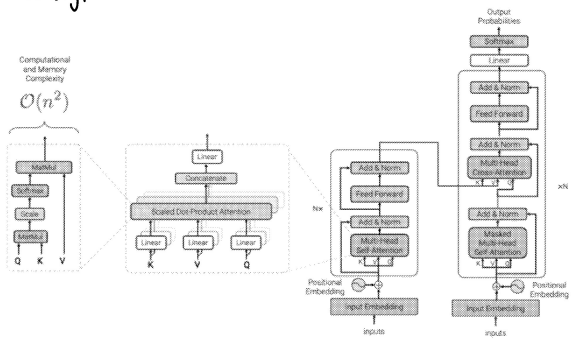
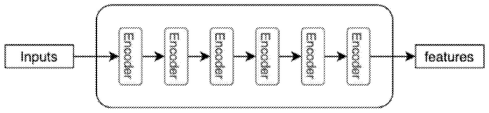
Transformer encoder

Summary:



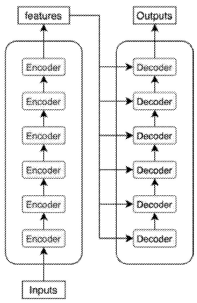
Sequence to sequence mapping

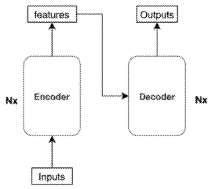
- Overview:
  - The source sequence is a fixed length sequence of embeddings (truncated and zero padded to standard length)
  - The target sequence is a fixed length sequence of embeddings (truncated and zero padded to standard length)
  - Sequences are fed together instead of one token at a time
  - Several **transformer encoder** blocks in sequence encode the input with self-attention to produce features



Sequence to sequence mapping

- Several **transformer decoder** blocks in sequence decode the encoded features with cross-attention between source and target sequences. Each decoder block receives the features from the encoder and an input from the previous decoder block

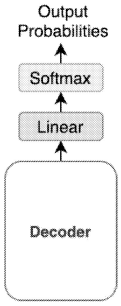




Sequence to sequence mapping

Decoder output:

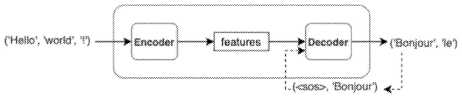
- The transformer decoder produces the output one token at a time in an autoregressive manner
- It starts the generation with a start of sentence token <SOS>
- The output vector from the decoder goes through a linear transformation that changes the dimension of the vector from the embedding vector size (512) into the size of the vocabulary (10,000).
- A softmax layer converts the output vector into 10,000 probabilities.
- Choosing the decoded word can be done by selecting the most probable word. Alternatively, beam search can be performed by looking for the best combination of tokens instead of selecting the most probable token each time.



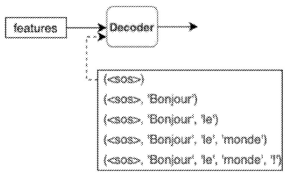
Sequence to sequence mapping

Auto-generative decoding:

- The predicted output is fed back as an input to the decoder to the decoder to produce the next token.

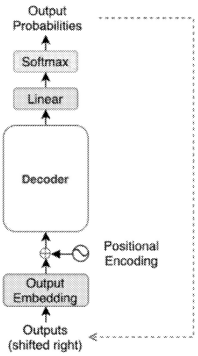


- Th entire list of decoded tokens available is fed at each iteration and the list increases by 1 every iteration.



Sequence to sequence mapping

- The decoder input is of fixed length (in the same way as for the encoder). Thus, the input sequence to the decoder may be zero padded. A mask indicates the location of the zero padded locations so that they could be ignored.
- If the fixed length of the decoded sequence is reached, the outputs are shifted right to discard early decoded word. This is for decoding purpose only. The complete decoded sequence is retained.
- The process repeats until the model predicts the end-of-sentence most probable output.

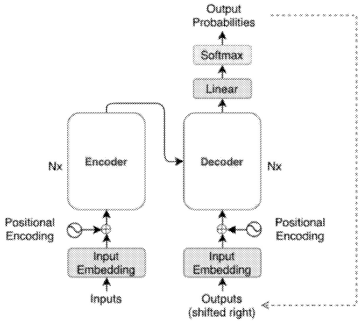


Sequence to sequence mapping

- Teacher forcing:
  - During training, teacher-forcing is used by feeding the expected output instead of the predicted one.
  - During inference, the inferred words are fed (the expected output is not known)

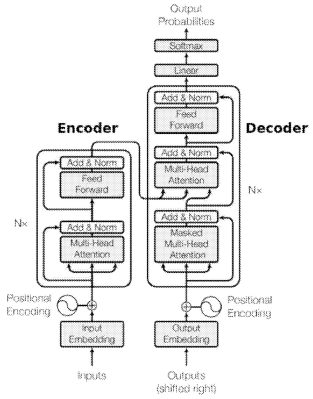
Sequence to sequence mapping

- Architecture summary



Sequence to sequence mapping

- Architecture summary



Vision transformer

An image is worth 16x16 words: Transformers for image recognition

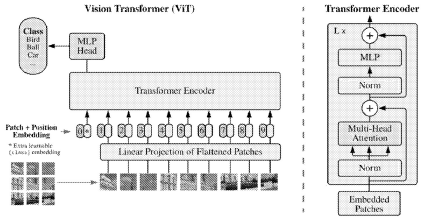


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification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

Vision transformer

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

Table 1: Details of Vision Transformer model variants.

Vision transformer

	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

Table 2: Comparison with state of the art on popular image classification benchmarks. We report mean and standard deviation of the accuracies, averaged over three fine-tuning runs. Vision Transformer models pre-trained on the JFT-300M dataset outperform ResNet-based baselines on all datasets, while taking substantially less computational resources to pre-train. ViT pre-trained on the smaller public ImageNet-21k dataset performs well too. *Slightly improved 88.5% result reported in Touvron et al. (2020).

Vision transformer

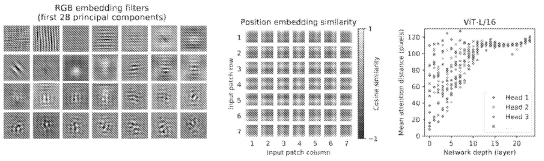


Figure 7: **Left:** Filters of the initial linear embedding of RGB values of ViT-L/32. **Center:** Similarity of position embeddings of ViT-L/32. Tiles show the cosine similarity between the position embedding of the patch with the indicated row and column and the position embeddings of all other patches. **Right:** Size of attended area by head and network depth. Each dot shows the mean attention distance across images for one of 16 heads at one layer. See Appendix D.7 for details.

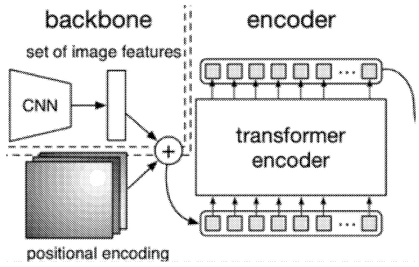
Notes:

- Embedding filters resemble basis functions
- Increased depth corresponds to increased receptive field

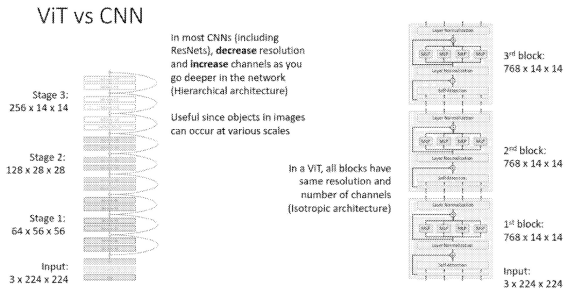
Vision transformer

Hybrid architecture

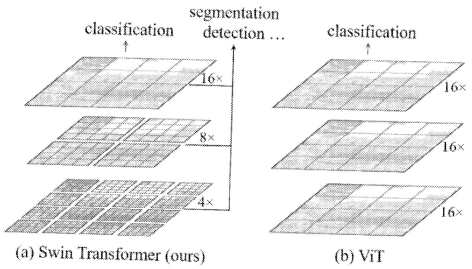
reduced parameters and complexity



Vision transformer

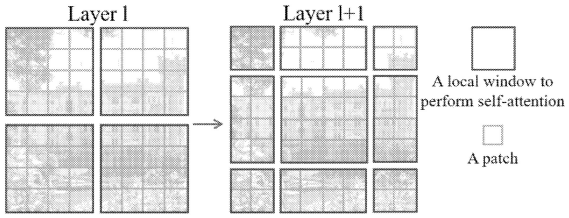


Swin Transformer



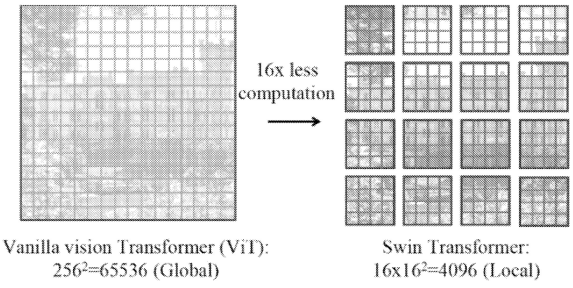
- **Hierarchical vision transformer using shifted windows**
- **Self similarity within patches in a hierarchical manner**

Swin Transformer

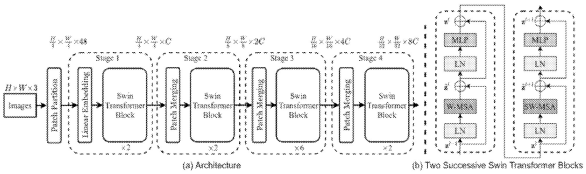


- **Use shifted windows for measuring self similarity**

Swin Transformer



Swin Transformer



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

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.

Swin Transformer

(b) ImageNet-22K pre-trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [34]	384^2	388M	204.6G	-	84.4
R-152x4 [34]	480^2	937M	840.5G	-	85.4
ViT-B/16 [19]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [19]	384^2	307M	190.7G	27.3	85.2
Swin-B	224^2	88M	15.4G	278.1	85.2
Swin-B	384^2	88M	47.0G	84.7	86.4
Swin-L	384^2	197M	103.9G	42.1	87.3

Table 1. Comparison of different backbones on ImageNet-1K classification. Throughput is measured using the GitHub repository of [62] and a V100 GPU, following [57].

Segmentation (SETR)

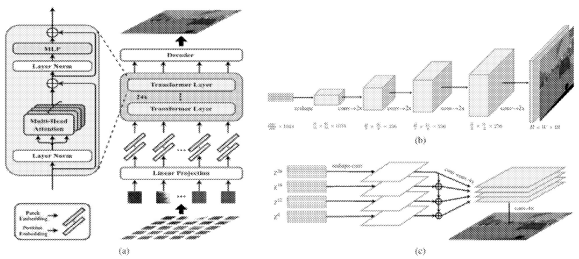


Figure 1. **Schematic illustration of the proposed *SEgmentation TRansformer* (SETR)** (a). We first split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. To perform pixel-wise segmentation, we introduce different decoder designs: (b) progressive upsampling (resulting in a variant called SETR-*PUP*); and (c) multi-level feature aggregation (a variant called SETR-*MLA*).

Detection (DETR)

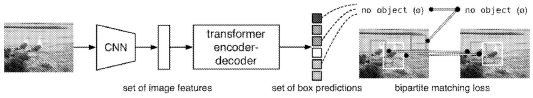


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "`no object`" (∅) class prediction.

Detection (DETR)

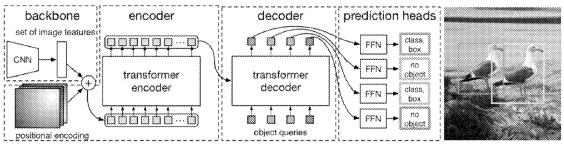


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "`no object`" class.

Detection (DETR)

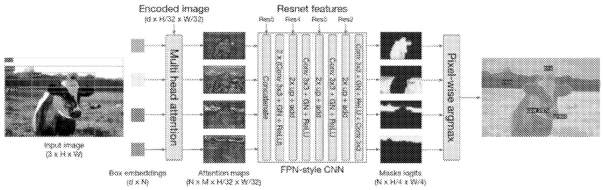


Fig. 8: Illustration of the panoptic head. A binary mask is generated in parallel for each detected object, then the masks are merged using pixel-wise argmax.

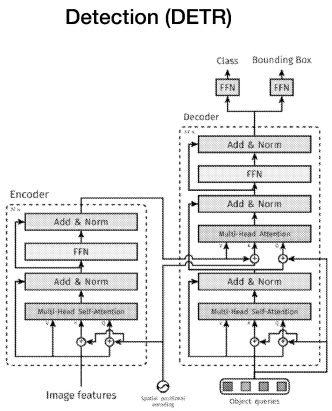


Fig. 10: Architecture of DETR's transformer. Please, see Section A.3 for details.