ROB 498/599: Deep Learning for Robot Perception (DeepRob)

Lecture 19: More Transformers

(Swin, DINO, Mamba, etc.)

03/26/2025

https://deeprob.org/w25/

Today

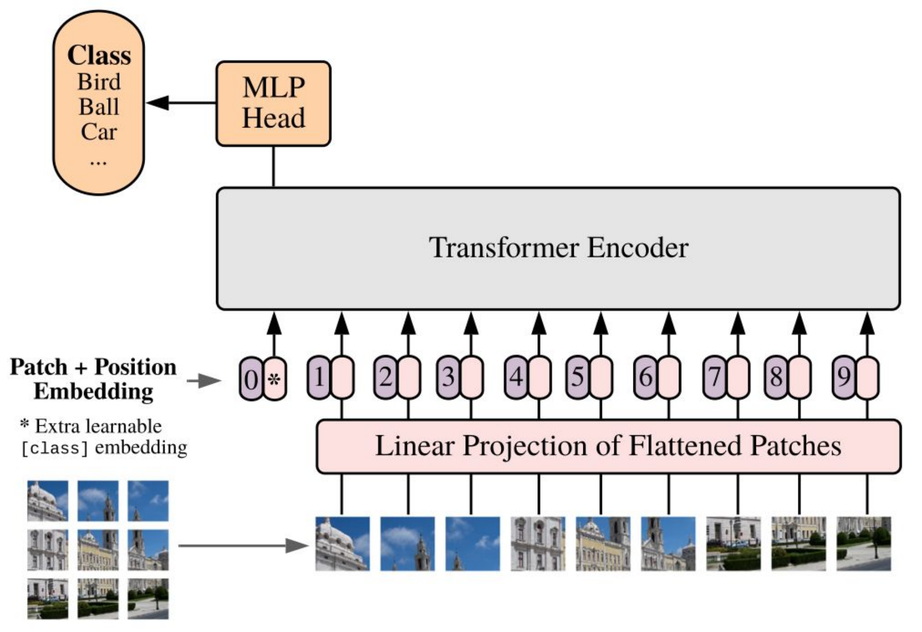
- Feedback and Recap (5min)
- Swin Transformer (25min)
- DINO (25min)
- Mamba (10min)
- Applications (10min)
- Summary and Takeaways (5min)

Reminders

- P4 Due March 30, 2025
  - Last test case “test_vit_model” has error on autograder, ignore for now (aim for 70/75) @198
  - FAQ: removing the ReLU at the end of translation and rotation branch (still need ReLU at the end of segmentation branch) @193
- Final Project Reminder
  - April 1st, 5-min poster “lightning talk” @ CSRB 2246 (everyone speaks)
  - April 22nd, final project showcase @FRB atrium
  - April 28th, final project (report, code, video/website) DUE

https://deeprob.org/w25/projects/finalproject/

Recap: ViT and CNN



https://ahaslides.com/BZRHL

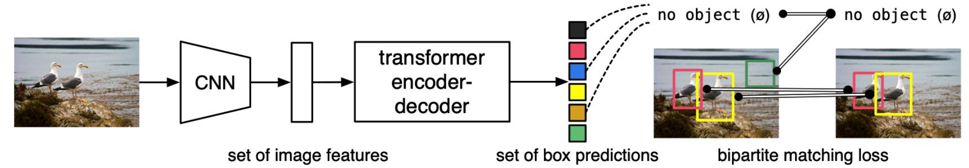


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "`no object`" (∅) class prediction.

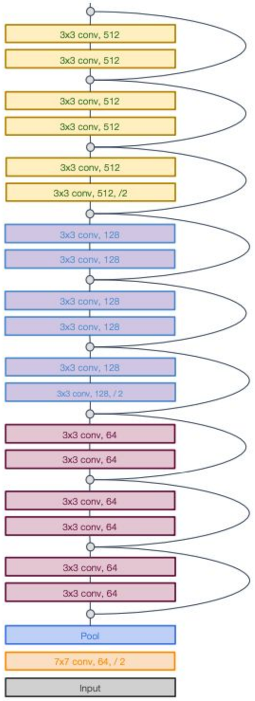
ViT vs. CNN

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

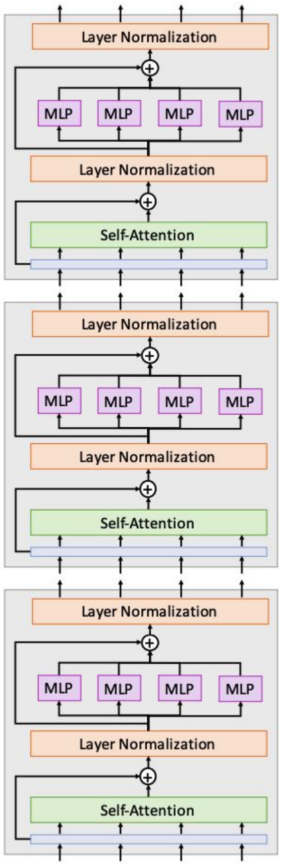


In most CNNs (including ResNets), **decrease** resolution and **increase** channels as you go deeper in the network (Hierarchical architecture)

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

hierarchical feature maps

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

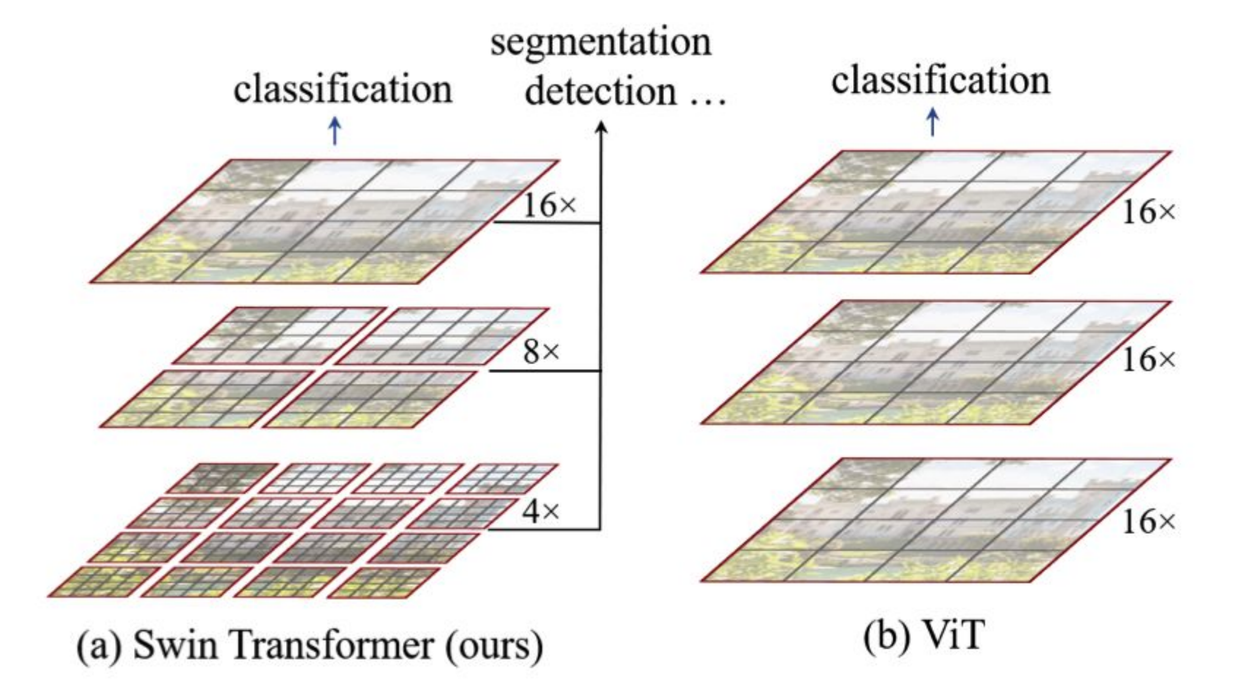
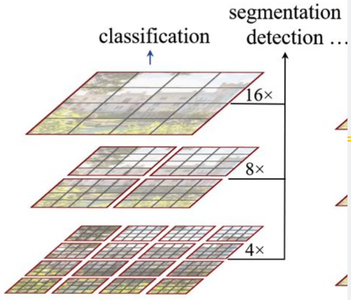


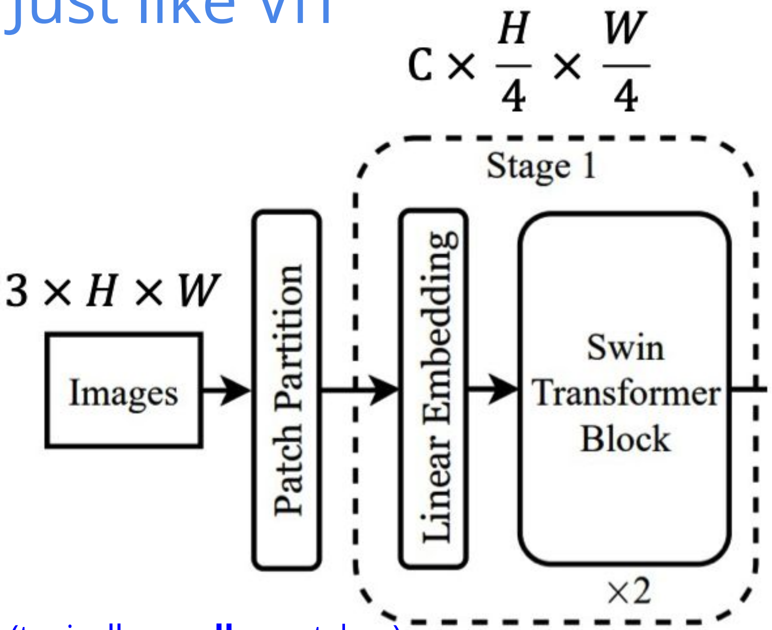
Figure 1. (a) The proposed Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window (shown in red). It can thus serve as a general-purpose 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.

Swin Transformer: Hierarchical ViT



(a) Swin Transformer (ours)

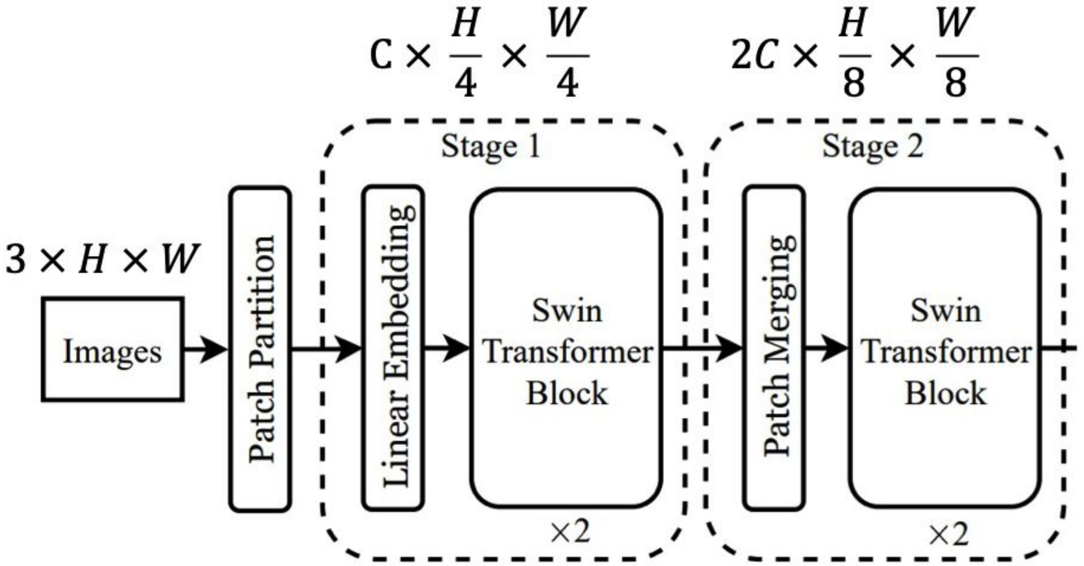
Just like ViT



(typically **smaller** patches)

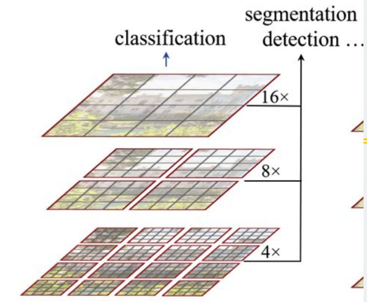
Divide image into 4x4 patches and project to C dimensions

Swin Transformer: Hierarchical ViT



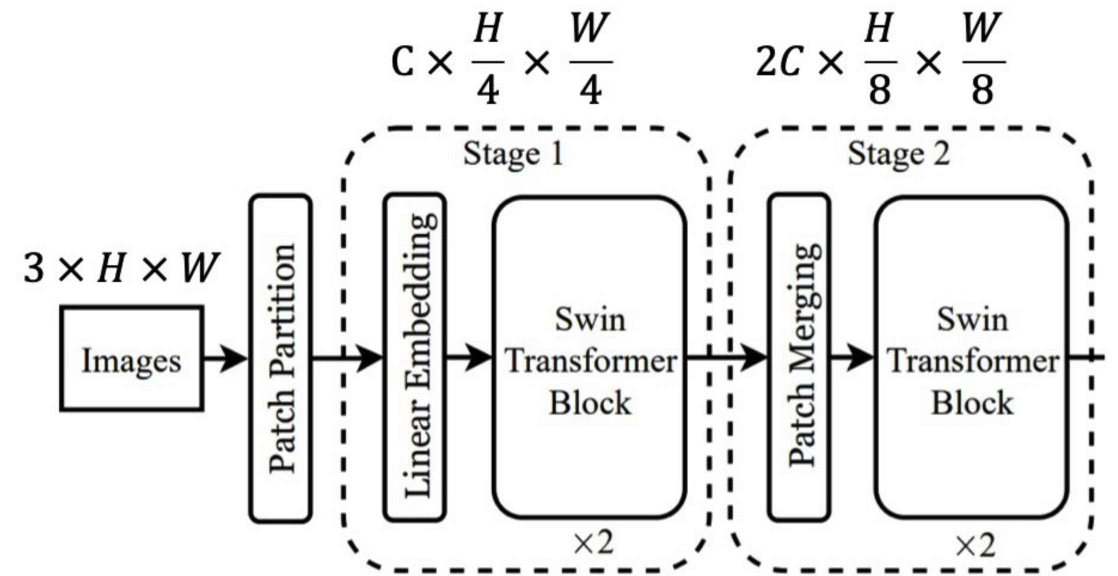
Divide image into 4x4 patches and project to C dimensions

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



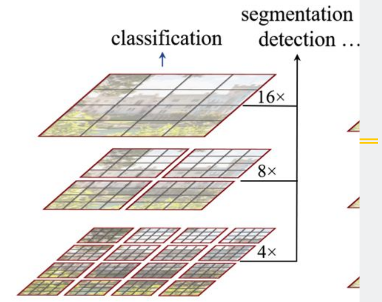
(a) Swin Transformer (ours)

Swin Transformer: Hierarchical ViT

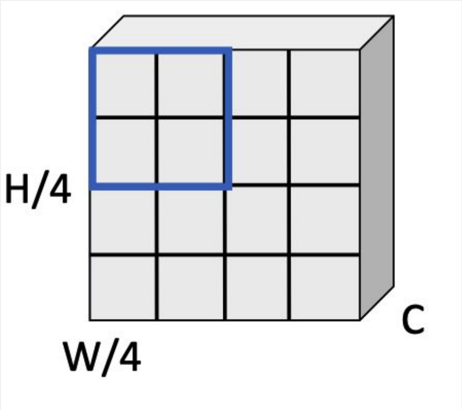


Divide image into 4x4 patches and project to C dimensions

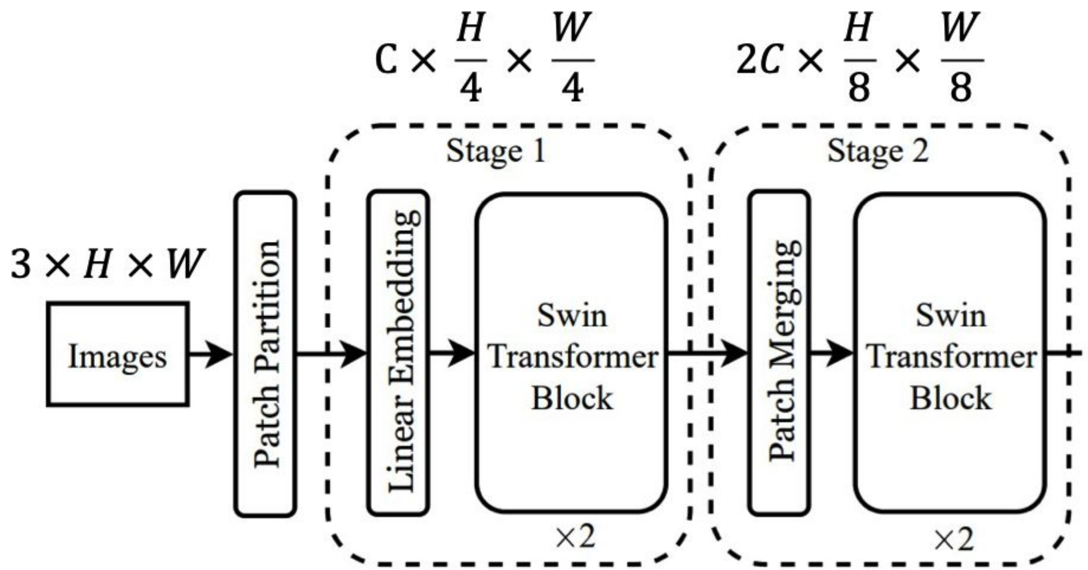
Merge 2x2 neighborhoods; now patches are (effectively) 8x8



(a) Swin Transformer (ours)

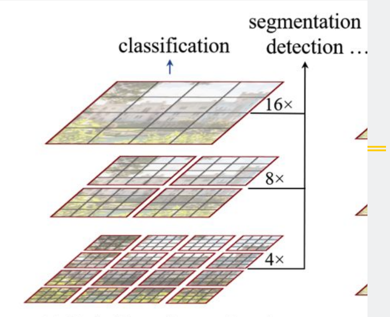


Swin Transformer: Hierarchical ViT

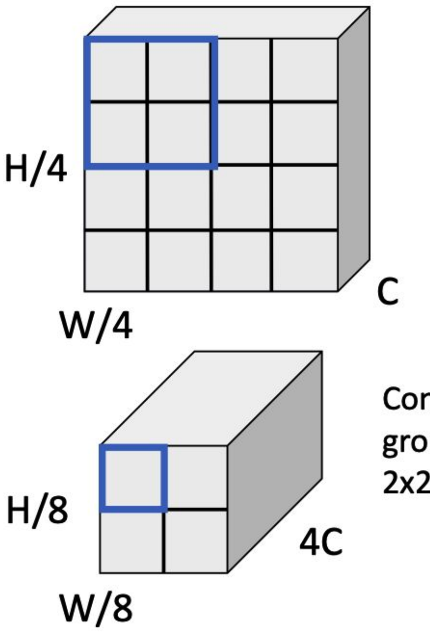


Divide image into 4x4 patches and project to C dimensions

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



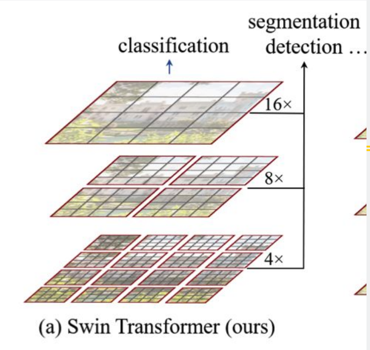
(a) Swin Transformer (ours)

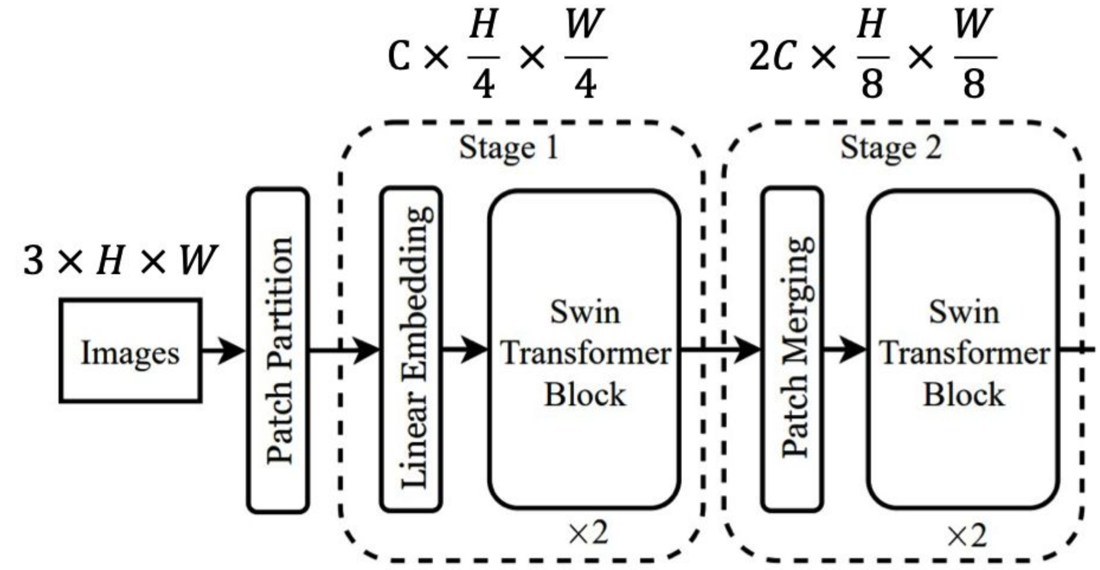


Lower spatial dimension, **higher** channel dimension

Concatenate groups of 2x2 features

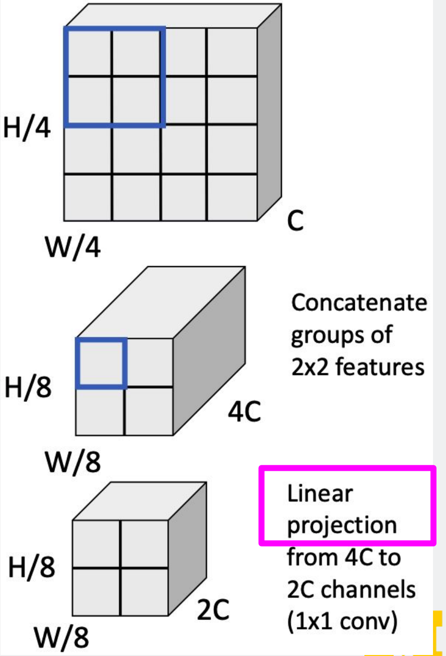
Swin Transformer: Hierarchical ViT



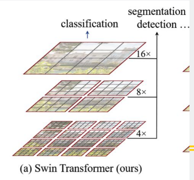


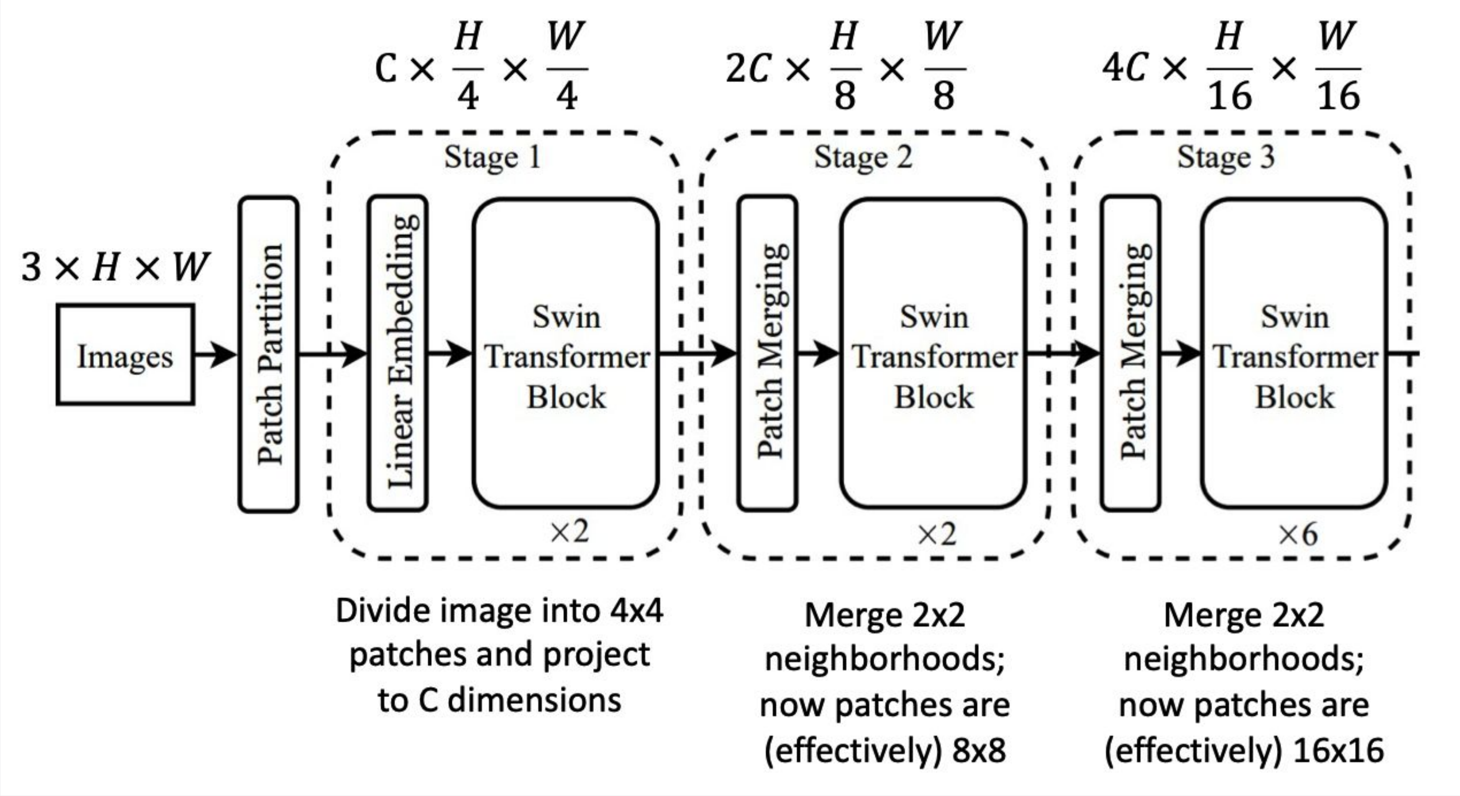
Divide image into 4x4 patches and project to C dimensions

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

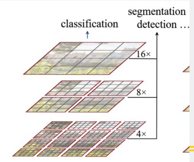


Swin Transformer: Hierarchical ViT





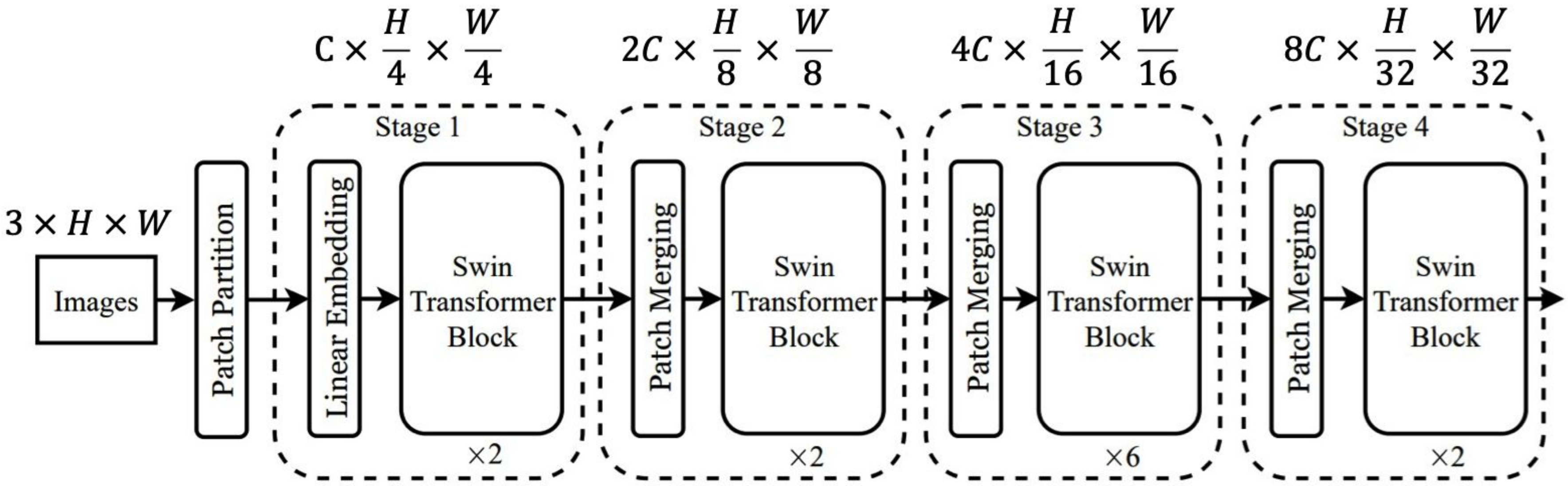
Swin Transformer: Hierarchical ViT



(a) Swin Transformer (ours)

Goal: Find high-resolution, fine-grain features

Stacks



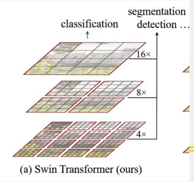
Divide image into 4x4 patches and project to C dimensions

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

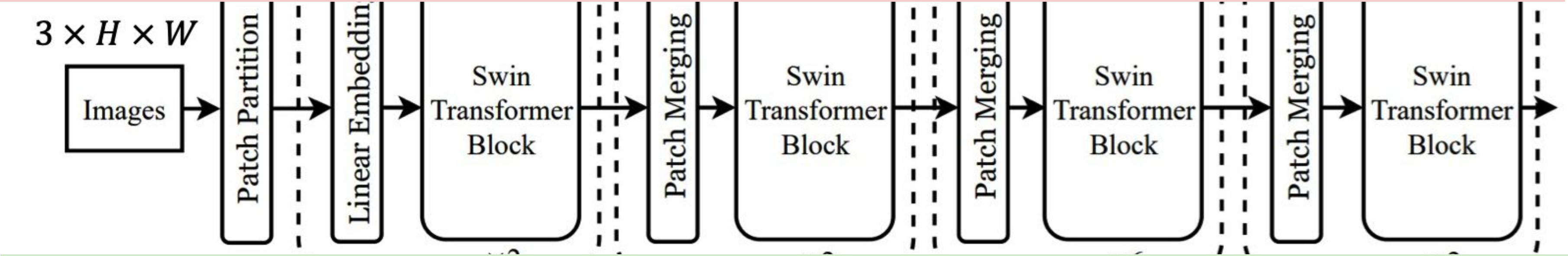
Merge 2x2 neighborhoods; now patches are (effectively) 16x16

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

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

Total size of all attention matrices is now:

Linear w.r.t. image size

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

Usually, small M

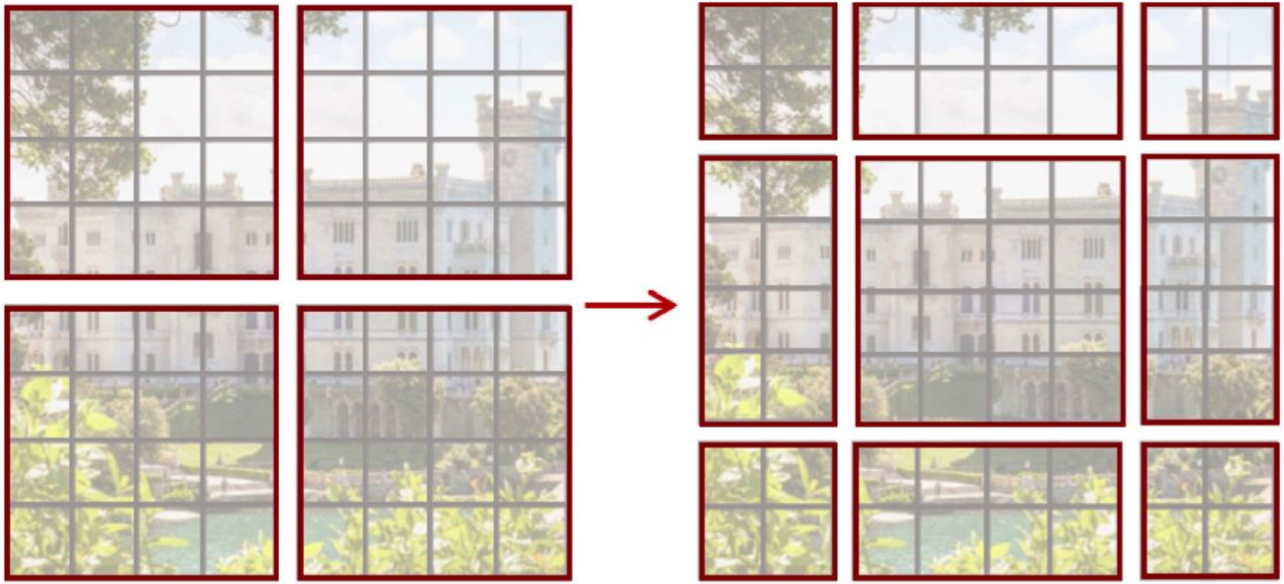
Swin Transformer: Window Attention

Problem: tokens only interact with other tokens within the same window; no communication across windows

ROBOTICS

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:

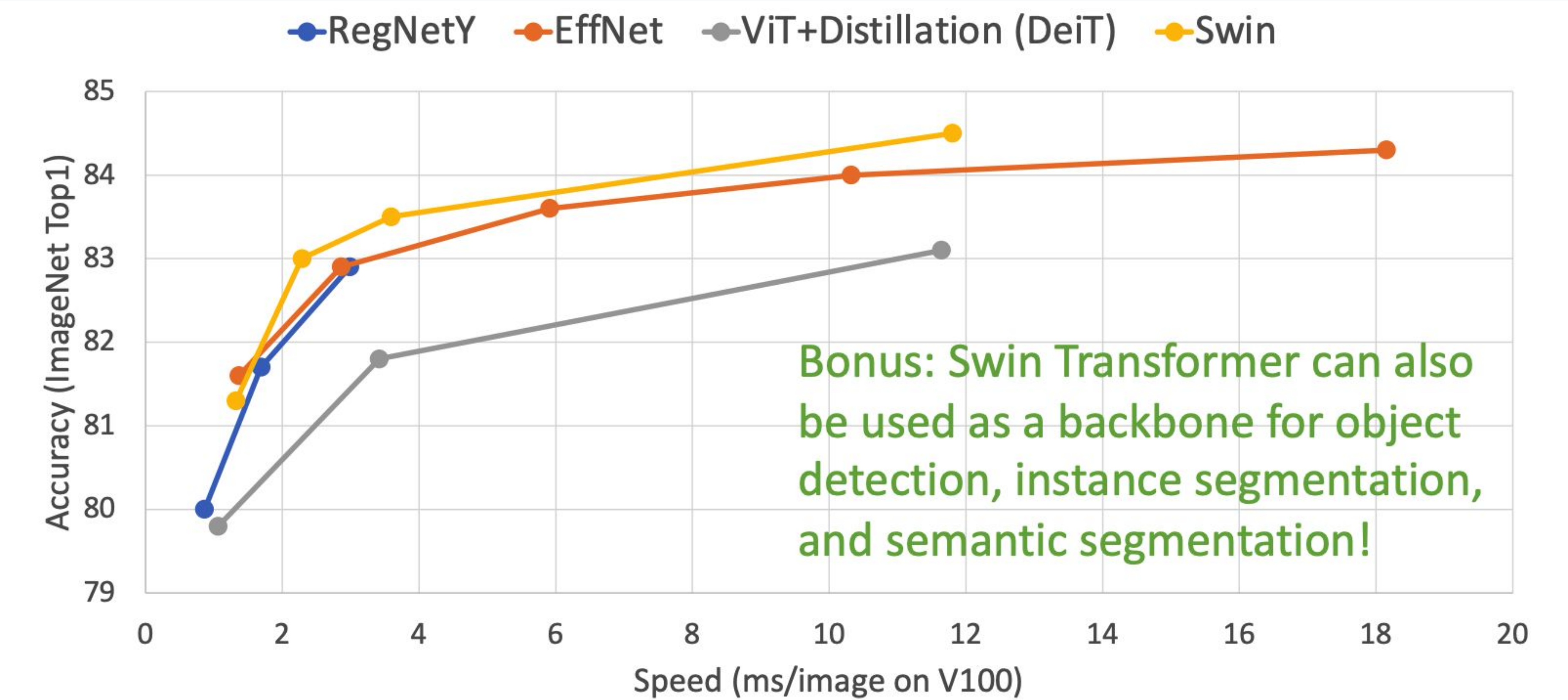
Attention with relative bias:

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

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

*Non-square windows at edges and corners

Swin Transformer: Speed vs Accuracy



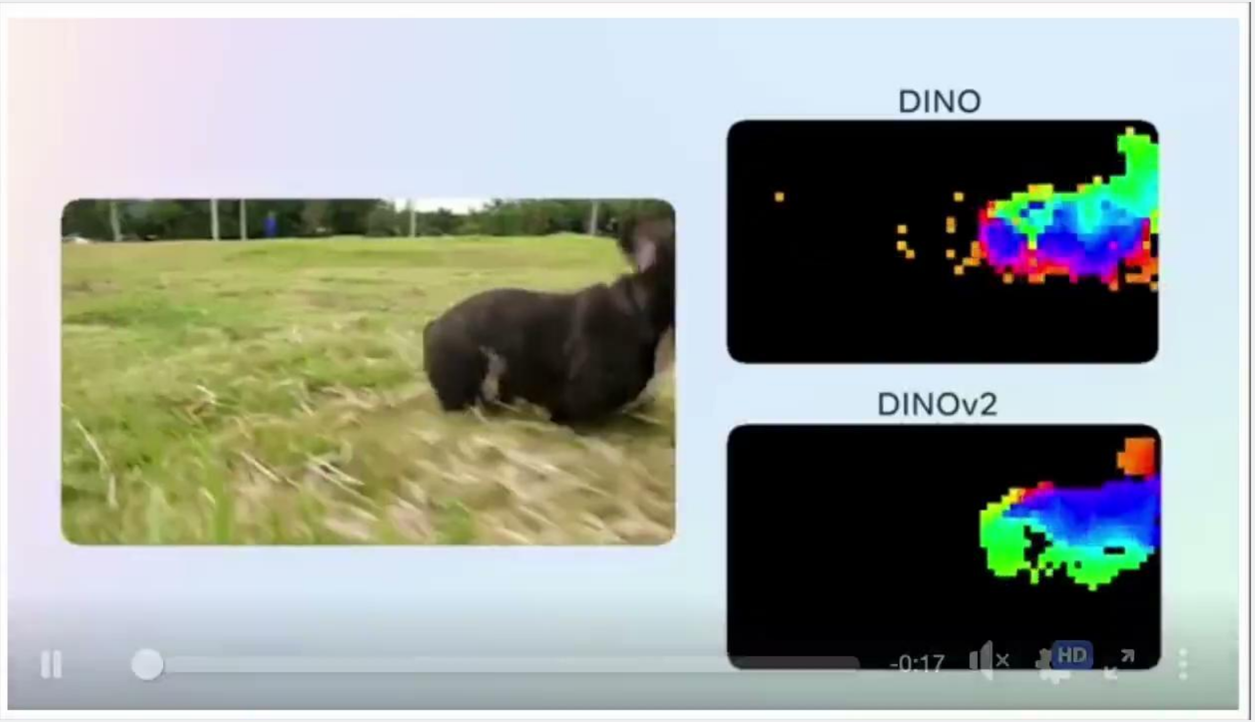
DINO

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

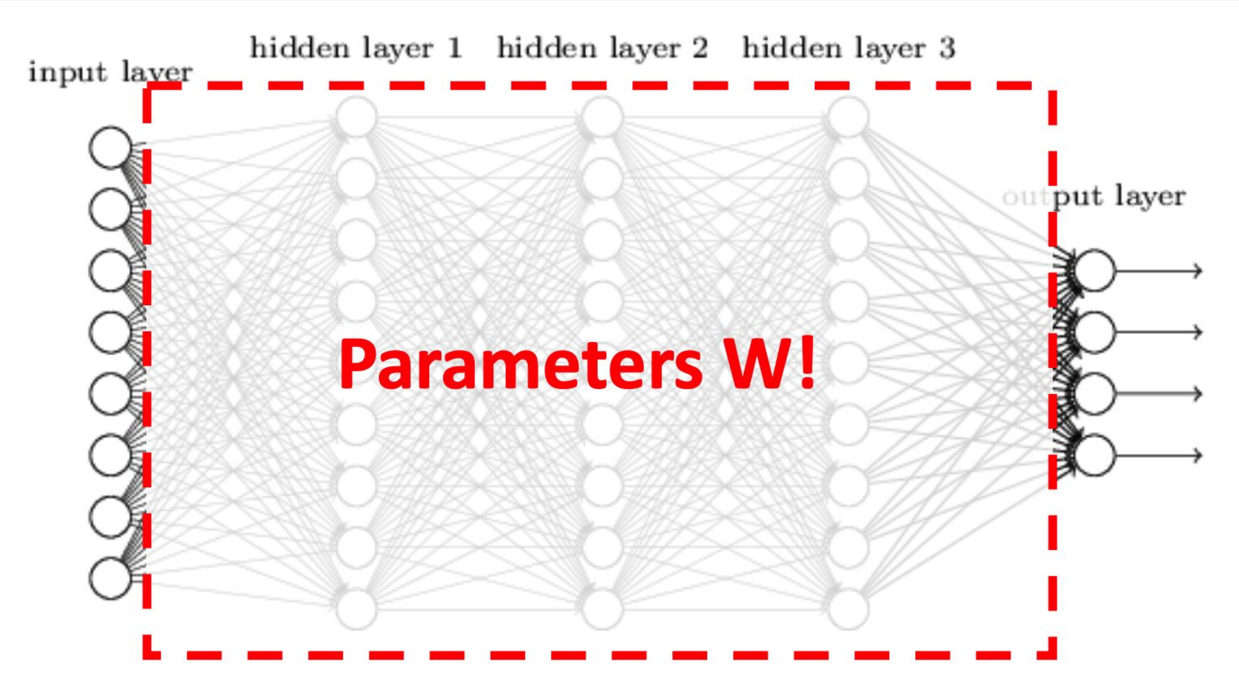
a larger pre-trained network or an ensemble of models

compact (smaller) network

ROBOTICS

Knowledge Distillation (KD)

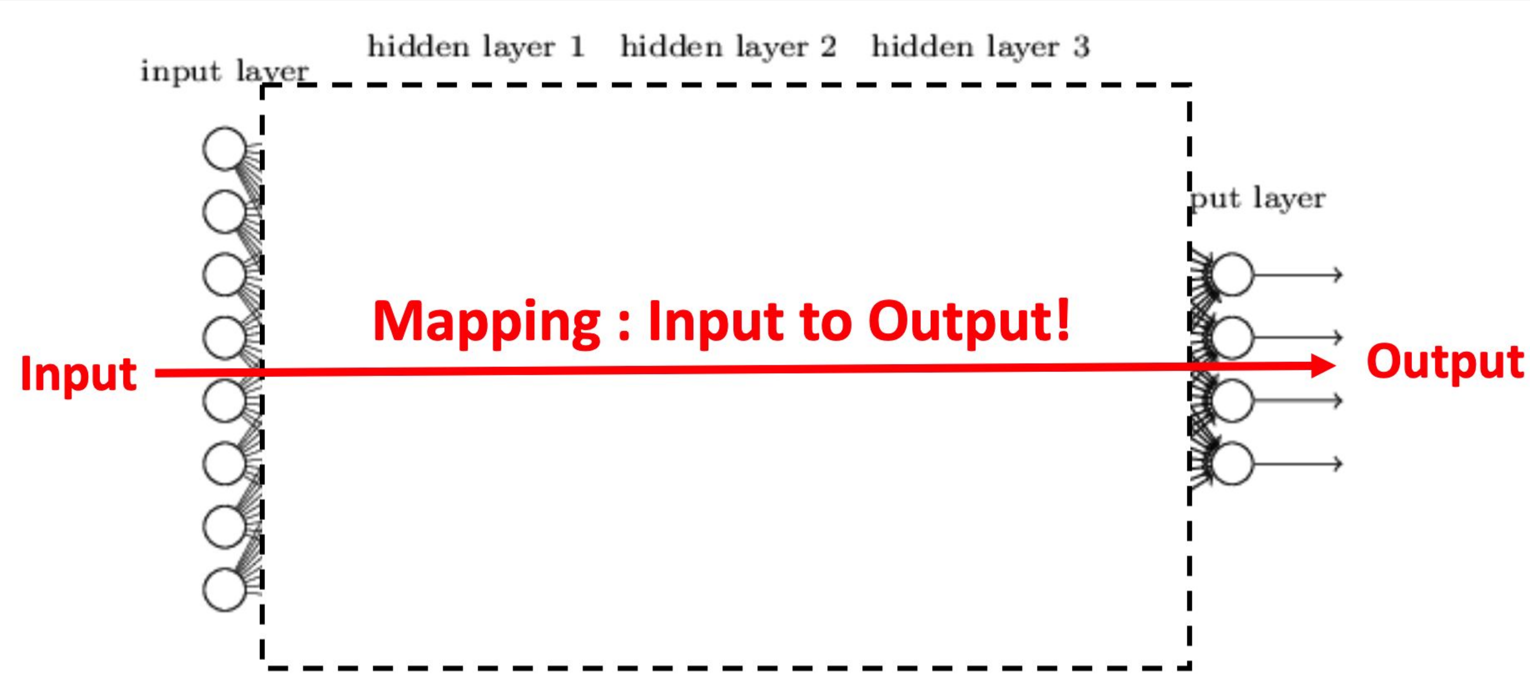
Q: What is "Knowledge"?



Knowledge Distillation (KD)

Q: What is "Knowledge"?

learned mapping from input vectors to output vectors



Knowledge Distillation (KD)

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

softmax

"temperature"

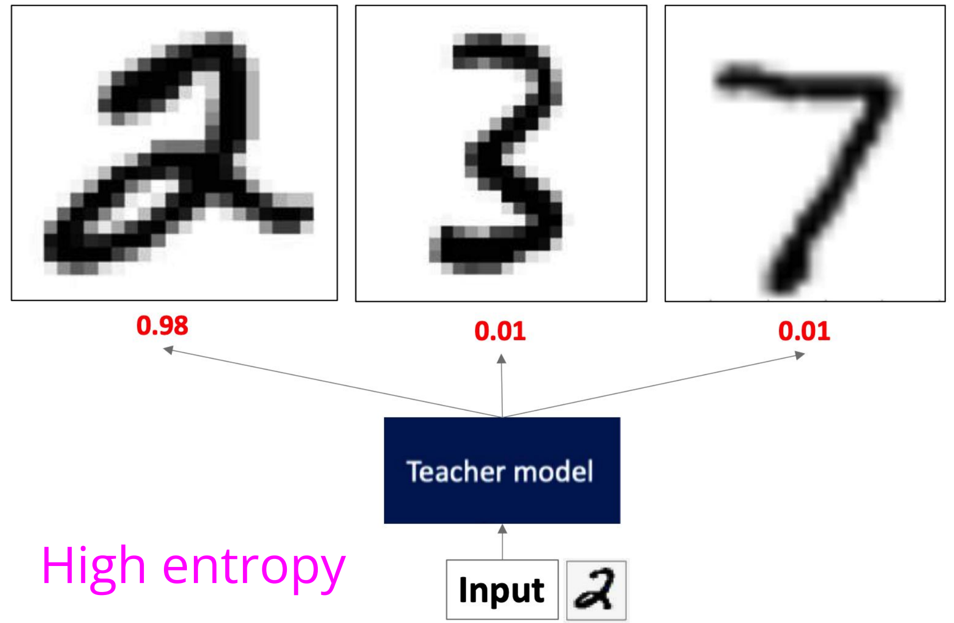
demo

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

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?



High entropy

Soft target

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

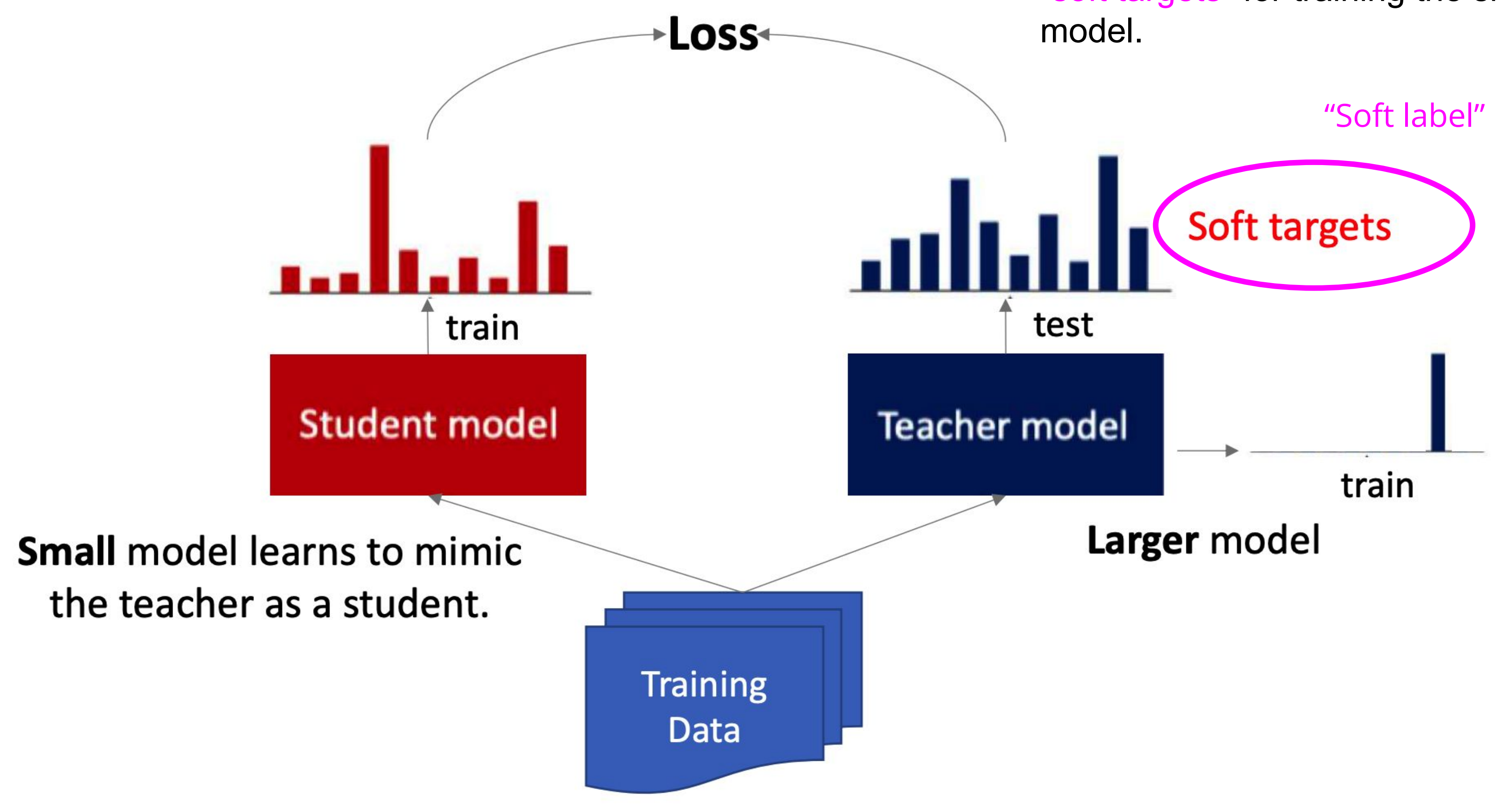
One-hot

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

ROBOTICS

Knowledge Distillation (KD)

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

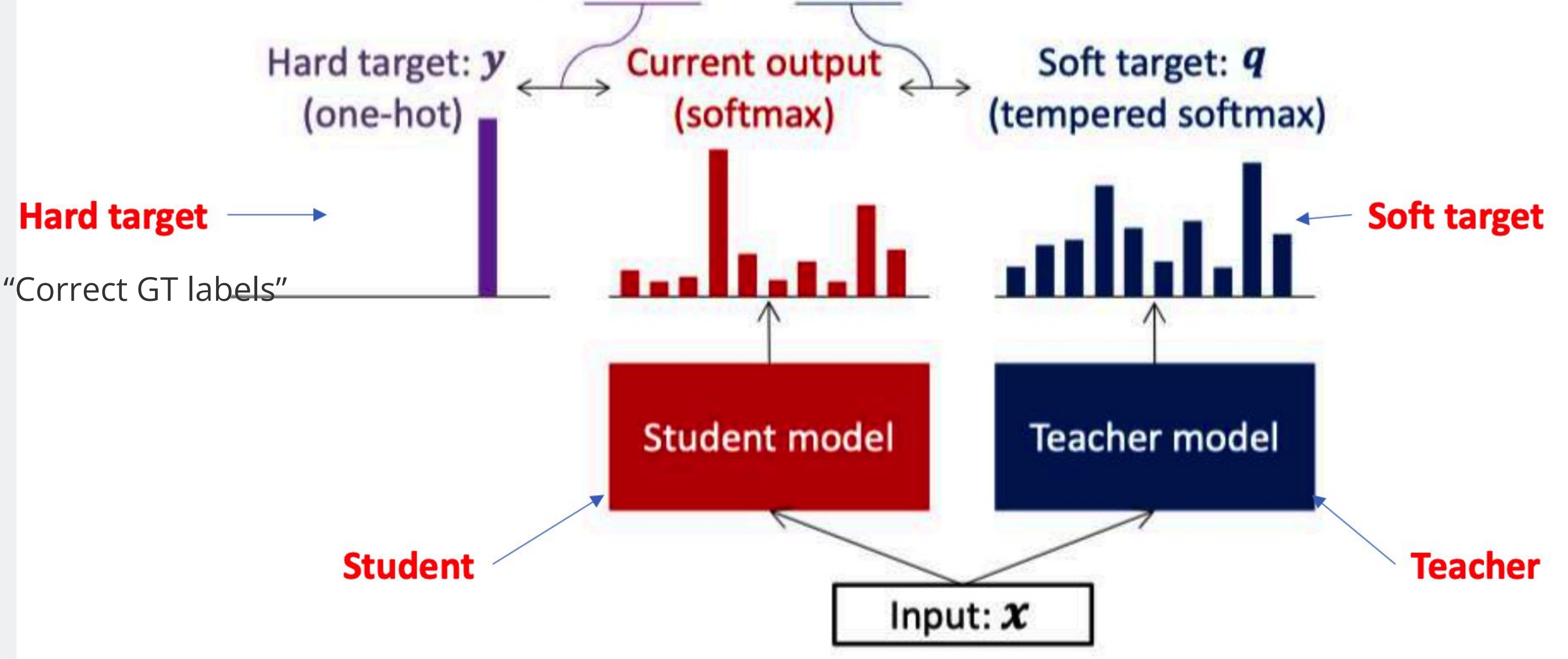


Knowledge Distillation (KD)

“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

$$L = (1-\rho)C_{hard} + \rho C_{soft}$$



ROBOTICS

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

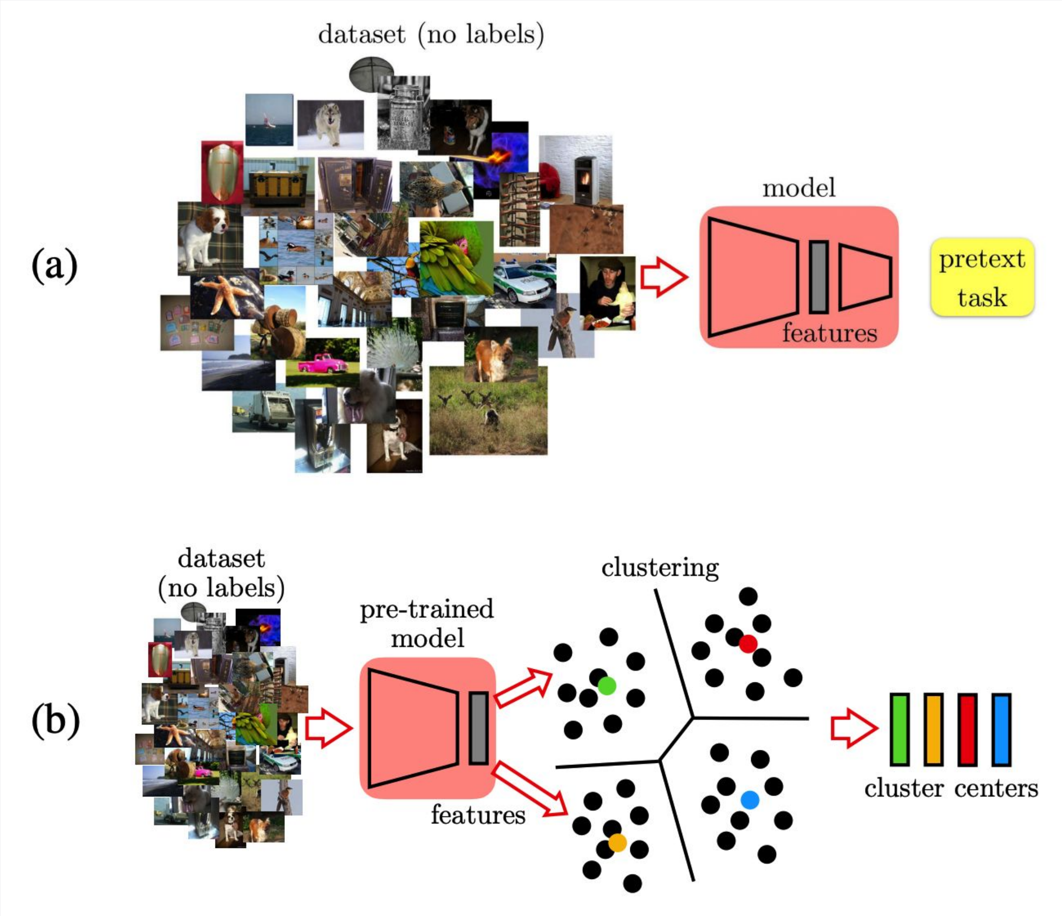
$$D_{\mathrm{KL}}(q \parallel p) = \sum_i q(i) \log \frac{q(i)}{p(i)} = H(q,p) - H(q)$$

Higher KL divergence ⇔ P&Q are more different

ROBOTICS

Self-Supervised Learning (SSL)

https://arxiv.org/pdf/1805.00385

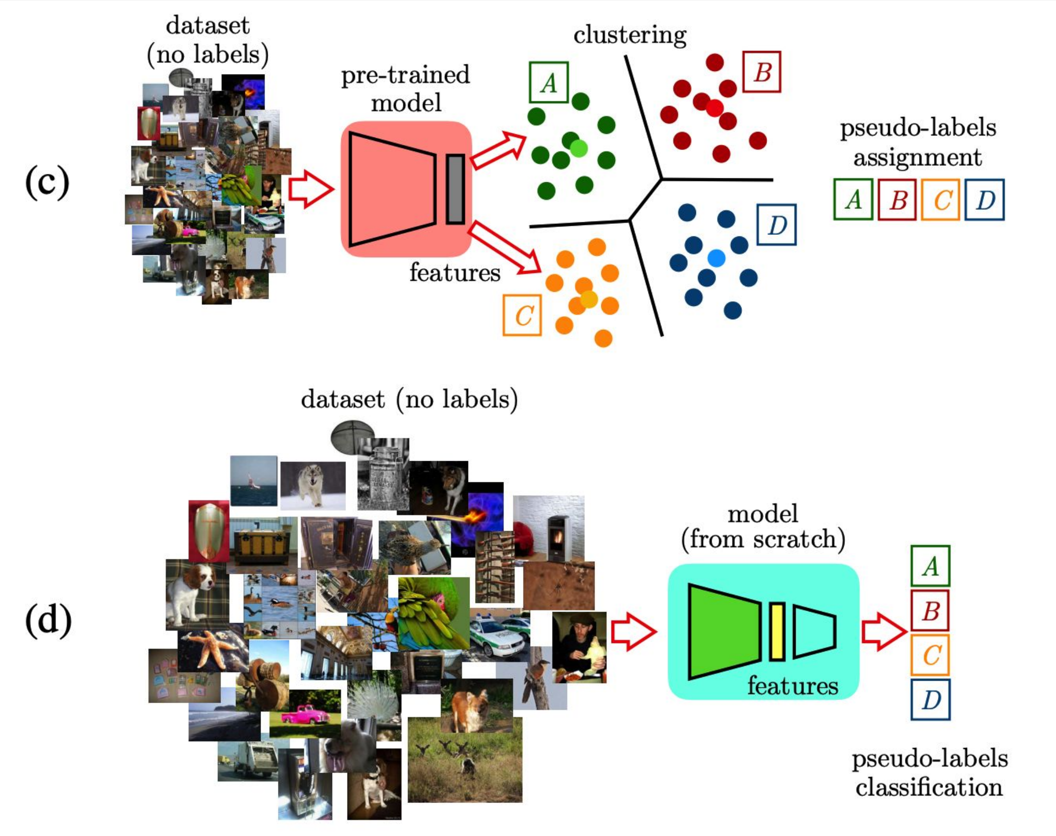


Learn with
no labels

(a) SSL Pre-Training
(b) Clustering

Self-Supervised Learning (SSL)

https://arxiv.org/pdf/1805.00385

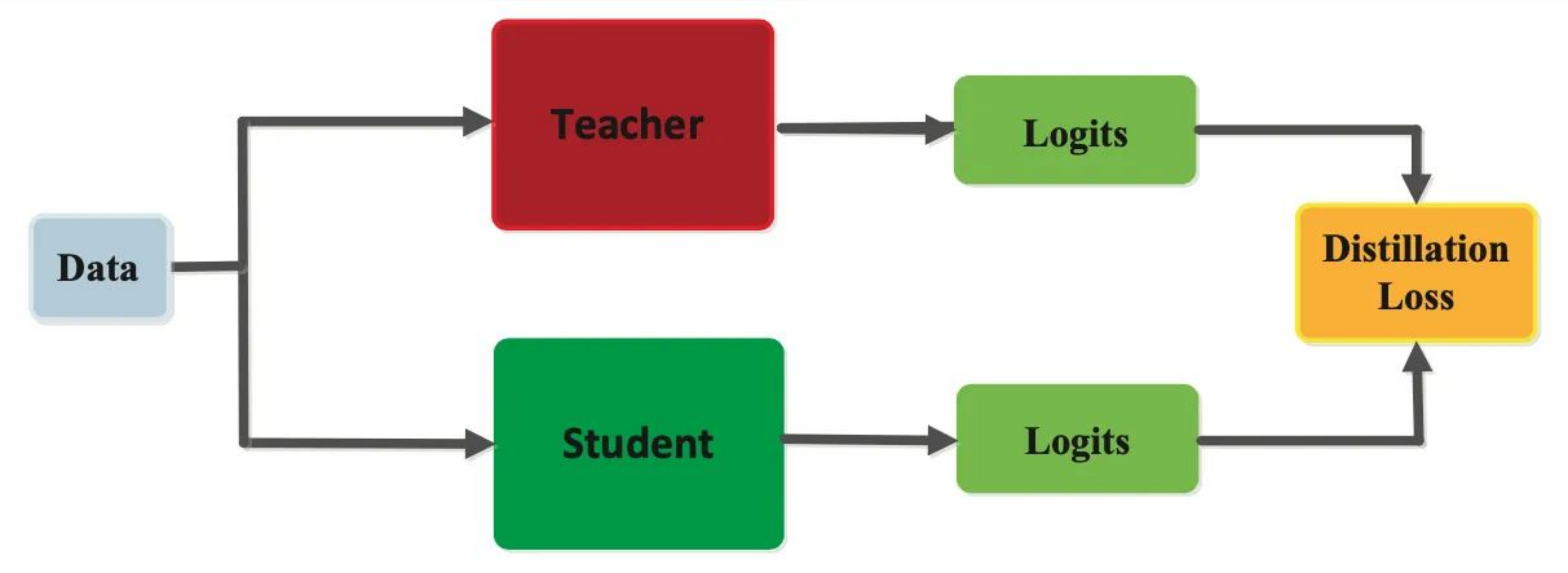


Learn with
no labels

(c) Pseudo-Labels
(cluster centers)

(d) train a classifier based on pseudo-labels

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

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

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

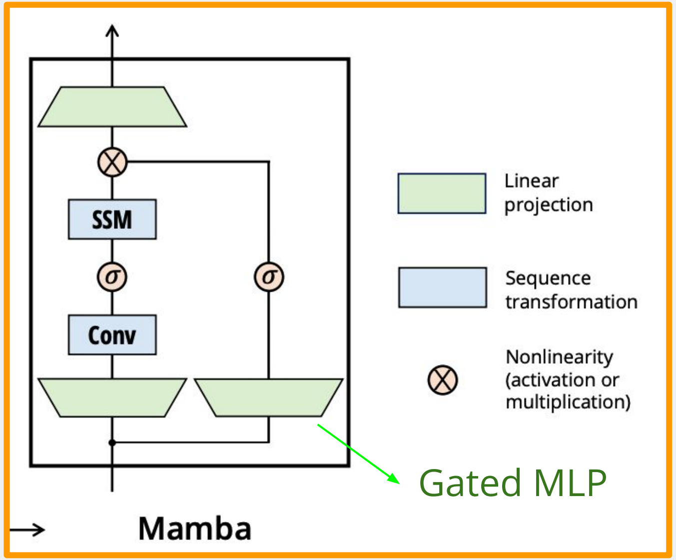
Evolution parameter

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

Projection parameter

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

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



Vision Mamba

https://arxiv.org/pdf/2401.09417

2D vision task

- spatial-aware
- bidirectional sequence modeling

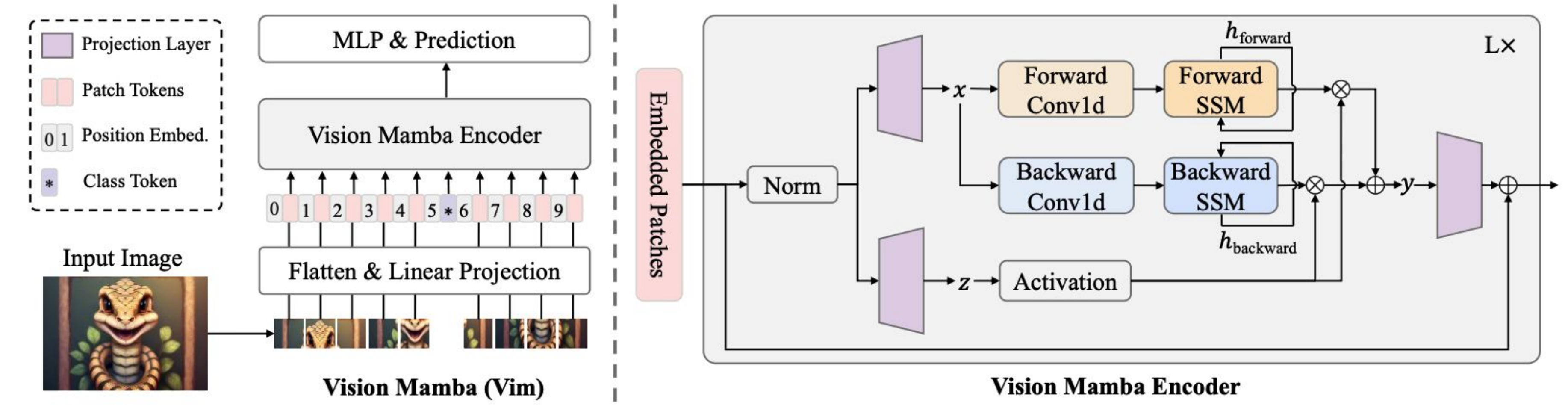


Figure 2: The overview of the proposed Vim model. We first split the input image into patches, and then project them into patch tokens. Last, we send the sequence of tokens to the proposed Vim encoder. To perform ImageNet classification, we concatenate an extra learnable classification token to the patch 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”

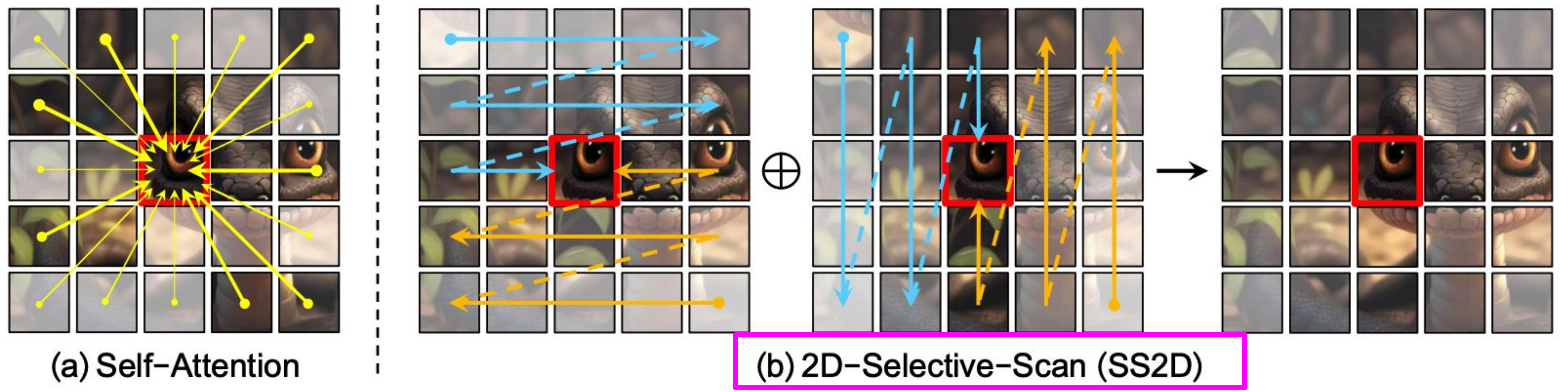


Figure 1: Comparison of the establishment of correlations between image patches through (a) self-attention and (b) the proposed 2D-Selective-Scan (SS2D). The red boxes indicate the query image patch, with its opacity representing the degree of information loss.

VMamba

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

SS2D

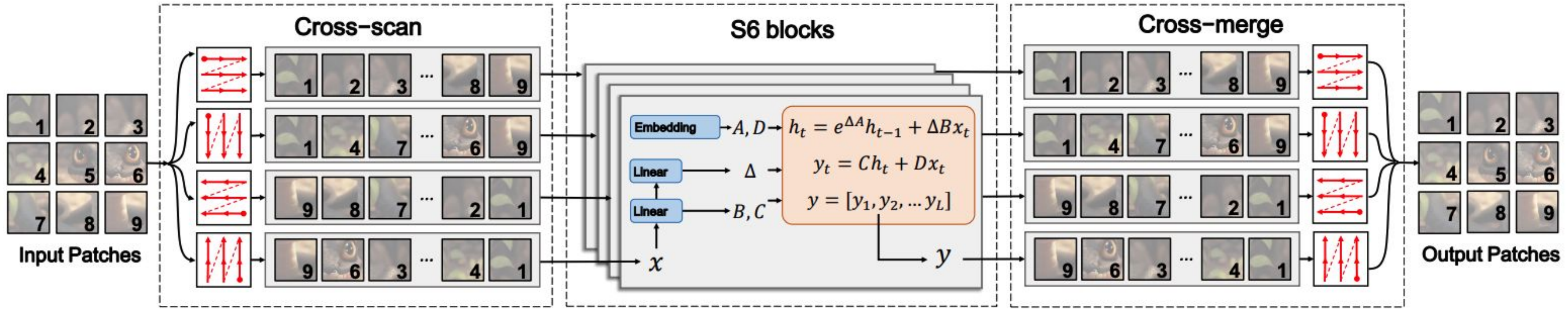
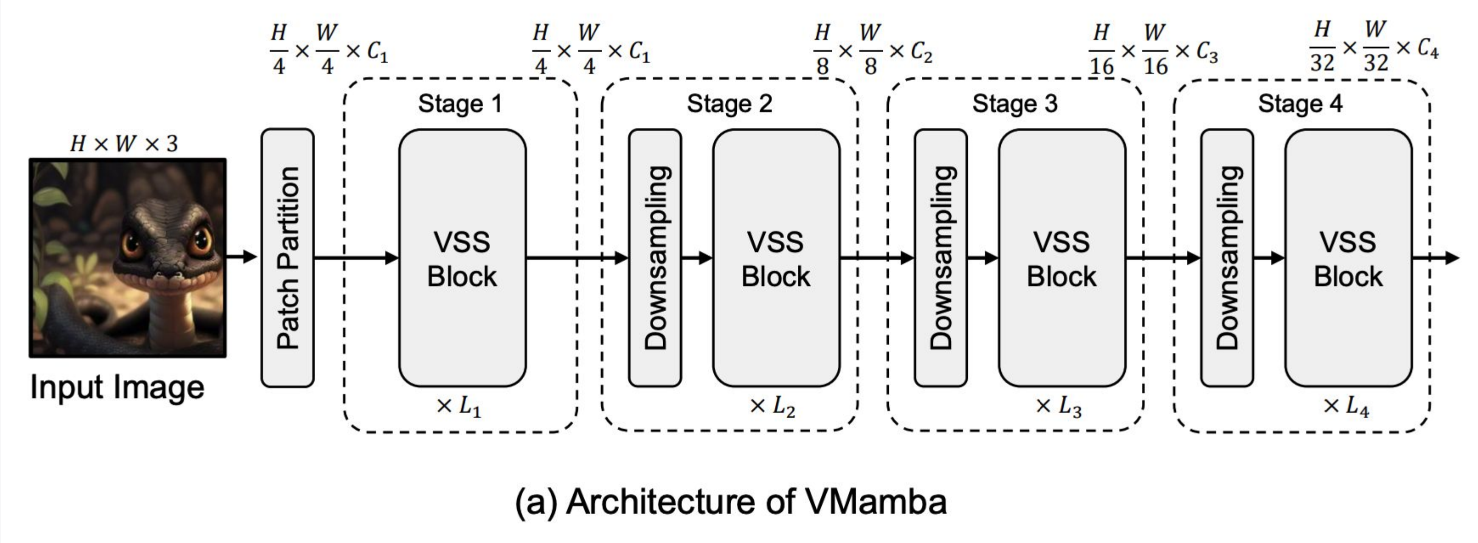


Figure 2: Illustration of 2D-Selective-Scan (SS2D). Input patches are traversed along four different scanning paths (*Cross-Scan*), with each sequence independently processed by separate S6 blocks. The results are then merged to construct a 2D feature map as the final output (*Cross-Merge*).

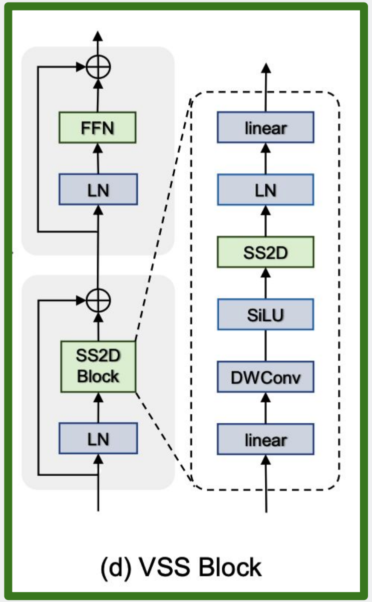
VMamba

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

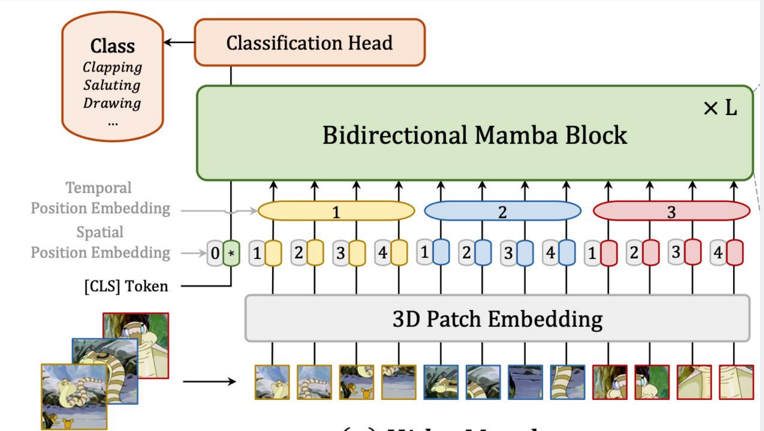


(d) VSS Block

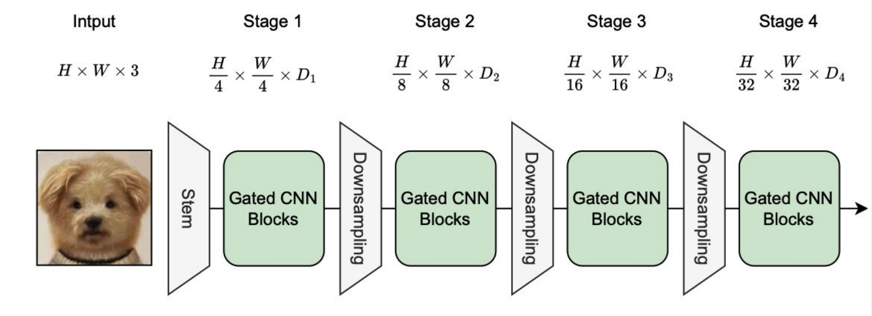
*Reminds of SwinTransformer?

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



(a) VideoMamba



(a) Overall framework of MambaOut

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)

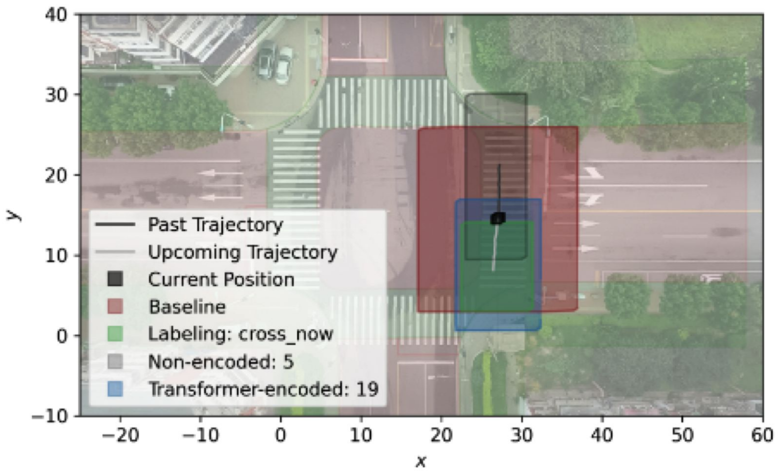
Sign	Gardiner code	Transliteration	Description
	G1	ꜣ	Egyptian vulture
	I9	f	Horned viper
	V24	wḏ	Cord wound on stick
	S12	nbw	Bead collar

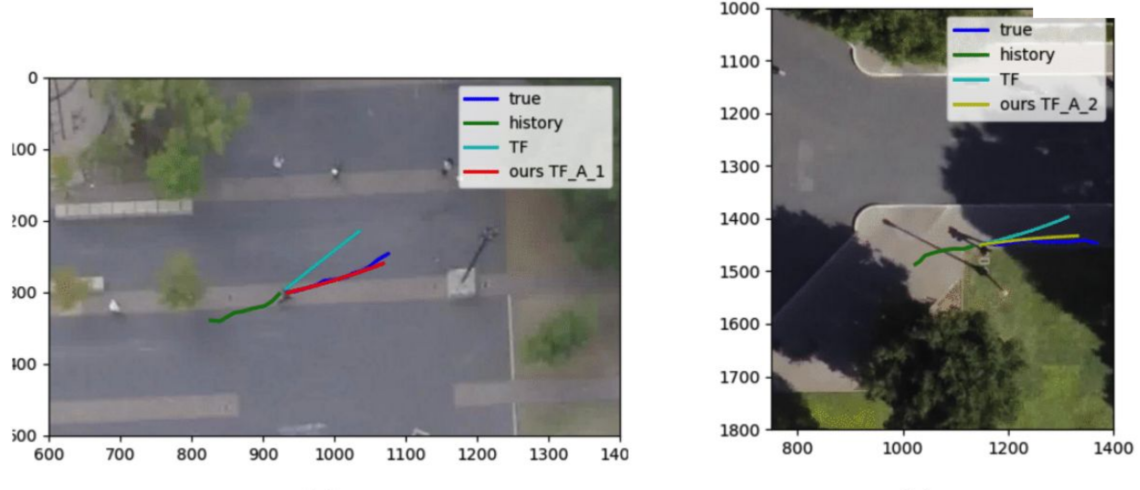
Table 1: Example of hieroglyphs and their Gardiner code, Transliteration and Description.

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

Motion/Trajectory Prediction

https://arxiv.org/pdf/2408.15250





(a) (b) (c)

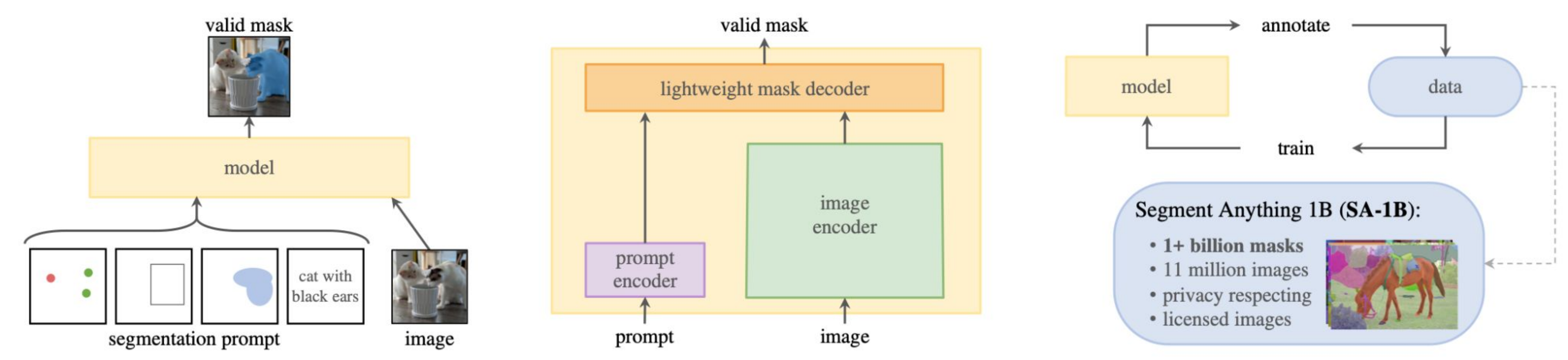
Du et al. Angle-Regulated Transformer Network for Pedestrian Trajectory Prediction [paper]

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)

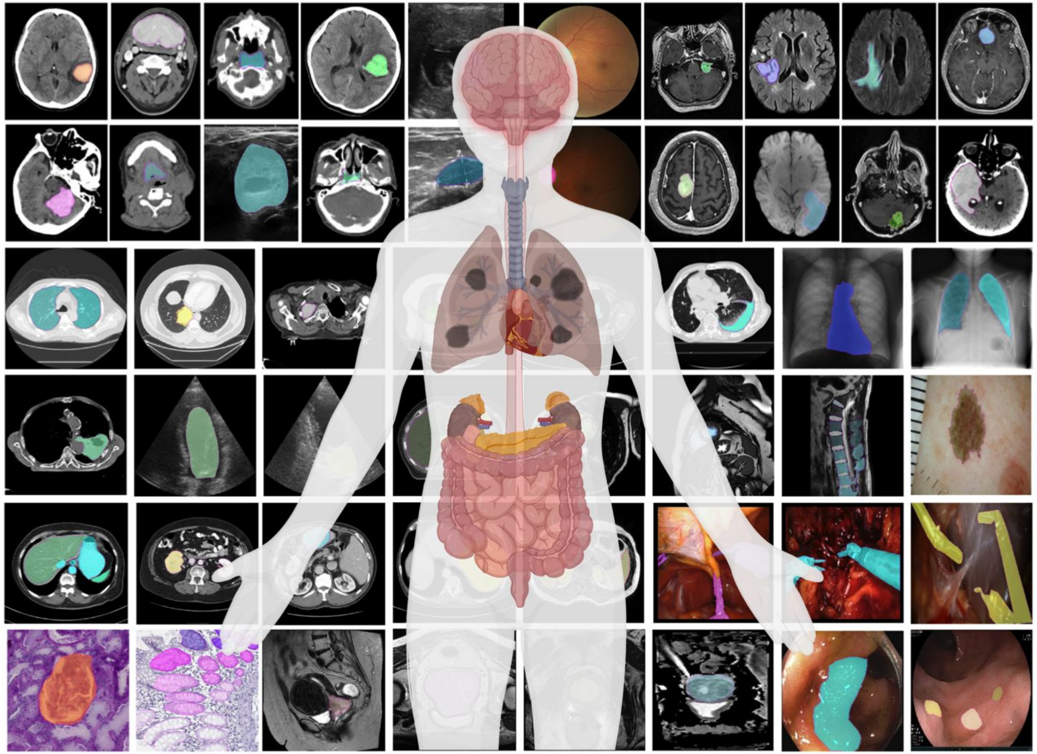
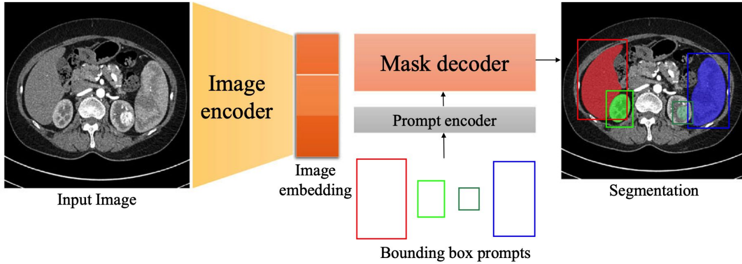


Fig. 1 | MedSAM is trained on a large-scale dataset that can handle diverse segmentation tasks. The dataset covers a variety of anatomical structures, pathological conditions, and medical imaging modalities. The magenta contours and mask overlays denote the expert annotations and MedSAM segmentation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



ROBOTICS

Graph Embodiment Transformer

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

Graph Embodiment Transformer

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

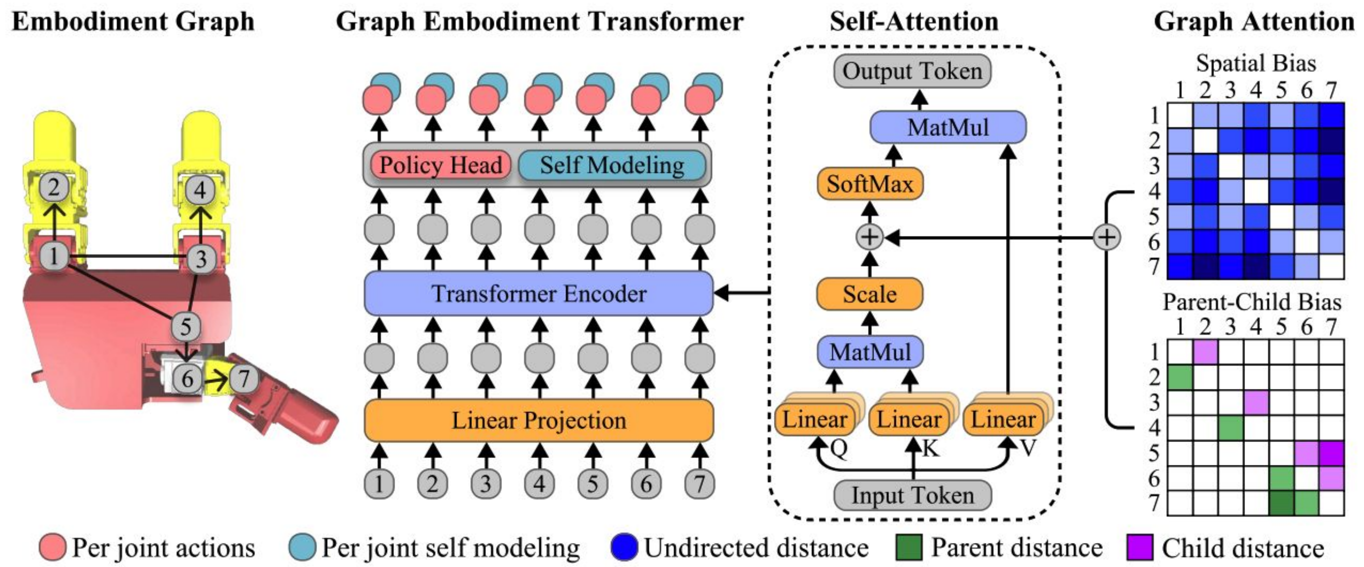
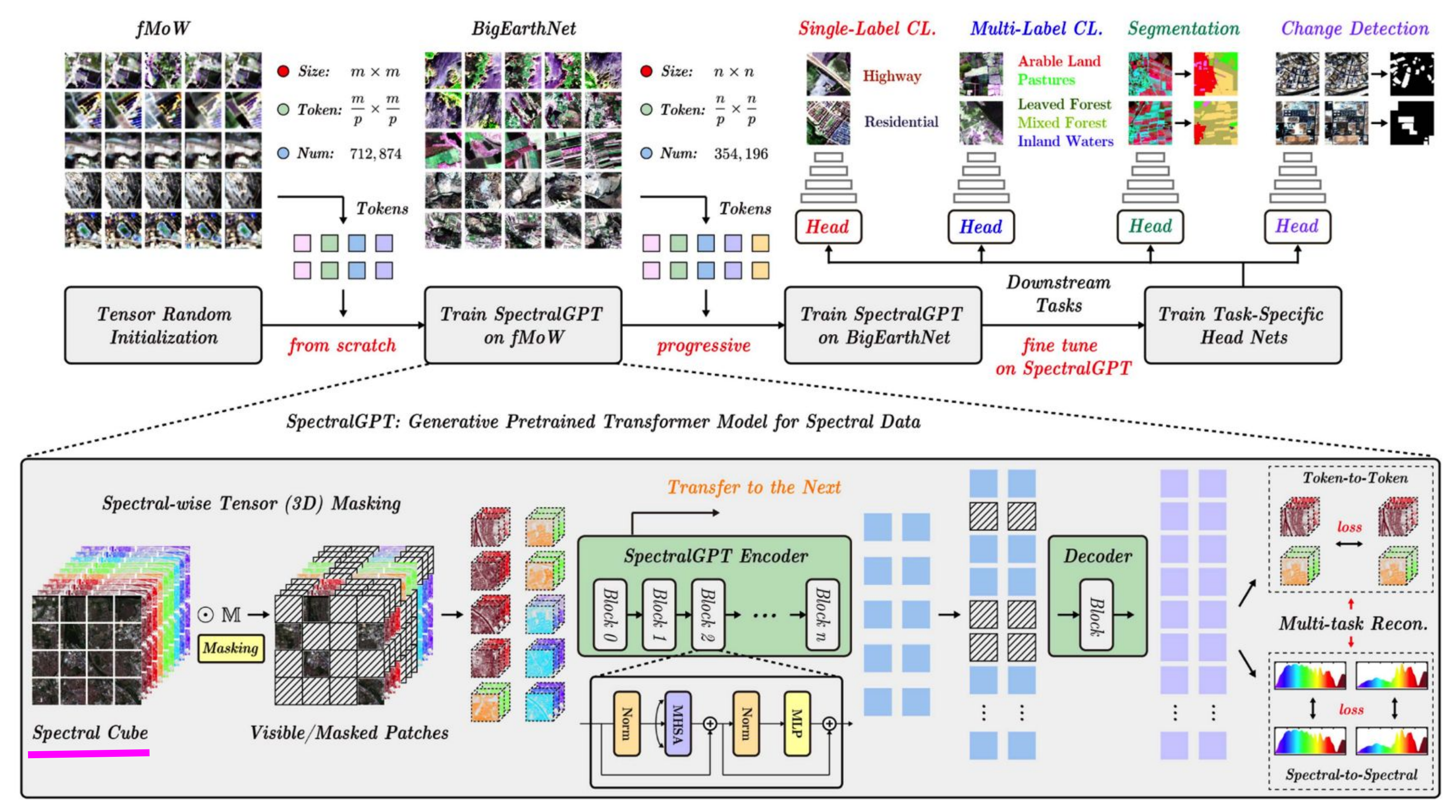


Fig. 2. **Graph Embodiment Transformer (GET).** GET is an embodiment-aware model based on a transformer encoder. Each joint forms separate tokens containing local sensory and embodiment information. The self-attention layers use an undirected (Spatial Bias) and directed (Parent-Child Bias) graph distance 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.

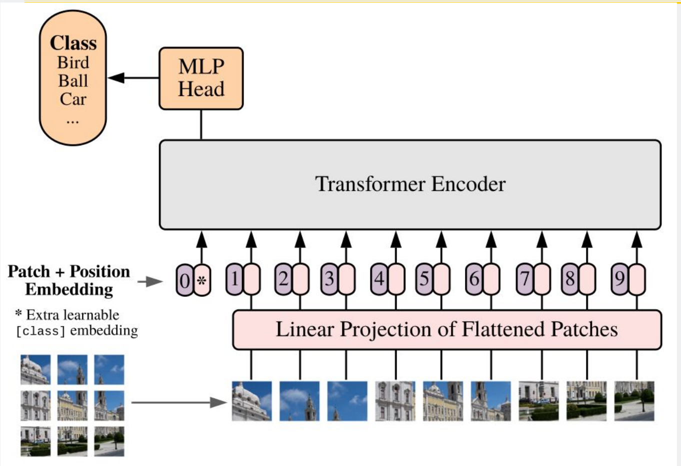
Spectral GPT

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



OTICS

Summary



- ViT
- DETR
- Swin Transformer
- DINO
- Mamba
- More...

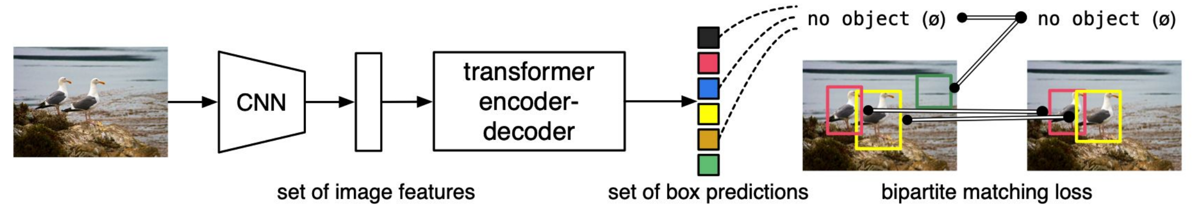


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "**no object**" (∅) class prediction.

Reminder:

- **P4** Due March 30, 2025
  (Lecture 13 on poseCNN, Lecture 17, 18 on attention and ViT)
- **Final Project** "lightning talk" April 1, 2025
- **Canvas Quiz** on transformers has been released (Due April 2, 2025)

M | ROBOTICS