Week 5

Transformer in Vision

[ECEA0649/ECE40049] Deep Learning for Image Processing | Spring 2026

Kihyun Na (Research Professor)

BK21 AI Project Group & Institute for Information and Communication Technology,
Handong Global University

HANDONG GLOBAL UNIVERSITY

Curriculum

** Adjusted based on survey results*

Wk 1	OT + Introduction	**Wk 9**	Foundation Models (CLIP, SAM) + Paper #1
Wk 2	DL Fundamentals Review	**Wk 10**	Diffusion Models + Paper #2
Wk 3	CNN	**Wk 11**	Conditional Generation + Paper #3
Wk 4	From Sequence Modeling to Transformer	**Wk 12**	Vision-Language Models + Paper #4
Wk 5	Transformer in Vision (Today)	**Wk 13**	VLM Applications + Paper #5
Wk 6	Detection + Segmentation	**Wk 14**	Video Understanding + Paper #6
Wk 7	Self-Supervised Learning	**Wk 15**	Embodied AI & Robot Vision + Paper #7
Wk 8	Review Literacy + Role Explanation	**Wk 16**	Miniconference (Final Project)

Lecture Lecture + Paper Miniconference

Today's Agenda

01

Vision Transformer (ViT)

Patch embedding, [CLS], position embedding, model variants, attention visualization

30 min

02

Improving ViT & Modern Transformer Design

DeiT, Swin, MAE — Overcoming limitations of ViT | How Transformer works nowadays

20 min

03

CNN vs ViT

Inductive bias, scaling, CNN vs ViT timeline, hybrid models, where we stand in 2026

15 min

04

Vision Encoders in the Wild

CLIP, SigLIP 2, DINOv2, SAM, VLMs — same ViT, different training, different superpowers

25 min

➡ **Note:** *Preview of post-Wk7 topics; covered in detail in later weeks*

Part 1

Vision Transformer (ViT)

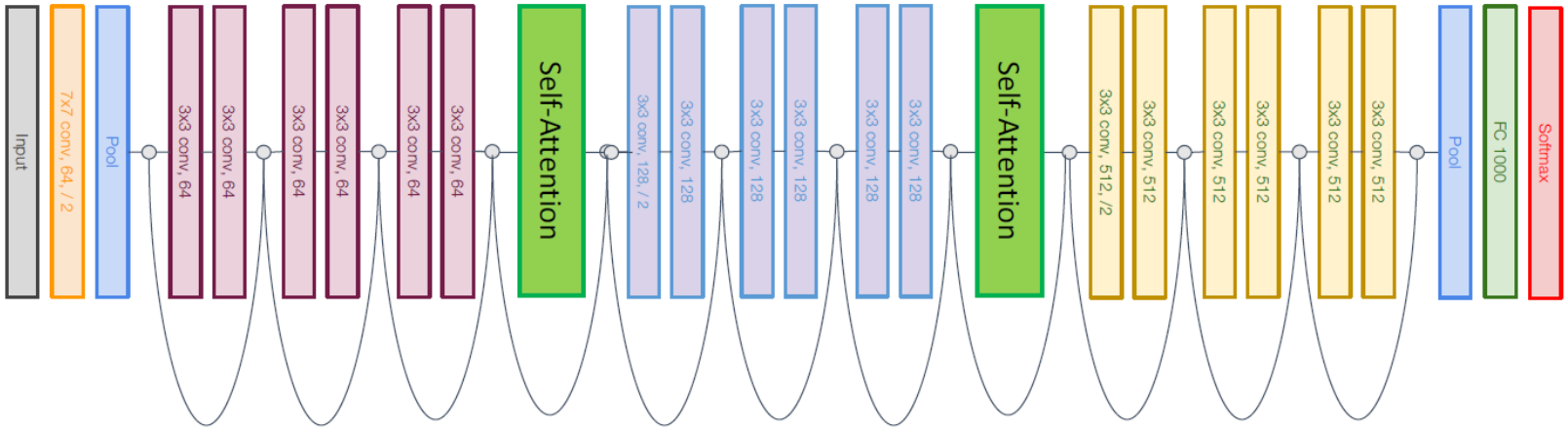
Transformer + Image Patches = Vision Transformer

How to use Transformers for Vision?

IDEA 1 Add attention to existing CNNs

Start from standard CNN architecture (e.g. ResNet)

Add Self-Attention blocks between existing ResNet blocks

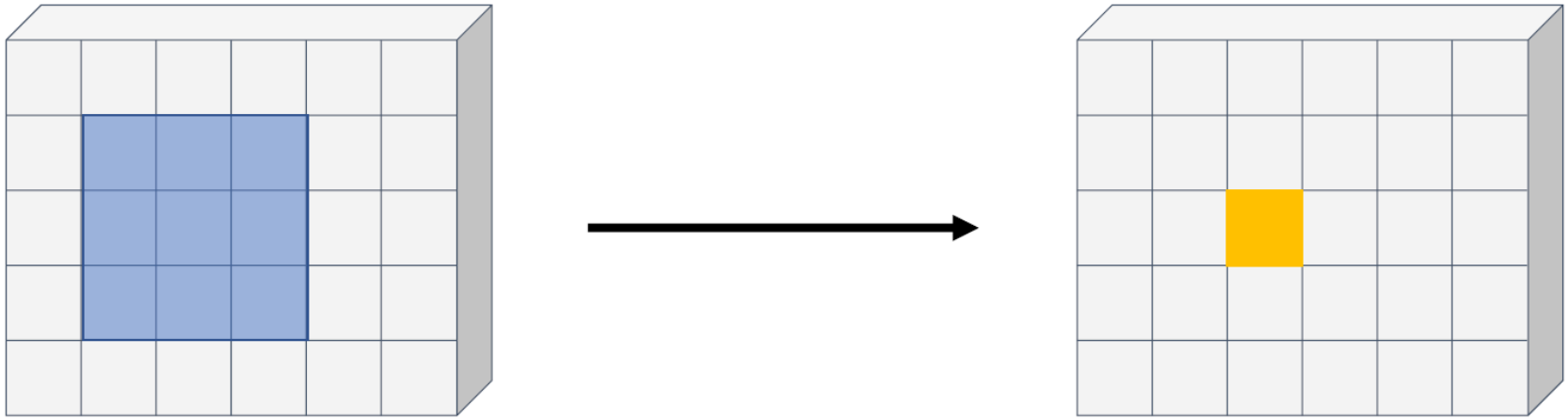


Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018
Wang et al, "Non-local Neural Networks", CVPR 2018

How to use Transformers for Vision?

IDEA 2 Replace Convolution with Local (Windowed) Attention

Convolution: Output at each position is inner product of conv kernel with receptive field in input



Input: C x H x W

Output: C' x H x W

Hu et al, "Local Relation Networks for Image Recognition", ICCV 2019; Ramachandran et al, "Stand-Alone Self-Attention in Vision Models", NeurIPS 2019

How to use Transformers for Vision?

IDEA 2 Replace Convolution with Local Attention

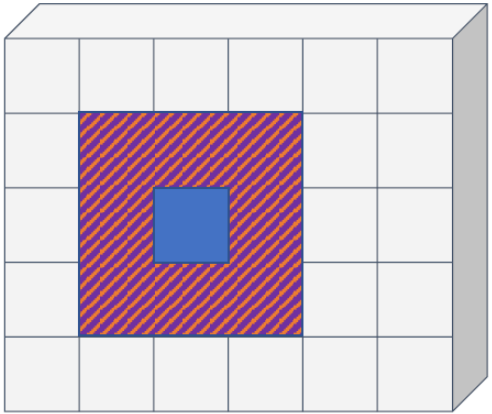
Map center of receptive field to **query**

Map each element in receptive field to **key** and **value**

Compute **output** using attention

Replace all conv in ResNet with local attention

Lots of tricky details, hard to implement, only marginally better than ResNets



Input: C x H x W

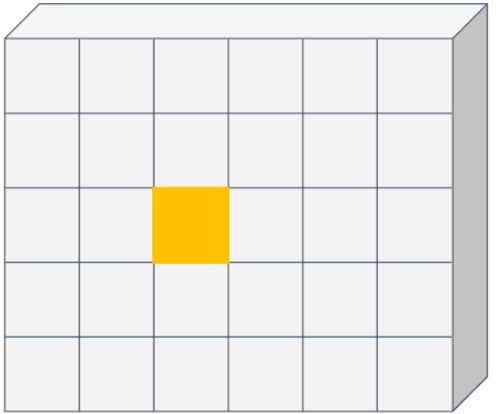
Query: D_Q

Keys: R x R x D_Q

Values: R x R x C'

Attention

Output: C



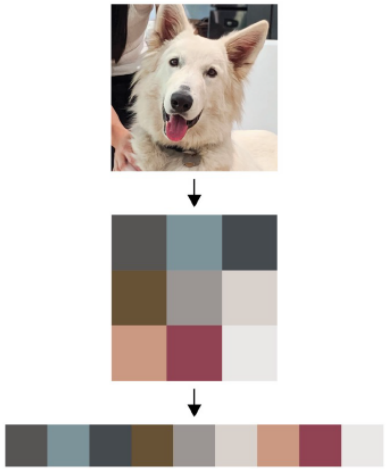
Output: C' x H x W

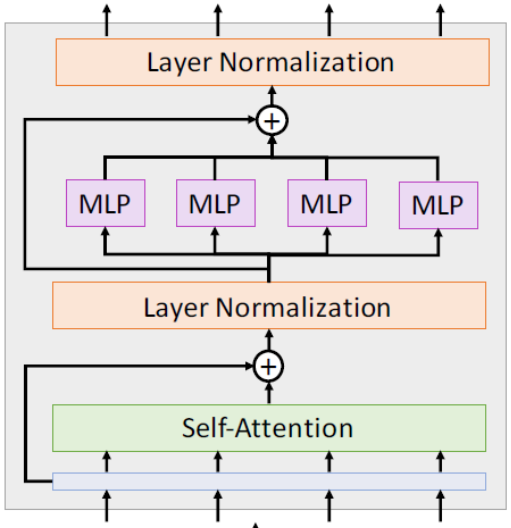
Hu et al, "Local Relation Networks for Image Recognition", ICCV 2019; Ramachandran et al, "Stand-Alone Self-Attention in Vision Models", NeurIPS 2019

How to use Transformers for Vision?

IDEA 3 Standard Transformers on Pixels

Treat an image as a set of pixel values





Feed as input to standard Transformer

Problem: Memory use!

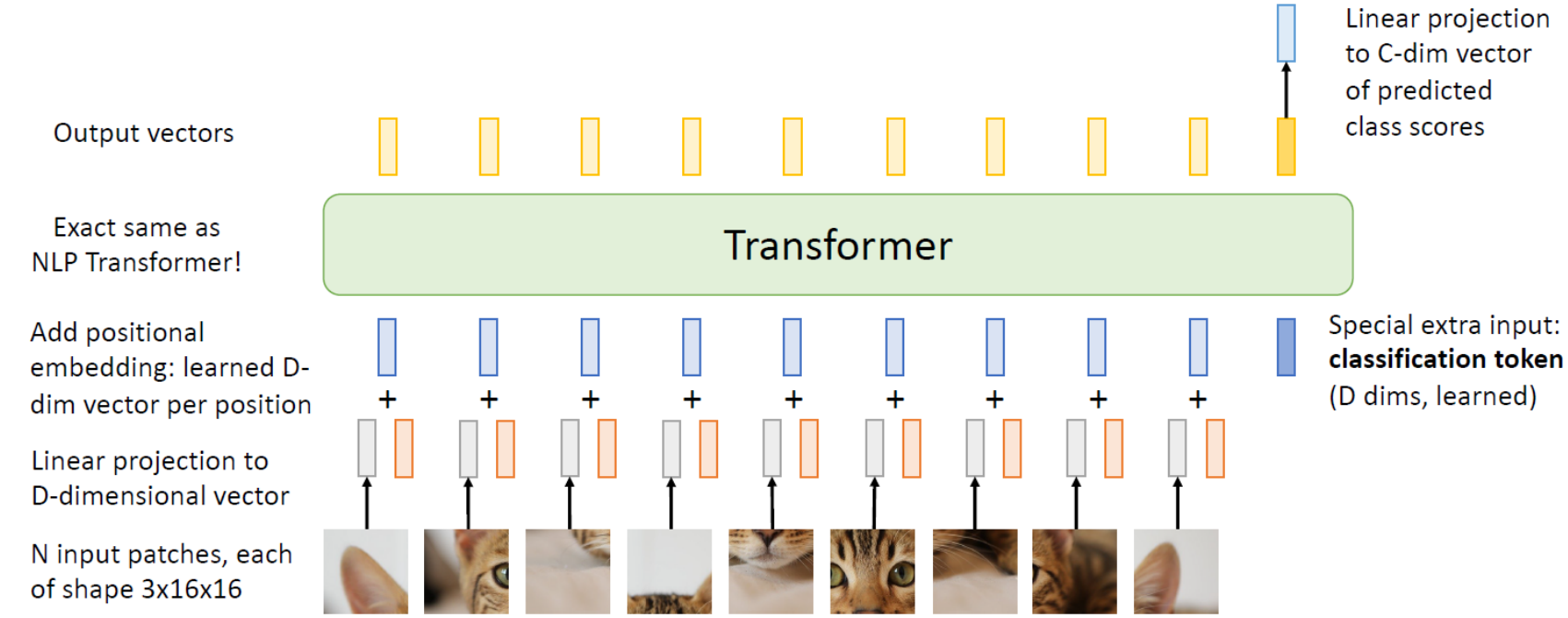
R x R image needs R^4 elements per attention matrix

R=128, 48 layers, 16 heads per layer takes 768GB of memory for attention matrices for a single example…

Chen et al, “Generative Pretraining from Pixels”, ICML 2020

How to use Transformers for Vision?

IDEA 4 Standard Transformers on Patches = ViT



Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Patch Embedding

The only vision-specific component. Everything else is standard Transformer.

How It Works

224×224×3 image
→ 14×14 grid of 16×16 patches
→ 196 patches total
→ Each patch: flatten to 768 dims
→ Linear(768, D) → D-dim token

196 tokens fed to Transformer.

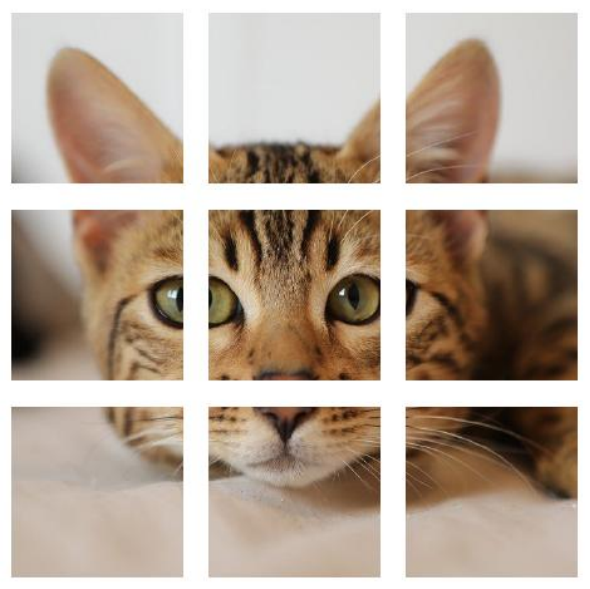
(c.f. Common sequence length in NLP is 128-512 tokens)

Equivalent to Conv2D

Flatten + Linear = Conv2D with kernel_size=16, stride=16

In PyTorch:
nn.Conv2d(3, D, kernel_size=16,
stride=16)

Same operation, different framing.
Wk3 CNN knowledge still applies here.



Wk4 recall "The Transformer doesn't care what the tokens are. Only the tokenization changes between domains." — here, tokens = patches.

Position Embedding in ViT

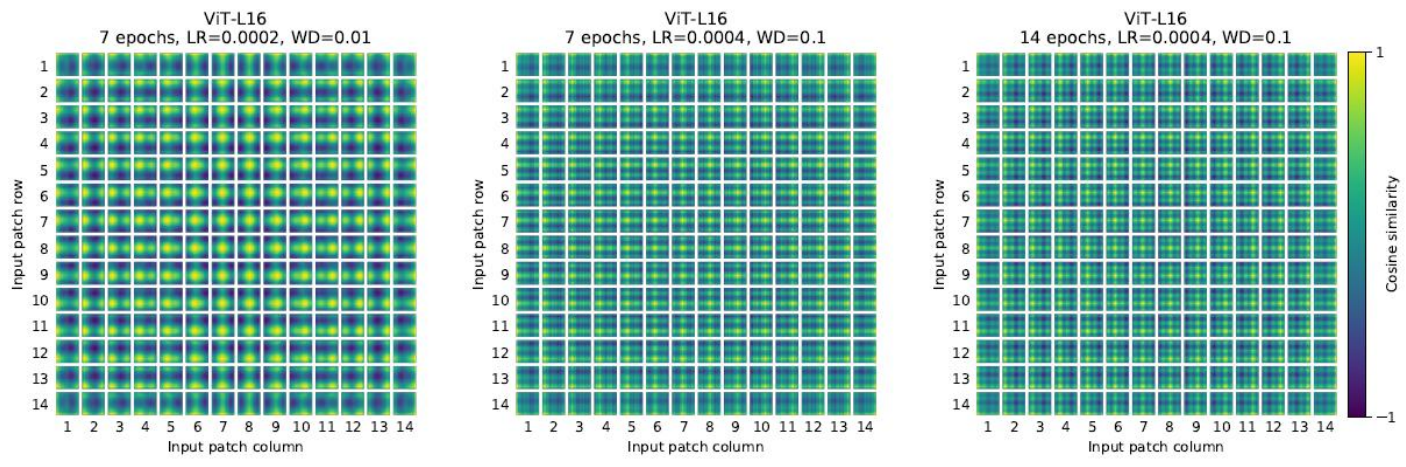
Wk4 recap: Self-attention is permutation-invariant. ViT uses learned 1D position embeddings.

Learned, not sinusoidal

- Each of 196+1 positions gets a learned D-dim embedding vector
- No 2D structure encoded explicitly — but the model learns spatial relations anyway
- Fine-tune at different resolution? → Interpolate PE to new grid size

Why it matters

- Without PE, self-attention treats all patches as an unordered set
- Cosine similarity of learned PE reveals 2D grid structure (see heatmap below)
- Nearby patches get similar embeddings → spatial locality emerges from data



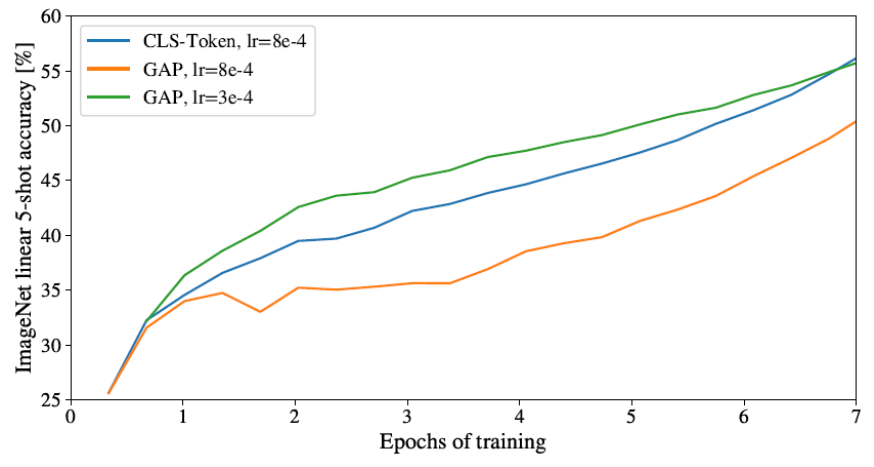
[CLS] Token & Classification

[CLS] Token (ViT default)

- Learnable D-dim vector prepended to the patch sequence
- Attends to all 196 patch tokens through self-attention layers
- Final output = global representation of the entire image
- MLP head on [CLS] output → class scores
- Borrowed from BERT — same idea

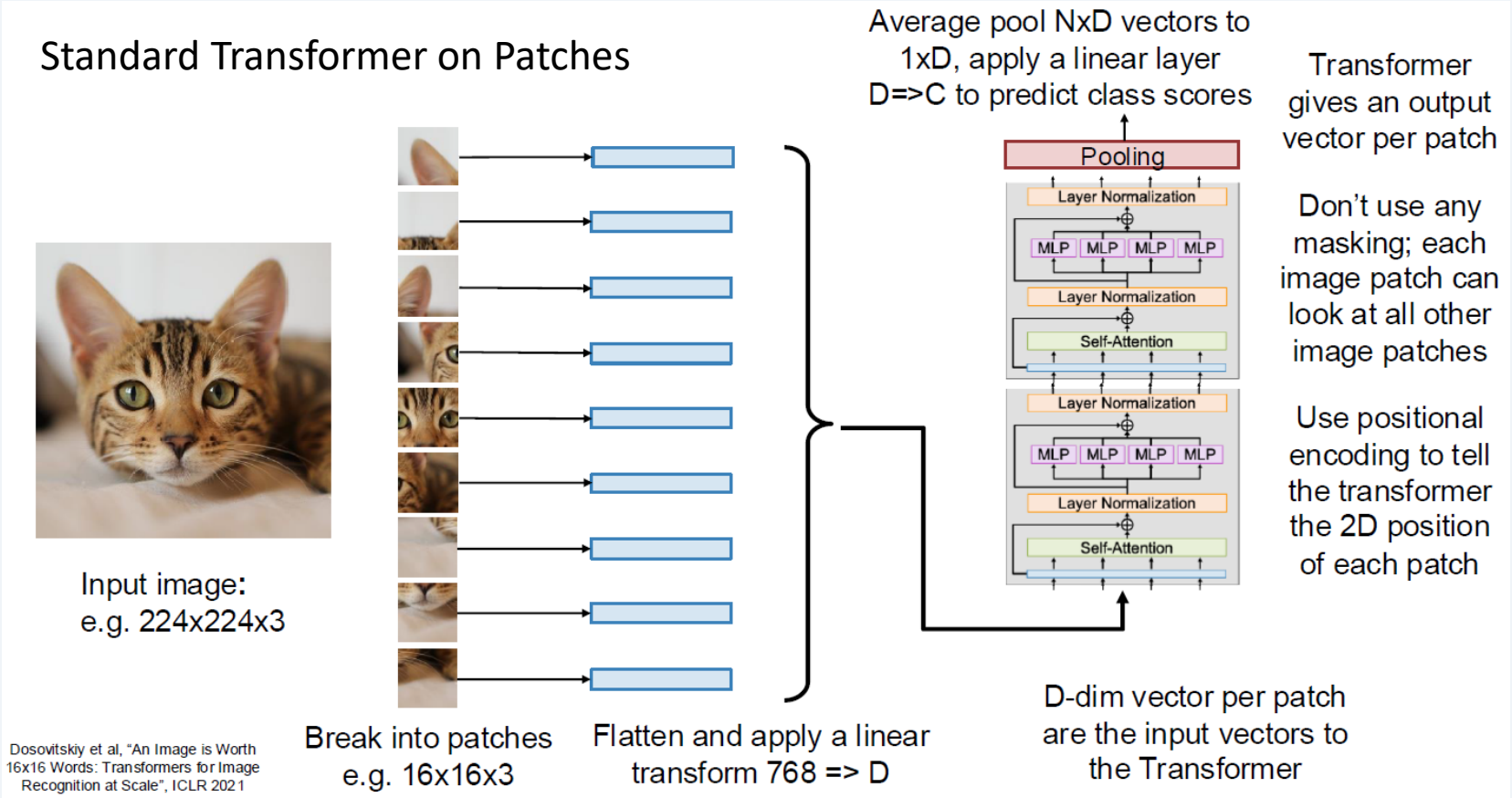
GAP Alternative (Wk3 recall)

- Global Average Pool over all 196 patch token outputs
- No extra [CLS] token needed
- Performance is similar in practice
- Used in some ViT variants (DeiT, BEiT v2)
- Same idea as GAP in CNN architectures (Wk3 Pooling)



ViT using different classifier

Dosovitskiy et al. (2020) — Apply the standard Transformer directly to image patches.



ViT Training: Data is Everything

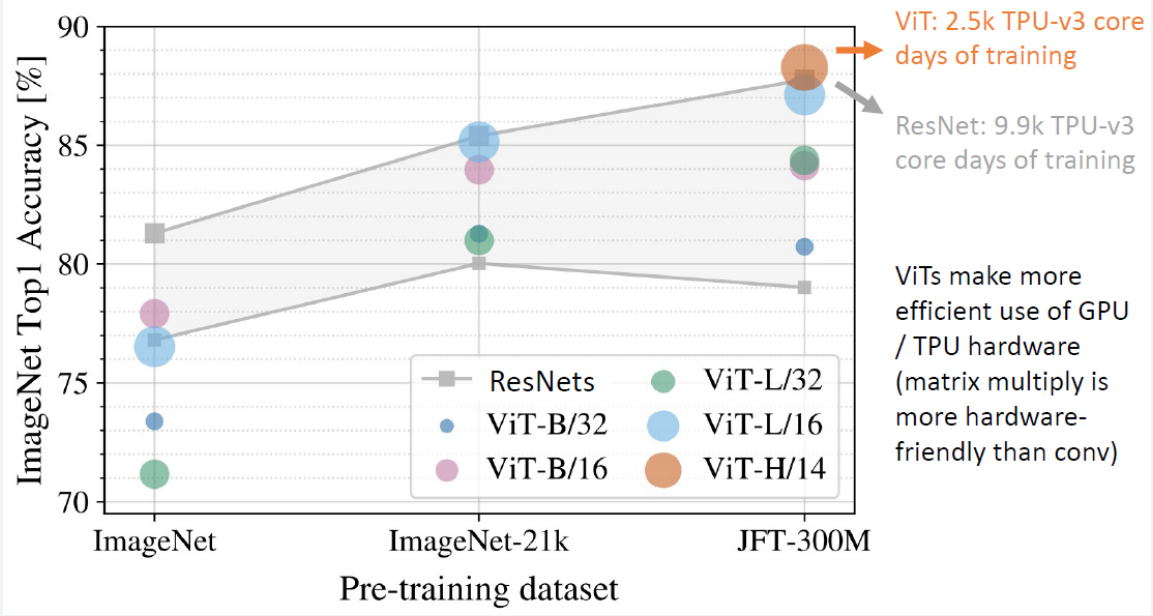
Original ViT Recipe

Pretrain on JFT-300M (Google internal, 300M images, 18K classes).

Fine-tune on ImageNet-1K.

Without JFT: ViT < ResNet.
With JFT: ViT > ResNet.

"ViT needs massive data to overcome lack of inductive bias."



This data dependency motivated DeiT (training recipe), MAE (self-supervised), and CLIP (web-scale text pairs). → Part 2 & 4.

ViT Model Family

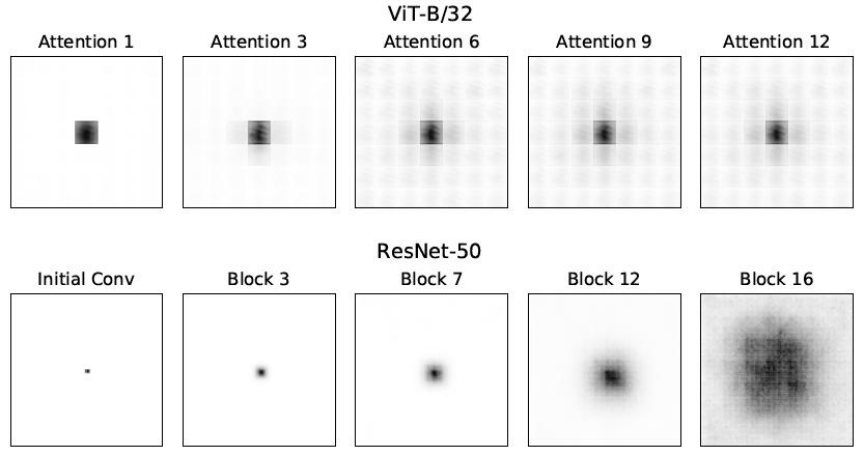
How to read ViT-L/14: Large model, patch size 14. Smaller patch = more tokens = better but slower.

Model	Layers	Hidden dim	Heads	Params	Patch	Tokens (224^2)
ViT-S	12	384	6	22M	16	196
ViT-B	12	768	12	86M	16	196
ViT-L	24	1024	16	307M	16	196
ViT-H	32	1280	16	632M	14	256

/16 vs /14 Patch size. /14 = smaller patches = more tokens (256 vs 196) = finer detail but more compute.
In Part 4, we'll see: CLIP uses ViT-L/14, SAM uses ViT-H/14. Now you can decode these names.

What Does ViT Attention See?

Different heads learn different patterns. Some local, some global — from the first layer.



CNN (Wk3)

Receptive field grows layer by layer.
Visualization: GradCAM (post-hoc).
Separate computation needed.

ViT

Global attention from layer 1.
Visualization: attention weights built-in.
DINO: unsupervised segmentation emerges.

Ref: Raghu et al. (2021) "Do Vision Transformers See Like CNNs?" — ViT uses global+local from early layers., Caron et al. (2021)

Part 2

Improving ViT

Overcoming Limitation of ViT

ViT's Limitations → Why Variants?

Data Hungry	Single Scale	$O(N^2)$ Cost	Training Instability
Needs JFT-300M to beat CNN. Most researchers don't have 300M labeled images.	All tokens at same resolution. No multi-scale feature maps. Can't do detection/segmentation directly.	Attention is quadratic in token count. High-res or video = prohibitive.	Sensitive to optimizer, hyperparams, schedule. Harder to train than CNN.
→ DeiT: training recipe → MAE: self-supervised	→ Swin: hierarchical + window attention	→ Swin: O(N) windowed → Wk4 Attention Zoo	→ Conv stem (Xiao 2021) → DeiT training recipe

Improving ViT: Augmentation and Regularization

Regularization for ViT models:
- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

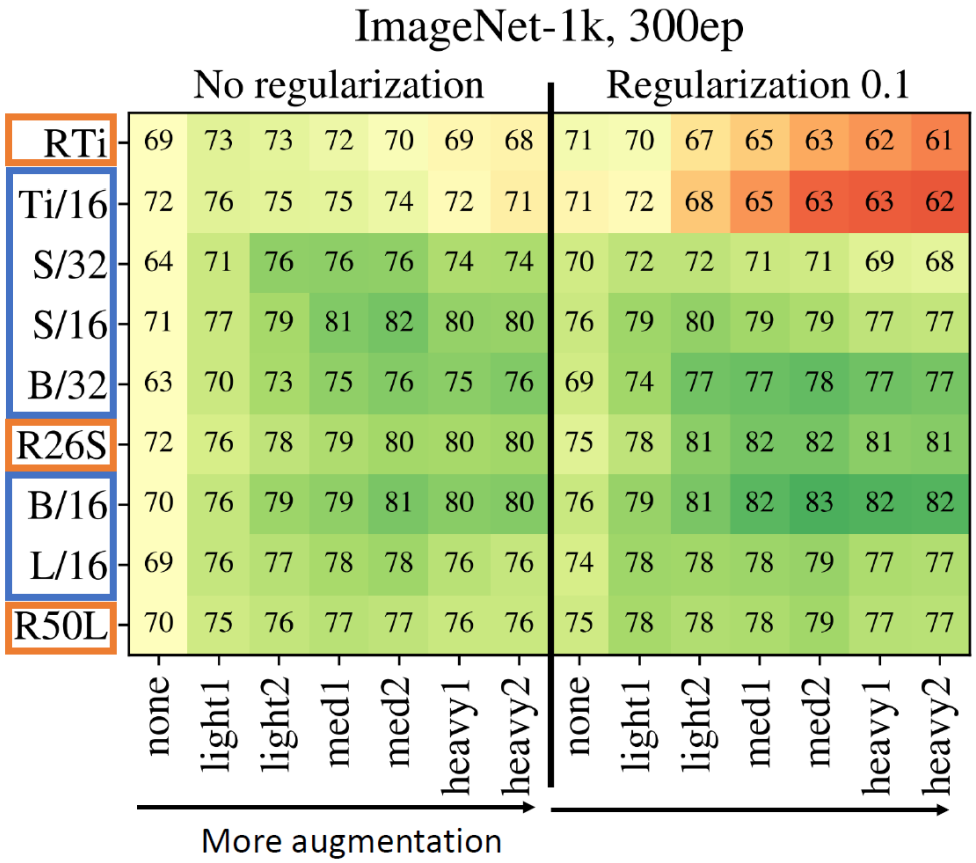
Data Augmentation for ViT models:
- MixUp
- RandAugment

Hybrid models: ResNet blocks, then ViT blocks

ViT models:
Ti = Tiny
S = Small
B = Base
L = Large

Original Paper:
77.9
76.53

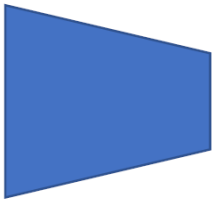
Lots of other patterns in full results



Steiner et al, “How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers”, arXiv 2021

Improving ViT: Distillation

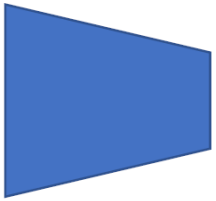
Step 1: Train a **teacher model** on images and ground-truth labels



P(cat) = 0.9
P(dog) = 0.1

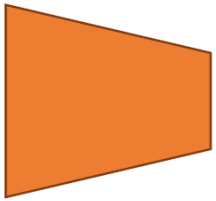
→ Cross Entropy Loss ← GT label: Cat

Step 2: Train a **student model** to match predictions from the **teacher** (sometimes also to match GT labels)



P(cat) = 0.1
P(dog) = 0.9

→ KL Divergence Loss



P(cat) = 0.2
P(dog) = 0.8

→ Cross Entropy Loss ← GT label: Dog

Hinton et al, "Distilling the knowledge in a neural network", NeurIPS Deep Learning and Representation Learning Workshop, 2015

DeiT: Data-Efficient ViT

Touvron et al. (2021) — Train ViT on ImageNet-1K only. Two key contributions.

Training Recipe

The real breakthrough — strong augmentation + regularization makes ViT work without JFT:

- AdamW optimizer
- 300 epochs (vs ViT's short schedule)
- RandAugment + Mixup + CutMix
- Label smoothing (0.1)
- Stochastic depth (drop path)
- Repeated augmentation
- Exponential Moving Average

This recipe was later adopted by ConvNeXt.

Knowledge Distillation

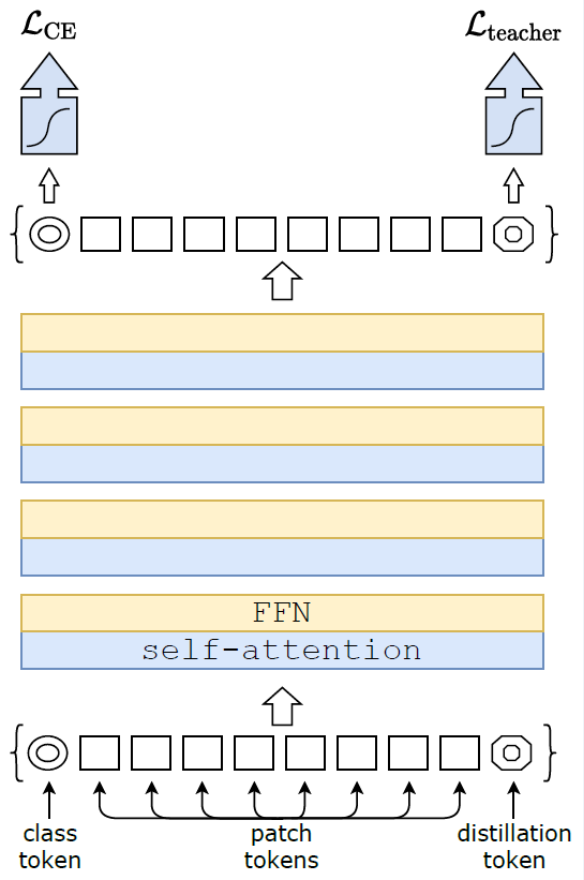
Add a [DIST] token alongside [CLS].

Teacher model: RegNet (CNN)
Student: DeiT (ViT)

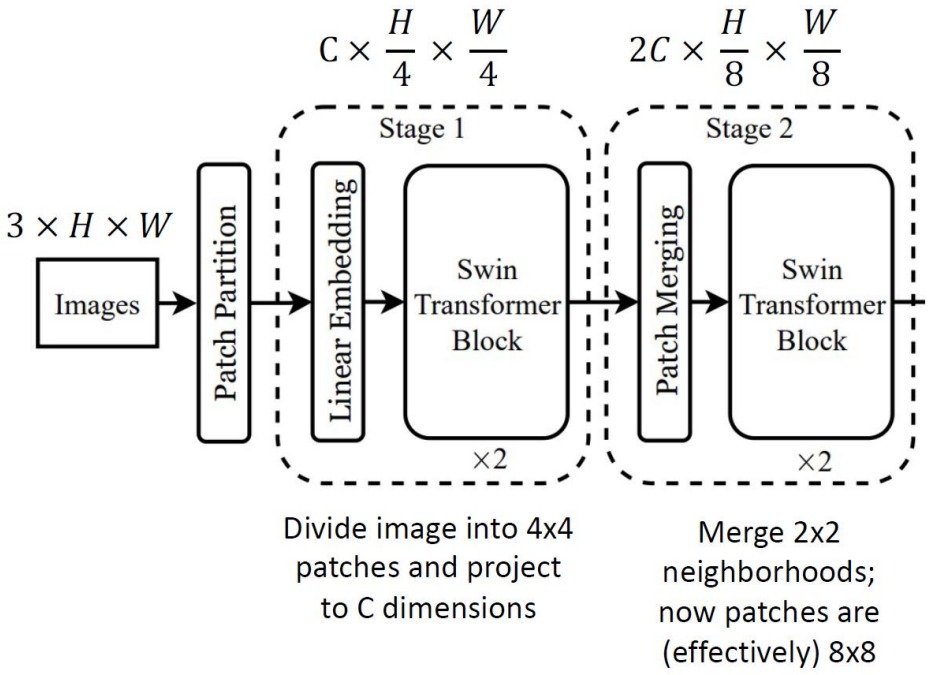
[DIST] learns to match teacher output.

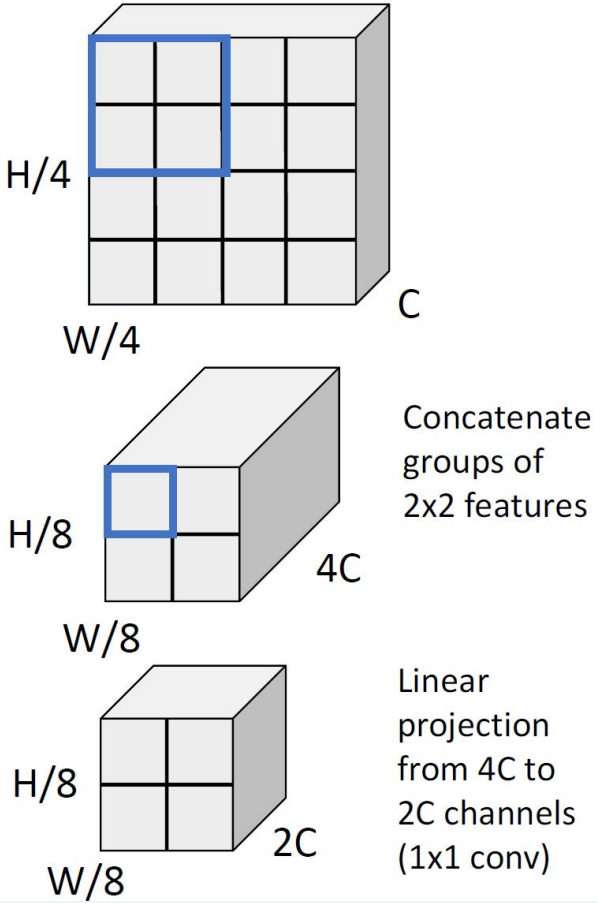
"Transfer CNN's inductive bias to ViT through soft labels."

CNN teaches ViT what to focus on, without changing ViT's architecture.



Hierarchical ViT: Swin Transformer

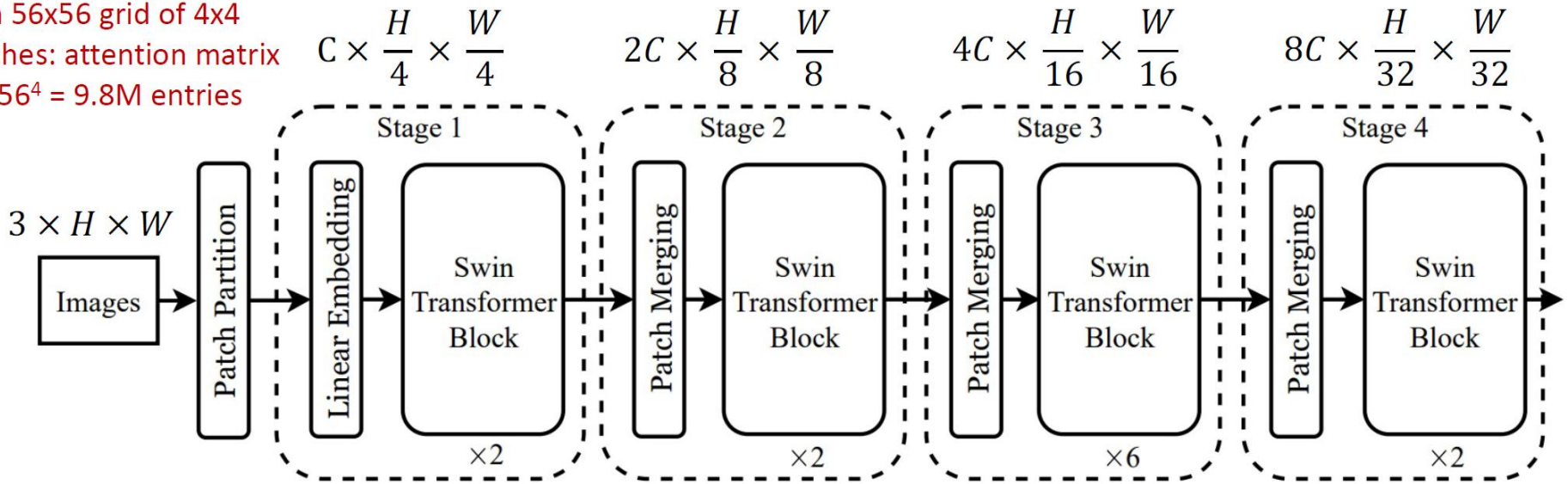




Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer

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

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

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

Swin Transformer: Window Attention

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

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

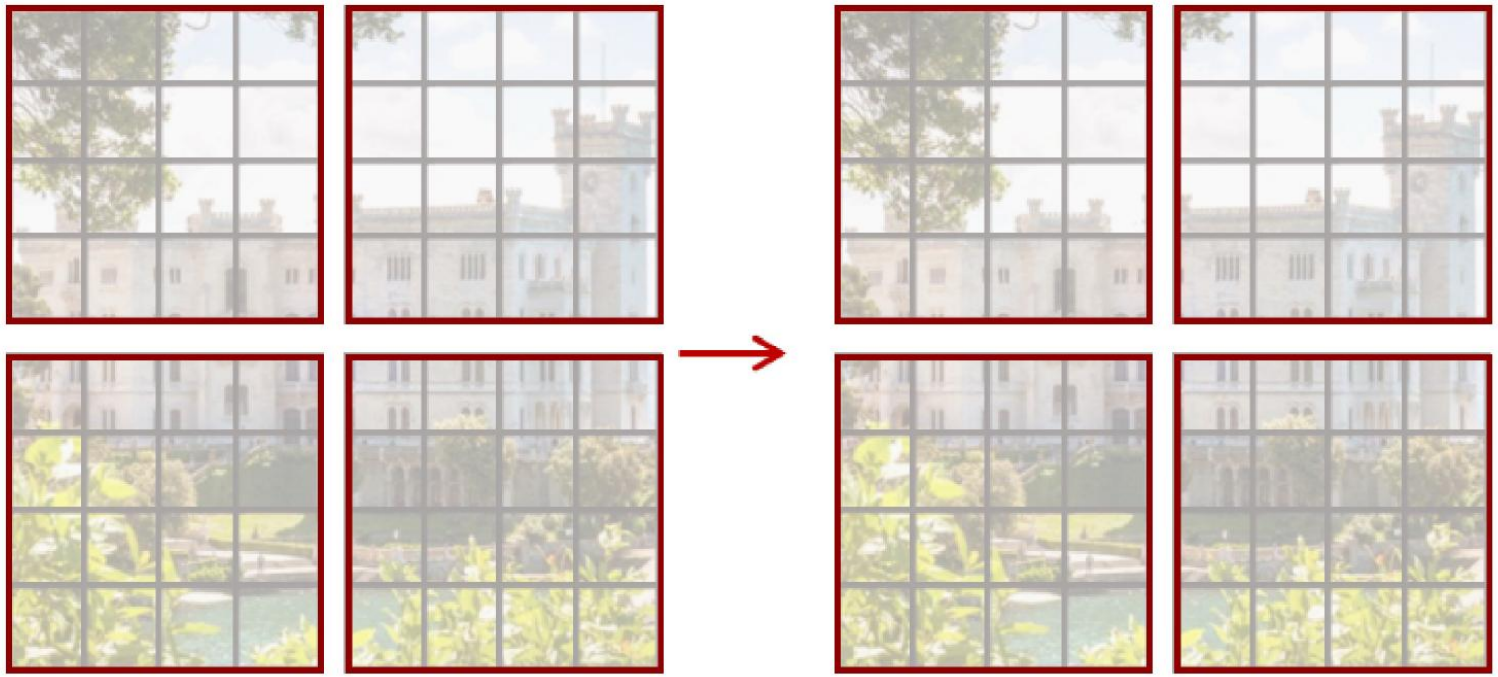
Total size of all attention matrices is now:
$M^4(H/M)(W/M) = M^2HW$

Linear in image size for fixed M!
Swin uses M=7 throughout the network

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

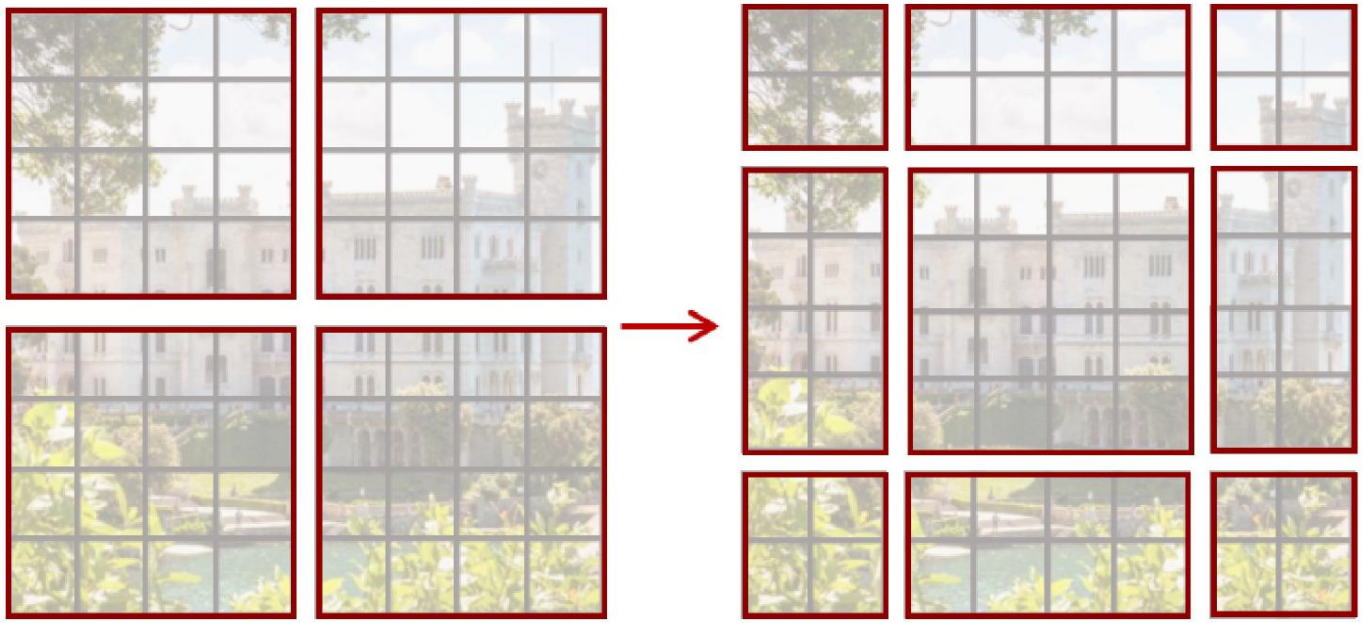
Swin Transformer: Window Attention

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



Swin Transformer: Shifted Window Attention

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



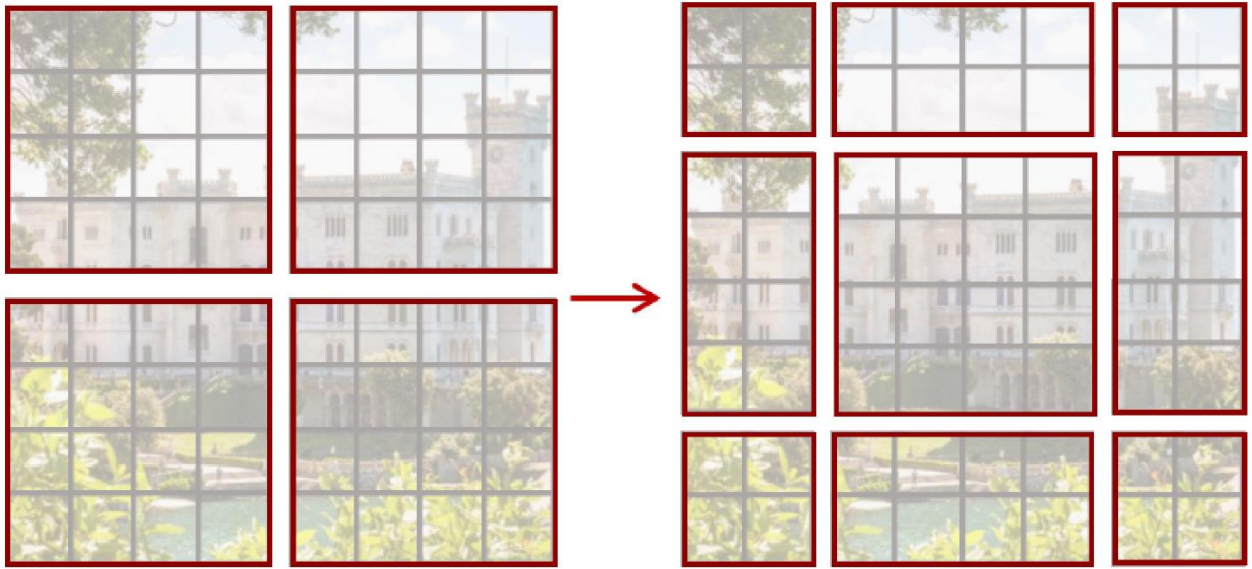
Block L: Normal windows

Block L+1: Shifted Windows

Ugly detail: Non-square windows at edges and corners

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)

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

Swin Transformer

Liu et al. (2021) — Hierarchical ViT with shifted windows. General-purpose vision backbone.

Window Attention

Attention within local windows (e.g., 7×7).
Shifted windows enable cross-window info.

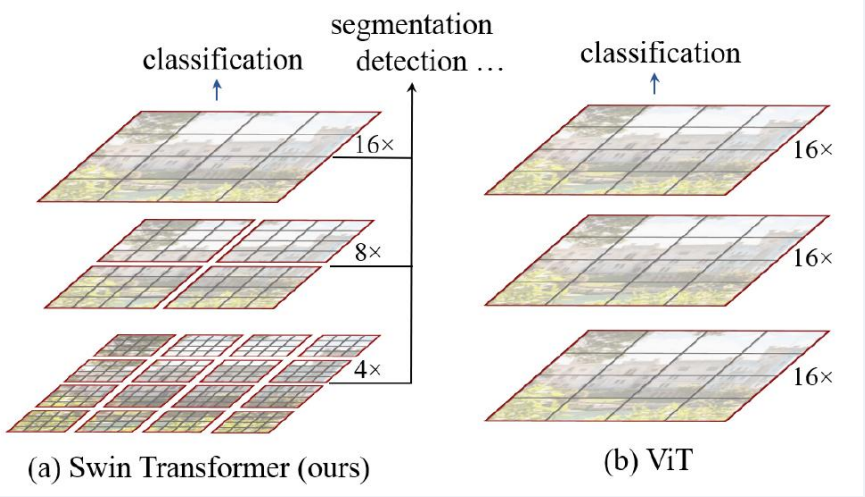
$O(N^2) \rightarrow O(N)$ — linear in image size.
Wk4 Attention Zoo "Windowed" card.

Similar idea: PVT (spatial reduction).

Hierarchical Features

Patch merging: downsample 2× per stage.
→ Multi-scale feature maps like CNN/FPN.

Stage 1: H/4 × W/4
Stage 2: H/8 × W/8
Stage 3: H/16 × W/16
Stage 4: H/32 × W/32



Swin V2 (Liu et al. 2022)

Scaling to 3B params + 1536^2 resolution:

- Cosine attention + res-post-norm
  → training stability at scale
- Log-spaced continuous position bias
  → low-res pretrain → high-res transfer
- SimMIM (masked pretraining)

→ Swin enables ViT for detection and segmentation.
Swin V2 enables scaling. We'll use this in Wk6.

Self-Supervised ViT: BEiT

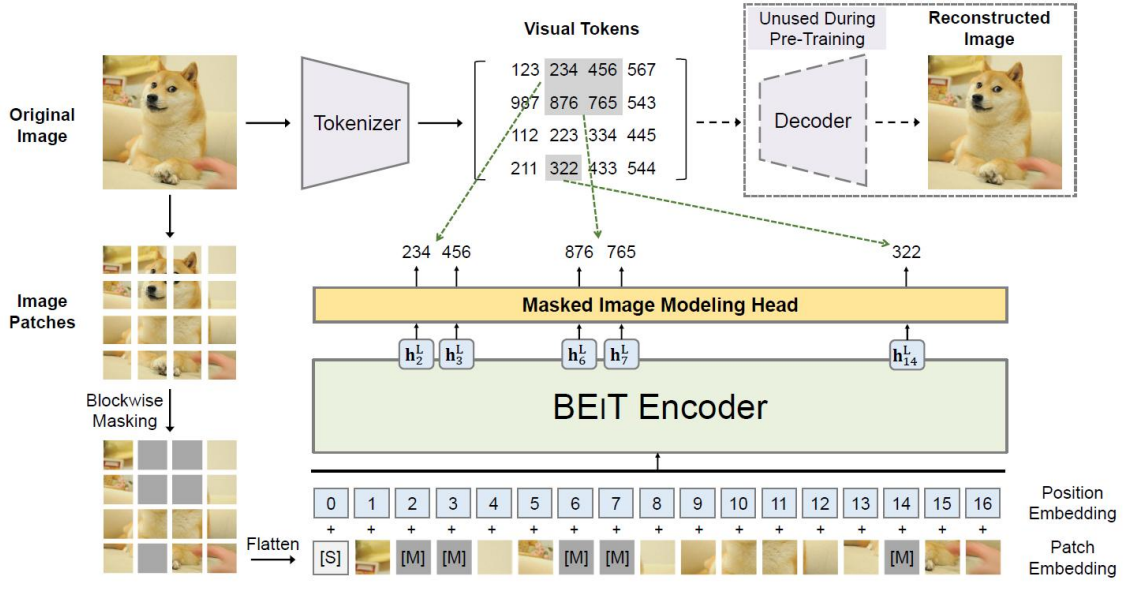
Masked Image Modeling — the vision counterpart of masked language modeling. → Wk7 SSL preview.

BERT Pre-Training of image Transformers (Bao et al. 2021)

Step 1: Train dVAE to tokenize images into discrete visual tokens.

Step 2: Mask ~40% of patches. Predict the visual tokens of masked patches (like BERT predicts words).

"BERT for images."



Impact First to apply masked language modeling idea to vision. Showed discrete visual tokens work as prediction targets.

Self-Supervised ViT: MAE

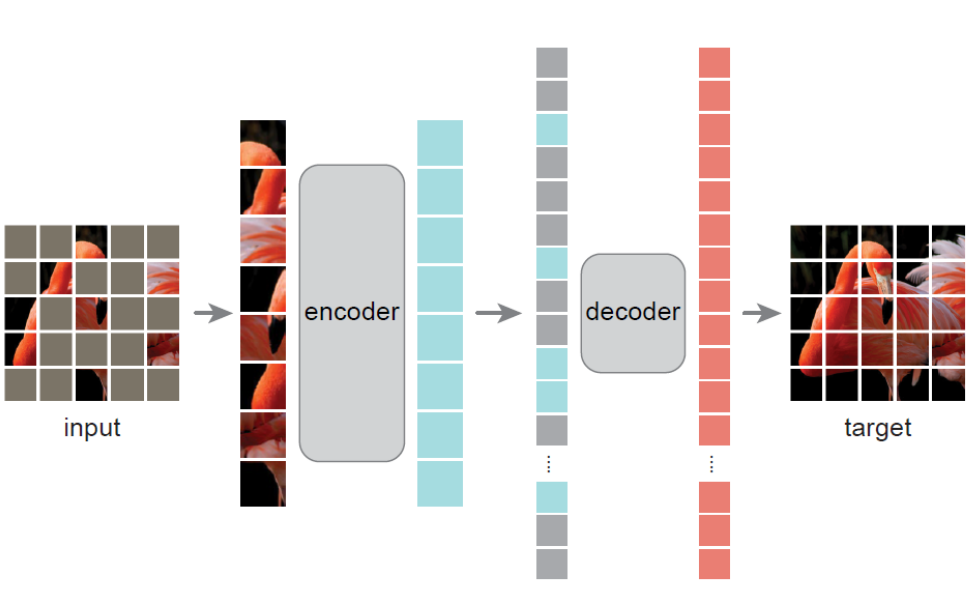
Masked Image Modeling — the vision counterpart of masked language modeling. → Wk7 SSL preview.

Masked Auto Encoder (He et al. 2022)

Mask 75% of patches (aggressive!).
Encoder processes only visible 25%
→ very efficient training.

Decoder reconstructs masked pixels.
No tokenizer needed — raw pixels.

"Autoencoders, but with masking."



Impact MAE showed ViT can be pretrained without any labels. SAM's ViT-H uses MAE pretraining. ConvNeXt V2 adapted MAE for CNNs.

Part 2.5

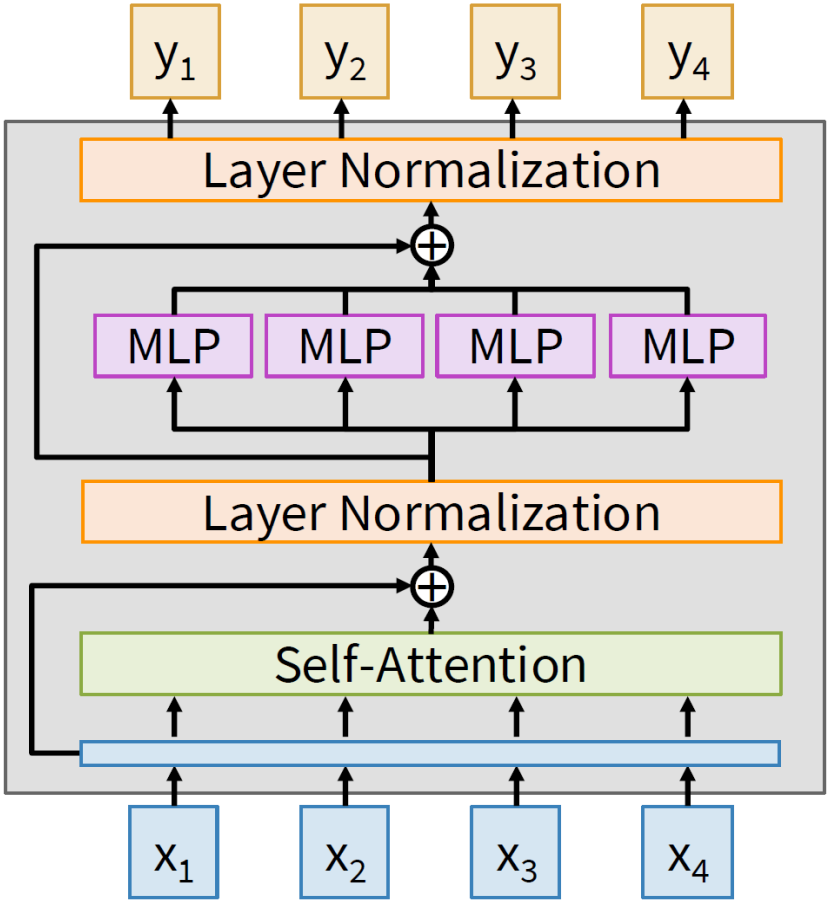
Modern Transformer Blocks

How Transformer works nowadays.

Post-Norm Transformer

Layer normalization is outside the residual connections

Kind of weird, the model can't actually learn the identify function

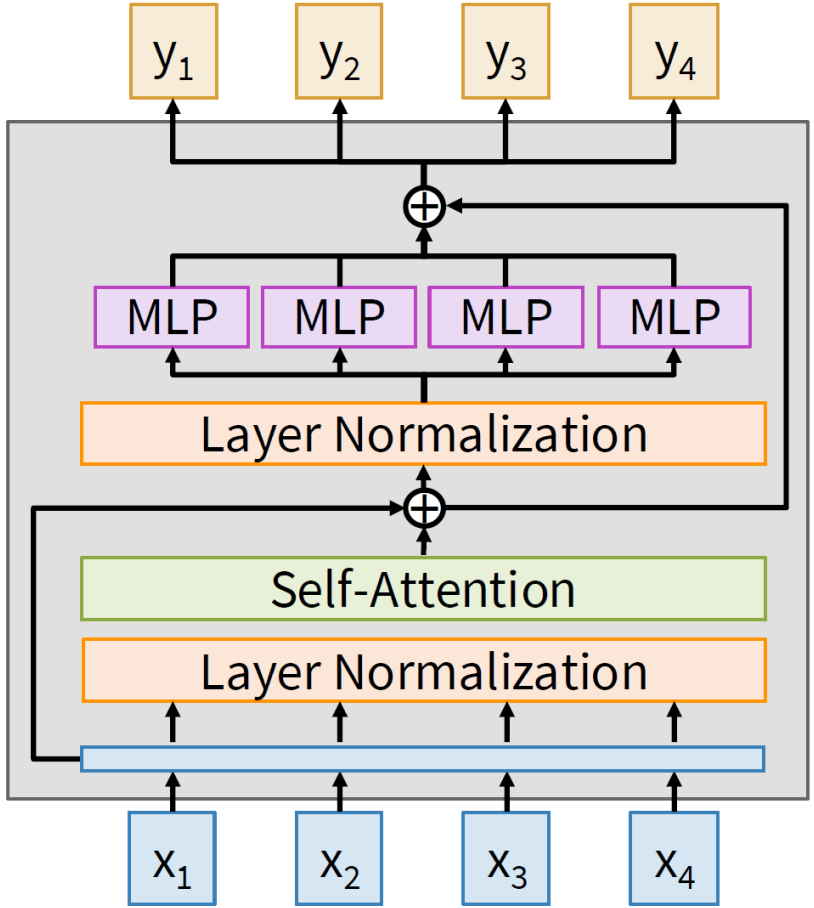


Pre-Norm Transformer

Layer normalization is outside the residual connections

Kind of weird, the model can't actually learn the identify function

Solution: Move layer normalization before the Self-Attention and MLP, inside the residual connections. Training is more stable.



RMSNorm

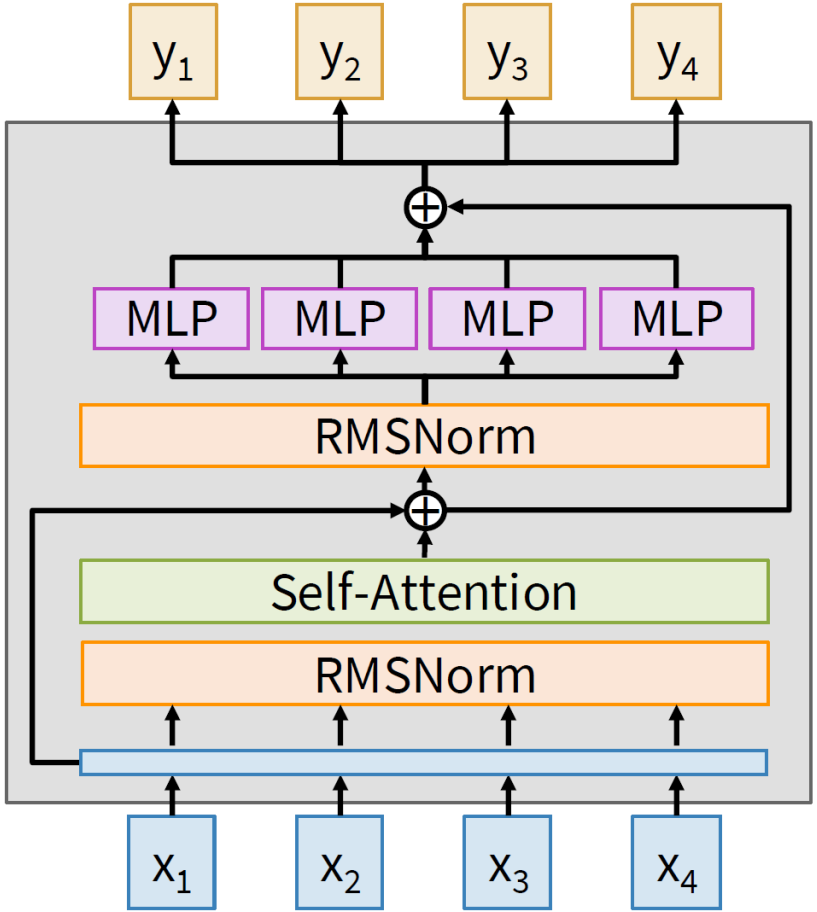
Replace Layer Normalization with Root-Mean-Square Normalization (RMSNorm)

Input: x [shape D]
Output: y [shape D]
Weight: γ [shape D]

$$y_i = \frac{x_i}{RMS(x)} * \gamma_i$$

$$RMS(x) = \sqrt{\varepsilon + \frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

Training is a bit more stable



SwiGLU MLP

Classic MLP:

Input: X [N x D]
Weights: W_1 [D x 4D]
W_2 [4D x D]
Output: $Y = \sigma(XW_1)W_2$ [N x D]

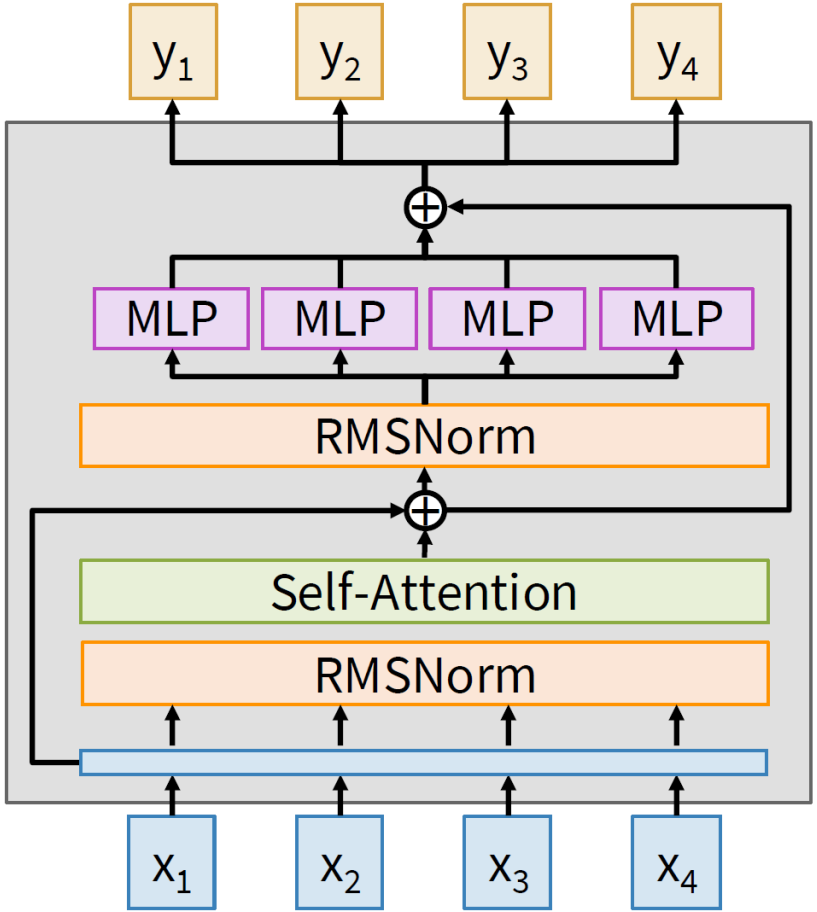
SwiGLU MLP: *Note: SwiGLU might cause "over-gating" issues in Vision → uses GELU instead (ViT-5)

Input: X [N x D]
Weights: W_1 , W_2 [D x H]
W_3 [H x D]
Output:

$$Y = (\sigma(XW_1) \odot XW_2)W_3$$

Setting H = 8D/3 keeps same total params

Shazeer, "GLU Variants Improve Transformers", 2020



Mixture of Experts (MoE)

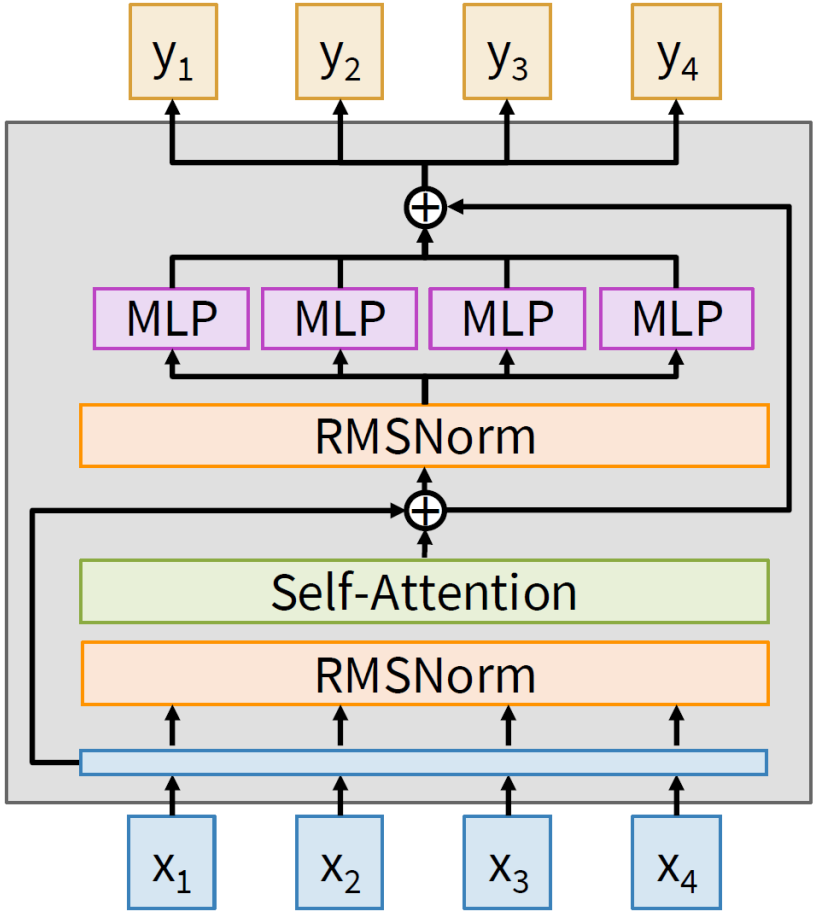
Learn E separate sets of MLP weights in each block; each MLP is an expert

W_1: [D x 4D] => [E x D x 4D]
W_2: [4D x D] => [E x 4D x D]

Each token gets routed to A < E of the experts. These are the active experts.

Increases params by E,
But only increases compute by A

All of the biggest LLMs today (e.g. GPT4o, GPT4.5, Claude 3.7, Gemini 2.5 Pro, etc) almost certainly use MoE and have > 1T params; but they don't publish details anymore



Modern Transformer Blocks

Not just new architectures — the components inside each block have been upgraded too.

Terminology: Token Mixer / Channel Mixer from MLP-Mixer (Tolstikhin et al. 2021).

Component	What you learned (Wk3-4)	Current Standard (2023→)	Why?
Norm Placement	Post-Norm Sublayer → Add → LN	**Pre-Norm** **LN → Sublayer → Add**	**Training stability**
Normalization	LayerNorm mean + variance	**RMSNorm** **variance only (no centering)**	**Faster same perf**
FFN Activation	GELU MLP $\text{GELU}(xW_1)W_2$	**SwiGLU MLP** **$(xW_1 \odot \text{SiLU}(xW\_g))W_2$** **or GELU (in ViT-5)**	**Gating better perf**
FFN Structure	Dense All tokens → same FFN	**MoE (optional)** **Router selects top-K experts**	**Scale model same compute**

MoE in 10 seconds

8 FFN experts, router selects top-2 per token.
Model params 8× but compute 2× → efficient scaling.
Mixtral, DeepSeek, GPT-4 (rumored). Details in Wk9.

Who uses what?

LLaMA / Gemma: Pre-Norm + RMSNorm + SwiGLU
Mixtral / DeepSeek: + MoE
ViT-5 (Wang et al., 2026): systematic adoption for ViT

Part 3

CNN vs ViT

The great debate of the 2020s — and its resolution.

Inductive Bias Viewpoint: CNN vs ViT

ViT paper: "ViT has much less image-specific inductive bias than CNNs. In CNNs, locality, 2D neighborhood structure, and translation equivariance are baked into each layer. In ViT, only MLP layers are local."

CNN

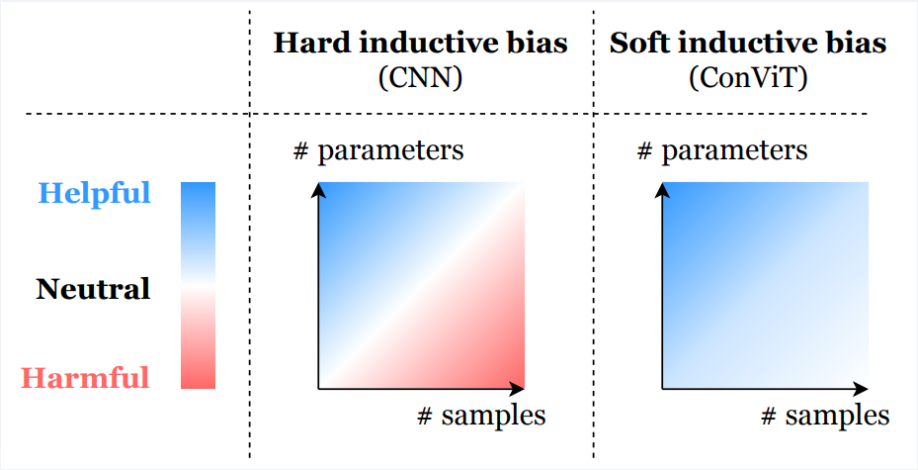
Local connectivity (3×3 kernel)
Translation equivariance (weight sharing)
2D neighborhood structure (every layer)
Hierarchical features (pooling)

Strong bias → data-efficient but limited flexibility at scale.

ViT

Global self-attention (no locality)
Permutation equivariant (need PE)
2D structure used only once (patchify)
Flat feature (single scale)

Weak bias → needs more data but higher ceiling with scale.



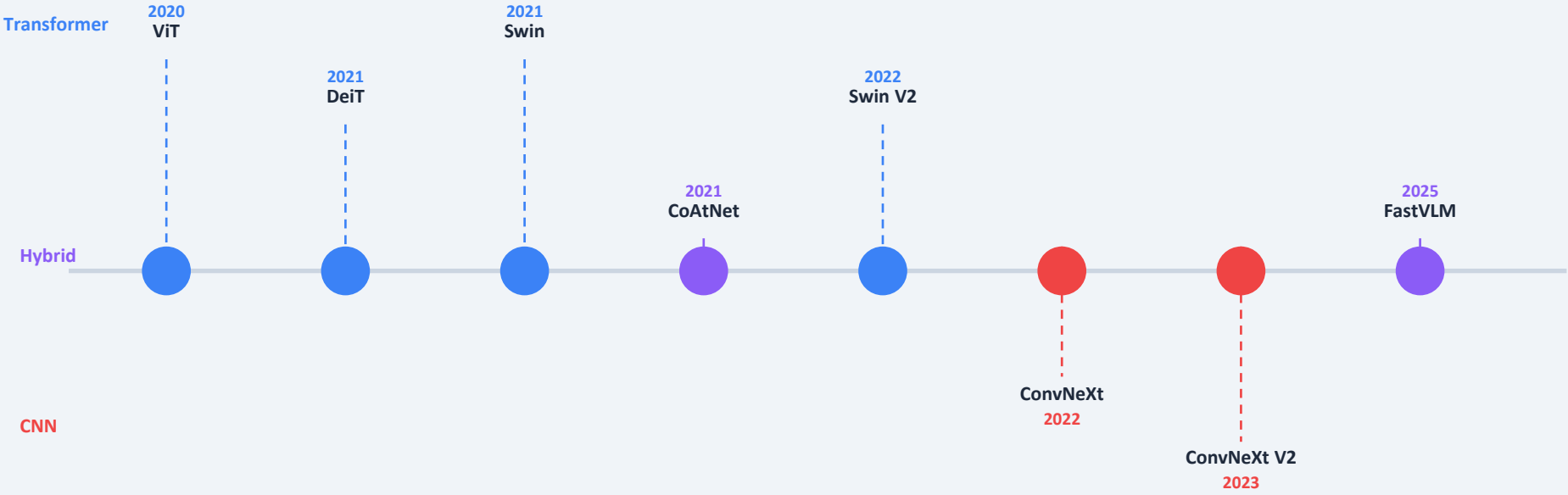
Inductive Bias Spectrum

← Weak bias — Strong bias →

MLP	ViT	Hybrid	CNN
No structure	Global Attention + learned PE	Conv stem + Attention	Locality + hierarchy

Note: Inductive bias is not a perfectly well-defined or precisely measurable concept.

CNN vs ViT: A Timeline of Convergence



Pattern ViT attacks → Transformer extends (DeiT, Swin) → CNN fights back (ConvNeXt) → CNN absorbs SSL (V2) → boundaries blur (Hybrid).

ConvNeXt V1,2: A ConvNet for the 2020s

Liu et al. (2022) — "Is it architecture or training recipe?" ResNet → modernize step by step → match Swin.

ConvNeXt: Modernizing ResNet

DeiT training recipe	300ep AdamW + augmentation	**+2.7%**
Stage ratio (3,3,9,3)	Follow Swin's compute distribution	**+0.6%**
Patchify stem (4×4 s4)	Replace 7×7 conv + maxpool	**+0.1%**
Depthwise conv + wider	Like per-head attention	**+1.0%**
Inverted bottleneck	Narrow→wide→narrow (= FFN)	**+0.1%**
7×7 depthwise kernel	Match Swin's 7×7 window	**+0.7%**
GELU, LayerNorm, fewer act	Micro design from Transformer	**+0.8%**
ResNet-50: 76.1% → ConvNeXt-T: 82.1%		**Total: +6.0%**

ConvNeXt V2 (2023)

Can CNNs use MAE?

FCMAE

Fully Convolutional MAE.
Sparse conv on visible pixels only.
No mask tokens needed for CNN.

GRN Layer

Global Response Normalization.
Fixes feature collapse (dead channels)
after FCMAE pretraining.

→ CNN can also benefit from MAE-style SSL pretraining.

Message Architecture matters less than training recipe + design choices. ConvNeXt V2 proves CNN is still competitive with modern training.

Hybrid: Conv + Transformer

Conv Stem
(Xiao et al. 2021)

Replace ViT's patchify stem (stride-16 conv) with stacked stride-2 3×3 convs.

Minimal change, big impact: +1-2% accuracy, much more stable training.

Stem only

CoAtNet
(Dai et al. 2021)

Systematic search for optimal layer composition.

Early stages: Depthwise Conv
Later stages: Self-Attention

"Best of both worlds" design.

Full architecture

FastVLM
(Apple, CVPR 2025)

FastViT: conv + transformer hybrid optimized for on-device deployment.

20× faster, 8× smaller than ViT-L/14.
MobileCLIP backbone.

On-device

Hybrid = spectrum from "stem only" to "full redesign". No single right answer — depends on task and deployment.

CNN vs ViT: Where We Stand in 2026

Foundation Models

ViT dominates

CLIP, SAM, LLaVA, Gemini all use ViT-based encoders.

Edge / Mobile Deployment

CNN or Hybrid

FastVLM, MobileNet, EfficientNet.
Conv is still more efficient.

Dense Prediction

Swin / Hybrid

Hierarchical features needed. Swin, ConvNeXt for det/seg.

The Real Answer

It depends.

Architecture matters less than training recipe + data + scale.

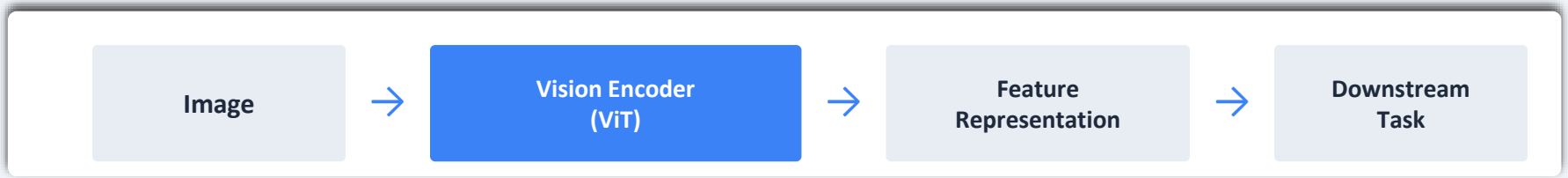
Part 4

Vision Encoders in the Wild

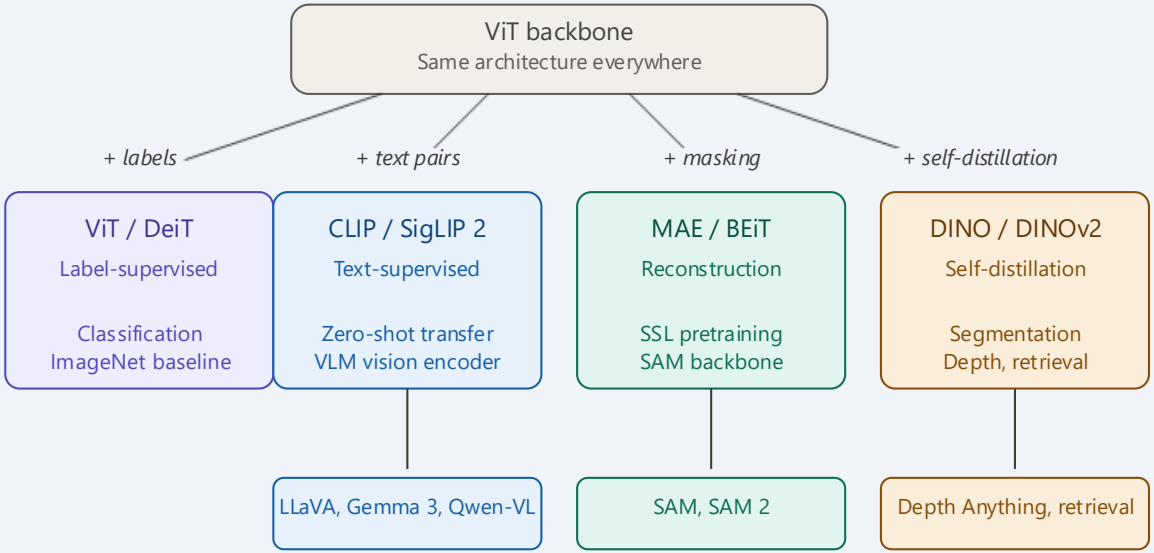
Same architecture, different training → different superpowers.

What is a Vision Encoder?

The "eyes" of every downstream model. How you train these eyes determines what they can see.



Same ViT architecture. Different training = different superpowers. What changes is the learning signal.



Vision Encoder Learning Paradigms

"Is there external supervision, or does the model learn from images alone?"

Supervised

External signal tells the model what to learn.

Label-Supervised

Image + Label

Human annotations. Predict class from image.
Models: ViT, DeiT
Historical starting point. Rarely used alone in modern VLMs.

Text-Supervised

Image + Text

Image-text pairs from web. Contrastive or captioning loss.
Models: CLIP, SigLIP, SigLIP 2
Dominant paradigm for VLM vision encoders.
Enables zero-shot transfer and multimodal alignment.

Self-Supervised

Model learns from images themselves. No labels, no text.

Reconstruction

Mask → Reconstruct

Mask patches, predict missing content.
Models: MAE (pixels), BEiT (visual tokens)
Learns spatial structure and local features.

Self-Distillation

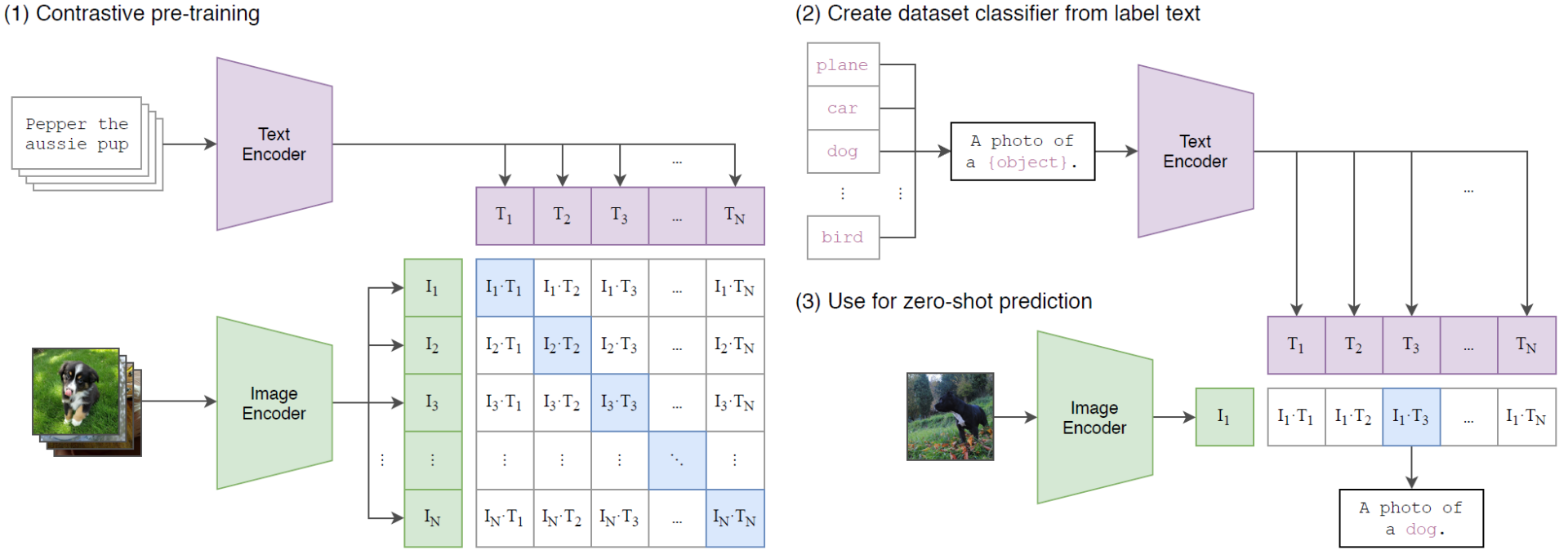
Teacher ← Student

Teacher = momentum average of student.
Different augmented views → consistent representation.
Models: DINO, DINOv2, BYOL
Learns global semantics. Attention maps segment objects.

Recent trend Boundaries blur. DINOv2 = distillation + reconstruction. SigLIP 2 = text + captioning + distillation + masking. DINOv3 = SSL + text.

CLIP: Contrastive Language-Image Pretraining

Radford et al. (2021, OpenAI) — 400M image-text pairs. Zero-shot transfer to any classification task.



Key models CLIP ViT-B/16, CLIP ViT-L/14 (most used). Used in: Stable Diffusion, LLaVA, many VLMs. → Wk7 SSL, Wk9 Foundation Models, Wk12 VLM.

SigLIP → SigLIP 2

SigLIP (Zhai et al. 2023)

Replace CLIP's softmax contrastive loss with pairwise sigmoid loss.

Each image-text pair = independent binary classification. No global normalization.

→ Better parallelization across devices.
→ Scales to larger batches more easily.

SigLIP 2 (Tschannen et al. 2025)

Unified recipe combining:
- Sigmoid contrastive loss
- Captioning-based pretraining (from CoCa)
- Self-distillation + masked prediction
- Online data curation
- NaFlex: dynamic resolution + aspect ratio

Used in: Gemma 3, PaLI.
Models: ViT-B to ViT-g (1B params).

Trend SigLIP 2 = all paradigms merged.
Contrastive + captioning + self-supervised. The boundaries we drew in the previous slide are already blurring.

DINOv2: All-Purpose Visual Features

Oquab et al. (2023/2024, Meta) — Self-supervised features that work everywhere, without fine-tuning.

Key Ideas

Combines DINO (self-distillation) +
iBOT (masked prediction) in one framework.

Trained on curated 142M images (LVD-142M)
— automatic curation pipeline, not web noise.

ViT-g: 1B params. Distilled to smaller models.

No text, no labels. Pure self-supervised.
Yet competitive with CLIP on many tasks.

Why It Matters

Frozen features work for:

- Classification (linear probe)
- Segmentation (dense)
- Depth estimation
- Retrieval

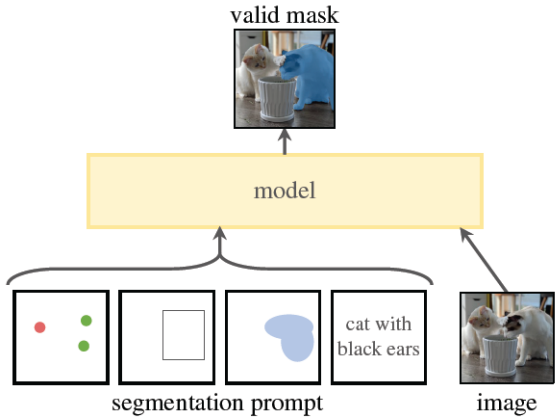
→ True "foundation" vision model.
No fine-tuning needed for many tasks.

DINOv3 (2025): adds text alignment.

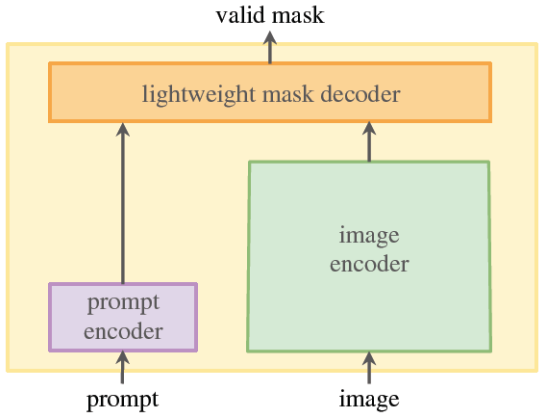
	INet-1k k-NN	INet-1k linear
iBOT	72.9	82.3
+(our reproduction)	74.5 ↑ 1.6	83.2 ↑ 0.9
+LayerScale, Stochastic Depth	75.4 ↑ 0.9	82.0 ↓ 1.2
+128k prototypes	76.6 ↑ 1.2	81.9 ↓ 0.1
+KoLeo	78.9 ↑ 2.3	82.5 ↑ 0.6
+SwiGLU FFN	78.7 ↓ 0.2	83.1 ↑ 0.6
+Patch size 14	78.9 ↑ 0.2	83.5 ↑ 0.4
+Teacher momentum 0.994	79.4 ↑ 0.5	83.6 ↑ 0.1
+Tweak warmup schedules	80.5 ↑ 1.1	83.8 ↑ 0.2
+Batch size 3k	81.7 ↑ 1.2	84.7 ↑ 0.9
+Sinkhorn-Knopp	81.7 =	84.7 =
+Untying heads = DINOv2	82.0 ↑ 0.3	84.5 ↓ 0.2

SAM: Segment Anything

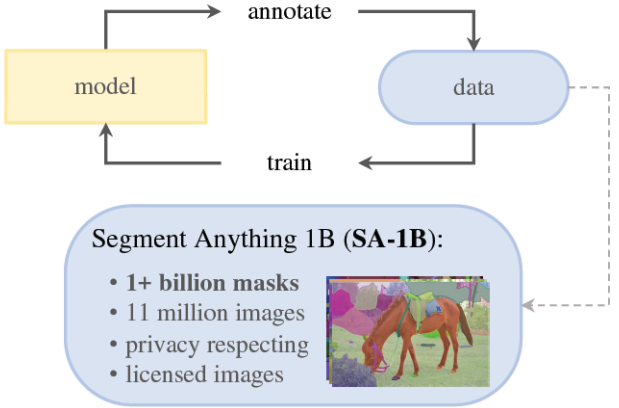
Kirillov et al. (2023, Meta) — Promptable segmentation for any image.



(a) **Task**: promptable segmentation



(b) **Model**: Segment Anything Model (**SAM**)



(c) **Data**: data engine (top) & dataset (bottom)

Vision encoder ViT-H/14 (632M params) pretrained with MAE. Dense feature extraction (not just [CLS]). SAM 2 (2024): video extension. → Wk6, Wk9, Wk14.

Vision Encoders in VLMs

How do multimodal models "see"? They borrow a pretrained vision encoder. → Wk12 VLM.

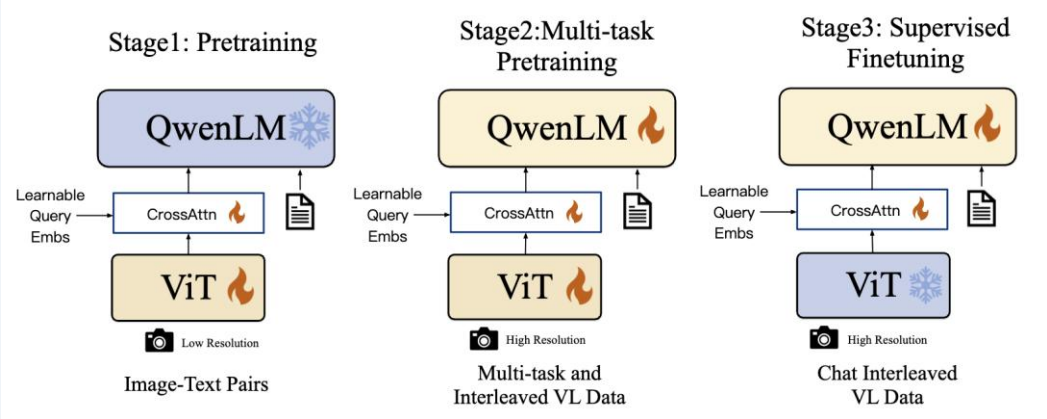
LLaVA (Liu et al. 2023)

- CLIP ViT-L/14 → MLP projection → LLM
- Vision encoder: frozen (pretrained CLIP)
- Only the MLP connector is trained
- "Use the best available eyes, just learn how to connect them to the language brain"
- LLaVA-NeXT: also supports SigLIP

Qwen2.5-VL / Kimi-VL

- Train own ViT with SigLIP + captioning
- Dynamic resolution: handle any aspect ratio natively
- Full fine-tune: vision encoder trained together with LLM
- "Build your own eyes and train everything end-to-end"

Two strategies: frozen pretrained encoder (LLaVA) vs end-to-end trained encoder (Qwen). Trade-off: simplicity vs performance.



Efficient Deployment

Making vision encoders fast enough for real-world use. Especially relevant for LPCVC challenge teams.

Mixed Precision (FP16 / BF16)

Stage: Training

Use half-precision floats.
Same accuracy, ~2× speedup,
~half memory.
BF16 preferred (same range as FP32).

Quantization (INT8 / INT4)

Stage: Inference

Reduce weight precision post-training. INT8: minimal accuracy loss. INT4: more loss but 4× smaller model.

Pruning & Distillation

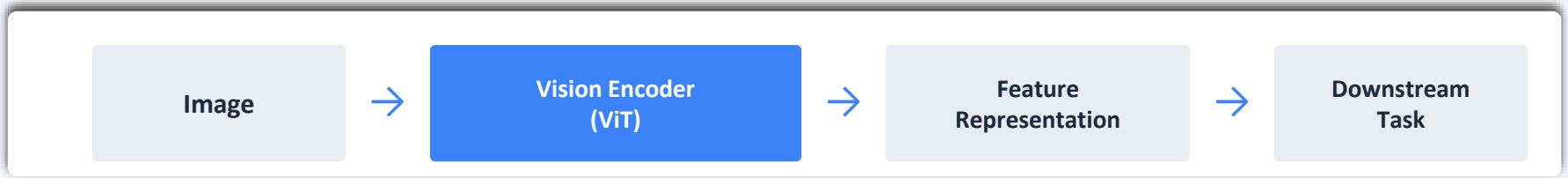
Stage: Model Design

Pruning: remove unimportant weights/heads. Distillation: train small model to mimic large one (DeiT, DINOv2).

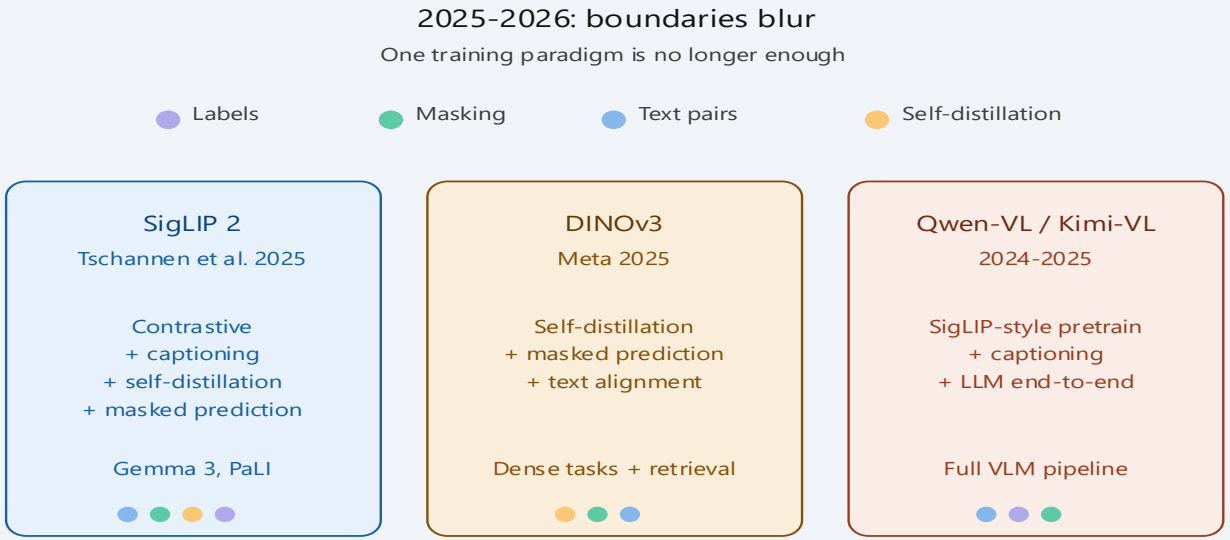
We'll revisit efficiency/deployment in depth at Wk9 Foundation Models. Michigan L18 has detailed FP16/BF16/TF32 comparison.

Vision Encoders: Evolution

From single-paradigm models to combined training — boundaries are blurring fast.



2025–2026: one training paradigm is no longer enough. The winning recipe combines them all.



2026 VLM Vision Encoder Map

What vision encoder does each major VLM actually use?

VLM	Vision Encoder	Training	Notable
GPT	Undisclosed	Undisclosed	
Gemini / Gemma 3	SigLIP 2	Contrastive + Caption + SSL	NaFlex (dynamic res)
Claude	Undisclosed	Undisclosed	
Qwen2.5-VL	Custom ViT	SigLIP + Captioning	Dynamic resolution
LLaVA-1.5/NeXT	CLIP ViT-L/14	Contrastive	Frozen encoder
InternVL 2.5	InternViT-6B	Contrastive	Largest open ViT
Kimi-VL	MoonViT	SigLIP + Captioning	Full fine-tune
FastVLM (Apple)	FastViT (Hybrid)	MobileCLIP	On-device, 20× faster

Trends SigLIP-style training becoming standard. Dynamic resolution is the new norm. Open models converge around 300M-6B param ViTs.

Summary & Resources

What We Covered

ViT: patches, [CLS], PE, model variants
ViT attention visualization (DINO)

DeiT: training recipe + distillation
Swin: hierarchical + windowed attention
Masked Modeling (BEiT/MAE)

CNN vs ViT: inductive bias, scaling
Timeline: ViT → ConvNeXt → Hybrid

Learning paradigms: Supervised/Self-Supervised
CLIP, SigLIP 2, DINOv2, SAM, VLMs
2026 VLM vision encoder landscape

Further Reading & Self-Study

Michigan EECS 498 L18: Vision Transformers

UvA DL Tutorial 15: ViT implementation
uvadlc-notebooks.readthedocs.io

Jay Alammar: The Illustrated Transformer
jalammar.github.io/illustrated-transformer

Lilian Weng: The Transformer Family v2
lilianweng.github.io

HuggingFace Blog: SigLIP 2
huggingface.co/blog/siglip2

ViT-5 (Want et al., 2026) — modern ViT design
arxiv.org/abs/2602.08071

Next Week Week 6: Detection & Segmentation - putting these vision encoders to work on spatial understanding

Acknowledgments

Some slides and figures in this course are adapted from or inspired by the following open resources.

- **Stanford CS231n: Deep Learning for Computer Vision (Spring 2025)**
  Fei-Fei Li, Ehsan Adeli, et al. | cs231n.stanford.edu
- **UMich EECS 498/598: Deep Learning for Computer Vision (Winter 2022)**
  Justin Johnson | web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/
- **MIT 6.7960: Deep Learning (Fall 2024)**
  Phillip Isola, Sara Beery, Jeremy Bernstein | ocw.mit.edu/courses/6-7960-deep-learning-fall-2024/
- **CMU 16-824: Visual Learning and Recognition (Fall 2025)**
  Jun-Yan Zhu | visual-learning.cs.cmu.edu
- **Key papers: ViT, DeiT, Swin, MAE, CLIP, SigLIP 2, DINOv2, SAM, LLaVA, ConvNeXt, …**