assumptions of ***locality*** and ***stationarity*** governing the $2D$ image signal.

- ***Locality*** refers to the fact that neighboring pixel intensity values tend to be more correlated than those of distant ones.
- ***Stationarity*** denotes that image statistics do not vary spatially (e.g., across image regions).

Motivation

Locally adaptive kernels

- Each CNN layer is ***static***. That is, the same fixed convolution kernel (CNN parameters) slides across different image regions.
- Data-dependent ***locally-adaptive kernels*** can facilitate the accurate cope with varying spatial image distributions.

- Such kernels (***Bilateral filter*** [ELA2002], ***Non-local means*** [BUA2005], ***LARK*** [TAK2007]) have widely been applied on image denoising.

Motivation

Locally adaptive kernels

- ***Bilateral filter***:

$$k_{ij} = \exp\left(\frac{-\left\|x_i - x_j\right\|^2}{b_x^2}\right)\exp\left(\frac{-\left\|\mathbf{p}_i - \mathbf{p}_j\right\|^2}{b_p^2}\right).$$

- k_{ij}: similarity between image pixels i and j.
- x_i: intensity of image pixel i .
- $\mathbf{p}_i \in \mathbb{R}^2$: image pixel i position.

Motivation

Locally adaptive kernels

- ***Non-local means*** is a generalization of bilateral filter.
- It operates at image patch than at image pixel level:

$$k_{ij} = \exp\left(\frac{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{b_x^2}\right)\exp\left(\frac{-\left\|\mathbf{p}_i - \mathbf{p}_j\right\|^2}{b_p^2}\right).$$

  - k_{ij}: similarity between two image patches i and j,
  - $\mathbf{x}_i \in \mathbb{R}^d$: vector of patch i pixel intensities.
  - $\mathbf{p}_i \in \mathbb{R}^2$: patch position on the image.

Motivation

Locally adaptive kernels

- ***Locally adaptive regression kernel*** **(*LARK*)** captures the local data structure, by estimating the local geodesic distance between nearby patches.

:

$$k_{ij} = \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{C}_{ij} (\mathbf{x}_i - \mathbf{x}_j)\right).$$

- k_{ij}: similarity between two image patches i and j,
- $\mathbf{x}_i \in \mathbb{R}^d$: vector of patch i pixel intensities.
- $\mathbf{C}_{ij} \in \mathbb{R}^{d\times d}$: covariance matrix of the gradient of the intensity values approximating the local geodetic distance.

Motivation

Locally adaptive kernels

- ***Scaled dot-product attention*** [VAS2017]:

$$a_{ij} = Softmax\left(\frac{\mathbf{q}_i^T \mathbf{k}_j}{\sqrt{d_k}}\right),$$

$$\mathbf{q}_i = \mathbf{W}_Q(\mathbf{x}_i + \mathbf{p}_i) \in \mathbb{R}^{d_k},$$

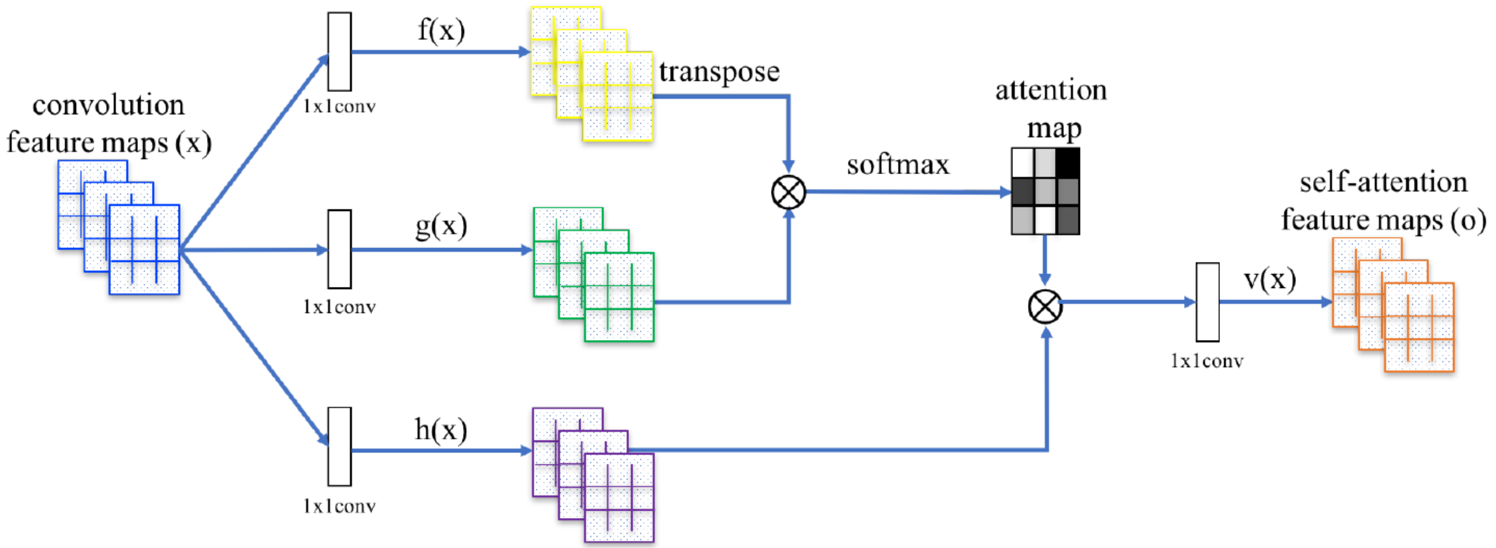
$$\mathbf{k}_j = \mathbf{W}_K(\mathbf{x}_j + \mathbf{p}_j) \in \mathbb{R}^{d_k}.$$

- a_{ij} : similarity between the image patches i and j represented by feature vectors $\mathbf{x}_i \in \mathbb{R}^{d_m}$ and $\mathbf{x}_j \in \mathbb{R}^{d_m}$,
- $\mathbf{p}_i, \mathbf{p}_j \in \mathbb{R}^{d_m}$: patch positional encodings.
- $\mathbf{W}_Q \in \mathbb{R}^{d_k \times d_m}, \mathbf{W}_K \in \mathbb{R}^{d_k \times d_m}$ learnable parameter matrices.

Related work

Augmenting CNNs with attention

- Scaled dot-product attention module of ***Self-Attention Generative Adversarial Networks*** (***SAGANs***)



- Operator $\otimes$ denotes matrix multiplication.
- $\mathbf{f}(\mathbf{x}), \mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x})$ can be considered as queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$, respectively.
- Softmax is performed row-wise [ZHA2018].

Related work

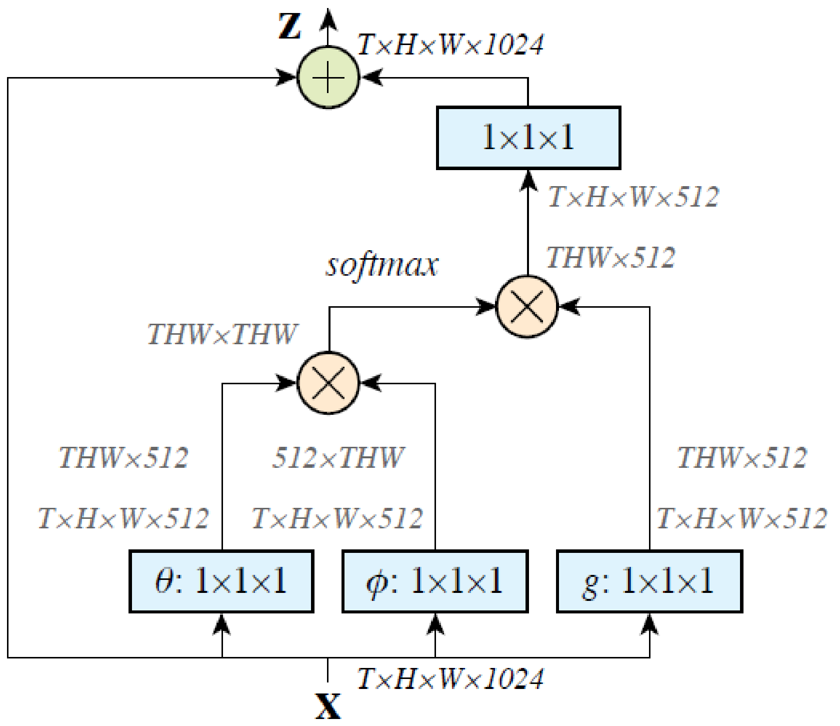
Augmenting CNNs with attention

Non-local neural networks

Output spatiotemporal feature map $\mathbf{Z} \in \mathbb{R}^{THW \times C'}$ is generated from $\mathbf{X} \in \mathbb{R}^{THW \times C}$:

$$\mathbf{Z} = Softmax(\mathbf{X}\mathbf{W}_\theta \mathbf{W}_\varphi^T \mathbf{X}^T)\mathbf{X}\mathbf{W}_g.$$

- $\mathbf{W}_\theta, \mathbf{W}_\varphi, \mathbf{W}_g \in \mathbb{R}^{C \times C'}$: learnable parameter matrices.
- T and H, W are the temporal and spatial dimensions.



Spatiotemporal non-local block.
⊗ and ⊕ denote matrix multiplication and addition respectively [WAN2018].

Related work

Augmenting CNNs with attention

- ***Non-local neural networks***

Correlations captured by non-local blocks [WAN2018].

Artificial Intelligence & Information Analysis Lab

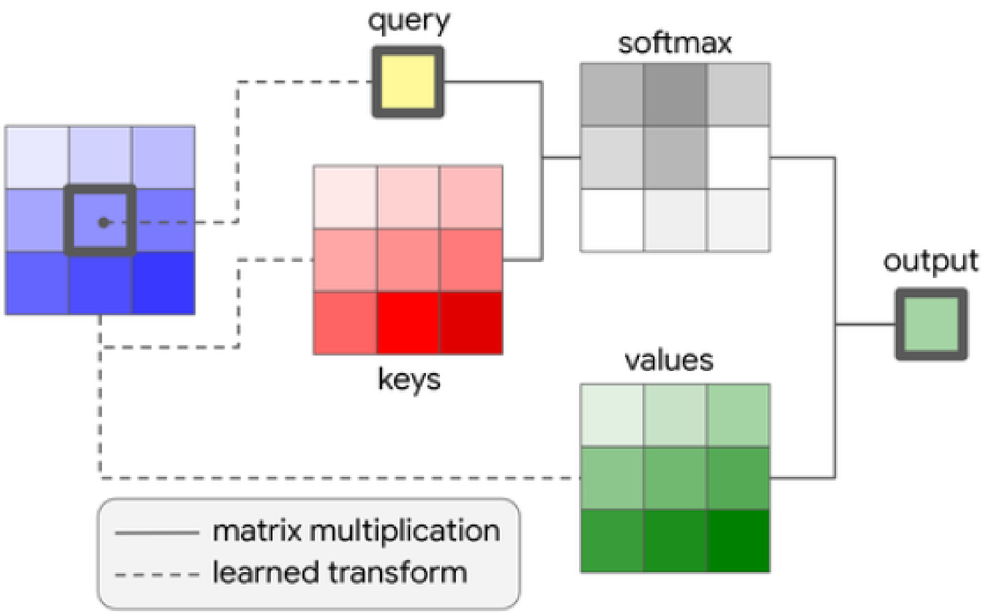
Related work

Replacing convolution with (local) attention

- ***Stand-Alone Self-Attention in Vision Models*** (***SASA***)

- Attention can used as stand-alone primitive for vision models instead of serving just as augmentation on top of convolutions.

- Attention kernel slides accross different image regions.



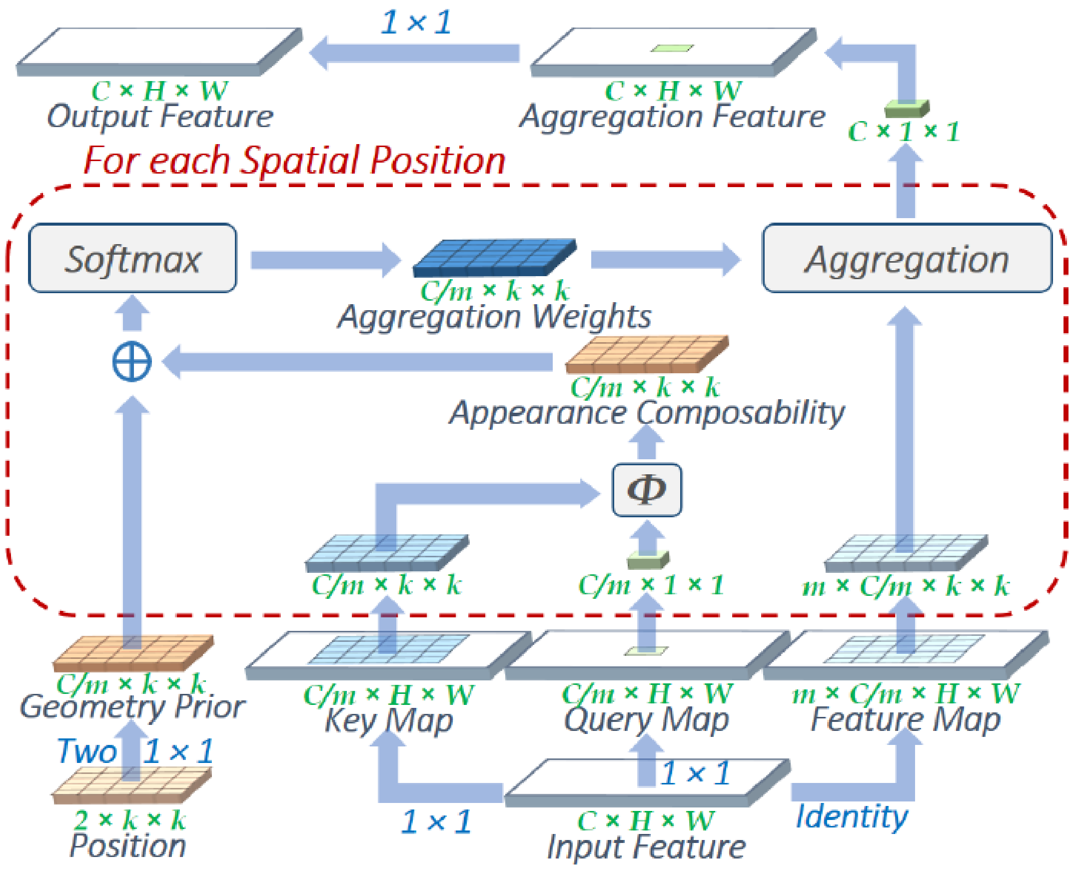
Spatial local attention layer [RAM2019].

Related work

Augmenting CNNs with attention

Local Relation Networks

A local relation layer ***adaptively determines the aggregation weights*** based on the compositional relationship of local pixel pairs.



The local relation layer [HU2019].

Related work

Augmenting CNNs with attention

- ***Local Relation Networks***

  Local relation layer:

$$\omega = \text{Softmax}\left(\Phi\left(f_{\theta_q}(\mathbf{x}_i), f_{\theta_k}(\mathbf{x}_j)\right) + f_{\theta_k}(\mathbf{p}_i - \mathbf{p}_j)\right).$$

- $\Phi\left(f_{\theta_q}(\mathbf{x}_i), f_{\theta_k}(\mathbf{x}_j)\right)$ is a measure of ***similarity*** between the target pixel $\mathbf{x}_i$ and a pixel $\mathbf{x}_j$ within its position scope.

- f_{θ_q} and f_{θ_k}: pixel transformation functions.

Related work

Augmenting CNNs with attention

- ***Local Relation Networks***
  - $\mathbf{p}_i \in \mathbb{R}^2$: position of pixel i.
  - $f_{\theta_k}(\mathbf{p}_i - \mathbf{p}_j)$: it defines the similarity of a pixel pair (i, j) based on a geometric prior.

  - The geometric term adopts the relative position as input and is translationally invariant.
  - It is encoded by a small network consisting of two channel transformation layers, with a ReLU activation in between.

Transformers in Computer Vision

- Motivation and related work
- **Transformers**
- Vision Transformer (ViT)
- Swin Transformer
- DEtection Transformer (DETR)
- SEgmentation TRansformer (SETR)
- Segment Anything Model (SAM)
- DINO
- Video ViT (ViViT)

Transformer architecture

- An image $\mathbf{X} \in \mathbb{R}^{HW \times C}$ is split into fixed-size patches $\mathbf{x} \in \mathbb{R}^{N^2 C}$.

- Each patch gets linearly embedded as following:

$$\mathbf{x}_i' = \mathbf{W}_e \mathbf{x}_i, \qquad i = 1, \ldots, HW$$

- $\mathbf{W}_e \in \mathbb{R}^{d_m \times N^2 C}$:learnable parameter matrix.

Image patches.

Transformer architecture

- Positional information is provided through ***additive learnable*** positional encodings of the same dimension d_m as the input vectors $\mathbf{z}_i$:

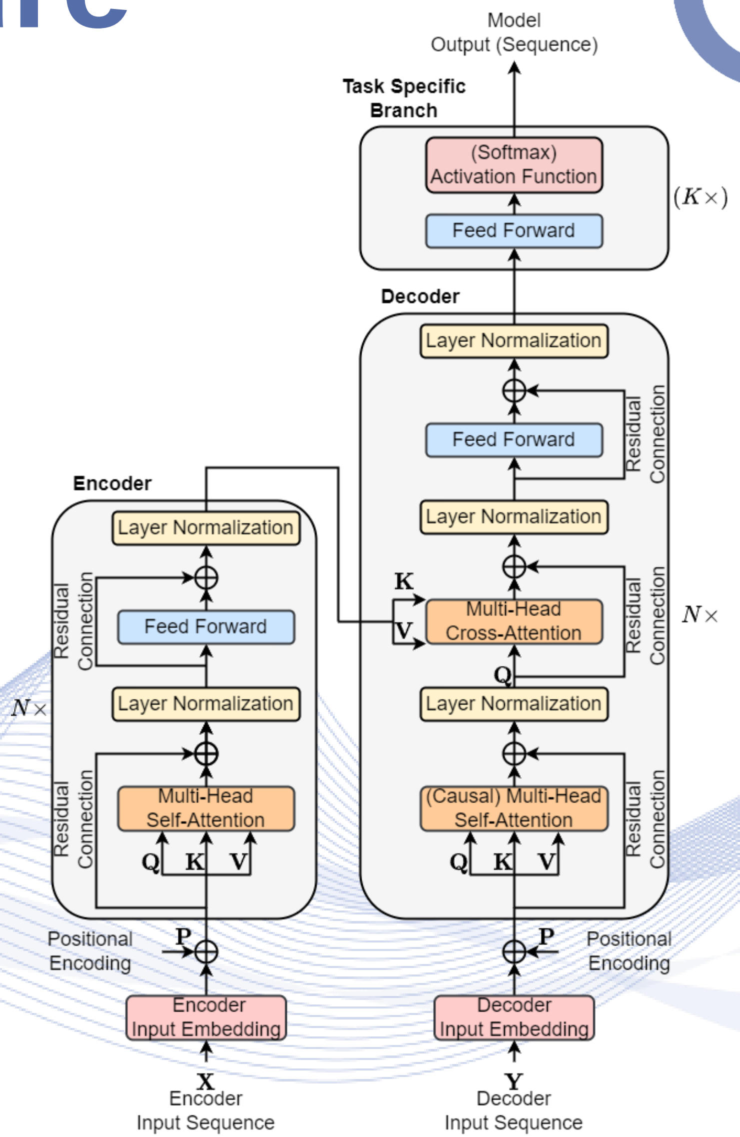
$$\mathbf{z}_i = \mathbf{x}'_i + \mathbf{p}_i.$$

- At initialization, the positional encodings carry no information about the $2D$ positions of the patches.
- All spatial relations between the patches are learned during training.
- The matrix of input embeddings $\mathbf{Z} \in \mathbb{R}^{L \times d_m}, L = HW/N^2$ is imported in the Transformer encoder.

Transformer architecture

Transformer was originally designed for neural sequence transduction.

It has an ***encoder-decoder*** structure followed by one or more ***task specific branches***.



General Transformer architecture.

Transformer architecture

Encoder

- The encoder consists of a stack of N identical blocks.

- Each block has two sub-layers:
  - A ***multi-head self-attention module.***
  - A ***position-wise*** fully connected ***feed-forward*** network.

- ***Residual connection*** [HE2016] is employed around each sub-layer followed by ***layer normalization*** [BA2016].

Transformer architecture

Decoder

- The decoder also consists of a stack of N identical blocks.
- Each block has three sub-layers.
  - A (***causal***) ***multi-head self-attention module***. Optionally a mask is employed to prevent current data point from attending subsequent ones.
  - A ***multi-head cross-attention module*** between encoder and decoder sequences.
  - A ***position-wise*** fully connected ***feed-forward*** network.
- Again, residual connection and layer normalization are applied around each sub-layer.

Transformer architecture

Scaled dot-product attention

Three new matrices $\mathbf{Q} \in \mathbb{R}^{L \times d_k}$ (***queries***), $\mathbf{K} \in \mathbb{R}^{L' \times d_k}$(***keys***), $\mathbf{V} \in \mathbb{R}^{L' \times d_v}$ (***values***) are generated:

$$\mathbf{Q} = \mathbf{Z}\mathbf{W}_Q + \mathbf{1}_{L \times 1}\mathbf{b}_Q, \qquad \mathbf{W}_Q \in \mathbb{R}^{d_m \times d_k}, \mathbf{b}_Q \in \mathbb{R}^{d_k},$$
$$\mathbf{K} = \mathbf{Z}'\mathbf{W}_K + \mathbf{1}_{L' \times 1}\mathbf{b}_K, \qquad \mathbf{W}_K \in \mathbb{R}^{d_m \times d_k}, \mathbf{b}_K \in \mathbb{R}^{d_k},$$
$$\mathbf{V} = \mathbf{Z}'\mathbf{W}_V + \mathbf{1}_{L' \times 1}\mathbf{b}_V, \qquad \mathbf{W}_V \in \mathbb{R}^{d_m \times d_v}, \mathbf{b}_V \in \mathbb{R}^{d_v},$$

by linearly transforming two matrices $\mathbf{Z} \in \mathbb{R}^{L \times d}$ and $\mathbf{Z}' \in \mathbb{R}^{L' \times d}$, where $L \neq L'$,

- Learnable parameters: $\mathbf{W}_Q, \mathbf{W}_K\ \mathbf{W}_V, \mathbf{b}_Q, \mathbf{b}_K, \mathbf{b}_V$.
- In the ***original model***, it is arbitrarily chosen that $d_k = d_v$.

Artificial Intelligence & Information Analysis Lab

Transformer architecture

Scaled dot-product attention

Using the terminology in [GRAV2014], attention is an averaging of ***values***, associated to ***keys*** matching to specific ***queries***.

In ***cross-attention*** each data point of sequence $\mathbf{X}'_e$ attends to all data points of sequence $\mathbf{Z}'$ in order to compute a new representation of sequence $\mathbf{Z}$:

$$\mathbf{Y} = Softmax\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}.$$

The ***row-wise Softmax operator*** renders a probability distribution, representing the normalized correlation scores of each query to all the keys.

Artificial Intelligence & Information Analysis Lab

Transformer architecture

Transformers

- They can have multiple heads, facilitating both parallelization and attention to different regions.
- Transformer encoder and decoder can have multiple layers.

Transformers in Computer Vision

- Motivation and related work
- Transformers
- **Vision Transformer (ViT)**
- Swin Transformer
- DEtection Transformer (DETR)
- SEgmentation TRansformer (SETR)
- Segment Anything Model (SAM)
- DINO
- Video ViT (ViViT)

Vision Transformer (ViT)

In ViT [DOS2021] an image is split into non overlapping ***patches***. The sequence of ***linear embeddings*** of these patches is provided as input to a ***Transformer encoder***.

The model is trained on ***image classification*** task in ***supervised*** fashion.

Vision Transformer (ViT)

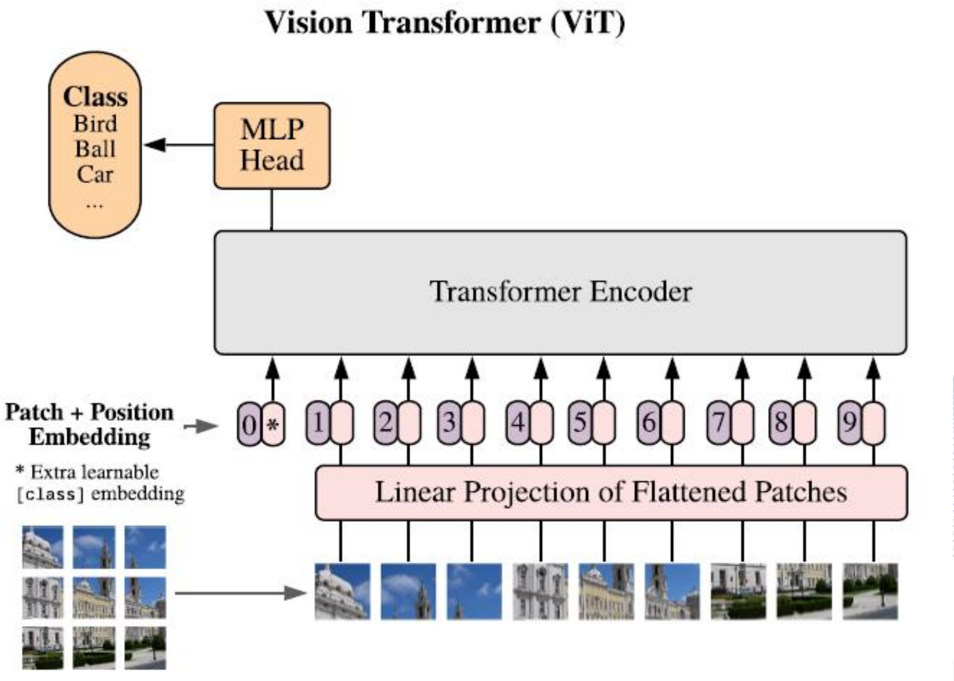
ViT architecture

- An image $\mathbf{X} \in \mathbb{R}^{HW \times C}$ is split into fixed-size patches $\mathbf{x} \in \mathbb{R}^{N^2C}$.

- Each patch gets linearly embedded as following:

$$\mathbf{x}'_i = \mathbf{W}_e \mathbf{x}_i, \qquad i = 1, \dots, HW$$

- $\mathbf{W}_e \in \mathbb{R}^{d_m \times N^2 C}$:learnable parameter matrix.



ViT overview [DOS2021].

Vision Transformer (ViT)

ViT architecture

- Positional information is provided through ***additive learnable*** positional encodings of the same dimension d_m as the input vectors $\mathbf{z}_i$:

$$\mathbf{z}_i = \mathbf{x}_i' + \mathbf{p}_i.$$

- At initialization, the positional encodings carry no information about the $2D$ positions of the patches.
- All spatial relations between the patches are learned during training.

Vision Transformer (ViT)

ViT architecture

- The matrix of input embeddings $\mathbf{Z} \in \mathbb{R}^{L \times d_m}, L = HW/N^2$ is imported in the Transformer encoder.
- Similar to BERT [class] token, an ***extra learnable embedding vector*** $\mathbf{z}_s \in \mathbb{R}^{d_m}$ is appended at the start of $\mathbf{Z}$ leading to $\mathbf{Z}' \in \mathbb{R}^{(L+1) \times d_m}$
- The state of $\mathbf{z}_s$ at the encoders output serves as the ***final image representation*** $\mathbf{y}_s \in \mathbb{R}^{d_m}$.

Vision Transformer (ViT)

ViT architecture

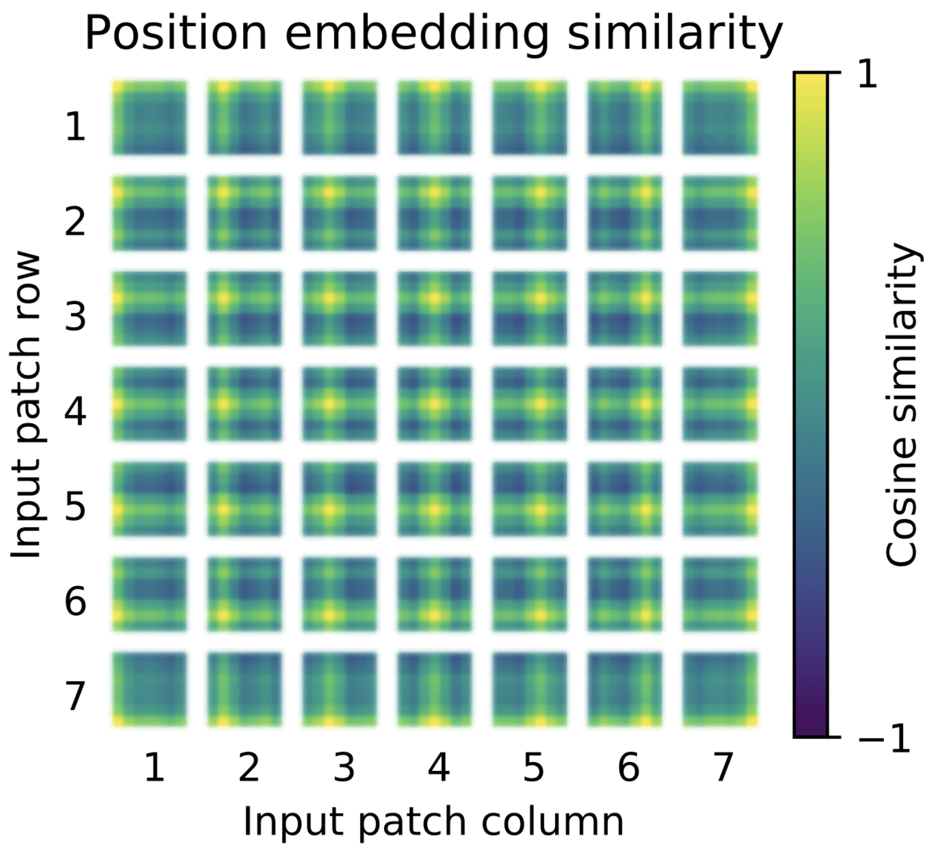
- The ***final image representation*** $\mathbf{y}_s \in \mathbb{R}^{d_m}$ is fed to a single linear layer parameterized by $\mathbf{W} \in \mathbb{R}^{d_m \times K}$ (for K classes) followed by a *Softmax* activation function to produce the final class probability distribution.

- Typically, ViT is pre-trained on large datasets and fine-tuned on downstream tasks.

Vision Transformer (ViT)

ViT architecture

- Neighboring image patches tend to have similar position embeddings.
- Patches in the same row/column have similar embeddings.
- Positional encoding pairwise similarities exhibit row-column structure.

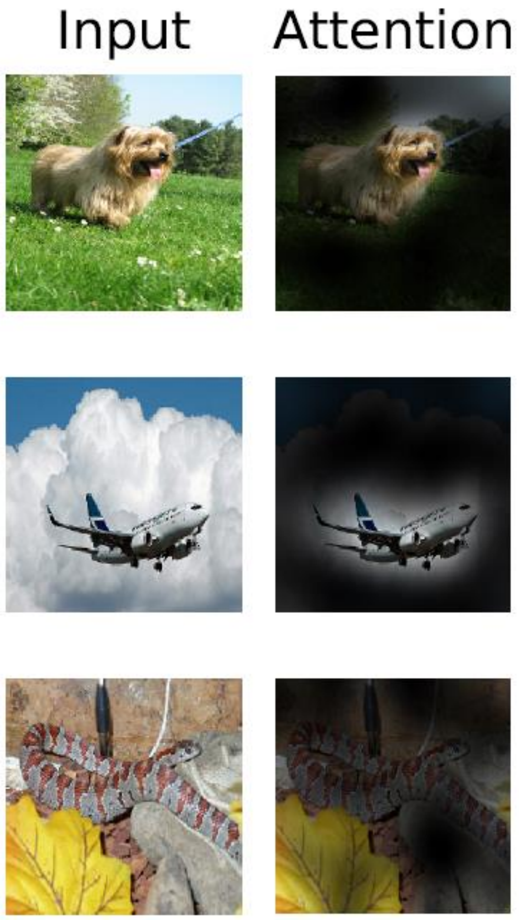


Positional encodings pairwise similarities [DOS2021].

Vision Transformer (ViT)

Remarks

- For extracting the distribution of class probabilities, the output token $\mathbf{y}_s \in \mathbb{R}^{d_m}$ attends to semantically relevant image regions.

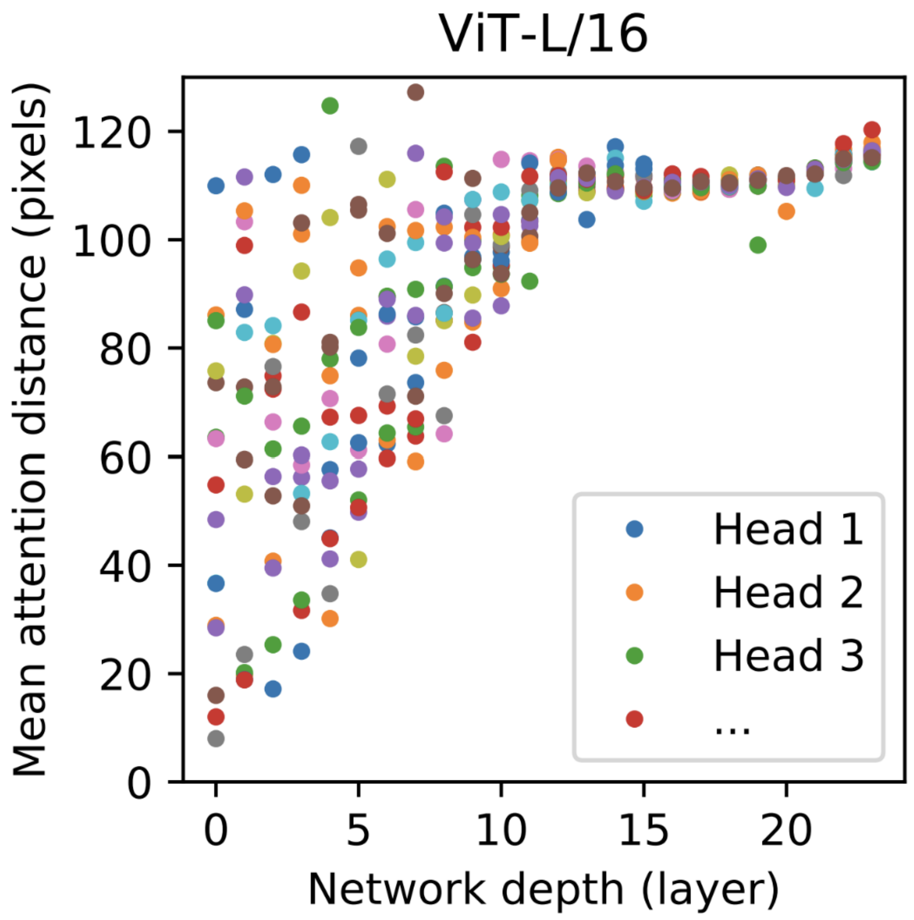


Attention of the output token to the input space [DOS2021].

Vision Transformer (ViT)

Remarks

- The ***attention distance*** increases with network depth.
- It can be considered bo be analogous to the receptive field in CNNs.
- Globally, the ViT model attends to image regions that are semantically similar for classification.



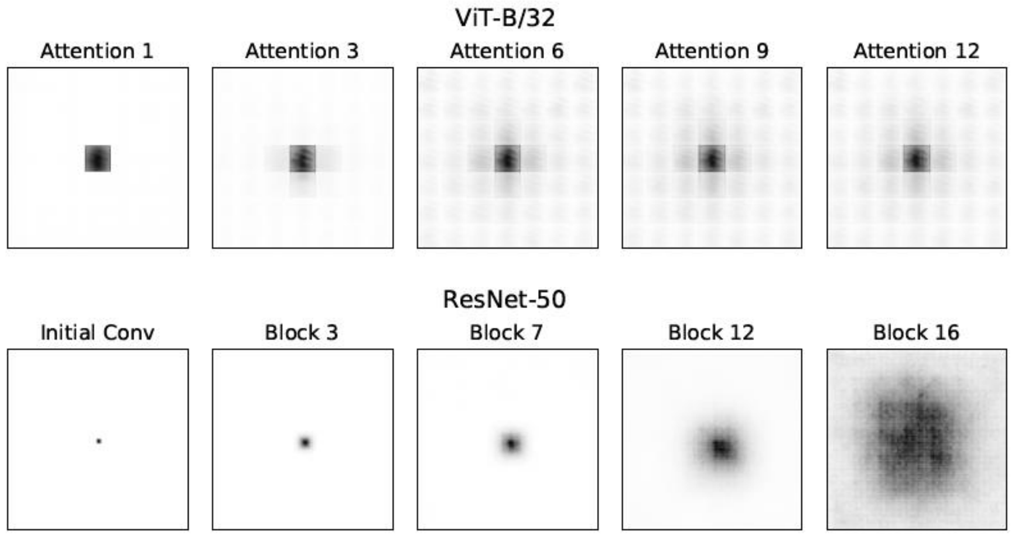
Distance of attended area by head and layer [DOS2021].

Vision Transformer (ViT)

Remarks

- ViT attention distances shift from local to global when moving deeper in the network.

- ResNet effective receptive fields are highly local and grow gradually, when moving deeper in the CNN network.

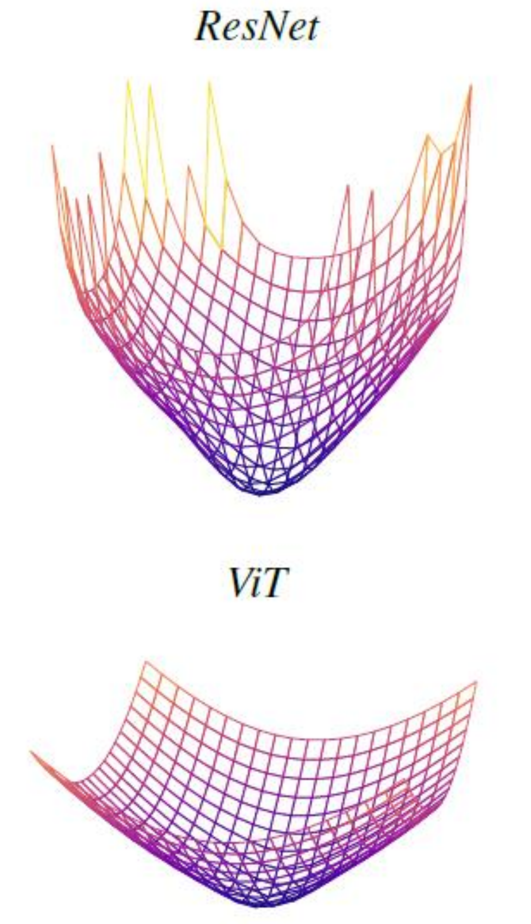


ViT attention distance vs ResNet receptive field [RAG2022].

Vision Transformer (ViT)

Remarks

- When trained on large datasets, multi-headed ViT attention ***flattens the loss function***, leading to better performance and generalization.



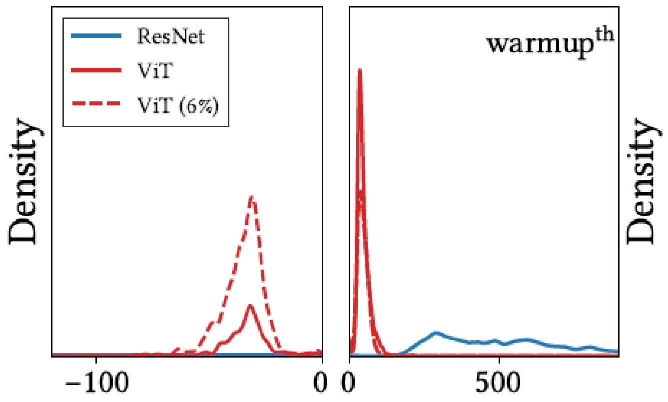
Negative log likelihood loss + ℓ_2 regularization [PAR2022].

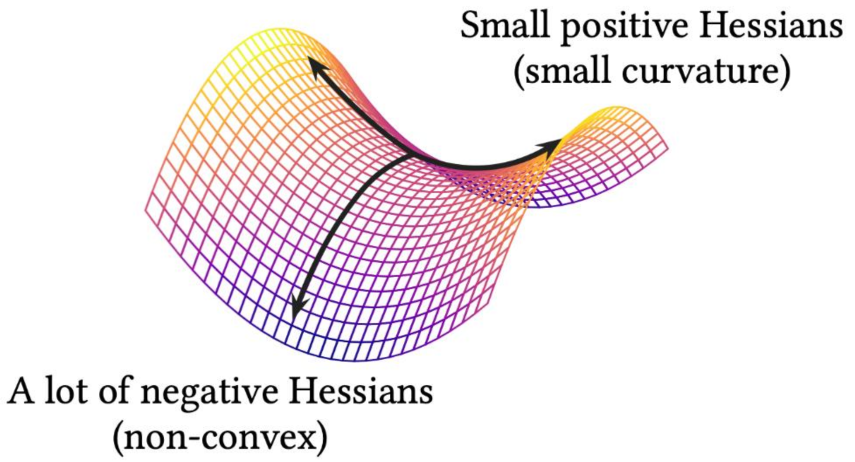
Artificial Intelligence & Information Analysis Lab

Vision Transformer (ViT)

Remarks

- When trained on small datasets, multi-headed ViT attention allows negative Hessian eigenvalues, leading to ***non-convex loss function forms***.



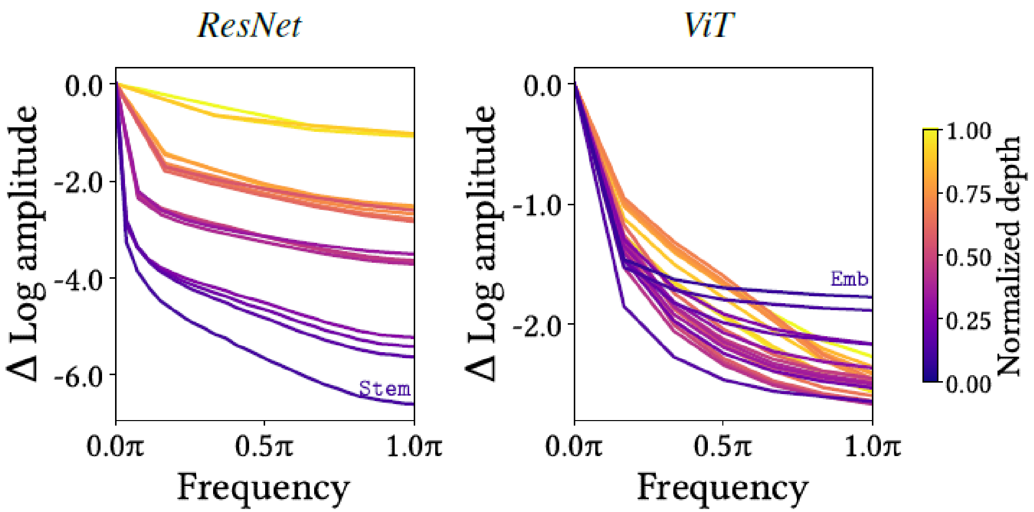


Negative Hessian eigenvalues and amplitude of positive Hessian eigenvalues for ImageNET. The dotted line corresponds to the 6% of the dataset [PAR2022].

Vision Transformer (ViT)

Remarks

- Fourier analysis of feature maps can show that ***Multi-headed Self-Attention*** (***MSA***) modules dampen high signal frequencies, while convolutional kernels tend to amplify them.
- Thus, MSA layers are homogeneous region-biased, while convolutional ones are texture-biased.



Relative log amplitudes of Fourier transformed feature map. Δ log amplitude of high-frequency signals is the difference between the log amplitude at normalized frequency 0.0π and at 1.0π [PAR2022].

Vision Transformer (ViT)

Hybrid ViT architecture

- The ViT input sequence can be formed from CNN feature maps of image patches.

- The input embedding projection using $\mathbf{W}_e \in \mathbb{R}^{d_m \times N^2 C}$ is applied to patches extracted from a CNN feature map.

- As a special case, the patches can have spatial size 1×1, which means that the input sequence is obtained by simply flattening the spatial dimensions of the feature map.

Transformers in Computer Vision

- Motivation and related work
- Transformers
- Vision Transformer (ViT)
- **Swin Transformer**
- DEtection Transformer (DETR)
- SEgmentation TRansformer (SETR)
- Segment Anything Model (SAM)
- DINO
- Video ViT (ViViT)

Artificial Intelligence & Information Analysis Lab

Swin Transformer

- Unlike the word tokens in Natural Language Processing (NLP), ***visual elements can vary substantially in scale***.

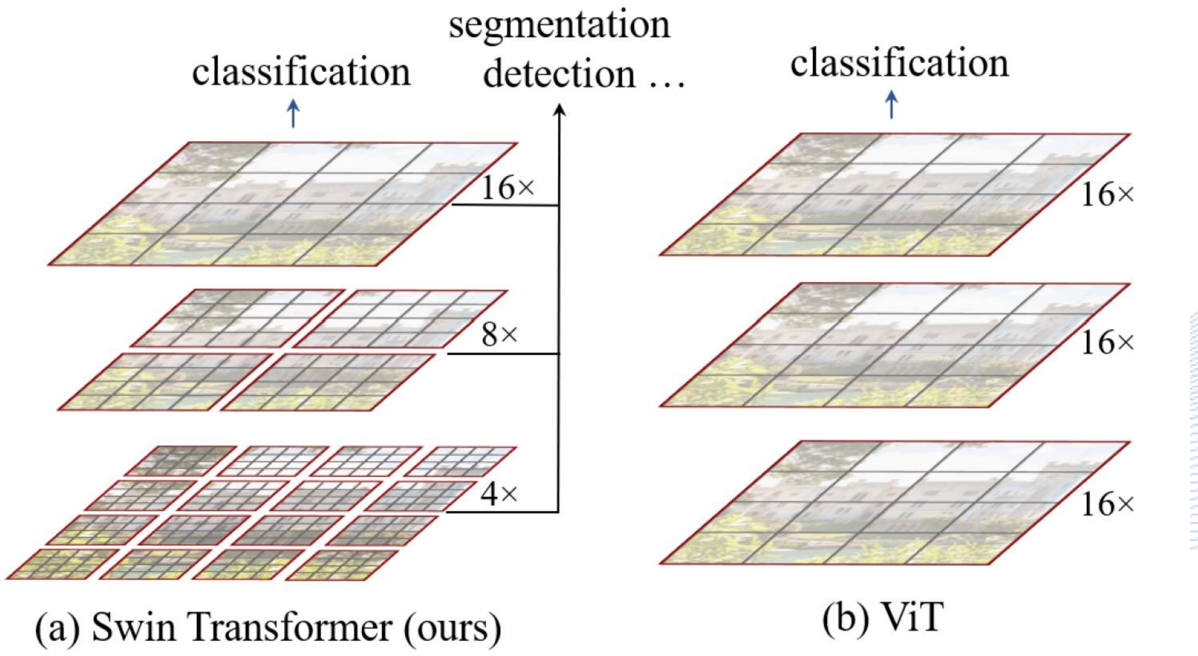
- This issue is important in vision tasks, such as object detection.

- Visual input embeddings of ***fixed scale*** are ***unsuitable*** for most vision applications.

Swin Transformer

Swin Transformer [LIU2021] constructs ***hierarchical*** feature maps by ***merging*** image patches in ***deeper layers***.

- ***Self-attention*** is computed locally within non-overlapping ***local*** windows (in red) consisting of $M \times M$ patches.
- ***Linear computation complexity***.



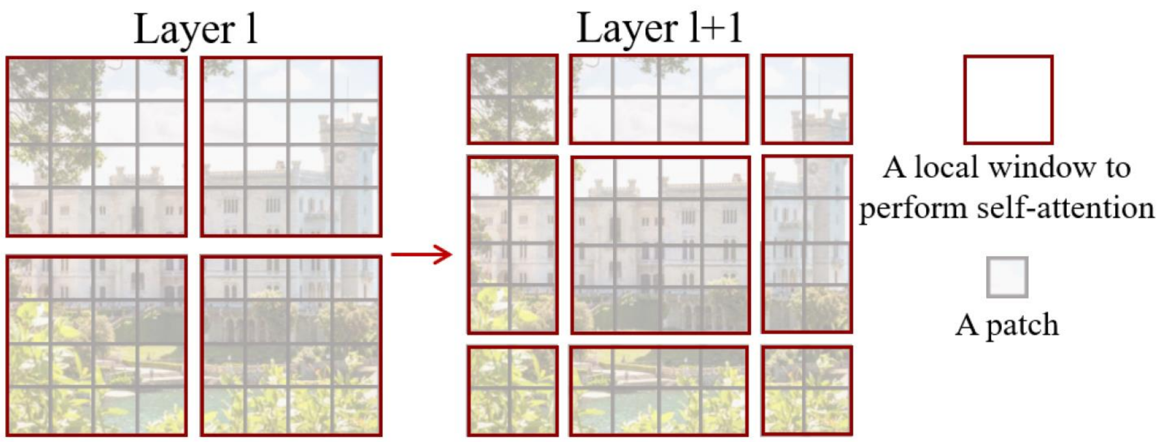
Hierarchical and fixed resolution feature maps of patches (in grey) [LIU2021].

Swin Transformer

- A ***hierarchical*** image representation is constructed by starting from small-sized patches and ***gradually merging neighboring*** ones in ***deeper layers***.
- The Swin Transformer employs these hierarchical feature maps to leverage advanced techniques for ***dense prediction***, such as ***feature pyramid networks*** (FPN) [LIN2017] or U-Net [RON2015].

Swin Transformer

- To introduce ***cross-window*** connections while maintaining computational efficiency, a ***cyclically shifted window*** (***Swin***) approach is utilized between layers.

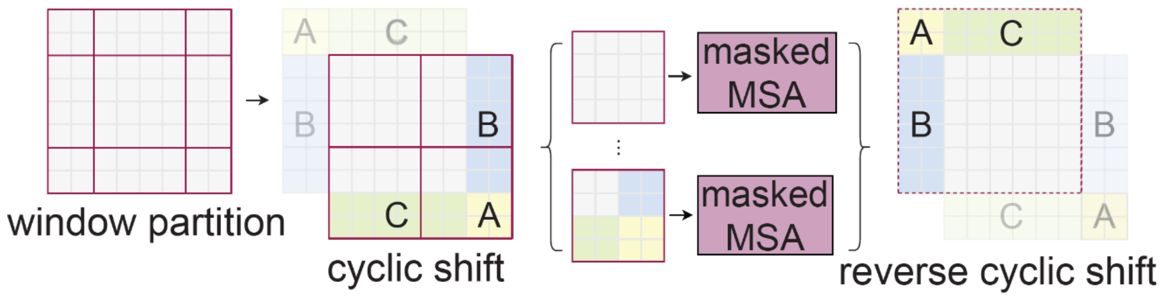


Shifted window approach [LIU2021] .

- Computation of self-attention in layer $l+1$ crosses the boundaries of windows in layer l.

Swin Transformer

- The window is ***shifted cyclically***, as it is typically done in cyclic convolutions.
- Assuming that the feature map is repeated periodically in both spatial dimensions, the window is shifted ***from top-left to bottom-right***.
- ***Masks*** are employed to prevent the computation of self-attention between patches that are ***not adjacent*** in the original image.



Cyclic shift of windows [LIU2021] .

Swin Transformer

Positional information in injected to the model, by including a ***learnable relative position*** bias $\mathbf{B} \in \mathbb{R}^{M^2 \times M^2}$ to each head of local self attention within a ***window*** consisting of M^2 patches:

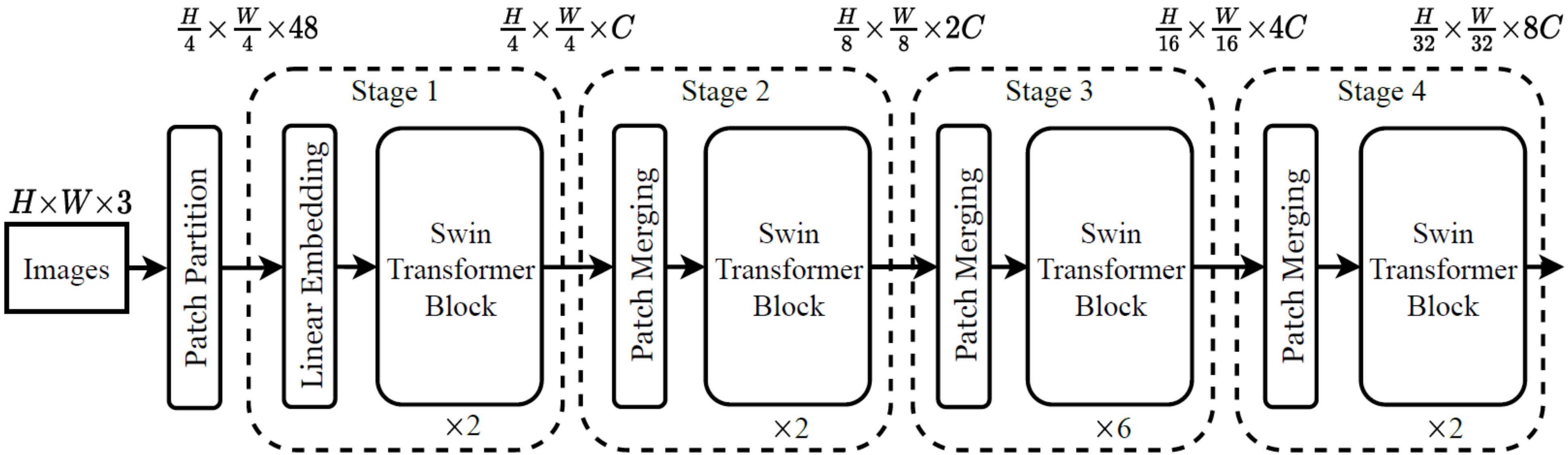
$$\mathbf{X}_w^h = Softmax\left(\frac{\mathbf{Q}_w^h (\mathbf{K}_w^h)^T}{\sqrt{D_k}} + \mathbf{B}^h\right)\mathbf{V}_w^h.$$

- $\mathbf{Q}_w^h, \mathbf{K}_w^h \in \mathbb{R}^{M^2 \times d_k}, \mathbf{V}_w^h \in \mathbb{R}^{M^2 \times d_v}$: queries, keys and values corresponding to window w and head h.

Swin Transformer

Swin Transformer architecture

- Shifted window configuration is utilized between consecutive blocks in each stage [LIU2021].



Artificial Intelligence & Information Analysis Lab

Transformers in Computer Vision

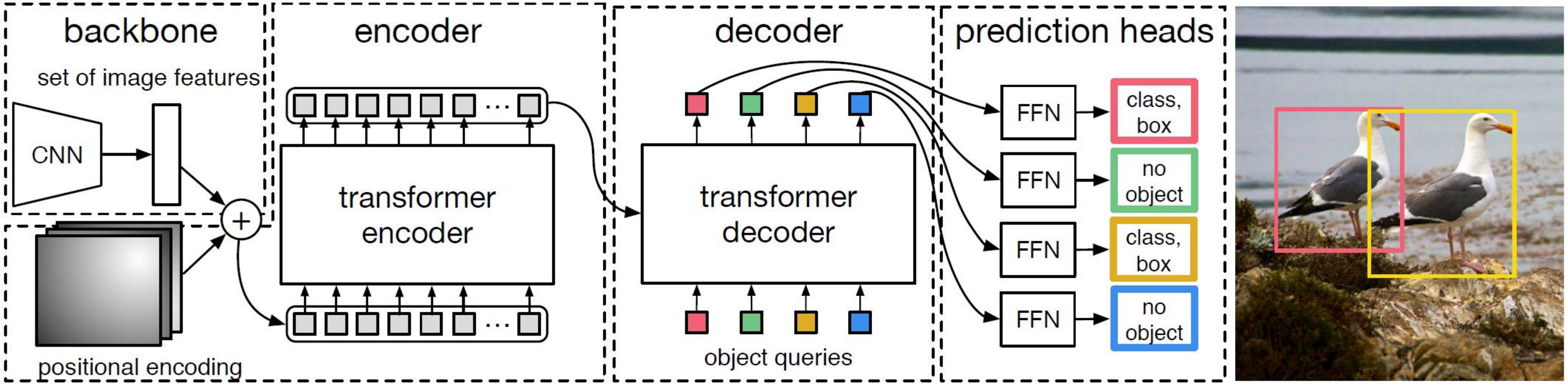
- Motivation and related work
- Transformers
- Vision Transformer (ViT)
- Swin Transformer
- **DEtection Transformer (DETR)**
- SEgmentation TRansformer (SETR)
- Segment Anything Model (SAM)
- DINO
- Video ViT (ViViT)

DEtection TRanformer (DETR)

- DETR [CAR2020] object detection employs neither hand-crafted components, such as ***Non-Maximum Suppression*** (NMS) nor ***anchor boxes*** that encode prior knowledge about the task.

- DETR employs the conjunction of a ***bipartite matching loss*** and a ***parallel decoding*** transformer (non auto-regressive).

DEtection TRanformer (DETR)



DETR architecture [CAR2020].

DEtection TRanformer (DETR)

DETR architecture

- Given an input image $\mathbf{X} \in \mathbb{R}^{C_0 \times H_0 W_0}$ DETR uses a conventional ***CNN backbone*** to learn a lower resolution feature map $\mathbf{X}' \in \mathbb{R}^{C \times HW}$.
- Typical values:

$$C = 2048, \quad H = H_0/32, \; W = W_0/32.$$

- The channel dimension C is reduced through an 1×1 convolution creating a new feature map $\mathbf{X}'' \in \mathbb{R}^{d_m \times HW}$.

DEtection TRanformer (DETR)

DETR architecture

- Positional information is provided through ***additive fixed*** vectors of the same dimension d_m as the input embeddings:

$$\mathbf{z}_i = \mathbf{x}_i'' + \mathbf{p}_i, \quad i = 1, \dots, HW$$

- The elements of $\mathbf{p}_i$ are computed through sinusoids, as in the original transformers [VAS2017].

DEtection TRanformer (DETR)

DETR architecture

- The final sequence is imported to a ***standard*** Transformer ***encoder*** consisting of N blocks.
- A matrix $\mathbf{O} \in \mathbb{R}^{d_m \times L_o}$ of L_o ***learned vectors***, namely ***object queries***, is imported to a standard Transformer ***decoder*** which ***decodes*** them in ***parallel***.

- K: the ***maximum number of objects*** in an image.
- L_o: hyperparameter obeying the restriction $L_o > K$.

DEtection TRanformer (DETR)

DETR architecture

- The decoder outputs a matrix $\mathbf{Y} \in \mathbb{R}^{d_m \times L_o}$. Each vector $\mathbf{y}_i \in \mathbb{R}^{d_m}$, $i = 1, \dots, L_o$ passes through ***two different branches***.

- One branch outputs distribution of class probabilities (***classification task***), while the other one regresses bounding box coordinates (***regression task***).

- Overall, the model produces L_o final predictions.

DEtection TRanformer (DETR)

DETR architecture

- The ***classification*** branch consists of a linear projection layer followed by a *Softmax* activation function.

- The ***regression*** branch consists of a 3-layer **perceptron** with *ReLU* activation function.
- It predicts the ***normalized center*** coordinates, ***height*** and ***width*** of the bounding box with respect to the input image.

DEtection TRanformer (DETR)

DETR architecture

- The DETR model predicts a set of L_o bounding boxes, where L_o is usually ***much larger*** than the actual ***number of objects*** in an image.

- An ***additional special class label*** $\emptyset$ is used to represent that ***no object is detected*** within a slot.
- The class $\emptyset$ plays a similar role to the "***background***" class in the standard object detection approaches.

DEtection TRanformer (DETR)

Bipartite matching loss

- Since the number of L_o predictions is much larger than the actual number of objects in an image, a ***special loss function is needed***.

- The loss function must produce an ***optimal bipartite matching*** between predicted and ground truth objects, and ***afterwards*** optimize ***object-specific*** (bounding box) ***losses***.

DEtection TRanformer (DETR)

Bipartite matching loss

- Given an image the set of ground-truth is denoted by $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{N_o}$ where N_o is the number of objects.
- $\mathbf{y}_i \in \mathbb{R}^{K+4}$, where K denotes the number of classes and 4 stands for the bounding box coordinates.
- The set of model predictions in denoted by $\hat{\mathbf{Y}} = \{\hat{\mathbf{y}}_i\}_{i=1}^{L_o}$ where $\hat{\mathbf{y}} \in \mathbb{R}^{K+4}$.
- Since $L_o > N_o$ the set $\mathbf{Y}$ must be padded with $\emptyset$ before finding a bipartite matching between ground-truth and predictions.

DEtection TRanformer (DETR)

Bipartite matching loss

- Bipartite matching is accomplished by finding a permutation of L_o elements, $\sigma \in \mathfrak{S}_{L_o}$ with the lowest cost:

$$\hat{\sigma} = arg \min_{\sigma \in \mathfrak{S}_{L_o}} \sum_{i=1}^{L_o} \mathcal{L}_m \left(\mathbf{y}_i, \hat{\mathbf{y}}_{\sigma(i)}\right).$$

- $\mathcal{L}_m(\mathbf{y}_i, \hat{\mathbf{y}}_{\sigma(i)})$ is a ***pair-wise matching cost*** between ground truth $\mathbf{y}_i$ and a prediction with index $\sigma(i)$.

DEtection TRanformer (DETR)

Bipartite matching loss

- The pair-wise matching cost between ground truth $\boldsymbol{y}_i$ and a prediction with index $\sigma(i)$ is computed as follows:

$$\mathcal{L}_m\left(\mathbf{y}_i, \hat{\mathbf{y}}_{\sigma(i)}\right) = -\mathbb{1}_{\{k_i \neq \emptyset\}} \hat{p}_{\sigma(i)}(k_i) + \mathbb{1}_{\{k_i \neq \emptyset\}} \mathcal{L}_b\left(\mathbf{b}_i, \hat{\mathbf{b}}_{\sigma(i)}\right).$$

- k_i is the ground truth class label.
- $\boldsymbol{b}_i \epsilon [0,1]^4$ is a vector that defines the normalized ground truth bounding-box coordinates, height and width.
- $\hat{p}_{\sigma(i)}$ is the predicted probability distribution corresponding to the ground-truth.

Artificial Intelligence & Information Analysis Lab

DEtection TRanformer (DETR)

Bipartite matching loss

- The regression loss $\mathcal{L}_b(\mathbf{b}_i, \hat{\mathbf{b}}_{\sigma(i)})$ is a linear combination of the L_1 loss and the generalized IoU loss:

$$\mathcal{L}_b\left(\mathbf{b}_i, \hat{\mathbf{b}}_{\sigma(i)}\right) = \lambda_{iou}\mathcal{L}_{iou}\left(\mathbf{b}_i, \hat{\mathbf{b}}_{\sigma(i)}\right) + \lambda_{L_1}\left\|\mathbf{b}_i - \hat{\mathbf{b}}_{\sigma(i)}\right\|_1.$$

  - $\lambda_{iou}, \lambda_{L_1} \in \mathbb{R}$: hyperparameters.

- Both terms are normalized by the number of objects inside the batch of images used during loss computation.

DEtection TRanformer (DETR)

Bipartite matching loss

- A ***Hungarian loss*** is computed for all the pairs matched in the previous step:

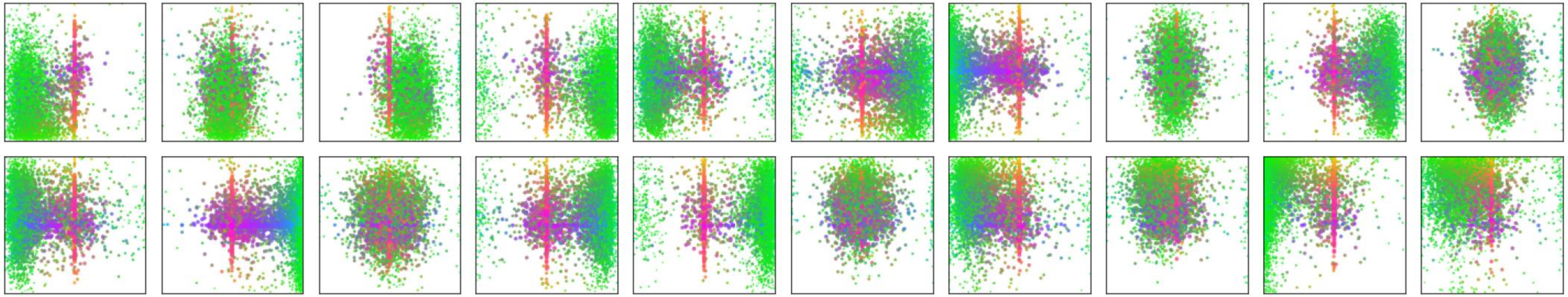
$$\mathcal{L}_H(\mathbf{Y},\hat{\mathbf{Y}}) = \sum_{i=1}^{L_o}\left[-log\hat{p}_{\hat{\sigma}(i)} + \mathbb{1}_{\{k_i \neq \emptyset\}}\mathcal{L}_b(\mathbf{b}_i, \hat{\mathbf{b}}_{\hat{\sigma}(i)})\right].$$

- $\hat{\sigma}$ is the optimal assignment computed in the previous step.

DEtection TRanformer (DETR)

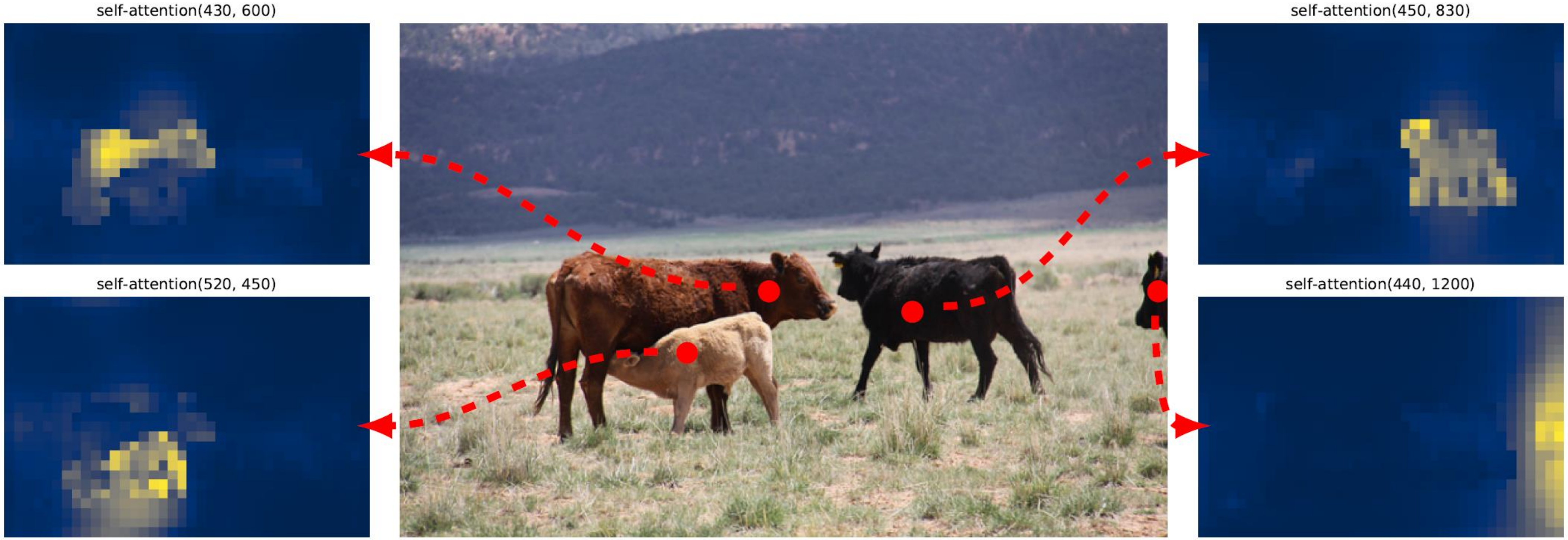
Object queries

- Each ***object query*** learns to ***specialize*** on certain ***areas*** and ***box sizes***.



Visualization of all box predictions on all images from COCO 2017 val set for 20 out of total $L_o = 100$ prediction slots in DETR decoder. Each box prediction is represented as a point with the coordinates of its center in the 1-by-1 square normalized by each image size. The points are color-coded so that green color corresponds to small boxes, red to large horizontal boxes and blue to large vertical boxes.[CAR2020].

DEtection TRanformer (DETR)

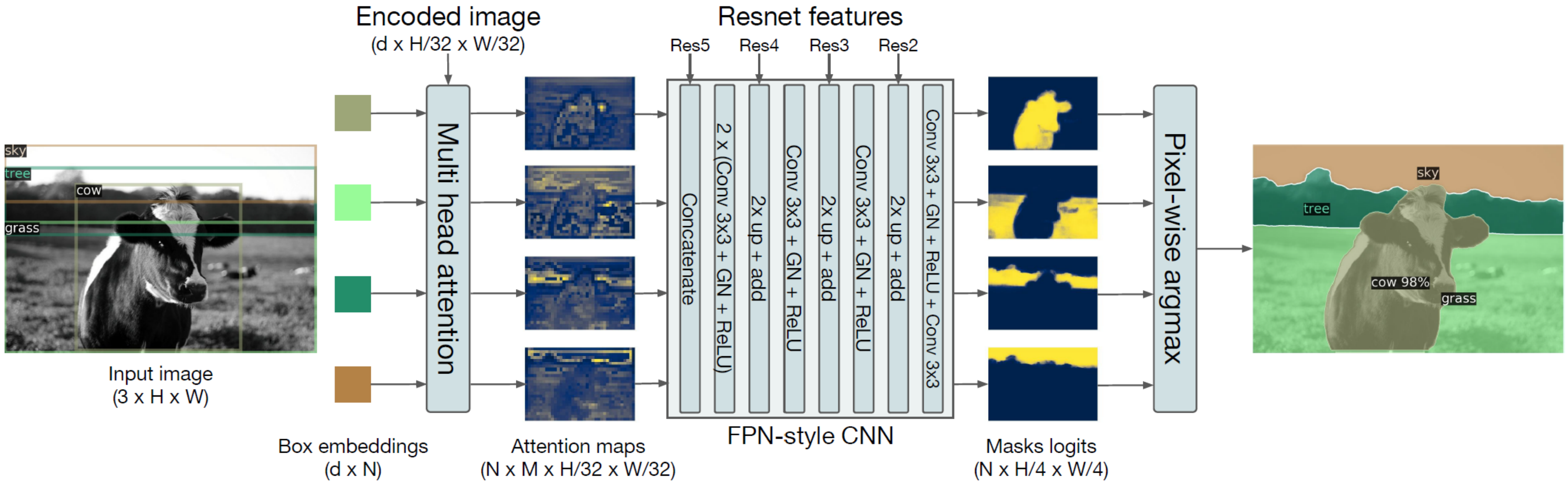


Encoder self-attention for a set of reference points. The encoder is able to separate individual instances. Predictions are made with baseline DETR model on a validation set image [CAR2020].

DEtection TRanformer (DETR)

DETR for panoptic segmentation

- DETR can be naturally extended by adding a segmentation branch on top of the decoder outputs.



DETR with panoptic head. A binary mask is generated in parallel for each detected object, then the masks are merged using pixel-wise argmax [CAR2020].

Artificial Intelligence & Information Analysis Lab

DEtection TRanformer (DETR)

Real Time Detection Transformer (RT-DETR) for forest fire detection.

DEtection TRanformer (DETR)

Pipe defect detection.

Artificial Intelligence &
Information Analysis Lab

DEtection TRanformer (DETR)

Application in industrial inspection

- DETR has been applied for defect detection on pipes for industrial inspection.
- ***Real Time DETR (RT-DETR) did not match YOLO performance.***

Pipe defect detection results [MEN2024].

Model	mAP 0.50	mAP 0.50:95	mAR 0.50:95
Yolov5	0.432	0.216	0.405
Yolov8	0.371	0.168	0.317
Yolo-Nas	0.319	0.140	0.359
Yolov6	**0.519**	**0.251**	**0.444**
RT-Detr	0.398	0.210	0.398

Transformers in Computer Vision

- Motivation and related work
- Transformers
- Vision Transformer (ViT)
- Swin Transformer
- DEtection Transformer (DETR)
- **SEgmentation TRansformer (SETR)**
- Segment Anything Model (SAM)
- DINO
- Video ViT (ViViT)

SEgmentation TRansformer

SETR encoder

- An image $\mathbf{X} \in \mathbb{R}^{HW \times C}$ is split into fixed-size patches $\mathbf{x} \in \mathbb{R}^{N^2C}$.

- Each patch gets linearly embedded as following:

$$\mathbf{x}'_i = \mathbf{W}_e \mathbf{x}_i, \qquad i = 1, \dots, HW$$

- $\mathbf{W}_e \in \mathbb{R}^{d_m \times N^2C}$:learnable parameter matrix.

SEgmentation TRansformer

SETR encoder

- A 24-layer ***pre-trained ViT*** [DOS2021] is employed to generate a matrix of discriminative feature representations on image patches, denoted by $\mathbf{Y} \in \mathbb{R}^{L \times d_m}$, where $L = HW/N^2$.

- In the pre-trained model, positional information is provided through ***additive learnable*** positional encodings of the same dimension d_m as the input vectors $\mathbf{z}_i$:

$$\mathbf{z}_i = \mathbf{x}'_i + \mathbf{p}_i.$$

SEgmentation TRansformer

SETR encoder

- In SETR [ZHE2021], positional encoding employs a $2D$ ***interpolation*** on the ***pre-trained position embeddings***, according to their location in the original image for different input size fine-tuning.

- Given $\mathbf{Y} \in \mathbb{R}^{L \times d_m}$, a decoder is used to recover the original image resolution. Crucially there is ***no down-sampling*** in spatial resolution, but ***global context modeling*** at every layer of the encoder transformer.

Artificial Intelligence & Information Analysis Lab

SEgmentation TRansformer

SETR decoder

- The goal of SETR decoder is to generate the segmentation results in the original $2D$ image space $\mathbb{R}^{H\times W\times C}$.

- The encoder features $\mathbf{Y} \in \mathbb{R}^{L\times d_m}$ must be translated into a $3D$ feature map $O \in \mathbb{R}^{H\times W\times C}$.

- Three different designs are explored: ***naïve***, progressive upsampling (***UP***) and multi-level feature aggregation (***MLA***).

SEgmentation TRansformer

SETR naïve decoder

- A simple ***2-layer network*** composed by 1×1 convolutions with ReLU activation function in between is used.
- The output of this network, is simply ***bilinearly up sampled*** to the original image resolution followed by a classification layer with pixel-wise cross-entropy loss.

SEgmentation TRansformer

SETR UP decoder

- Instead of one-step upscaling which may introduce noisy predictions, a ***progressive upsampling strategy*** that alternates convolutional layers and upsampling operations is considered.

- To maximally mitigate the adversarial effect, each ***upsampling is restricted to*** $2\times$.

SEgmentation TRansformer

SETR MLA decoder

- ***Multi-level feature aggregation*** is employed in similar spirit of feature pyramid networks.

- Intermediate feature representations $\mathbf{Y}^{l_e}$, $l_e = 1, \dots, L_e$ at the encoder l_e^{th} layer) share the same resolution.
- Multi-level feature aggregation is applied through sampling feature representations $\mathbf{Y}^m$ from M layers: $m \in \left\{\frac{L_e}{M}, 2\frac{L_e}{M}, \dots, M\frac{L_e}{M}\right\}$.

SEgmentation TRansformer

SETR MLA decoder

- M streams are deployed, with each focusing on one specific encoder layer.

- Each $\mathbf{Y}^m \in \mathbb{R}^{L \times d_m}$ is mapped to a 3D feature map $Y^m \in \mathbb{R}^{\frac{H}{N} \times \frac{W}{H} \times d_m}$.

SEgmentation TRansformer

SETR MLA decoder

- A 3-layer (kernel sizes 1×1, 3×3, and 3×3) network is applied with the feature channels halved at the first and third layers respectively, and the spatial resolution upscaled $4\times$ by bilinear operation after the third layer.

- To enhance the interactions across different streams, a top-down ***aggregation via element-wise addition*** after the first layer is introduced.

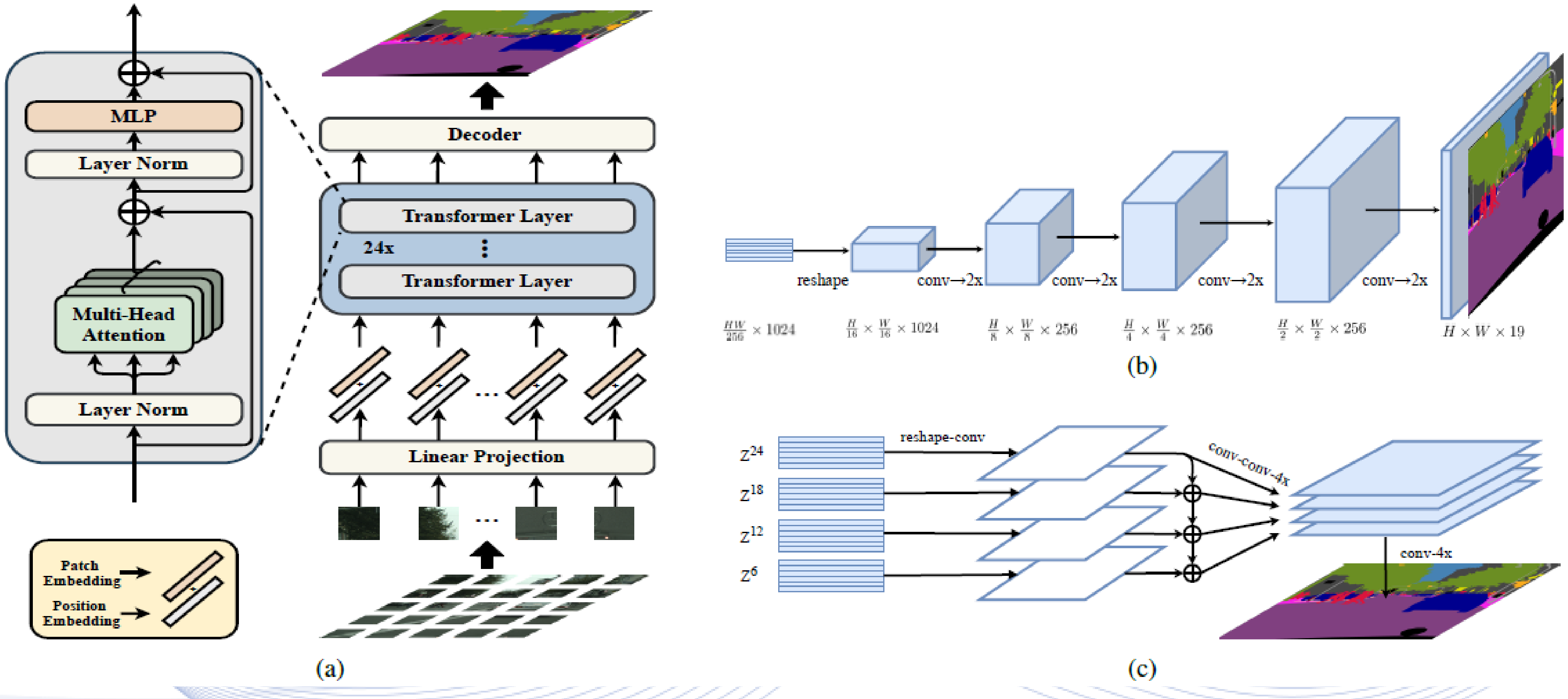
SEgmentation TRansformer

SETR MLA decoder

- An additional 3×3 convolutional layer is applied after the element-wise feature sum.

- After the third layer, the fused feature from all the streams via channel-wise concatenation is obtained which is the bilinearly up-sampled $4 \times$ to the full resolution.

SEgmentation TRansformer



a) SETR architecture, b) UP decoder, c) MLA decoder [ZHE2021].

Transformers in Computer Vision

- Motivation and related work
- Transformers
- Vision Transformer (ViT)
- Swin Transformer
- DEtection Transformer (DETR)
- SEgmentation TRansformer (SETR)
- **Segment Anything Model (SAM)**
- DINO
- Video ViT (ViViT)

Segment Anything Model (SAM)

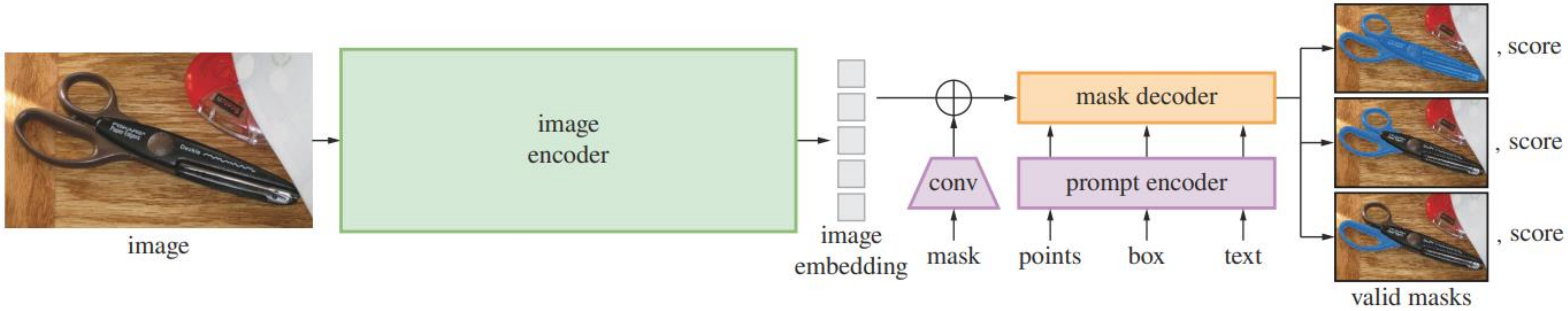
Segment Anything Model (***SAM***) [KIR2023] stands as a ***foundational model*** for object and image region segmentation.

- It is designed to be ***prompt-responsive***, accepting both:
  - ***sparse prompts*** (including points, bounding boxes, and text) and
  - ***dense prompts*** (masks) alongside the input image.

- The main novelty of SAM is the ***prompt encoder***.

Segment Anything Model (SAM)

Model architecture



SAM architecture [KIR2023].

Segment Anything Model (SAM)

SAM architecture

- ***Image encoder***: A ***Masked Auto Encoding*** (MAE) pre-trained Vision Transformer (ViT) encoder that embeds the image and extracting its essential features.
- ***Prompt encoder***: Lightweight prompt encoder designed to transform user prompts into embedding vectors in real time.
- ***Mask decoder***: Lightweight decoder dedicated to predicting segmentation masks, by integrating both the image and prompt embeddings.

Segment Anything Model (SAM)

Image Encoder is any network with the following input and output:

- Input: Image $\mathbf{X} \in \mathbb{R}^{H_0 \times W_0 \times C_0}$, typically rescaled and padded to an analysis of $1024 \times 1024 \times 3$.
- Output: Image embedding $\mathbf{Y} \in \mathbb{R}^{H \times W \times C}$, typically with size $64 \times 64 \times 256$.

- SAM image encoder is the MAE pre-trained Vision Transformer (ViT).

Segment Anything Model (SAM)

Prompt Encoder

- Input: N_t sparse prompts (points, bounding boxes and text).
- Output: N_t vectorial embeddings (one per prompt).

- ***Point***: Sum of positional encoding of points location $\mathbf{p}_i \in \mathbb{R}^{256}$ and a learned embedding $\mathbf{x}_i \in \mathbb{R}^{256}$.
- ***Bounding box***: Embedding pair of upper left and lower right corner $\mathbf{x}_{iu} \in \mathbb{R}^{256}$, $\mathbf{x}_{il} \in \mathbb{R}^{256}$.
- ***Text***: Text prompts are fed into the CLIP text encoder, generating an output embedding $\mathbf{x}_i \in \mathbb{R}^{256}$ which serves as the input for the prompt encoder.

Segment Anything Model (SAM)

Mask prompt Encoder

Dense prompts (masks) are embedded using 1×1 and 2×2 convolutions to produce $\mathbf{Y}_m \in \mathbb{R}^{H \times W \times C}$.

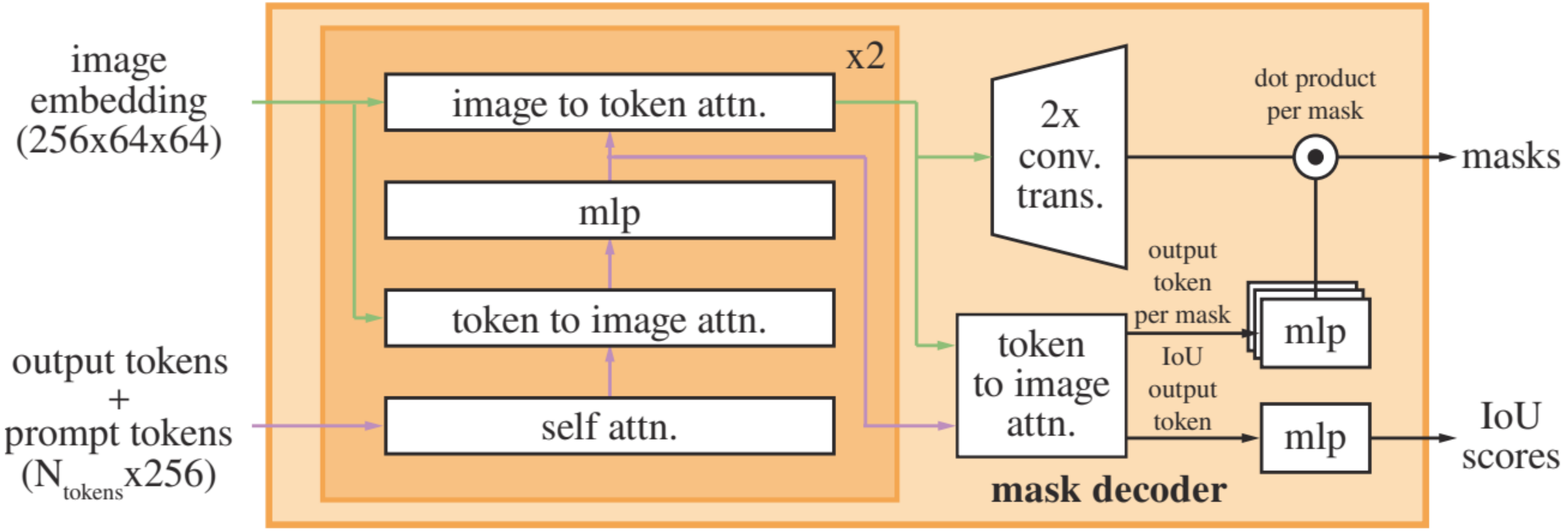
- Typically, one segmentation mask is provided.
- The mask and image embeddings $\mathbf{Y}, \mathbf{Y}_m$ are added element-wise:

$$\mathbf{Y}' = \mathbf{Y} + \mathbf{Y}_m.$$

- If there is no dense prompt, then a default learned embedding $\mathbf{Y}_o \in \mathbb{R}^{H \times W \times C}$ is added to the image embedding.

Segment Anything Model (SAM)

Mask Decoder (modified Transformer decoder) maps the image embedding $\mathbf{Y}'$ and a set of prompt embeddings $\mathbf{x}_i, i = 1, \ldots, N_t$ to output masks $\mathbf{Y}_o \in \mathbb{R}^{H \times W \times 3}$.



SAM mask decoder [KIR2023].

Segment Anything Model (SAM)

Mask Decoder layer has 4 steps:

- ***Self-attention*** on the prompt embeddings.
- ***Cross-attention*** from prompt embeddings (as queries) to the image embedding.
- ***Point-wise MLP*** to update the prompt embeddings.
- ***Cross-attention*** from image embeddings (as queries) to the prompt embedding.

Segment Anything Model (SAM)

Mask Decoder

- 3 learned output token embeddings are inserted in the set of prompt embeddings.

- A small MLP head estimates the IoU between each predicted mask and the object it covers, ranking the predicted masks.

Segment Anything Model (SAM)

SAM loss functions

- SAM loss is the sum of a mask loss and an IoU loss.
- The ***mask loss***, the loss in the supervised mask prediction, is a linear combination of ***focal loss*** and ***dice loss*** in a $20:1$ focal loss to dice loss ratio.
- The ***IoU loss***, for the IoU prediction head, is the mean-square-error loss between the IoU prediction and the predicted mask IoU with the ground truth mask.
- The IoU loss is added to the mask loss with a constant scaling factor of 1.0.

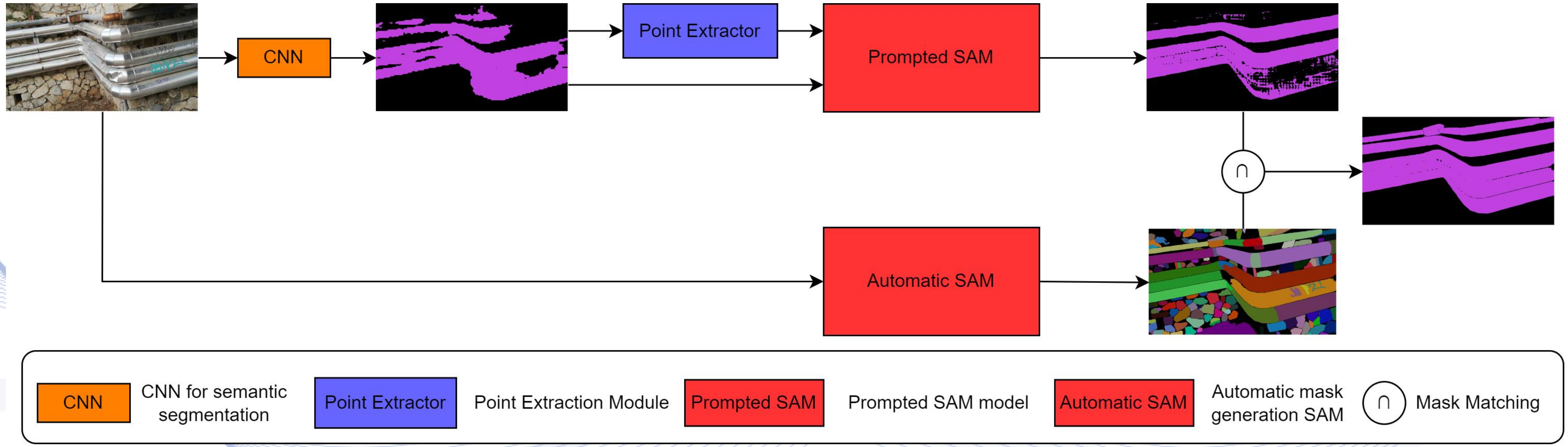
Segment Anything Model (SAM)

Industrial Inspection Applications

SAM has been applied to real-world use cases, such as pipe region segmentation.

- A CNN model and SAM were combined in order to produce masks of pipes in the image.

Segment Anything Model (SAM)



Pipe Image Segmentation Architecture [PSA2024].

Segment Anything Model (SAM)

Pipe Image Segmentation

Transformers in Computer Vision

- Motivation and related work
- Transformers
- Vision Transformer (ViT)
- Swin Transformer
- DEtection Transformer (DETR)
- SEgmentation TRansformer (SETR)
- Segment Anything Model (SAM)
- **DINO**
- Video ViT (ViViT)

DINO

DINO [CAR2021] is a ***self-supervised ViT*** trained in a self-***DI***stillation with ***NO*** labels fashion.

- DINO is used in:
  - Image feature extraction
  - Image classification.
- Feature representations extracted from DINO contain ***explicit information*** about the ***semantic segmentation*** of an image and they are excellent ***k-NN classifiers*** (78.3% top-1 accuracy on ImageNet).

DINO

DINO architecture

- Two different random transformations of an input image are passed through two ***different versions*** of the ***same ViT***.
- The two versions of ViT are called ***student and teacher network*** and they are denoted by $g_s(\cdot\,; \mathbf{W}_s)$ and $g_t(; \mathbf{W}_t)$respectively.
- Both student and teacher have the same architecture.
- The ***teacher parameters*** are updated with an ***exponential moving average*** of the ***student ones***.

Artificial Intelligence & Information Analysis Lab

DINO

DINO architecture

- The output of the teacher network is centered with a mean computed over the batch.

- Each network outputs a K dimensional feature that is normalized with a temperature *Softmax* over the feature dimension.

- Their similarity is measured with a cross-entropy loss.

DINO

DINO architecture

- Given an input image $\mathbf{X} \in \mathbb{R}^{C \times HW}$ both networks output probability distributions denoted by $p_s(\mathbf{X}), p_t(\mathbf{X}) \in \mathbb{R}^K$.

- The temperature *Softmax* for the student is computed by:

$$p_s^j(\mathbf{X}_i) = \frac{\exp(\boldsymbol{g}_s^j(\mathbf{X}\,;\mathbf{W}_s)/\tau_s)}{\sum_{k=1}^{K} \exp(\boldsymbol{g}_t^j(\mathbf{X}\,;\mathbf{W}_t)/\tau_s)}.$$

- τ_s is the temperature parameter controlling the sharpness of distribution.

DINO

DINO architecture

- Each input image $\mathbf{X} \in \mathbb{R}^{C \times H \times W}$ is randomly cropped multiple times forming **2 global crops** at resolution covering 50% of the original image and several ***local crops*** covering less than 50% of the original image.

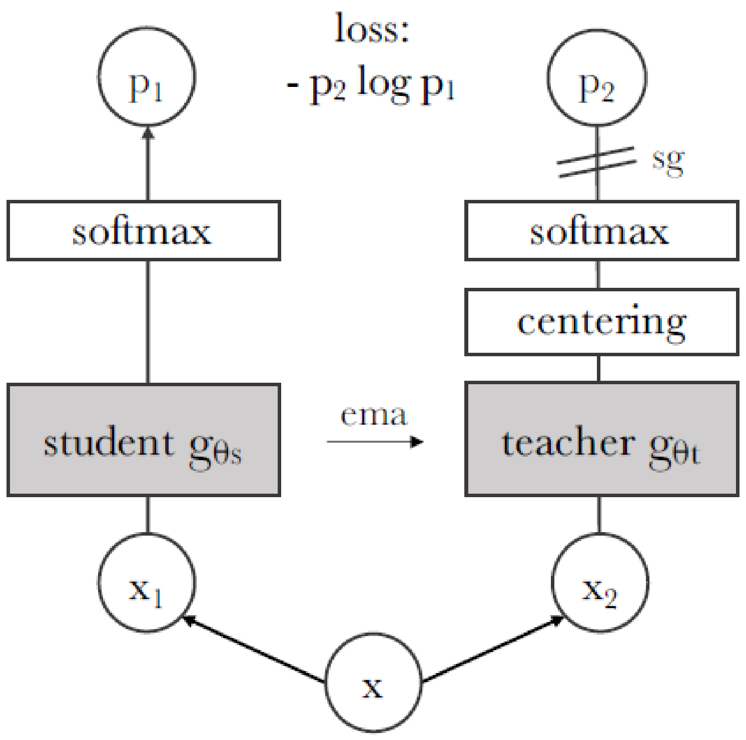
- All crops are passed through the student network, while only the global ones are passed through the teacher one.

- For a specific image, the network is self-trained based on all pairs of random crops.

DINO

DINO architecture

- A stop-gradient (sg) operator is applied on the teacher network to propagate gradients only through the student one.
- Centering prevents one dimension to dominate, but encourages collapse to the uniform distribution.
- *Softmax* temperature compensates for this.

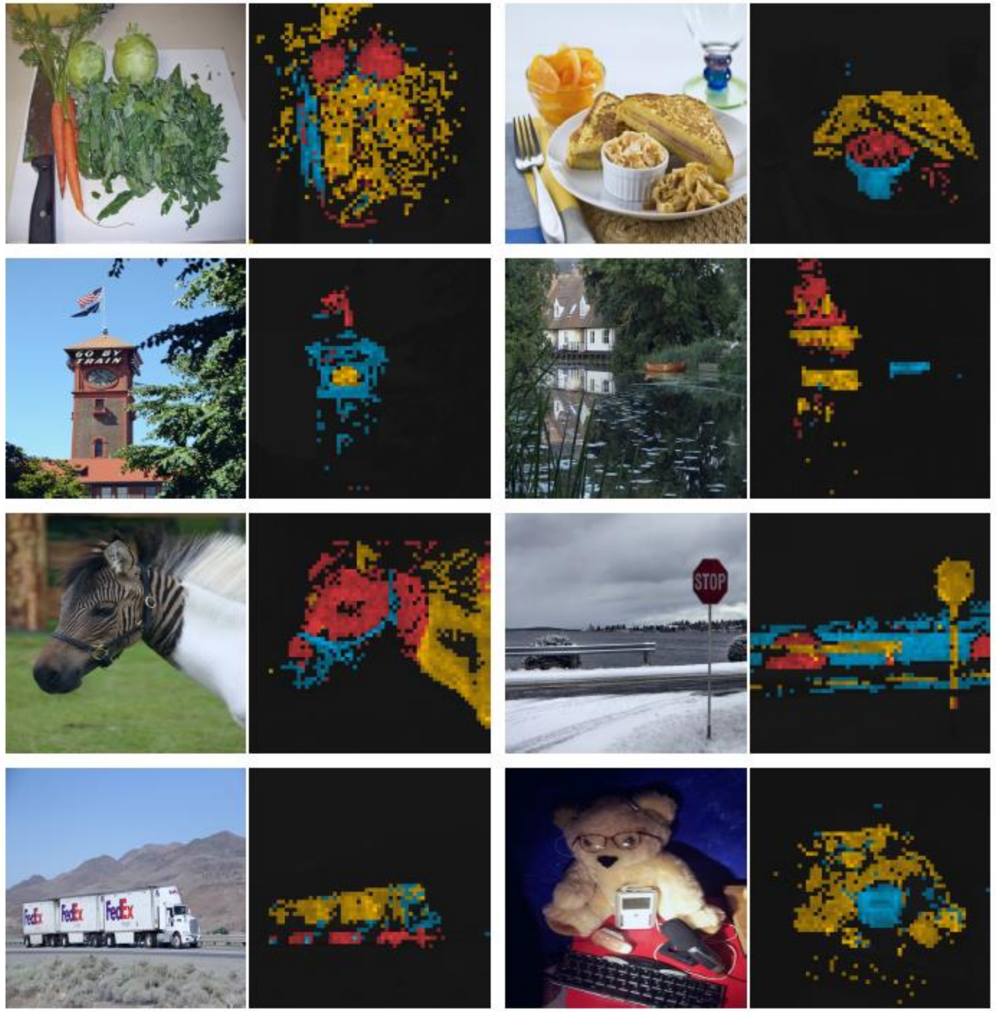


DINO architecture [CAR2021].

DINO

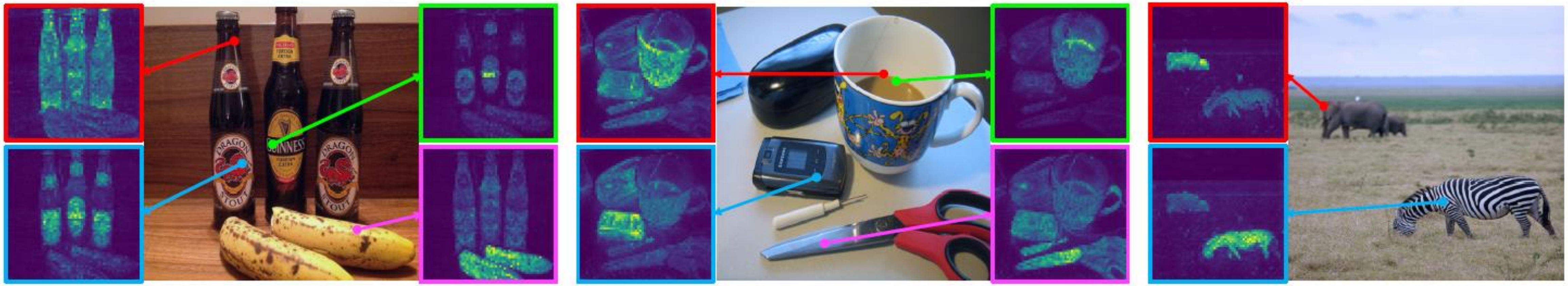
- Different heads focus on different objects or parts.
- This is visualized using different colors.

- Attention is plotted for the query corresponding to the extra learnable embedding.



Self-attention per head on the last DINO layer [CAR2021].

DINO



Self-attention on the last DINO layer for a set of reference points [CAR2021].

Artificial Intelligence & Information Analysis Lab

Transformers in Computer Vision

- Motivation and related work
- Transformers
- Vision Transformer (ViT)
- Swin Transformer
- DEtection Transformer (DETR)
- SEgmentation TRansformer (SETR)
- Segment Anything Model (SAM)
- DINO
- **Video ViT (ViViT)**

Video ViT (ViViT)

ViViT [ARN2021] extracts ***spatio-temporal tokens*** from input ***video*** and encodes them through a series of Transformer ***encoder*** layers.

- ViViT is used in:
  - video feature extraction
  - video classification.

Video ViT (ViViT)

Four different ViViT variants ***factorize*** different components of the transformer encoder over the ***spatial*** and ***temporal*** dimensions:

- Spatio-temporal attention.
- Factorized encoder.
- Factorized self-attention (each single head is factorized).
- Factorized multi-headed attention (factorization across heads).

Video ViT (ViViT)

ViViT architecture

- A video $\mathbf{V} \in \mathbb{R}^{THW \times C}$ is mapped into a sequence of spatio-temporal fixed-size patches $\mathbf{v}_p \in \mathbb{R}^{N_T N_H N_W C}$.
- This can be done through ***uniform frame sampling*** or ***tubelet embedding***.
- Each patch gets linearly embedded as following:

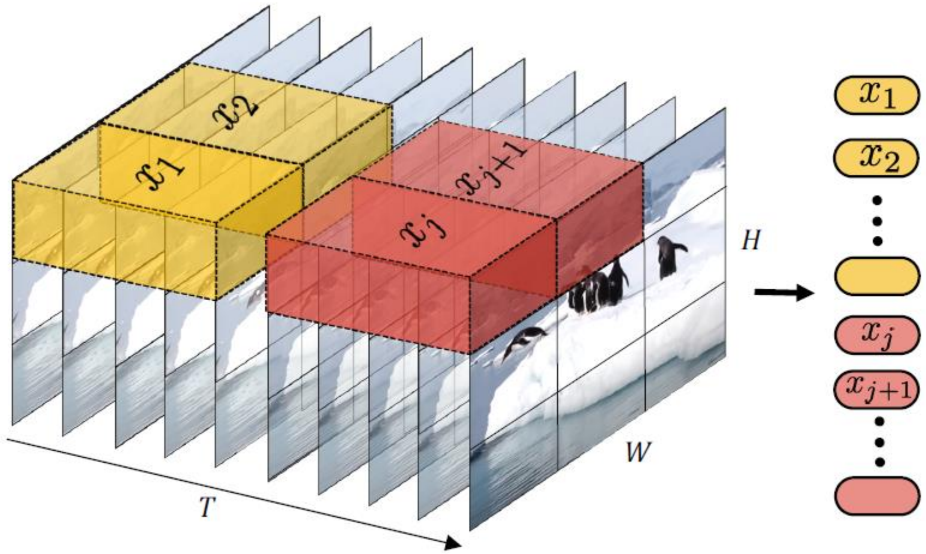
$$\mathbf{v}'_p = \mathbf{W}_e \mathbf{v}_p \in \mathbb{R}^{d_m}.$$

- $\mathbf{W}_e \in \mathbb{R}^{d_m \times N_T N_H N_W C}$ is learnable.

Video ViT (ViViT)

ViViT architecture

- In ***tubelet embedding***, spatio-temporal patches of dimensions $N_T \times N_H \times N_W$ are extracted.



Tubelet embedding [ARN2021].

Video ViT (ViViT)

ViViT architecture

- Positional information is provided through ***additive learnable*** positional encodings of the same dimension d_m as the input embeddings:

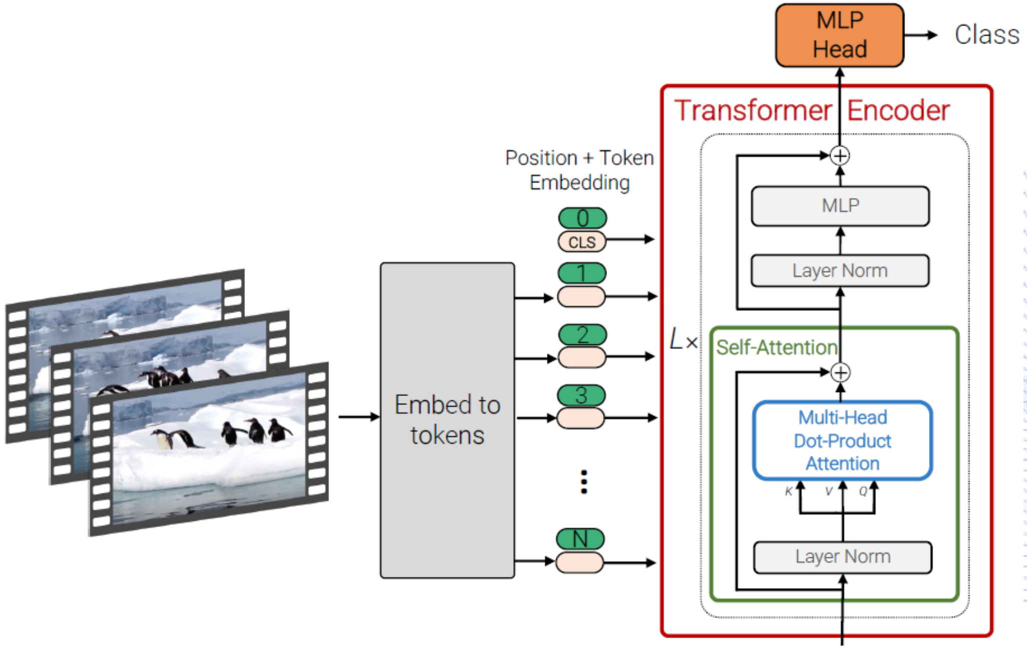
$$\mathbf{z}_{pi} = \mathbf{v}'_{pi} + \mathbf{p}_i.$$

- The video model processes N_T times more tokens than the one pre-trained on images.
- Thus, as an initialization step, the ***positional encodings pre-trained on images*** are repeated temporally to all frames. Then they are fine-tuned on video.

Video ViT (ViViT)

Spatio-temporal attention

- All spatio-temporal tokens are simply forwarded through the Transformer encoder.



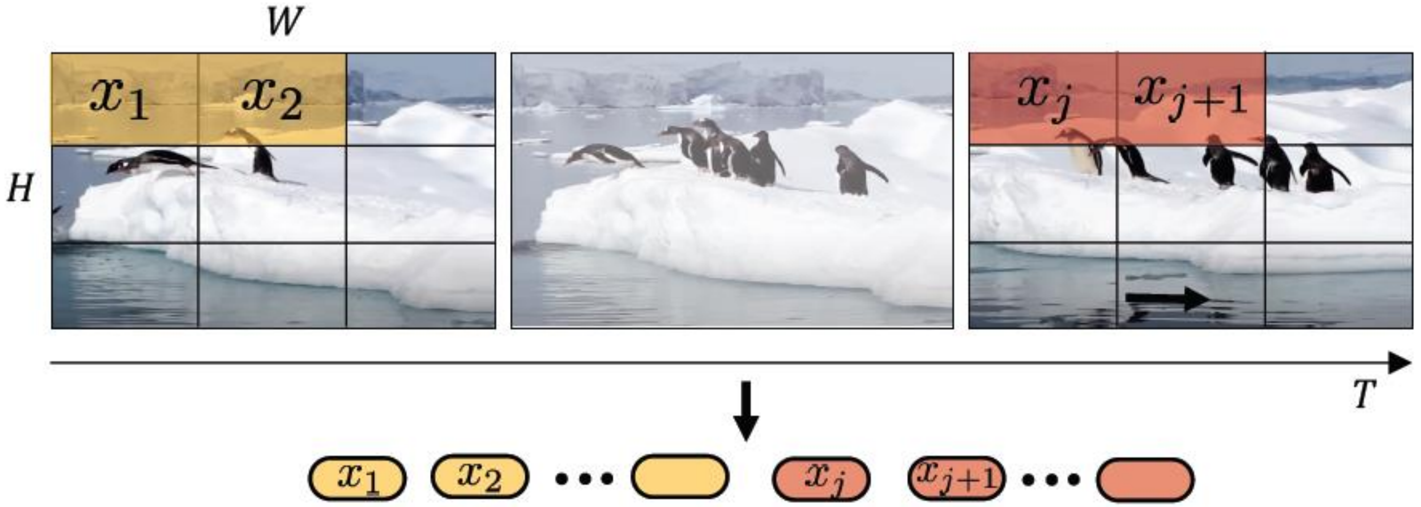
Tubelet embedding [ARN2021].

Artificial Intelligence & Information Analysis Lab

Video ViT (ViViT)

Factorized ViViT architecture

- In ***uniform frame sampling***, N_T video frames are uniformly sampled and each $2D$ frame is split into patches of dimensions $N_H \times N_W$.



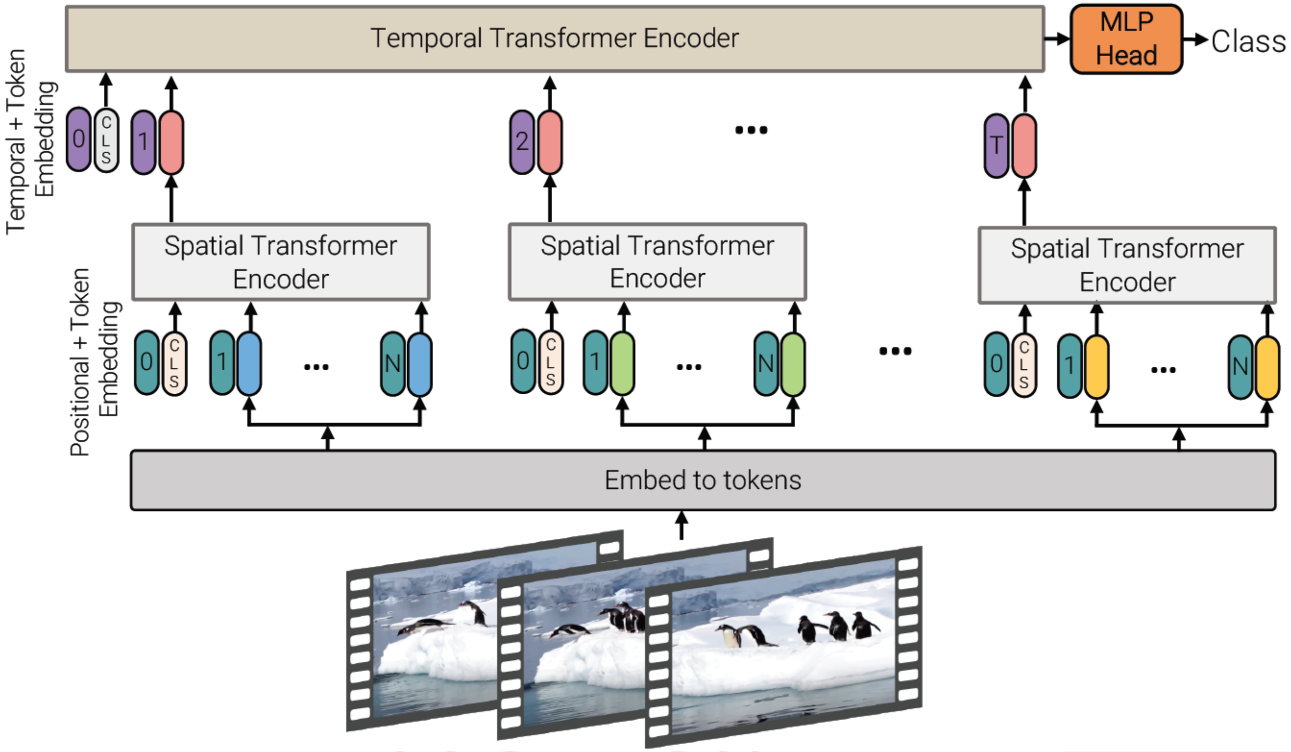
Uniform video frame sampling [ARN2021].

Video ViT (ViViT)

Factorized encoder

- The model consists of two separate encoders in series, a spatial and a temporal one.

- It corresponds to a late fusion of spatial and temporal information.



Factorized encoder [ARN2021].

Video ViT (ViViT)

Factorized encoder

- The ***spatial encoder*** captures correlations between tokens extracted from the same video frame, to produce a latent representation per frame.

- Like ViT, an ***extra learnable embedding*** is appended to the beginning of the spatial sequence.
- Its state at the spatial encoder output serves as the ***latent frame representation*** $\mathbf{h}_t \in \mathbb{R}^{d_m}$ where t denotes time.

Video ViT (ViViT)

Factorized encoder

- The ***temporal encoder*** models interactions between latent representations at different time instances.

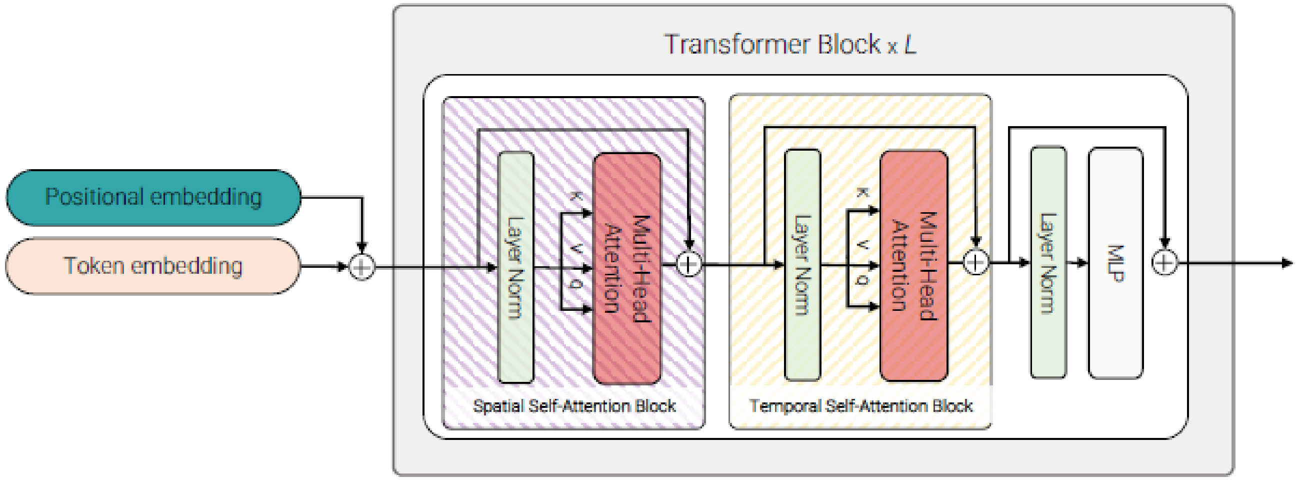
- Again, an ***extra learnable embedding*** is appended to the beginning of the temporal sequence.

- Its state at the temporal encoder output serves as the ***final video representation*** $\mathbf{y} \in \mathbb{R}^{d_m}$.

Video ViT (ViViT)

Factorized self-attention

- Within each transformer block, the multi-headed self-attention operation is factorized into two operations that first only compute self-attention spatially, and then temporally.

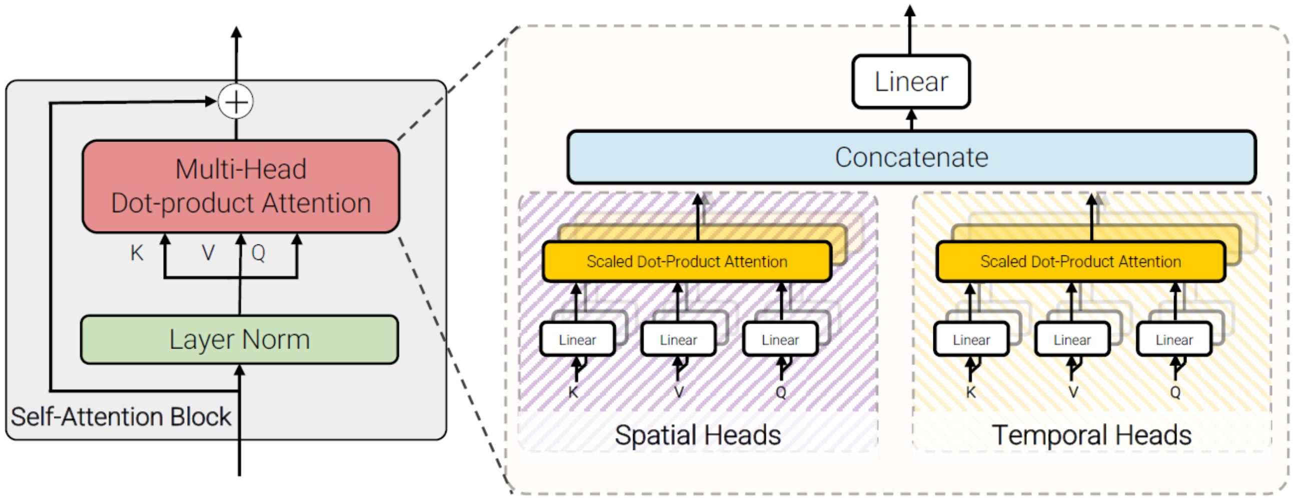


Factorized self-attention [ARN2021].

Video ViT (ViViT)

Factorized multi-headed attention

- Half of the heads compute self-attention over the spatial axis, while the other half compute self-attention over the temporal axis.



Factorized multi-headed attention [ARN2021].

Bibliography

[ELA2002] M. Elad, "On the origin of the bilateral filter and ways to improve it", IEEE Transactions on Image Processing, vol. 11, issue 10, pp. 1141-1151, 2002

[BUA2005] A. Buades, B. Coll, J.M. Morel, "A Review of Image Denoising Algorithms, with a New One", Multiscale Modelling & Simulation, vol. 4, issue 2, pp. 490-530, 2005

[TAK2007] H. Takeda, S. Farsiu, P. Milanfar, Kernel regression for image processing and reconstruction, IEEE Transactions on Image Processing, vol. 16, issue 2, pp. 349-366, 2007

[VAS2017] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, "Attention Is All You Need", Advances in Neural Information Processing Systems (NIPS), 2017.

[GRAV2014] A. Graves, G. Wayne, I. Danihelka, Neural Turing Machines, arxiv preprint arXiv:1410.5401, 2014

[ZHA2018] H. Zhang, I. Goodfellow, D. Metaxas, A. Odena, "Self-Attention Generative Adversarial Networks", arxiv preprint arXiv:1805.08318, 2018

[WAN2018] X. Wang, R. Girshick, A. Gupta, K. He, "Non-local Neural Networks", Conference on Computer Vision and Pattern Recognition (CVPR), 2018

Bibliography

[RAM2019] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, J. Shlens, "Stand-Alone Self-Attention in Vision Models", Advances in Neural Information Processing Systems (NeurIPS), 2019

[HU2019] H. Hu, Z. Zhang, Z. Xie, S. Lin, "Local Relation Networks for Image Recognition", International Conference on Computer Vision (ICCV) 2019

[RAD2018] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, "Improving Language Understanding by Generative Pre-Training", OpenAI Blog, 2018.

[RAD2018] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, "Improving Language Understanding by Generative Pre-Training", OpenAI Blog, 2018.

[DEV2018] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", arxiv preprint arXiv:1810.04805, 2018.

[DOS2021] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, "An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale", International Conference on Learning Representations (ICLR), 2021

Bibliography

[PAR2022] N. Park, S. Kim, "How do vision transformers work?", International Conference on Learning Representations (ICLR), 2022

[RAG2022] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, A. Dosovitskiy, "Do Vision Transformers See Like Convolutional Neural Networks?", Neural Information Processing Systems (NeuIPS), 2022

[LIU2021] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", International Conference on Computer Vision (ICCV), 2021

[LIN2017] T.Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, S. Belongie, "Feature Pyramid Networks for Object Detection", Conference on Computer Vision and Pattern Recognition (CVPR), 2017

[RON2015] O. Ronneberger, P. Fischer, T. Brox, "U-net: Convolutional networks for biomedical image segmentation", International Conference on Medical image computing and computer-assisted intervention, 2015

[CAR2020] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, "End-to-End Object Detection with Transformers", European Conference on Computer Vision (ECCV), 2020

Bibliography

[ARN2021] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, C. Schmid, "ViViT: A Video Vision Transformer", arxiv preprint arXiv:2103.15691, 2021

[CAR2021] M. Caron, H. Touvron, I. Misra, H. Jegou, J. Mairal, P. Bojanowski, A. Joulin, "Emerging Properties in Self-Supervised Vision Transformers", International Conference on Computer Vision (ICCV), 2021

[ZHE2021] S. Zheng, J. Lu, H. Zao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P.H.S. Torr, L. Zhang, "Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers", Conference on Computer Vision and Pattern Recognition (CVPR) 2021

Artificial Intelligence & Information Analysis Lab

Q & A

Thank you very much for your attention!

More material in
http://icarus.csd.auth.gr/cvml-web-lecture-series/

Contact: Prof. I. Pitas
pitas@csd.auth.gr