Transformers for image modeling



Two dogs running in a field

Transformers for image modeling: Outline

- Architectures
- Self-supervised learning
- Autoregressive text-to-image generation
- Trends

Self-attention for images

• Idea: self-attention produces adaptive receptive fields that can better capture non-local structure



H. Zhang, I. Goodfellow, D. Metaxas, A. Odena. Self-Attention Generative Adversarial Networks. ICML 2019

Self-attention for images



H. Zhang, I. Goodfellow, D. Metaxas, A. Odena. Self-Attention Generative Adversarial Networks. ICML 2019

Image GPT – OpenAl



M. Chen et al. Generative pretraining from pixels. ICML 2020

Image GPT

- Image resolution up to 64x64, color values quantized to 512 levels (9 bits), dense attention
- For transfer learning, average-pool encoded features across all positions





Figure 4. Comparison of auto-regressive pre-training with BERT pre-training using iGPT-L at an input resolution of $32^2 \times 3$. Blue bars display linear probe accuracy and orange bars display fine-tune accuracy. Bold colors show the performance boost from ensembling BERT masks. We see that auto-regressive models produce much better features than BERT models after pre-training, but BERT models catch up after fine-tuning.

M. Chen et al. Generative pretraining from pixels. ICML 2020

Vision transformer (ViT) – Google

- Split an image into patches, feed linearly projected patches into standard transformer encoder
 - With patches of 14x14 pixels, you need 16x16=256 patches to represent 224x224 images



A. Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. ICLR 2021

Vision transformer (ViT)

• Trained in a supervised fashion, fine-tuned on ImageNet



A. Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. ICLR 2021

Hierarchical transformer: Swin



(a) Swin Transformer (ours)

(b) ViT

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 selfattention globally.

Z. Liu et al. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. ICCV 2021

Hierarchical transformer: Swin





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.

Z. Liu et al. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. ICCV 2021

Swin results

(a) Regu	lar Im	ageNet-	1K traiı	ned models	
mathod	image	#norom		throughput	ImageNet
meulou	size	#param.	FLOFS	(image / s)	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
(b) Ima	ageNet	t-22K pr	e-traine	ed models	
method	image	#norom		throughput	ImageNet
meulou	size	#param.	FLOFS	(image / s)	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

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].

384² 197M 103.9G

88M 47.0G

 384^{2}

Swin-B

Swin-L

COCO detection and segmentation

(a) Various frameworks							
Method	Backbone	APbox	AP_{50}^{box}	AP_{75}^{box}	#param.	FLOPs	FPS
Cascade	R-50	46.3	64.3	50.5	82M	739G	18.0
Mask R-CNN	Swin-T	50.5	69.3	54.9	86M	745G	15.3
4755	R-50	43.5	61.9	47.0	32M	205G	28.3
A155	Swin-T	47.2	66.5	51.3	36M	215G	22.3
Don Doints V2	R-50	46.5	64.6	50.3	42M	274G	13.6
RepPoints v 2	Swin-T	50.0	68.5	54.2	45M	283G	12.0
Sparse	R-50	44.5	63.4	48.2	106M	166G	21.0
R-CNN	Swin-T	47.9	67.3	52.3	110M	172G	18.4
			0	1 14			

(b) Various backbones w. Cascade Mask R-CNN

86.4

87.3

84.7

42.1

	AP ^{box}	AP ₅₀ ^{box}	AP ₇₅ ^{box}	AP ^{mask}	AP ₅₀ ^{mask}	AP ₇₅ ^{mask}	param	FLOPs	FPS
DeiT-S [†]	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	50.5	69.3	54.9	43.7	66.6	47.1	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	51.8	70.4	56.3	44.7	67.9	48.5	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	51.9	70.9	56.5	45.0	68.4	48. 7	145M	982G	11.6

Diffusion models: Image-text cross-attention

• Text prompt is encoded by a language model and injected into the denoising U-Net using *cross-attention*



Image source

Diffusion models: Image-text cross-attention

• Text prompt is encoded by a language model and injected into the denoising U-Net using *cross-attention*



Image source

Diffusion transformer



W. Peebles and S. Xie. Scalable Diffusion Models with Transformers. ICCV 2023

Diffusion transformer



Figure 2: ImageNet generation with Diffusion Transformers (DiTs). Bubble area indicates the flops of the diffusion model. *Left:* FID-50K (lower is better) of our DiT models at 400K training iterations. Performance steadily improves in FID as model flops increase. *Right:* Our best model, DiT-XL/2, is compute-efficient and outperforms all prior U-Net-based diffusion models, like ADM and LDM.

W. Peebles and S. Xie. Scalable Diffusion Models with Transformers. ICCV 2023

Diffusion transformer

Increasing transformer size



W. Peebles and S. Xie. Scalable Diffusion Models with Transformers. ICCV 2023

Outline

- Architectures
- Self-supervised learning

DINO: Self-distillation with no labels



Figure 2: Self-distillation with no labels. We illustrate DINO in the case of one single pair of views (x_1, x_2) for simplicity. The model passes two different random transformations of an input image to the student and teacher networks. Both networks have the same architecture but different parameters. The output of the teacher network is centered with a mean computed over the batch. Each networks outputs a K dimensional feature that is normalized with a temperature softmax over the feature dimension. Their similarity is then measured with a cross-entropy loss. We apply a stop-gradient (sg) operator on the teacher to propagate gradients only through the student. The teacher parameters are updated with an exponential moving average (ema) of the student parameters.

Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
gs, gt: student and teacher networks
 C: center (K)
 tps, tpt: student and teacher temperatures
 1, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views
    s1, s2 = qs(x1), qs(x2) # student output n-by-K
   t1, t2 = gt(x1), gt(x2) # teacher output n-by-K
   loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate
    # student, teacher and center updates
   update(qs) # SGD
    gt.params = 1*gt.params + (1-1)*gs.params
   C = m * C + (1-m) * cat([t1, t2]).mean(dim=0)
def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
   t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

M. Caron et al. Emerging Properties in Self-Supervised Vision Transformers. ICCV 2021

DINO: Self-distillation with no labels



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.

M. Caron et al. Emerging Properties in Self-Supervised Vision Transformers. ICCV 2021

Method	Arch.	Pa	iram.	im/s	Linear	k-NN
Supervised	RN50		23	1237	79.3	79.3
SCLR [11]	RN50		23	1237	69.1	60.7
MoCov2 [13]	RN50		23	1237	71.1	61.9
InfoMin [54]	RN50		23	1237	73.0	65.3
BarlowT [66]	RN50		23	1237	73.2	66.0
OBoW [21]	RN50		23	1237	73.8	61.9
BYOL [23]	RN50		23	1237	74.4	64.8
DCv2 [9]	RN50		23	1237	75.2	67.1
SwAV [9]	RN50		23	1237	75.3	65.7
DINO	RN50		23	1237	75.3	67.5
Supervised	ViT-S		21	1007	79.8	79.8
BYOL* [23]	ViT-S		21	1007	71.4	66.6
MoCov2* [13]	ViT-S		21	1007	72.7	64.4
SwAV* [9]	ViT-S		21	1007	73.5	66.3
DINO	ViT-S		21	1007	77.0	74.5

DINO: Self-distillation with no labels

Table 2: Linear and k-NN classification on ImageNet. We report top-1 accuracy for linear and k-NN evaluations on the validation set of ImageNet for different self-supervised methods. We focus on ResNet-50 and ViT-small architectures, but also report the best results obtained across architectures. * are run by us. We run the k-NN evaluation for models with official released weights. The throughput (im/s) is calculated on a NVIDIA V100 GPU with 128 samples per forward. Parameters (M) are of the feature extractor.

M. Caron et al. Emerging Properties in Self-Supervised Vision Transformers. ICCV 2021

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 to produce representations for recognition tasks.

K. He et al. Masked autoencoders are scalable vision learners. CVPR 2022

Masked autoencoders



Figure 2. Example results on ImageNet *validation* images. For each triplet, we show the masked image (left), our MAE reconstruction[†] (middle), and the ground-truth (right). The masking ratio is 80%, leaving only 39 out of 196 patches. More examples are in the appendix. [†]As no loss is computed on visible patches, the model output on visible patches is qualitatively worse. One can simply overlay the output with the visible patches to improve visual quality. We intentionally opt not to do this, so we can more comprehensively demonstrate the method's behavior.

K. He et al. Masked autoencoders are scalable vision learners. CVPR 2022

Masked autoencoders: Results



Figure 8. MAE pre-training vs. supervised pre-training, evaluated by fine-tuning in ImageNet-1K (224 size). We compare with the original ViT results [16] trained in IN1K or JFT300M.

method	pre-train data	ViT-B	ViT-L	ViT-H	ViT-H ₄₄₈
scratch, our impl.	-	82.3	82.6	83.1	-
DINO [5]	IN1K	82.8	-	-	-
MoCo v3 [9]	IN1K	83.2	84.1	2-	-
BEiT [2]	IN1K+DALLE	83.2	85.2	-	-
MAE	IN1K	83.6	85.9	86.9	87.8

		AP ^{box}		AP	nask
method	pre-train data	ViT-B	ViT-L	ViT-B	ViT-L
supervised	IN1K w/ labels	47.9	49.3	42.9	43.9
MoCo v3	IN1K	47.9	49.3	42.7	44.0
BEiT	IN1K+DALLE	49.8	53.3	44.4	47.1
MAE	IN1K	50.3	53.3	44.9	47.2

Table 4. **COCO object detection and segmentation** using a ViT Mask R-CNN baseline. All entries are based on our implementation. Self-supervised entries use IN1K data *without* labels. Mask AP follows a similar trend as box AP.

K. He et al. Masked autoencoders are scalable vision learners. CVPR 2022



A. Bar et al. Visual prompting via image inpainting. NeurIPS 2022



Figure 3: **Random images from our Computer Vision Figures dataset**. We curated a dataset of 88k unlabeled figures from Computer Vision academic papers. During training, we randomly sample crops from these figures, without any additional parsing.



Figure 4: Visual prompting prediction examples. Each visual prompt was fed to an MAE-VQGAN model trained on the Figures dataset. For each visual prompt, the result is marked in red.



Figure 10: **Style and content extrapolation using MAE-VQGAN**. The model can extrapolate the style of a new content (a), but fails to predict a new content (b). The model struggles to extrapolate new style and content of longer sequences (c-e).



Task ambiguity

Non-aligned input-output

Out of dist. decoding

Figure 11: Limitations and failure cases. Single input-output example might be ambiguous and can lead to unintended completions. The MAE-VQGAN model performs worse given non-aligned input-output example, and by using a VQGAN vocabulary, it is limited in synthesizing out-of-distribution pixels (like blurry images).

Outline

- Architectures
- Self-supervised learning
- Autoregressive text-to-image generation

DALL-E

Learn a joint sequential transformer model that can be used to generate image based on text prompt



accordion.

a tapir with the texture of an hedgehog in a christmas "backprop". a neon sign that sweater walking a dog

(a) a tapir made of accordion. (b) an illustration of a baby (c) a neon sign that reads reads "backprop". backprop neon sign

> A. Ramesh et al., Zero-Shot Text-to-Image Generation, arXiv 2021 https://openai.com/blog/dall-e/

DALL-E: Image encoding

• Train convolutional encoder and decoder to compress images to 32x32 grids of discrete tokens (each assuming 8192 values)



Figure 1. Comparison of original images (top) and reconstructions from the discrete VAE (bottom). The encoder downsamples the spatial resolution by a factor of 8. While details (e.g., the texture of the cat's fur, the writing on the storefront, and the thin lines in the illustration) are sometimes lost or distorted, the main features of the image are still typically recognizable. We use a large vocabulary size of 8192 to mitigate the loss of information.

DALL-E: Transformer architecture and training

- Concatenate up to 256 text tokens with 32x32=1024 image tokens, learn a transformer model with 64 layers and 12B parameters
- Dataset: 250M image-text pairs from the Internet
- Transformer model details
 - Decoder-only architecture
 - 64 self-attention layers,
 - 62 attention heads, sparse attention patterns
 - Mixed-precision training, distributed optimization



(a) Row attention mask.

(b) Column attention mask.

(c) Column attention mask with (d) Convolutional attention mask. transposed image states.

Figure 11. Illustration of the three types of attention masks for a hypothetical version of our transformer with a maximum text length of 6 tokens and image length of 16 tokens (i.e., corresponding to a 4×4 grid). Mask (a) corresponds to row attention in which each image token attends to the previous 5 image tokens in raster order. The extent is chosen to be 5, so that the last token being attended to is the one in the same column of the previous row. To obtain better GPU utilization, we transpose the row and column dimensions of the image states when applying column attention, so that we can use mask (c) instead of mask (b). Mask (d) corresponds to a causal convolutional attention pattern with wraparound behavior (similar to the row attention) and a 3×3 kernel. Our model uses a mask corresponding to an 11×11 kernel.

DALL-E: Generating images given text

• Output samples reranked using CLIP



Figure 6. Effect of increasing the number of images for the contrastive reranking procedure on MS-COCO captions.

PARTI: Pathways auto-regressive text-to-image



J. Yu et al. Scaling Autoregressive Models for Content-Rich Text-to-Image Generation. arXiv 2022

Model	Encoder Layers	Decoder Layers	Model Dims	MLP Dims	Heads	Total Params
Parti-350M	12	12	1024	4096	16	350M
Parti-750M	12	36	1024	4096	16	750M
Parti-3B	12	36	2048	8192	32	3B
Parti	16	64	4096	16384	64	20B



Parti-750M

Parti-3B

Parti-20B



A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!



A green sign that says "Very Deep Learning" and is at the edge of the Grand Canyon. Puffy white clouds are in the sky.



A. A photo of a frog reading the newspaper named "Toaday" written on it. There is a frog printed on the newspaper too.



B. A portrait of a statue of the Egyptian god Anubis wearing aviator goggles, white t-shirt and leather jacket. The city of Los Angeles is in the background. Hi-res DSLR photograph.



C. A high-contrast photo of a panda riding a horse. The panda is wearing a wizard hat and is reading a book. The horse is standing on a street against a gray concrete wall. Colorful flowers and the word "PEACE" are painted on the wall. Green grass grows from cracks in the street. DSLR photograph. daytime lighting.



D. A giant cobra snake made from $X. X \in \{$ "salad", "pancakes", "sushi", "corn" $\}$

E. A wombat sits in a yellow beach chair, while sipping a martini that is on his laptop keyboard. The wombat is wearing a white panama hat and a floral Hawaiian shirt. Out-of-focus palm trees in the background. DSLR photograph. Wide-angle view.

F. *The saying "BE EXCELLENT TO EACH OTHER"* ..., (a) brick wall and alien (b) driftwood. (c) old wooden boat with reflection. (d) stained glass. (See text for full prompts.)



a mountain road and looking over a green valley on a cloudy day. DSLR photograph. $X \in \{blue, red, yellow\}, Y \in \{1977, 1997,$ 2017}, $Z \in \{Porsche 911, Corvette, Ford F-150\}$

ing a cane. The raccoon is holding a garbage bag. Oil paint- the style of Egyptian hieroglyphics. $X \in \{$ "pandas", "toucans", ing in the style of X. $X \in \{$ "Rembrandt", "Vincent Van Gogh", "pangolins" $\}, Y \in \{$ "tennis", "soccer", "basketball" $\}$ "Hokusai", "pixel art", "pointillism", "abstract cubism", "Egyptian tomb heiroglyphics", "traditional Chinese painting", "Madhubani art"}

G. Three-quarters front view of a XYZ coming around a curve in H. A raccoon wearing formal clothes, wearing a tophat and hold- I. A photo of an Athenian vase with a painting of X playing Y in

Approach	Model Type	MS-COCO FID (\downarrow)		LN-COCO FID (\downarrow)	
		Zero-shot	Finetuned	Zero-shot	Finetuned
Random Train Images [10]	-	2.	47		-
Retrieval Baseline	-	17.97	6.82	33.59	16.48
TReCS [46]	GAN	-	-	-	48.70
XMC-GAN [47]	GAN	-	9.33	-	14.12
DALL-E [2]	Autoregressive	~ 28	-	-	-
CogView [3]	Autoregressive	27.1	-	-	-
CogView2 [61]	Autoregressive	24.0	17.7	-	-
GLIDE [11]	Diffusion	12.24	-	-	-
Make-A-Scene [10]	Autoregressive	11.84	7.55	-	_
DALL-E 2 [12]	Diffusion	10.39	-	-	-
Imagen [13]	Diffusion	7.27	-	-	-
Parti	Autoregressive	7.23	3.22	15.97	8.39

Table 5: Comparison with previous work on the MS-COCO (2014) [16] and Localized Narratives (COCO split) [29] validation sets. When available, we report results for both zero-shot and finetuned models. Retrieval models either perform retrieval over our training set ("zero-shot"), or the respective MS-COCO and LN-COCO training sets ("finetuned"). Parti samples 16 images per text prompt and uses a CoCa model to rank the outputs (Section 2.4). Similar to DALL-E 2 [12], we use guidance scale 1.2 for all above results. We report zero-shot FID score of other model sizes in Figure 9.

Outline

- Architectures
- Self-supervised learning
- Autoregressive text-to-image generation
- Trends

Trends

- Architectures
- Unification of tasks, datasets, modalities: "foundation models"
- Availability of increasingly powerful and reliable high-level primitives

Hybrid of CNNs and transformers?



T. Xiao et al. Early convolutions help transformers see better. NeurIPS 2021

Beyond transformers?



Figure 1: MLP-Mixer consists of per-patch linear embeddings, Mixer layers, and a classifier head. Mixer layers contain one token-mixing MLP and one channel-mixing MLP, each consisting of two fully-connected layers and a GELU nonlinearity. Other components include: skip-connections, dropout, and layer norm on the channels.

I. Tolstikhin et al. MLP-Mixer: An all-MLP Architecture for Vision. NeurIPS 2021

Or completely back to CNNs?



Figure 1. **ImageNet-1K classification** results for • ConvNets and • vision Transformers. Each bubble's area is proportional to FLOPs of a variant in a model family. ImageNet-1K/22K models here take $224^2/384^2$ images respectively. ResNet and ViT results were obtained with improved training procedures over the original papers. We demonstrate that a standard ConvNet model can achieve the same level of scalability as hierarchical vision Transformers while being much simpler in design.

Z. Liu et al. <u>A ConvNet for the 2020s</u>. CVPR 2022

Back to CNNs?



Figure 2. We modernize a standard ConvNet (ResNet) towards the design of a hierarchical vision Transformer (Swin), without introducing any attention-based modules. The foreground bars are model accuracies in the ResNet-50/Swin-T FLOP regime; results for the ResNet-200/Swin-B regime are shown with the gray bars. A hatched bar means the modification is not adopted. Detailed results for both regimes are in the appendix. Many Transformer architectural choices can be incorporated in a ConvNet, and they lead to increasingly better performance. In the end, our pure ConvNet model, named ConvNeXt, can outperform the Swin Transformer.

Z. Liu et al. A ConvNet for the 2020s. CVPR 2022





Z. Liu et al. <u>A ConvNet for the 2020s</u>. CVPR 2022

ConvNeXT v2



Figure 2. **Our FCMAE framework.** We introduce a fully convolutional masked autoencoder (FCMAE). It consists of a sparse convolution-based ConvNeXt encoder and a lightweight ConvNeXt block decoder. Overall, the architecture of our autoencoder is asymmetric. The encoder processes only the visible pixels, and the decoder reconstructs the image using the encoded pixels and mask tokens. The loss is calculated only on the masked region.

Backbone	Method	#param	PT epoch	FT acc.
ViT-B	BEiT	88M	800	83.2
ViT-B	MAE	88M	1600	83.6
Swin-B	SimMIM	88M	800	84.0
ConvNeXt V2-B	FCMAE	89M	800	84.6
ConvNeXt V2-B	FCMAE	89M	1600	<u>84.9</u>
ViT-L	BEiT	307M	800	85.2
ViT-L	MAE	307M	1600	<u>85.9</u>
Swin-L	SimMIM	197M	800	85.4
ConvNeXt V2-L	FCMAE	198M	800	85.6
ConvNeXt V2-L	FCMAE	198M	1600	85.8
ViT-H	MAE	632M	1600	86.9
Swin V2-H	SimMIM	658M	800	85.7
ConvNeXt V2-H	FCMAE	659M	800	85.8
ConvNeXt V2-H	FCMAE	659M	1600	86.3

Table 4. Comparisons with previous masked image modeling approaches. The pre-training data is the IN-1K training set. All self-supervised methods are benchmarked by the end-to-end fine-tuning performance with an image size of 224. We underline the highest accuracy for each model size and bold our best results.

S. Woo et al. ConvNeXt V2: Co-designing and Scaling ConvNets with Masked Autoencoders. CVPR 2023

Trends

- Architectures
- Unification of tasks, datasets, modalities: "foundation models"
- Availability of increasingly powerful and reliable high-level primitives

Segment Anything Model



Figure 1: We aim to build a foundation model for segmentation by introducing three interconnected components: a promptable segmentation *task*, a segmentation *model* (SAM) that powers data annotation and enables zero-shot transfer to a range of tasks via prompt engineering, and a *data* engine for collecting SA-1B, our dataset of over 1 billion masks.

> A. Kirillov et al. <u>Segment Anything</u>. arXiv 2023 <u>https://segment-anything.com/</u>

Visual Programming Compositional Visual Question Answering		Natural Language Image Editing
Prediction Rationale	IMAGE: Question: Are there both ties and glasses in the picture? Program: BOX0=Loc(image=IMAGE, object='ties') MASKERA-Count(box=BOXN) BOX1=Loc(image=IMAGE, object='glasses') MASKERA-Count(box=BOXN) BOX1=Loc(image=IMAGE, object='glasses') MASKERIA-Count(box=BOXN) RSSULT=NASKERIA MASKERIA-Count(box=BOXN) RSSULT=NASKERIA MASKERIA-Count(box=BOXN) 0 and (ANSMERI) > 0 else 'no'") RESULT=NASKERIA Prediction: no	TMAGE: Prediction: TMAGE1 Image: Image: The text of the text of
VISPROG Program Interpreter	Natural Language Visual Reasoning LEFT: RIGHT:	Program: Constraints
High-level Program VISPROG	Statement: The left and right image contains a total of six people and two boats. Program: ANSUER8-Vaq(image-LEFT, question-'How many people are in the image?') ANSUER8-Vaq(image-LEFT, question-'How many boats are in the im	Instruction: Replace desert with lush green grass Program: 0031>561(cinage=NMGE) 0031>561(cinage=NGGE, object=0830, query='desert', category=None)
Input Inage(s) Natural Language Instruction Instruction-program pairs	Traction Howebege Object ragging TMMGE: Time: Prediction: DMGG0 Discretion:	IMAGE=Replace(Image=IMAGE, object=0631, prompt='lush green grass') RESULT=IMAGE IMAGE: Prediction: IMAGE0 Image: Image=Image Prediction: Image0 Instruction: Create a color pop of Barack Obama (person) Program: 0810-Seg(image-IMAGE, object=0630, query='Barack Obama', category='person') IMAGE0 RISULT=IMAGE0

Figure 1. **VISPROG is a modular and interpretable neuro-symbolic system for compositional visual reasoning.** Given a few examples of natural language instructions and the desired high-level programs, VISPROG generates a program for any new instruction using *incontext learning* in GPT-3 and then executes the program on the input image(s) to obtain the prediction. VISPROG also summarizes the intermediate outputs into an interpretable *visual rationale* (Fig. 4). We demonstrate VISPROG on tasks that require composing a diverse set of modules for image understanding and manipulation, knowledge retrieval, and arithmetic and logical operations.

T. Gupta and A. Kembhavi. Visual Programming: Compositional visual reasoning without training. CVPR 2023

Modules supported in VisProg



• Image editing with natural language



Reasoning about image pairs



Example results ullet

Replace Leonardo DiCaprio with Leonardo DiCaprio wearing sunglasses



Tag the women leaders of Germany, Taiwan, and New Zealand



Tag the three female lead characters from Friends series

Create a color pop of the

woman in blue and the woman in



Tag these Scandinavian flags with their countries

Tag the painting of Girl with a Pearl Earring with its painter









Replace the couch with a







Replace the desert by sandy beach



red and blur the background



ViperGPT



S. Menon et al. ViperGPT: Visual Inference via Python Execution for Reasoning. arXiv 2023

ViperGPT

Query: How many muffins can each kid have for it to be fair? Execution kid patches = image_patch.find("kid") **Generated** Code muffin patches = image_patch.find("muffin") def execute_command(image): image_patch = ImagePatch(image) muffin patches = image patch.find("muffin") len(muffin patches)=8 kid_patches = image_patch.find("kid") len(kid_patches)=2 return str(len(muffin_patches) // len(kid_patches)) ▶8//2 = 4 Result:4 Query: Drink with zero alcohol drink_patches= bdrink_name = 'tullamore dew' >alcoholic = 'yes' def execute_command(image): Result: image_patch = ImagePatch(image) bdrink_name = 'bacardi' drink patches = image patch.find("drink") alcoholic = 'yes' for drink_patch in drink_patches: drink_name = drink_patch.simple_query("What is this?") alcoholic = llm_query(f"Does the {drink_name} have alcohol?") bdrink name = 'gin' if alcoholic == "no": >alcoholic = 'yes' return drink_patch return None >drink_name = 'dr pepper' >alcoholic = 'no' Query: What would the founder of the brand of the car on the left say to the founder of the brand of the car on the right? car_patches = def execute command(image): image_patch.find("car") image_patch = ImagePatch(image) car patches = image patch.find("car") left_car_brand='lamborghini' car_patches.sort(key=lambda car: car.horizontal_center) right_car_brand='ferrari' left_car = car_patches[0] >left_car_founder='Ferruccio Lamborghini' right car = car patches[-1] car_patches.sort(...) right_car_founder='Enzo Ferrari'

left_car_brand = left_car.simple_query("What is the brand of this car?")
right_car_brand = right_car.simple_query("What is the brand of this car?")
left_car_founder = llm_query(f"Who is the founder of {left_car_brand}?")
right_car_founder = llm_query(f"Who is the founder of {right_car_brand}?")
return llm_query(f"What would {left_car_founder} say to {right_car_founder}?")

Result: "Ferruccio Lamborghini might say, 'It's been an honor to be a rival of yours for so many years, Enzo. May our cars continue to push each other to be better and faster!' "