Računalni vid

Modeli utemeljeni na pažnji

Marin Oršić
Siniša Šegvić
UniZg-FER D307

Agenda

Part 1: attention in natural language understanding

- complementing recurrent models with attention
- basic and extended formulation, transformer architecture

Part 2: attention for visual recognition

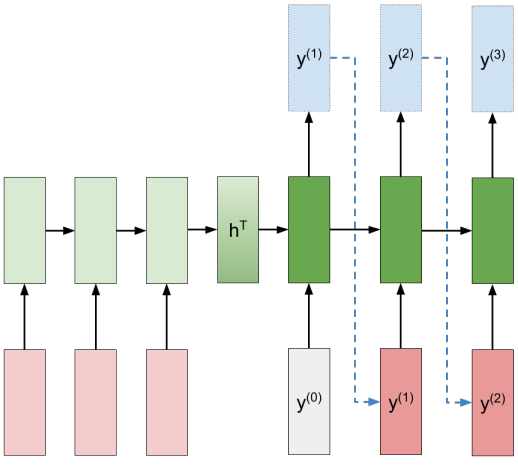
- visual transformers, shifted windows etc

Part 3: properties of visual transformers

- self-supervised learning
- robustness, generalization quality, inductive biases

NLP: Recurrent recognition

Classic recurrent models for natural language (cca 2014) involve recurrent encoder (left) and recurrent decoder (right):



[tutek22du]

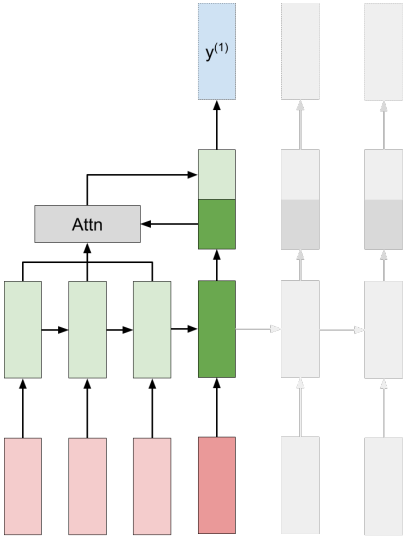
Hidden state of the encoder (light green) absorbs the input (bottom-left).

Decoder inputs the encoder state (left) and the translation so far (bottom-right).

Decoder produces the output while using its hidden state (dark green) as memory

Weaknesses: i) the encoder state unable to remember the entire input sequence, ii) slow sequential learning.

NLP: Recurrence + attention



[tutek22du]

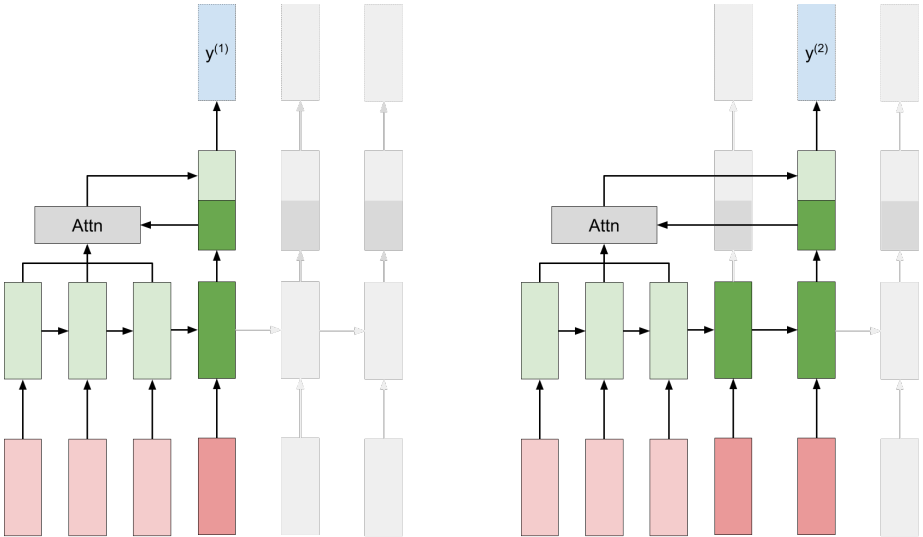
Idea: instead of remembering everything, the decoder learns to associate its state with an appropriate mix of encoder states.

The mix is produced by weighted pooling within the attention module.

The weights correspond to the similarity between the current decoder state and all encoder states.

Each decoder output observes the current decoder state (that encodes the broad sense) and a pool of encoder states (that encode the details).

NLP: Recurrence + attention



[tutek22du]

NLP: Basic attention

All kinds of attention rest upon similarity between keys K and the query q.

In recurrent seq2seq models these are defined as:

$$k^{(t')} = h_{enc}^{(t')}, t' \in [0..T\rangle$$
$$q^{(t)} = h_{dec}^{(t)}.$$

We denote scalar similarity between the $q^{(t)}$ and all $k^{(t')}$ as:

$$\mathrm{s}^{(t,t')} = \mathrm{sim}(q^{(t)}, k^{(t')})$$

NLP: Basic attention (2)

The basic attention requires similarity between the query and *all* T keys:

$$s_K^{(t)} = \text{sim}(q^{(t)}, K), \text{ where } s_K^{(t)} \in \mathbb{R}^T, \text{ and } K = [k^{(1)}, \ldots, k^{(T)}] .$$

Similarities are normalized to a probability distribution:

$$\alpha_K^{(t)} = \text{softmax}(s_K^{(t)}) .$$

Finally, the attention outputs a weighted pool of the hidden encoder states:

$$\text{attn}(q^{(t)}, K) = \sum_{t'}^{T} \alpha_{t'}^{(t)} k^{(t')} .$$

NLP: Basic attention (3)

To conclude, the basic attention recovers a conditioned pool of the input set K.

- the pooling mechanism is conditioned upon the query q.
- different queries give rise to different pools
- no parameters involved so far

Important property: attention is invariant to permutations

- extremely good fit for recognition upon graphs!
- works quite good for vision as well

NLP: Similarity

How to formulate similarity?

1. differentiable module with parameters W_1 (matrix) and w_2 (vector) [bahdanau14iclr]:

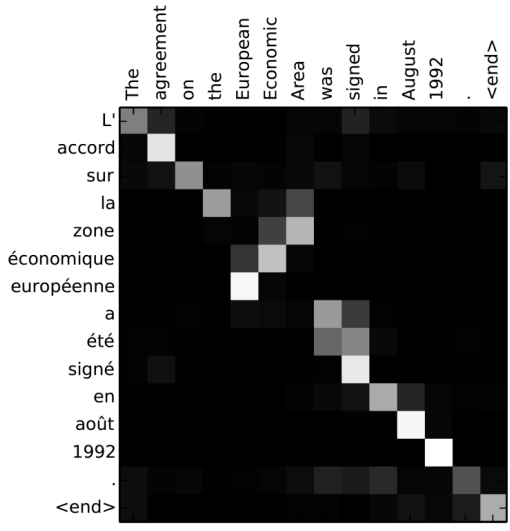
$$s^{(t,t')} = w_2^\top \cdot \tanh(W_1 \cdot [q^{(t)}; k^{(t')}]) \; .$$

2. Scalar product (condition: $\dim(q) = \dim(k)$)

$$s^{(t,t')} = \frac{q^{(t)\top} k^{(t')}}{\sqrt{\dim(k)}} \; .$$

- this is the most popular formulation, at least in vision
- still no parameters involved
- why do we scale with dimension size k?

NLP: Similarity - visualization



[tutek22du]

Similarity between the hidden encoder and the decoder for French to English translation.

NLP: Extended attention

Extended attention derives **keys** and **values** from the same input:

$$k_i = f_k(x_i), \qquad v_i = f_v(x_i) \ .$$

In practice, f_k and f_v are projections:

$$k_i = W_k x_i, \qquad v_i = W_v x_i \ .$$

Extended attention pools values according to similarity of the keys and the query:

$$\alpha_K = \text{softmax}(\text{sim}(q, K)),$$
$$z = \sum_t^T \alpha_t v^{(t)} \ .$$

NLP: Self-attention?

Suppose we wish to use basic attention as a **layer** that operates on input representation K.

We are tempted to use queries from the **same** input, however then $\text{attn}(k_i, K)$ may approach one-hot vector e_i...

- we can avoid this with **learned queries** as we show here;
- note that recent algorithms appear not to suffer from this problem!

Self-attention with a learned query (free parameter) w_j:

$$\hat{\alpha}_K^{(j)} = \text{softmax}(\text{attn}(w_j, K)),$$
$$z_j = \sum_t^T \hat{\alpha}_t^{(j)} v^{(t)} .$$

Intuitively, w_j correspond to latent topics such as *slang*, *football*, *middle east*, etc.

NLP: Extended self-attention

Suppose we have a set of tokens $X_{N\times D} = \{x_i^\top\}$.

We project X onto keys, queries and values by simple matrix multiplication:

$$K_{N\times F} = X \cdot W_k^\top$$
$$Q_{N\times F} = X \cdot W_q^\top$$
$$V_{N\times D'} = X \cdot W_v^\top$$

We determine similarity between all queries q_i and all keys k_i:

$$s_{ij} = \mathrm{sim}(q_i, k_j)$$
$$S = Q \cdot K^\top \ .$$

NLP: Extended self-attention (2)

The weight matrix α now activates rows of S with softmax:

$$\alpha = softmax(S/\sqrt{F}, axis = 1) \ .$$

` Outputs $Z = \{z_i\}$ are linear combinations of values V.

- of course, the weights correspond to the elements of α:

$$z_i = \sum_j \alpha_{ij} \cdot v_j \ .$$

The above formulation can be used as a standard layer of a deep model!

- parameters: W_k ($F \times D$), W_q ($F \times D$), W_v ($D' \times D$).
- it has been found useful beyond sequence-to-sequence translation

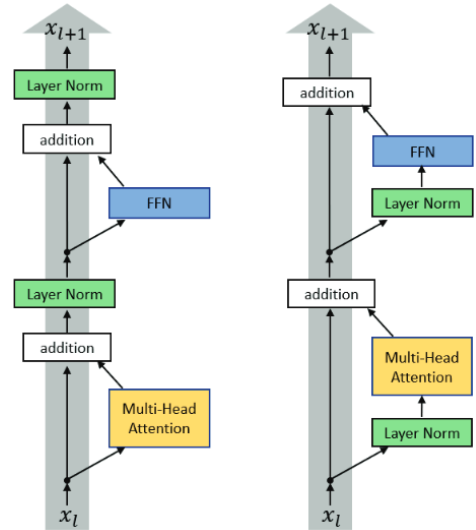
NLP: Self-attention in practice

Transformers alternate cross-token mixing through attention with intra-token mixing through projection.

- related to fully connected "mixers" (resMLP)
- caching the similarity matrix is O(n^2)

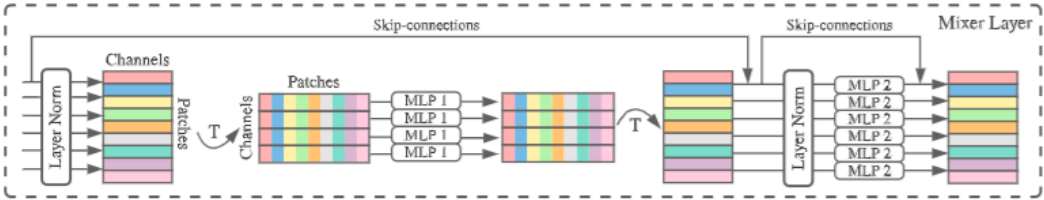
Consider the sentence: "Let us start right now":

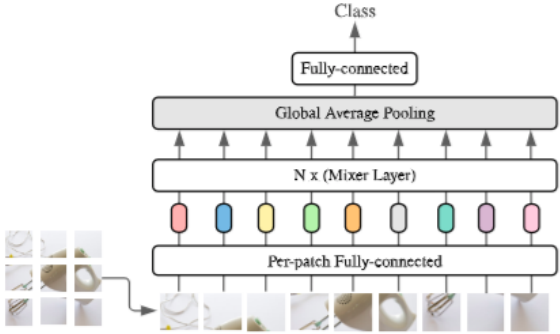
- the word right can denote opposite than left, forward, correct, entitlement
- a red blob can be an apple or a Japanese flag
- the transformers disambiguate the input by consulting the context through attention

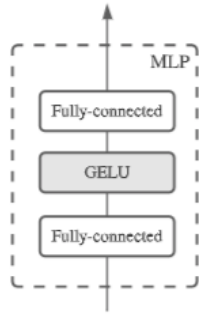


[xiong20icml]

NLP: Cross-token mixing with MLP-mixer





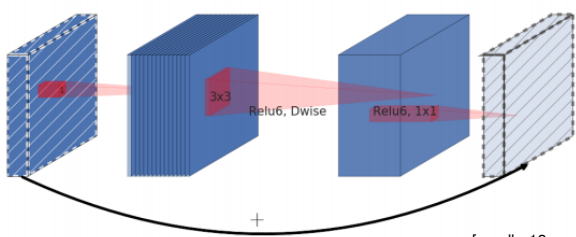


[tolstikhin21neurips]

NLP: Multi-head attention

We can increase the capacity of such layers by supplying h triplets of W_k, W_q, W_v.

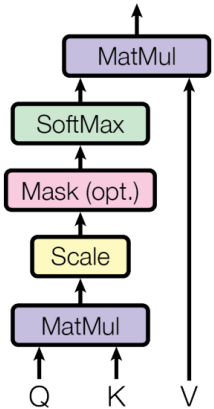
- different triplets lead to different similarities and different pooling sources.
- we denote each of these triplets as an attention head.
- if we wish that the output Z has the same shape as input X, we choose $h = D/D'$.
- if we assume W_k and W_q with binary columns, we come close to grouped convolutions where output maps perceive only a subset of all input maps.
- if we assume fixed inter-token weights, we approach depthwise-separable convolution



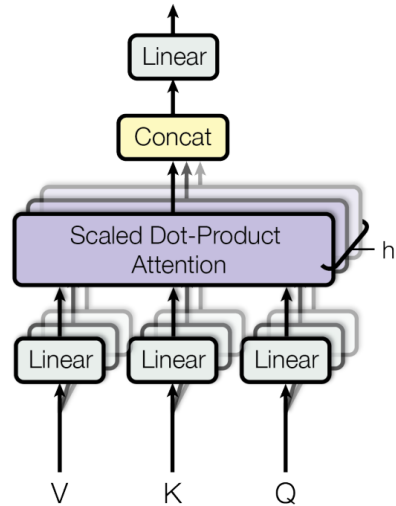
[sandler18cvpr]

NLP: Attention is all you need

Scaled Dot-Product Attention

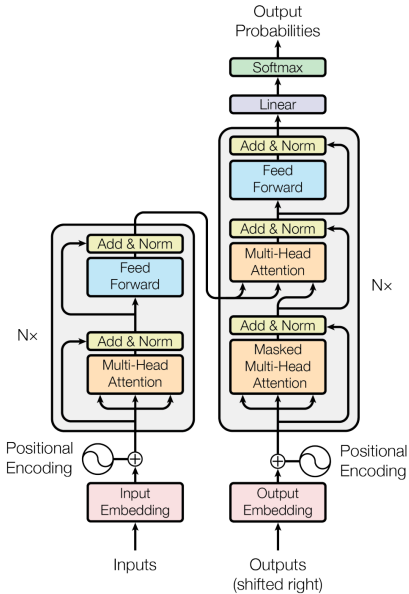


Multi-Head Attention



[vaswani17nips]

NLP: Attention is all you need (2)

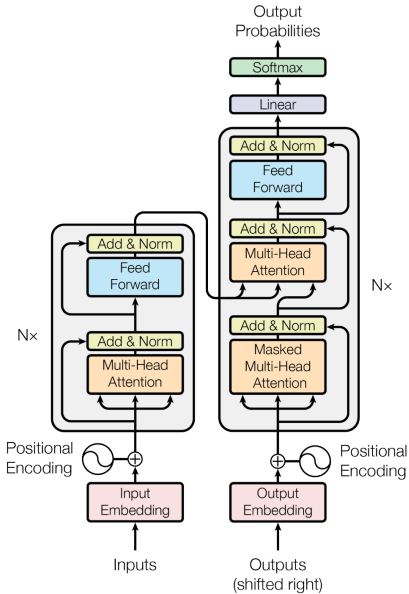


Transformer architecture:

- encoder and decoder inputs (word embeddings) are extended with positional encoding
- encoder with N encoding modules
  - reads the whole sentence in one go
- decoder with N decoding modules
  - infers autoregressively for each output word
  - trains in parallel on whole sentences (!)
- linear projection + softmax

Positional codes are non-optional: if omitted, the model will be invariant to permutation of input tokens.

NLP: Attention is all you need (3)



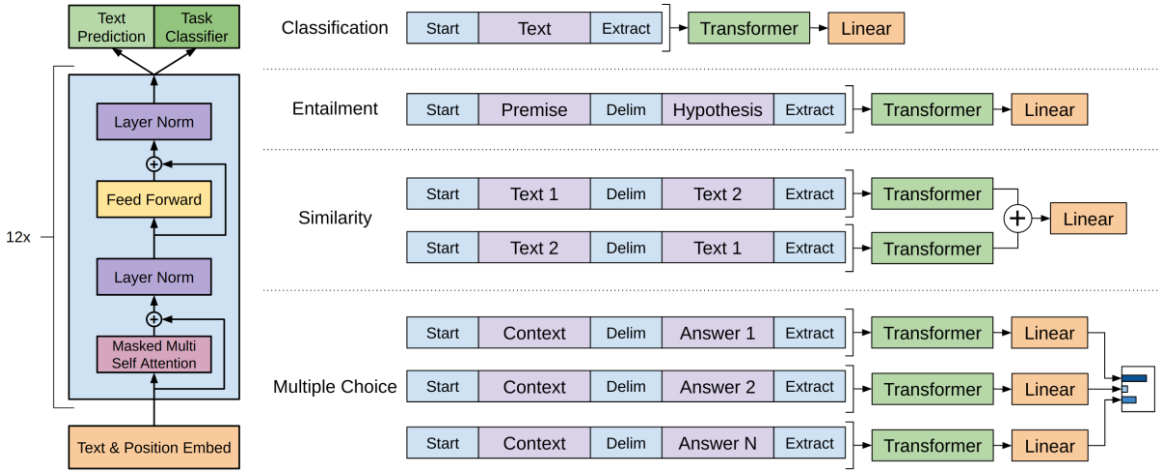
Encoder modules consist of:

- multi-head attention
- layer normalization + residual connection
- fully connected module
- layer normalization + residual connection

Decoder modules consist of:

- masked MHA + layernorm + residual
  - masking allows training on whole sentences
- multi-head cross-attention wrt encoder + layernorm + residual
- FC module + layernorm + residual

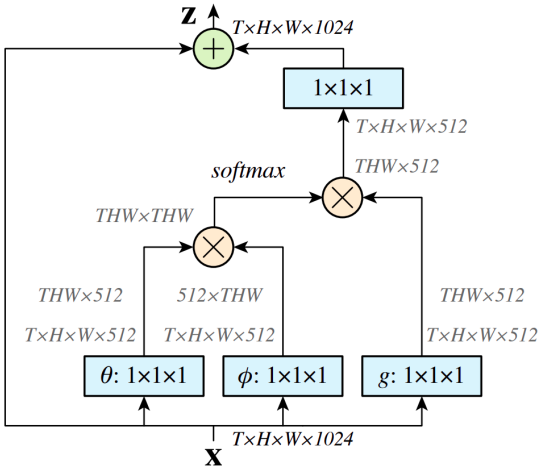
NLP: Generative Pre-trained Transformer



[radford18openai]

Vision: Non-local connections

Some convolutional models capture **long-range** dependencies through attention.



[wang18cvpr]

Input: abstract representation X

- 4th-o tensor T$\times$H$\times$W$\times$1024
- can be viewed as THW$\times$1024.
- H - height, W - width, T - time

Output: representation Z with improved long-distance connectivity

Input X is projected onto queries (θ), keys (ϕ) and values (g).

Vision: Non-local connections (2)

Each spatio-temporal feature $x_i \in R^{1024}$ is both a query and a key.

The similarity matrix S (THW $\times$ THW) compares queries with values.

The similarity matrix again can be obtained through matrix multiplication:

$$\begin{aligned} S &= (W_\theta X^\top)^\top \cdot (W_\phi X^\top), \\ &= (X W_\theta^\top) \cdot (W_\phi X^\top) \ . \end{aligned}$$

Other formulations of similarity are easily plugged-in.

Vision: Non-local connections (3)

The weight matrix α is again obtained by activating rows of S with softmax.

- s_{ij} reflects similarity of the query $W_\theta x_i$ wrt the value $W_g x_j$

$$\alpha = softmax(S, axis = 1) \ .$$

` Outputs $Z = \{z_i\}$ are again weighted pools of values $V = g(x_i)$:

- of course, the weights correspond to the elements of α

$$z_i = \sum_j \alpha_{ij} \cdot g(x_j) \ .$$

Vision: An Image is Worth 16x16 Words

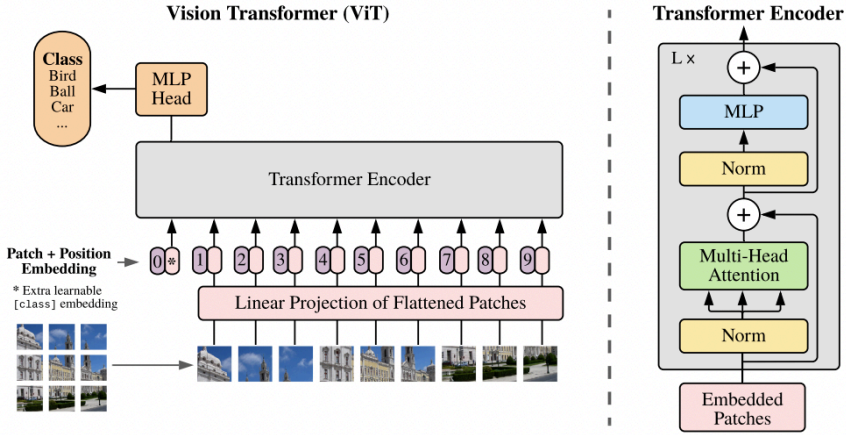
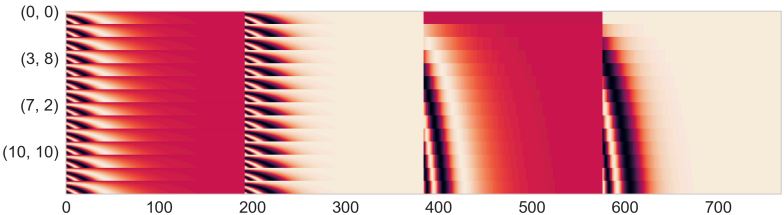


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

Vision: Positional encoding - 2D sin/cos formulation

We encode positions above a 2D grid so that nearby parches receive similar encodings.

Axes locations can be encoded according to the following handcrafted 2D-aware scheme:



$$\begin{cases} \sin\left(\frac{x}{10000^{\frac{i}{4F}}}\right) & i \in [0, \frac{F}{4}[\\ \cos\left(\frac{x}{10000^{\frac{i}{4F}}}\right) & i \in [\frac{F}{4}, \frac{F}{2}[\\ \sin\left(\frac{y}{10000^{\frac{i}{4F}}}\right) & i \in [\frac{F}{2}, 3\frac{F}{4}[\\ \cos\left(\frac{y}{10000^{\frac{i}{4F}}}\right) & i \in [3\frac{F}{4}, F[\end{cases}$$

Still, learned 1D positional embeddings perform equally well.

Vision: ViT instances

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

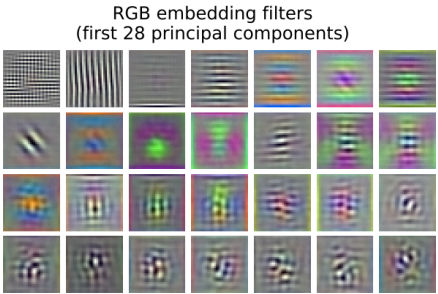
[dosovitskiy20iclr]

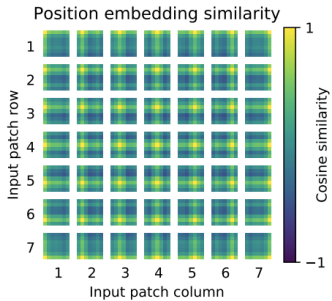
Vision: ViT performance

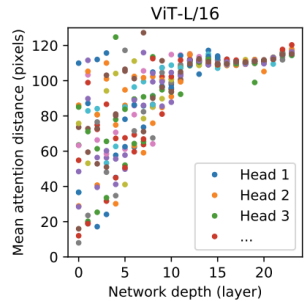
	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 0.06	99.42 $\pm$ 0.03	99.15 $\pm$ 0.03	99.37 $\pm$ 0.06	–
CIFAR-100	**94.55** $\pm$ 0.04	93.90 $\pm$ 0.05	93.25 $\pm$ 0.05	93.51 $\pm$ 0.08	–
Oxford-IIIT Pets	**97.56** $\pm$ 0.03	97.32 $\pm$ 0.11	94.67 $\pm$ 0.15	96.62 $\pm$ 0.23	–
Oxford Flowers-102	99.68 $\pm$ 0.02	**99.74** $\pm$ 0.00	99.61 $\pm$ 0.02	99.63 $\pm$ 0.03	–
VTAB (19 tasks)	**77.63** $\pm$ 0.23	76.28 $\pm$ 0.46	72.72 $\pm$ 0.21	76.29 $\pm$ 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

[dosovitskiy20iclr]

Vision: ViT insight

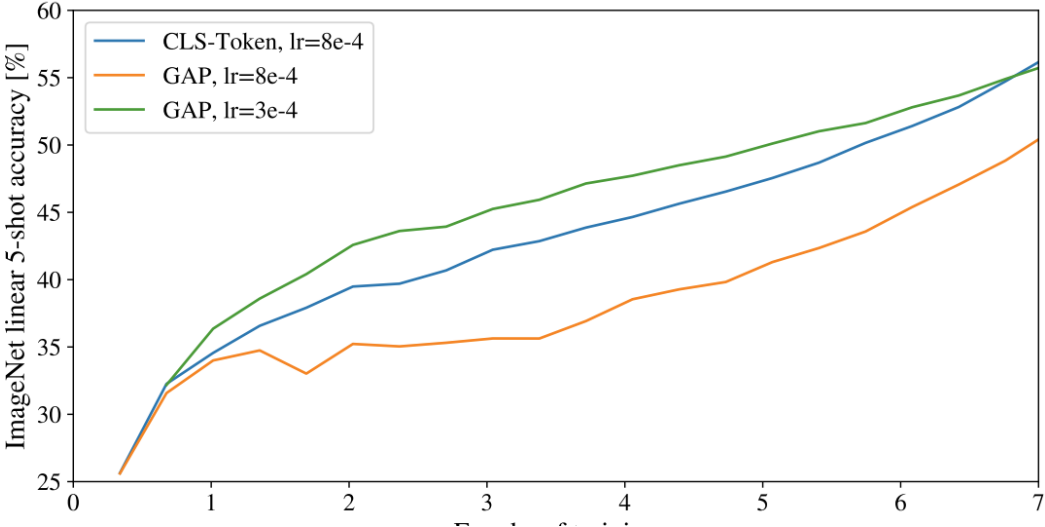






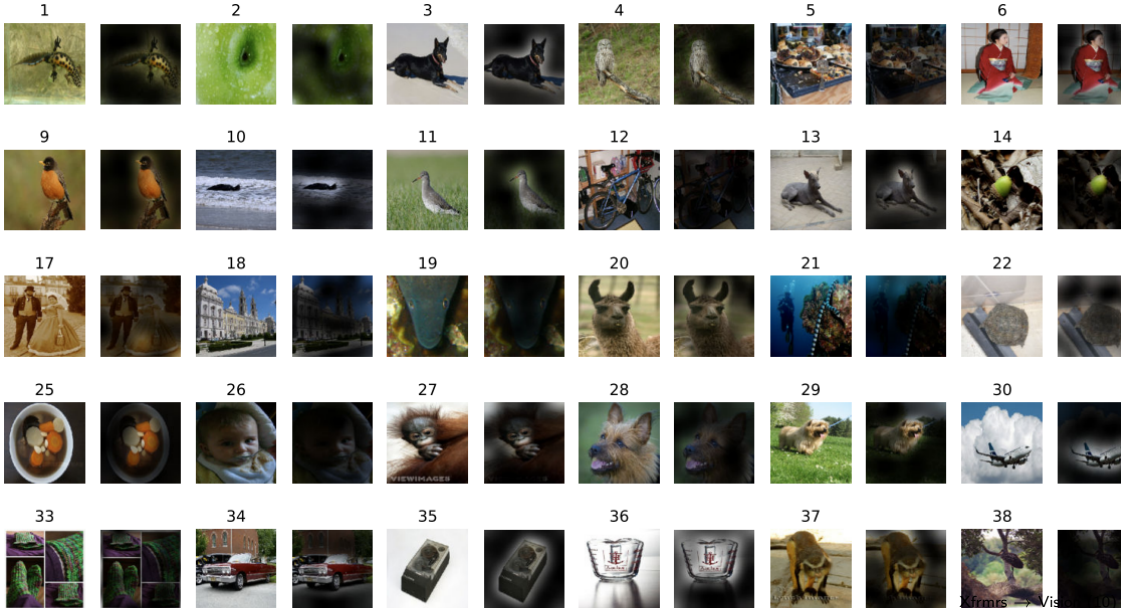
[dosovitskiy20iclr]

Vision: ViT global average pool

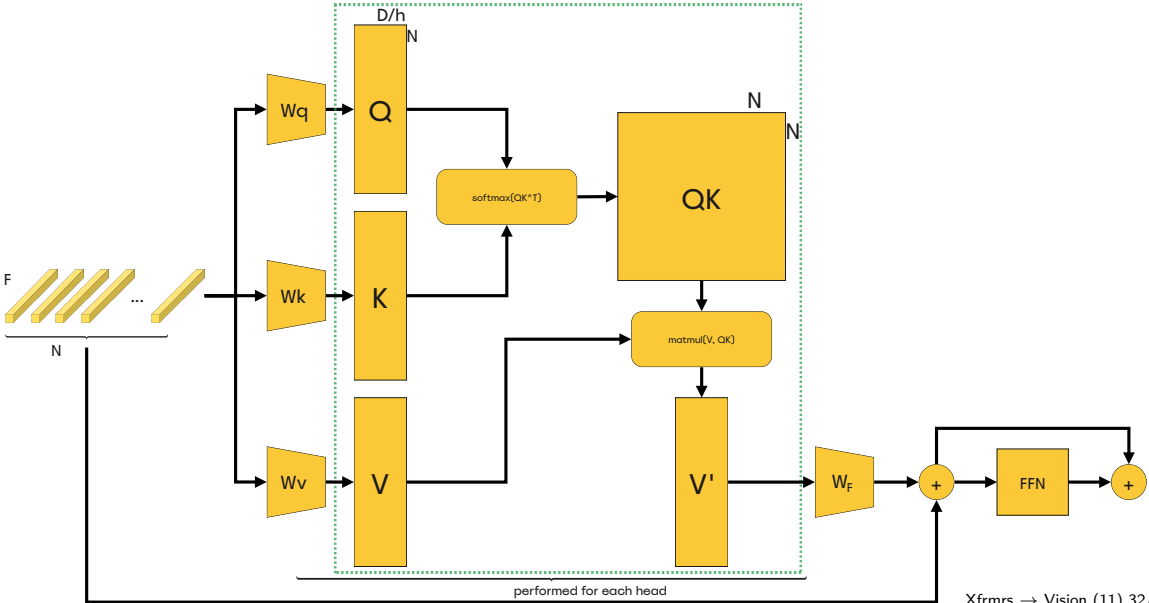


[dosovitskiy20iclr]

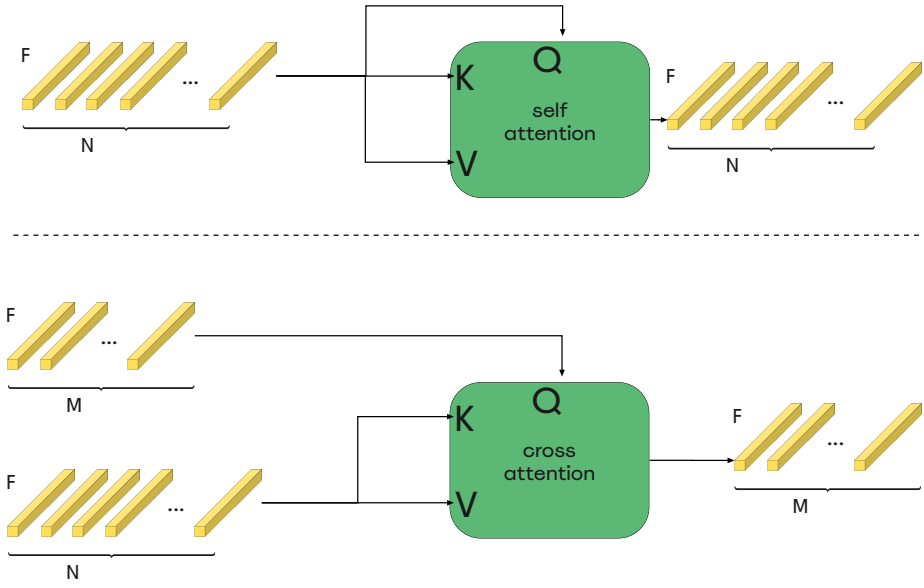
Vision: ViT attention rollout



Vision: attention



Vision: cross-attention

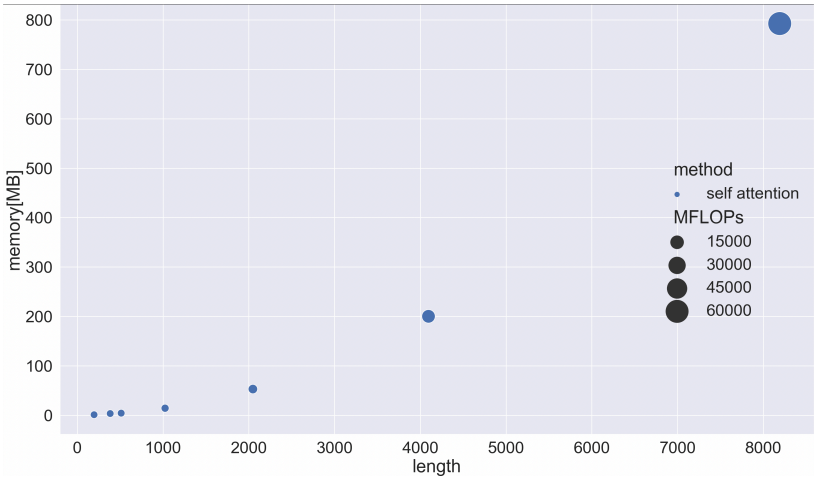


[orsic23fer]

Vision: attention - footprint

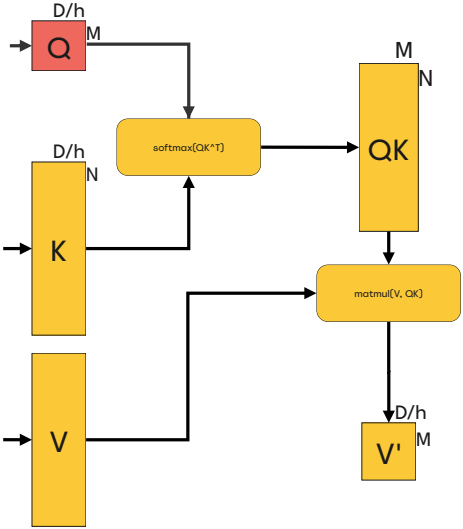
Attention is $\mathcal{O}(n^2)$ in both memory and time.

We wish to scale model inputs for larger image sizes.

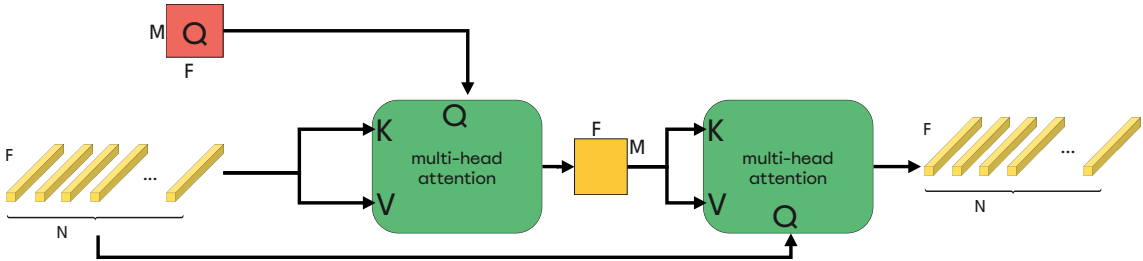


Vision: cross attention

Cross attention is $\mathcal{O}(nm)$ in memory and time.



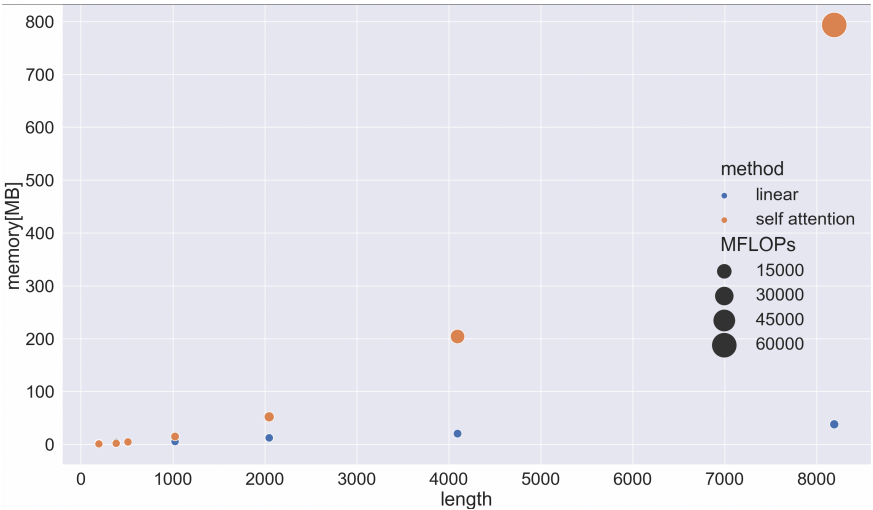
Vision: Learned queries for linear complexity



[orsic23fer]

Vision: cross-attention - footprint

Cross attention is $\mathcal{O}(nm)$ in memory and time.



Vision: Swin Transformer

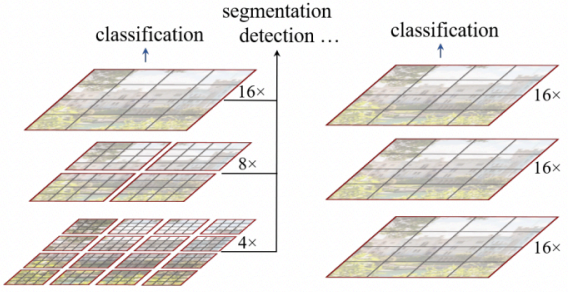


Figure 1. (a) The proposed Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window (shown in red). It can thus serve as a general-purpose backbone for both image classification and dense recognition tasks. (b) In contrast, previous vision Transformers [19] produce feature maps of a single low resolution and have quadratic computation complexity to input image size due to computation of self-attention globally.

Vision: Swin Transformer

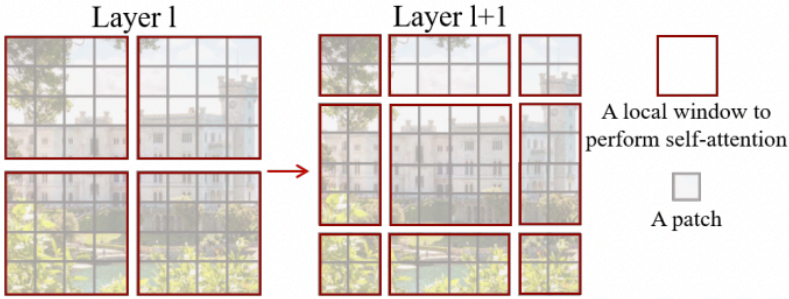


Figure 2. An illustration of the *shifted window* approach for computing self-attention in the proposed Swin Transformer architecture. In layer l (left), a regular window partitioning scheme is adopted, and self-attention is computed within each window. In the next layer $l + 1$ (right), the window partitioning is shifted, resulting in new windows. The self-attention computation in the new windows crosses the boundaries of the previous windows in layer l, providing connections among them.

Vision: Swin Transformer

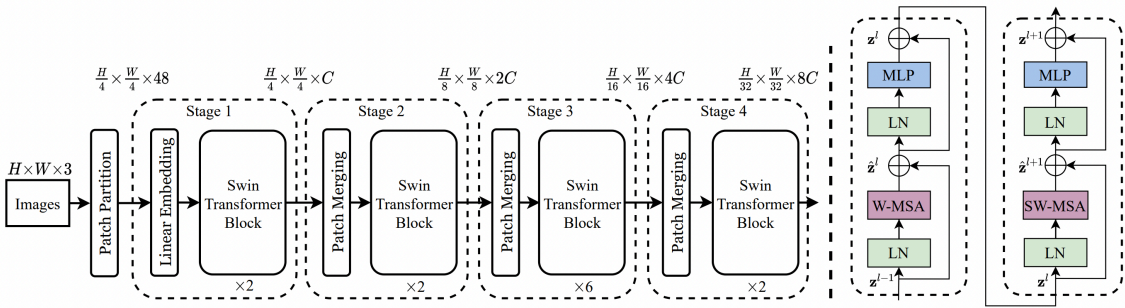


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

[liu21iccv]

Vision: ConvNeXt

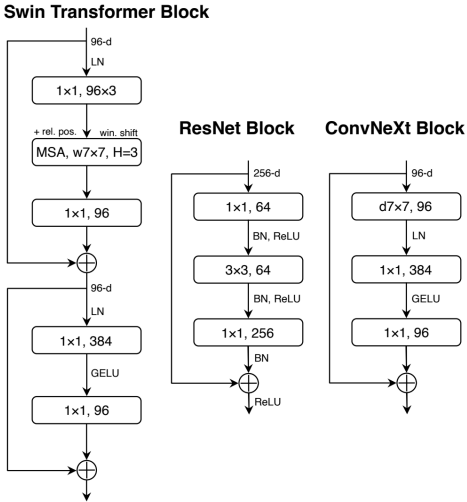


Figure 4. **Block designs** for a ResNet, a Swin Transformer, and a ConvNeXt. Swin Transformer's block is more sophisticated due to the presence of multiple specialized modules and two residual connections. For simplicity, we note the linear layers in Transformer MLP blocks also as "1×1 convs" since they are equivalent.

Changes wrt ResNet:

- stem → patchify
- compute ratio: 3-4-6-3 → 3-3-9-3
- depthwise separable placement
- inverted residuals,
- larger convolutional kernels.

Vision: ViT vs Swin vs ConvNext

(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [44]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [44]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [44]	224^2	84M	16.0G	334.7	82.9
ViT-B/16 [19]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [19]	384^2	307M	190.7G	27.3	76.5
DeiT-S [57]	224^2	22M	4.6G	940.4	79.8
DeiT-B [57]	224^2	86M	17.5G	292.3	81.8
DeiT-B [57]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

model	image size	#param.	FLOPs	throughput (image / s)	IN-1K top-1 acc.
ImageNet-1K trained models					
• RegNetY-16G [54]	224^2	84M	16.0G	334.7	82.9
• EffNet-B7 [71]	600^2	66M	37.0G	55.1	84.3
• EffNetV2-L [72]	480^2	120M	53.0G	83.7	85.7
◦ DeiT-S [73]	224^2	22M	4.6G	978.5	79.8
◦ DeiT-B [73]	224^2	87M	17.6G	302.1	81.8
◦ Swin-T	224^2	28M	4.5G	757.9	81.3
• ConvNeXt-T	224^2	29M	4.5G	774.7	**82.1**
◦ Swin-S	224^2	50M	8.7G	436.7	83.0
• ConvNeXt-S	224^2	50M	8.7G	447.1	**83.1**
◦ Swin-B	224^2	88M	15.4G	286.6	83.5
• ConvNeXt-B	224^2	89M	15.4G	292.1	**83.8**
◦ Swin-B	384^2	88M	47.1G	85.1	84.5
• ConvNeXt-B	384^2	89M	45.0G	95.7	**85.1**
• ConvNeXt-L	224^2	198M	34.4G	146.8	**84.3**
• ConvNeXt-L	384^2	198M	101.0G	50.4	**85.5**

[liu21iccv,liu22cvpr]

SELF-SUPERVISED: INTRO

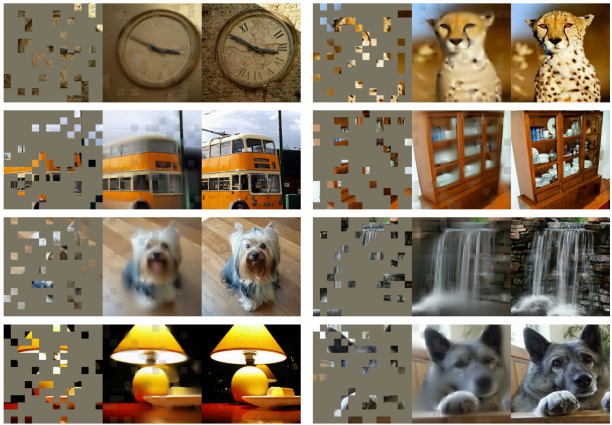
Self-supervised learning learns from unlabeleded data by leveraging some surogate loss.

ViT models have two main architectural differences wrt convnets:

- there is no information “leak” between neighbours
- theoretical receptive field is global.

These properties are very useful for obtaining unsupervised representations.

SELF-SUPERVISED: MASKED AUTOENCODERS



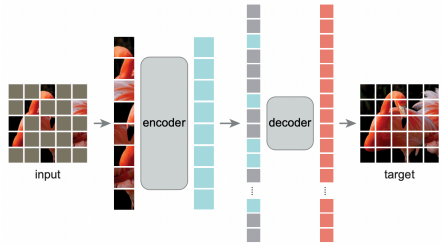


Figure 1. **Our MAE architecture**. During pre-training, a large random subset of image patches (*e.g.*, 75%) is masked out. The encoder is applied to the small subset of *visible patches*. Mask tokens are introduced *after* the encoder, and the full set of encoded patches and mask tokens is processed by a small decoder that reconstructs the original image in pixels. After pre-training, the decoder is discarded and the encoder is applied to uncorrupted images (full sets of patches) for recognition tasks.

[he22cvpr]

SELF-SUPERVISED: MASKED SIAMESE NETWORKS

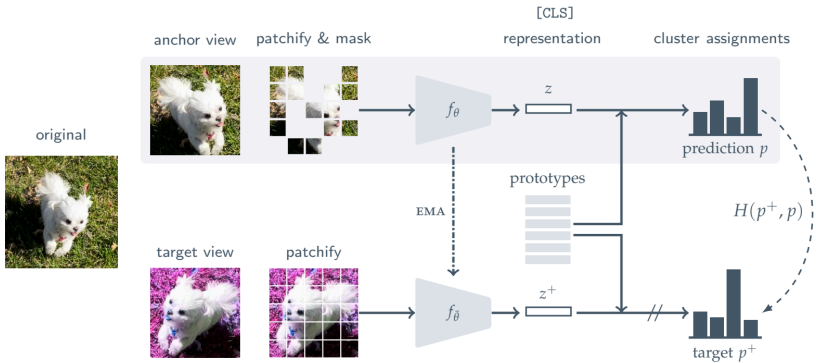


Fig. 1: **Masked Siamese Networks.** First use random data augmentations to generate two views of an image, referred to as the anchor view and the target view. Subsequently, a random mask is applied to the anchor view, while the target view is left unchanged. The objective is then to assign the representation of the masked anchor view to the same clusters as the representation of the unmasked target view. A standard cross-entropy loss is used as the criterion to optimize.

[assran22eccv]

SELF-SUPERVISED: MASKED SIAMESE NETWORKS (2)

Table 3: Linear evaluation on ImageNet-1K using 100% of the labels.

Method	Architecture	Params.	Epochs	Top 1
Comparing similar architectures				
SimCLRv2 [14]	RN50	24M	800	71.7
BYOL [25]	RN50	24M	1000	74.4
DINO [11]	ViT-S/16	22M	800	77.0
iBOT [61]	ViT-S/16	22M	800	**77.9**
MSN	ViT-S/16	22M	600	76.9
Comparing larger architectures				
MAE [27]	ViT-H/14	632M	1600	76.6
BYOL [25]	RN200 (2×)	250M	800	79.6
SimCLRv2 [14]	RN151+SK (3×)	795M	800	79.8
iBOT [61]	ViT-B/16	86M	400	79.4
DINO [11]	ViT-B/8	86M	300	80.1
MoCov3 [16]	ViT-BN-L/7	304M	300	**81.0**
MSN	ViT-L/7	304M	200	**80.7**

Table 1: **Extreme low-shot.** We evaluate the label-efficiency of self-supervised models pretrained on the ImageNet-1K dataset. For evaluation, we use an extremely small number of the ImageNet-1K labels and report the mean top-1 accuracy and standard deviation across 3 random splits of the data.

			Images per Class		
Method	**Architecture**	**Epochs**	1	2	5
iBOT [61]	ViT-S/16	800	40.4 ± 0.5	50.8 ± 0.8	59.9 ± 0.2
	ViT-B/16	400	46.1 ± 0.3	56.2 ± 0.7	64.7 ± 0.3
DINO [11]	ViT-S/16	800	38.9 ± 0.4	48.9 ± 0.3	58.5 ± 0.1
	ViT-B/16	400	41.8 ± 0.3	51.9 ± 0.6	61.4 ± 0.2
	ViT-S/8	800	45.5 ± 0.4	56.0 ± 0.7	64.7 ± 0.4
	ViT-B/8	300	45.8 ± 0.5	55.9 ± 0.6	64.6 ± 0.2
MAE [27]	ViT-B/16	1600	8.2 ± 0.3	25.0 ± 0.3	40.5 ± 0.2
	ViT-L/16	1600	12.3 ± 0.2	19.3 ± 1.8	42.3 ± 0.3
	ViT-H/14	1600	11.6 ± 0.4	18.6 ± 0.2	32.8 ± 0.2
MSN (Ours)	ViT-S/16	800	47.1 ± 0.1	55.8 ± 0.6	62.8 ± 0.3
	ViT-B/16	600	49.8 ± 0.2	58.9 ± 0.4	65.5 ± 0.3
	ViT-B/8	600	55.1 ± 0.1	64.9 ± 0.7	71.6 ± 0.3
	ViT-L/7	200	**57.1 ± 0.6**	**66.4 ± 0.6**	**72.1 ± 0.2**

[assran22eccv

PROPERTIES: ROBUSTNESS

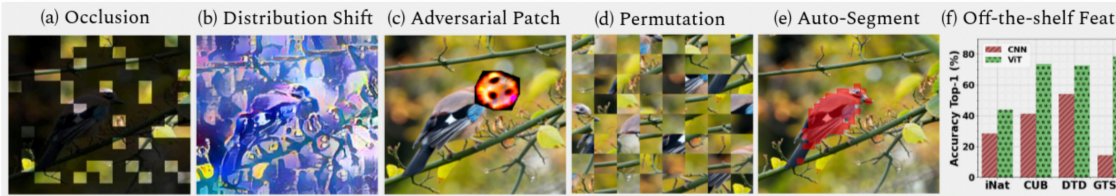
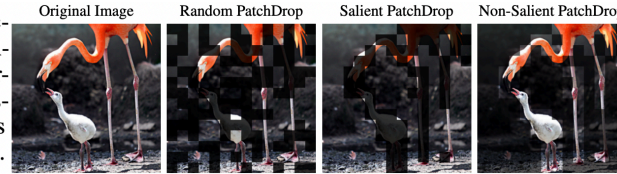


Figure 1: We show intriguing properties of ViT including impressive robustness to `(a)` severe occlusions, `(b)` distributional shifts (*e.g.,* stylization to remove texture cues), `(c)` adversarial perturbations, and `(d)` patch permutations. Furthermore, our ViT models trained to focus on shape cues can segment foregrounds without any pixel-level supervision `(e)`. Finally, off-the-shelf features from ViT models generalize better than CNNs `(f)`.

[naseer21neurips]

PROPERTIES: OCCLUSION

Figure 2: An example image with its occluded versions (Random, Salient and Non-Salient). The occluded images are correctly classified by Deit-S [3] but misclassified by ResNet50 [28]. Pixel values in occluded (black) regions are set to zero.



[naseer21neurips]

PROPERTIES: OCCLUSION (2)

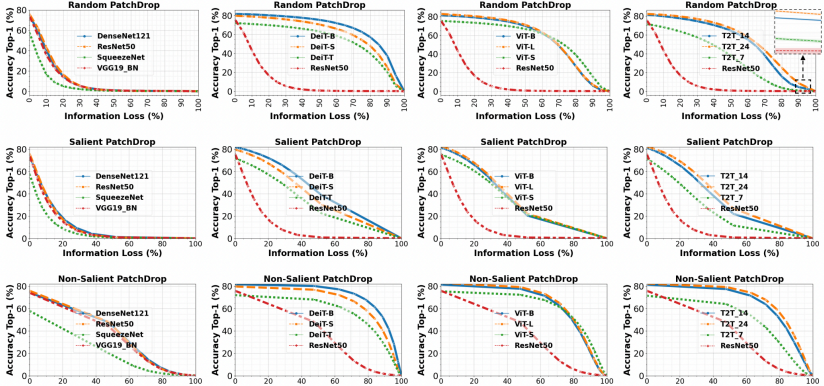


Figure 3: Robustness against object occlusion in images is studied under three PatchDrop settings (see Sec 3.1). (*left*) We study the robustness of CNN models to occlusions, and identify ResNet50 as a strong baseline. (*mid-left*) We compare the DeiT model family against ResNet50 exhibiting their superior robustness to object occlusion. (*mid-right*) Comparison against ViT model family. (*right*) Comparison against T2T model family.

[naseer21neurips]

PROPERTIES: SHUFFLING

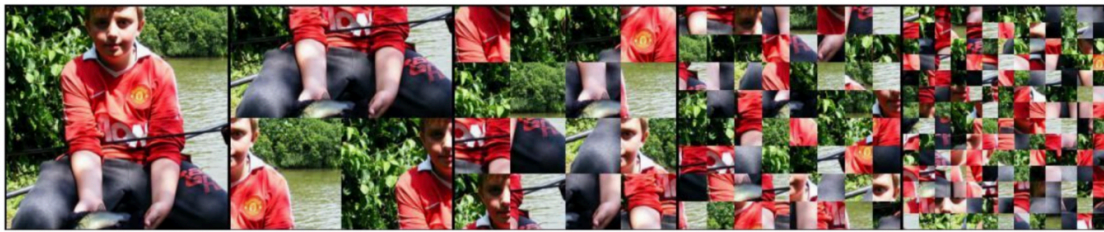


Figure 9: An illustration of shuffle operation applied on images used to eliminate their structural information. (*best viewed zoomed-in*)

[naseer21neurips]

PROPERTIES: SHUFFLING (2)

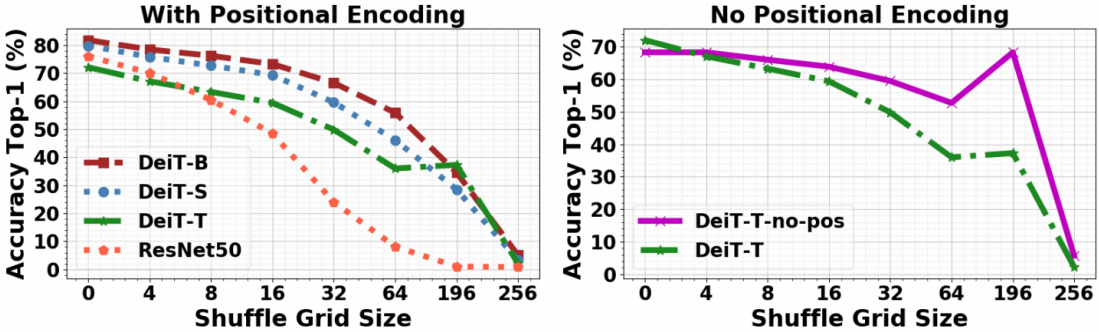


Figure 10: Models trained on 196 image patches. Top-1 (%) accuracy over ImageNet val. set when patches are shuffled. Note the performance peaks when shuffle grid size is equal to the original number of patches used during training, since it equals to only changing the position of input patch (and not disturbing the patch content).

PROPERTIES: IMAGENET-C

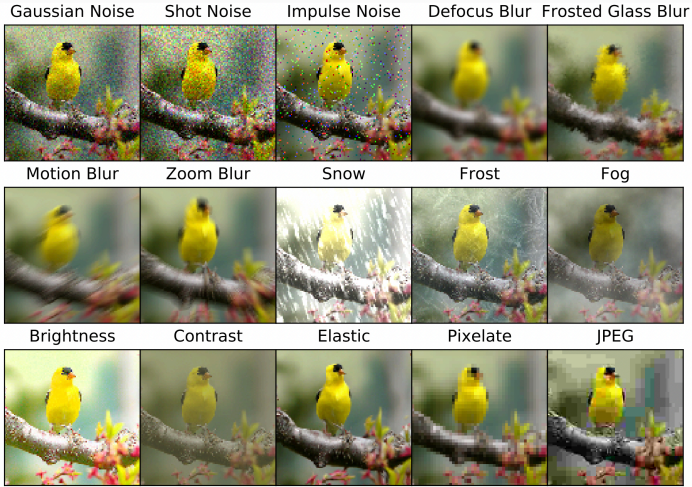


Figure 1: Our IMAGENET-C dataset consists of 15 types of algorithmically generated corruptions from noise, blur, weather, and digital categories. Each type of corruption has five levels of severity, resulting in 75 distinct corruptions. See different severity levels in Appendix B.

[hendrycks19iclr]

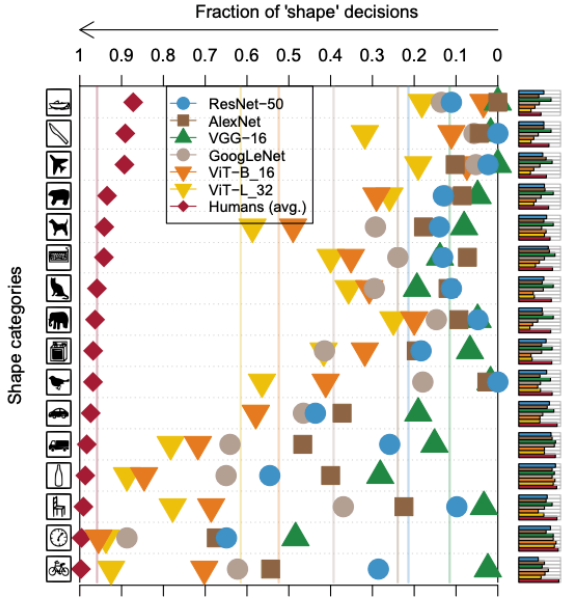
PROPERTIES: MORE ROBUSTNESS

Trained with Augmentations						Trained without Augmentation			
DeiT-B	DeiT-S	DeiT-T	T2T-24	TnT-S	Augmix	ResNet50	ResNet50-SIN	DeiT-T-SIN	DeiT-S-SIN
48.5	54.6	71.1	49.1	53.1	65.3	76.7	77.3	94.4	84.0

Table 4: mean Corruption Error (mCE) across common corruptions [13] (lower the better). While ViTs have better robustness compared to CNNs, training to achieve a higher shape-bias makes both CNNs and ViTs more vulnerable to natural distribution shifts. All models trained with augmentations (ViT or CNN) have lower mCE in comparison to models trained without augmentations on ImageNet or SIN.

[naseer21neurips]

PROPERTIES: BIAS - SHAPE VS TEXTURE



Thank you for your attention!

Questions?

This presentation would not have been possible without insightful ideas and hard work of Matej Grcić, Jakob Verbeek, Ivan Krešo, Marin Oršić, Petra Bevandić, Josip Šarić, Ivan Grubišić, Marin Kačan, Iva Sović, Nenad Markuš and Jelena Bratulić.

This research has been supported by Croatian Science Foundation (MULTICLOD, ADEPT), ERDF (DATACROSS, A-UNIT, SAFETRAM), NVidia Academic Hardware Grant Program, Rimac automobili, Microblink, Gideon brothers, Romb technologies, Promet i prostor, and Končar.