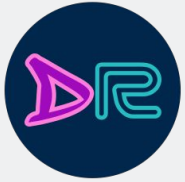
ROB 430/599.430: Deep Learning for Robot Perception and Manipulation (DeepRob)

Lecture 18: Vision Transformers

03/23/2026



https://deeprob.org/w26/

M | ROBOTICS

Today

- Feedback and Recap (5min)
- Attention (30min)
  - Example 1: Language translation
  - Example 2: video classification
- Transformers (30min)
- Vision Transformers (10min)
- Summary and Takeaways (5min)

Can we apply Transformers to Images?

Yes!

Idea: Treat the image as a set of patches of pixels

Vision Transformers

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy[*,†], **Lucas Beyer**[*], **Alexander Kolesnikov**[*], **Dirk Weissenborn**[*], **Xiaohua Zhai**[*], **Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby**[*,†]

[*]equal technical contribution, [†]equal advising

Google Research, Brain Team

{adosovitskiy, neilhoulsby}@google.com

Dosovitskiy et al. ICLR'21

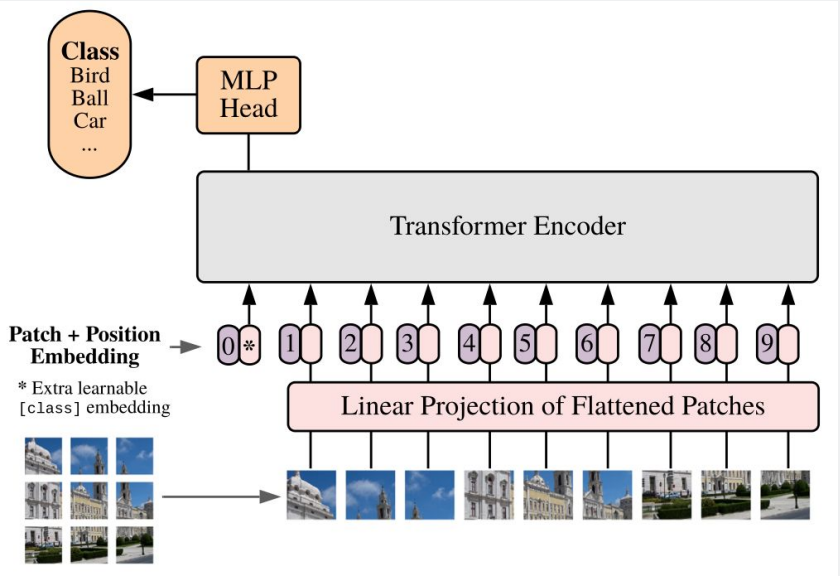
Cited by 89424

https://arxiv.org/pdf/2010.11929

M | ROBOTICS

Vision Transformers

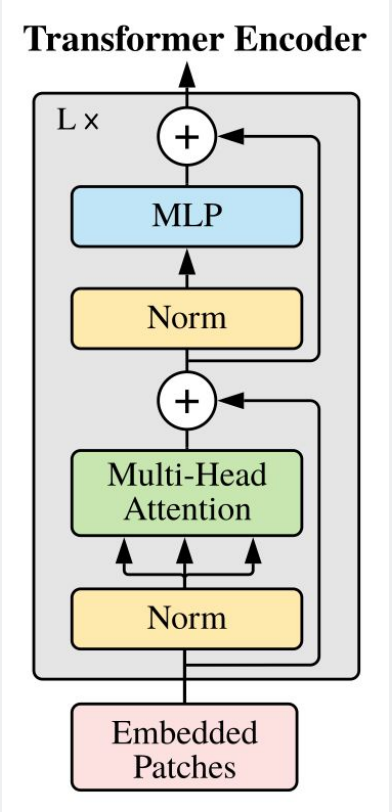
- Convert image into 16x16 patches
  - E.g. (1, 240, 240, 3) -> (1, 15x15, 16x16x3)
- Apply shared linear projection to each patch
  - E.g. (1, 15x15, 16x16x3) -> (1, 15x15, 64)
- Concatenate learnable class token for classifier output
  - E.g. (1, 1+15x15, 64)
- Add position embedding to each patch
  - E.g. (1, 1+15x15, 64) + (1, 1+15x15, 64)



(P4) patchify

Vision Transformer Encoder

- Based on the Transformer encoder
- Sequence of **LNorm->MHSA->LNorm->MLP** with residual skip connections
- For input embedded patches: (1, 1+15x15, D_in)
  - Output: (1, 1+15x15, D_out)
- For final classification decision:
  - Apply MLP and softmax to the class token
  - (1, 1, D_out) -> (1, 1, N_classes)



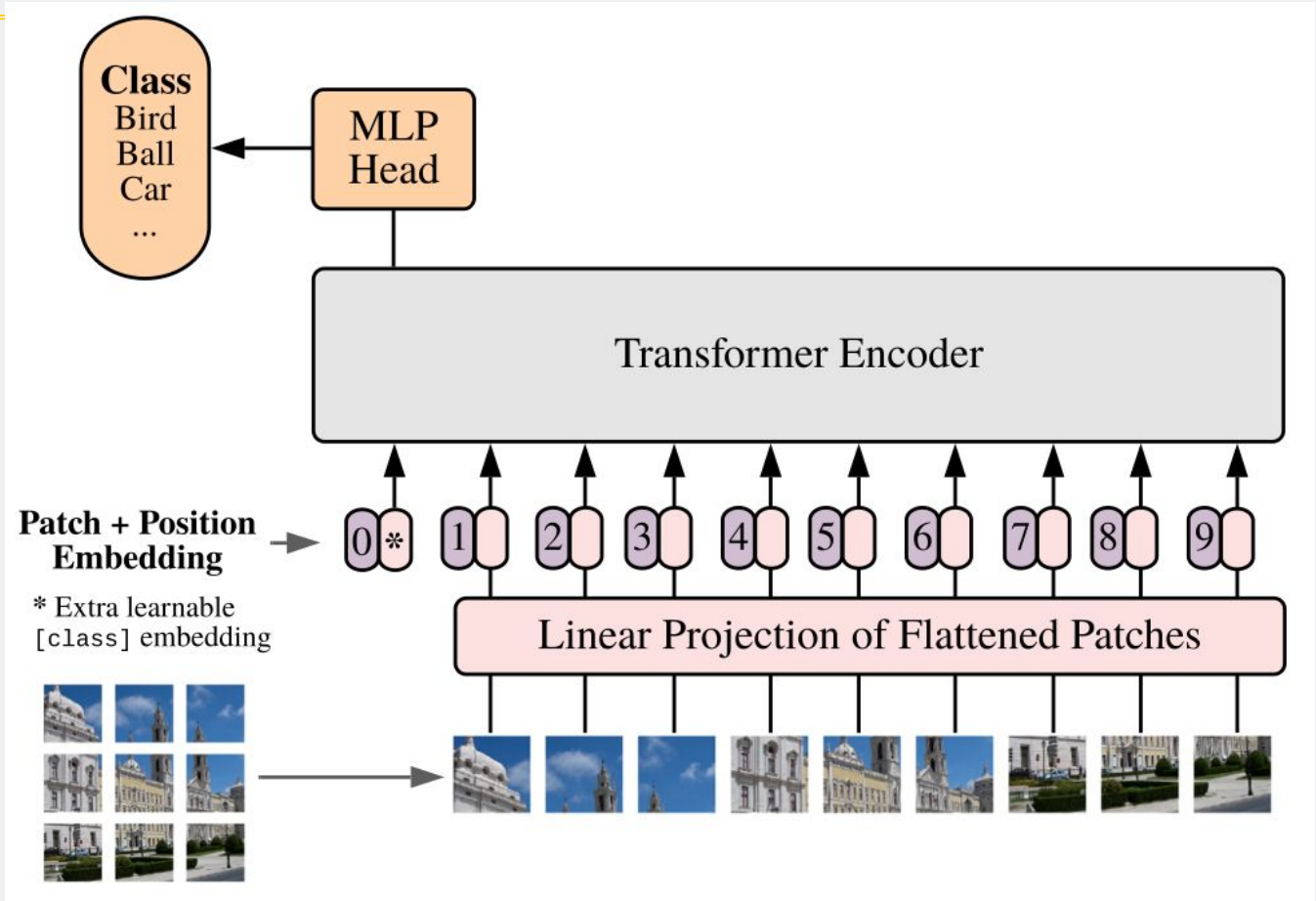
Dosovitskiy et al. ICLR'21

Vision Transformers - cls token

(also appears in **BERT**)
https://arxiv.org/pdf/1810.04805

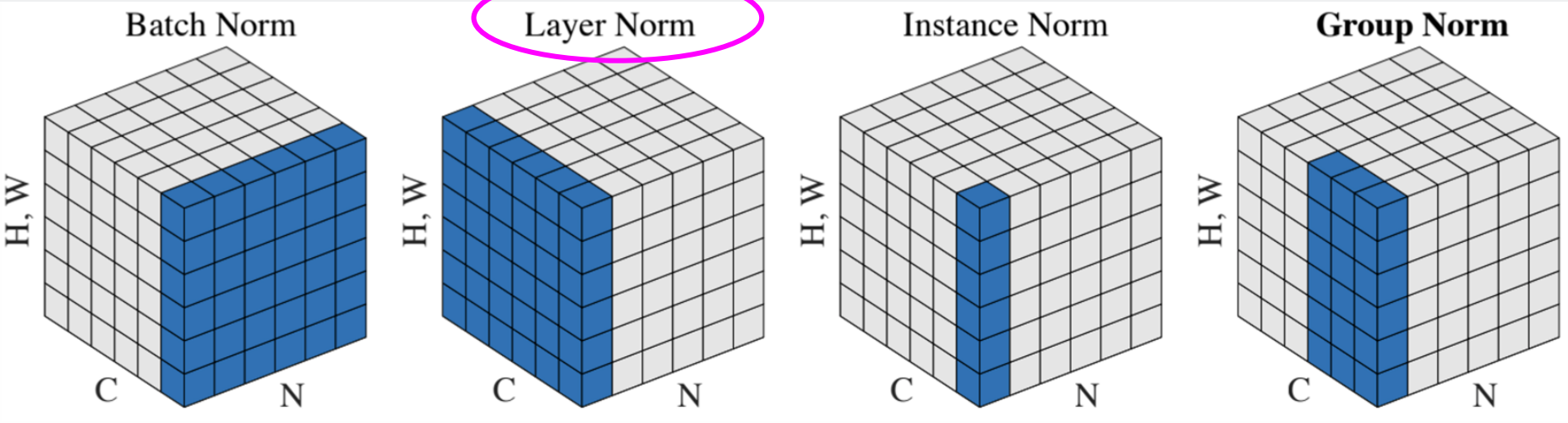
[cls] token

Added in front of every input example



Layer Normalization (LayerNorm)

Recall: From Lecture 7



https://arxiv.org/abs/1607.06450

(Also check the supplementary materials)

Layer Normalization (LayerNorm)

Why?

"One of the challenges of deep learning is that the gradients with respect to the weights in one layer are highly dependent on the outputs of the neurons in the previous layer especially if these outputs change in a highly correlated way."

"covariate shift"

BatchNorm: over the whole data distribution

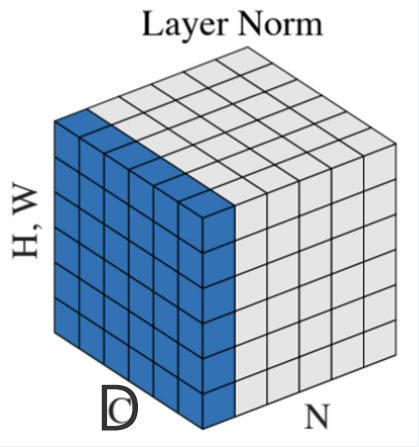
- Can be impractical
- Hard to apply to recurrent NNs

LayerNorm: Fixing the mean and variance within each layer

https://arxiv.org/abs/1607.06450

M | ROBOTICS

Layer Normalization (LayerNorm)



Forward:

Input: `x`: shape (*, D)

1. Compute mean and variance over the features (D) for each data point (N)
2. Normalizing the input data of shape (*, D)
3. Scale and shift using gamma and beta

```
out = gamma * div + beta
```

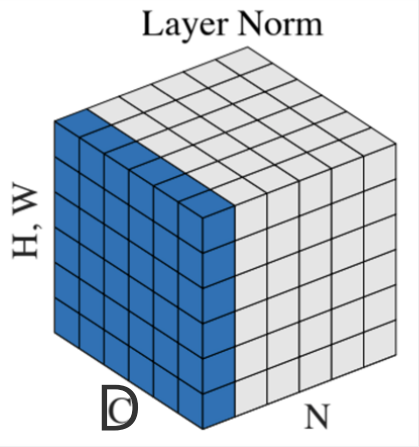
4. Store cache needed for backward pass

$$div = \frac{x - sampleMean}{\sqrt{sampleVar + eps}}$$

https://arxiv.org/abs/1607.06450

Layer Normalization (LayerNorm)

$$div = \frac{x - sampleMean}{\sqrt{sampleVar + eps}}$$



Backward:

Input: `dout`: Upstream derivatives, of shape (*, D)

```
out = gamma * div + beta
```

1. Compute gradients w.r.t. gamma and beta
2. Compute gradients w.r.t. input x
   a. ddiv = dout * gamma
   b. dx

```
dgamma, dbeta
```

```
return dx, dgamma, dbeta
```

https://robotchinwag.com/posts/layer-normalization-deriving-the-gradient-for-the-backward-pass/

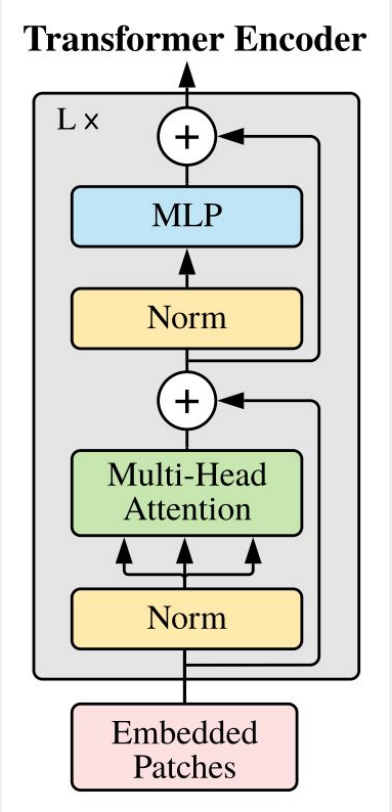
M | ROBOTICS

Vision Transformer Encoder

`Class TransformerEncoder (P4)`

- Based on the Transformer encoder
- Sequence of **LNorm->MHSA->LNorm->MLP** with residual skip connections
- For input embedded patches: (1, 1+15x15, D_in)
  - Output: (1, 1+15x15, D_out)
- For final classification decision:
  - Apply MLP and softmax to the class token
  - (1, 1, D_out) -> (1, 1, N_classes)

MLP: “two layers with a GELU non-linearity.”
Linear -> GELU -> Linear

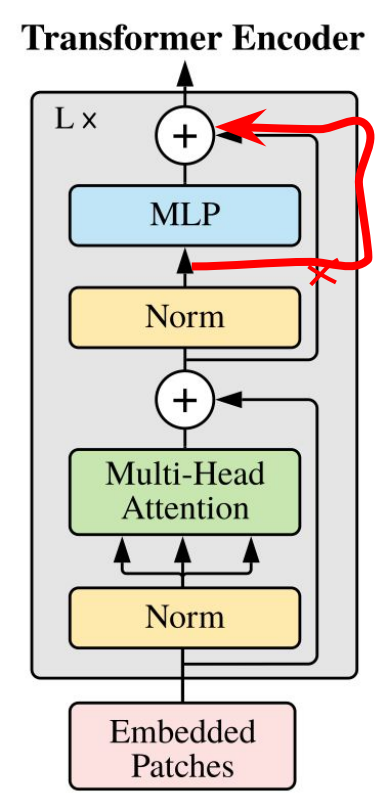


Dosovitskiy et al. ICLR’21

Vision Transformer Encoder

- Based on the Transformer encoder
- Sequence of **LNorm->MHSA->LNorm->MLP** with residual skip connections
- *One note on current autograder:
  - `x_norm2 + mlp_out`

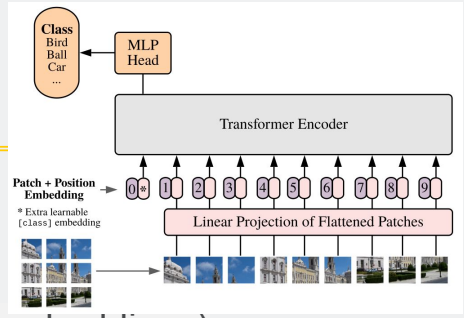
(in TransformerEncoder class)



Dosovitskiy et al. ICLR'21

Vision Transformer

Based on Fig.1 of ViT paper



`Class VisionTransformer (P4)`

`__INIT__` part: Initialize

1. a projection (linear) layer, that takes in patches and output `embed_dim` (embeddings)
2. transformer layers, where you have `range(self.num_layers)` of `TransformerEncoder` layers.
3. the MLP head, where it is essentially a `LayerNorm` + a `Linear` layer.

`forward` part: Pass through input x through all these layers. So:

1. patchify
2. call the projection layer
3. get cls_tokens (repeat `self.cls_token` for N times)
4. concatenate outputs from 2 and 3
5. add position embedding to 4. This will be the input that goes in stacks of transformer layers
6. loop through all `self.transformer_layers` , this gives you the `out_tokens` and `attention maps`. (see `TransformerEncoder` class for more details).
7. pass the `out_tokens` from 6 through your `mlp_head` (that you initialized in step 3 in the __init__). This will be `out_cls`.

M | ROBOTICS

Vision Transformer - Attention Maps Visualization

Input

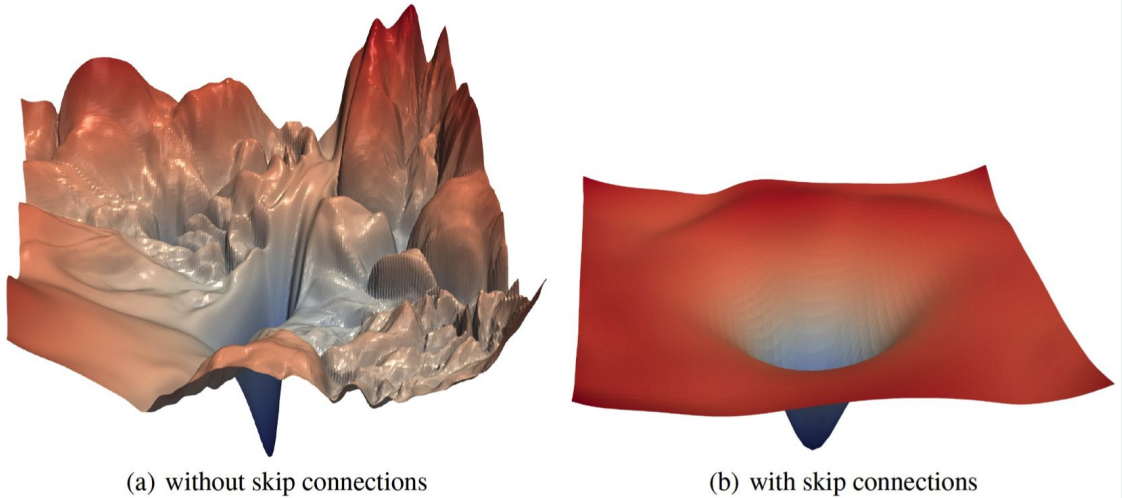
Attention

https://arxiv.org/pdf/2010.11929

Figure 6: Representative examples of attention from the output token to the input space. See Appendix D.7 for details.

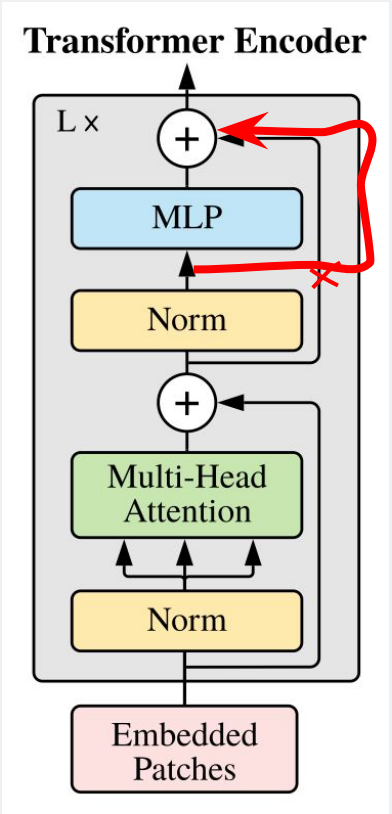
Vision Transformer Encoder

- Q: Why **skip connection**?



(a) without skip connections (b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections.



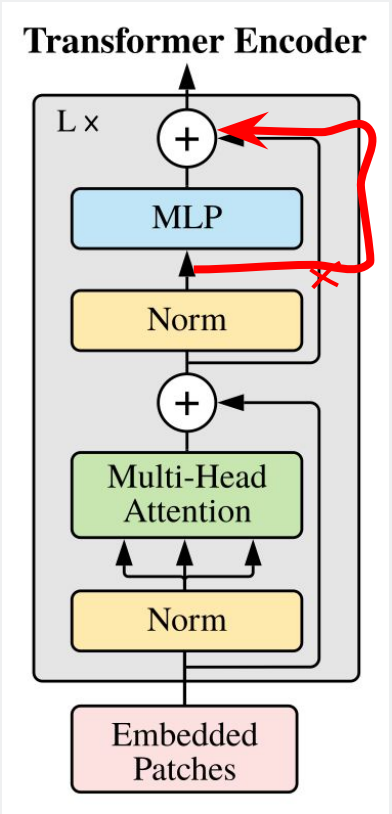
https://arxiv.org/pdf/1712.09913

Dosovitskiy et al. ICLR'21

Vision Transformer Encoder

- Q: What is **inductive bias**?

The inductive bias (also known as learning bias) of a learning algorithm is the set of assumptions that the learner uses to predict outputs of given inputs that it has not encountered



- Pre-trained on larger scale (14M-300M images)
- (authors claim) Vision Transformer has much less image-specific inductive bias than CNNs

https://arxiv.org/pdf/2010.11929

Improving Vision Transformer

- **Regularization** for ViT models:
  - Weight Decay
  - Stochastic Depth
  - Dropout (in FFN layers of transformer)
- **Data Augmentation** for ViT models:
  - MixUp
  - RandAugment

https://arxiv.org/pdf/2106.10270

- Revisit **High-frequency Components**

https://arxiv.org/pdf/2204.00993

- ViT + **Evolutionary Algorithm**

https://link.springer.com/content/pdf/10.1007/s11263-024-02034-6.pdf

Large-scale pre-training

- **Distillation** for ViT models:
  - Teacher-Student model
  - (will discuss more on this)

https://arxiv.org/pdf/1503.02531

- **ConViT: Soft** convolutional inductive bias (gated positional self-attention)

https://proceedings.mlr.press/v139/d-ascoli21a/d-ascoli21a.pdf

DETR

Cited by 24266

DETR: DEtection TRansformer

https://arxiv.org/pdf/2005.12872 (Facebook AI, 2020)

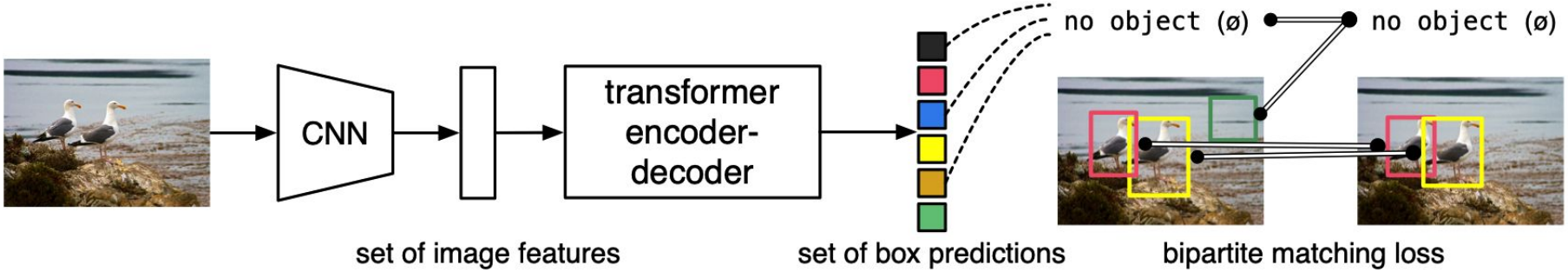


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "`no object`" ($\varnothing$) class prediction.

DETR: DEtection TRansformer

https://arxiv.org/pdf/2005.12872 (Facebook AI, 2020)

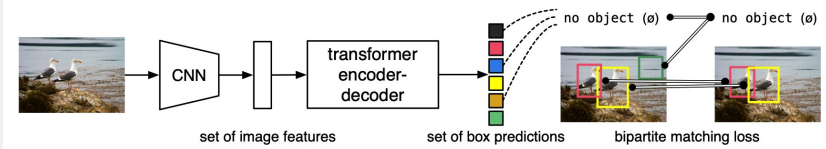


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "`no object`" ($\varnothing$) class prediction.

Motivation:

- End-to-end
- No NMS (post-processing)
- Direct "set prediction" problem - removing duplicate

Key architecture components:

- Transformer Encoder-Decoder
- Bipartite loss - Hungarian Algorithm

*Fixed-size N (large) predictions

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{i}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$$

M | ROBOTICS

DETR: DEtection TRansformer

https://arxiv.org/pdf/2005.12872 (Facebook AI, 2020)

https://github.com/facebookresearch/detr (Archived 2024)

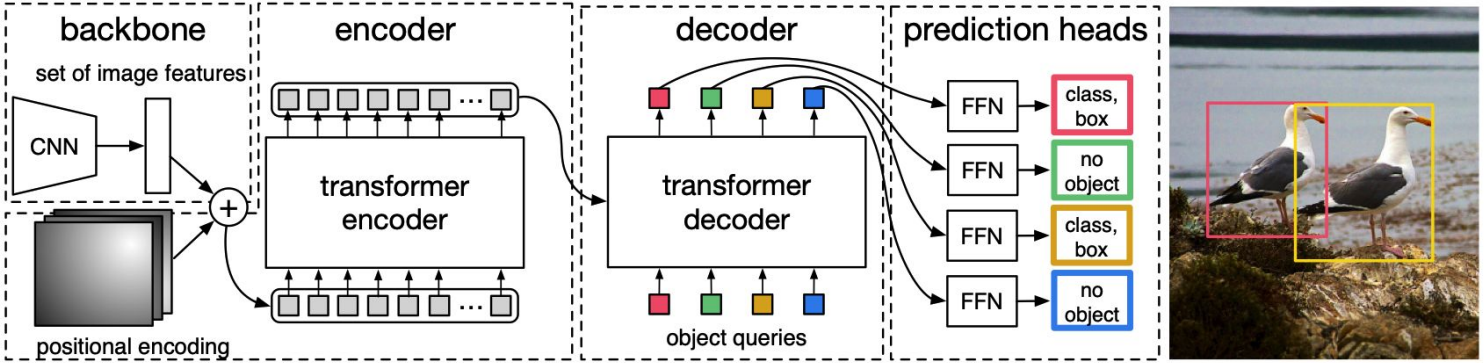
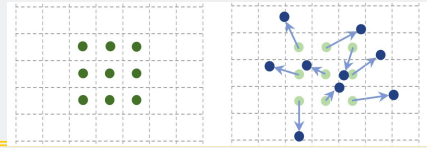


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings,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

M | ROBOTICS

Deformable-DETR



https://openaccess.thecvf.com/content_ICCV_2017/papers/Dai_Deformable_Convolutional_Networks_ICCV_2017_paper.pdf

https://arxiv.org/pdf/2010.04159
(ICLR 2021)

Address two DETR issues:
(1) It requires much longer training epochs to converge than the existing object detectors

(2) DETR delivers relatively low performance at detecting small objects

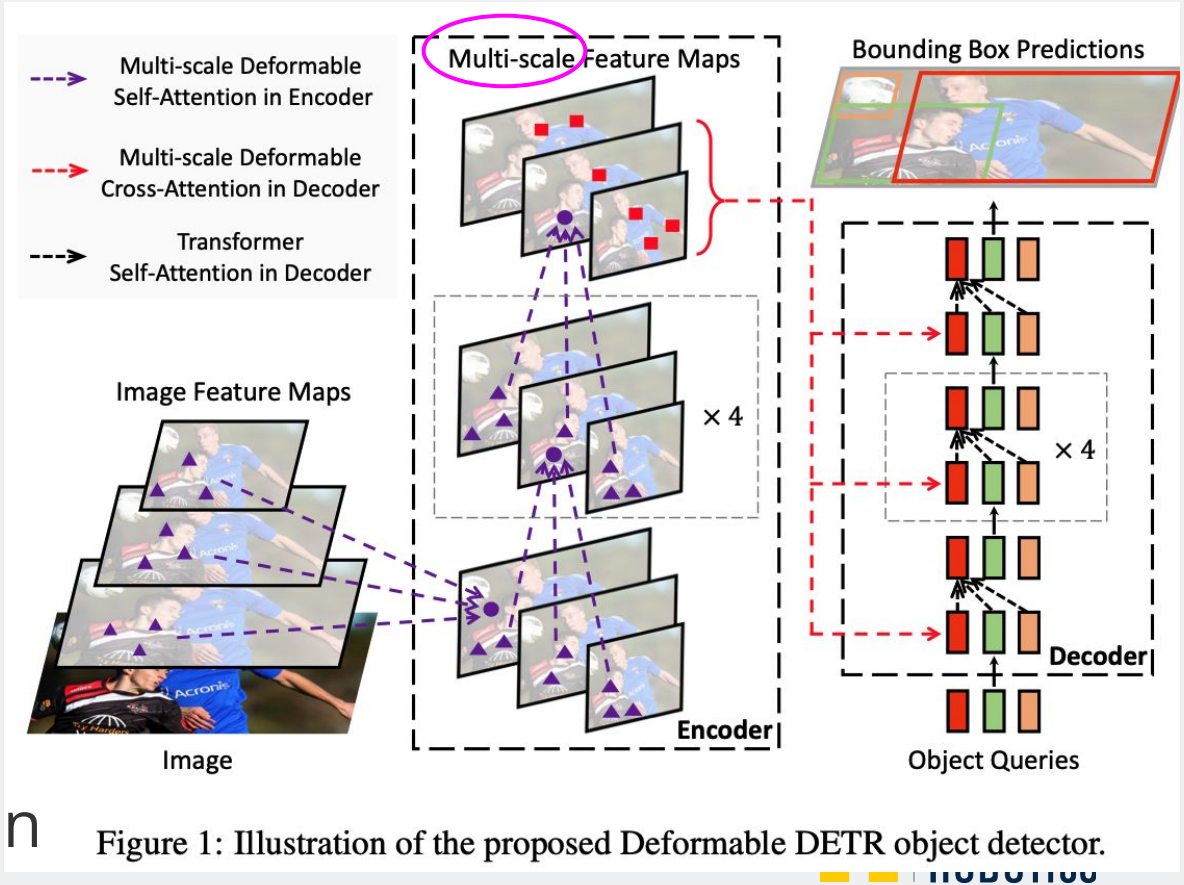
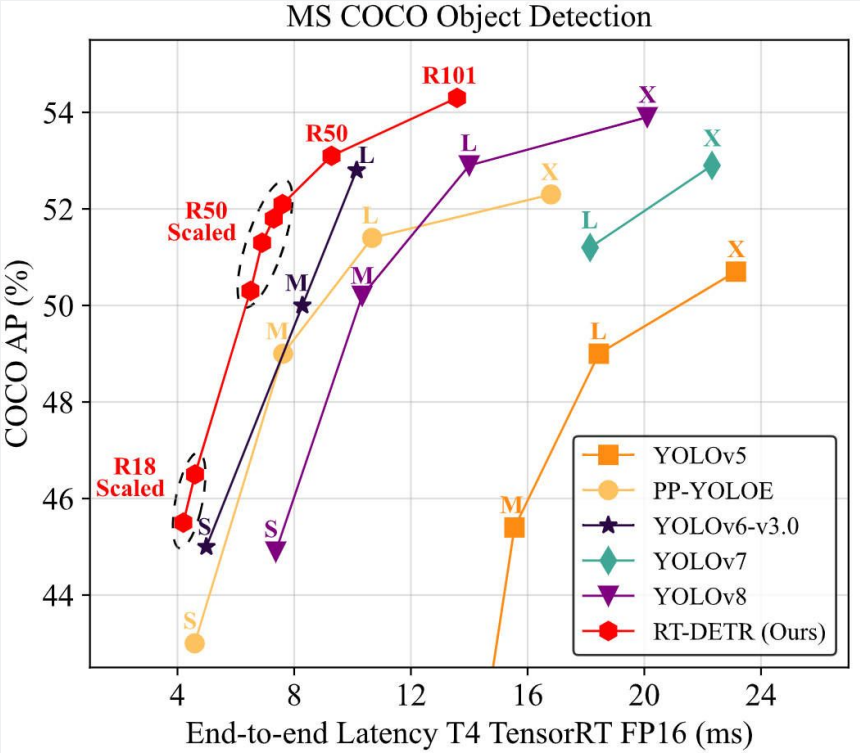


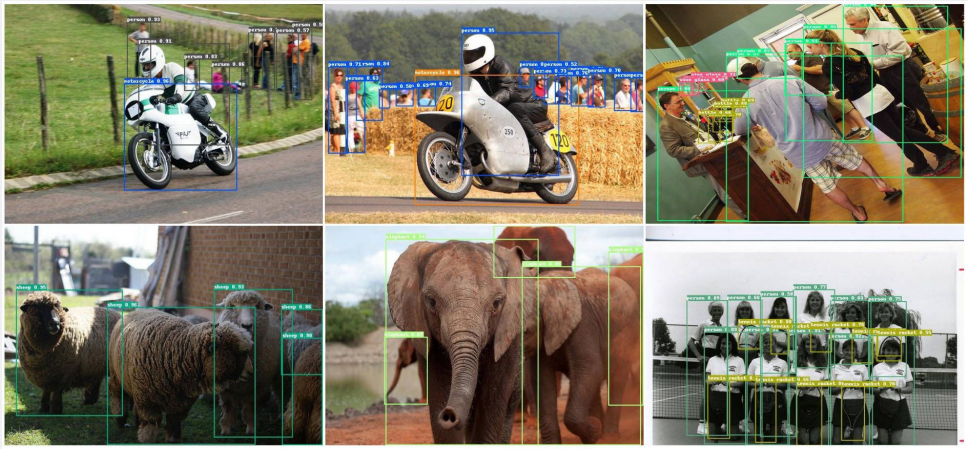
Figure 1: Illustration of the proposed Deformable DETR object detector.

➡ Deformable attention

RT-DETR (real-time)

https://zhao-yian.github.io/RTDETR/ (CVPR 2024)





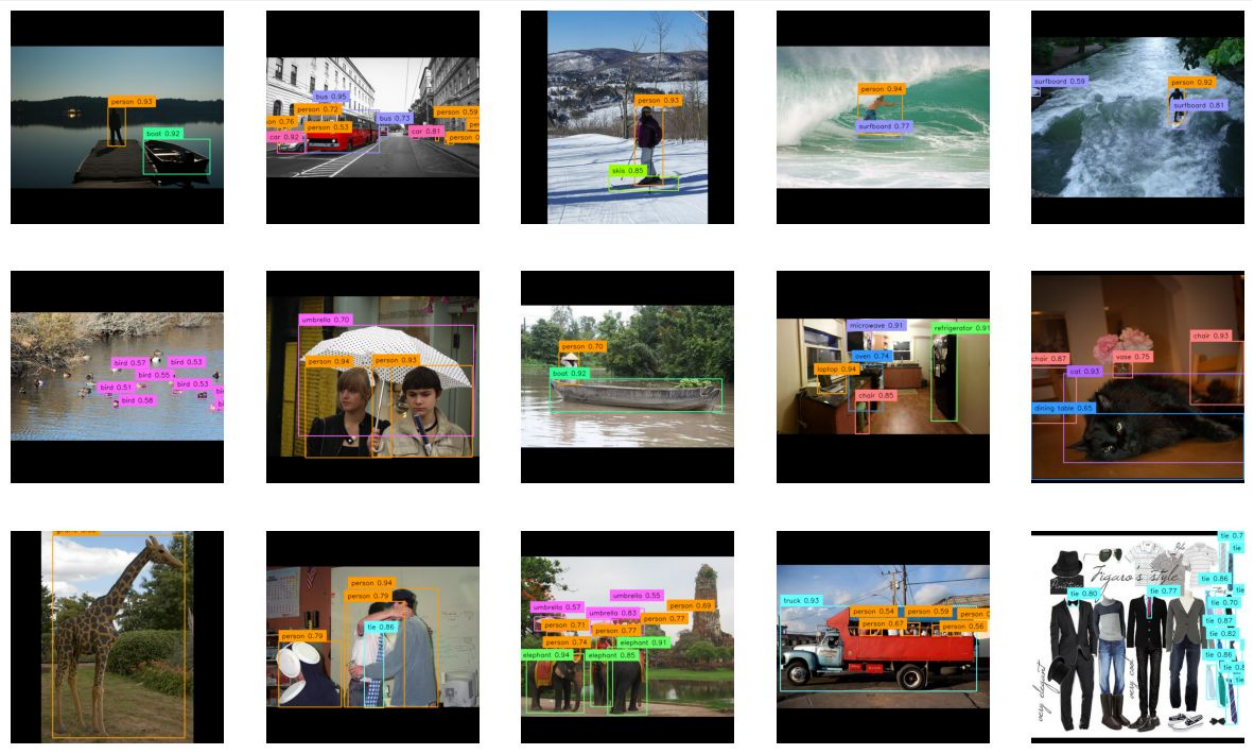
https://openaccess.thecvf.com/content/CVPR2024/papers/Zhao_DETRs_Beat_YOLOs_on_Real-time_Object_Detection_CVPR_2024_paper.pdf

M | ROBOTICS

RF-DETR (real-time)

https://github.com/roboflow/rf-detr (released March 20, 2025)

(claims >60AP)



ROBOTICS

“Any detector can detect anything”

https://openaccess.thecvf.com/content/WACV2026/html/Huang_Any_Detector_Can_Detect_Anything_WACV_2026_paper.html (WACV 2026)

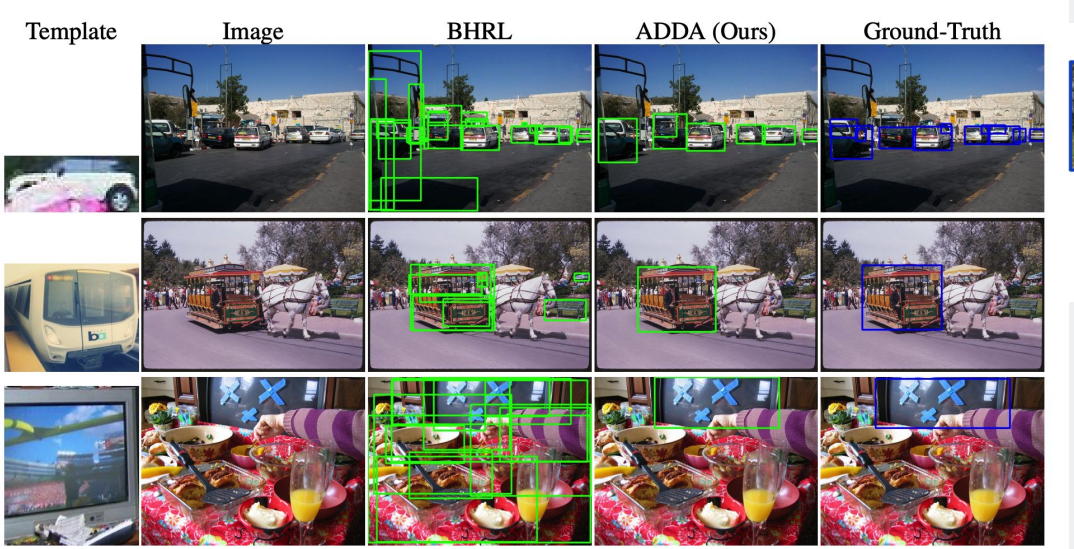
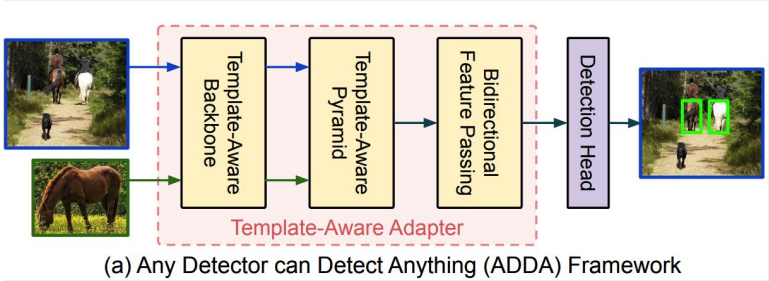


Figure 4. Qualitative comparison between BHRL [45] and ADDA on novel classes in COCO [25]. ADDA produces more accurate detections and fewer false positives.



(a) Any Detector can Detect Anything (ADDA) Framework

Segment Anything
https://aidemos.meta.com/segment-anything/

M | ROBOTICS

Detectron2

https://github.com/facebookresearch/detectron2

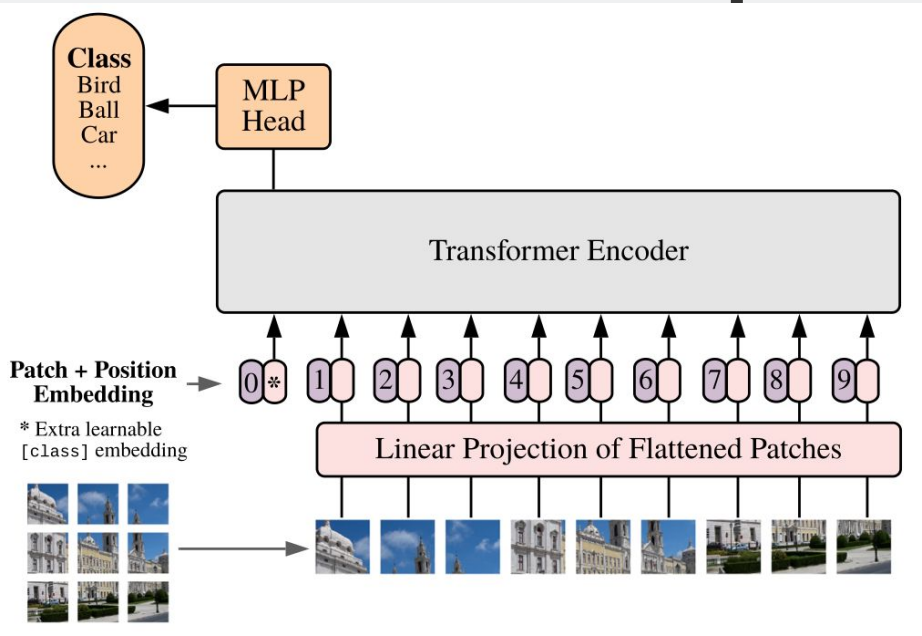
Reminder

- *P4 Due March 29, 2026*
- Canvas quiz 20260323 Transformers, due March 27, 2026

- Poster or Slides (**March 31 “lightning talk”**, 6min per group, 5% grade)
  Consider:
  - Topic/Title
  - Aim to reproduce/run 1-2 baseline method by then. Prelim results.
  - MUST-DO Goals + Stretch Goals
  - Some Lit Review (“Gaps”)
  - Group member roles/sub-tasks
  - Timeline/Milestone
  - Material (if any)
  - *Bonus: for showcasing on a real robot!*

M | ROBOTICS

Recap: ViT and CNN



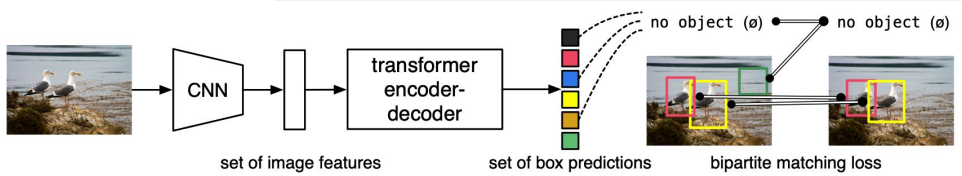


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "`no object`" ($\varnothing$) class prediction.

ROBOTICS

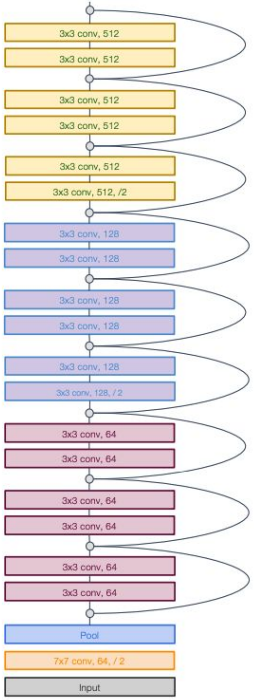
ViT vs. CNN

Stage 3:
256 x 14 x 14

Stage 2:
128 x 28 x 28

Stage 1:
64 x 56 x 56

Input:
3 x 224 x 224

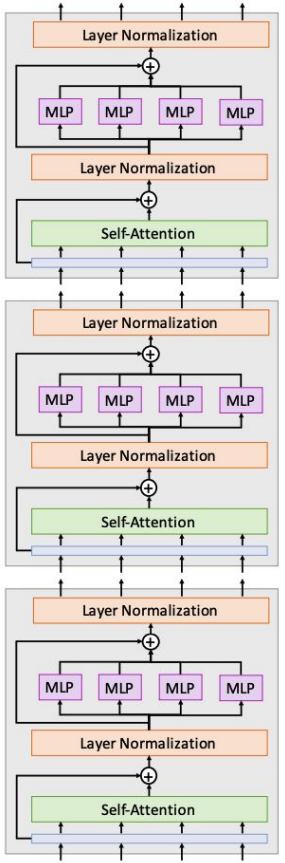


In most CNNs (including ResNets), **decrease** resolution and **increase** channels as you go deeper in the network (Hierarchical architecture)

Useful since objects in images can occur at various scales

➡ In a ViT, all blocks have same resolution and number of channels (Isotropic architecture)

Can we build a **hierarchical** ViT model?



3rd block:
768 x 14 x 14

2nd block:
768 x 14 x 14

1st block:
768 x 14 x 14

Input:
3 x 224 x 224

Swin Transformer

Cited by 28534

Swin Transformer

https://arxiv.org/pdf/2103.14030 (CVPR 2021)
https://github.com/microsoft/Swin-Transformer

hierarchical feature maps

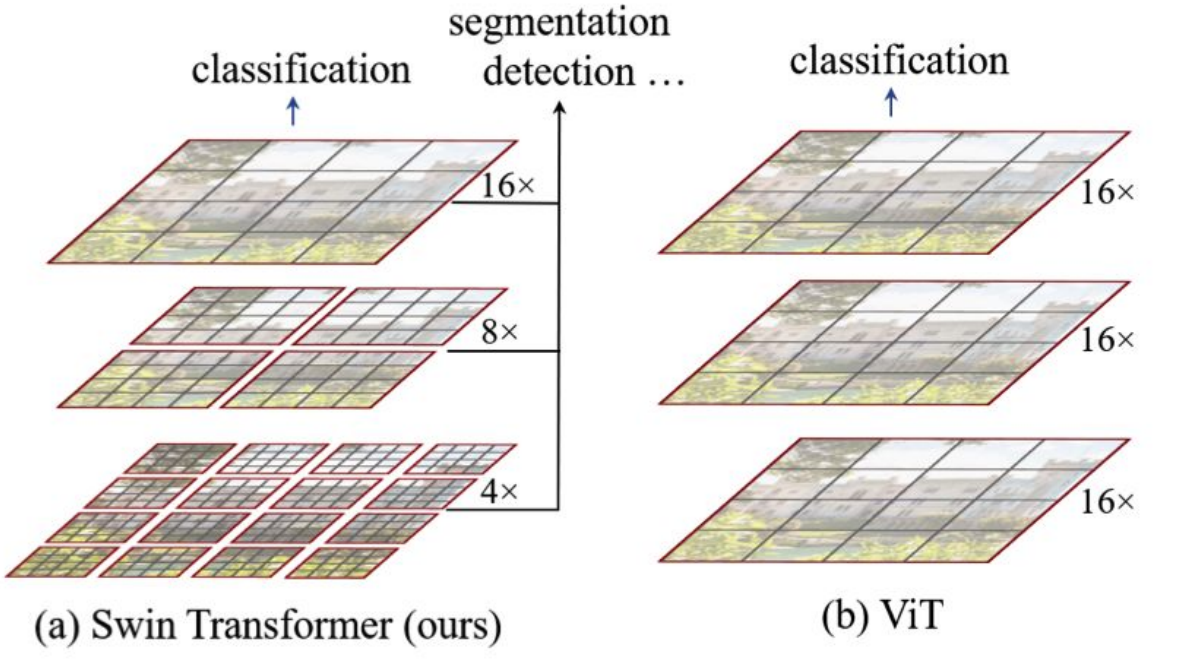
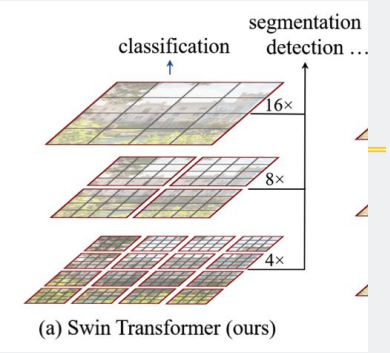


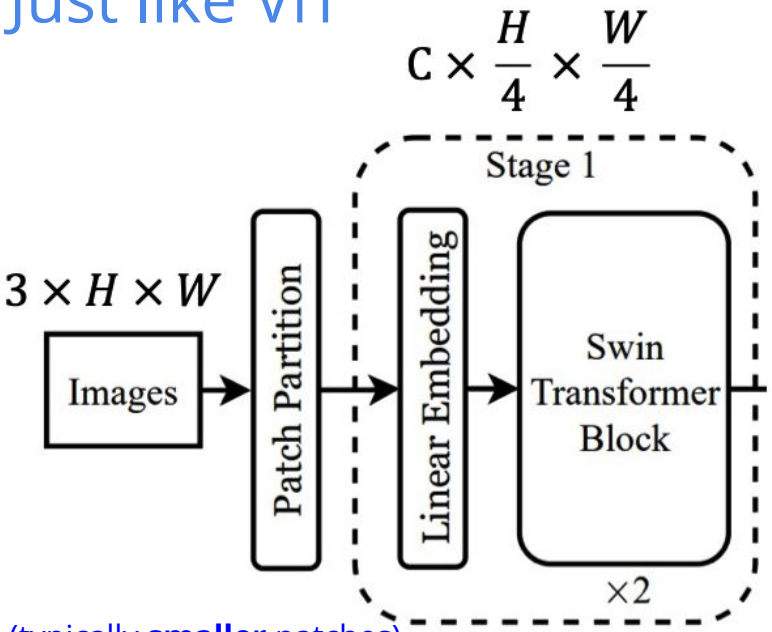
Figure 1. (a) The proposed Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window (shown in red). It can thus serve as a general-purpose backbone for both image classification and dense recognition tasks. (b) In contrast, previous vision Transformers [20] produce feature maps of a single low resolution and have quadratic computation complexity to input image size due to computation of self-attention globally.

M | ROBOTICS

Swin Transformer: Hierarchical ViT



Just like ViT

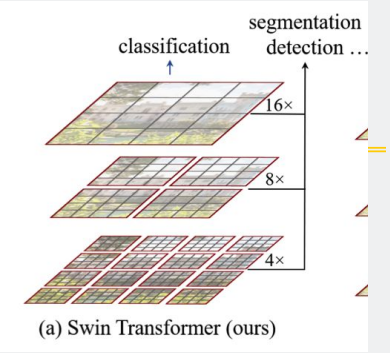


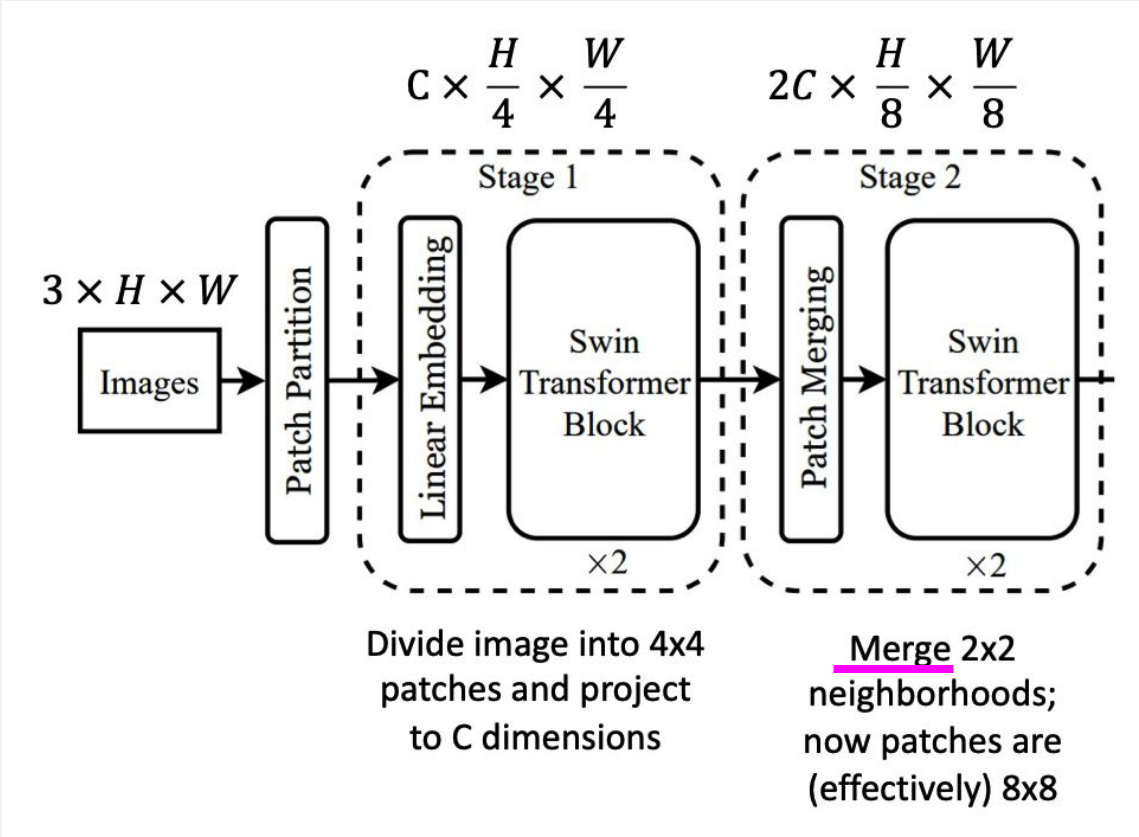
(typically **smaller** patches)

Divide image into 4x4 patches and project to C dimensions

M | ROBOTICS

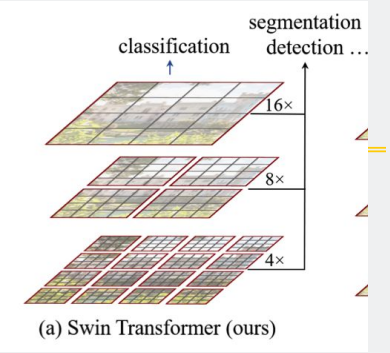
Swin Transformer: Hierarchical ViT

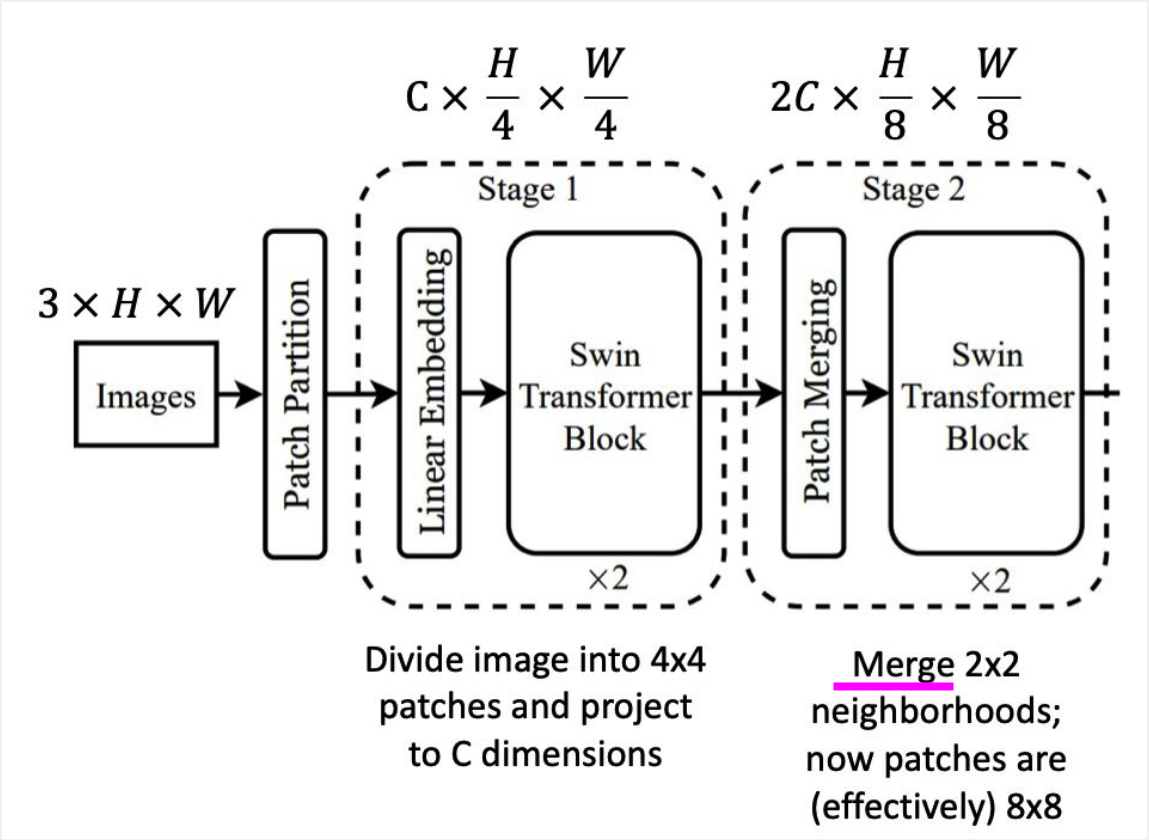


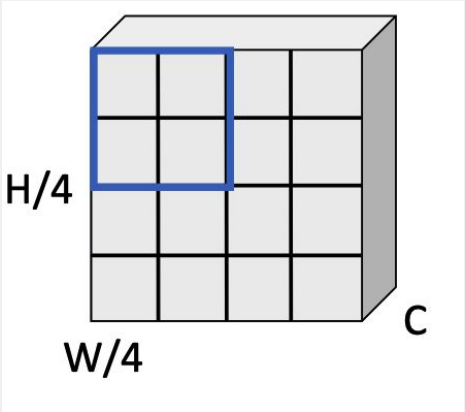


M | ROBOTICS

Swin Transformer: Hierarchical ViT

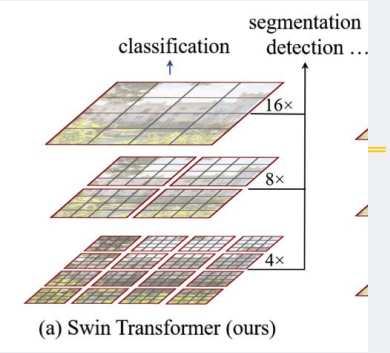


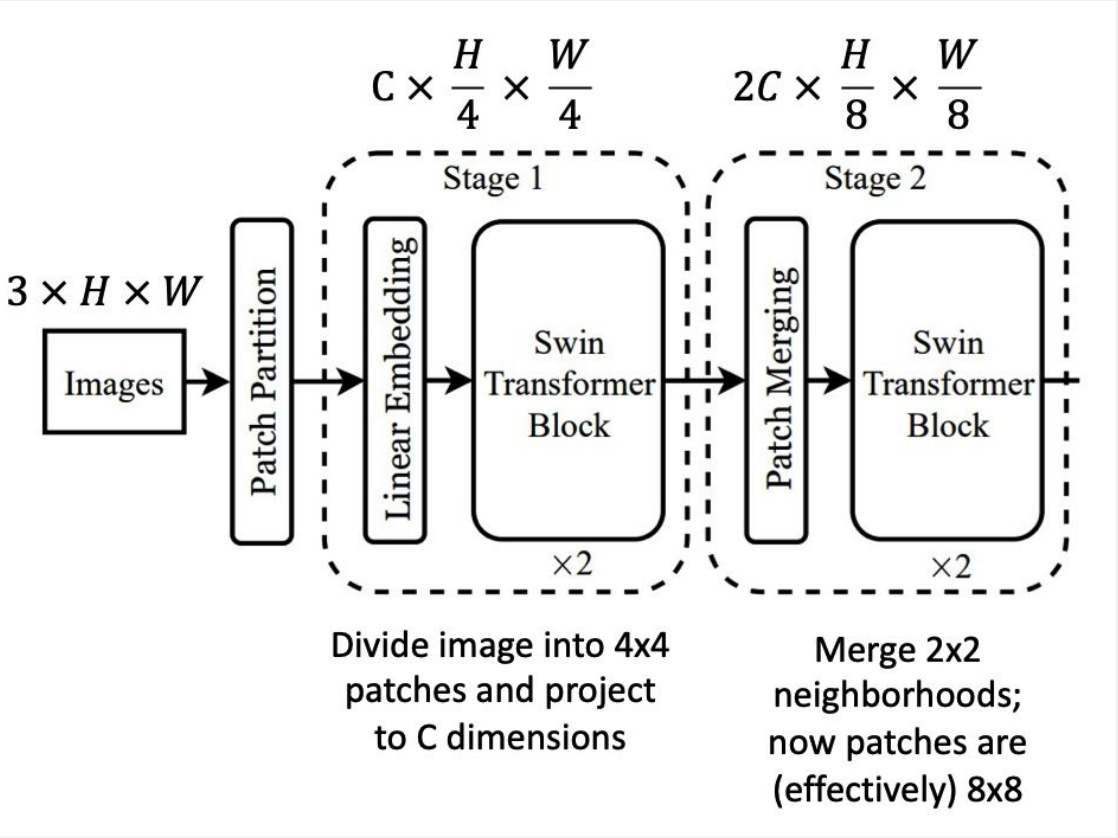


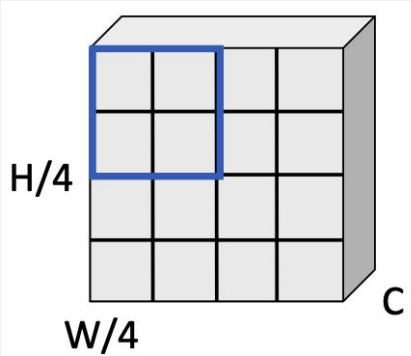


M | ROBOTICS

Swin Transformer: Hierarchical ViT

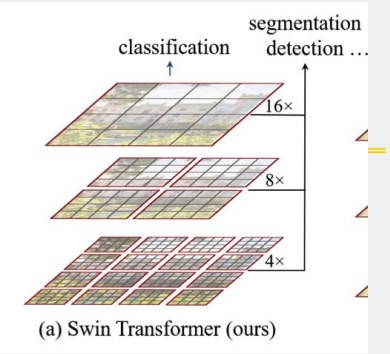


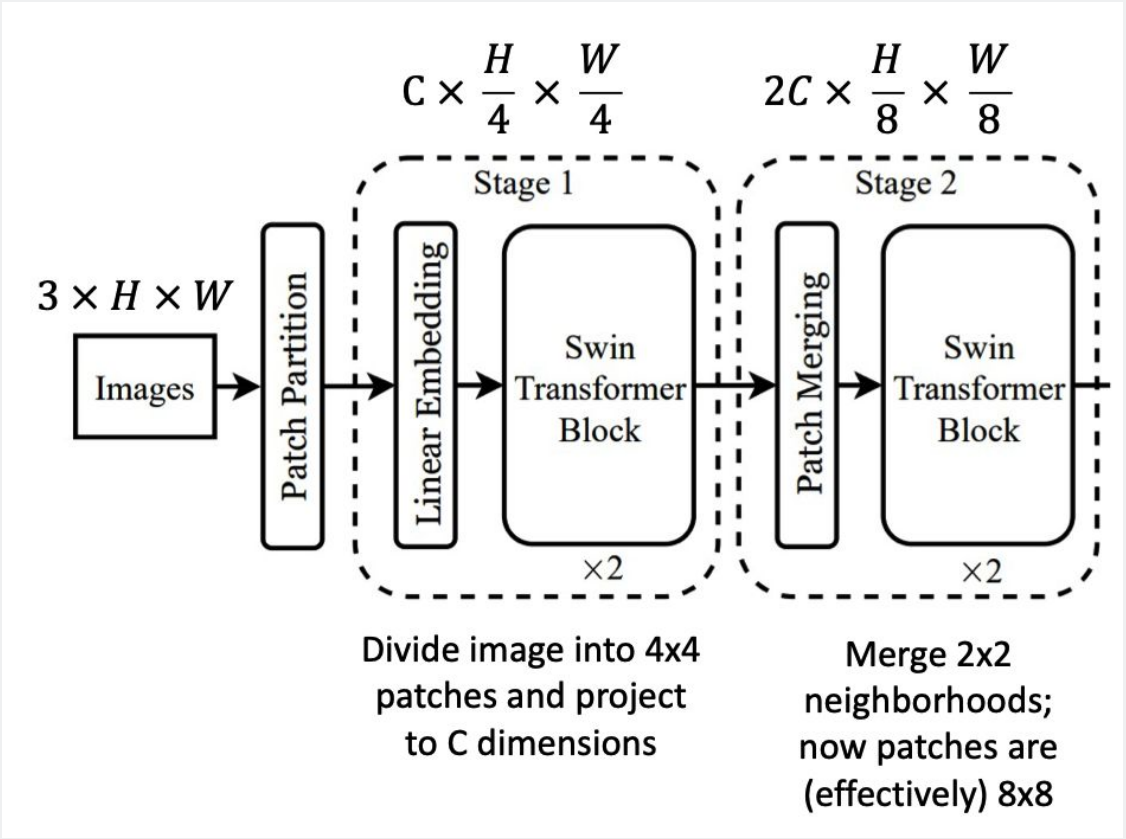


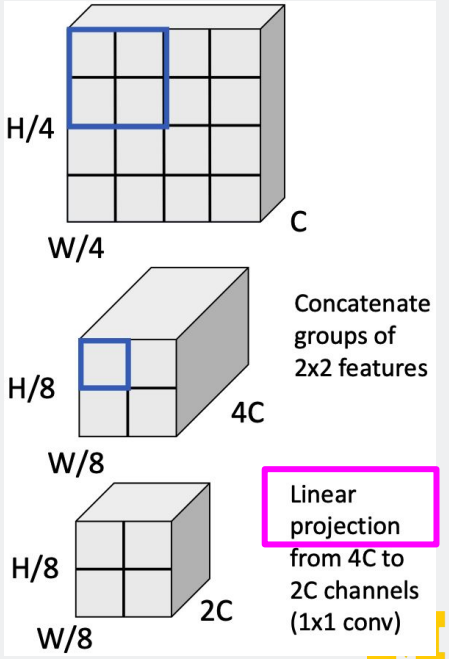


Lower spatial dimension, **higher** channel dimension

Swin Transformer: Hierarchical ViT

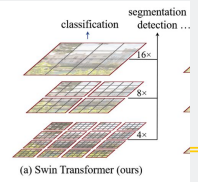


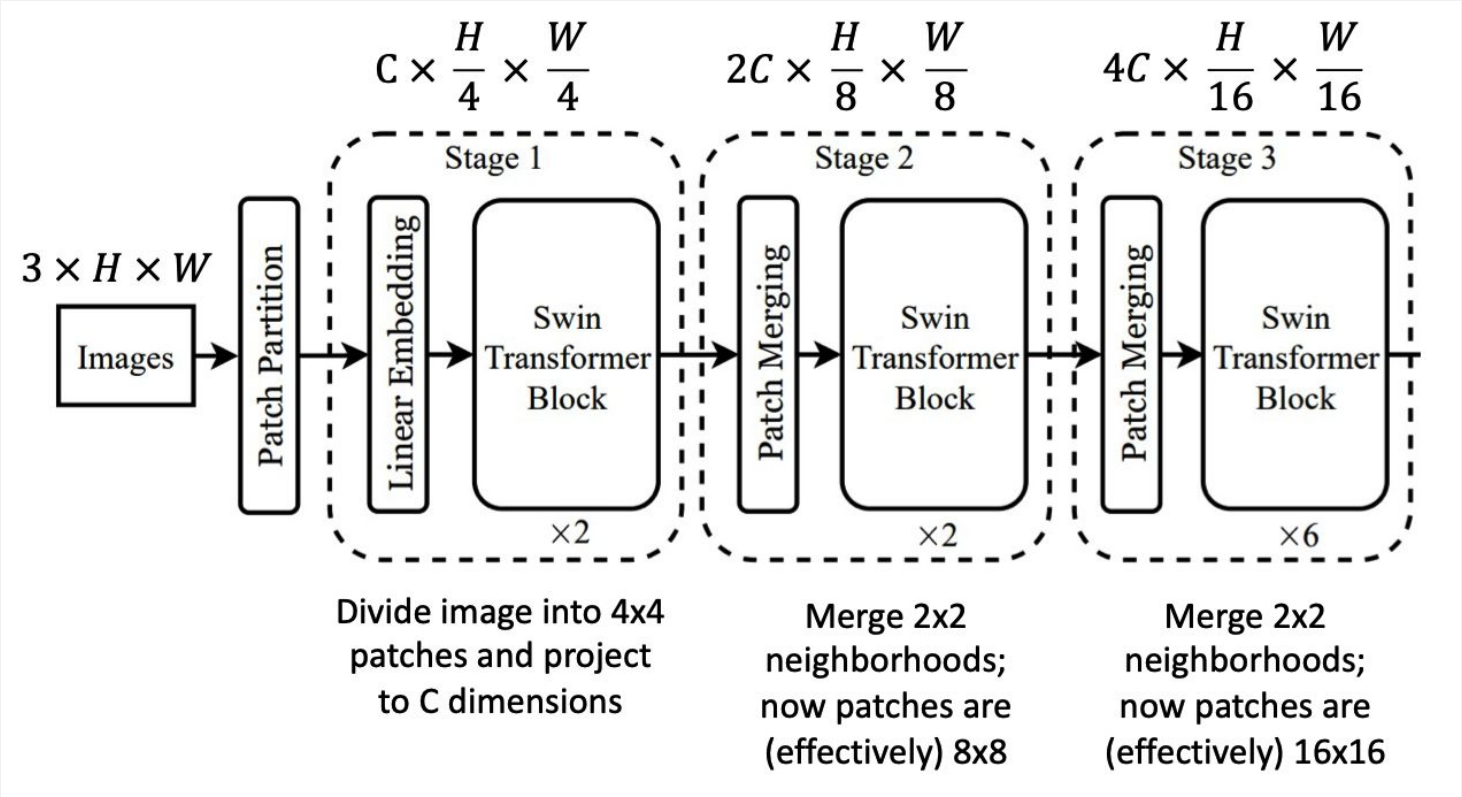




ROBOTICS

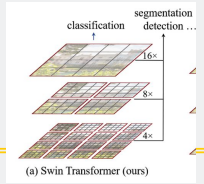
Swin Transformer: Hierarchical ViT





M | ROBOTICS

Swin Transformer: Hierarchical ViT



Goal: Find high-resolution, fine-grain features

Stacks

$3 \times H \times W$

Images

Patch Partition

$C \times \frac{H}{4} \times \frac{W}{4}$

Stage 1

Linear Embedding

Swin Transformer Block

×2

Divide image into 4x4 patches and project to C dimensions

$2C \times \frac{H}{8} \times \frac{W}{8}$

Stage 2

Patch Merging

Swin Transformer Block

×2

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

$4C \times \frac{H}{16} \times \frac{W}{16}$

Stage 3

Patch Merging

Swin Transformer Block

×6

Merge 2x2 neighborhoods; now patches are (effectively) 16x16

$8C \times \frac{H}{32} \times \frac{W}{32}$

Stage 4

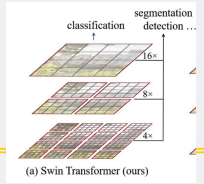
Patch Merging

Swin Transformer Block

×2

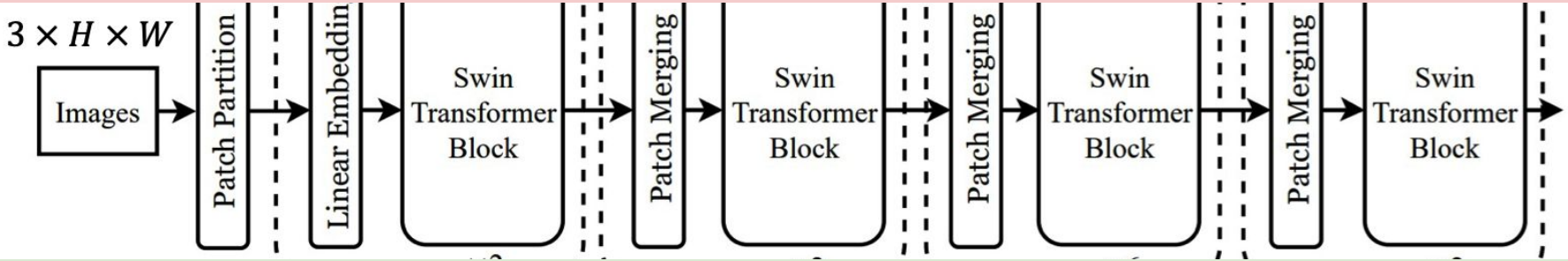
Merge 2x2 neighborhoods; now patches are (effectively) 32x32

Swin Transformer: Hierarchical ViT



Problem: 224x224 image with 56x56 grid of 4x4 patches

➡ attention matrix has 56^4 = 9.8M entries



Solution: don't use full attention,

➡ instead, use attention over **patches**

(effectively) 8x8 (effectively) 16x16 (effectively) 32x32

Swin Transformer: Window Attention

Usually, small M

With H x W grid of tokens, each attention matrix is
– **quadratic** in image size H^2W^2

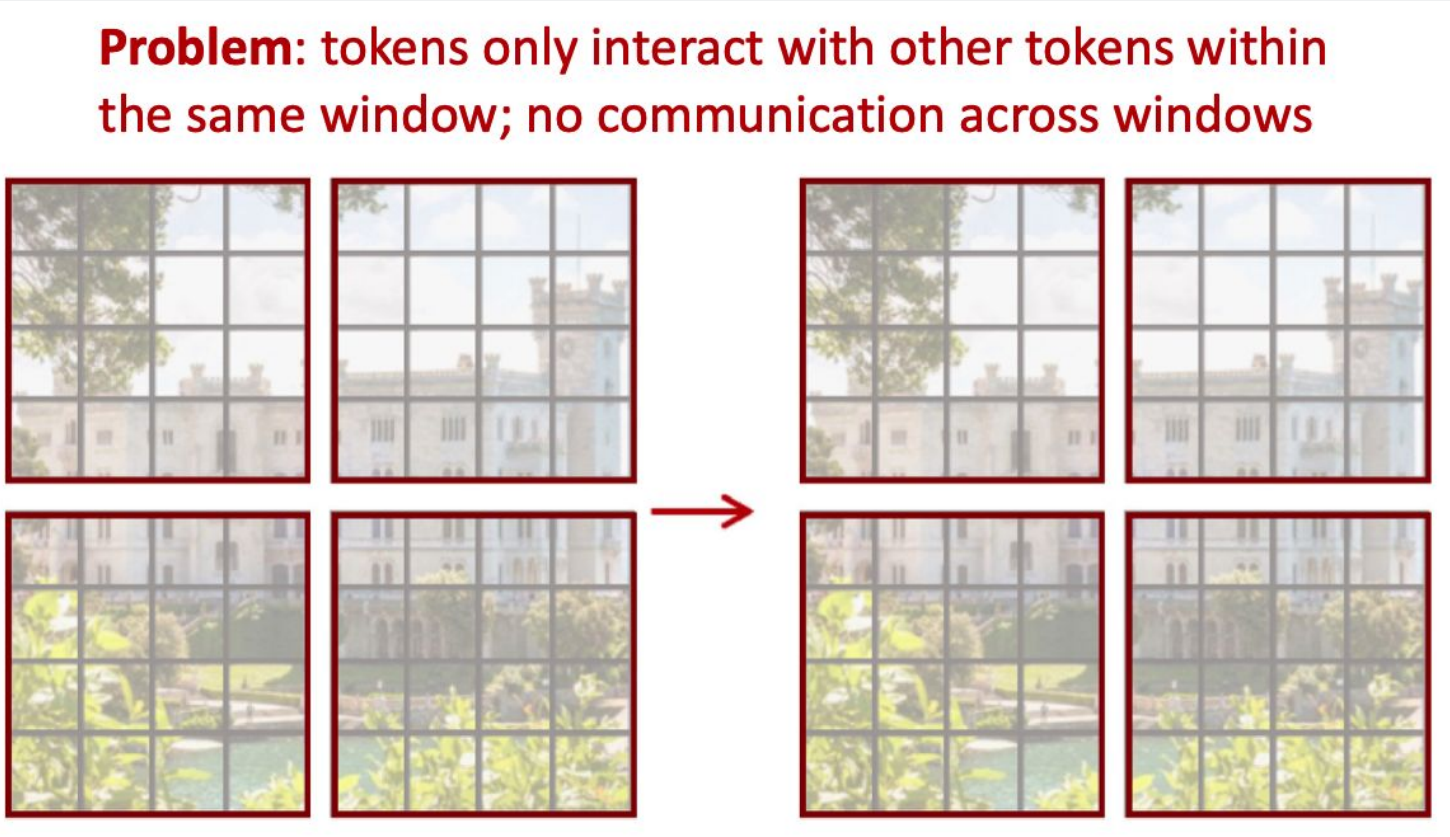
Rather than allowing each token to attend to all other tokens, instead divide into windows of M x M tokens (here M=4); only compute attention within each window

Linear w.r.t. image size

Total size of all attention matrices is now:

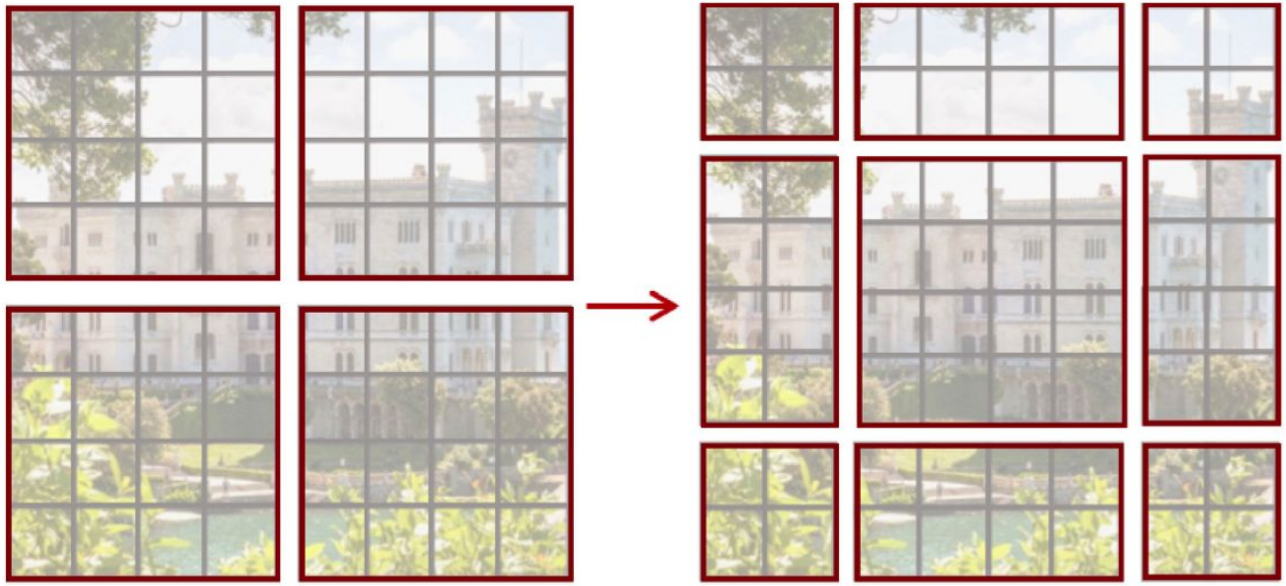
$$M^4(H/M)(W/M) = M^2HW$$

Swin Transformer: Window Attention



Swin Transformer: Shifted Window Attention

Solution: Alternate between normal windows and shifted windows in successive Transformer blocks



Block L: Normal windows

Block L+1: Shifted Windows

Detail: Relative Positional Bias

ViT adds positional embedding to input tokens, encodes *absolute position* of each token in the image

Swin does not use positional embeddings, instead encodes *relative position* between patches when computing attention:

Attention with relative bias:

$$A = Softmax\left(\frac{QK^T}{\sqrt{D}} + B\right)V$$

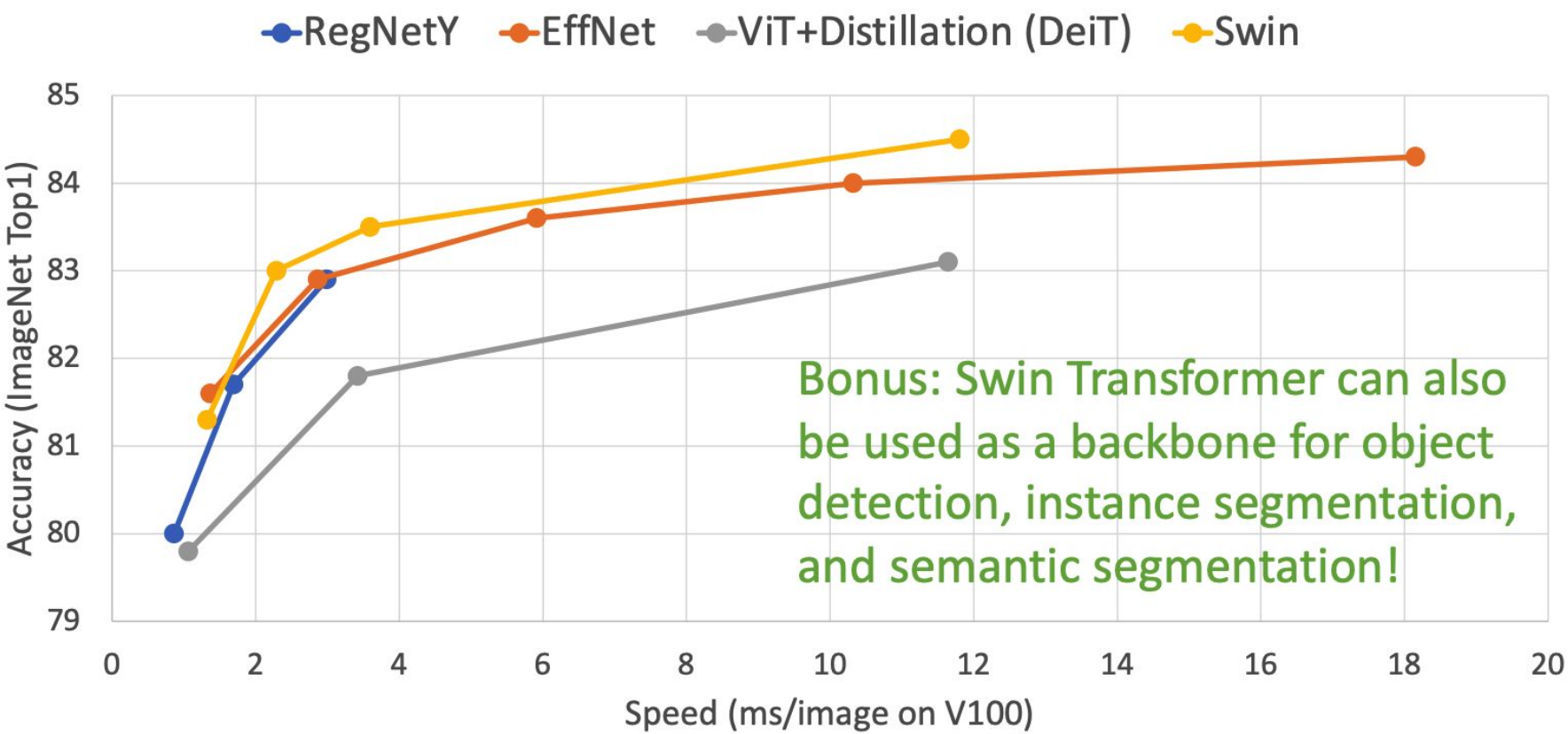
$Q, K, V: M^2 \times D$ (Query, Key, Value)

$B: M^2 \times M^2$ (learned biases)

*Non-square windows at edges and corners

M | ROBOTICS

Swin Transformer: Speed vs Accuracy



Applications

(not limited to)

Language

BERT https://arxiv.org/pdf/1810.04805
RoBERTa https://arxiv.org/pdf/1907.11692
Low Rank Adapters (LoRA) https://arxiv.org/pdf/2106.09685
GPT (Generative Pre-trained Transformer)
LLaMA (Large Language Model Meta AI)

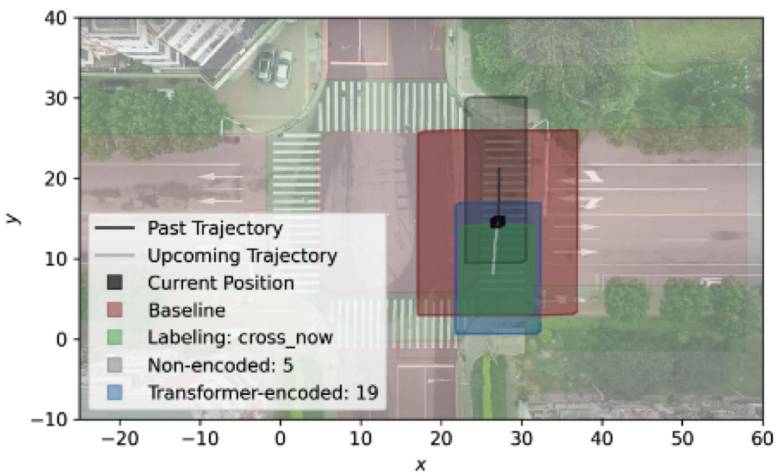
Sign	Gardiner code	Transliteration	Description
	G1	ꜣ	Egyptian vulture
	I9	f	Horned viper
	V24	wḏ	Cord wound on stick
	S12	nbw	Bead collar

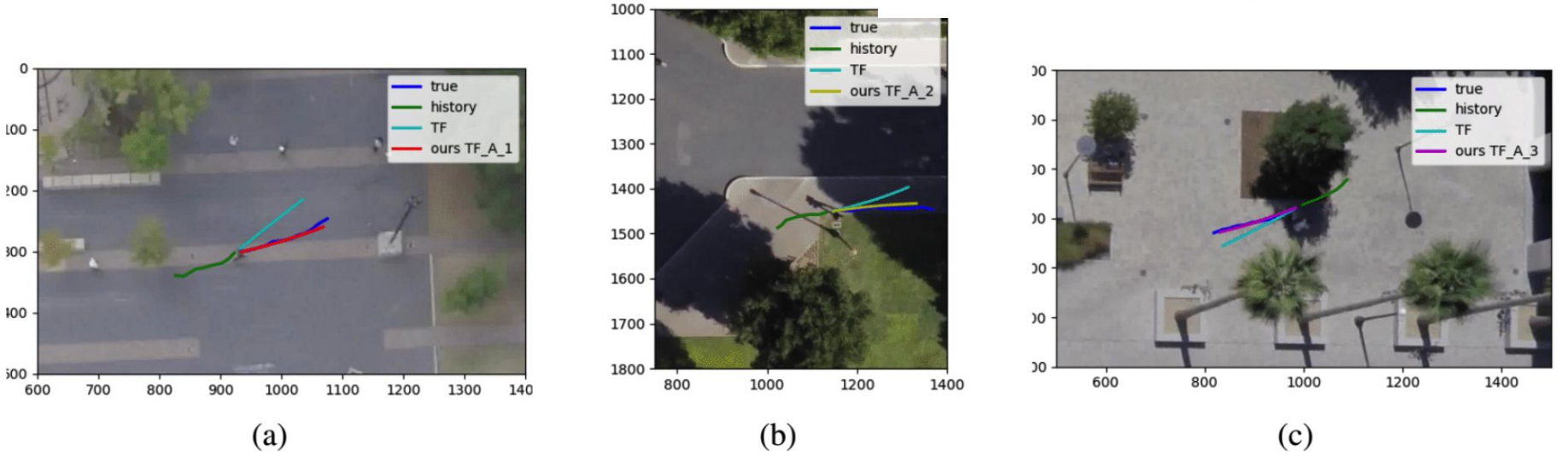
Table 1: Example of hieroglyphs and their Gardiner code, Transliteration and Description.

https://aclanthology.org/2024.ml4al-1.9.pdf

Motion/Trajectory Prediction

https://arxiv.org/pdf/2408.15250





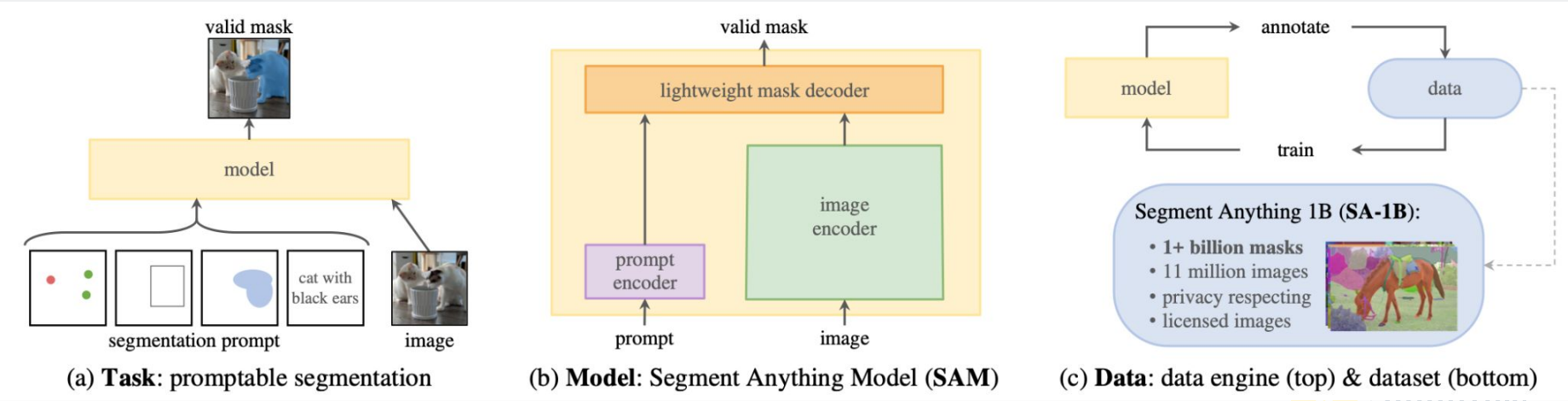
Du et al. Angle-Regulated Transformer Network for Pedestrian Trajectory Prediction [paper]

Segment Anything

Pre-trained ViT + Prompt + Mask

https://segment-anything.com
https://arxiv.org/pdf/2304.02643
SegFormer https://arxiv.org/pdf/2105.15203

"A foundation model is any model that is trained on broad data (generally using self-supervision at scale) that can be adapted (e.g., fine-tuned) to a wide range of downstream tasks" https://arxiv.org/pdf/2108.07258



(a) **Task**: promptable segmentation (b) **Model**: Segment Anything Model (**SAM**) (c) **Data**: data engine (top) & dataset (bottom)

Example: Segment Anything in Medical Images

https://www.nature.com/articles/s41467-024-44824-z.pdf (Nature Communications, 2024)

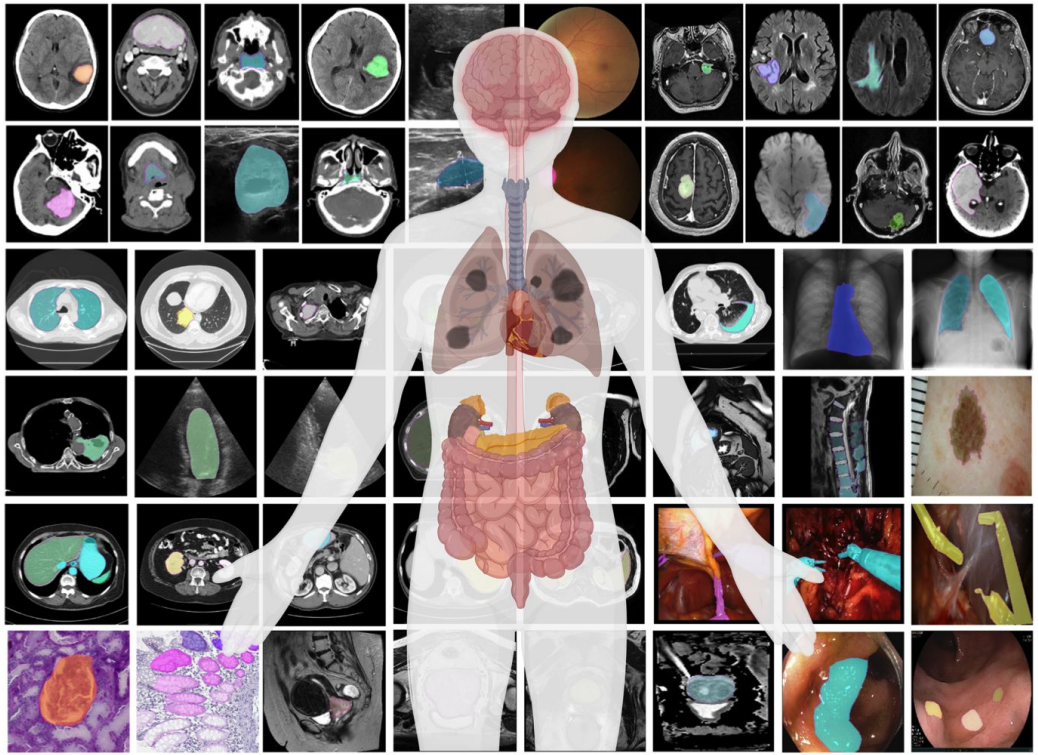
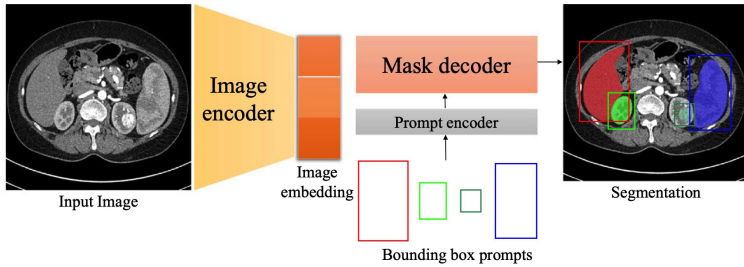


Fig. 1 | MedSAM is trained on a large-scale dataset that can handle diverse segmentation tasks. The dataset covers a variety of anatomical structures, pathological conditions, and medical imaging modalities. The magenta contours and mask overlays denote the expert annotations and MedSAM segmentation results, respectively.



M | ROBOTICS

Graph Embodiment Transformer

https://get-zero-paper.github.io (ICRA 2025)

ROBOTICS

Graph Embodiment Transformer

https://get-zero-paper.github.io (ICRA 2025)

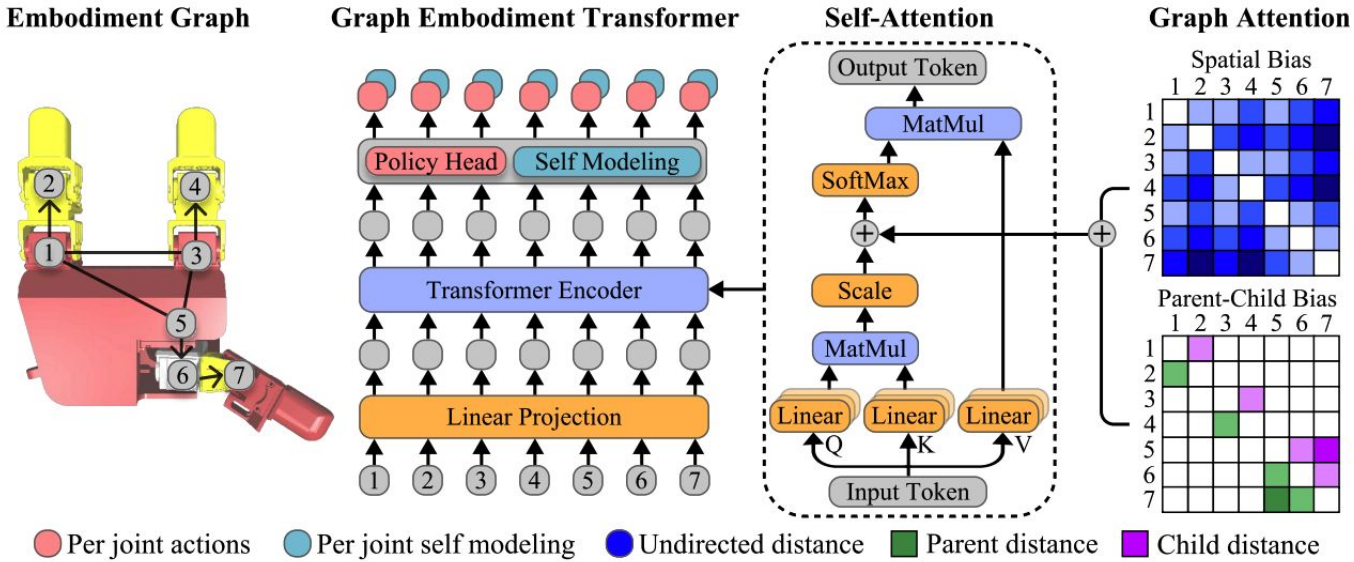
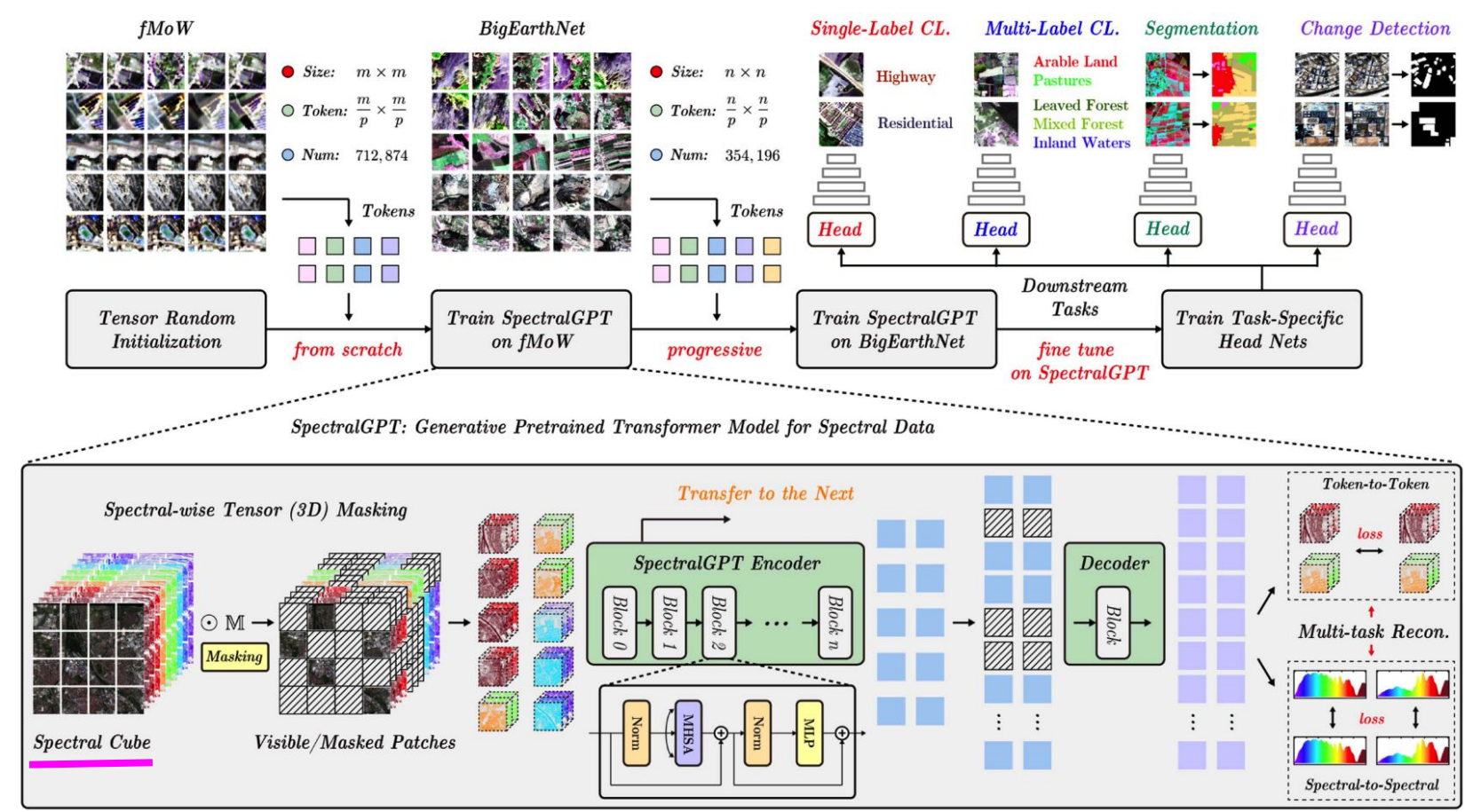


Fig. 2. **Graph Embodiment Transformer (GET).** GET is an embodiment-aware model based on a transformer encoder. Each joint forms separate tokens containing local sensory and embodiment information. The self-attention layers use an undirected (Spatial Bias) and directed (Parent-Child Bias) graph distance to bias the attention scores between joints according to the embodiment graph (grid color intensity indicates distance between nodes). A policy head predicts actions and a self modeling head predicts meta-properties about the embodiment, such as forward kinematics.

Spectral GPT

https://ieeexplore.ieee.org/document/10490262 (TPAMI, 2024)
https://github.com/danfenghong/IEEE_TPAMI_SpectralGPT



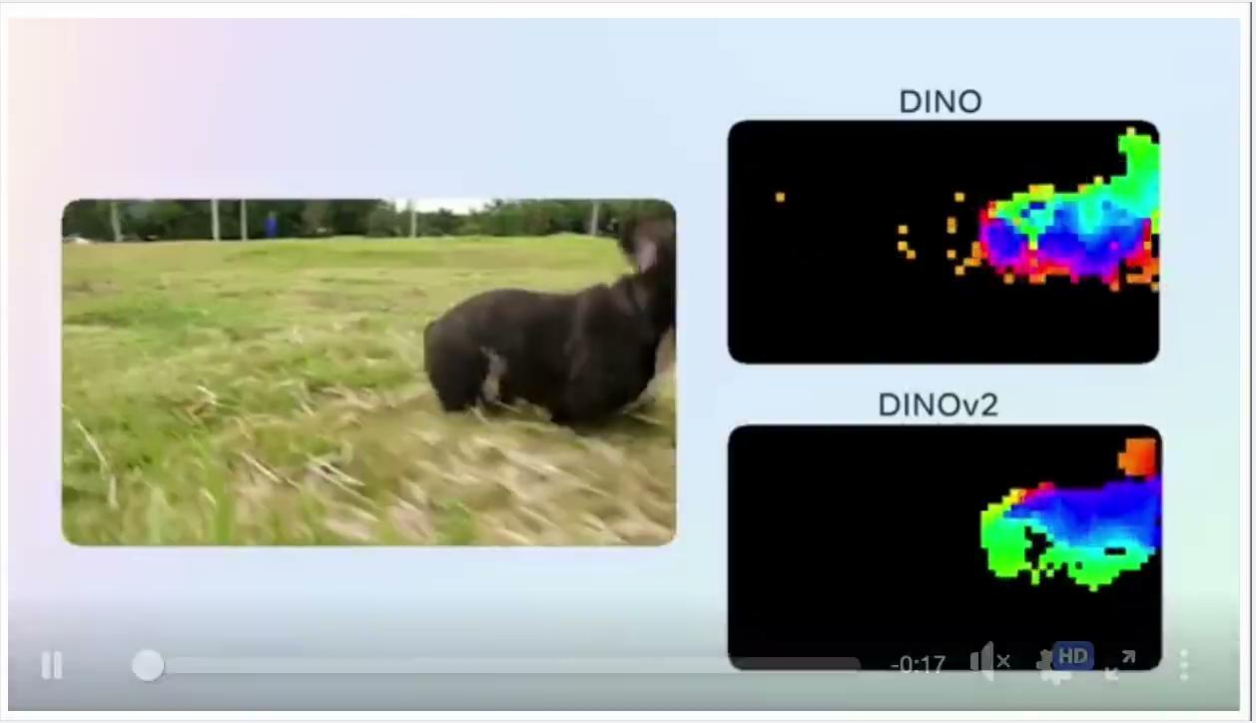
DINO

Cited by 6638 v1

Cited by 2705 v2

DINO (Self-Supervised Vision Transformer)

https://arxiv.org/pdf/2104.14294 (2021)
https://github.com/facebookresearch/dino (DINO) https://github.com/facebookresearch/dinov2 (DINO v2)
https://ai.meta.com/blog/dino-v2-computer-vision-self-supervised-learning/ DINO v2 (2023)



Knowledge Distillation

https://arxiv.org/pdf/1503.02531

- transferring knowledge from a large model to a smaller one
- a **model compression** method in which a small model is trained to mimic a pre-trained, larger model (or ensemble of models).
- a process of distilling or transferring the knowledge from a (set of) large, cumbersome model(s) to a lighter, easier-to-deploy single model, without significant loss in performance

Problem:
Ensemble model - too cumbersome, maybe too computationally expensive

Solution:
"Distillation" to transfer knowledge from large to small

"Teacher-Student" model

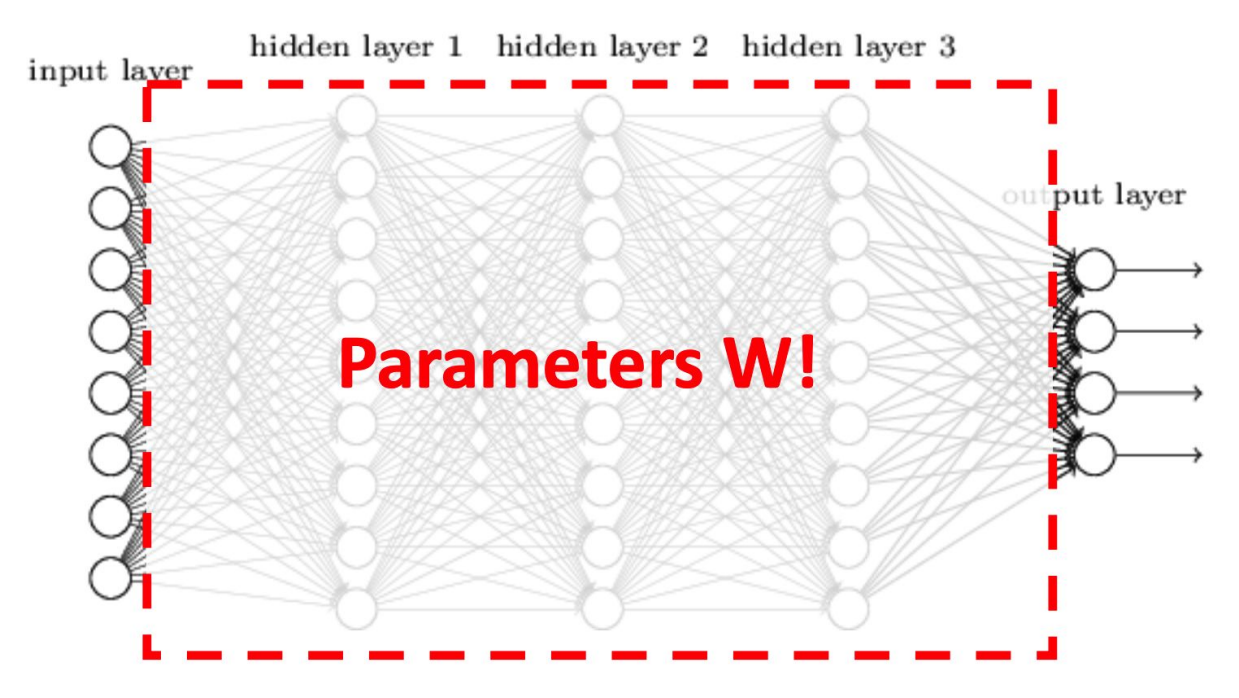
a larger pre-trained network or an ensemble of models

compact (smaller) network

M | ROBOTICS

Knowledge Distillation (KD)

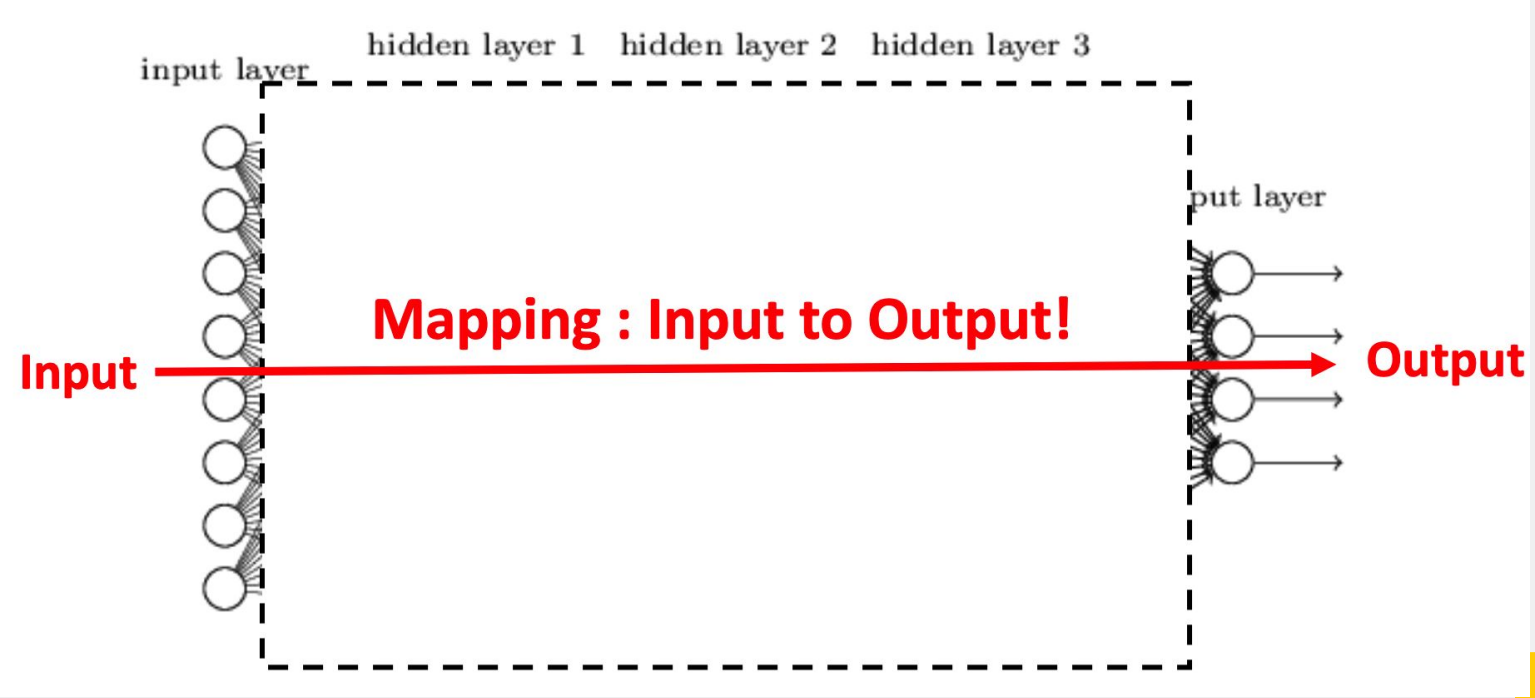
Q: What is “Knowledge”?



Knowledge Distillation (KD)

Q: What is “Knowledge”?

learned mapping from input vectors to output vectors



Knowledge Distillation (KD)

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

demo

softmax

"temperature"

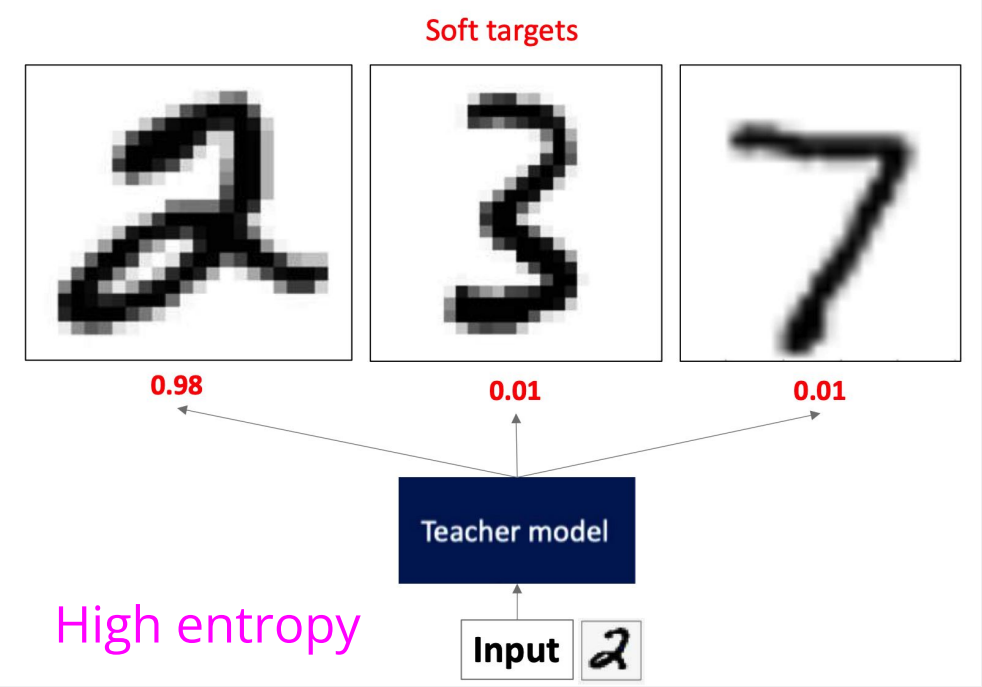
Using a higher value for T produces a **softer** probability distribution over classes

M | ROBOTICS

Knowledge Distillation (KD)

use the class probabilities produced by the cumbersome (big) model as "soft targets" for training the small model.

Q: Why soft target?



Soft target

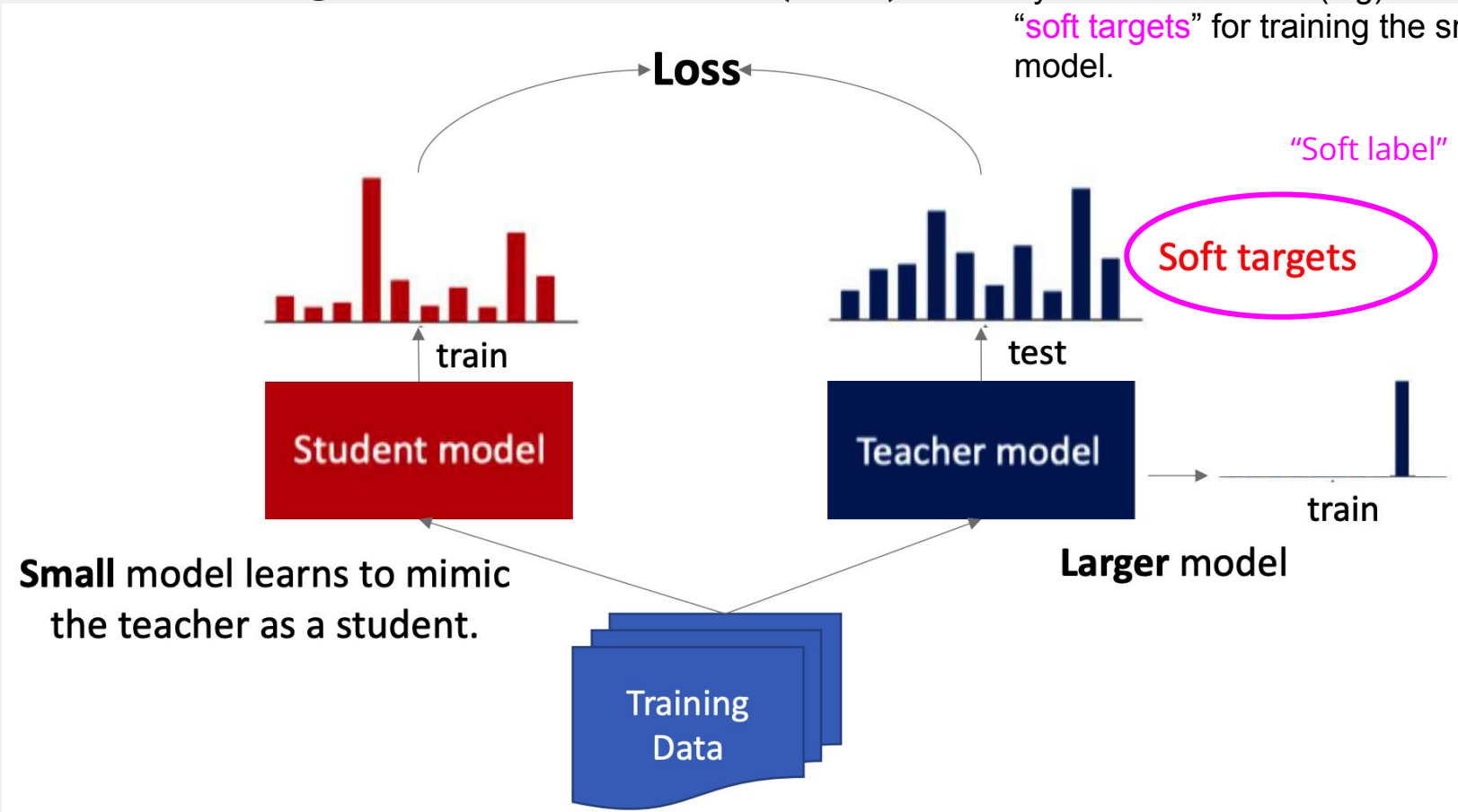
- 2 is similar to 3 and 7
- Contiguous distribution
- **Inter-Class variance** ✓
- **Between-Class distance** ✓

One-hot

- 2 independent of 3 and 7
- Discrete distribution
- **Inter-Class variance**
- **Between-Class distance**

Knowledge Distillation (KD)

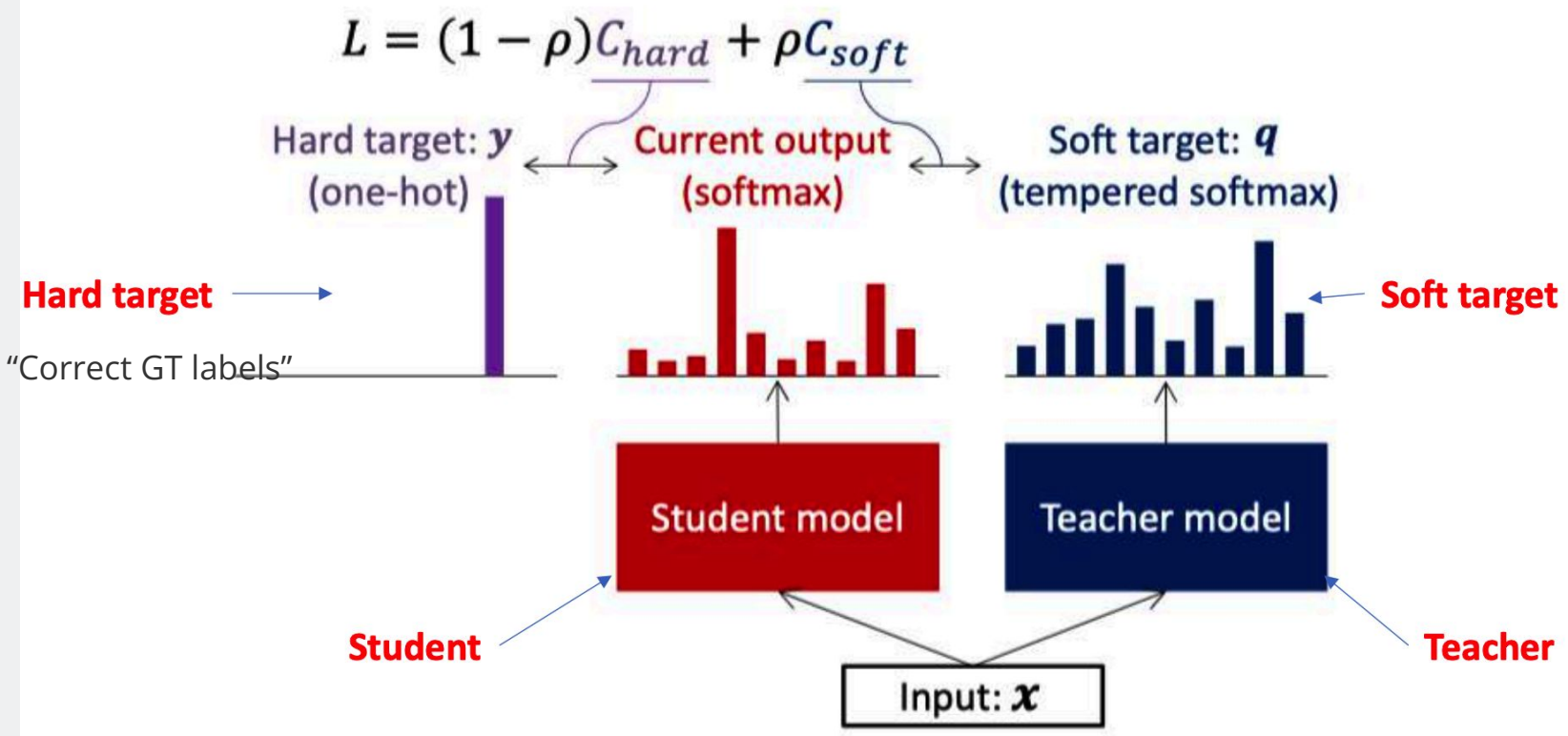
use the class probabilities produced by the cumbersome (big) model as "soft targets" for training the small model.



Knowledge Distillation (KD)

"we found that a better way is to simply use a weighted average of two different objective functions"
https://arxiv.org/pdf/1503.02531

Transfer set = unlabeled data + original training set



KL Divergence

https://medium.com/@buroojghani/why-kl-divergence-in-knowledge-distillation-1375d555a728

Cross Entropy loss

GT pred

$$H(q,p) = -\sum_i q(i) \log p(i)$$

a measure of how much a model probability distribution P is different from a true probability distribution Q

Kullback–Leibler divergence

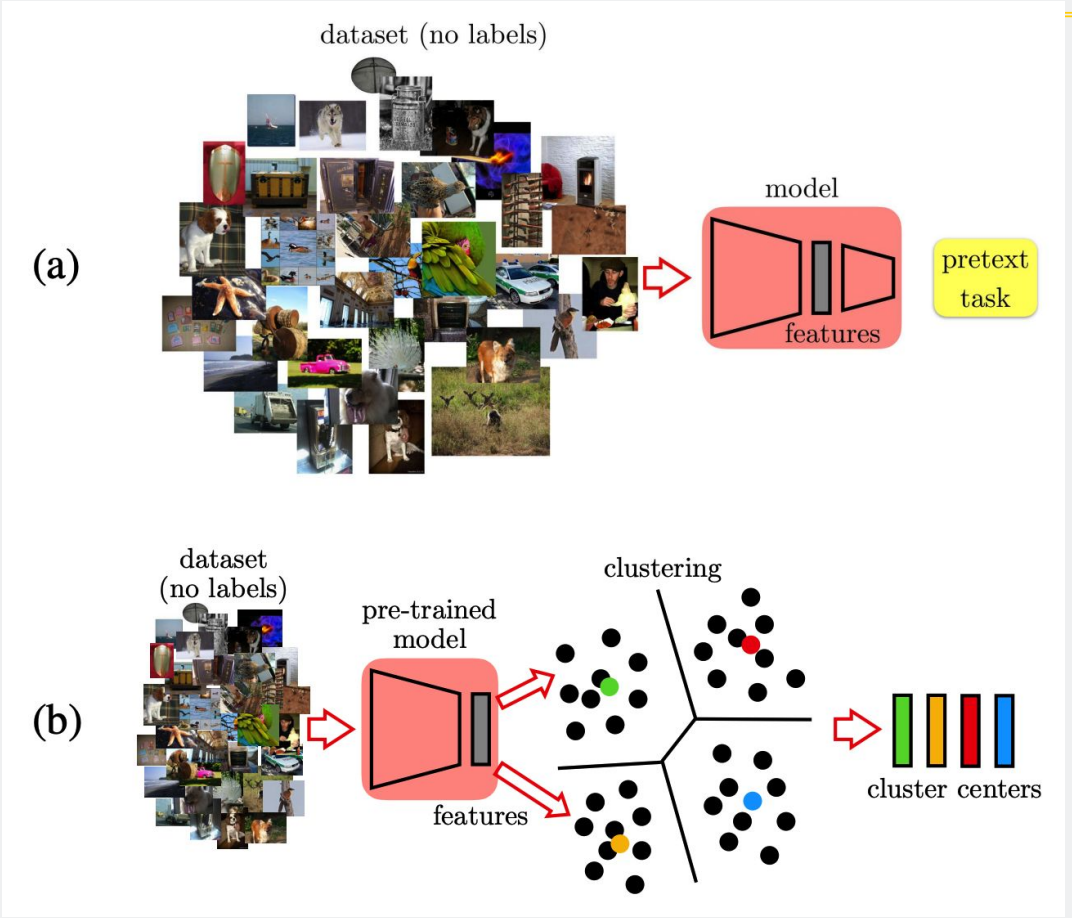
$$D_{\mathrm{KL}}(q \parallel p) = \sum_i q(i) \log \frac{q(i)}{p(i)} = H(q,p) - H(q)$$

Higher KL divergence ⇔ P&Q are more different

M | ROBOTICS

Self-Supervised Learning (SSL)

https://arxiv.org/pdf/1805.00385



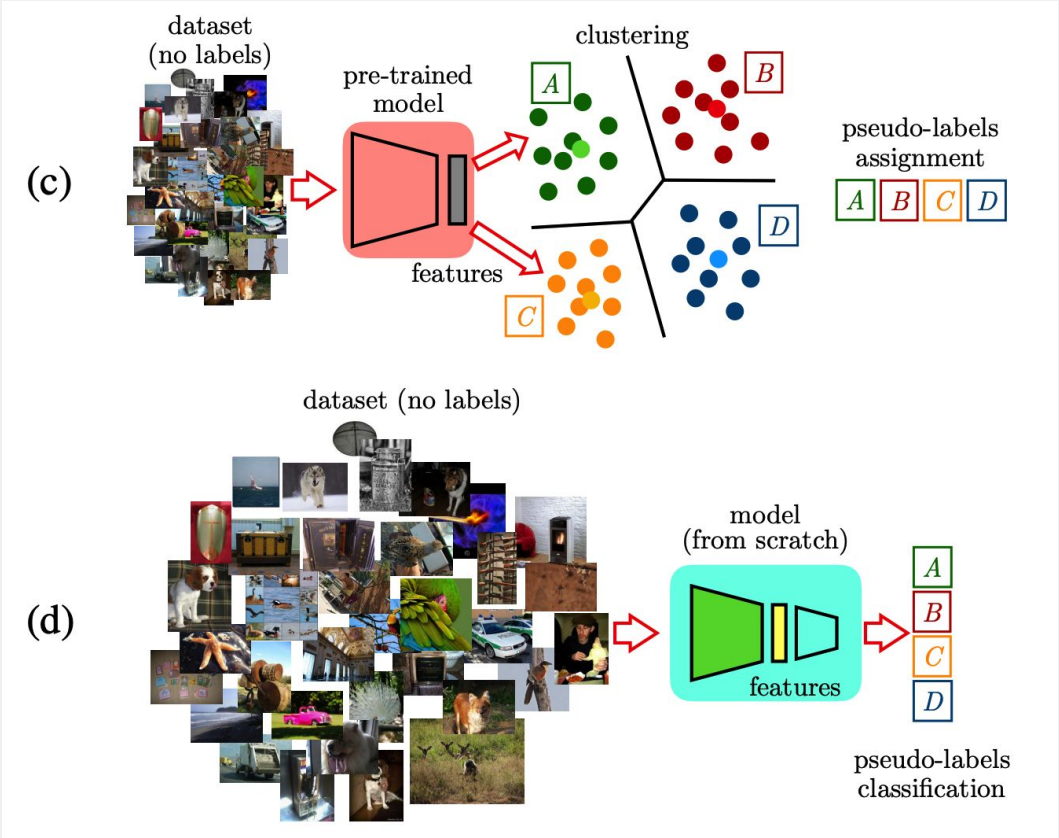
Learn with
no labels

(a) SSL Pre-Training
(b) Clustering

M | ROBOTICS

Self-Supervised Learning (SSL)

https://arxiv.org/pdf/1805.00385



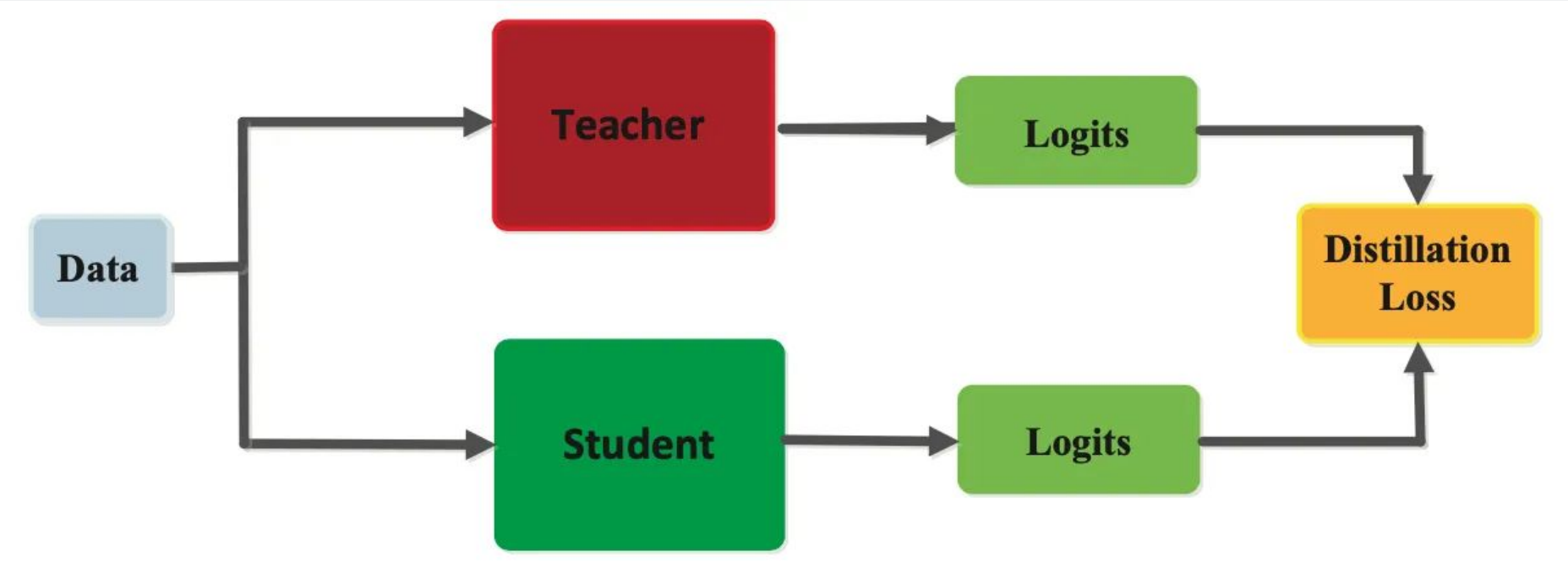
Learn with
no labels

(c) Pseudo-Labels (cluster centers)

(d) train a classifier based on pseudo-labels

M | ROBOTICS

DINO: Distillation with NO Labels



DINO: Distillation with NO Labels

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

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, 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 = gs(x1), gs(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(gs) # SGD
    gt.params = l*gt.params + (1-l)*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()
```

Random views/ crops of images

(data augmentation)

2 *global* view + several *local* view

M | ROBOTICS

DINO: Distillation with NO Labels

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

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, 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 = gs(x1), gs(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(gs) # SGD
    gt.params = l*gt.params + (1-l)*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()
```

Random views/ crops of images

(data augmentation)

2 *global* view + several *local* view

- Student: all crops
- Teacher: only global views

Encourages “local-to- global” correspondences

M | ROBOTICS

DINO: Distillation with NO Labels

https://dinov2.metademolab.com/

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

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, 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 = gs(x1), gs(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(gs) # SGD
    gt.params = l*gt.params + (1-l)*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()
```

ViT

Exponential moving average (EMA)

Cross entropy loss

M | ROBOTICS

Mamba
(VMamba, Vision Mamba …)

Cited by 2663

Mamba (2024)

https://arxiv.org/pdf/2312.00752
https://github.com/state-spaces/mamba

"Structured state space sequence models" (S4)

Input: $x(t) \in \mathbb{R}$

Output: $y(t) \in \mathbb{R}$

$$\mathbf{A} \in \mathbb{R}^{N \times N}$$

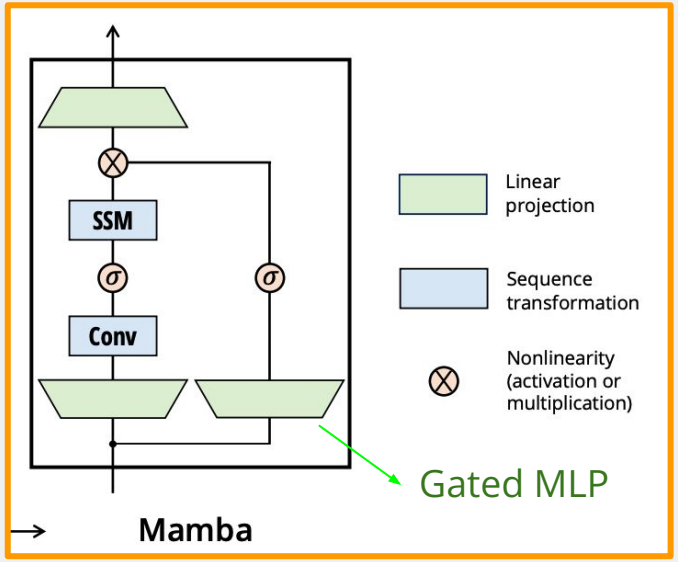
Evolution parameter

$$h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t),$$
$$y(t) = \mathbf{W}h'(t),$$

Projection parameter

$$\mathbf{B} \in \mathbb{R}^{N \times 1}, \mathbf{W} \in \mathbb{R}^{1 \times N}$$

- Sequence-to-sequence transformation
- 1D sequence
- Continuous -> discrete



M | ROBOTICS

Vision Mamba

https://arxiv.org/pdf/2401.09417

2D vision task

- spatial-aware
- bidirectional sequence modeling

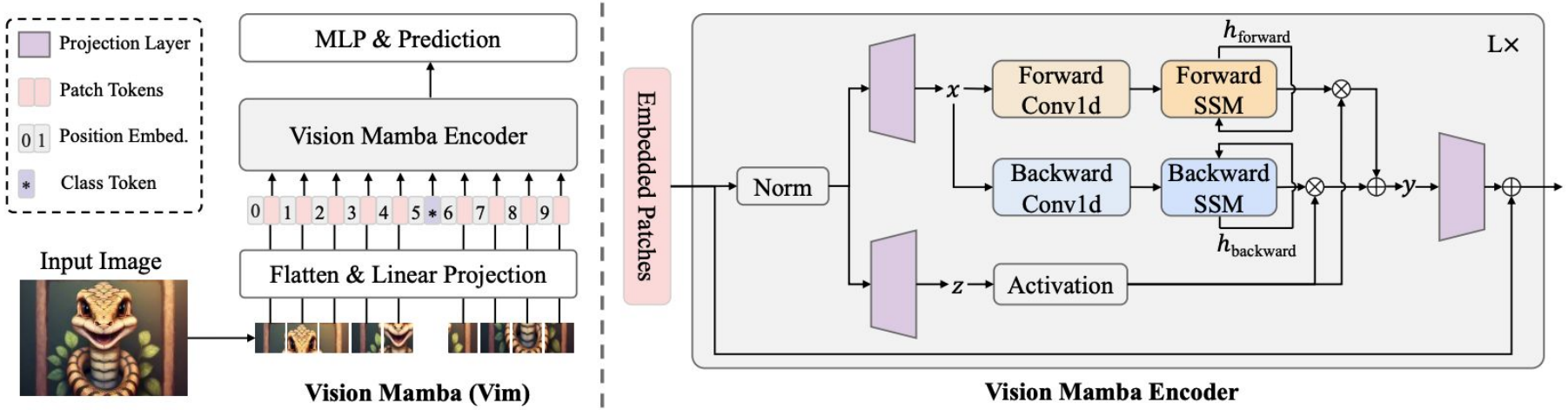


Figure 2: The overview of the proposed Vim model. We first split the input image into patches, and then project them into patch tokens. Last, we send the sequence of tokens to the proposed Vim encoder. To perform ImageNet classification, we concatenate an extra learnable classification token to the patch token sequence. Different from Mamba for text sequence modeling, Vim encoder processes the token sequence with both forward and backward directions.

VMamba

https://proceedings.neurips.cc/paper_files/paper/2024/file/baa2da9ae4bfed26520bb61d259a3653-Paper-Conference.pdf

"SSM-based vision backbone for visual representation learning with linear time complexity"

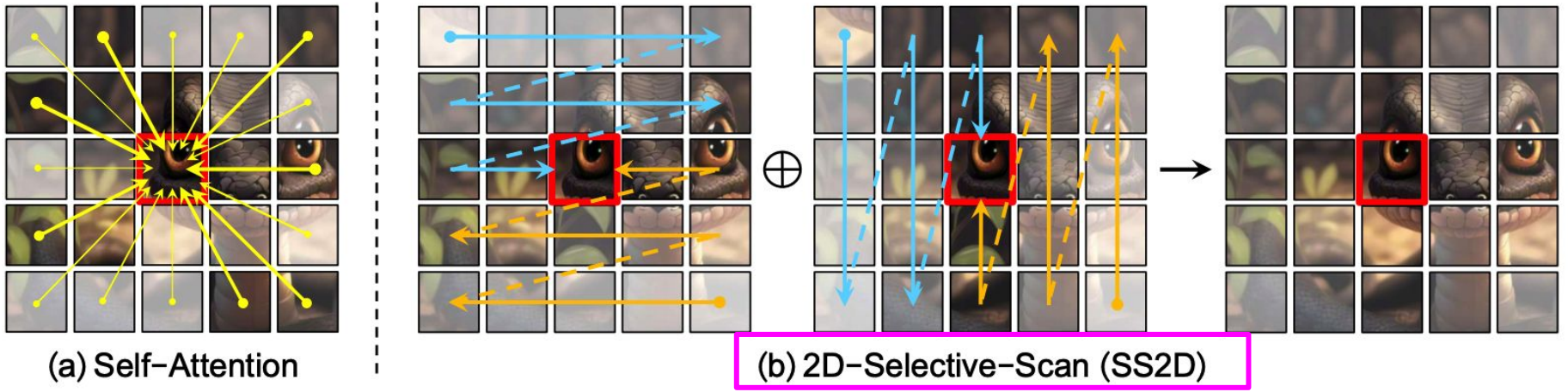


Figure 1: Comparison of the establishment of correlations between image patches through (a) self-attention and (b) the proposed 2D-Selective-Scan (SS2D). The red boxes indicate the query image patch, with its opacity representing the degree of information loss.

M | ROBOTICS

VMamba

https://proceedings.neurips.cc/paper_files/paper/2024/file/baa2da9ae4bfed26520bb61d259a3653-Paper-Conference.pdf

SS2D

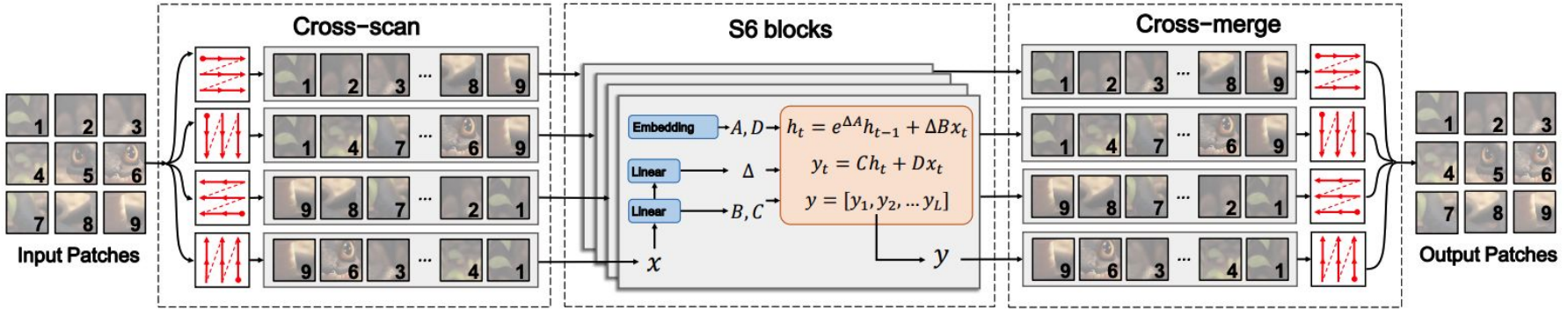


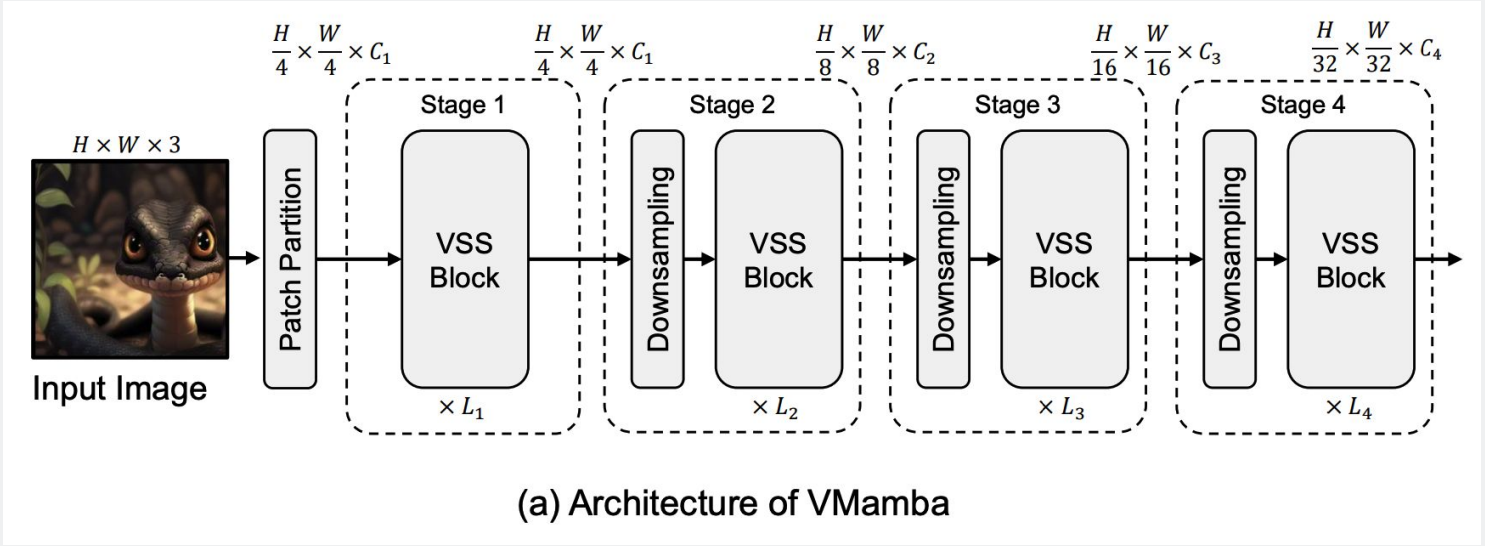
Figure 2: Illustration of 2D-Selective-Scan (SS2D). Input patches are traversed along four different scanning paths (*Cross-Scan*), with each sequence independently processed by separate S6 blocks. The results are then merged to construct a 2D feature map as the final output (*Cross-Merge*).

M | ROBOTICS

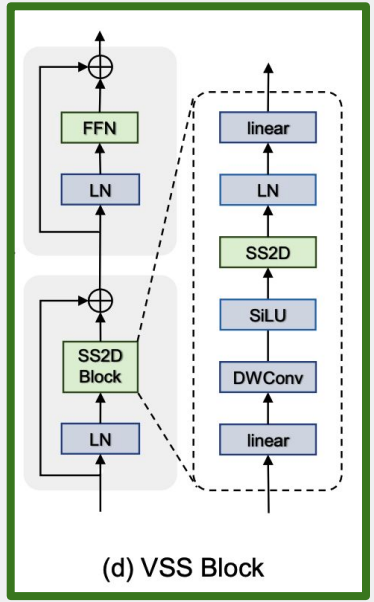
VMamba

https://proceedings.neurips.cc/paper_files/paper/2024/file/baa2da9ae4bfed26520bb61d259a3653-Paper-Conference.pdf

"SSM-based vision backbone for visual representation learning with linear time complexity"



(a) Architecture of VMamba



(d) VSS Block

*Reminds of SwinTransformer?

M | ROBOTICS

More on Mamba…

- Vision Mamba: A Comprehensive Survey and Taxonomy
  https://arxiv.org/pdf/2405.04404
- Video Mamba: https://arxiv.org/pdf/2403.06977
- Motion Mamba: https://arxiv.org/pdf/2403.07487
- MambaOut: Do We Really Need Mamba for Vision?
  https://arxiv.org/pdf/2405.07992 (CVPR 2025)
  https://github.com/yuweihao/MambaOut