Transformers

한양대학교

컴퓨터소프트웨어학부

박현준

Contents

ViT : Vision Transformer

DETR : Detection Transformer

Segmenter : Transformer for Semantic Segmentation

ViT
: Vision Transformer

2021 ICLR

Introduction

Transformer

- Computational efficiency
- Scalability (can train over 100B parameters)

➡ Apply Transformer directly to image, with the fewest possible modifications

- Split an image into patches
- Provide sequence of linear embeddings as an input to Transformer

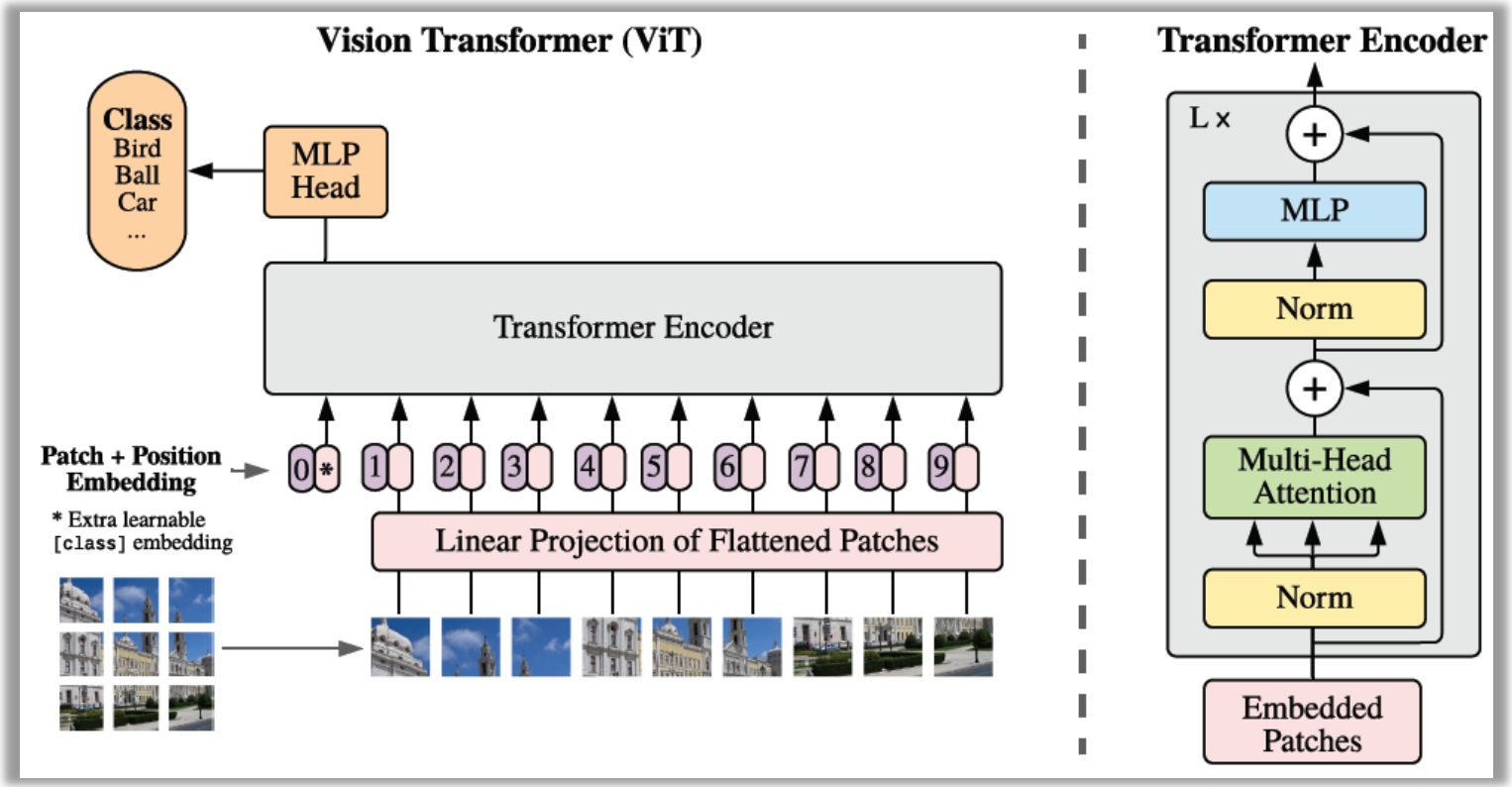
Inductive Bias

CNN : ↑ Inductive bias (Translation equivariance, Locality) ⇒ generalize X when trained on insufficient data

Transformer : lack some of the inductive bias inherent to CNNs

Large scale training trumps inductive bias

Vision Transformer

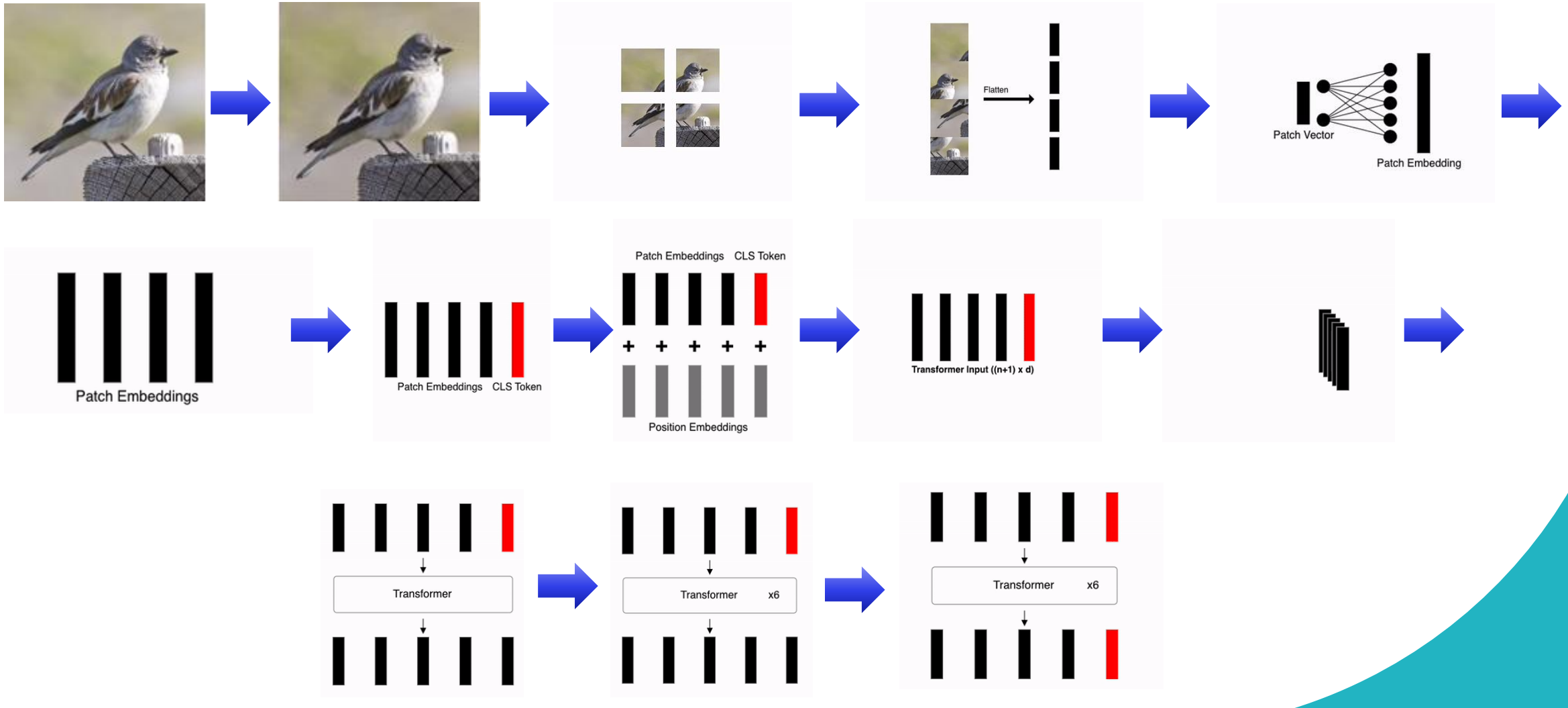


- Reshape the image $x \in \mathbb{R}^{H \times W \times C}$ into a sequence of flattened 2D patches $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$
- Prepend a learnable embedding to the sequence of embedded patches ($z_0^0 = x_{class}$) ≒ BERT $[class]$ token
- Position embeddings are added to the patch embeddings to retain positional information
- Encoder consists of alternating layers of MSA, MLP blocks
- LN is applied before every block, and residual connections after every block

Vision Transformer

Vision Transformer

Procedure



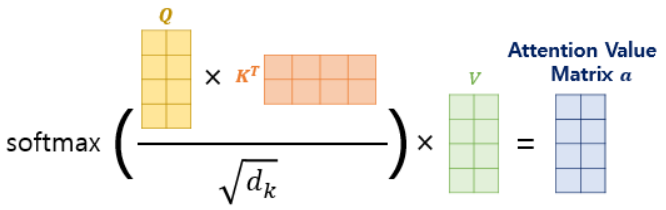
Vision Transformer

MLP

$$z_0 = [x_{class}; x_p^1 E; x_p^2 E; \cdots; x_p^N E] + E_{pos}$$
$$z_l' = MSA(LN(z_{l-1})) + z_{l-1}$$
$$z_l = MLP(LN(z_l')) + z_l'$$
$$y = LN(z_L^0)$$

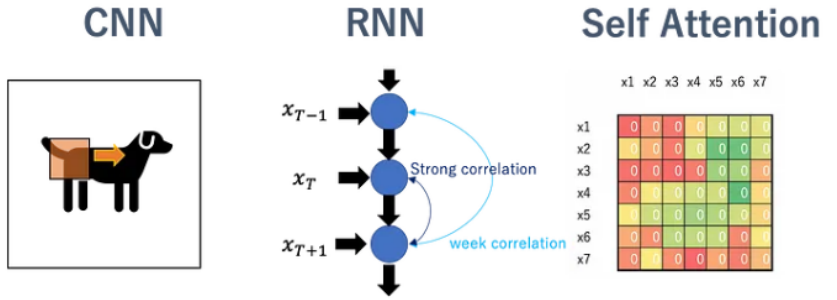
MSA

$$[q, k, v] = z\mathbb{U}_{qkv}$$
$$A = softmax(qk^T/\sqrt{D_h})$$
$$SA(z) = Av$$
$$MSA(z) = [SA_1(z); SA_2(z); \cdots; SA_k(z)]\mathbb{U}_{msa}$$



Inductive bias

Position embeddings at initialization time carry no information about 2D positions of the patches and all spatial relations between the patches have to be learned from scratch



(Left) CNN, which has a strong inductive bias that the information is locally aggregated. (center) RNN, which has a strong inductive bias in that it is strongly correlated with the previous time (right) Self-Attention, which has a relatively weak inductive bias because it only correlates all features.

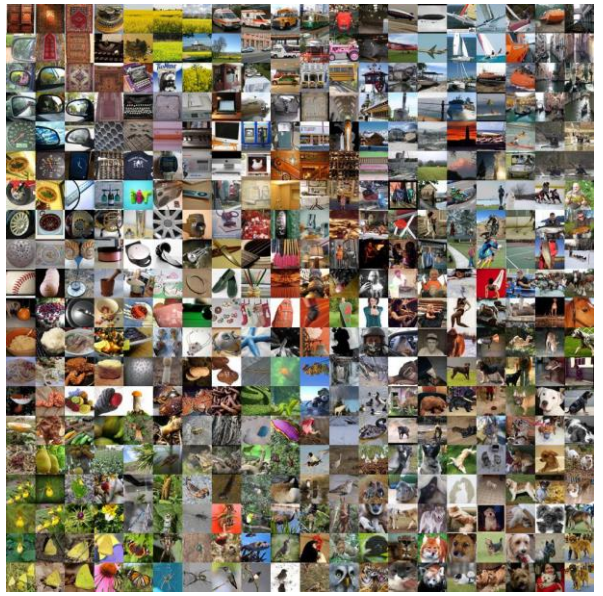
⇒ **Much less image-specific inductive bias than CNN**

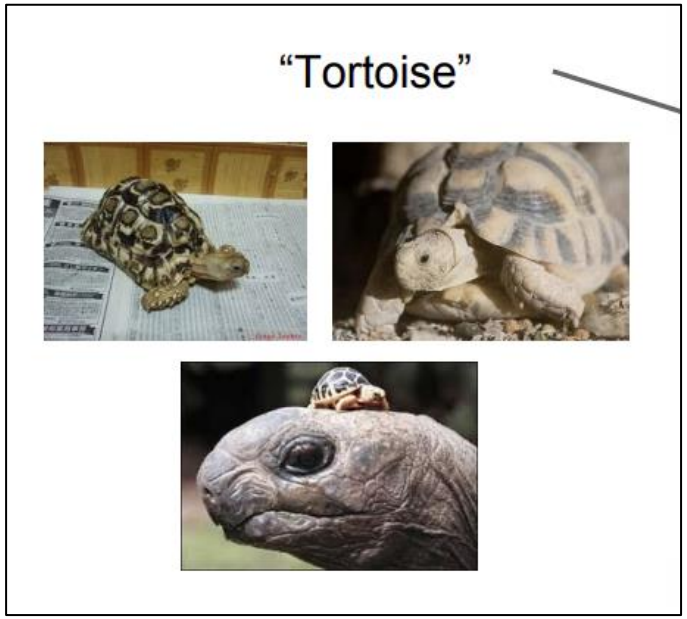
(∵ only MLP layers are local and translational invariant, self-attention layers are global)

Result

Datasets

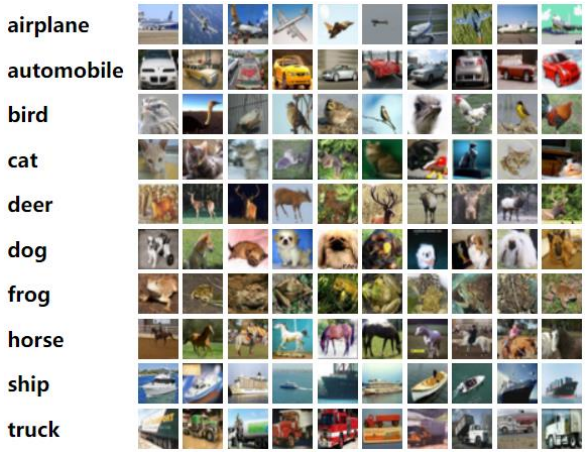
- ILSVRC-2012 : 1k classes and 1.3M images
- ImageNet-21k : 21k classes and 14M images
- JFT : 18k classes and 303M high-resolution images

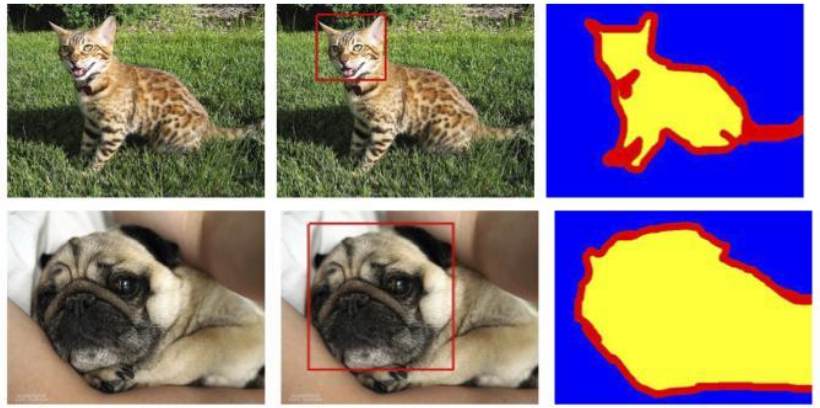




Transfer pre-trained model to several benchmark tasks

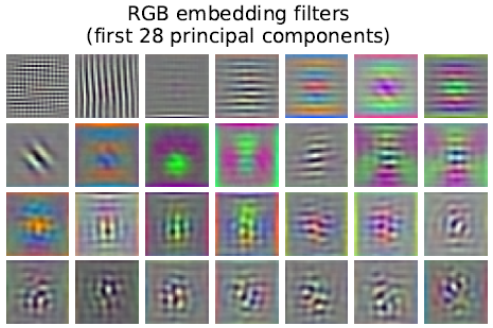
- ImageNet on the original validation labels
- Cleaned-up ReaL labels
- CIFAR-10/100
- Oxford-IIIT Pets
- Oxford Flowers-102

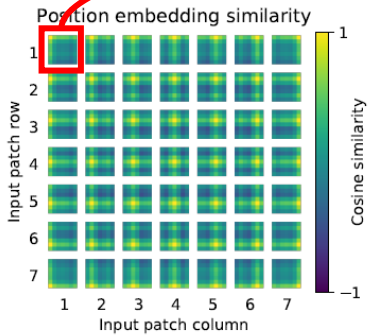


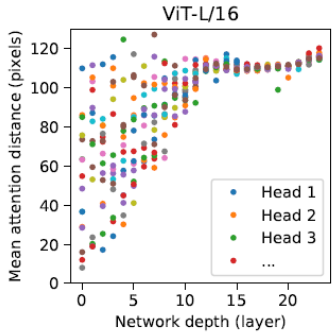


Result

Attention







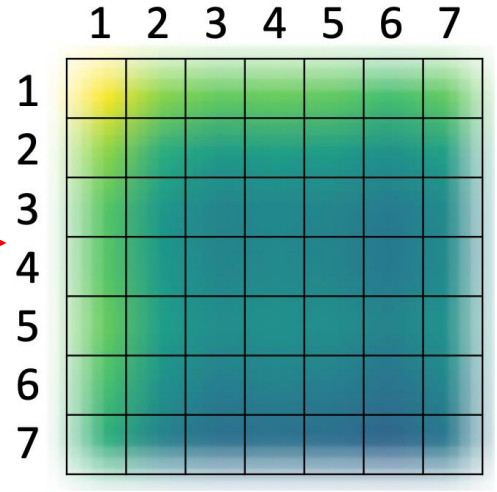
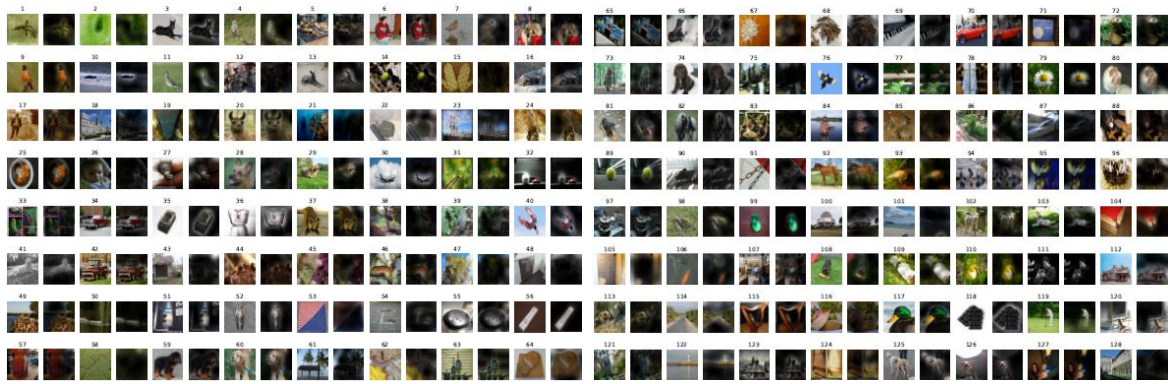
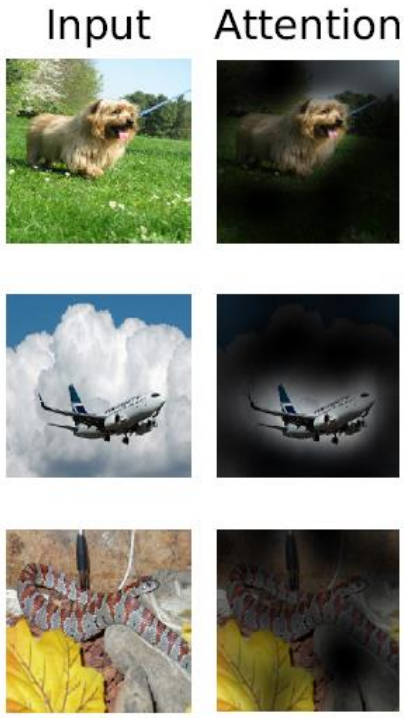


Figure 7: **Left:** Filters of the initial linear embedding of RGB values of ViT-L/32. **Center:** Similarity of position embeddings of ViT-L/32. Tiles show the cosine similarity between the position embedding of the patch with the indicated row and column and the position embeddings of all other patches. **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.





Result

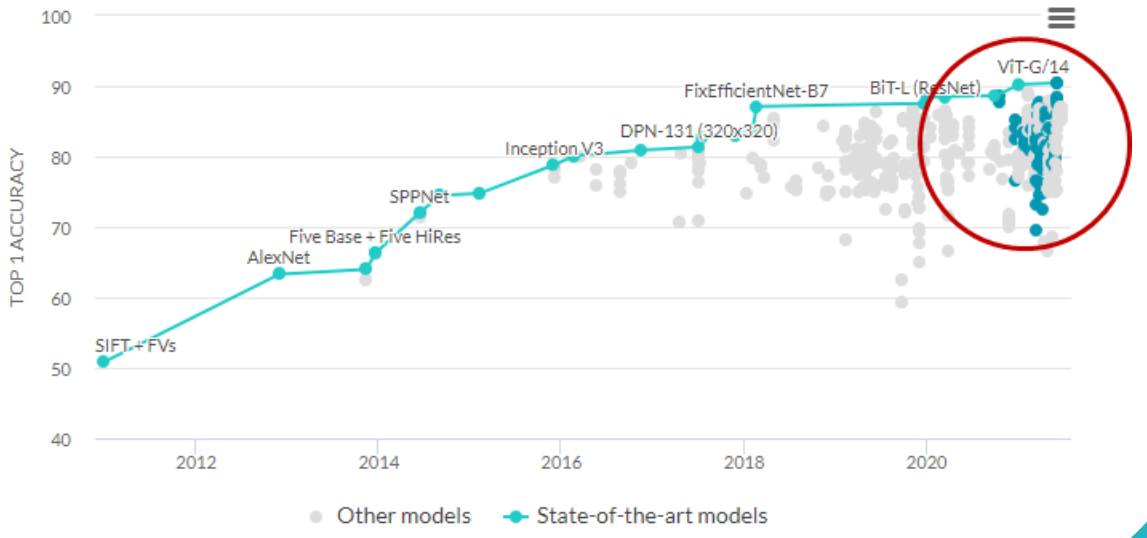
	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15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

Dataset	ResNet50 Adam	ResNet50 SGD	ResNet152x2 Adam	ResNet152x2 SGD
ImageNet	77.54	78.24	84.97	84.37
CIFAR10	97.67	97.46	99.06	99.07
CIFAR100	86.07	85.17	92.05	91.06
Oxford-IIIT Pets	91.11	91.00	95.37	94.79
Oxford Flowers-102	94.26	92.06	98.62	99.32
Average	89.33	88.79	94.01	93.72

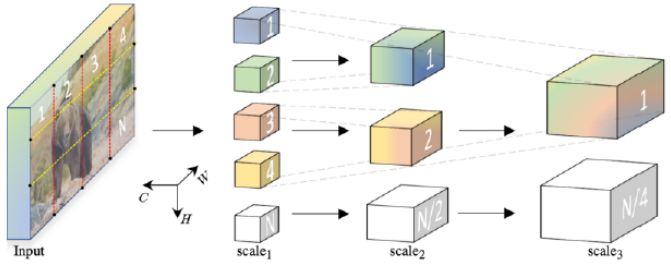
Table 7: Fine-tuning ResNet models pre-trained with Adam and SGD.

		ViT-B/16	ViT-B/32	ViT-L/16	ViT-L/32	ViT-H/14
ImageNet	CIFAR-10	98.13	97.77	97.86	97.94	-
	CIFAR-100	87.13	86.31	86.35	87.07	-
	ImageNet	77.91	73.38	76.53	71.16	-
	ImageNet ReaL	83.57	79.56	82.19	77.83	-
	Oxford Flowers-102	89.49	85.43	89.66	86.36	-
	Oxford-IIIT-Pets	93.81	92.04	93.64	91.35	-
ImageNet-21k	CIFAR-10	98.95	98.79	99.16	99.13	99.27
	CIFAR-100	91.67	91.97	93.44	93.04	93.82
	ImageNet	83.97	81.28	85.15	80.99	85.13
	ImageNet ReaL	88.35	86.63	88.40	85.65	88.70
	Oxford Flowers-102	99.38	99.11	99.61	99.19	99.51
	Oxford-IIIT-Pets	94.43	93.02	94.73	93.09	94.82
JFT-300M	CIFAR-10	99.00	98.61	99.38	99.19	99.50
	CIFAR-100	91.87	90.49	94.04	92.52	94.55
	ImageNet	84.15	80.73	87.12	84.37	88.04
	ImageNet ReaL	88.85	86.27	89.99	88.28	90.33
	Oxford Flowers-102	99.56	99.27	99.56	99.45	99.68
	Oxford-IIIT-Pets	95.80	93.40	97.11	95.83	97.56

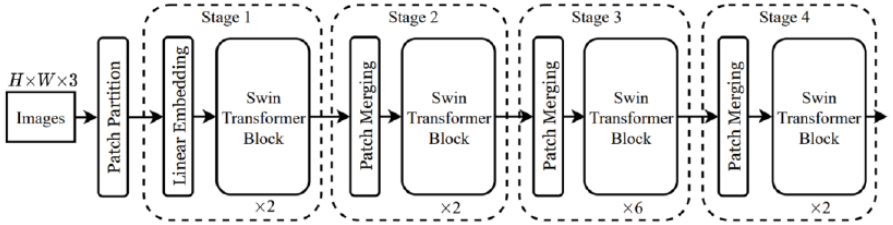
Table 5: Top1 accuracy (in %) of Vision Transformer on various datasets when pre-trained on ImageNet, ImageNet-21k or JFT300M. These values correspond to Figure 3 in the main text. Models are fine-tuned at 384 resolution. Note that the ImageNet results are computed without additional techniques (Polyak averaging and 512 resolution images) used to achieve results in Table 2.



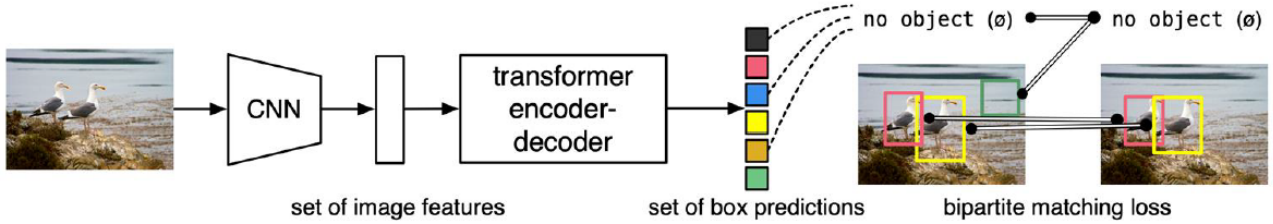
Conclusion



Fan et al, “Multiscale Vision Transformers”, ICCV 2021



Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021



Carion et al, “End-to-End Object Detection with Transformers”, ECCV 2020

Strength

- Apply Transformer architectures commonly used in NLP to Computer Vision Task
- Away from the existing limited attention mechanism, most of the CNN structure is replaced by Transformer
- Scalability

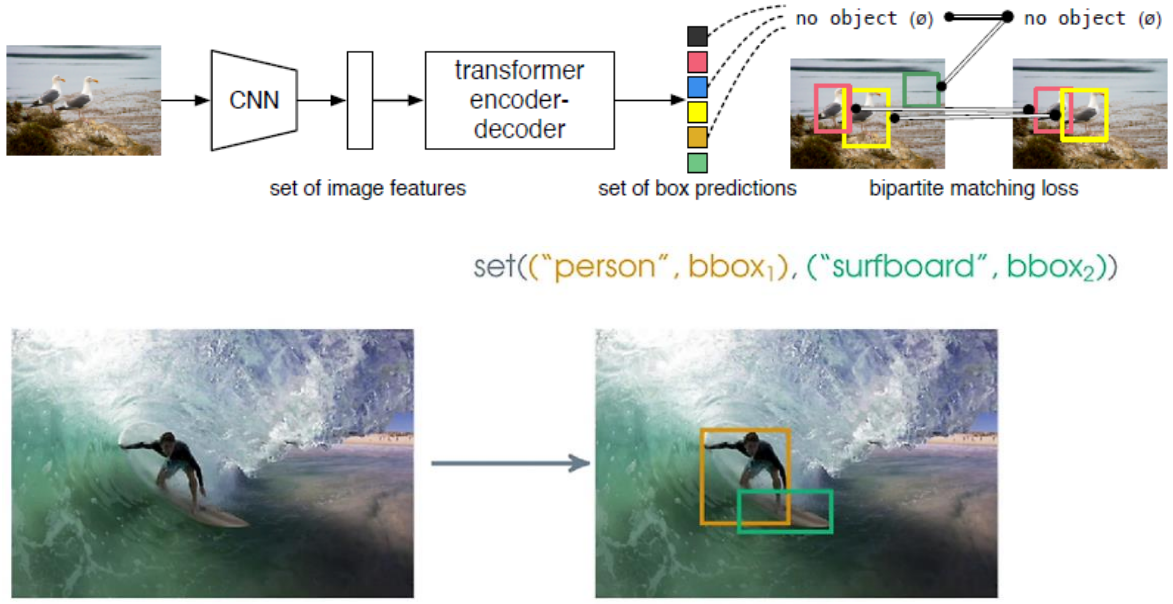
Weakness

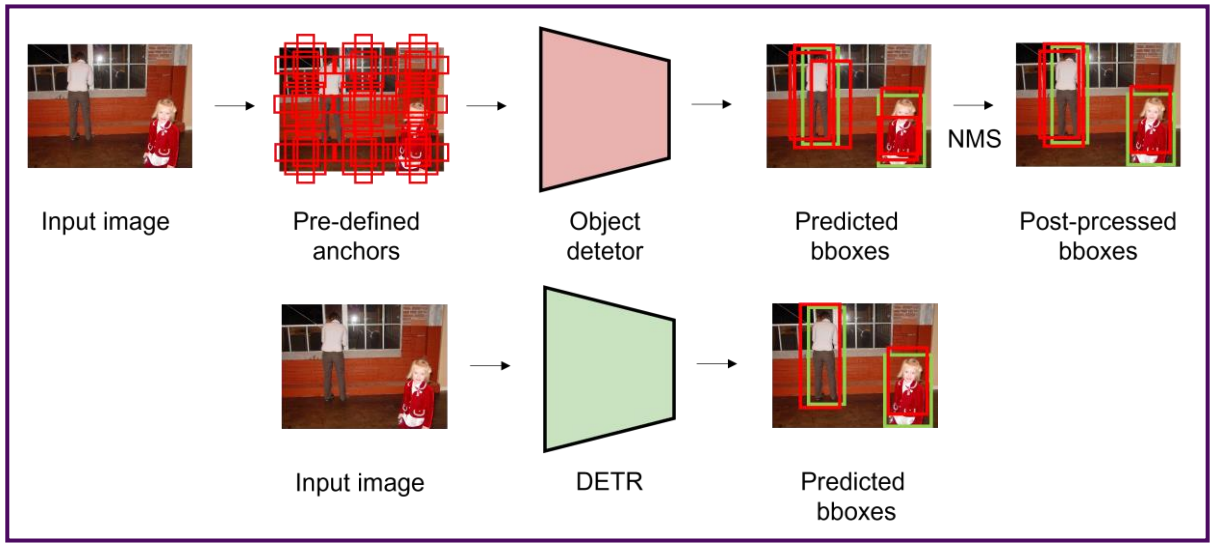
- Need more data than CNNs because of lack of inductive bias
- Computational complexity

DETR
: Detection Transformer

2020 CVPR

Introduction





Conjunction of the bipartite matching loss & Transformer

Encoder-decoder architecture : **Sequence prediction**

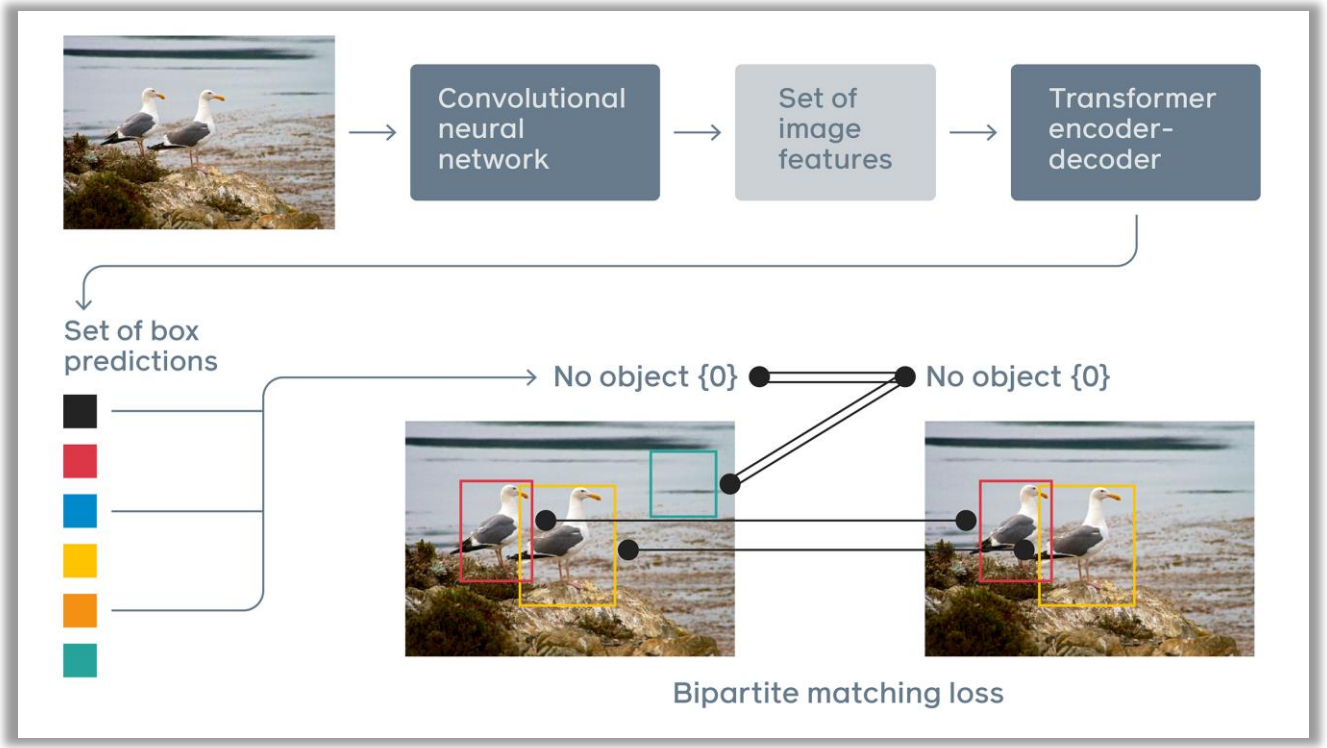
Simplifies the detection pipeline by dropping spatial anchors or NMS ⇒ postprocessing-free

Not require any customized layers → can be reproduced easily in any framework

DETR model

Bipartite Matching

$$\hat{\sigma} = argmin_{\sigma \in \mathfrak{S}_N} \sum_{i}^{N} \underbrace{\mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)})}_{\text{pair-wise matching cost}}$$



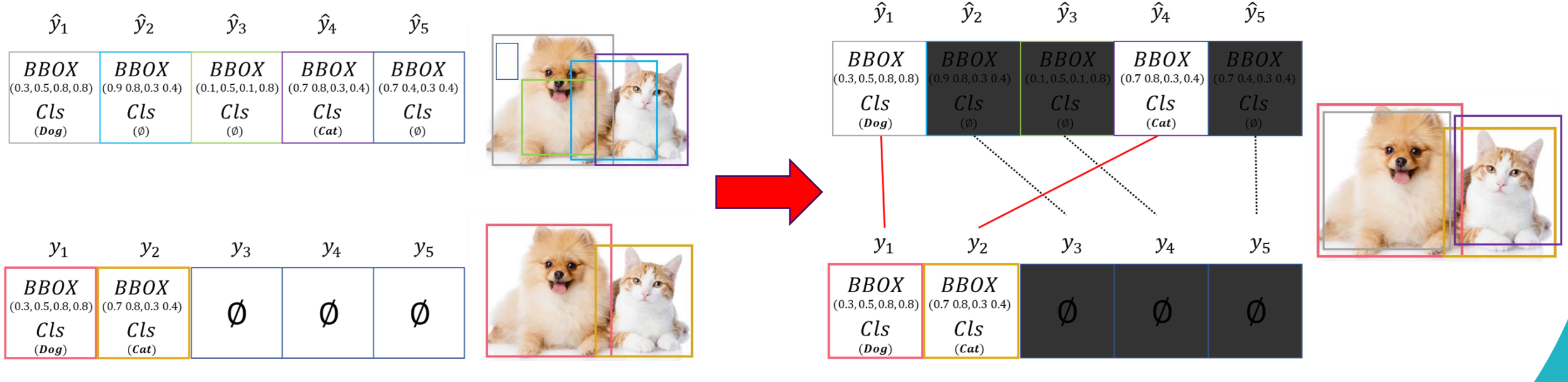
$$\mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}) = -\mathbb{I}_{\{c_i \neq \emptyset\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{I}_{\{c_i \neq \emptyset\}} \mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)})$$

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

$$\mathcal{L}_{iou}(b_{\sigma(i)}, \hat{b}_i) = 1 - \left(\frac{\left| b_{\sigma(i)} \cap \hat{b}_i \right|}{\left| b_{\sigma(i)} \cup \hat{b}_i \right|} - \frac{\left| B(b_{\sigma(i)}, \hat{b}_i) \backslash \mathrm{b}_{\sigma(i)} \cup \hat{b}_i \right|}{\left| B(b_{\sigma(i)}, \hat{b}_i) \right|} \right)$$

DETR model

Bipartite Matching



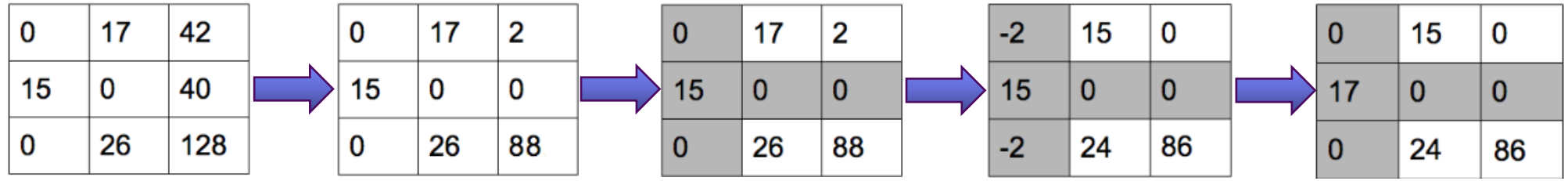
DETR model

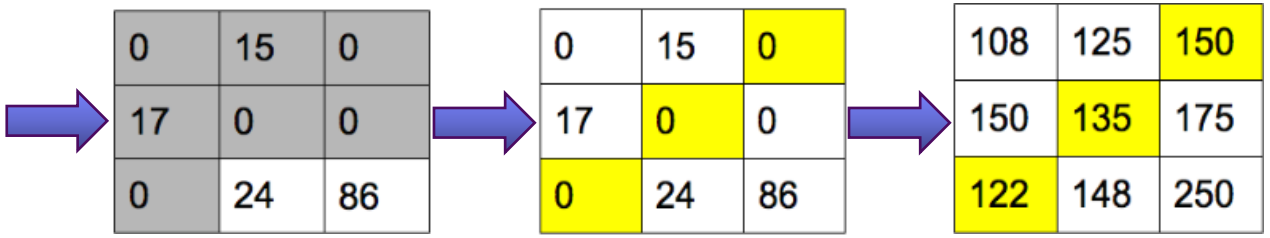
Hungarian Algorithm

Company	Cost for Musician	Cost for Chef	Cost for Cleaners
Company A	$108	$125	$150
Company B	$150	$135	$175
Company C	$122	$148	$250

Find bipartite matching between ground-truth and prediction
→ enforce permutation-invariance and guarantee unique match

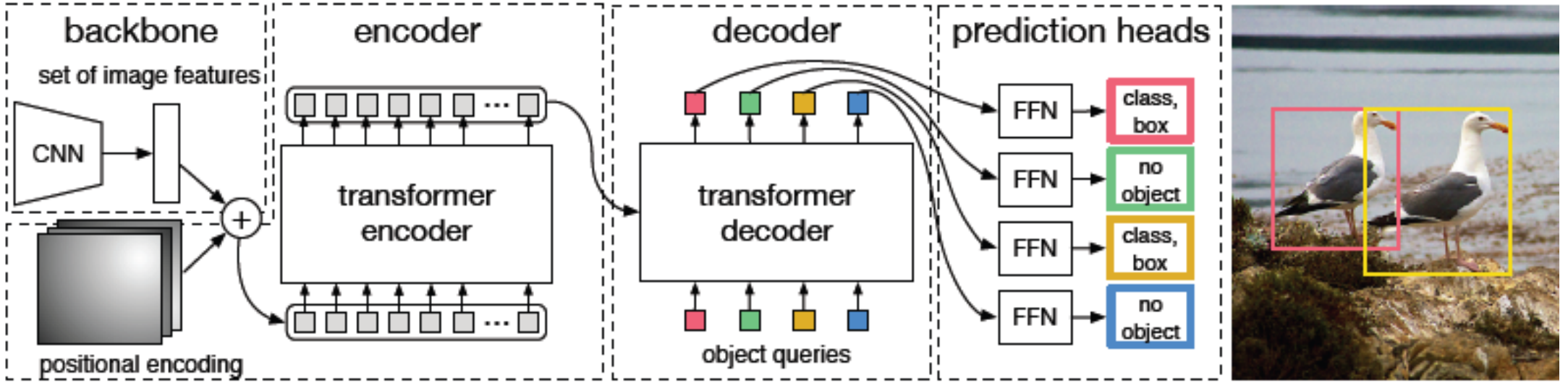
$$\mathcal{L}_{hungarian}(y, \hat{y}) = \sum_{i=1}^{N} \left[\underbrace{-\log \hat{p}_{\hat{\sigma}(i)}(c_i)}_{\text{Class prediction}} + \underbrace{\mathbb{1}_{\{c_i \neq \emptyset\}} \mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}}(i))}_{\text{Box loss}} \right]$$





108 + 135 + 250 = 493
108 + 148 + 175 = 431
150 + 125 + 250 = 525
150 + 148 + 150 = 448
122 + 125 + 175 = 422
122 + 135 + 150 = 407.

Transformer



Backbone

: CNN → Extract compact feature representation

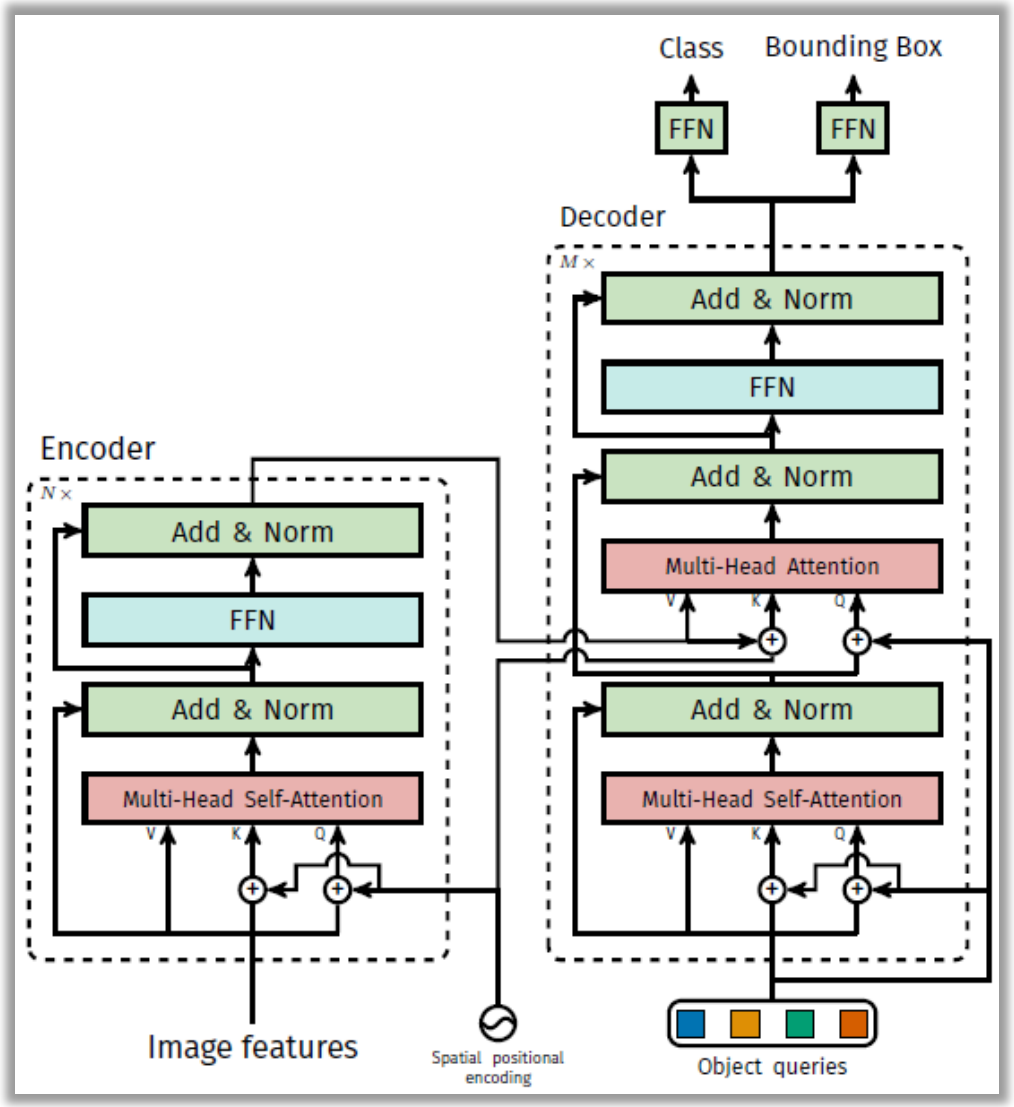
Transformer Encoder-Decoder

- Encoder : Reduce & Flatten activation map, applying MHA and FFN with positional encodings
- Decoder : Transform & Decode embeddings in parallel into box coordinates and classes

FFN (Feed Forward Network)

: 3-layer perceptron with ReLU → Predict normalized center coordinates

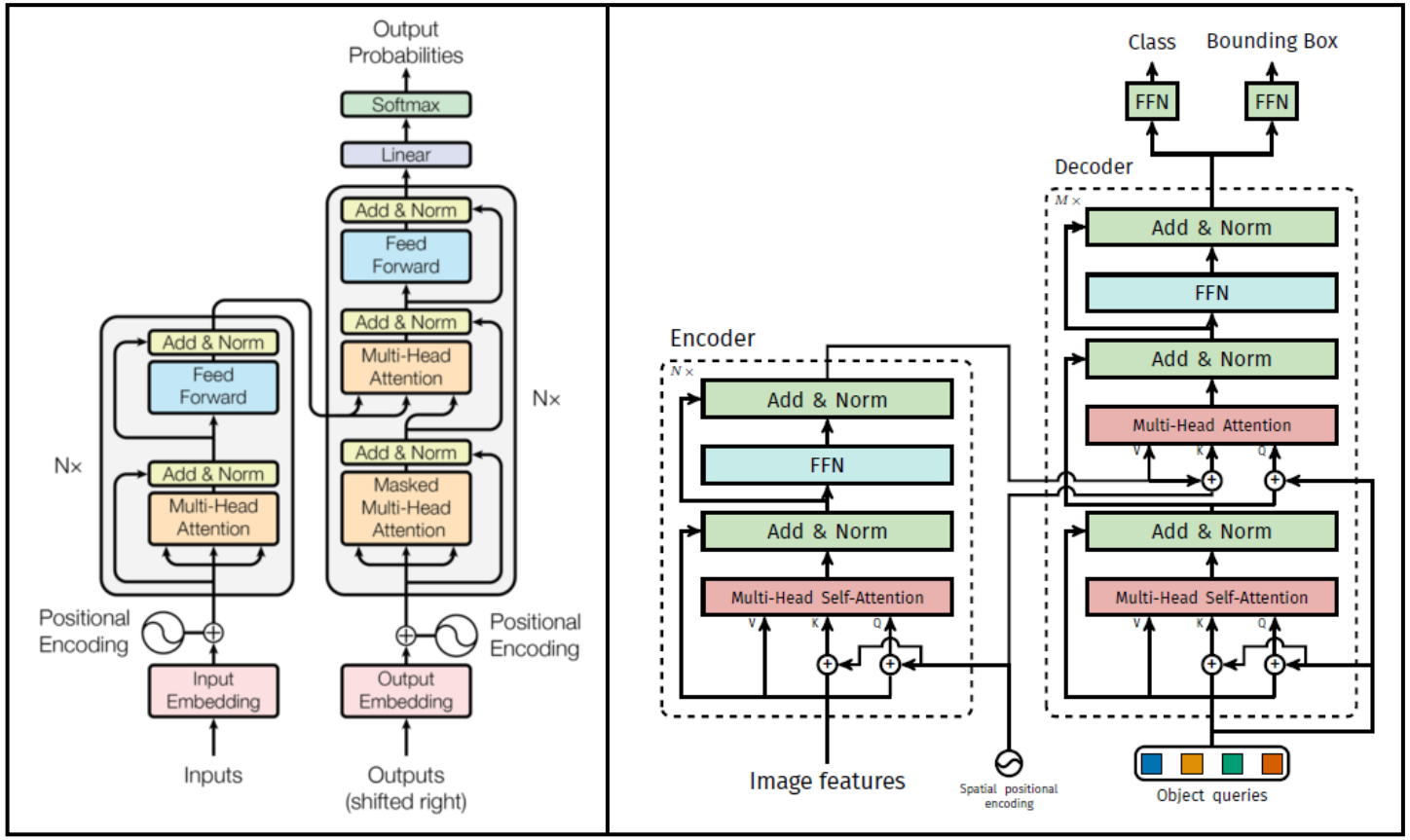
Transformer



[Strength]

① Understand context information of total image by using "Attention"
② Easy to identify interaction between each instances in image
③ Easy to identify correlations between pixels that are far from each other in large bounding box

Transformer

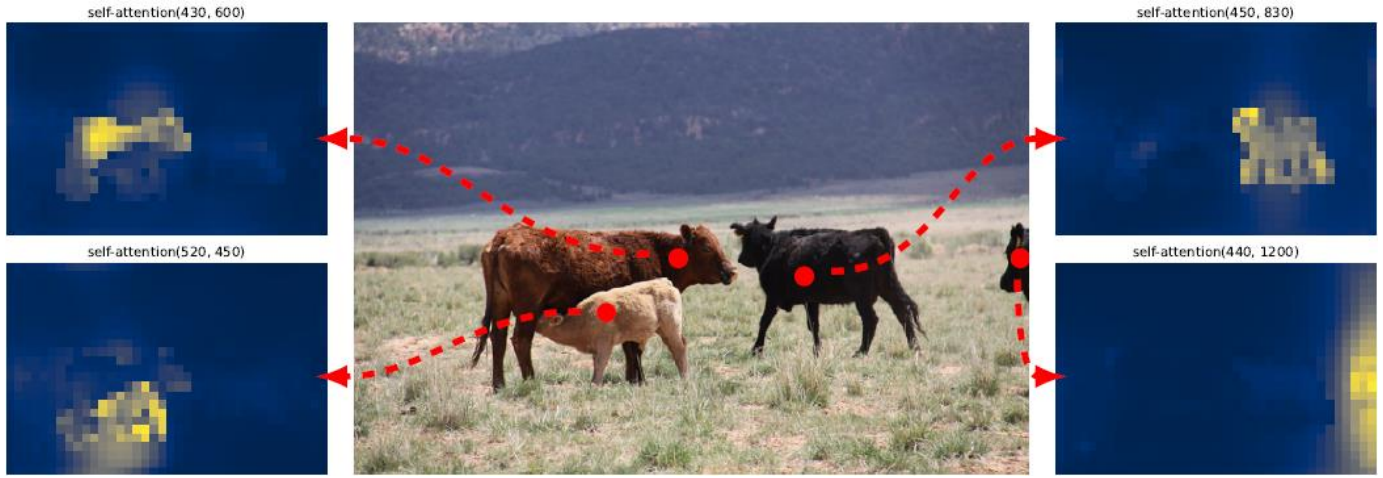


[Difference]

① Positional Encoding
② Auto-regression || Parallel
③ <u>Encoder</u>
Word embedding || Image feature map
④ <u>Decoder</u>
Target embedding || Object queries
⑤ Masked MHA || Multi-Head Self-Attention
⑥ 1 Head || 2 Head

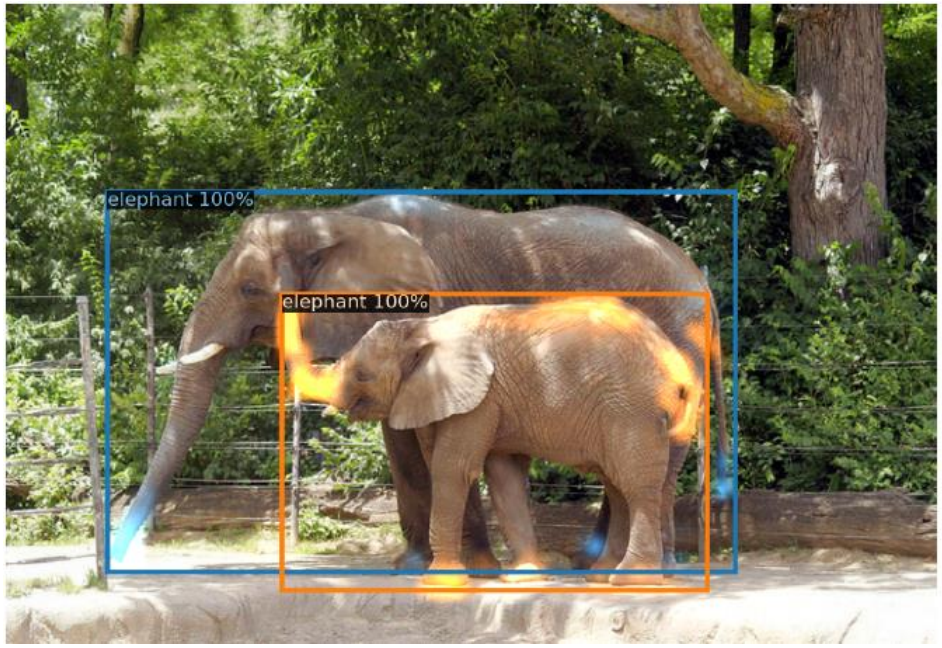
Transformer

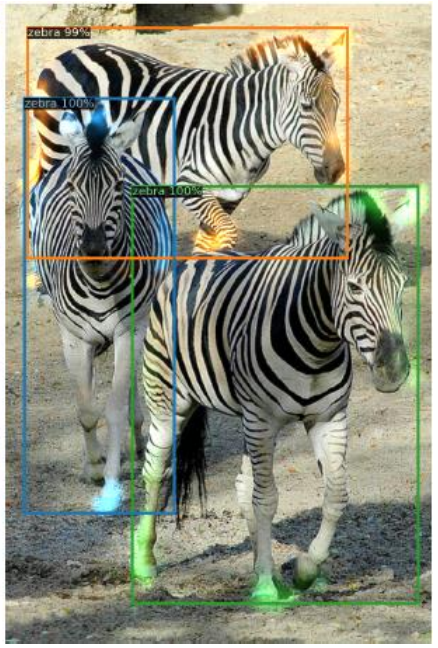
Encoder



Separate each instances
by global attention

Decoder





Extract boundary line
between each instance
and class

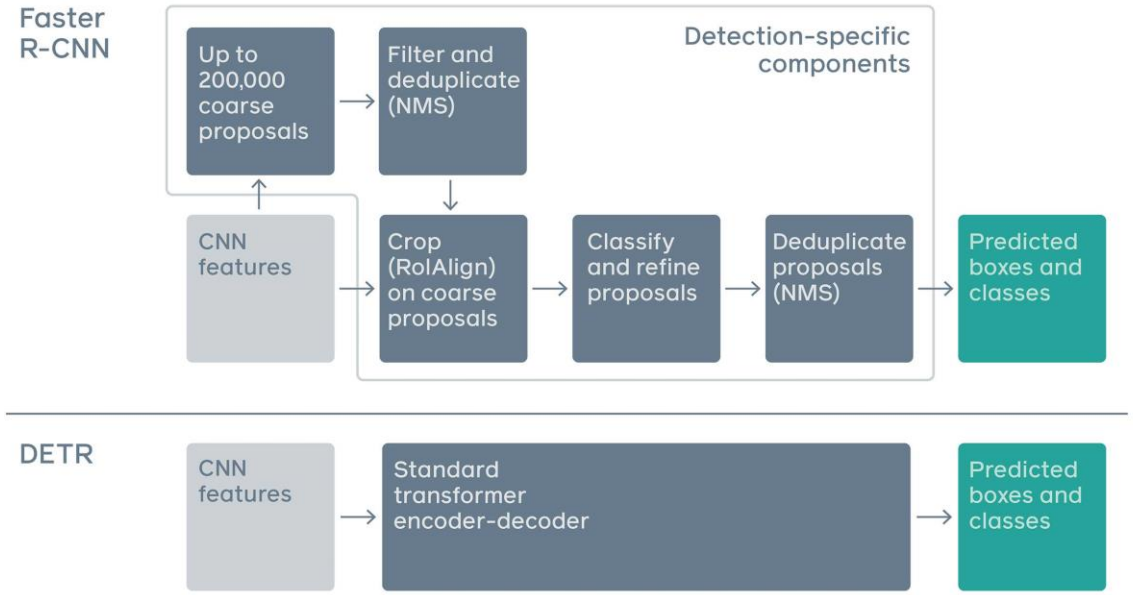
High attention score
in extermities

Dataset

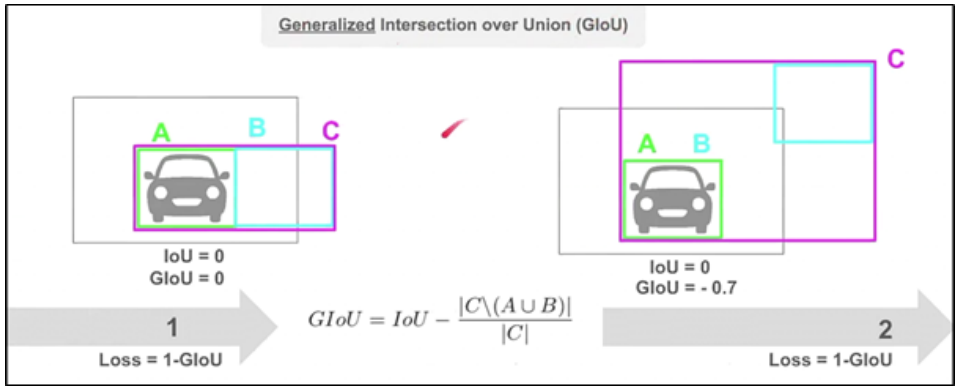
COCO 2017 detection and panoptic segmentation datasets

- 118k training images
- 5k validation images
- **For comparison with Faster R-CNN**

.vs Faster R-CNN



Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

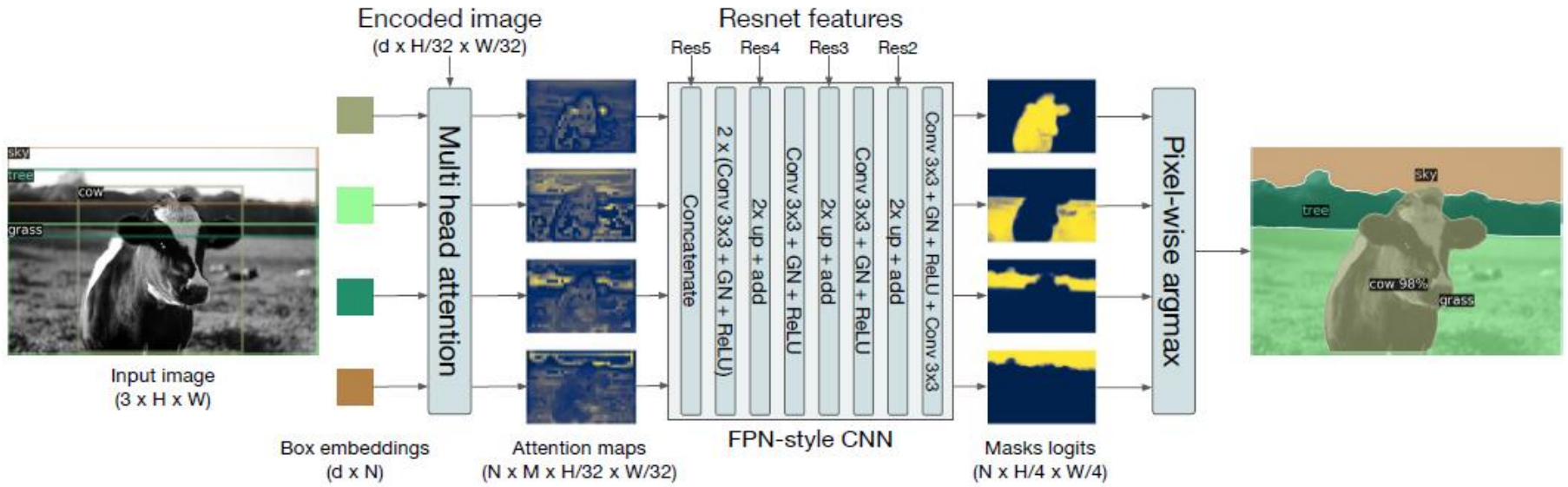


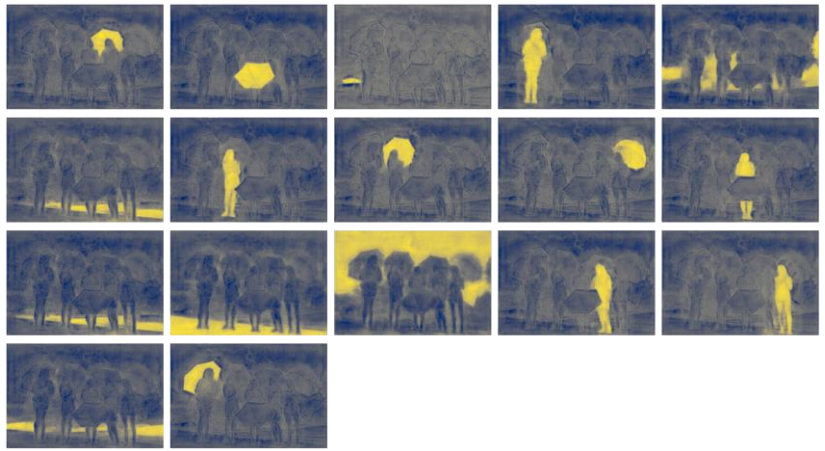
[Optimizer]
Transformer : Adam, Adagrad
Faster R-CNN : SGD with minimal data augmentation

Add **generalized IoU** to the box loss to align Faster R-CNN baseline with DETR

Panoptic Segmentation

Architecture (Head)





Produce panoptic segmentation by treating $stuff$ and $thing$ classes in a unified way

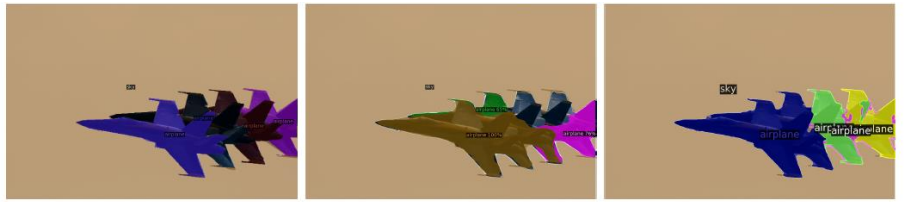
Versatile & Extensible

Panoptic Segmentation

Result

Fig. 9: Qualitative results for panoptic segmentation generated by DETR-R101. DETR produces aligned mask predictions in a unified manner for things and stuff.

Model	Backbone	PQ	SQ	RQ	PQ^{th}	SQ^{th}	RQ^{th}	PQ^{st}	SQ^{st}	RQ^{st}	AP
PanopticFPN++	R50	42.4	79.3	51.6	49.2	82.4	58.8	32.3	74.8	40.6	37.7
UPSnet	R50	42.5	78.0	52.5	48.6	79.4	59.6	33.4	75.9	41.7	34.3
UPSnet-M	R50	43.0	79.1	52.8	48.9	79.7	59.7	34.1	78.2	42.3	34.3
PanopticFPN++	R101	44.1	79.5	53.3	**51.0**	**83.2**	60.6	33.6	74.0	42.1	**39.7**
DETR	R50	43.4	79.3	53.8	48.2	79.8	59.5	36.3	78.5	45.3	31.1
DETR-DC5	R50	44.6	79.8	55.0	49.4	80.5	60.6	**37.3**	**78.7**	**46.5**	31.9
DETR-R101	R101	**45.1**	**79.9**	**55.5**	50.5	80.9	**61.7**	37.0	78.5	46.0	33.0



(a) Failure case with overlapping objects. PanopticFPN misses one plane entirely, while DETR fails to accurately segment 3 of them.

(b) `Things` masks are predicted at full resolution, which allows sharper boundaries than PanopticFPN

Fig. 11: Comparison of panoptic predictions. From left to right: Ground truth, PanopticFPN with ResNet 101, DETR with ResNet 101

Result

Table 3: Results for different positional encodings compared to the baseline (last row), which has fixed sine pos. encodings passed at every attention layer in both the encoder and the decoder. Learned embeddings are shared between all layers. Not using spatial positional encodings leads to a significant drop in AP. Interestingly, passing them in decoder only leads to a minor AP drop. All these models use learned output positional encodings.

spatial pos. enc. encoder	spatial pos. enc. decoder	output pos. enc. decoder	AP	Δ	AP_{50}	Δ
none	none	learned at input	32.8	-7.8	55.2	-6.5
sine at input	sine at input	learned at input	39.2	-1.4	60.0	-1.6
learned at attn.	learned at attn.	learned at attn.	39.6	-1.0	60.7	-0.9
none	sine at attn.	learned at attn.	39.3	-1.3	60.3	-1.4
sine at attn.	sine at attn.	learned at attn.	**40.6**	-	**61.6**	-

Table 4: Effect of loss components on AP. We train two models turning off ℓ_1 loss, and GIoU loss, and observe that ℓ_1 gives poor results on its own, but when combined with GIoU improves AP_M and AP_L. Our baseline (last row) combines both losses.

class	ℓ_1	GIoU	AP	Δ	AP_{50}	Δ	AP_S	AP_M	AP_L
✓	✓		35.8	-4.8	57.3	-4.4	13.7	39.8	57.9
✓		✓	39.9	-0.7	**61.6**	0	**19.9**	43.2	57.9
✓	✓	✓	**40.6**	-	**61.6**	-	**19.9**	**44.3**	**60.2**

Fig. 5: Out of distribution generalization for rare classes. Even though no image in the training set has more than 13 giraffes, DETR has no difficulty generalizing to 24 and more instances of the same class.

Inference

```python
import torch
from torch import nn
from torchvision.models import resnet50

class DETR(nn.Module):

    def __init__(self, num_classes, hidden_dim, nheads,
                 num_encoder_layers, num_decoder_layers):
        super().__init__()
        # We take only convolutional layers from ResNet-50 model
        self.backbone = nn.Sequential(*list(resnet50(pretrained=True).children())[:-2])
        self.conv = nn.Conv2d(2048, hidden_dim, 1)
        self.transformer = nn.Transformer(hidden_dim, nheads,
                                          num_encoder_layers, num_decoder_layers)
        self.linear_class = nn.Linear(hidden_dim, num_classes + 1)
        self.linear_bbox = nn.Linear(hidden_dim, 4)
        self.query_pos = nn.Parameter(torch.rand(100, hidden_dim))
        self.row_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))
        self.col_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))

    def forward(self, inputs):
        x = self.backbone(inputs)
        h = self.conv(x)
        H, W = h.shape[-2:]
        pos = torch.cat([
            self.col_embed[:W].unsqueeze(0).repeat(H, 1, 1),
            self.row_embed[:H].unsqueeze(1).repeat(1, W, 1),
        ], dim=-1).flatten(0, 1).unsqueeze(1)
        h = self.transformer(pos + h.flatten(2).permute(2, 0, 1),
                             self.query_pos.unsqueeze(1))
        return self.linear_class(h), self.linear_bbox(h).sigmoid()

detr = DETR(num_classes=91, hidden_dim=256, nheads=8, num_encoder_layers=6, num_decoder_layers=6)
detr.eval()
inputs = torch.randn(1, 3, 800, 1200)
logits, bboxes = detr(inputs)
```

Listing 1: DETR PyTorch inference code. For clarity it uses learnt positional encodings in the encoder instead of fixed, and positional encodings are added to the input only instead of at each transformer layer. Making these changes requires going beyond PyTorch implementation of transformers, which hampers readability. The entire code to reproduce the experiments will be made available before the conference.

Conclusion

Strength

- Apply Transformer architecture to Object Detection
- End-to-end training
- Faster R-CNN baseline level accuracy and runtime performance were shown for COCO dataset.

Weakness

- Require lots of time to training because of Transformer architecture
- Degradation of performance in small object

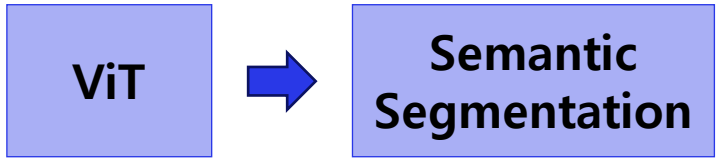
Segmenter

: Transformer for Semantic Segmentation

2021 ICCV

Introduction

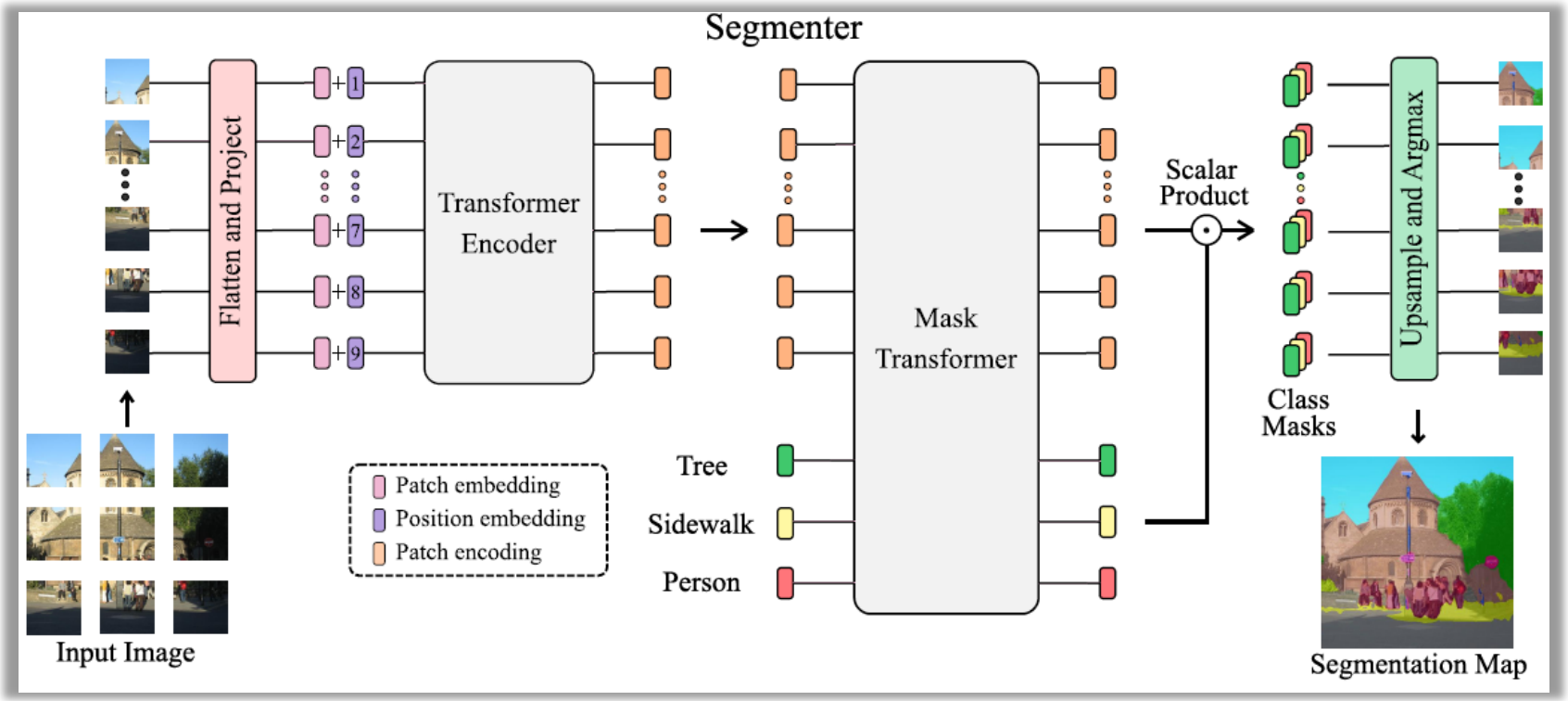
Image Segmentation = level of individual image patches + contextual information



Contextual information → channel, spatial attention, point-wise attention
: Convolutional backbone … biased towards local interactions

- Semantic segmentation as sequence-to-sequence problem
- Transformer architecture to leverage contextual information at every stage of the model
  ⇒ Capture global interactions between elements of a scene + No built-in inductive prior

Architecture



$$a_{i-1} = MSA\big(LN(z_{i-1})\big) + z_{i-1}$$
$$z_i = MLP\big(LN(a_{i-1})\big) + a_{i-1}$$

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

$$Masks(z'_M, c) = z'_M c^T$$

- ViT backbone
- Mask decoder inspired by DETR
- No convolution

Model	Backbone	Layers	Token size	Heads	Params
Seg-Ti	ViT-Ti	12	192	3	6M
Seg-S	ViT-S	12	384	6	22M
Seg-B	ViT-B	12	768	12	86M
Seg-B†	DeiT-B	12	768	12	86M
Seg-L	ViT-L	24	1024	16	307M

Table 1: Details of Transformer variants.

Attention Map

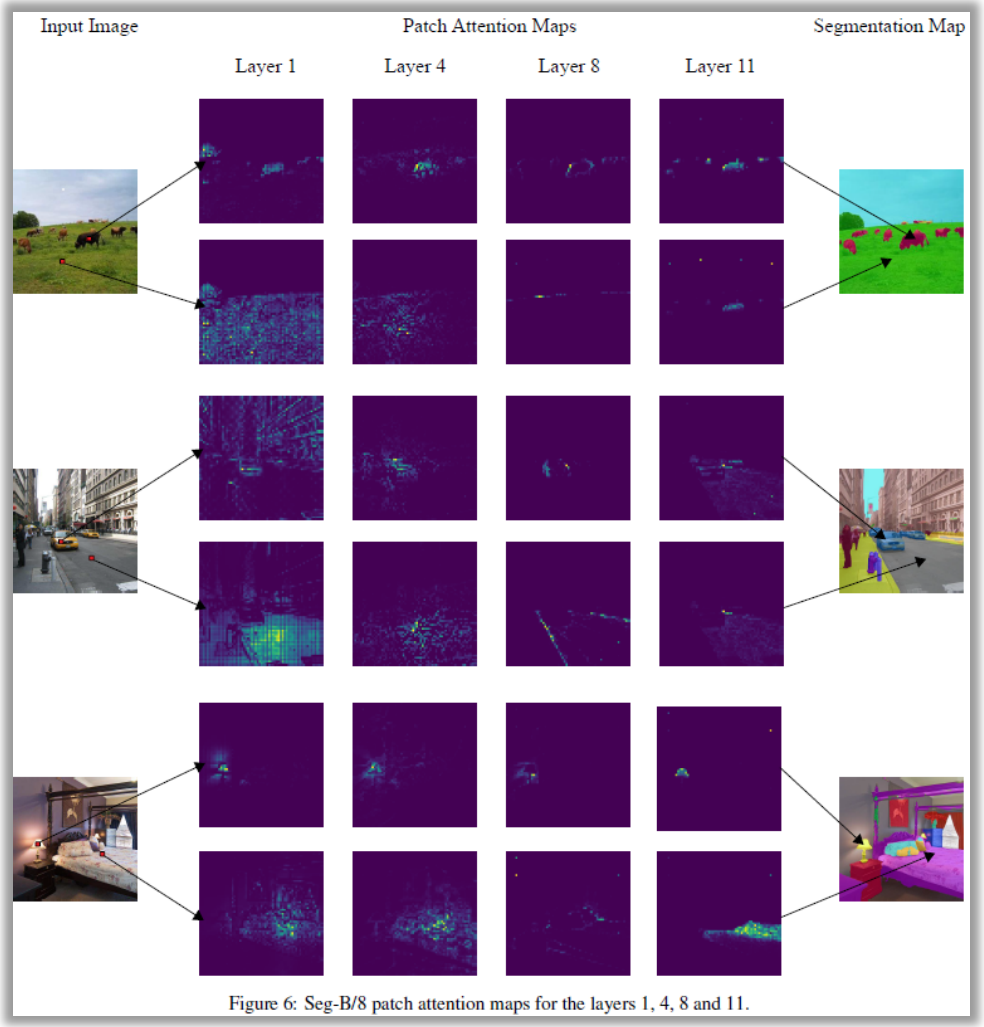
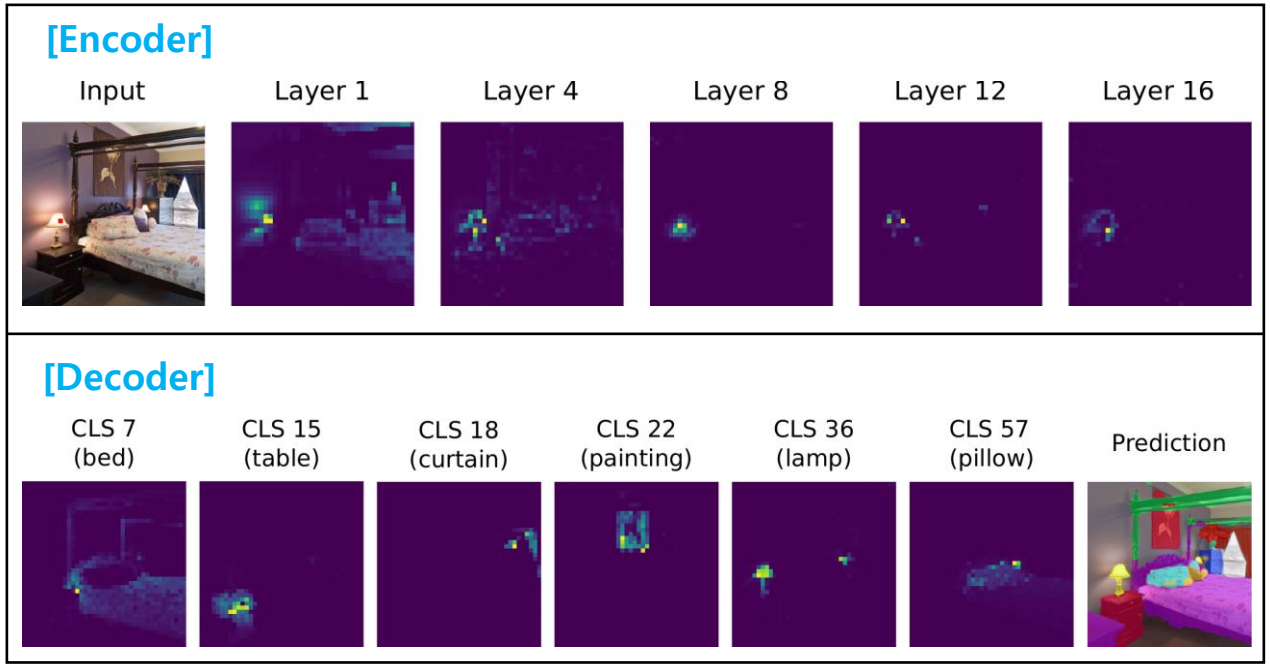


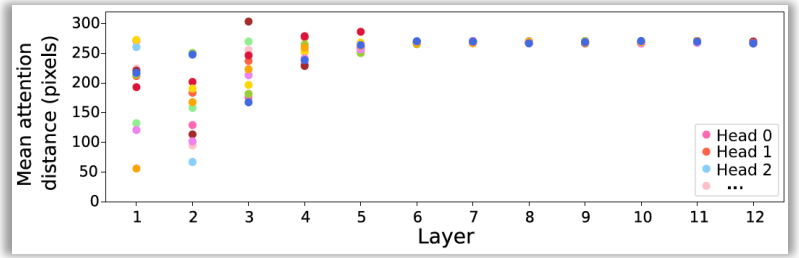
Figure 6: Seg-B/8 patch attention maps for the layers 1, 4, 8 and 11.

Attention map field-of-view adapts to the input image and the instance size, gathering global information on large instances and focusing on local information on smaller instances



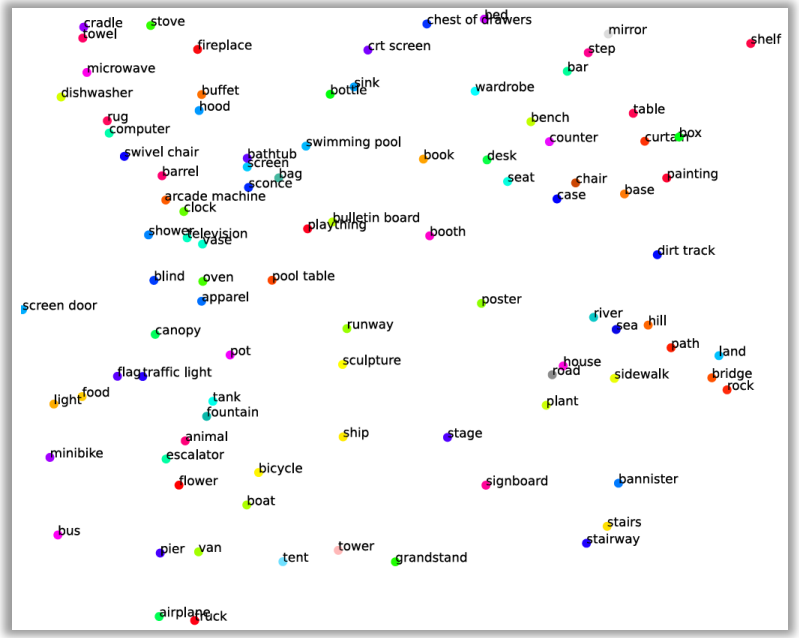
Class embeddings

Size of Attended area



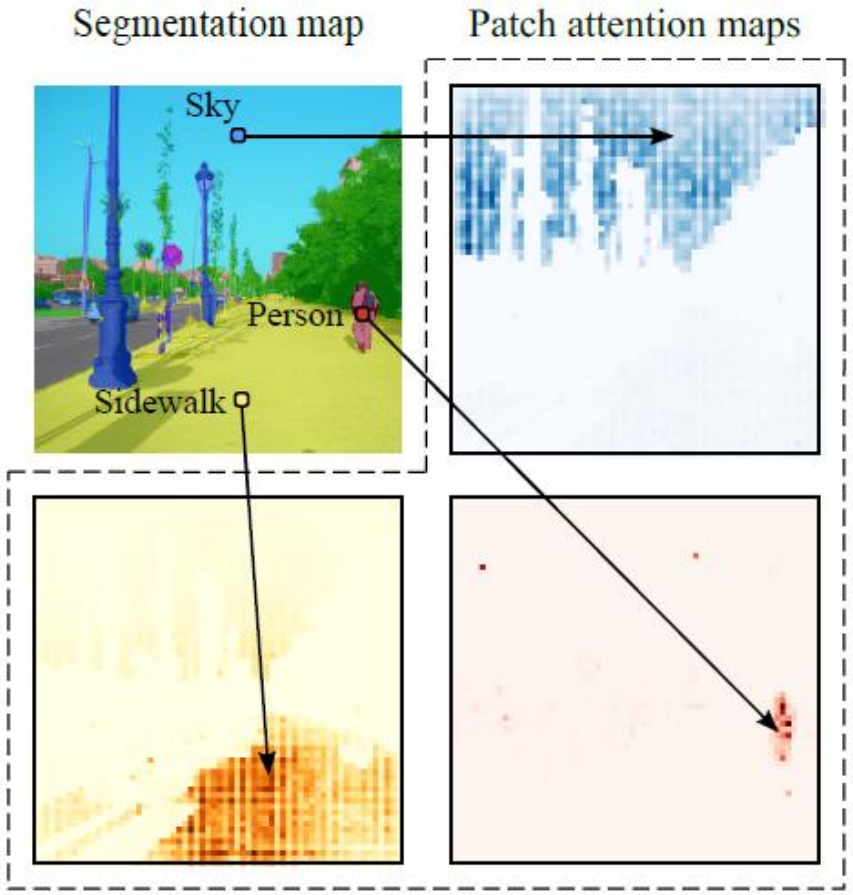
To illustrate larger receptive field size compared to CNNs

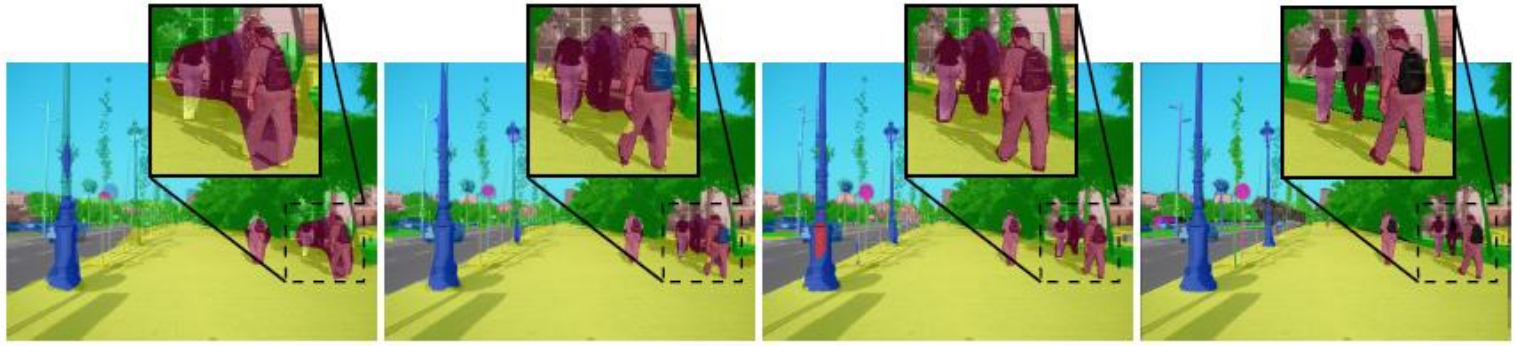
Class embeddings learned with mask transformer



Implicit clustering of semantically related categories

Segmentation Map





(a) Patch size 32 × 32 (b) Patch size 16 × 16 (c) Patch size 8 × 8 (d) Ground Truth

Figure 3: Impact of the model patch size on the segmentation maps.

Method	Backbone	Patch size	Im/sec	ImNet acc.	mIoU (SS)
Seg-Ti/16	ViT-Ti	16	396	78.6	39.03
Seg-S/32	ViT-S	32	1032	80.5	40.64
Seg-S/16	ViT-S	16	196	83.7	45.37
Seg-B†/16	DeiT-B	16	92	85.2	47.08
Seg-B/32	ViT-B	32	516	83.3	43.07
Seg-B/16	ViT-B	16	92	86.0	48.06
Seg-B/8	ViT-B	8	7	85.7	49.54
Seg-L/16	ViT-L	16	33	87.1	50.71

Table 3: Performance comparison of different Segmenter models with varying backbones and input patch sizes on ADE20K validation set.

.vs FCN

Method	Decoder	Small	Medium	Large	mIoU (SS)
DeepLab RNeSt-101	UNet	37.85	50.89	50.67	46.47
Seg-B/32	Linear	31.95	47.82	49.44	43.07
Seg-B-Mask/32	Mask	32.29	49.44	50.82	44.19
Seg-B†/16	Linear	38.31	50.91	52.08	47.10
Seg-B†-Mask/16	Mask	40.49	51.37	54.24	48.70
Seg-B/16	Linear	39.57	51.32	53.28	48.06
Seg-B-Mask/16	Mask	40.16	52.61	52.66	48.48
Seg-B/8	Linear	41.43	54.35	52.85	49.54
Seg-L/16	Linear	42.08	54.67	55.39	50.71
Seg-L-Mask/16	Mask	42.02	54.83	57.06	51.30

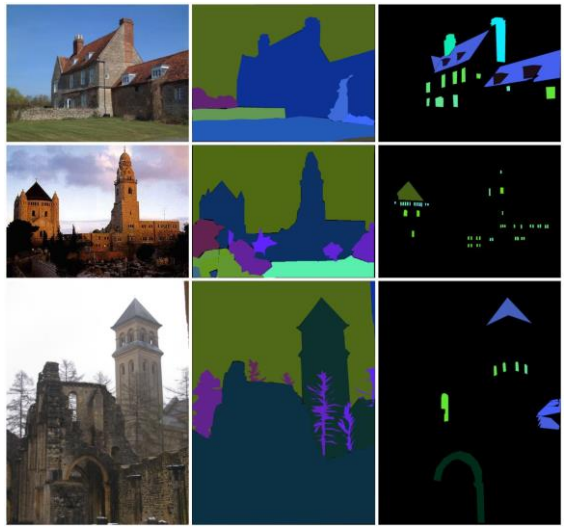
Table 4: Evaluation with respect to the object size on ADE20k validation set (mean IoU). Comparison of DeepLabv3+ ResNeSt-101 to Segmenter models with a linear or a mask transformer decoder.

Method	Backbone	Im/sec	mIoU	+MS
OCR [60]	HRNetV2-W48	83	-	45.66
ACNet [24]	ResNet-101	-	-	45.90
DNL [57]	ResNet-101	-	-	45.97
DRANet [22]	ResNet-101	-	-	46.18
CPNet [58]	ResNet-101	-	-	46.27
DeepLabv3+ [10]	ResNet-101	76	45.47	46.35
DeepLabv3+ [10]	ResNeSt-101	15	46.47	47.27
DeepLabv3+ [10]	ResNeSt-200	-	-	48.36
SETR-L MLA [67]	ViT-L/16	34	48.64	50.28
Swin-L UperNet [35]	Swin-L/16	34	52.10	**53.50**
Seg-B†/16	DeiT-B/16	77	47.08	48.05
Seg-B†-Mask/16	DeiT-B/16	76	48.70	50.08
Seg-L/16	ViT-L/16	33	50.71	52.25
Seg-L-Mask/16	ViT-L/16	31	51.82	**53.63**

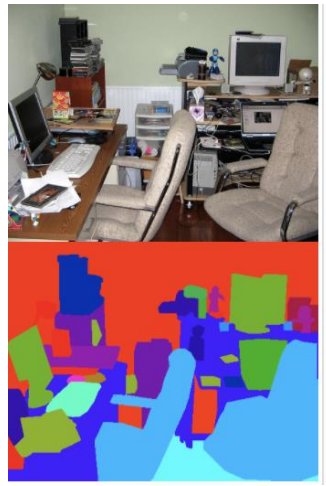
Table 6: State-of-the-art comparison on ADE20K validation set.

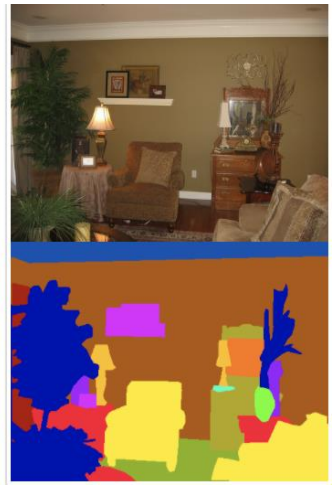
Result

Datasets

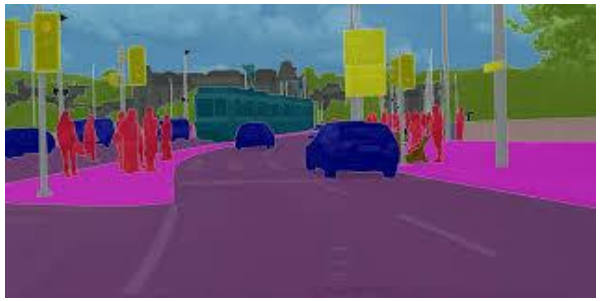


- ADE20k
  - 20,210 images with 150 semantic classes
  - Validation : 2,000 & Test : 3,352





- Pascal Context
  - 4,996 images with 59 semantic classes + background class
  - Validation : 5,104 images
- Cityscape
  - 5,000 images from 50 different cities with 19 semantic classes
  - Training : 2,975 & Validation : 500 & Test : 1,525



Result

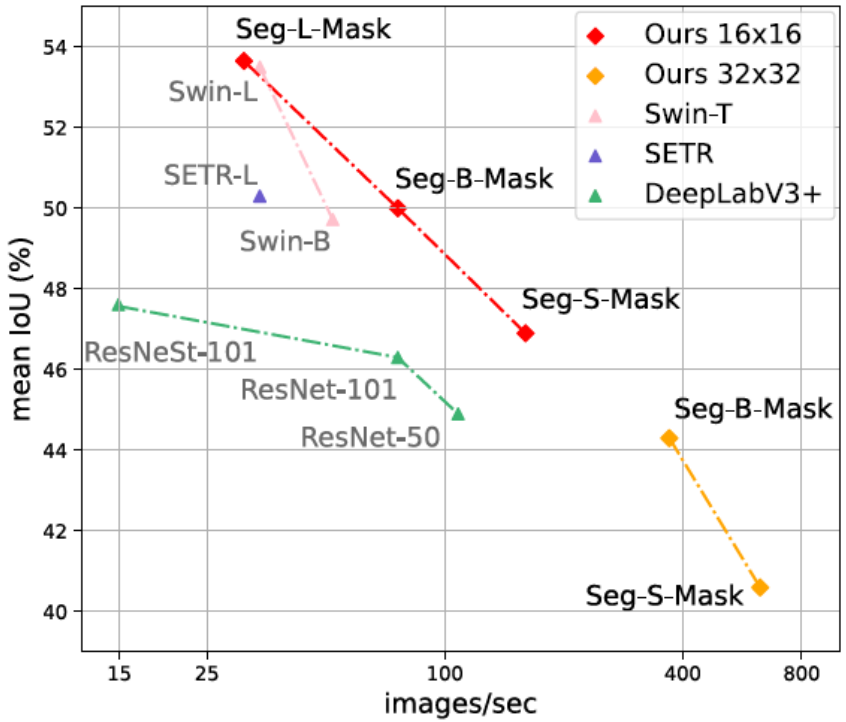


Figure 4: Images per second and mean IoU for our approach compared to other methods on ADE20K validation set. Segmenter models offer a competitive trade-off in terms of performance and precision.

Method	Decoder	Small	Medium	Large	mIoU (SS)
DeepLab RNeSt-101	UNet	37.85	50.89	50.67	46.47
Seg-B/32	Linear	31.95	47.82	49.44	43.07
Seg-B-Mask/32	Mask	32.29	49.44	50.82	44.19
Seg-B†/16	Linear	38.31	50.91	52.08	47.10
Seg-B†-Mask/16	Mask	40.49	51.37	54.24	48.70
Seg-B/16	Linear	39.57	51.32	53.28	48.06
Seg-B-Mask/16	Mask	40.16	52.61	52.66	48.48
Seg-B/8	Linear	41.43	54.35	52.85	49.54
Seg-L/16	Linear	42.08	54.67	55.39	50.71
Seg-L-Mask/16	Mask	42.02	54.83	57.06	51.30

Table 4: Evaluation with respect to the object size on ADE20k validation set (mean IoU). Comparison of DeepLabv3+ ResNeSt-101 to Segmenter models with a linear or a mask transformer decoder.

Dataset Size	4k	8k	12k	16k	20k
mIoU (SS)	38.31	41.87	43.42	44.61	45.37

Table 5: Performance comparison of Seg-S/16 models trained with increasing dataset size and evaluated on ADE20K validation set.

Result


DeepLabv3+
Segmenter
Ground Truth
ADE20K
(a)
(b)
(c)
(d)
(e)
CITYSCAPES
(f)


DeepLabv3+
Segmenter
Ground Truth
(a)
(b)
(c)
(d)

Result

Failures

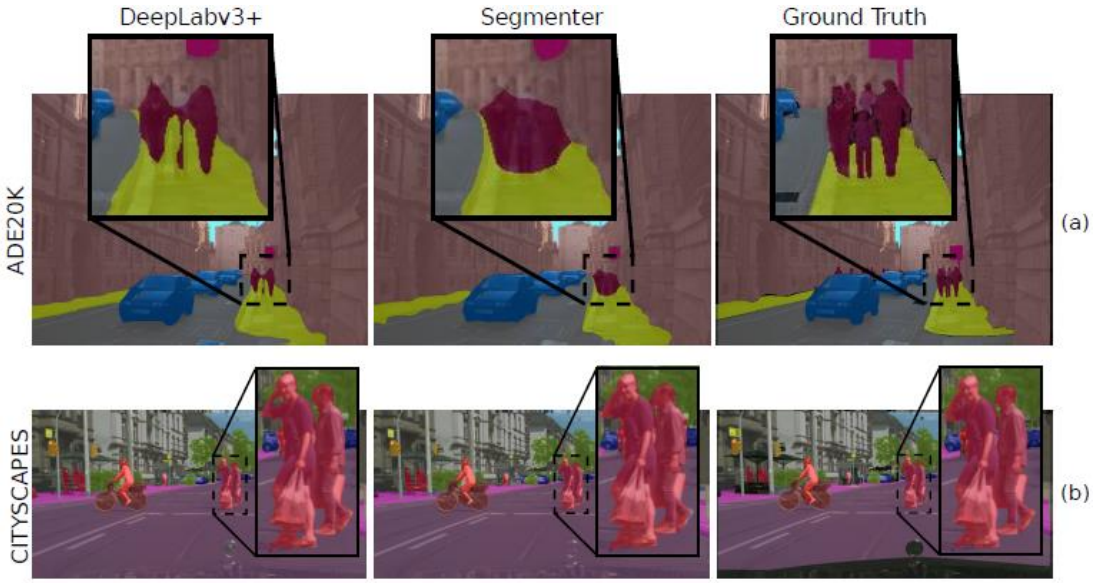


Figure 11: Comparison of Seg-L-Mask/16 with DeepLabV3+ ResNeSt-101 for images with near-by persons. We can observe that DeepLabV3+ localizes boundaries better.

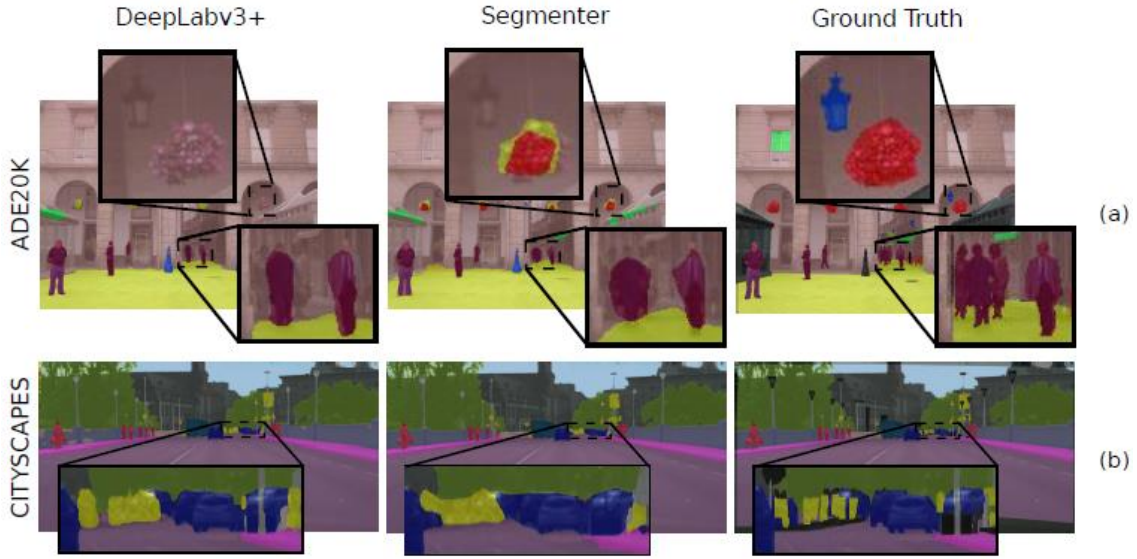


Figure 12: Failure cases of DeepLabV3+ ResNeSt-101 and Seg-L-Mask/16, for small instances such as (a) lamp, people, flowers and (b) cars, signals.

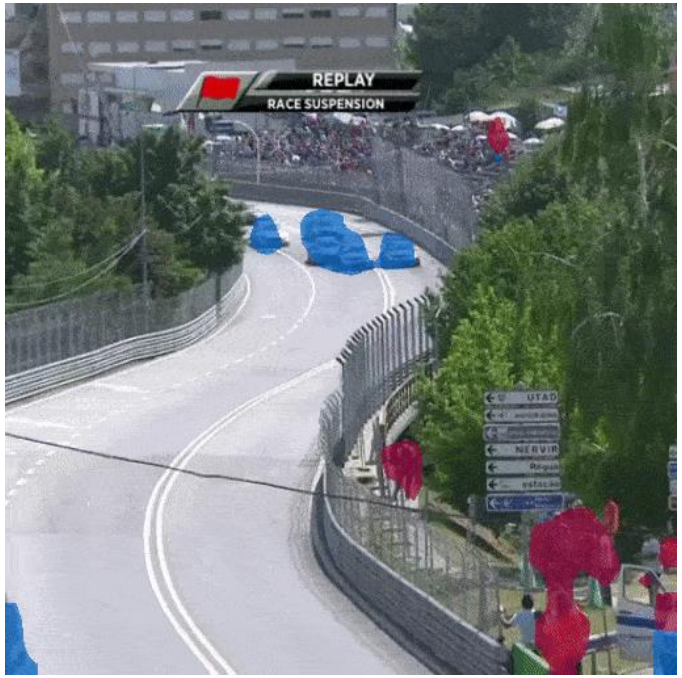
Conclusion

Strength

- Apply Transformer architecture to Semantic Segmentation
- Result in competitive performance on standard image without convolutions
- Capture global image context by design
- End-to-end encoder-decoder transformer

Weakness

- High computation cost



Reference

- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiauhua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvian Gelly, Jakob Uszkoreit, Neil Houlsby.「**An IMAGE IS WORTH 16x16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE**」, 2020
- Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, Sergey Zagoruyko.「**End-to-End Object Detection with Transformers**」, 2020
- Robin Strudel, Ricardo Garcia, Ivan Laptev, Cordelia Schmid.「**Segmenter: Transformer for Semantic Segmentation**」, 2020
- Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girschick.「**Mask R-CNN**」, 2017
- Jonathan Long, Evan Shelhamer, Trevor Darrell.「**Fully Convolutional Networks for Semantic Segmentation**」, 2014
- Alexander Kirillov, Kaiming He, Ross Girschick, Carsten Rother, Piotr Dollar.「**Panoptic Segmentation**」, 2018
- https://github.com/facebookresearch/detr
- https://github.com/rstrudel/segmenter
- https://github.com/google-research/vision_transformer
- https://discuss.pytorch.kr/t/vision-transformer-a-visual-guide-to-vision-transformers/4158

THANK YOU