A Survey on Vision Transformer

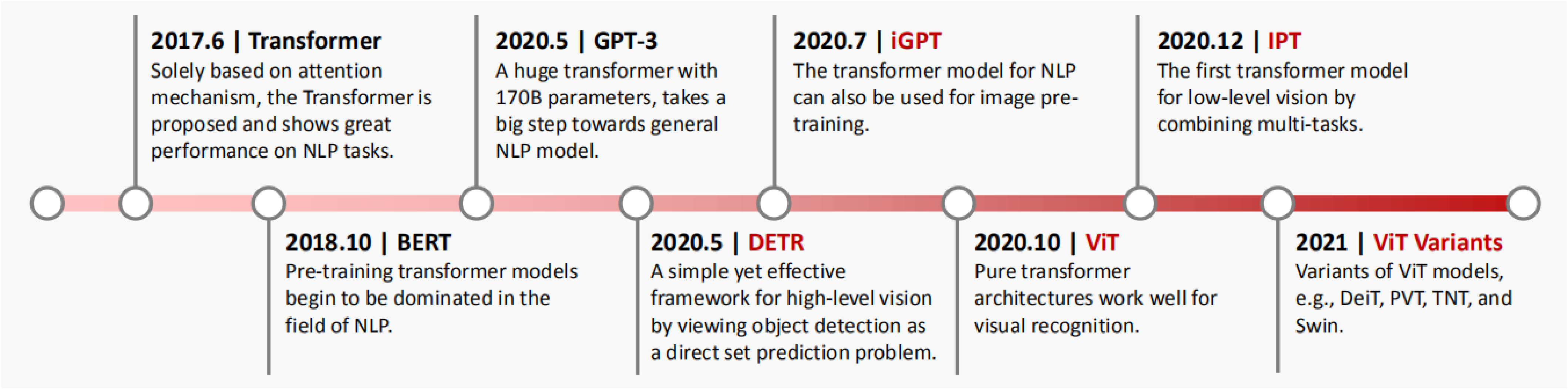
Tech report

Kai Han (韩凯)
Huawei Noah’s Ark Lab

Contents

History of Transformer

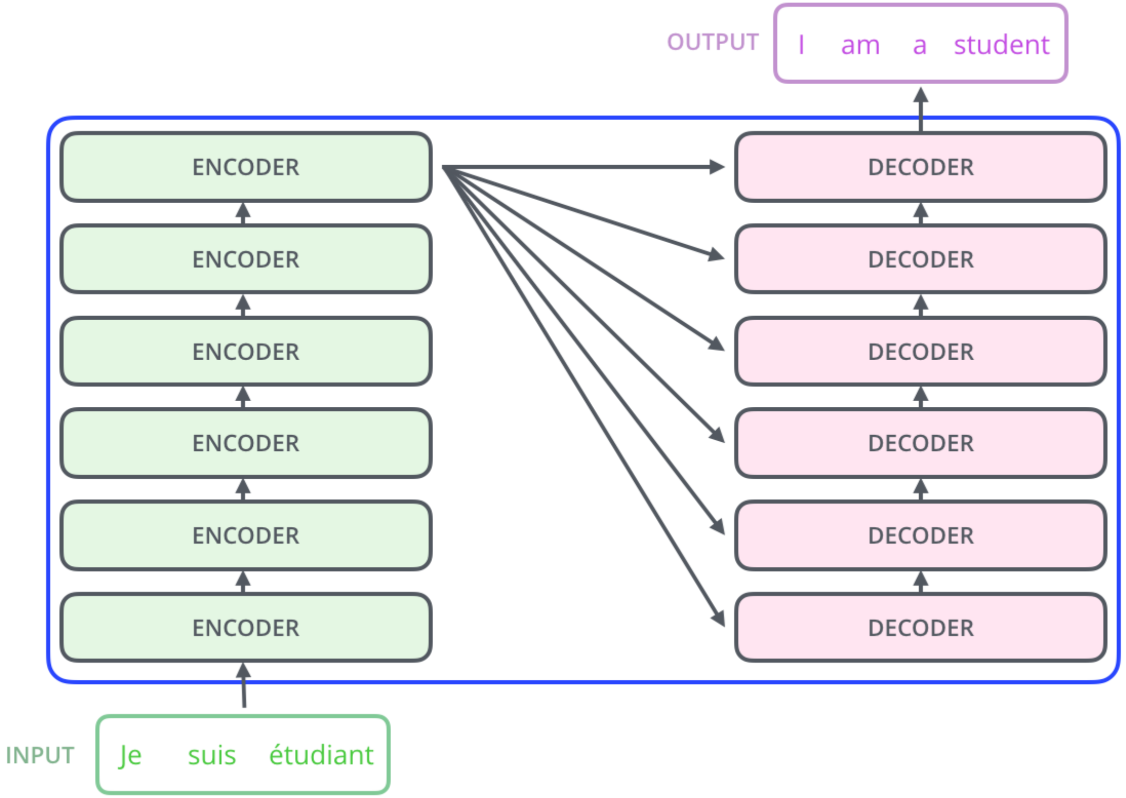
- Transformer is a type of deep neural network mainly based on the self-attention mechanism.
- Transformer is first widely applied to the field of natural language processing, and appears to achieve competitive performance on **computer vision** tasks.



Han, Kai, et al. "A Survey on Vision Transformer." *arXiv preprint arXiv:2012.12556* (2020).

Transformer: A High-level Look

- Transformer is used to process sequence data.

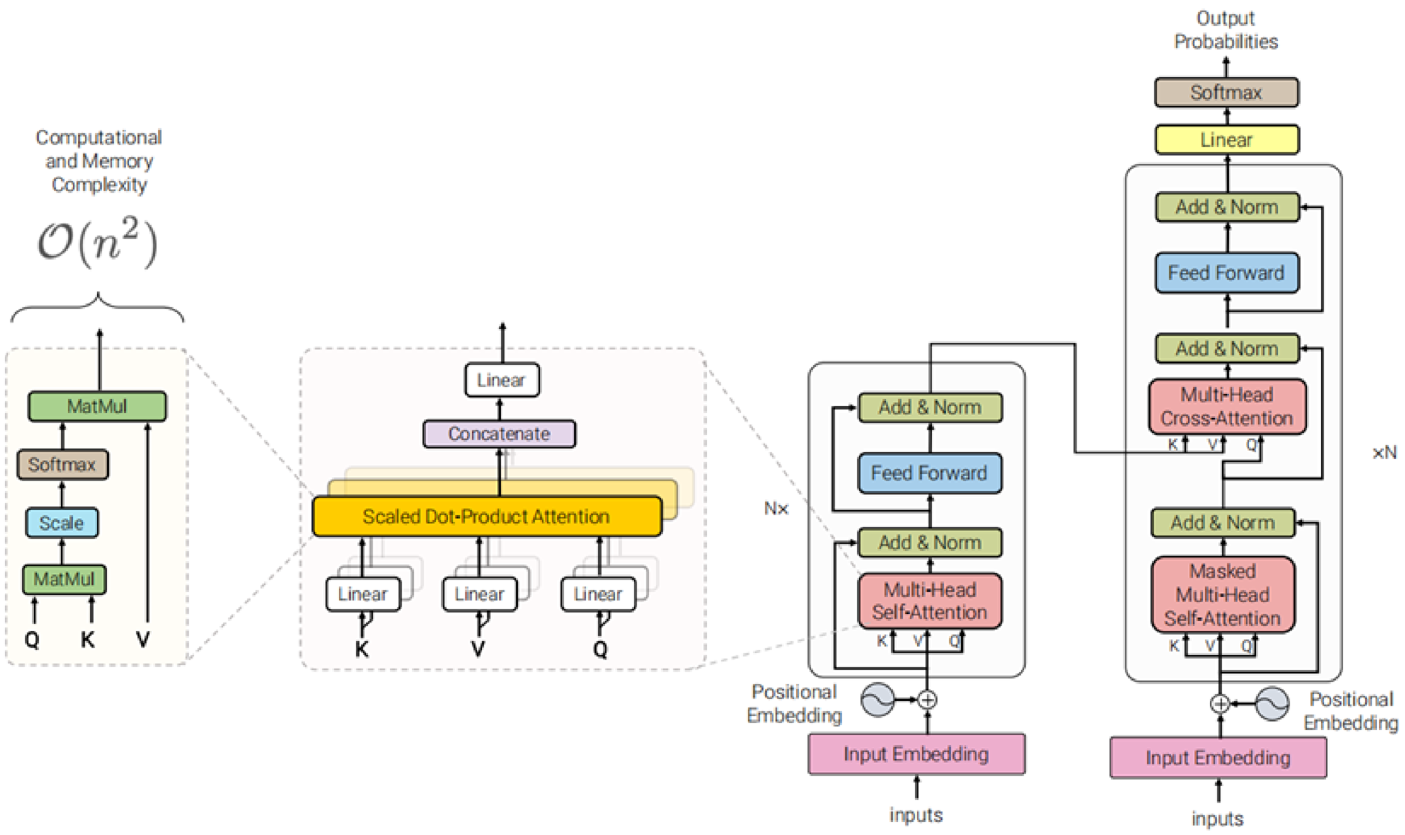


http://jalammar.github.io/illustrated-transformer/

Transformer: Components

- Components of Transformer
  - Multi-head self-attention
  - Feed-forward network
  - Layer normalization
  - Shortcut connection
  - Position encoding

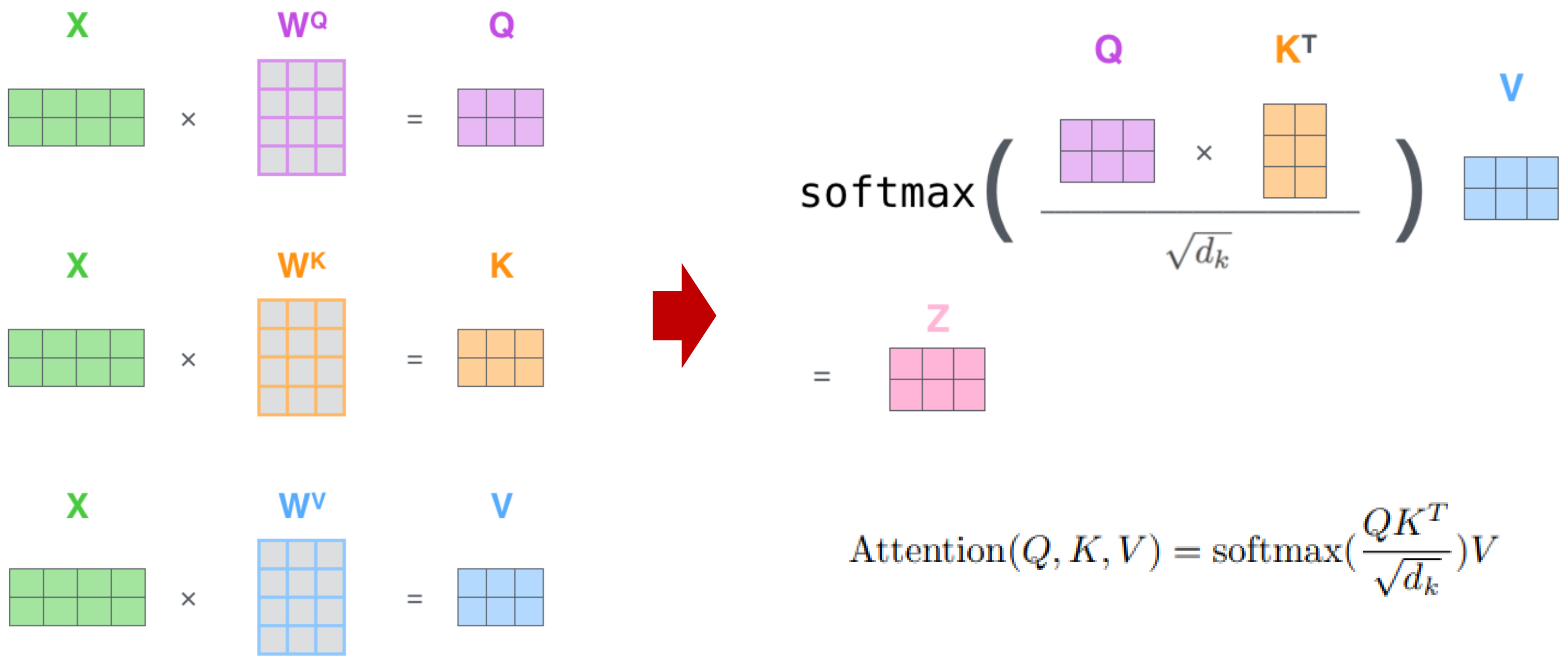
- Advantages of Transformer
  - Long-range relationships
  - Parallelized computing
  - Capacity for big data
  - Less inductive bias
  - etc



Vaswani, Ashish, et al. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Multi-head self-attention

- Self-attention

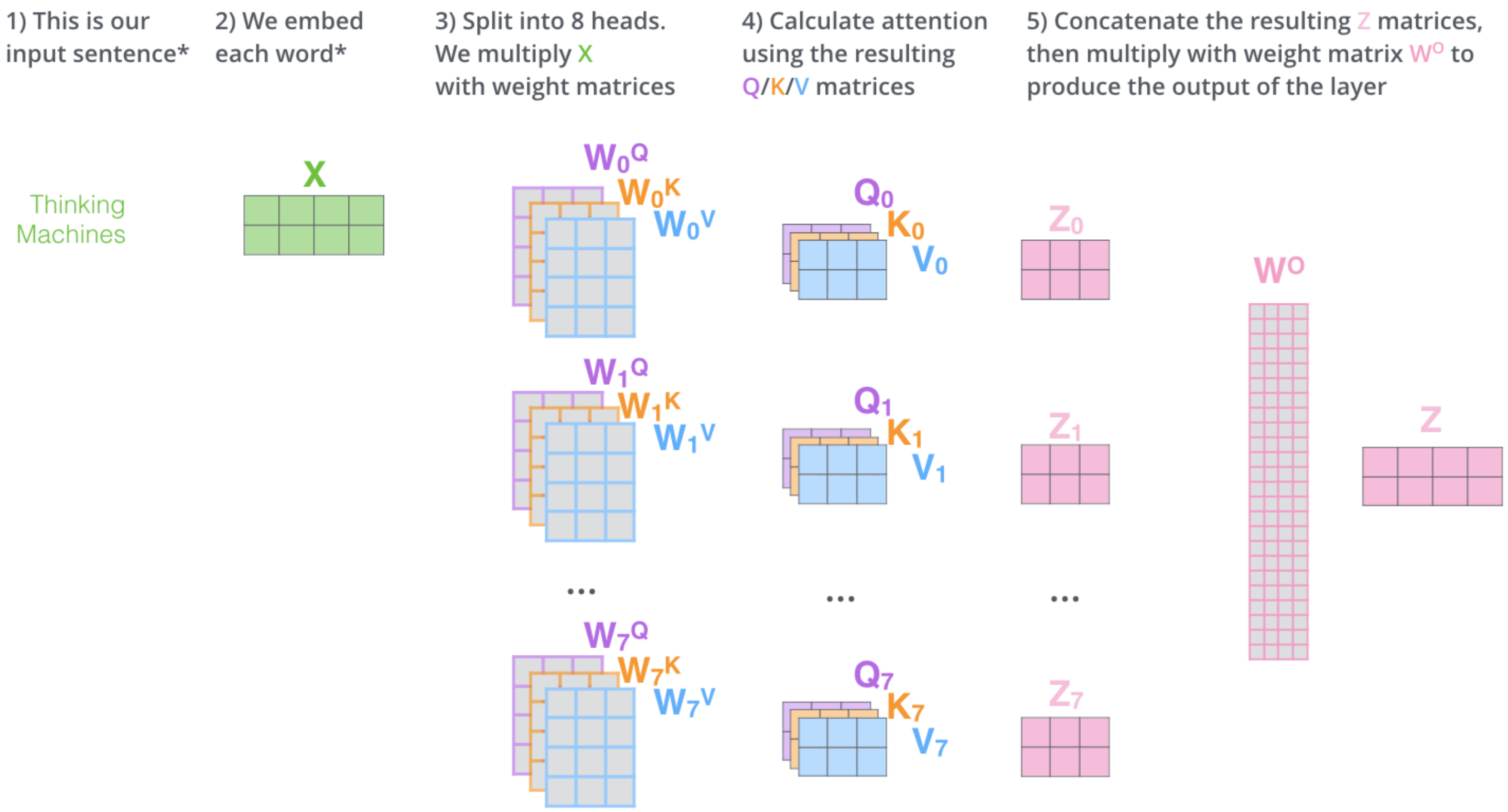


$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Multi-head self-attention

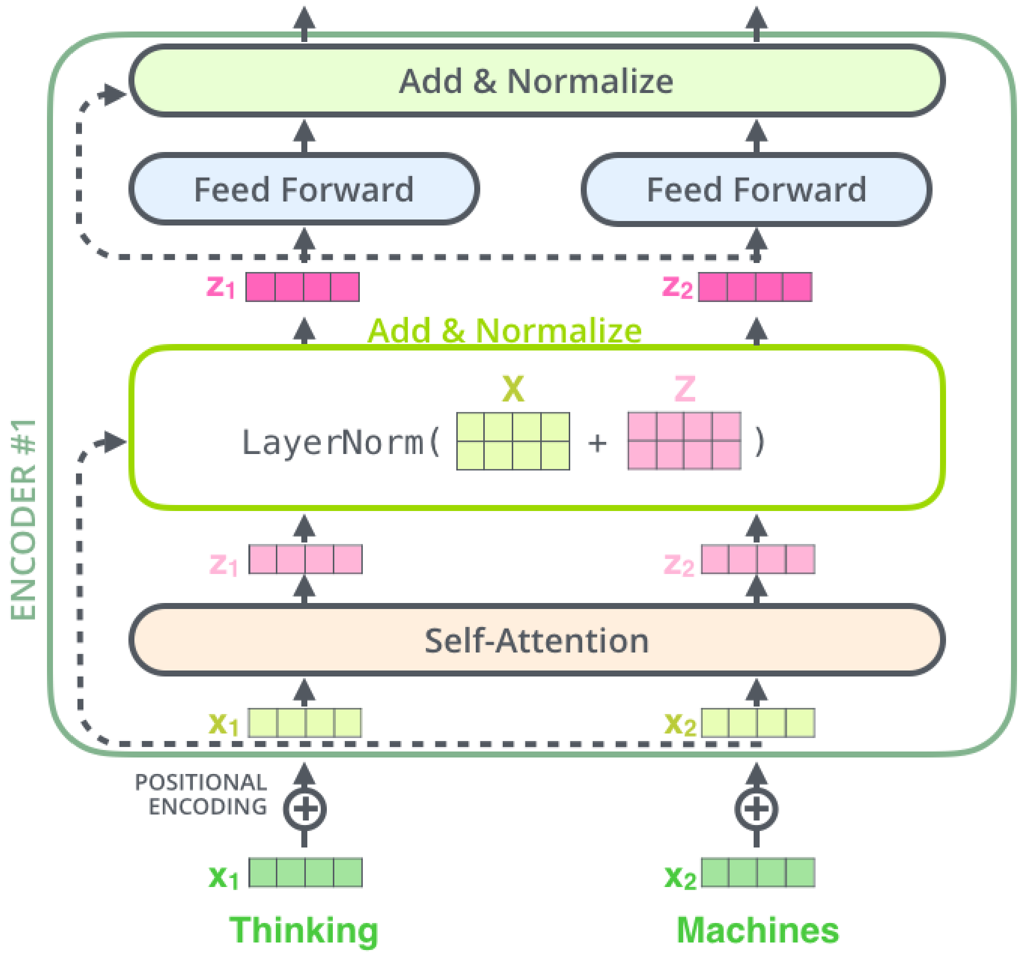
- Multi-head Self-attention

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



FFN & LayerNorm & Shortcut

- A transformer block

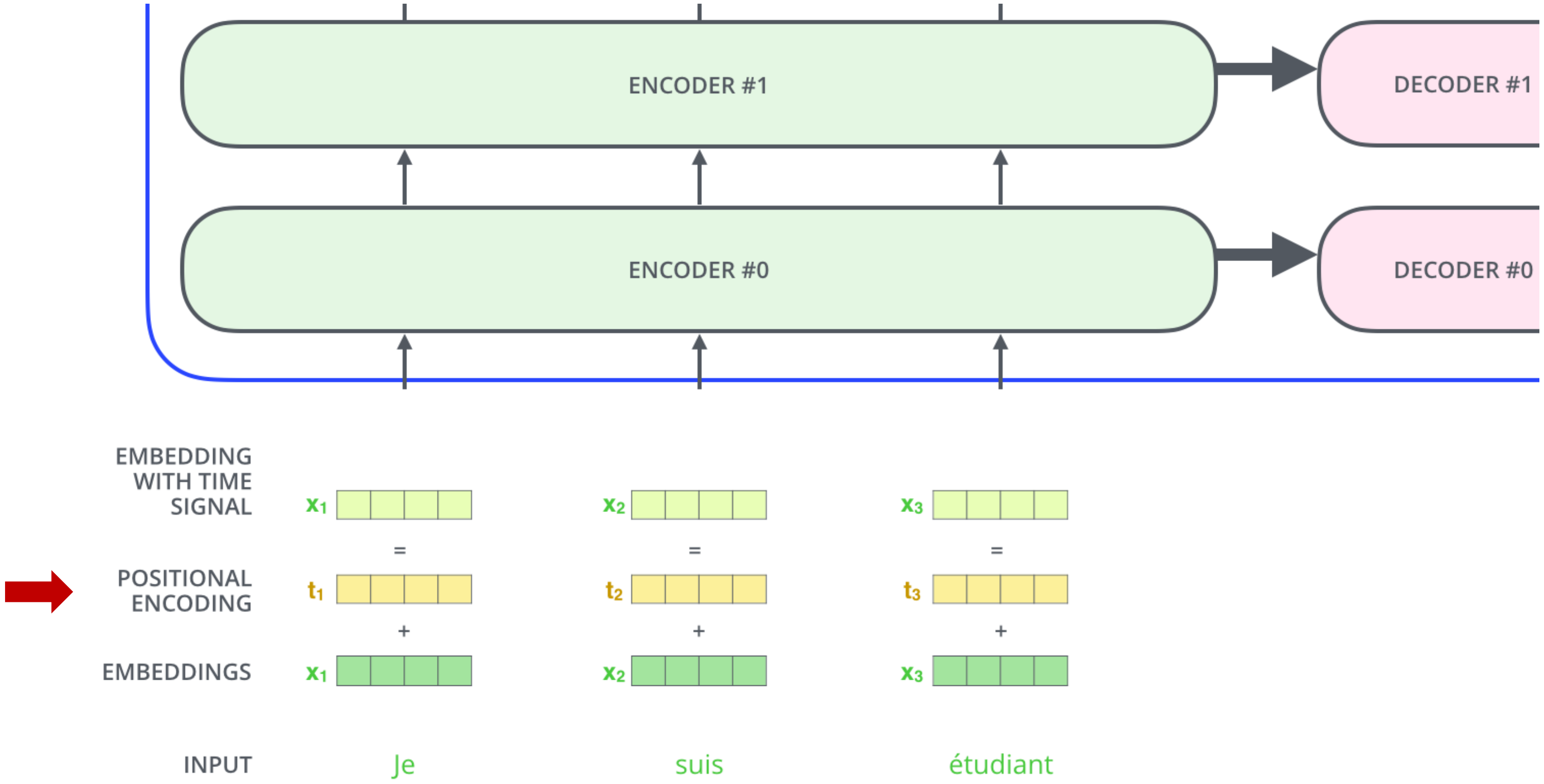


$$\mathrm{FFN}(X) = W_2\sigma(W_1X),$$

$$\mathrm{LayerNorm}(X + \mathrm{Attention}(X)).$$

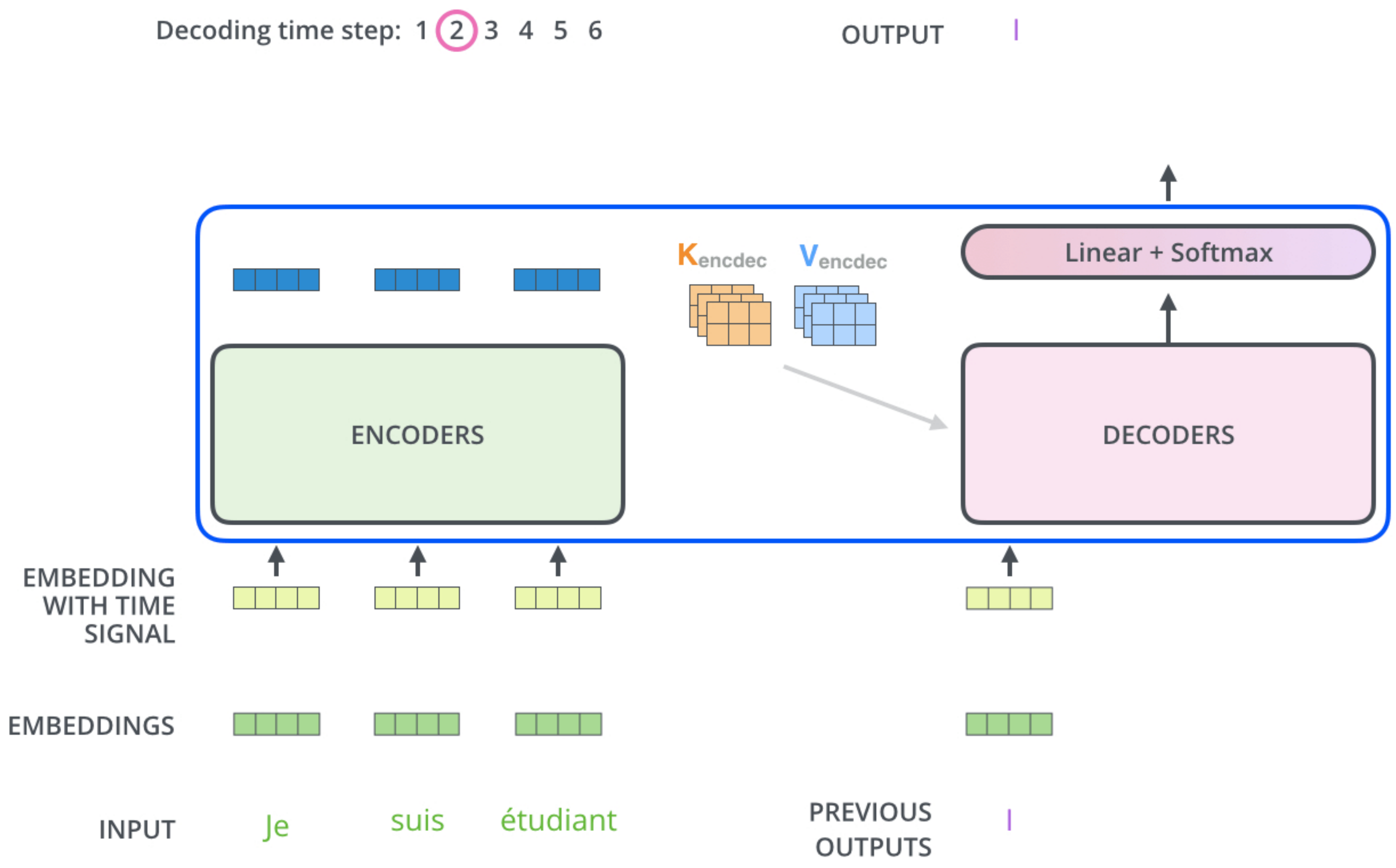
Positional Encoding

- Representing The Order of The Sequence Using Positional Encoding



Decoder

- Decoder for generating sequence data.



Transformer in Vision

Category	Sub-category	Method	Highlights	Publication
Backbone	Supervised pretraining	ViT [55]	Image patches, standard transformer	ICLR 2021
		TNT [85]	Transformer in transformer, local attention	NeurIPS 2021
		Swin [17]	Shifted window, window-based self-attention	ICCV 2021
	Self-supervised pretraining	iGPT [29]	Pixel prediction self-supervised learning, GPT model	ICML 2020
		MoCo v3 [32]	Contrastive self-supervised learning, ViT	ICCV 2021
High/Mid-level vision	Object detection	DETR [19]	Set-based prediction, bipartite matching, transformer	ECCV 2020
		Deformable DETR [291]	DETR, deformable attention module	ICLR 2021
		UP-DETR [49]	Unsupervised pre-training, random query patch detection	CVPR 2021
	Segmentation	Max-DeepLab [228]	PQ-style bipartite matching, dual-path transformer	CVPR 2021
		VisTR [235]	Instance sequence matching and segmentation	CVPR 2021
		SETR [285]	sequence-to-sequence prediction, standard transformer	CVPR 2021
	Pose Estimation	Hand-Transformer [102]	Non-autoregressive transformer, 3D point set	ECCV 2020
		HOT-Net [103]	Structured-reference extractor	MM 2020
		METRO [138]	Progressive dimensionality reduction	CVPR 2021
Low-level vision	Image generation	Image Transformer [171]	Pixel generation using transformer	ICML 2018
		Taming transformer [58]	VQ-GAN, auto-regressive transformer	CVPR 2021
		TransGAN [111]	GAN using pure transformer architecture	arXiv 2021
	Image enhancement	IPT [27]	Multi-task, ImageNet pre-training, transformer model	CVPR 2021
		TTSR [251]	Texture transformer, RefSR	CVPR 2020
Video processing	Video inpainting	STTN [268]	Spatial-temporal adversarial loss	ECCV 2020
	Video captioning	Masked Transformer [288]	Masking network, event proposal	CVPR 2018
Multimodality	Classification	CLIP [180]	NLP supervision for images, zero-shot transfer	arXiv 2021
	Image generation	DALL-E [185]	Zero-shot text-to image generation	ICML 2021
		Cogview [51]	VQ-VAE, Chinese input	arXiv 2021
	Multi-task	UniT [100]	Different NLP & CV tasks, shared model parameters	arXiv 2021
Efficient transformer	Decomposition	ASH [159]	Number of heads, importance estimation	NeurIPS 2019
	Distillation	TinyBert [113]	Various losses for different modules	EMNLP Findings 2020
	Quantization	FullyQT [176]	Fully quantized transformer	EMNLP Findings 2020
	Architecture design	ConvBert [112]	Local dependence, dynamic convolution	NeurIPS 2020

Backbone: iGPT (Self-supervised Learning) by OpenAI

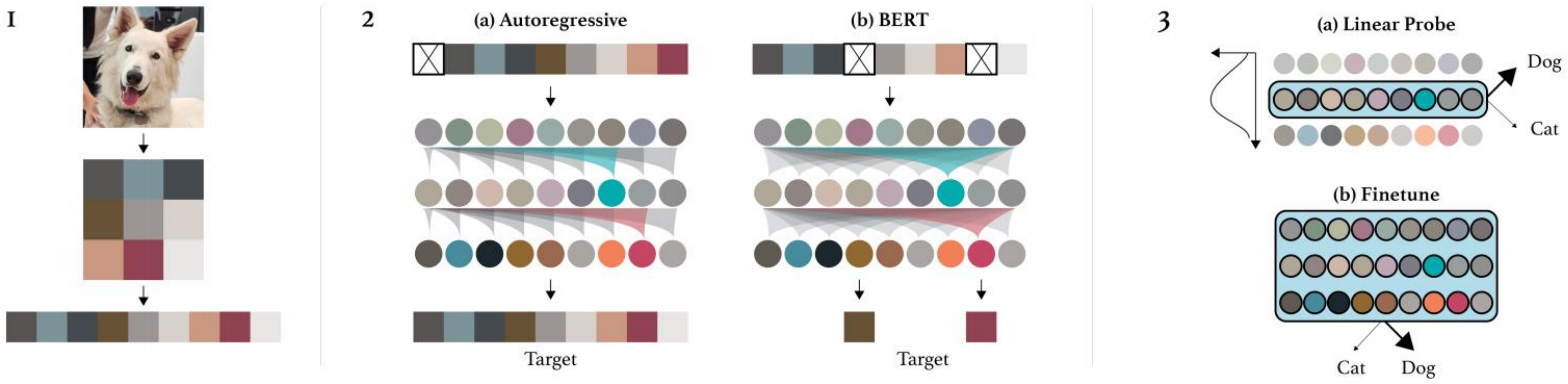


Image DownSample

9-bit color palette to represent pixels

Pre-Training

Decoder
Image generation

Encoder
Image Completion

No positional Encoding!

Classification

https://openai.com/blog/image-gpt/

Backbone: iGPT (Self-supervised Learning) by OpenAI

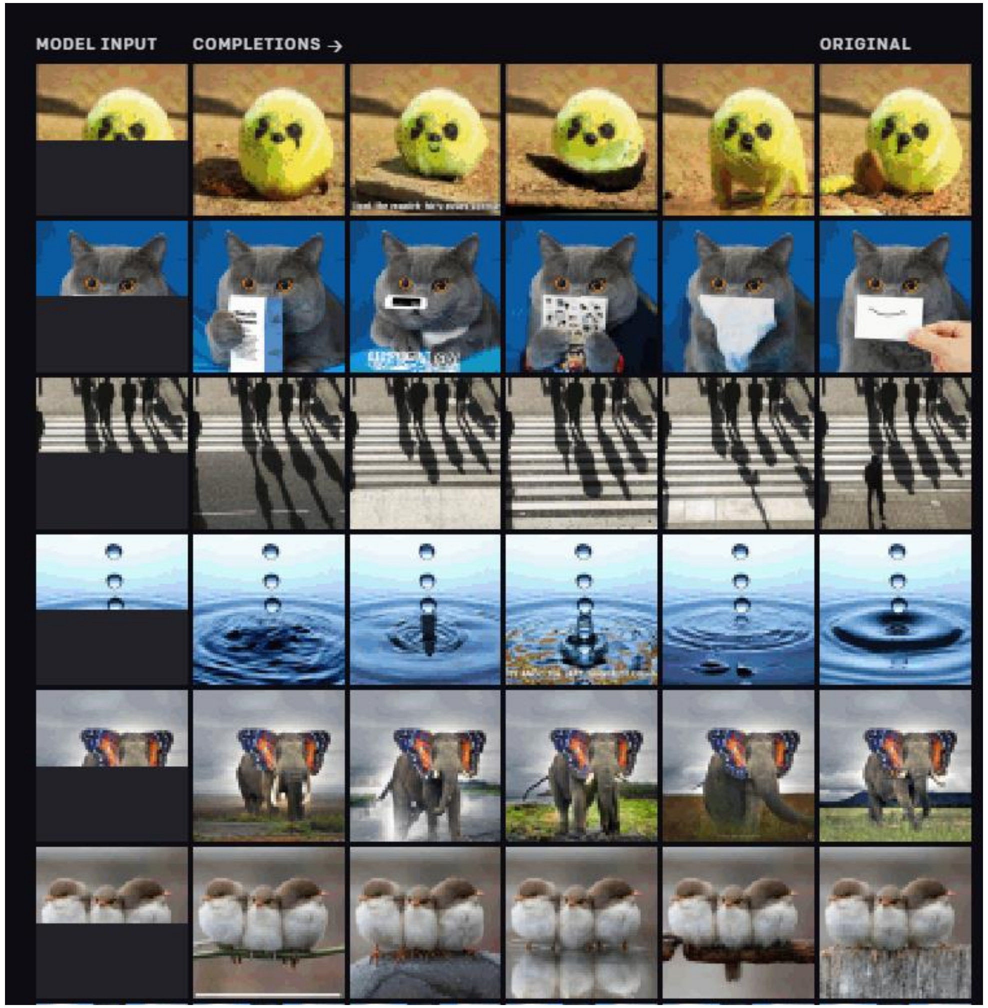


Image completion

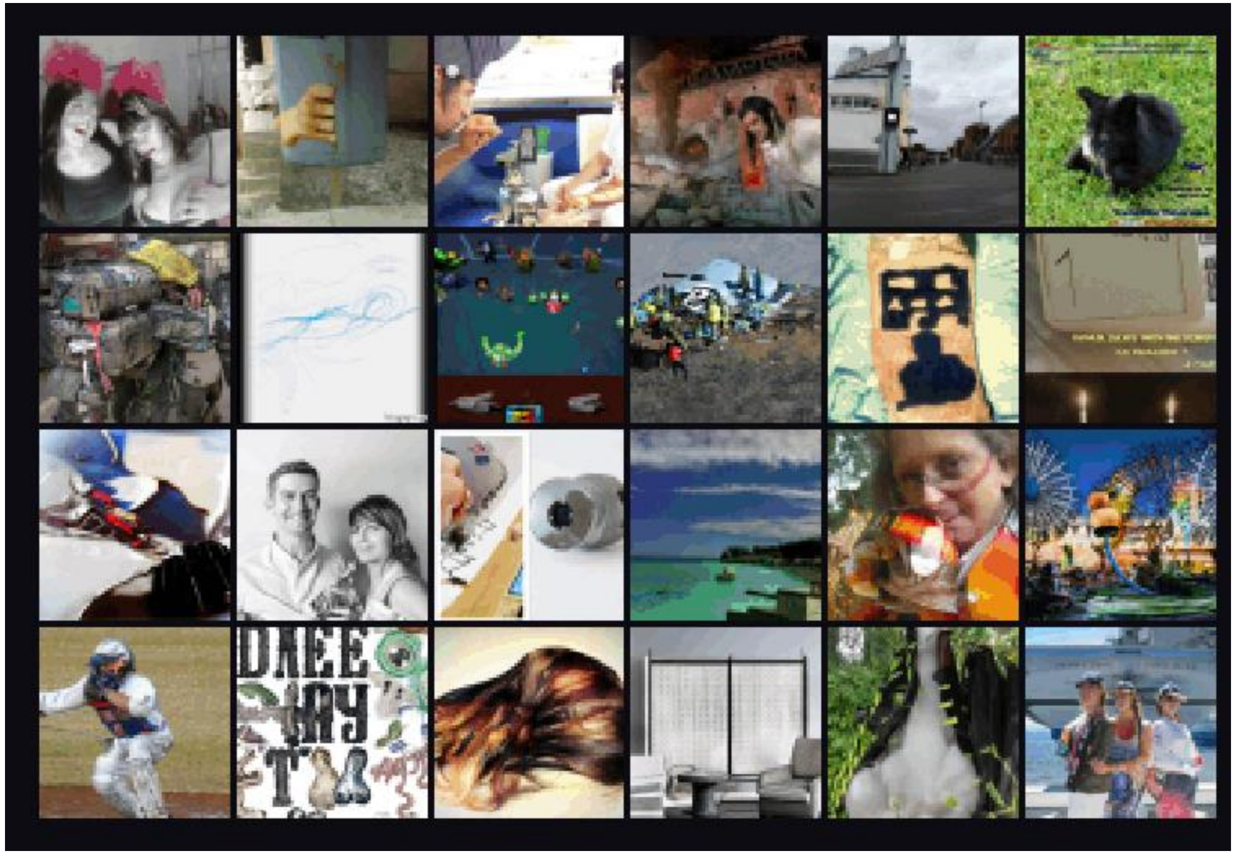


Image generation

Backbone: iGPT (Self-supervised Learning) by OpenAI

EVALUATION	DATASET	OUR RESULT		BEST NON-iGPT RESULT	
Logistic regression on learned features (linear probe)	CIFAR-10	**96.3**	iGPT-L 32x32 w/ 1536 features	95.3	SimCLR[12] w/ 8192 features
	CIFAR-100	**82.8**	iGPT-L 32x32 w/ 1536 features	80.2	SimCLR w/ 8192 features
	STL-10	**95.5**	iGPT-L 32x32 w/ 1536 features	94.2	AMDIM[13] w/ 8192 features
	ImageNet	72.0	iGPT-XL[a] 64x64 w/ 15360 features	**76.5**	SimCLR w/ 8192 features
Full fine-tune	CIFAR-10	**99.0**	iGPT-L 32x32, trained on ImageNet	**99.0**[b]	GPipe,[15] trained on ImageNet
	ImageNet 32x32	66.3	iGPT-L 32x32	**70.2**	Isometric Nets[16]

a. We only show ImageNet linear probe accuracy for iGPT-XL since other experiments did not finish before we needed to transition to different supercomputing facilities.

b. Bit-L,[14] trained on JFT (300M images with 18K classes), achieved a result of 99.3.

Backbone: ViT (Image Classification) by Google

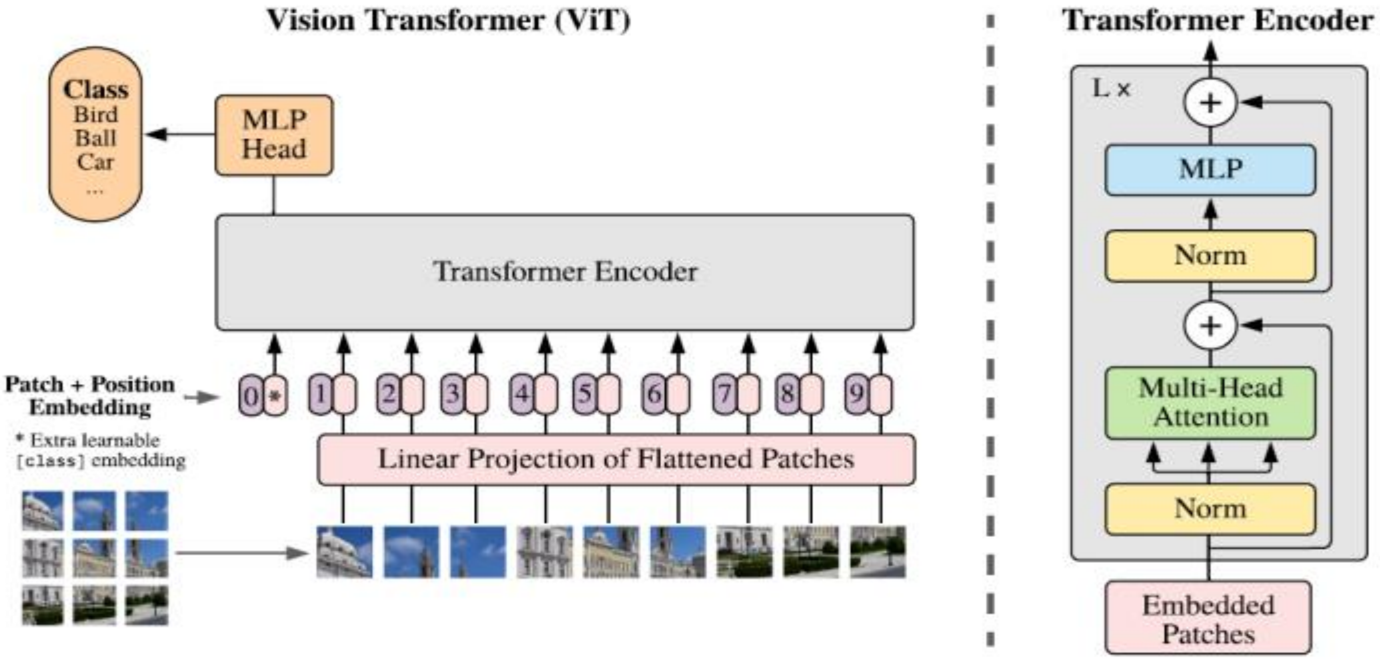


Figure 5: The framework of the Vision Transformer (The image is from [36]).

Backbone: ViT (Image Classification) by Google

- Comparable performance with the best CNN

	Ours (ViT-H/14)	Ours (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	88.36	87.61 ± 0.03	87.54 ± 0.02	88.4/**88.5***
ImageNet ReaL	**90.77**	90.24 ± 0.03	90.54	90.55
CIFAR-10	$\mathbf{99.50} \pm 0.06$	99.42 ± 0.03	99.37 ± 0.06	–
CIFAR-100	$\mathbf{94.55} \pm 0.04$	93.90 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	$\mathbf{97.56} \pm 0.03$	97.32 ± 0.11	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	$\mathbf{99.74} \pm 0.00$	99.63 ± 0.03	–
VTAB (19 tasks)	$\mathbf{77.16} \pm 0.29$	75.91 ± 0.18	76.29 ± 1.70	–
TPUv3-days	2.5k	0.68k	9.9k	12.3k

Table 2: Comparison with state of the art on popular image classification datasets benchmarks. Vision Transformer models pre-trained on the JFT300M dataset often match or outperform ResNet-based baselines while taking substantially less computational resources to pre-train. *Slightly improved 88.5% result reported in Touvron et al. (2020).

Backbone: ViT (Image Classification) by Google

- Bigger data， better Transformer （without inductive bais）

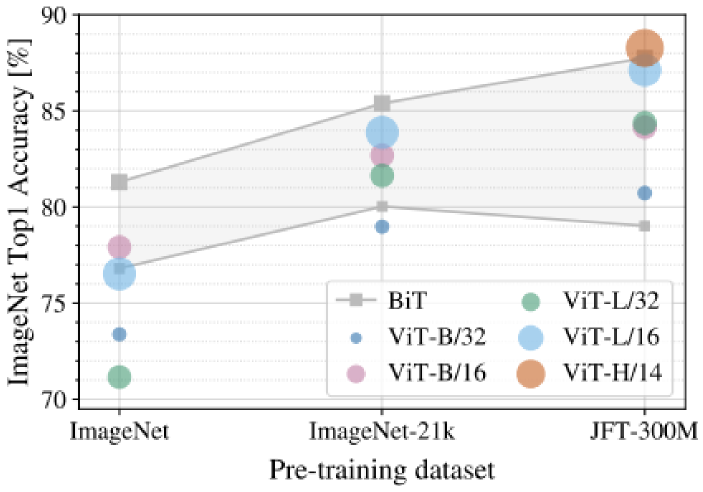


Figure 3: Transfer to ImageNet. While large ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets, they shine when pre-trained on larger datasets. Similarly, larger ViT variants overtake smaller ones as the dataset grows.

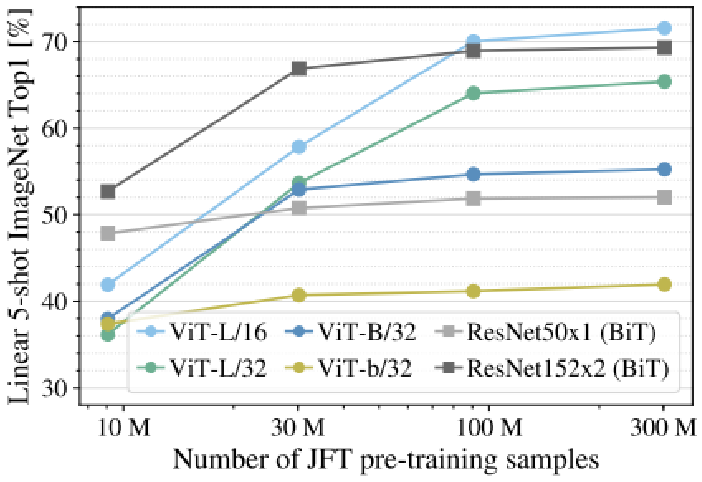


Figure 4: Linear few-shot evaluation on ImageNet versus pre-training size. ResNets perform better with smaller pre-training datasets but plateau sooner than ViT which performs better with larger pre-training. ViT-b is ViT-B with all hidden dimensions halved.

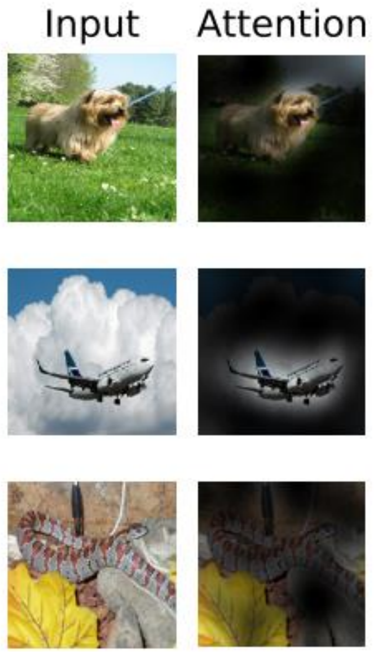


Figure 6: Representative examples of attention from the output token to the input space. See Appendix C.6 for details.

Backbone: DeiT (Image Classification) by Facebook

- Training tricks & knowledge distillation

Ablation on ↓	Pre-training	Fine-tuning	Rand-Augment	AutoAug	Mixup	CutMix	Erasing	Stoch. Depth	Repeated Aug.	Dropout	Exp. Moving Avg.	top-1 accuracy: pre-trained 224^2	top-1 accuracy: fine-tuned 384^2
none: DeiT-B	adamw	adamw	✓	✗	✓	✓	✓	✓	✓	✗	✗	81.8 ±0.2	83.1 ±0.1
optimizer	SGD	adamw	✓	✗	✓	✓	✓	✓	✓	✗	✗	74.5	77.3
	adamw	SGD	✓	✗	✓	✓	✓	✓	✓	✗	✗	81.8	83.1
data augmentation	adamw	adamw	✗	✗	✓	✓	✓	✓	✓	✗	✗	79.6	80.4
	adamw	adamw	✗	✓	✓	✓	✓	✓	✓	✗	✗	81.2	81.9
	adamw	adamw	✓	✗	✗	✓	✓	✓	✓	✗	✗	78.7	79.8
	adamw	adamw	✓	✗	✓	✗	✓	✓	✓	✗	✗	80.0	80.6
	adamw	adamw	✓	✗	✗	✗	✓	✓	✓	✗	✗	75.8	76.7
regularization	adamw	adamw	✓	✗	✓	✓	✗	✓	✓	✗	✗	4.3*	0.1
	adamw	adamw	✓	✗	✓	✓	✓	✗	✓	✗	✗	3.4*	0.1
	adamw	adamw	✓	✗	✓	✓	✓	✓	✗	✗	✗	76.5	77.4
	adamw	adamw	✓	✗	✓	✓	✓	✓	✓	✓	✗	81.3	83.1
	adamw	adamw	✓	✗	✓	✓	✓	✓	✓	✗	✓	81.9	83.1

Table 8: Ablation study on training methods on ImageNet [42]. The top row ("none") corresponds to our default configuration employed for DeiT. The symbols ✓ and ✗ indicates that we use and do not use the corresponding method, respectively. We report the accuracy scores (%) after the initial training at resolution 224×224, and after fine-tuning at resolution 384×384. The hyper-parameters are fixed according to Table 9, and may be suboptimal.

* indicates that the model did not train well, possibly because hyper-parameters are not adapted.

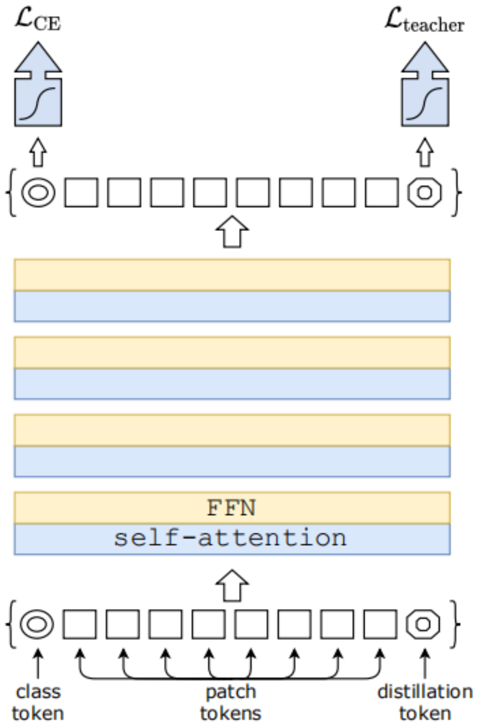
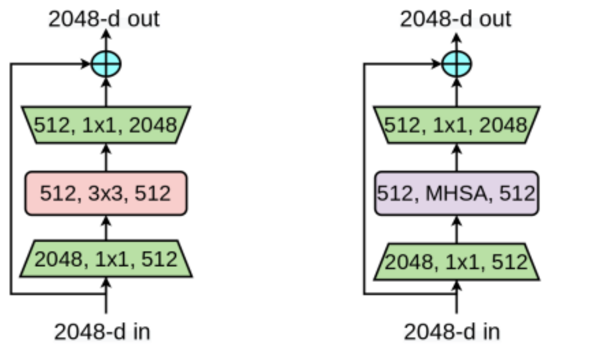
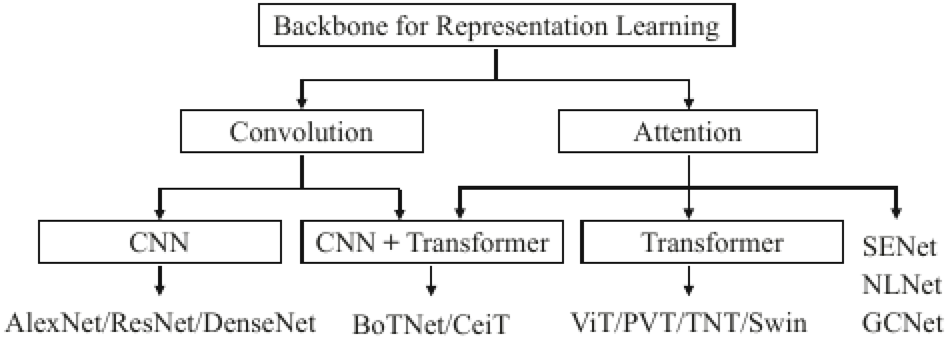


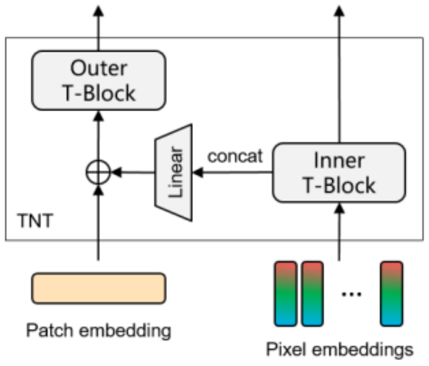
Figure 2: Our distillation procedure: we simply include a new *distillation token*. It interacts with the class and patch tokens through the self-attention layers. This distillation token is employed in a similar fashion as the class token, except that on output of the network its objective is to reproduce the (hard) label predicted by the teacher, instead of true label. Both the class and distillation tokens input to the transformers are learned by back-propagation.

Backbone: Other Variants of ViT

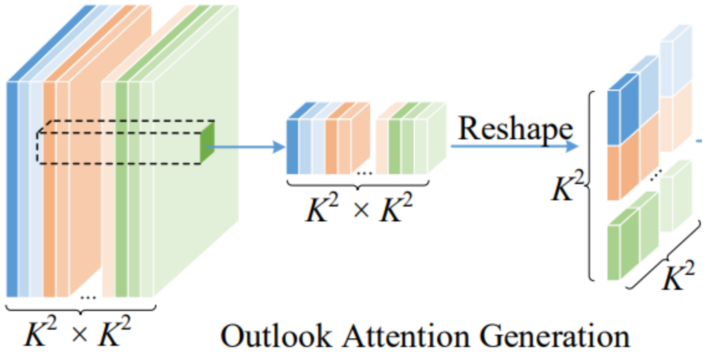


ResNet Bottleneck **Bottleneck Transformer**

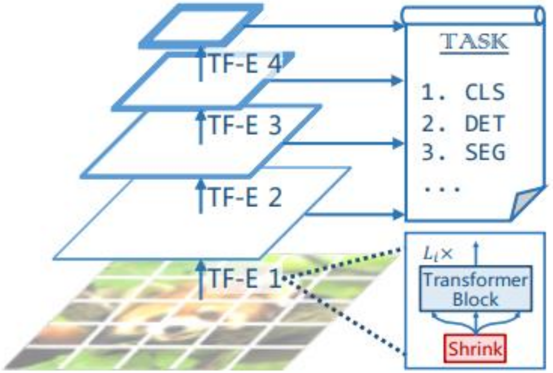




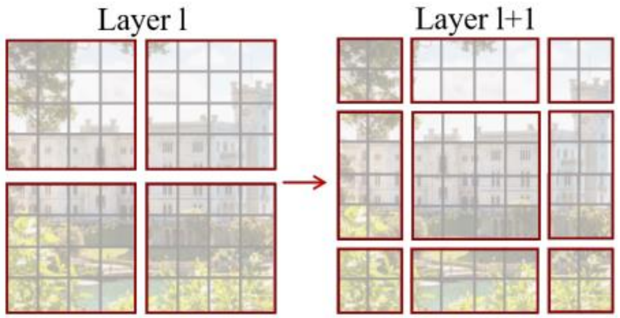
TNT



Outlook Attention Generation



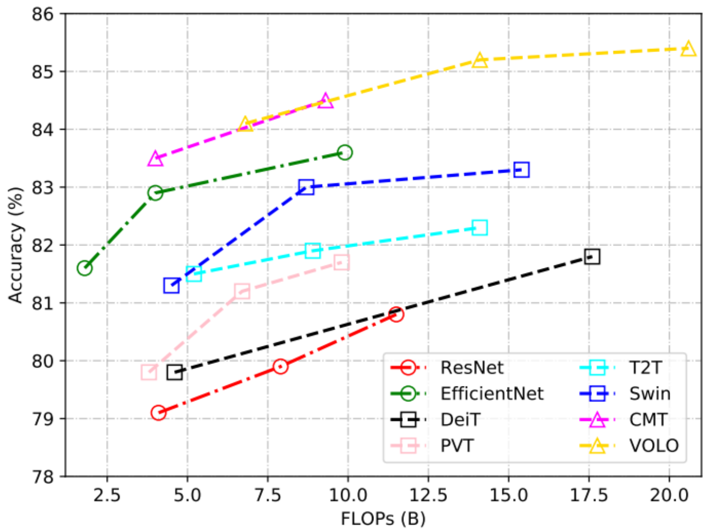
PVT



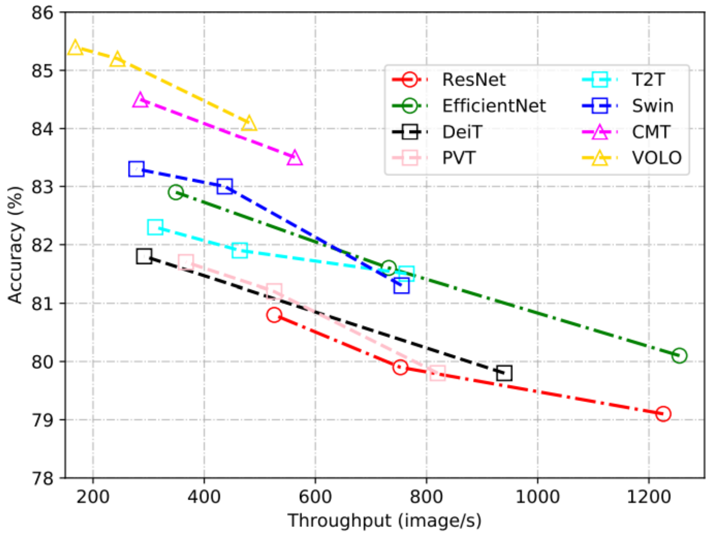
Swin Transformer

Backbone: Comparison

Model	Params (M)	FLOPs (B)	Throughput (image/s)	Top-1 (%)
CNN				
ResNet-50 [89], [260]	25.6	4.1	1226	79.1
ResNet-101 [89], [260]	44.7	7.9	753	79.9
ResNet-152 [89], [260]	60.2	11.5	526	80.8
EfficientNet-B0 [213]	5.3	0.39	2694	77.1
EfficientNet-B1 [213]	7.8	0.70	1662	79.1
EfficientNet-B2 [213]	9.2	1.0	1255	80.1
EfficientNet-B3 [213]	12	1.8	732	81.6
EfficientNet-B4 [213]	19	4.2	349	82.9
Pure Transformer				
DeiT-Ti [55], [219]	5	1.3	2536	72.2
DeiT-S [55], [219]	22	4.6	940	79.8
DeiT-B [55], [219]	86	17.6	292	81.8
T2T-ViT-14 [260]	21.5	5.2	764	81.5
T2T-ViT-19 [260]	39.2	8.9	464	81.9
T2T-ViT-24 [260]	64.1	14.1	312	82.3
PVT-Small [232]	24.5	3.8	820	79.8
PVT-Medium [232]	44.2	6.7	526	81.2
PVT-Large [232]	61.4	9.8	367	81.7
TNT-S [85]	23.8	5.2	428	81.5
TNT-B [85]	65.6	14.1	246	82.9
CPVT-S [44]	23	4.6	930	80.5
CPVT-S-GAP [44]	23	4.6	942	81.5
CPVT-B [44]	88	17.6	285	82.3
Swin-T [148]	29	4.5	755	81.3
Swin-S [148]	50	8.7	437	83.0
Swin-B [148]	88	15.4	278	83.3
CNN + Transformer				
Twins-SVT-S [43]	24	2.9	1059	81.7
Twins-SVT-B [43]	56	8.6	469	83.2
Twins-SVT-L [43]	99.2	15.1	288	83.7
Shuffle-T [105]	29	4.6	791	82.5
Shuffle-S [105]	50	8.9	450	83.5
Shuffle-B [105]	88	15.6	279	84.0
XCiT-S12/16 [56]	26	4.8	781	83.3
CMT-S [77]	25.1	4.0	563	83.5
CMT-B [77]	45.7	9.3	285	84.5
VOLO-D1 [261]	27	6.8	481	84.2
VOLO-D2 [261]	59	14.1	244	85.2
VOLO-D3 [261]	86	20.6	168	85.4
VOLO-D4 [261]	193	43.8	100	85.7
VOLO-D5 [261]	296	69.0	64	86.1



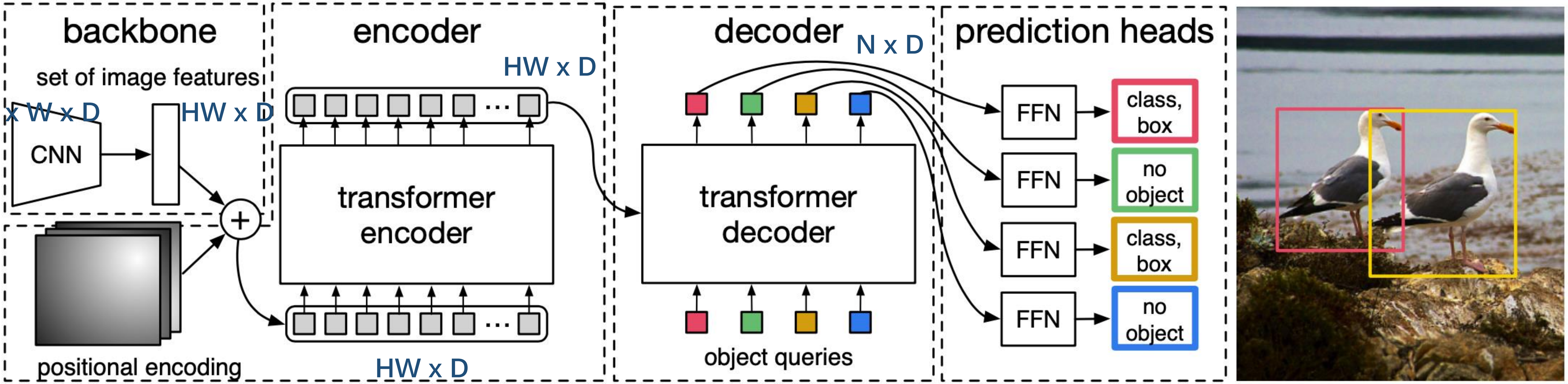
FLOPs对比



速度对比

DETR (Object Detection) by Facebook

H’ x W’ x 3



(c, x, y, h, w)

Magic Here!

N x D

Initially Random, Totally Learnt

DETR (Object Detection) by Facebook

- **Bipartite Matching Loss**

Predefined N=5

Prediction	*Ground Truth*
(Bird, x1, y1, h1, w1)	(Bird, x1, y1, h1, w1)
(Bird, x2, y2, h2, w2)	(Bird, x2, y2, h2, w2)
(Bird, x3, y3, h3, w3)	(None, x3, y3, h3, w3)
(None, x4, y4, h4, w4)	(None, x4, y4, h4, w4)
(None, x5, y5, h5, w5)	(None, x5, y5, h5, w5)

Step 1: Find optimal assignment

Step 2: Compute total loss for training

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}}(i)) \right]$$

DETR (Object Detection) by Facebook

Table 1: Comparison with Faster R-CNN with a ResNet-50 and ResNet-101 backbones on the COCO validation set. The top section shows results for Faster R-CNN models in Detectron2 [50], the middle section shows results for Faster R-CNN models with GIoU [38], random crops train-time augmentation, and the long `9x` training schedule. DETR models achieve comparable results to heavily tuned Faster R-CNN baselines, having lower AP_S but greatly improved AP_L. We use torchscript Faster R-CNN and DETR models to measure FLOPS and FPS. Results without R101 in the name correspond to ResNet-50.

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**

DETR (Object Detection) by Facebook

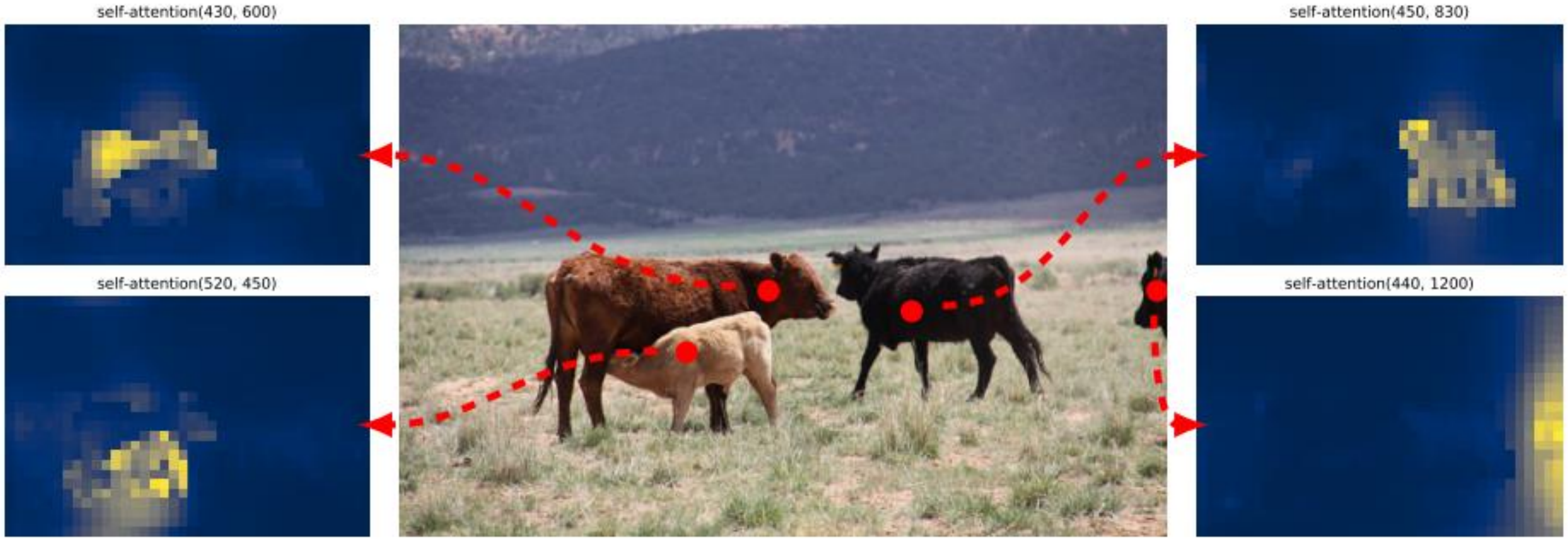
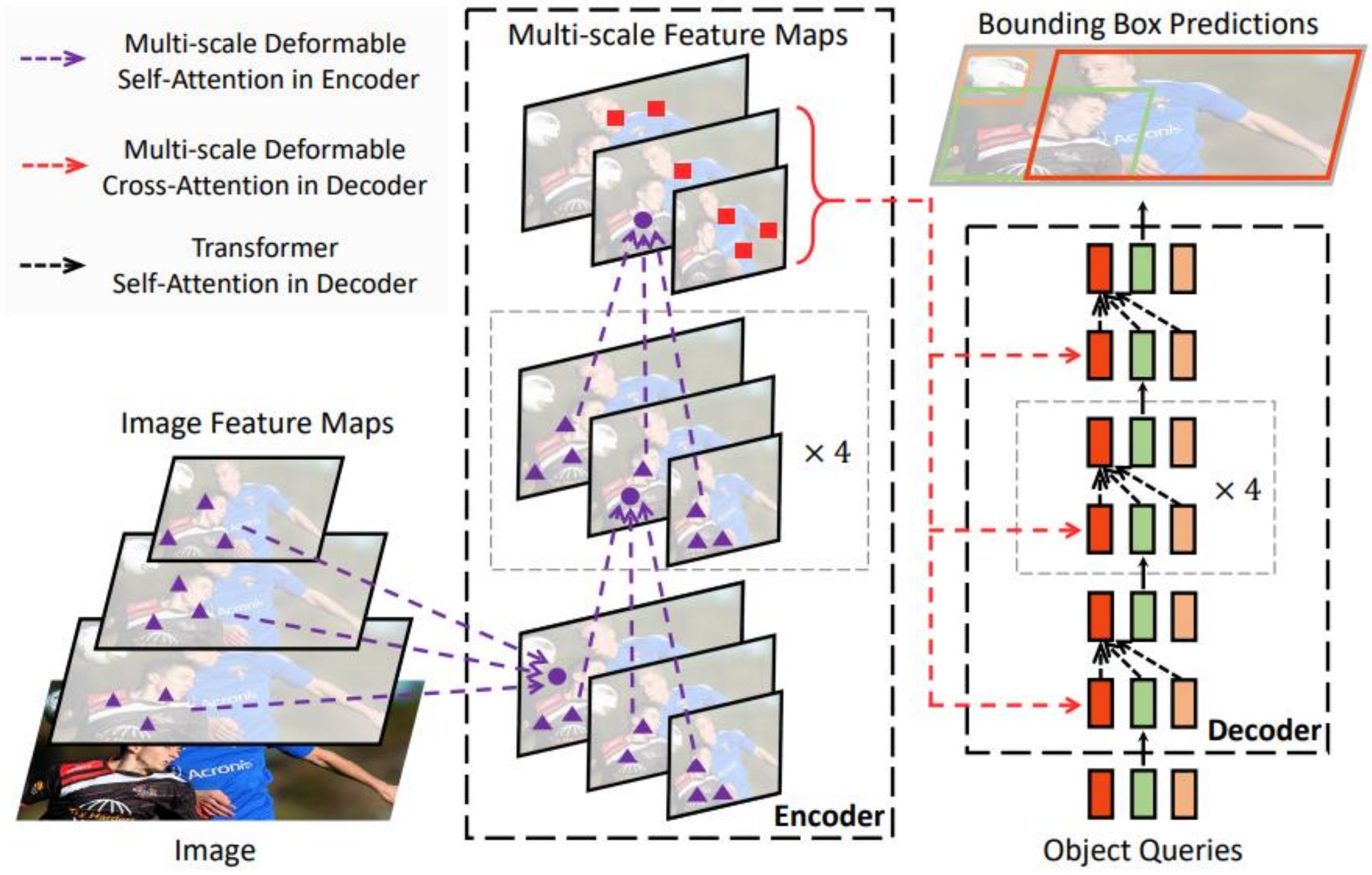


Fig. 3: Encoder self-attention for a set of reference points. The encoder is able to separate individual instances. Predictions are made with baseline DETR model on a validation set image.

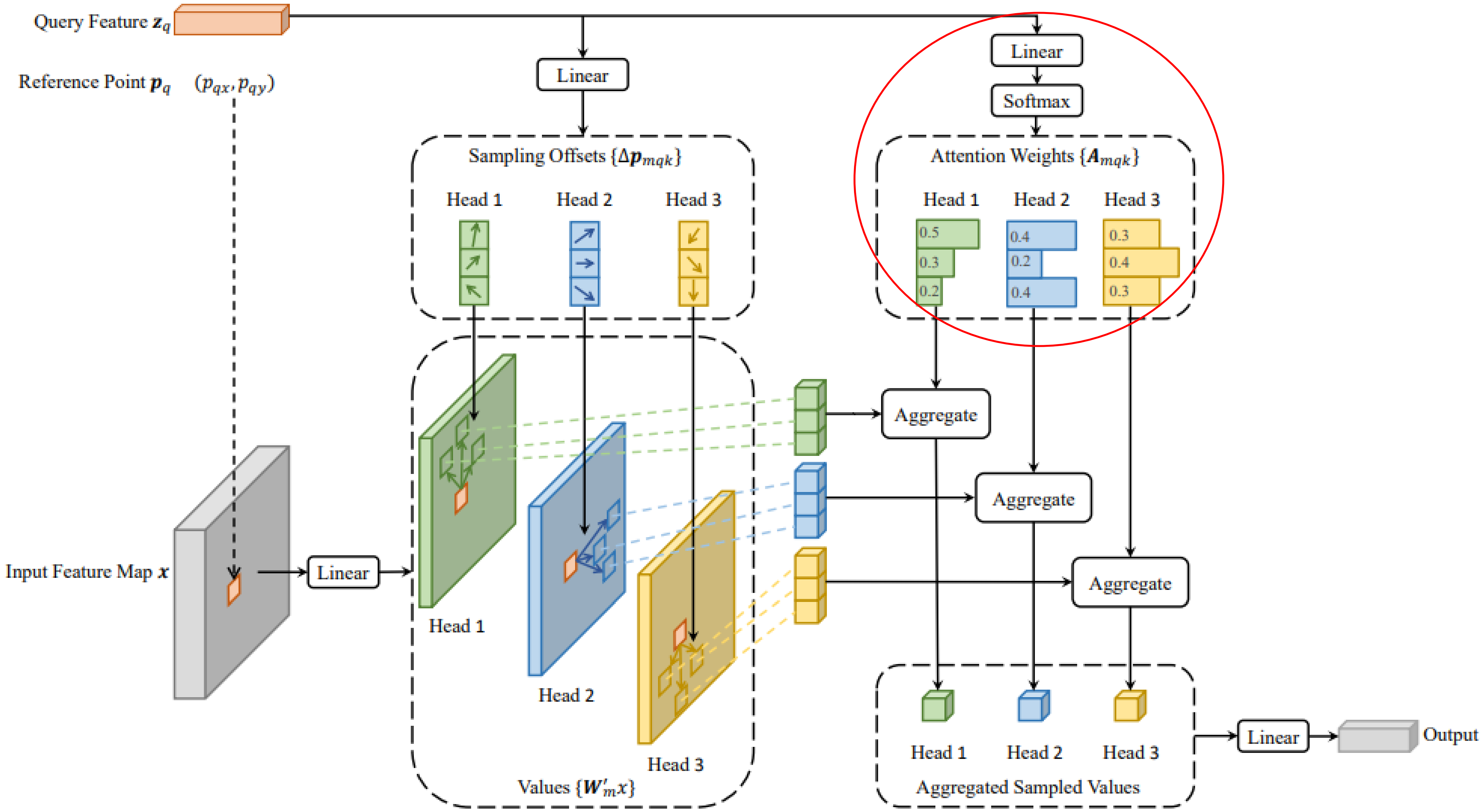
Deformable DETR (Object Detection) by SenseTime

- Deformable Self-Attention (10x faster)
- Multi-scale Feature



Deformable DETR (Object Detection) by SenseTime

For a query，sample K*M points



Deformable DETR (Object Detection) by SenseTime

Method	Epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	#Params (M)	GFLOPs	FPS
CNN based										
FCOS [147]	36	41.0	59.8	44.1	26.2	44.6	52.2	-	177	$23^{\dagger}$
Faster R-CNN + FPN [127]	109	42.0	62.1	45.5	26.6	45.4	53.4	42	180	26
Transformer based										
DETR [15]	500	42.0	62.4	44.2	20.5	45.8	61.1	41	86	28
DETR-DC5 [15]	500	43.3	63.1	45.9	22.5	47.3	61.1	41	187	12
Deformable DETR [193]	50	46.2	65.2	50.0	28.8	49.2	61.7	40	173	19
TSP-FCOS [143]	36	43.1	62.3	47.0	26.6	46.8	55.9	-	189	$20^{\dagger}$
TSP-RCNN [143]	96	45.0	64.5	49.6	29.7	47.7	58.0	-	188	$15^{\dagger}$
ACT+MKKD (L=32) [189]	-	43.1	-	-	61.4	47.1	22.2	-	169	$14^{\dagger}$
ACT+MKKD (L=16) [189]	-	40.6	-	-	59.7	44.3	18.5	-	156	$16^{\dagger}$
ViT-B/16-FRCNN‡ [8]	21	36.6	56.3	39.3	17.4	40.0	55.5	-	-	-
ViT-B/16-FRCNN* [8]	21	37.8	57.4	40.1	17.8	41.4	57.3	-	-	-
UP-DETR [33]	150	40.5	60.8	42.6	19.0	44.4	60.0	41	-	-
UP-DETR [33]	300	42.8	63.0	45.3	20.8	47.1	61.7	41	-	-

Transformer for Detection: Comparison

Method	Epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	#Params (M)	GFLOPs	FPS
CNN based										
FCOS [216]	36	41.0	59.8	44.1	26.2	44.6	52.2	-	177	$23^{\dagger}$
Faster R-CNN + FPN [186]	109	42.0	62.1	45.5	26.6	45.4	53.4	42	180	26
CNN Backbone + Transformer Head										
DETR [19]	500	42.0	62.4	44.2	20.5	45.8	61.1	41	86	28
DETR-DC5 [19]	500	43.3	63.1	45.9	22.5	47.3	61.1	41	187	12
Deformable DETR [291]	50	46.2	65.2	50.0	28.8	49.2	61.7	40	173	19
TSP-FCOS [210]	36	43.1	62.3	47.0	26.6	46.8	55.9	-	189	$20^{\dagger}$
TSP-RCNN [210]	96	45.0	64.5	49.6	29.7	47.7	58.0	-	188	$15^{\dagger}$
ACT+MKKD (L=32) [284]	-	43.1	-	-	61.4	47.1	22.2	-	169	$14^{\dagger}$
ACT+MKKD (L=16) [284]	-	40.6	-	-	59.7	44.3	18.5	-	156	$16^{\dagger}$
SMCA [71]	108	45.6	65.5	49.1	25.9	49.3	62.6	-	-	-
Efficient DETR [257]	36	45.1	63.1	49.1	28.3	48.4	59.0	35	210	-
UP-DETR [49]	150	40.5	60.8	42.6	19.0	44.4	60.0	41	-	-
UP-DETR [49]	300	42.8	63.0	45.3	20.8	47.1	61.7	41	-	-
Transformer Backbone + CNN Head										
ViT-B/16-FRCNN‡ [10]	21	36.6	56.3	39.3	17.4	40.0	55.5	-	-	-
ViT-B/16-FRCNN* [10]	21	37.8	57.4	40.1	17.8	41.4	57.3	-	-	-
PVT-Small+RetinaNet [232]	12	40.4	61.3	43.0	25.0	42.9	55.7	34.2	118	-
Twins-SVT-S+RetinaNet [43]	12	43.0	64.2	46.3	28.0	46.4	57.5	34.3	104	-
Swin-T+RetinaNet [148]	12	41.5	62.1	44.2	25.1	44.9	55.5	38.5	118	-
Swin-T+ATSS [148]	36	47.2	66.5	51.3	-	-	-	36	215	-
Pure Transformer based										
PVT-Small+DETR [232]	50	34.7	55.7	35.4	12.0	36.4	56.7	40	-	-
TNT-S+DETR [85]	50	38.2	58.9	39.4	15.5	41.1	58.8	39	-	-
YOLOS-Ti [64]	300	30.0	-	-	-	-	-	6.5	21	-
YOLOS-S [64]	150	37.6	57.6	39.2	15.9	40.2	57.3	28	179	-
YOLOS-B [64]	150	42.0	62.2	44.5	19.5	45.3	62.1	127	537	-

SETR (Semantic Segmentation) by Fudan Univ.

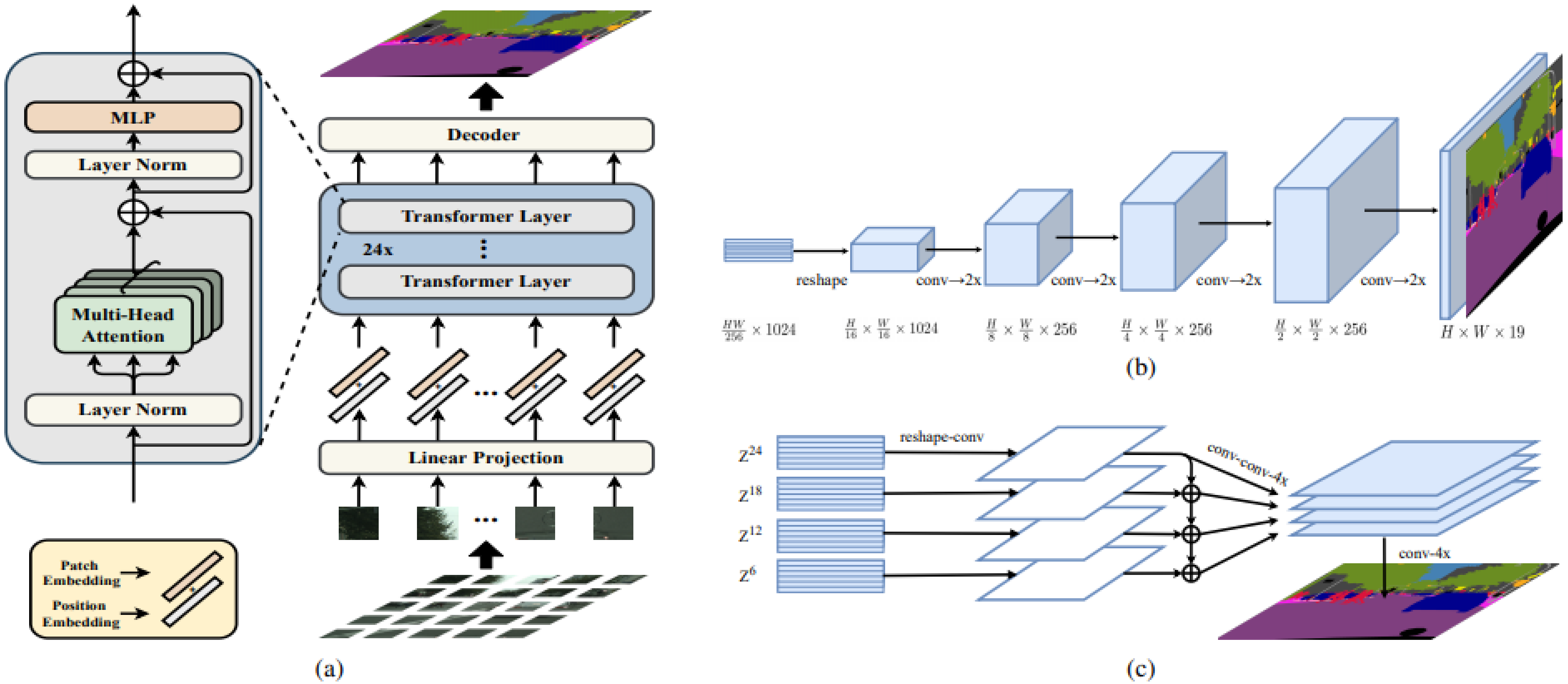


Figure 1. **Schematic illustration of the proposed *SEgmentation TRansformer* (SETR)** (a). We first split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. To perform pixel-wise segmentation, 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*).

SETR (Semantic Segmentation) by Fudan Univ.

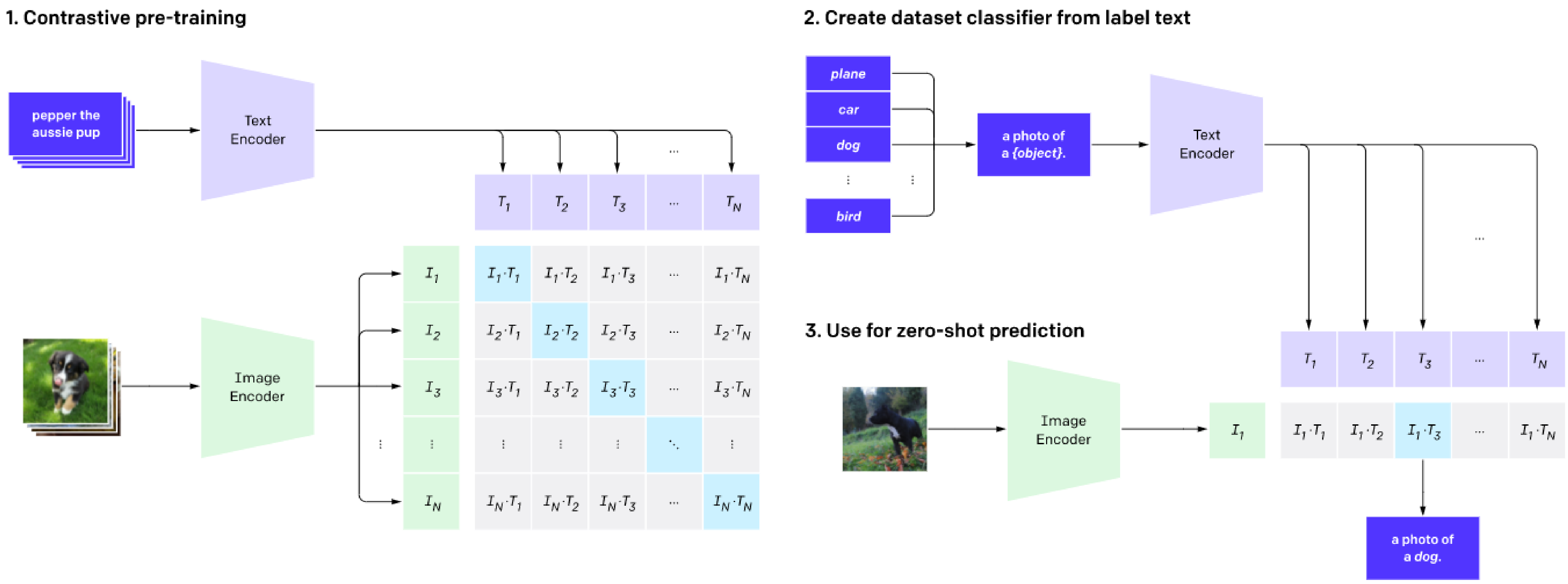
Method	Pre	Backbone	#Params	40k	80k
FCN [39]	1K	R-101	68.59	73.93	75.52
Semantic FPN [39]	1K	R-101	47.51	-	75.80
Hybrid-Base	R	T-Base	112.59	74.48	77.36
Hybrid-Base	21K	T-Base	112.59	76.76	76.57
Hybrid-DeiT	21K	T-Base	112.59	77.42	78.28
SETR-*Naïve*	21K	T-Large	305.67	77.37	77.90
SETR-*MLA*	21K	T-Large	310.57	76.65	77.24
SETR-*PUP*	21K	T-Large	318.31	78.39	79.34
SETR-*PUP*	R	T-Large	318.31	42.27	-
SETR-*Naïve-Base*	21K	T-Base	87.69	75.54	76.25
SETR-*MLA-Base*	21K	T-Base	92.59	75.60	76.87
SETR-*PUP-Base*	21K	T-Base	97.64	76.71	78.02
SETR-*Naïve-DeiT*	1K	T-Base	87.69	77.85	78.66
SETR-*MLA-DeiT*	1K	T-Base	92.59	78.04	78.98
SETR-*PUP-DeiT*	1K	T-Base	97.64	**78.79**	**79.45**

Table 2. **Comparing SETR variants** on different pre-training strategies and backbones. All experiments are trained on Cityscapes train fine set with batch size 8, and evaluated using the single scale test protocol on the Cityscapes validation set in mean IoU (%) rate. "Pre" denotes the pre-training of transformer part. "R" means the transformer part is randomly initialized.

Method	Backbone	mIoU	Pixel Acc.
FCN (16, 160k, SS) [39]	ResNet-101	39.91	79.52
FCN (16, 160k, MS) [39]	ResNet-101	41.40	80.65
EncNet [54]	ResNet-101	44.65	81.69
PSPNet [59]	ResNet-269	44.94	81.69
DMNet [18]	ResNet-101	45.50	-
CCNet [25]	ResNet-101	45.22	-
Strip pooling [23]	ResNet-101	45.60	82.09
APCNet [19]	ResNet-101	45.38	-
OCNet [53]	ResNet-101	45.45	-
SETR-*Naïve* (16, 160k, SS)	T-Large	48.06	82.40
SETR-*Naïve* (16, 160k, MS)	T-Large	48.80	82.92
SETR-*PUP* (16, 160k, SS)	T-Large	48.58	82.90
SETR-*PUP* (16, 160k, MS)	T-Large	50.09	**83.58**
SETR-*MLA* (16, 160k, SS)	T-Large	48.64	82.64
SETR-*MLA* (16, 160k, MS)	T-Large	**50.28**	83.46

Table 4. **State-of-the-art comparison on the ADE20K dataset.** Performances of different model variants are reported. SS: Single-scale inference. MS: Multi-scale inference.

CLIP (Connecting Text and Images) by OpenAI



https://openai.com/blog/clip/

CLIP (Connecting Text and Images) by OpenAI

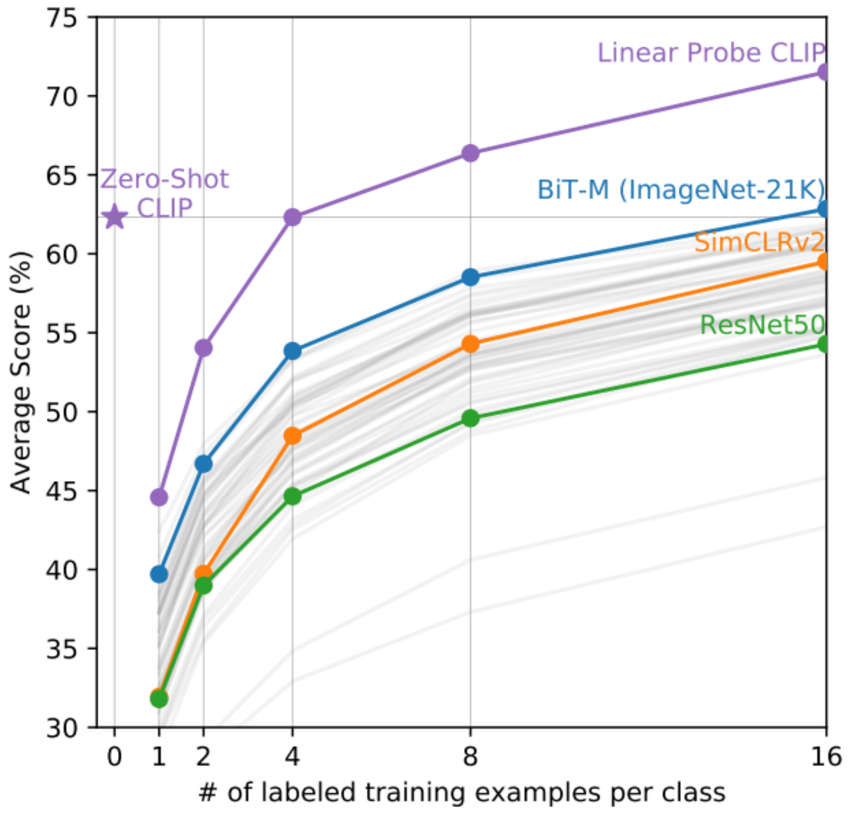
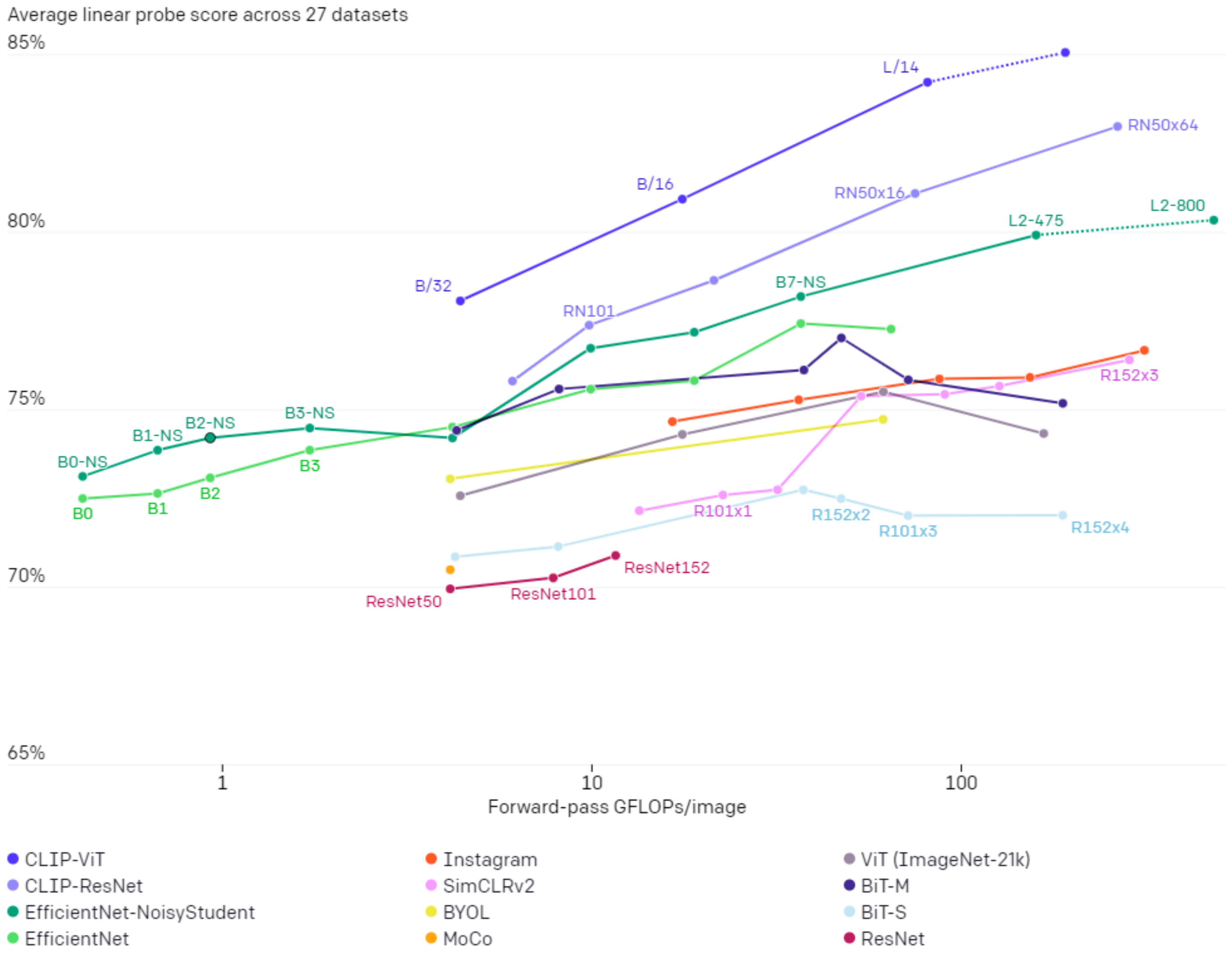
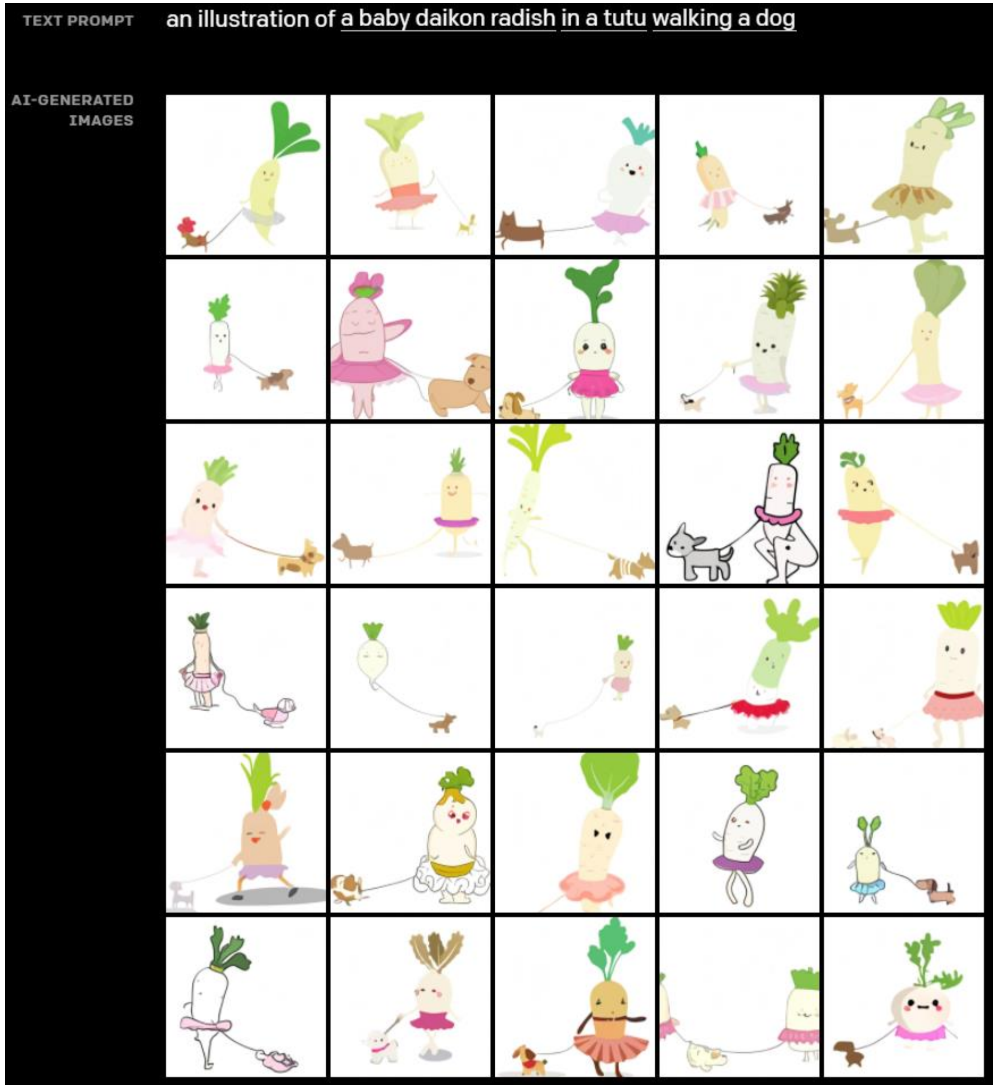
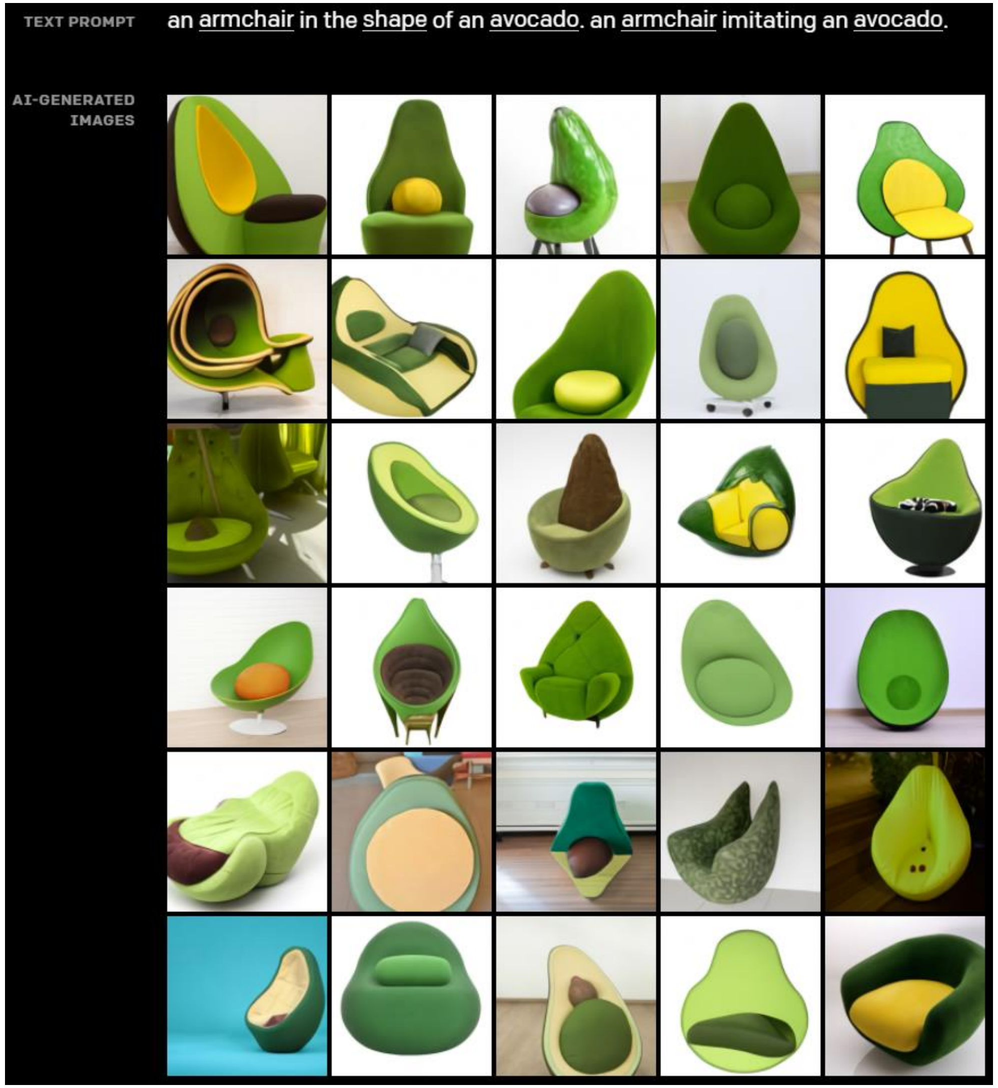


Figure 6. **Zero-shot CLIP outperforms few-shot linear probes.**



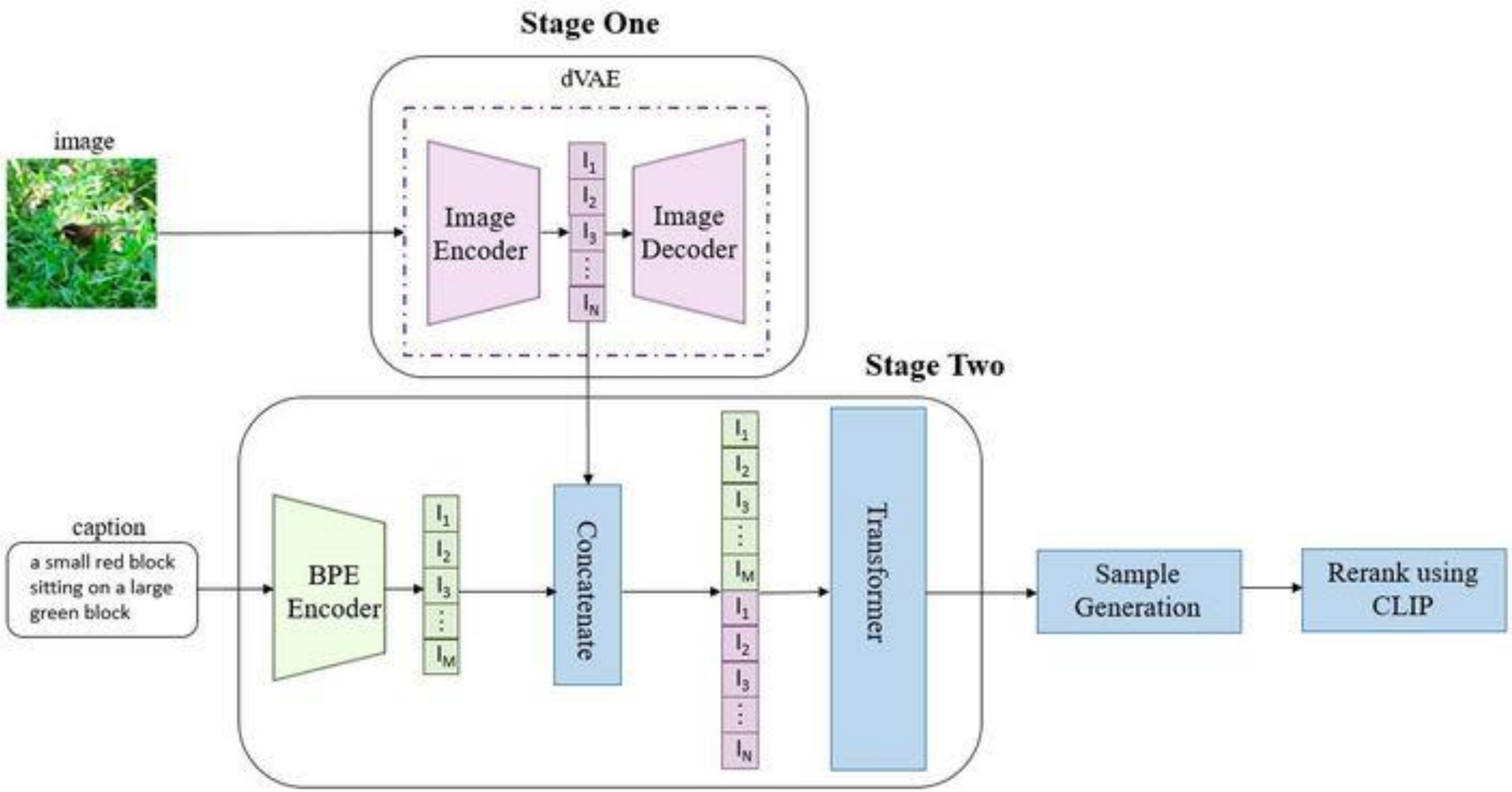
DALL·E (Creating Images from Text) by OpenAI





https://openai.com/blog/dall-e/

DALL·E (Creating Images from Text) by OpenAI

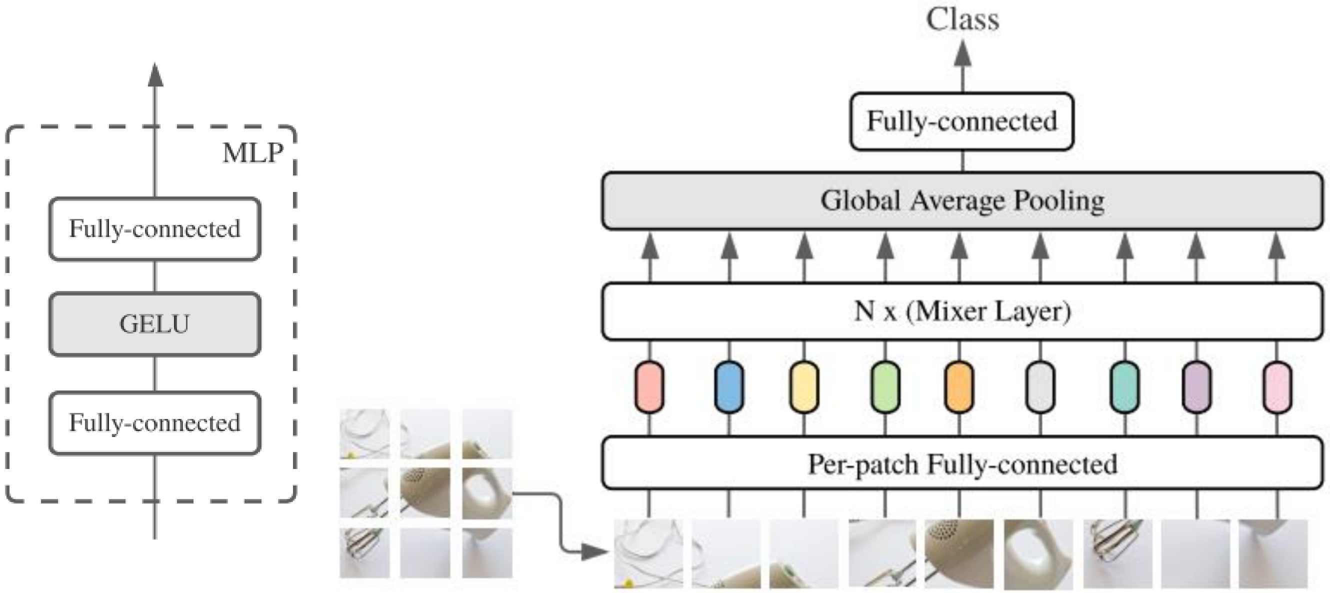


Transformer in Vision

Category	Sub-category	Method	Highlights	Publication
Backbone	Supervised pretraining	ViT [55]	Image patches, standard transformer	ICLR 2021
		TNT [85]	Transformer in transformer, local attention	NeurIPS 2021
		Swin [17]	Shifted window, window-based self-attention	ICCV 2021
	Self-supervised pretraining	iGPT [29]	Pixel prediction self-supervised learning, GPT model	ICML 2020
		MoCo v3 [32]	Contrastive self-supervised learning, ViT	ICCV 2021
High/Mid-level vision	Object detection	DETR [19]	Set-based prediction, bipartite matching, transformer	ECCV 2020
		Deformable DETR [291]	DETR, deformable attention module	ICLR 2021
		UP-DETR [49]	Unsupervised pre-training, random query patch detection	CVPR 2021
	Segmentation	Max-DeepLab [228]	PQ-style bipartite matching, dual-path transformer	CVPR 2021
		VisTR [235]	Instance sequence matching and segmentation	CVPR 2021
		SETR [285]	sequence-to-sequence prediction, standard transformer	CVPR 2021
	Pose Estimation	Hand-Transformer [102]	Non-autoregressive transformer, 3D point set	ECCV 2020
		HOT-Net [103]	Structured-reference extractor	MM 2020
		METRO [138]	Progressive dimensionality reduction	CVPR 2021
Low-level vision	Image generation	Image Transformer [171]	Pixel generation using transformer	ICML 2018
		Taming transformer [58]	VQ-GAN, auto-regressive transformer	CVPR 2021
		TransGAN [111]	GAN using pure transformer architecture	arXiv 2021
	Image enhancement	IPT [27]	Multi-task, ImageNet pre-training, transformer model	CVPR 2021
		TTSR [251]	Texture transformer, RefSR	CVPR 2020
Video processing	Video inpainting	STTN [268]	Spatial-temporal adversarial loss	ECCV 2020
	Video captioning	Masked Transformer [288]	Masking network, event proposal	CVPR 2018
Multimodality	Classification	CLIP [180]	NLP supervision for images, zero-shot transfer	arXiv 2021
	Image generation	DALL-E [185]	Zero-shot text-to image generation	ICML 2021
		Cogview [51]	VQ-VAE, Chinese input	arXiv 2021
	Multi-task	UniT [100]	Different NLP & CV tasks, shared model parameters	arXiv 2021
Efficient transformer	Decomposition	ASH [159]	Number of heads, importance estimation	NeurIPS 2019
	Distillation	TinyBert [113]	Various losses for different modules	EMNLP Findings 2020
	Quantization	FullyQT [176]	Fully quantized transformer	EMNLP Findings 2020
	Architecture design	ConvBert [112]	Local dependence, dynamic convolution	NeurIPS 2020

Transformer in Vision

- **Transformer统一CV/MLP？其他神经网络形态（MLP）异军突起？**



MLP-Mixer (2021)：
只需要MLP的神经网络

Conclusion

- Paper:
  - A survey on vision transformer (https://arxiv.org/abs/2012.12556)
- Related papers:
  - Transformer in Transformer. NeurIPS 2021.
  - (https://arxiv.org/abs/2103.00112)
  - Augmented Shortcuts for Vision Transformers . NeurIPS 2021.
  - (https://arxiv.org/abs/2106.15941)
  - Post-Training Quantization for Vision Transformer . NeurIPS 2021.
  - (https://arxiv.org/abs/2106.14156)
- Code:
  - https://github.com/huawei-noah/CV-Backbones

Code

- 小广告
  - 华为诺亚方舟实验室【校招、实习】
  - 诚聘计算机视觉、模型压缩、AI系统开发相关
  - 简历投递：kai.han@huawei.com

Thanks

kai.han@huawei.com