INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

IIIT Hyderabad

Transformers

(continued)

Makarand Tapaswi

CS7.505 Spring 2024

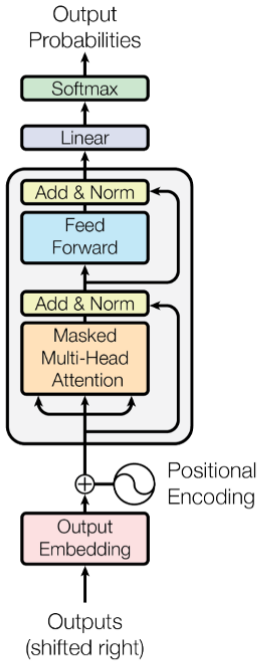
21st February 2024

Practical Transformers: Sequence length

- Batch
- Padding
- Masking

Decoder-only GPT

[sat_]

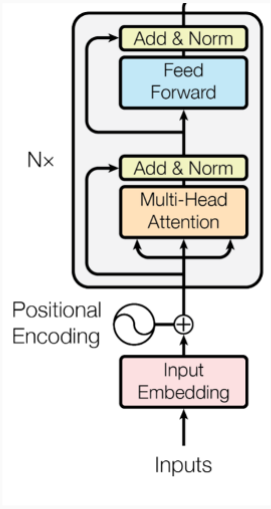


[START] [The_] [cat_]

Transformer image source: "Attention Is All You Need" paper

Encoder-only BERT

[*] [*] **[sat_]** [*] **[the_]** [*]



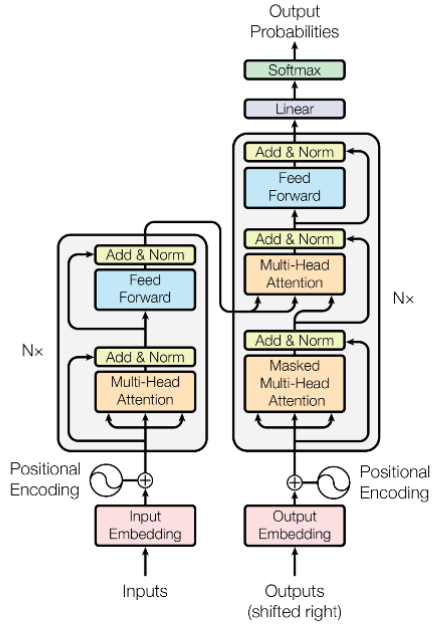
[The_] [cat_] **[MASK]** [on_] **[MASK]** [mat_]

Enc-Dec T5

Das ist gut.

A storm in Attala caused 6 victims.

This is not toxic.



Translate EN-DE: This is good.

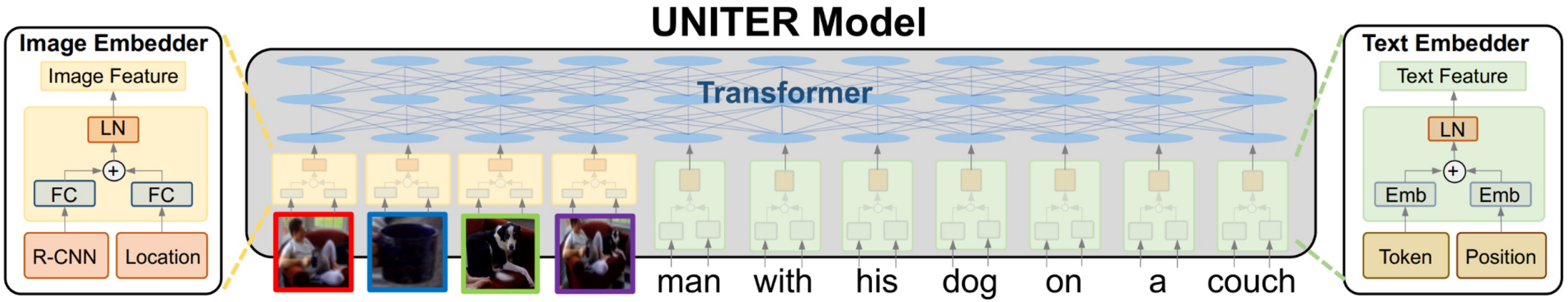
Summarize: state authorities dispatched…

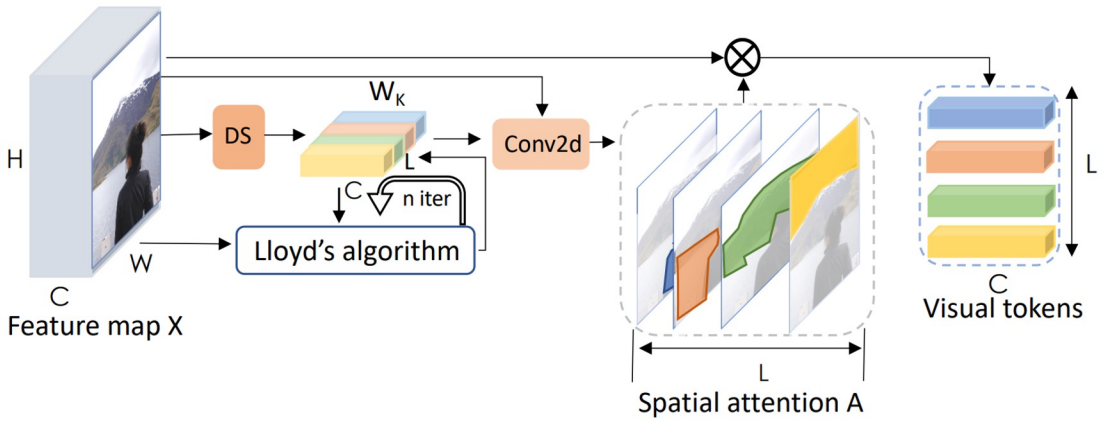
Is this toxic: You look beautiful today!

Vision Transformers

- Pixels as a sequence?
- Problem?
- So, what can we do?

UNITER





https://arxiv.org/abs/2010.11929

ViT: 16x16 patches is “all you need”!

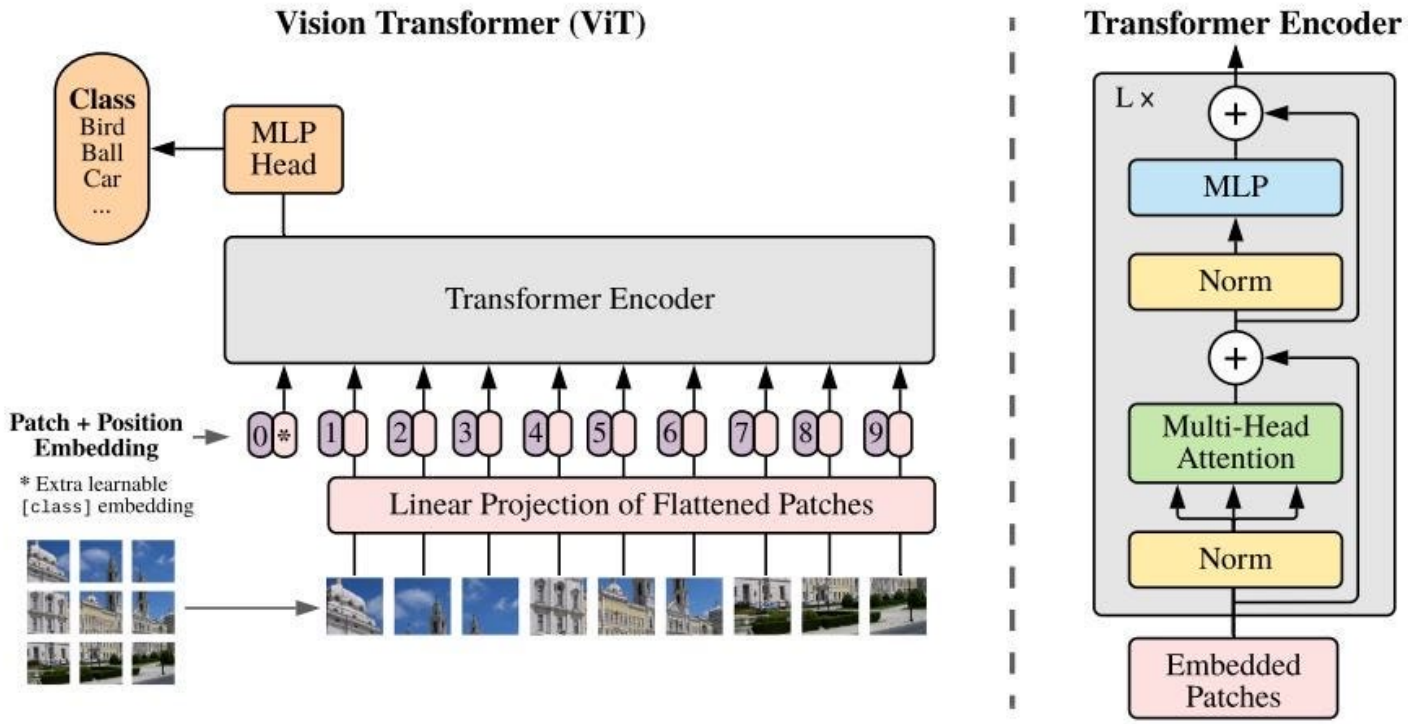


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable “classification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

Position encoding?

- 1d?
- 2d?
- none?

D.4 Positional Embedding

We ran ablations on different ways of encoding spatial information using positional embedding. We tried the following cases:

- Providing no positional information: Considering the inputs as a *bag of patches*.
- 1-dimensional positional embedding: Considering the inputs as a sequence of patches in the raster order (default across all other experiments in this paper).
- 2-dimensional positional embedding: Considering the inputs as a grid of patches in two dimensions. In this case, two sets of embeddings are learned, each for one of the axes, X-embedding, and Y-embedding, each with size $D/2$. Then, based on the coordinate on the path in the input, we concatenate the X and Y embedding to get the final positional embedding for that patch.
- Relative positional embeddings: Considering the relative distance between patches to encode the spatial information as instead of their absolute position. To do so, we use 1-dimensional Relative Attention, in which we define the relative distance all possible pairs of patches. Thus, for every given pair (one as query, and the other as key/value in the attention mechanism), we have an offset $p_q - p_k$, where each offset is associated with an embedding. Then, we simply run extra attention, where we use the original query (the content of query), but use relative positional embeddings as keys. We then use the logits from the relative attention as a bias term and add it to the logits of the main attention (content-based attention) before applying the softmax.

Torch hub!

VisionTransformer

The VisionTransformer model is based on the An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale paper.

Model builders

The following model builders can be used to instantiate a VisionTransformer model, with or without pre-trained weights. All the model builders internally rely on the `torchvision.models.vision_transformer.VisionTransformer` base class. Please refer to the source code for more details about this class.

`vit_b_16`(*[, weights, progress])	Constructs a vit_b_16 architecture from An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.
`vit_b_32`(*[, weights, progress])	Constructs a vit_b_32 architecture from An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.
`vit_l_16`(*[, weights, progress])	Constructs a vit_l_16 architecture from An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.
`vit_l_32`(*[, weights, progress])	Constructs a vit_l_32 architecture from An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.
`vit_h_14`(*[, weights, progress])	Constructs a vit_h_14 architecture from An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

Dino, some beautiful properties!

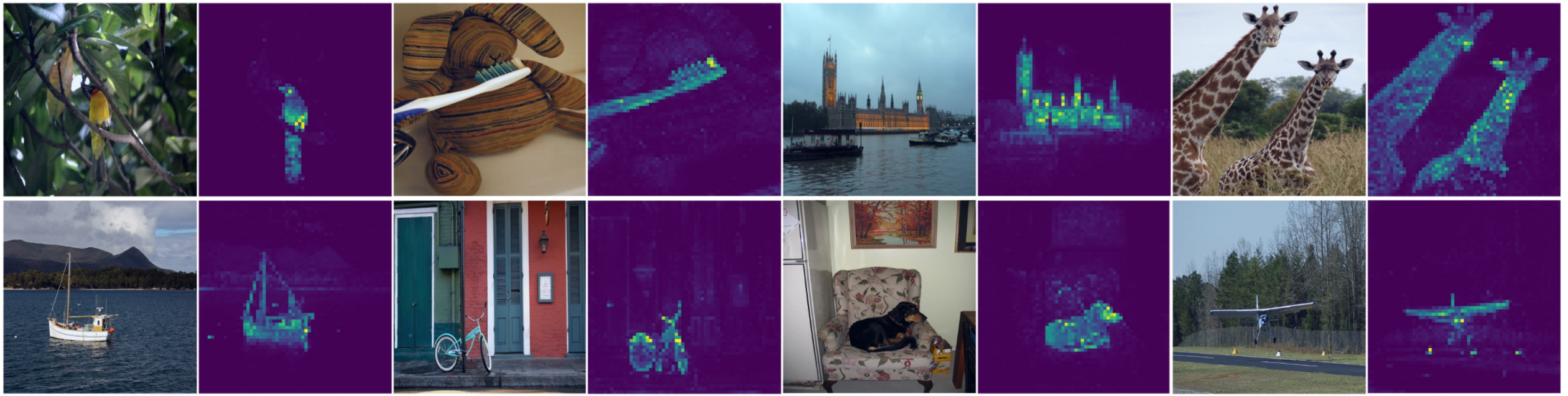


Figure 1: **Self-attention from a Vision Transformer with 8×8 patches trained with no supervision.** We look at the self-attention of the `[CLS]` token on the heads of the last layer. This token is not attached to any label nor supervision. These maps show that the model automatically learns class-specific features leading to unsupervised object segmentations.

https://arxiv.org/abs/2104.14294

Dino, some beautiful properties!

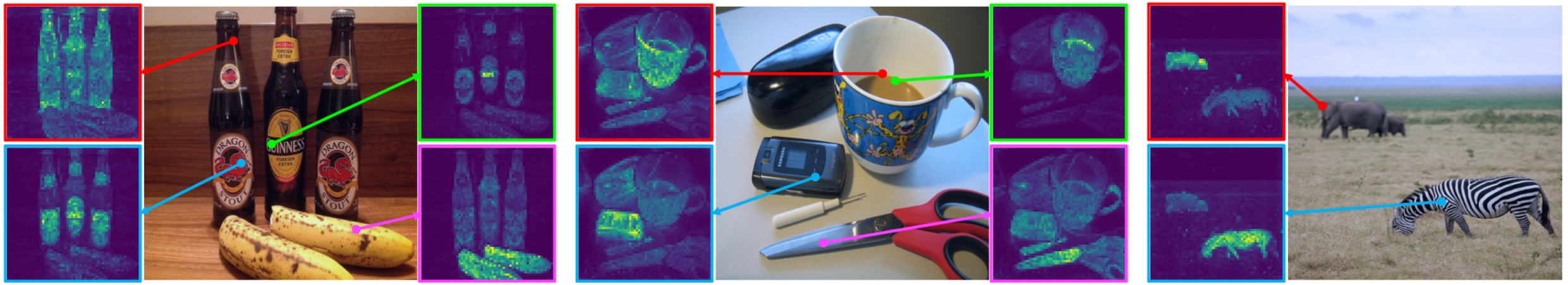
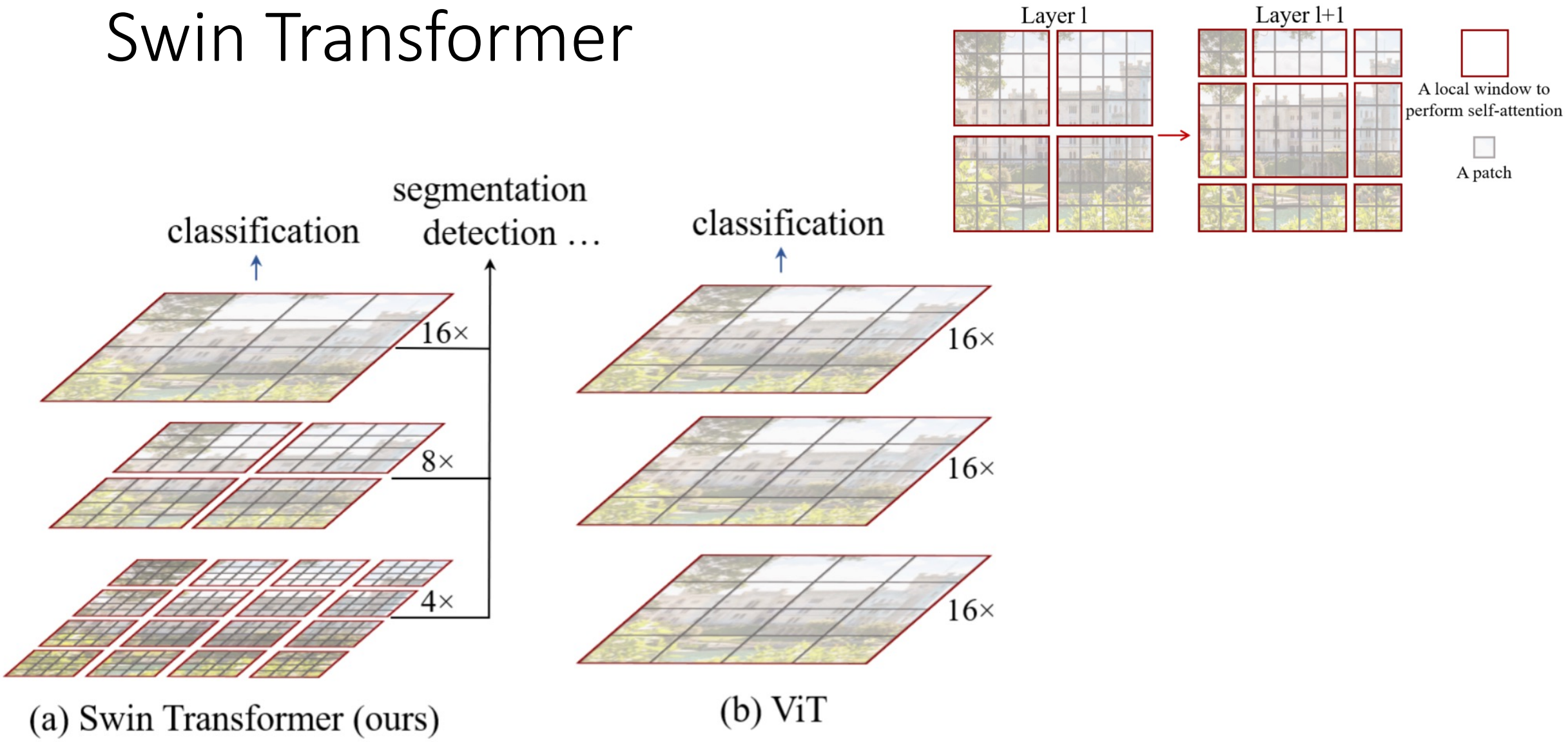


Figure 8: **Self-attention for a set of reference points.** We visualize the self-attention module from the last block of a ViT-S/8 trained with DINO. The network is able to separate objects, though it has been trained with no supervision at all.

https://arxiv.org/abs/2104.14294

Swin Transformer



https://arxiv.org/abs/2103.14030

ViT sizes

Table 2: Model architecture details.

Name	Width	Depth	MLP	Heads	Mio. Param	GFLOPs	
						224^2	384^2
s/28	256	6	1024	8	5.4	0.7	2.0
s/16	256	6	1024	8	5.0	2.2	7.8
S/32	384	12	1536	6	22	2.3	6.9
Ti/16	192	12	768	3	5.5	2.5	9.5
B/32	768	12	3072	12	87	8.7	26.0
S/16	384	12	1536	6	22	9.2	31.2
B/28	768	12	3072	12	87	11.3	30.5
B/16	768	12	3072	12	86	35.1	111.3
L/16	1024	24	4096	16	303	122.9	382.8
g/14	1408	40	6144	16	1011	533.1	1596.4
G/14	1664	48	8192	16	1843	965.3	2859.9

Thank you!