

CENG501 – Deep Learning

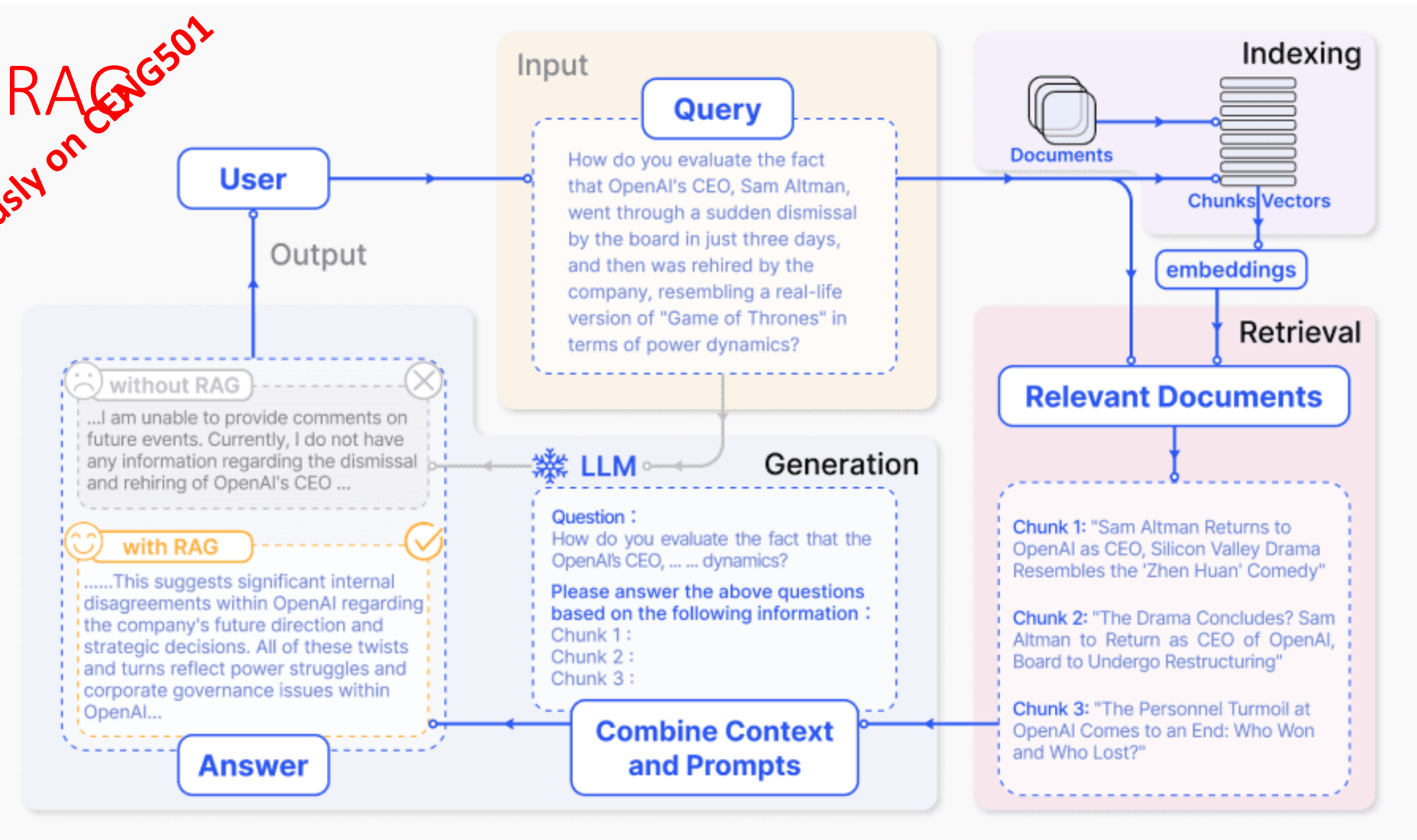
Week 10

Spring 2026

Sinan Kalkan

Dept. of Computer Engineering, METU

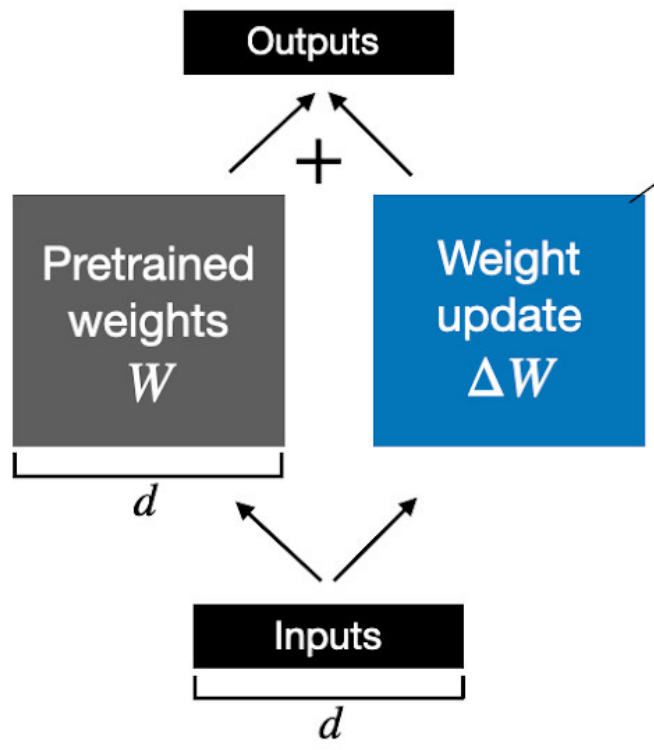
Previously on CENG501



Previously on ~~CEA~~ **LoRA** G501

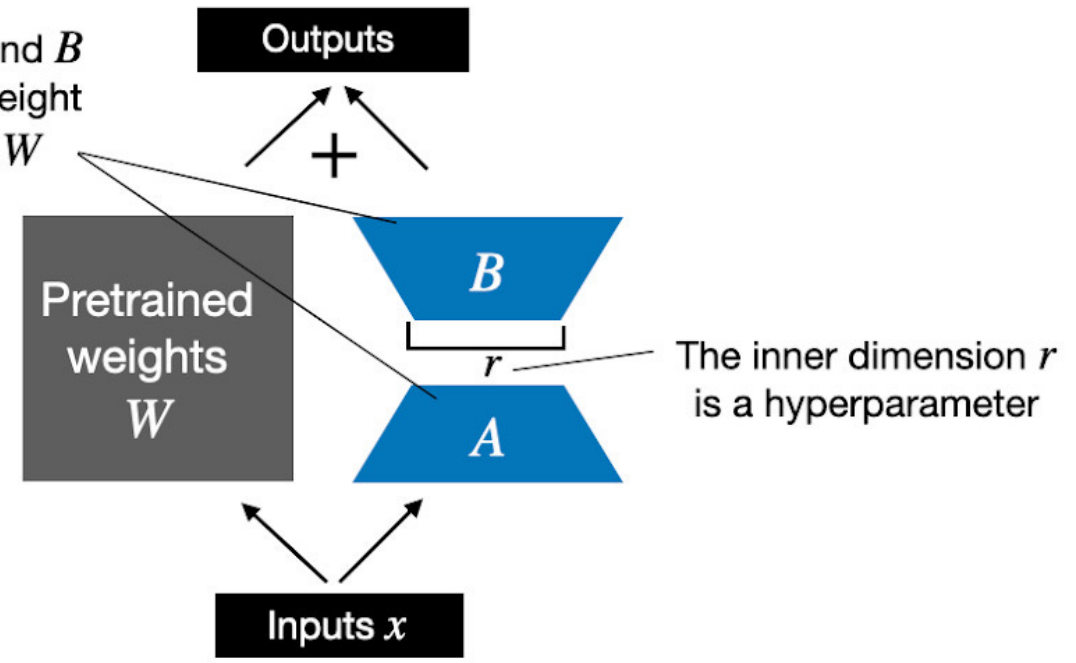
Hu et al., "LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS", 2021.

Weight update in regular finetuning



Weight update in LoRA

LoRA matrices A and B approximate the weight update matrix ΔW



Target: Attention blocks

Previously on TA
LORENG501

Hu et al., "LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS", 2021.

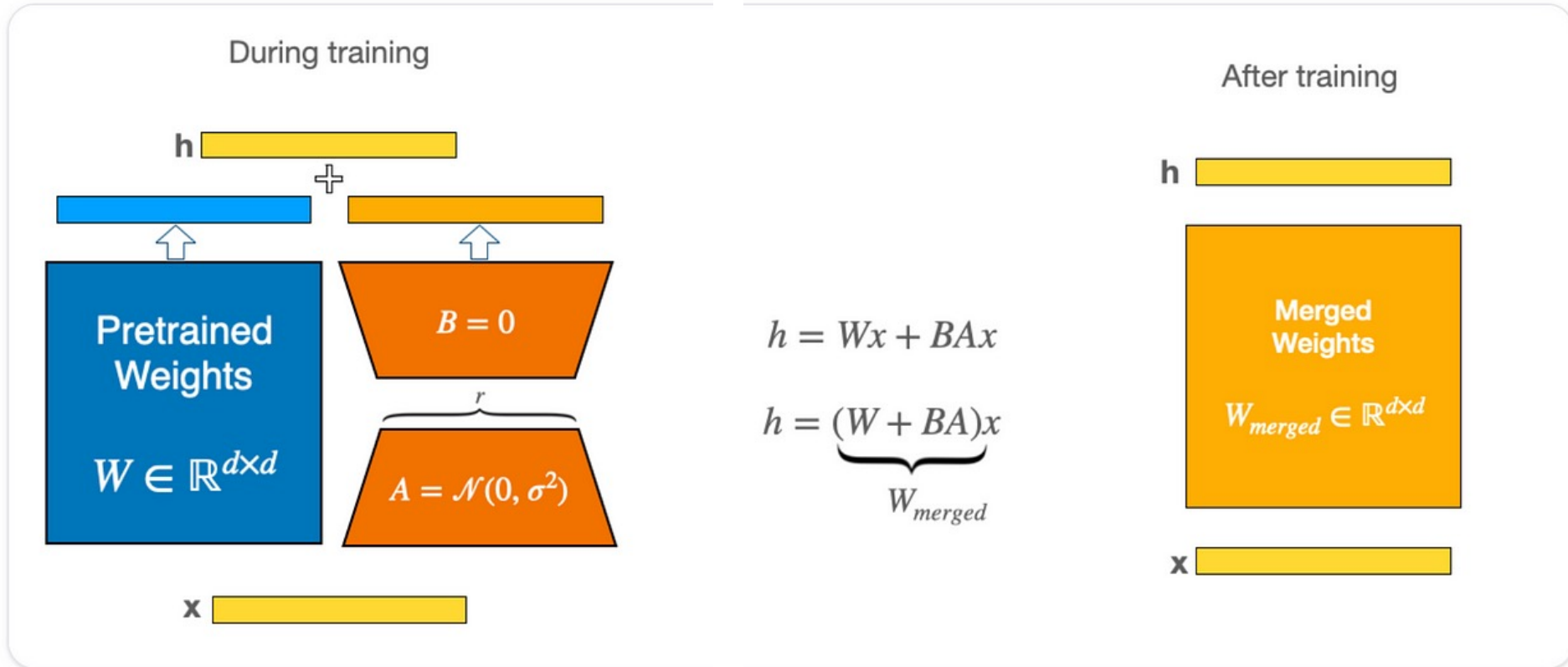


Fig: <https://medium.com/@kailash.thiyagarajan/fine-tuning-large-language-models-with-lora-demystifying-efficient-adaptation-25fa0a389075>

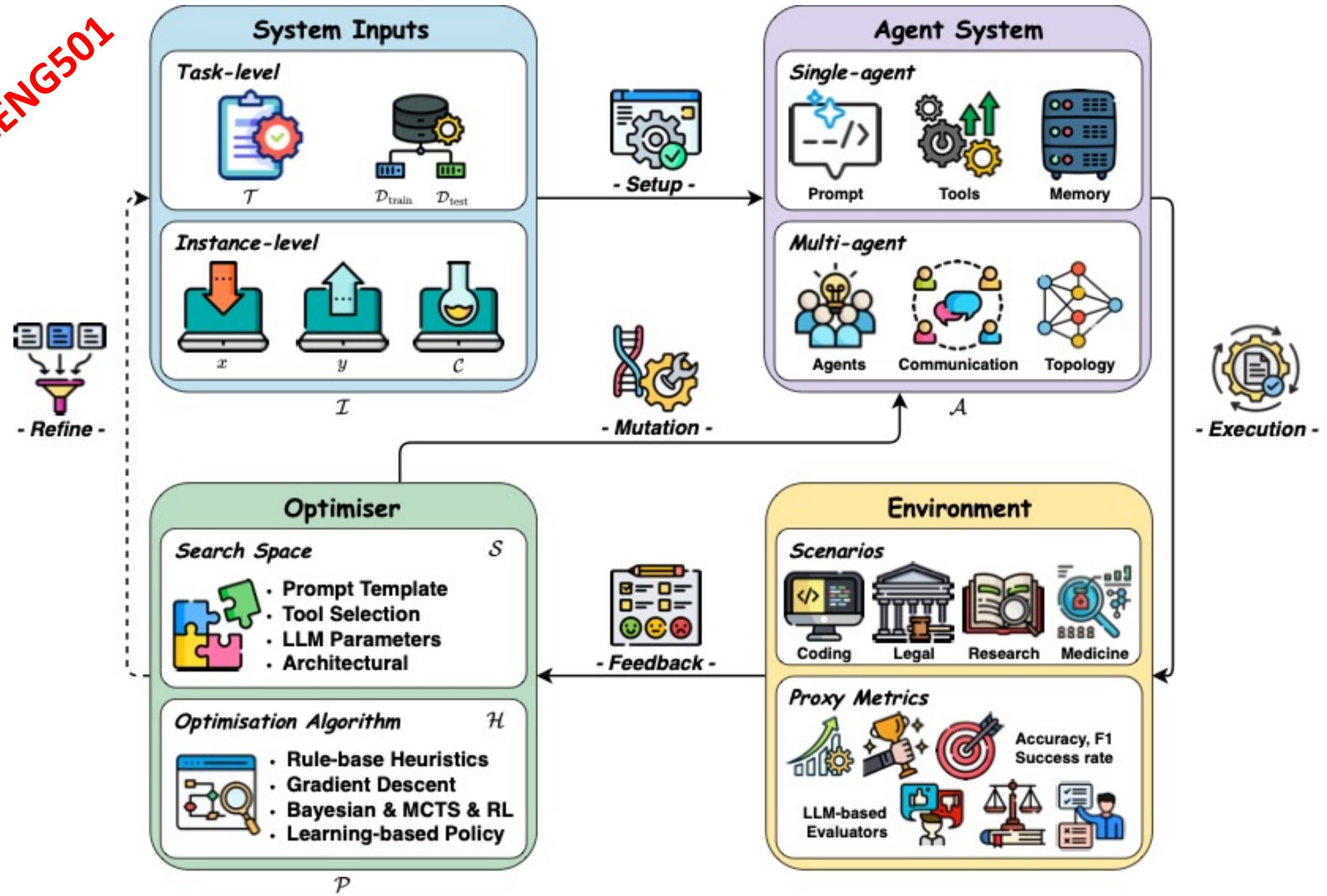
Previously on ENG501

Finetuning vs. RAG

Factors to consider:

- Nature of the task: Specialized tasks might benefit from finetuning. RAG is better for problems requiring external / up-to-date knowledge.
- Data availability: Finetuning requires a lot of data. RAG can utilize existing data.
- Resource: Finetuning can be expensive. RAG is easy to integrate.

Previously on CENG501



Material from: <https://arxiv.org/pdf/2508.07407>

Previously on CENG501

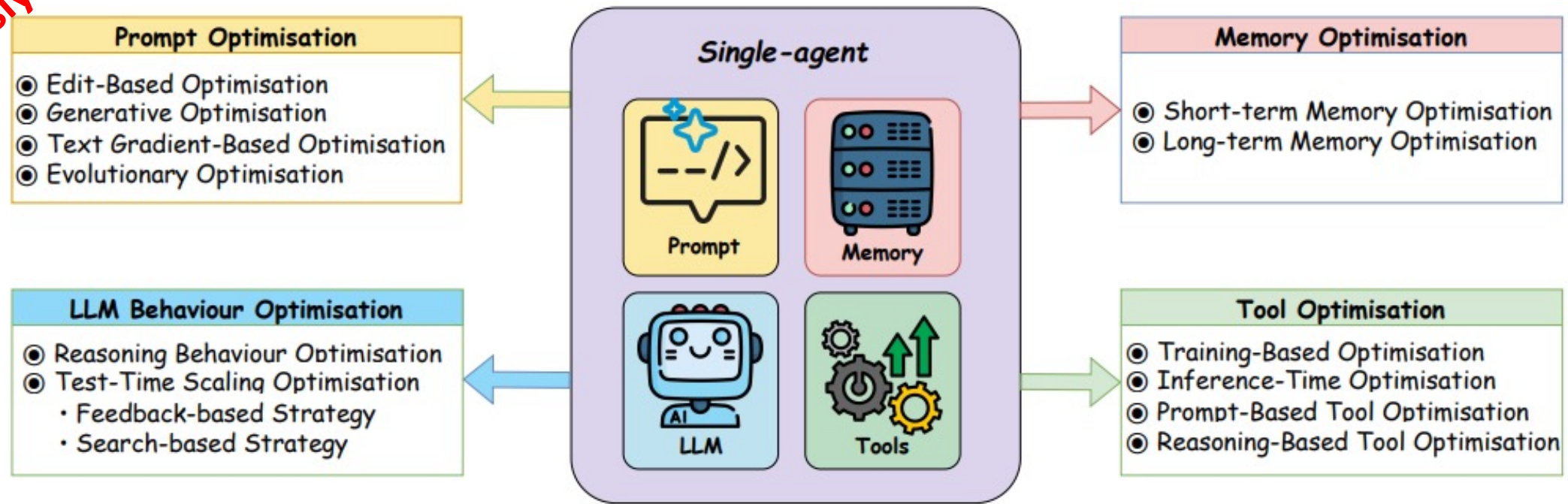


Figure 4 An overview of single-agent optimisation approaches, categorised by the targeted component within the agent system: prompt, memory, and tool.

YOUR AGENT MAY MISEVOLVE: EMERGENT RISKS IN SELF-EVOLVING LLM AGENTS

Shuai Shao^{1,2*}, Qihan Ren^{1,2*†}, Dongrui Liu^{1*}, Chen Qian^{1,3}, Boyi Wei⁴, Dadi Guo^{1,5},
Jingyi Yang^{1,6}, Xinhao Song^{1,2}, Linfeng Zhang², Weinan Zhang², Jing Shao^{1§}

¹Shanghai Artificial Intelligence Laboratory ²Shanghai Jiao Tong University

³Renmin University of China ⁴Princeton University

⁵Hong Kong University of Science and Technology ⁶Fudan University

{shaoshuai.ederson, renqihan}@sjtu.edu.cn {liudongrui, shaojing}@pjlab.org.cn

ABSTRACT

Advances in Large Language Models (LLMs) have enabled a new class of *self-evolving agents* that autonomously improve through environmental interaction, demonstrating strong capabilities. However, self-evolution also introduces novel risks overlooked by current safety research. In this work, we study case where an agent’s self-evolution deviates in unintended ways, leading to undesirable or even harmful outcomes. We refer to this as *Misevolution*. We evaluate misevolution along four key evolutionary pathways: model, memory, tool, and workflow. Our empirical findings reveal that misevolution is a widespread risk, affecting agents built even on top-tier LLMs (e.g., Gemini-2.5-Pro). Different emergent risks are observed, such as degradation of safety alignment after memory accumulation, or unintended introduction of vulnerabilities in tool creation and reuse. To our knowledge, this is the first study to systematically conceptualize misevolution and provide empirical evidence of its occurrence, highlighting an urgent need for new safety paradigms for self-evolving agents. Finally, we discuss potential mitigation strategies to inspire further research on building safer and more trustworthy self-evolving agents. Our code is available here.

Hallucination

- LLMs lack epistemic uncertainty (a measure of lack of knowledge)
- They can confidently fabricate data, even when they admit so when confronted
- LLMs struggle with negative constraints

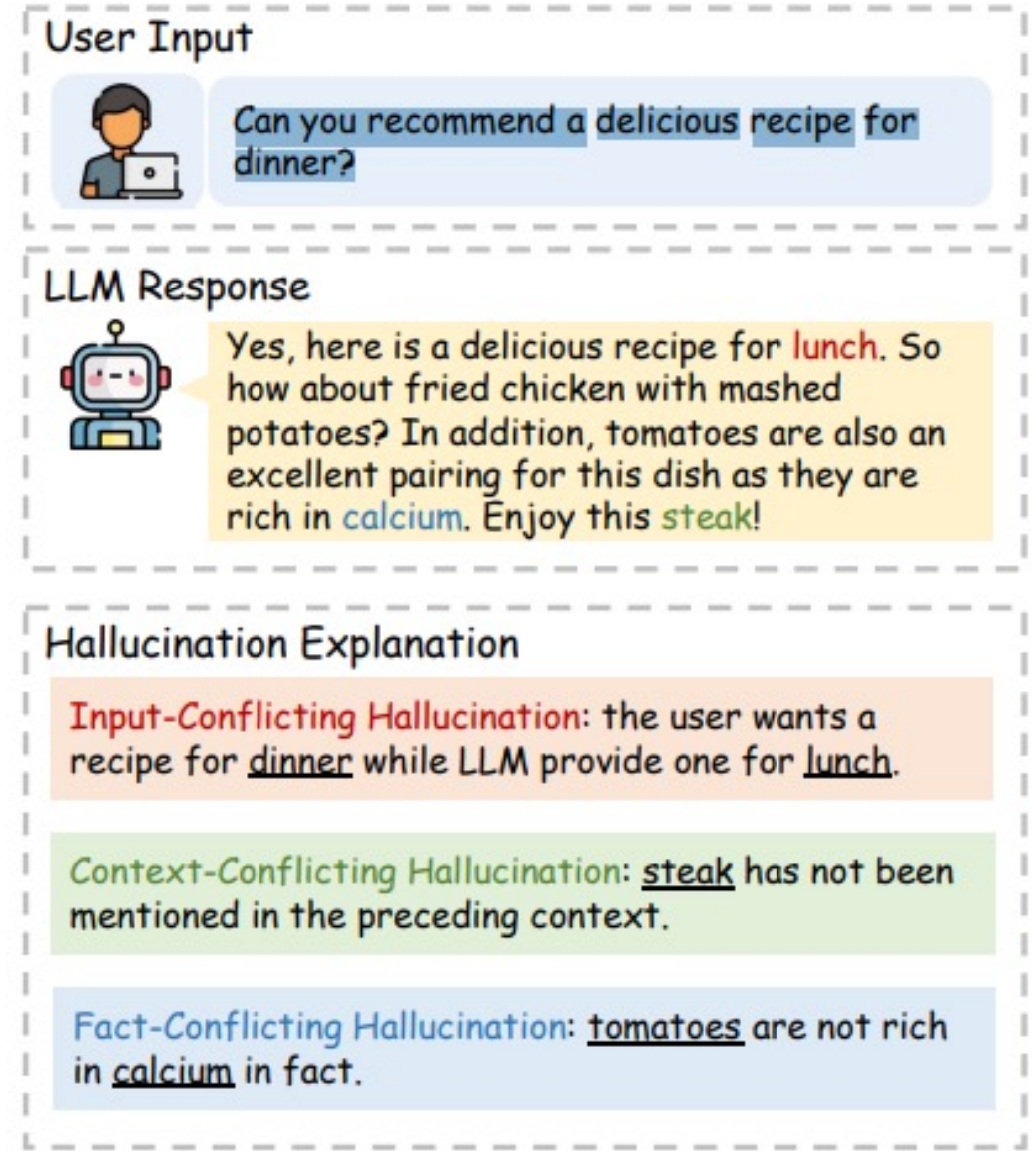


Fig: Unveiling Hallucination in Text, Image, Video, and Audio Foundation Models: A Comprehensive Survey, 2024.

Trustworthiness and Robustness Issues

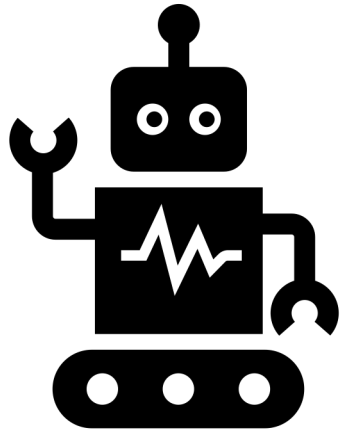
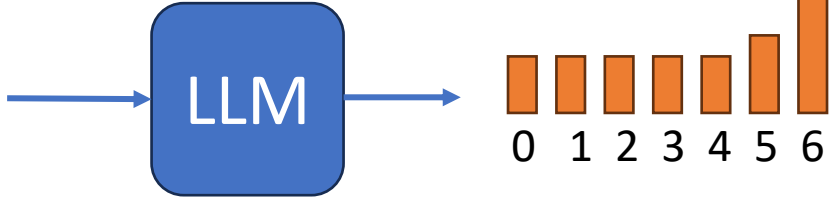
- Distribution shift
 - An LLM will become obsolete if its training data becomes outdated
- Issues with Private Data
 - LLMs can memorize sensitive data
- Bias issues
- Adversarial vulnerability
- Sycophancy

Previously on CENG501

Lack of Grounding

- LLMs are statistical pattern matchers trained for next-token-prediction tasks
- When an LLM is used on a robot or talks about a physical concept, event, world,
 - it just does so without a physical grounding/understanding.
 - it just estimates the next most likely tokens.

“Bir sepette 12 yumurta var, altı çıktı, kaç kaldı?”



Challenges with Causal, Logical Reasoning

Previously of CENG501

I need to get my car washed. There is a car wash 100m away from my house. Should I walk or drive?

Walk.

Driving 100 m is one of those cases where the overhead dominates the task:

- By the time you get in, start the car, maneuver, and park, you've already spent more time than walking.
- Cold starts increase fuel consumption and engine wear disproportionately.
- You avoid a completely unnecessary short trip (which is actually one of the least efficient use cases for a car).

At a normal pace, 100 m is about **1–1.5 minutes** of walking—basically negligible effort.

The only reasonable exceptions:

- Heavy rain or extreme weather
- Mobility issues
- You need to transport something bulky

Otherwise, walking is strictly better—simpler, faster in practice, and mildly healthier.

Today

- Vision Transformers
 - ViT
 - Swin v1, v2
- Dense prediction with vision transformers
- ConvNext: Modern CNNs with inspiration from Swin
- Fast/Faster ViTs
- Pretraining Vision Transformers

Administrative Notes

- Project next steps:
 - Milestones:
 1. Milestone (April 10, midnight):
 - Read & understand the paper
 - Download the datasets
 - Prepare the Readme file excluding the results & conclusion
 2. Milestone (May 4, midnight)
 - The results of the first experiment
 3. Milestone (June 1, midnight)
 - Final report (Readme file)
 - Repo with all code & trained models

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, neilhoulby}@google.com

Vision Transformers (ViT)

ViT: Motivation

- NLP:
 - Successful results of Transformers
 - De facto architecture
- Vision:
 - CNNs are dominantly used
- Prior work
 - Self-attention among pixels. Limit attention to local regions to reduce complexity (Parmar et al., 2018)
 - Self-attention among blocks of different sizes (Weissenborn et al., 2019)
 - Self attention between 2x2 patches (Cordonnier et al., 2020)

ViT: Motivation

- Prior work
 - Self attention between 2x2 patches (Cordonnier et al., 2020)

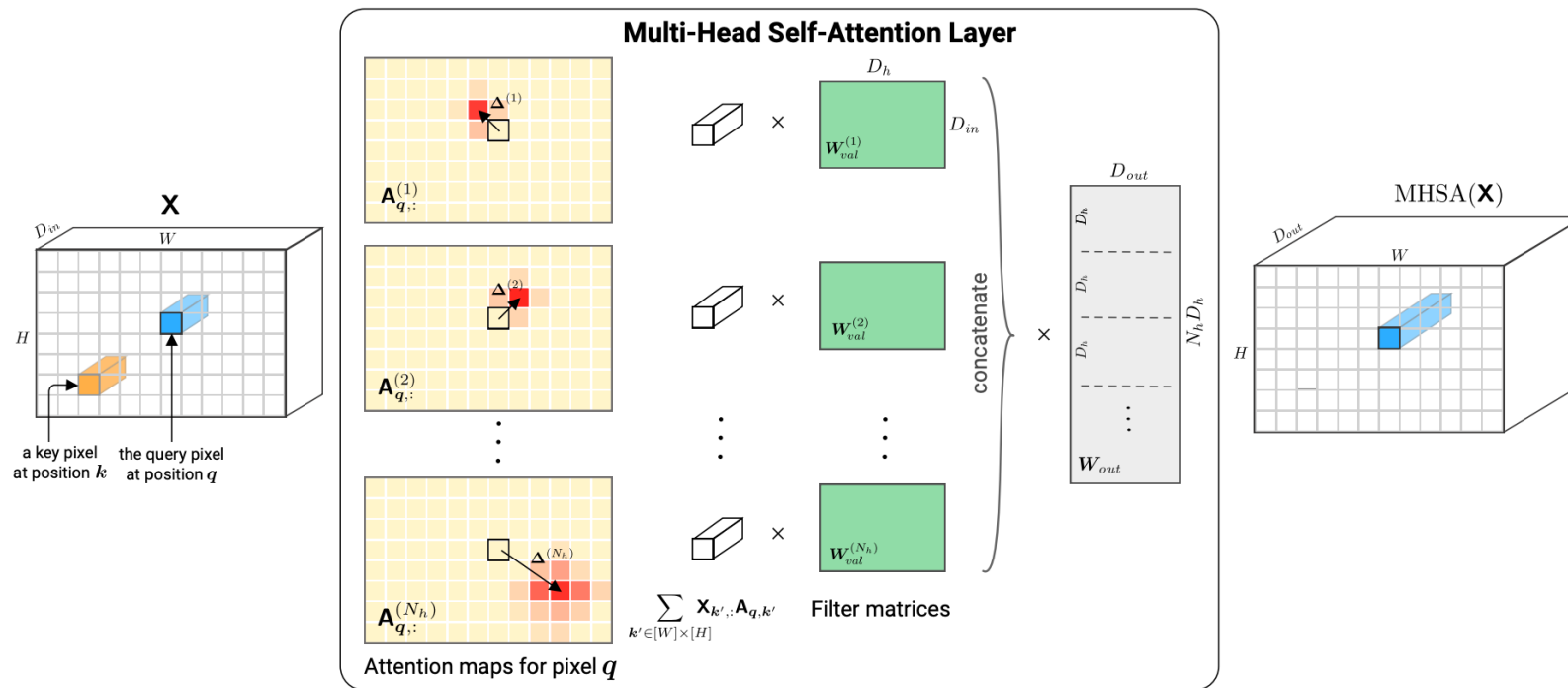


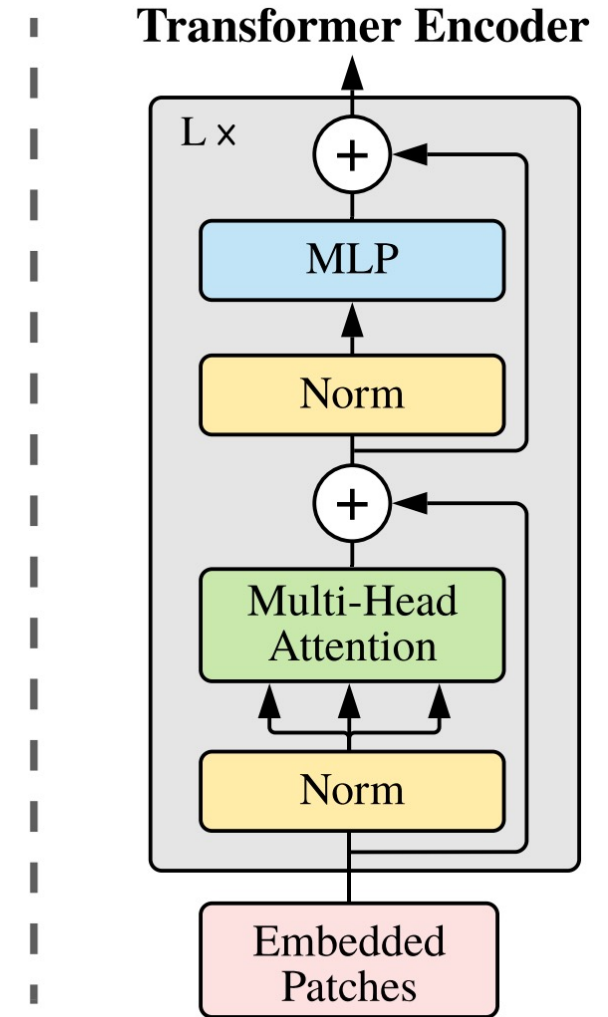
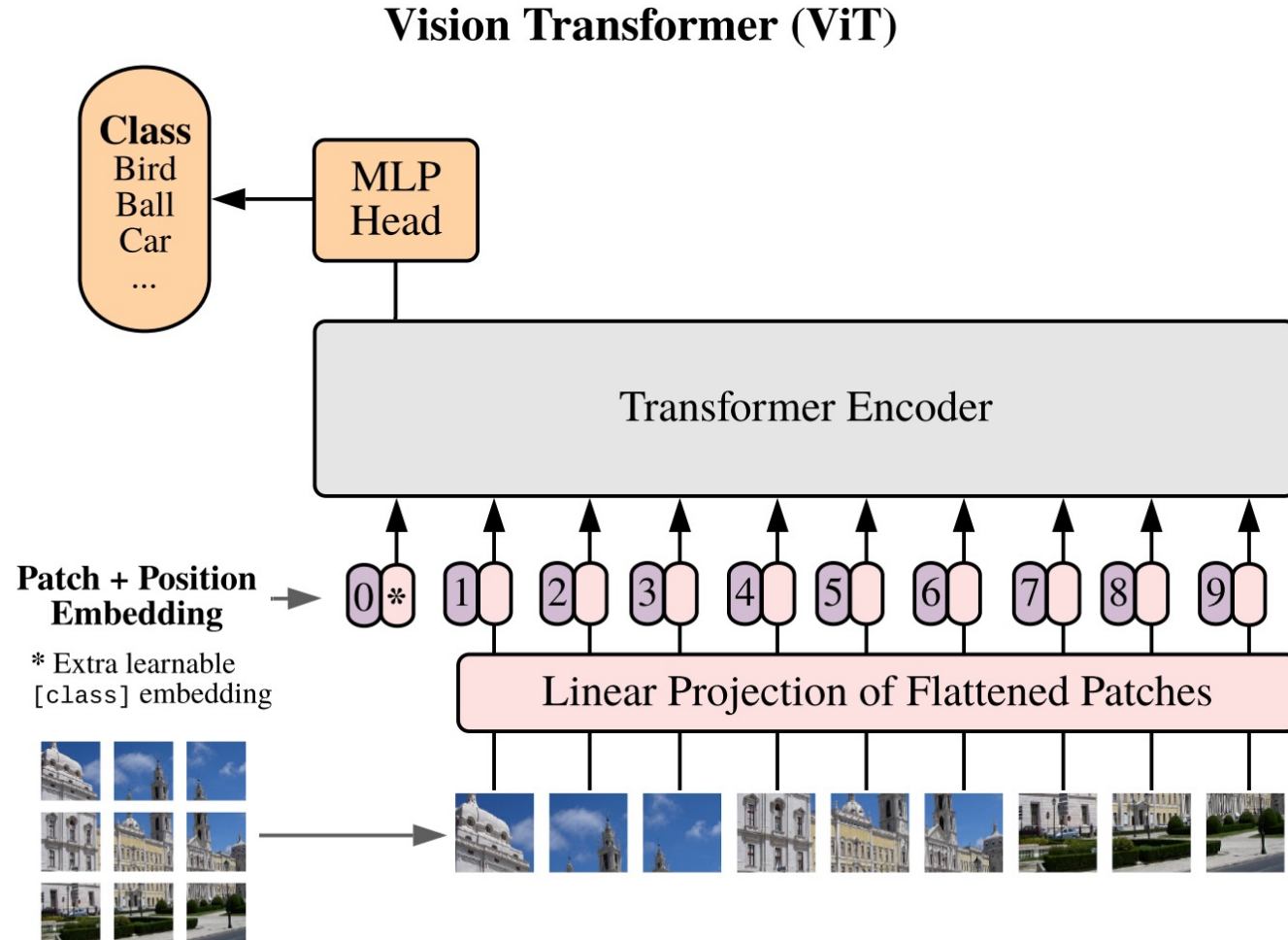
Figure 1: Illustration of a Multi-Head Self-Attention layer applied to a tensor image \mathbf{X} . Each head h attends pixel values around shift $\Delta^{(h)}$ and learn a filter matrix $\mathbf{W}_{val}^{(h)}$. We show attention maps computed for a query pixel at position q .

ViT: Motivation

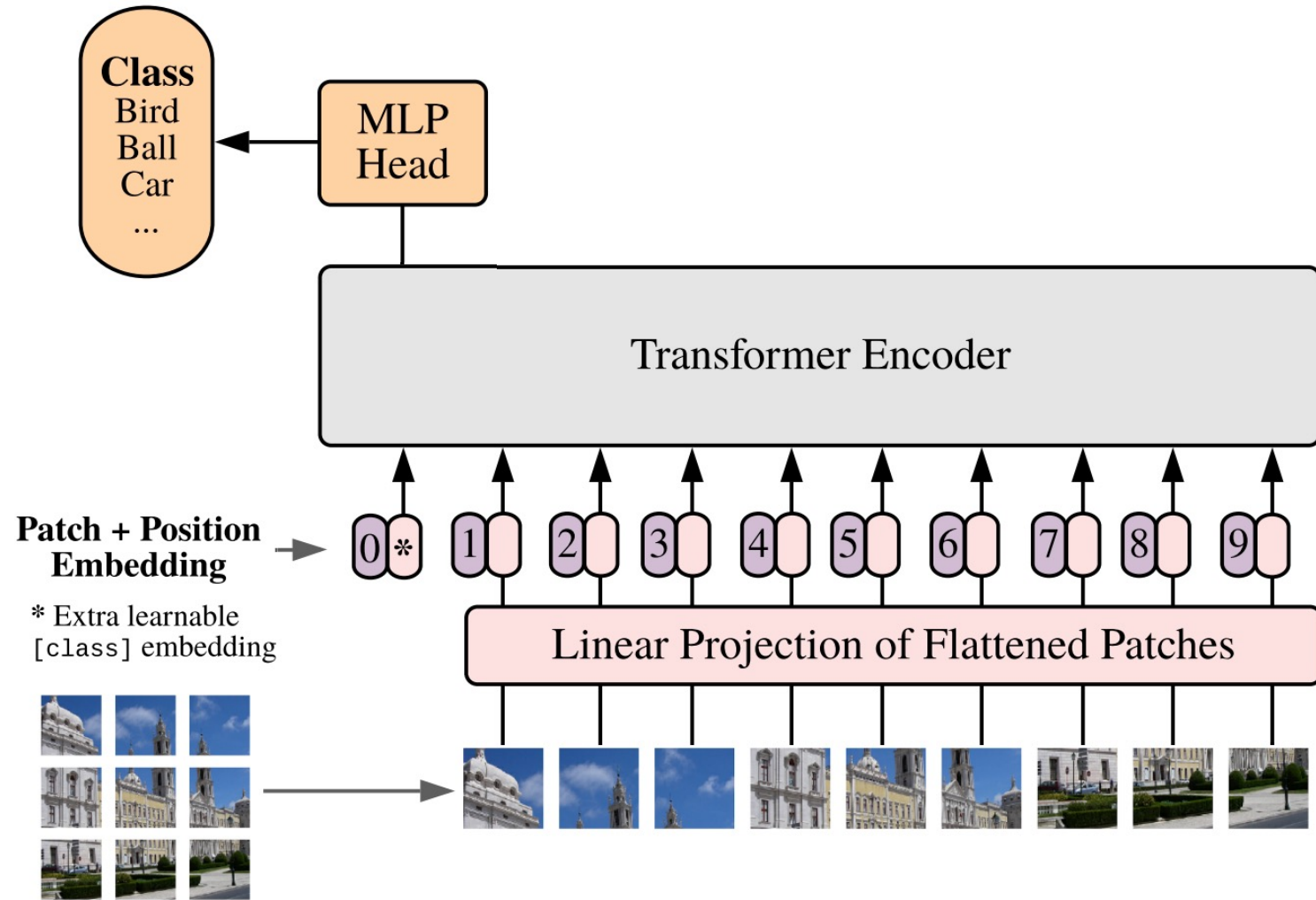
- Contributions
 - Use plain transformers without modifications
 - Compared to Cordonnier et al. (2020):
 - ViT shows strong results with large datasets & pretraining
 - ViT works with larger patches (generalizes better)
- Challenges in using Transformers with images:
 - How to convert an image into tokens?
 - How to encode position?
 - Transformers lack translation invariance and locality
 - Spoiler: They underperform on “small” datasets

Applied to classification tasks only!

ViT: Architecture



ViT: Architecture



ViT: Pretraining

- Supervised pretraining on large datasets and finetuning on smaller downstream tasks
- Higher resolutions:
 - The whole network can easily be applied to longer sequences (= higher resolutions)
 - Learned position embeddings are interpolated

ViT vs. CNNs

- “When trained on mid-sized datasets such as ImageNet without strong regularization, these models yield modest accuracies of a few percentage points below ResNets of comparable size.

This seemingly discouraging outcome may be expected: Transformers lack some of the inductive biases inherent to CNNs, such as translation equivariance and locality, and therefore do not generalize well when trained on insufficient amounts of data.”

- “However, the picture changes if the models are trained on larger datasets (14M-300M images). We find that large scale training trumps inductive bias.”

Locality

Definition: “pixels close to each other are highly correlated and form meaningful local structures (like edges, textures, or corners), while pixels far apart are less immediately related.” -- Gemini

- CNNs with small (local) receptive fields provide locality by nature.
- ViTs use self-attention, which is global: All patches attend to all other patches.

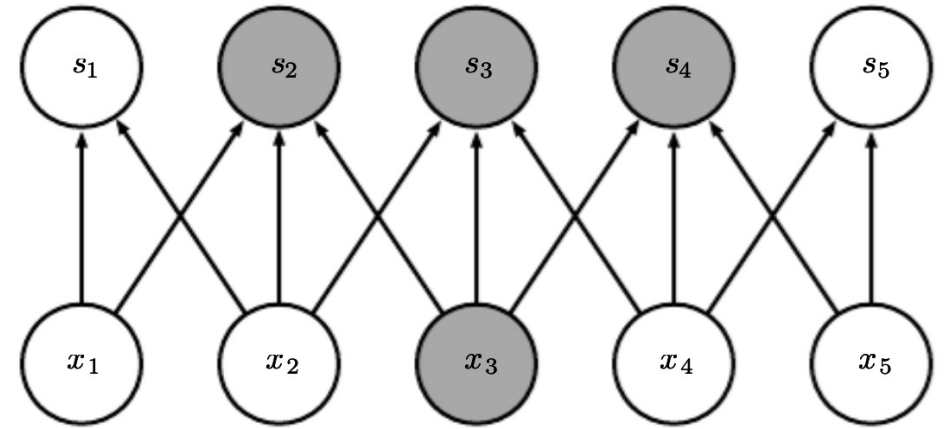


Figure: Goodfellow et al., “Deep Learning”, MIT Press, 2016.

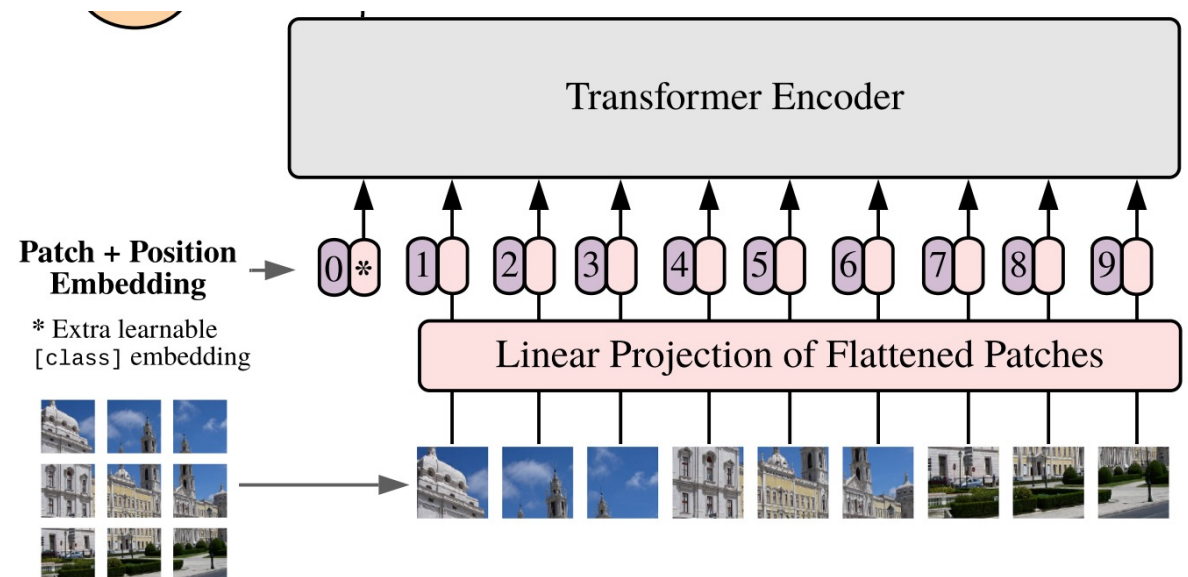
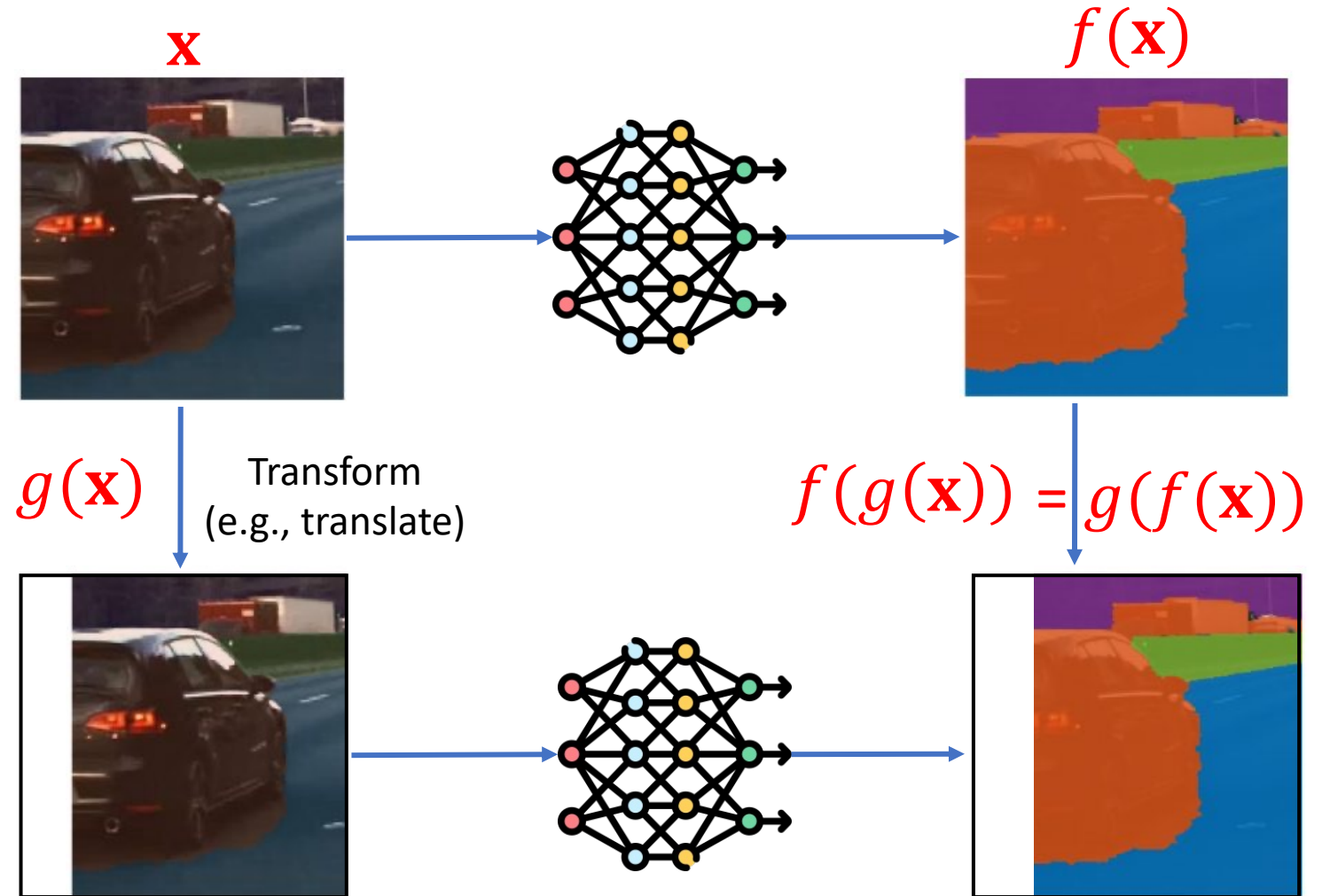


Figure: Dosovitskiy et al., “An Image is Worth 16x16 words...”, 2021.

Translation Equivariance

Definition: Transformation ($g(\cdot)$) applied on the input is transferred to the output:

$$f(g(\mathbf{x})) = g(f(\mathbf{x}))$$



Translation Equivariance in CNNs

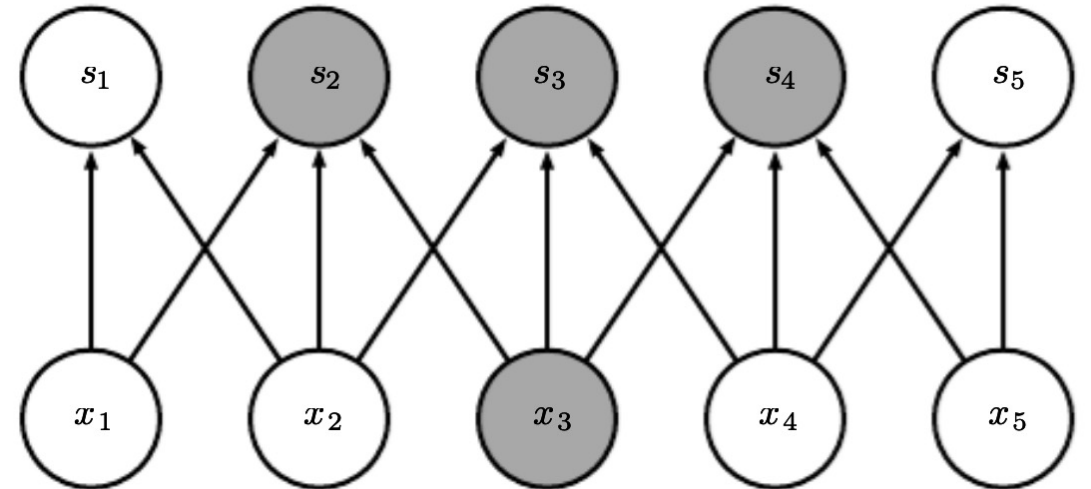
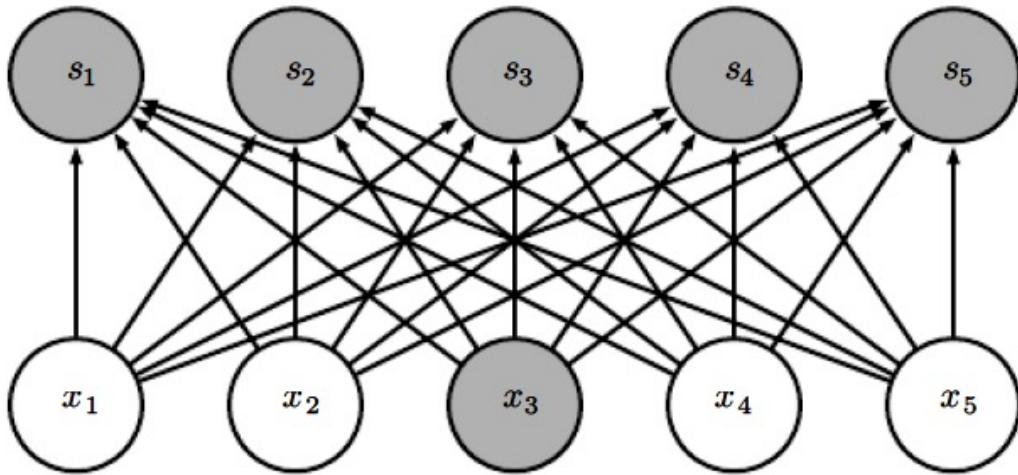


Figure: Goodfellow et al., "Deep Learning", MIT Press, 2016.

Translation Equivariance in ViTs

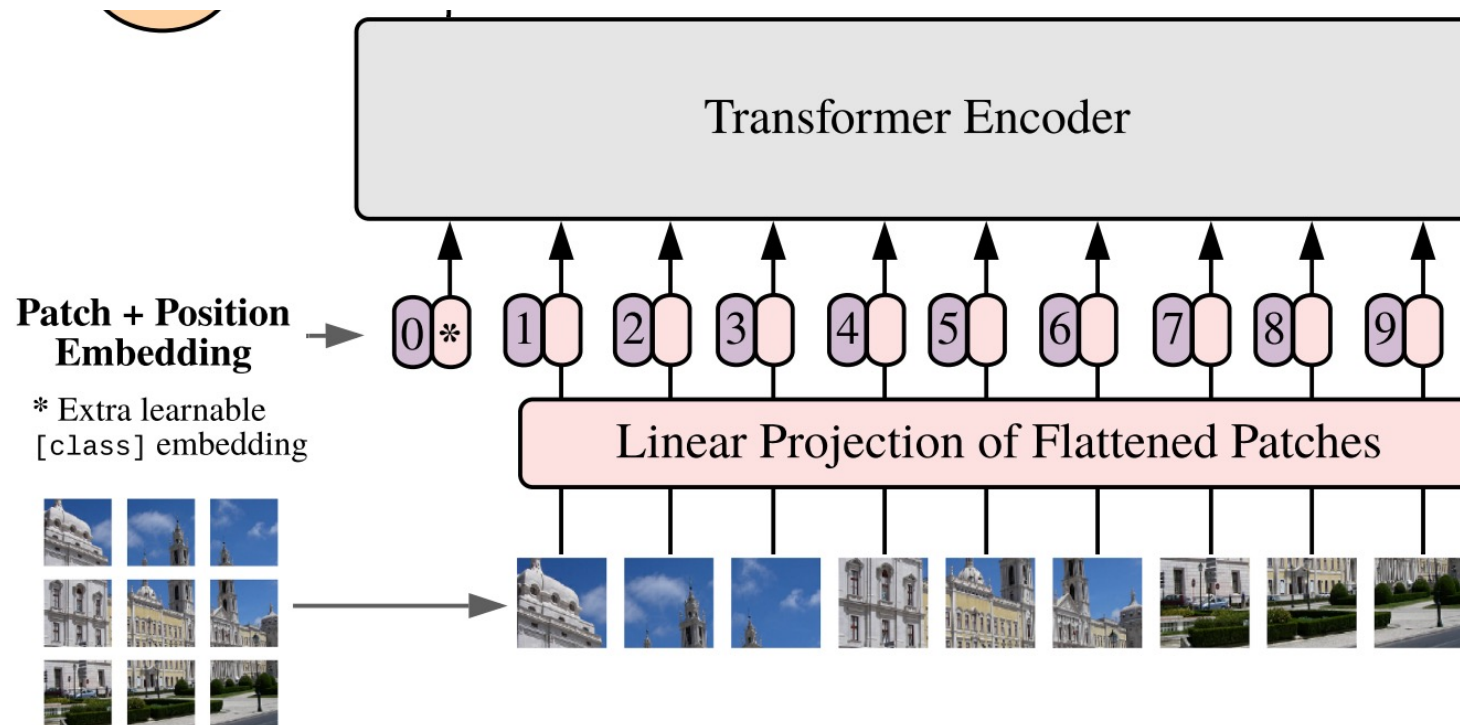
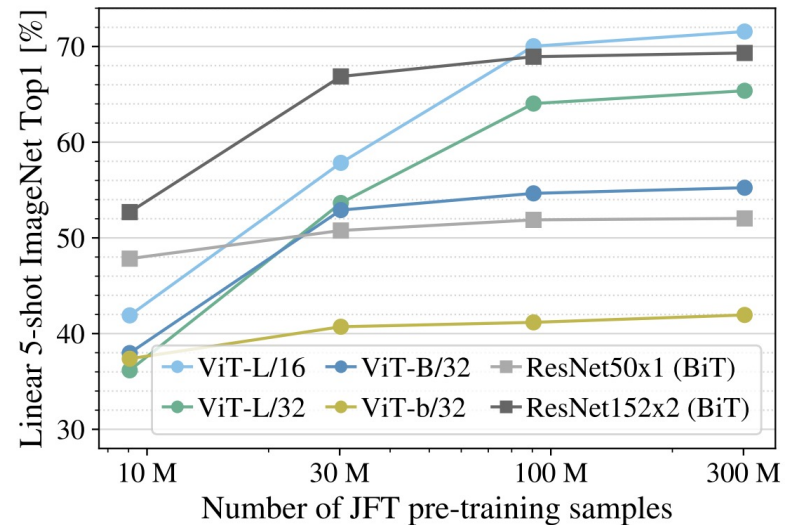
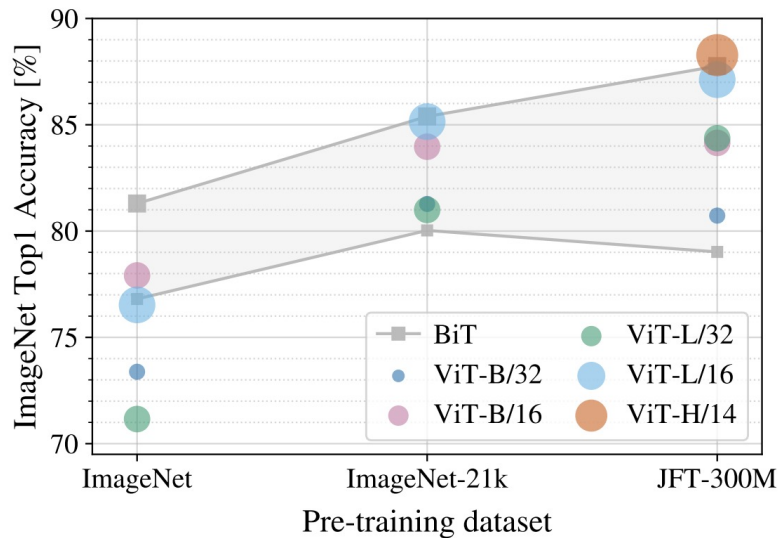


Figure: Dosovitskiy et al., "An Image is Worth 16x16 words...", 2021.

ViT: Results

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	88.55 ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	90.72 ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	99.50 ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	—
CIFAR-100	94.55 ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	—
Oxford-IIIT Pets	97.56 ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	—
Oxford Flowers-102	99.68 ± 0.02	99.74 ± 0.00	99.61 ± 0.02	99.63 ± 0.03	—
VTAB (19 tasks)	77.63 ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	—
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k



Swin Transformer: Hierarchical Vision Transformer using Shifted WindowsZe Liu^{1,2†*} Yutong Lin^{1,3†*} Yue Cao^{1*} Han Hu^{1*‡} Yixuan Wei^{1,4†}Zheng Zhang¹ Stephen Lin¹ Baining Guo¹¹Microsoft Research Asia ²University of Science and Technology of China³Xian Jiaotong University ⁴Tsinghua University

{v-zeliul, v-yutlin, yuecao, hanhu, v-yixwe, zhez, stevelin, bainguo}@microsoft.com

Swin Transformer V2: Scaling Up Capacity and ResolutionZe Liu^{2*} Han Hu^{1*†} Yutong Lin³ Zhuliang Yao⁴ Zhenda Xie⁴ Yixuan Wei⁴ Jia Ning⁵Yue Cao¹ Zheng Zhang¹ Li Dong¹ Furu Wei¹ Baining Guo¹¹Microsoft Research Asia ²University of Science and Technology of China³Xian Jiaotong University ⁴Tsinghua University ⁵Huazhong University of Science and Technology

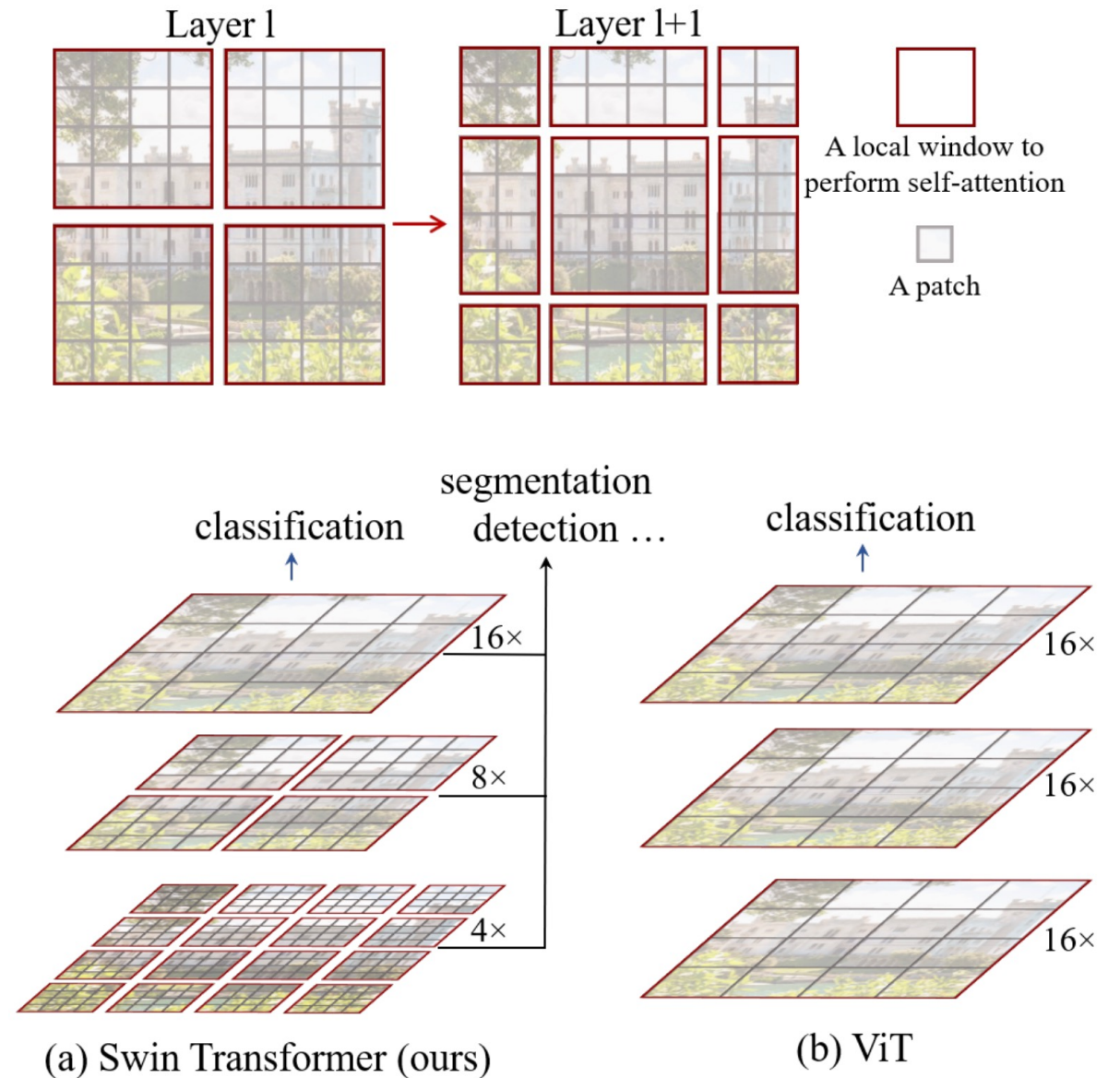
{t-liuze, hanhu, t-yutonglin, t-zhuyao, t-zhxie, t-yixuanwei, v-jianing}@microsoft.com

{yuecao, zhez, lidong1, fuwei, bainguo}@microsoft.com

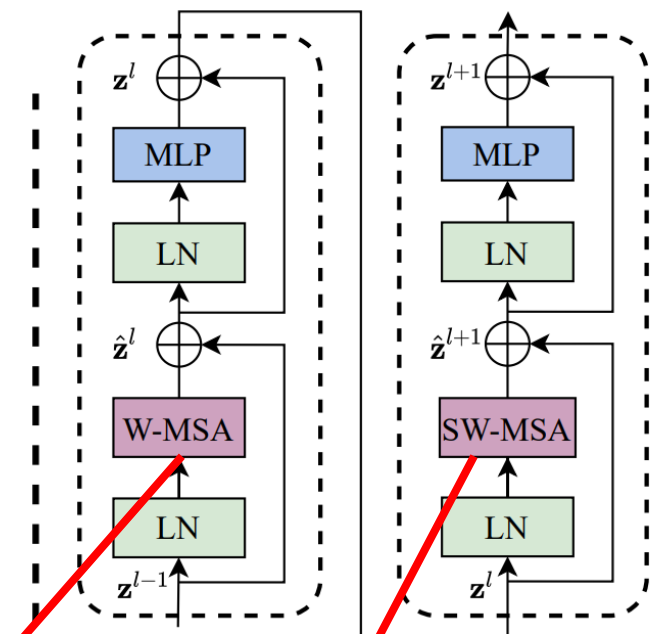
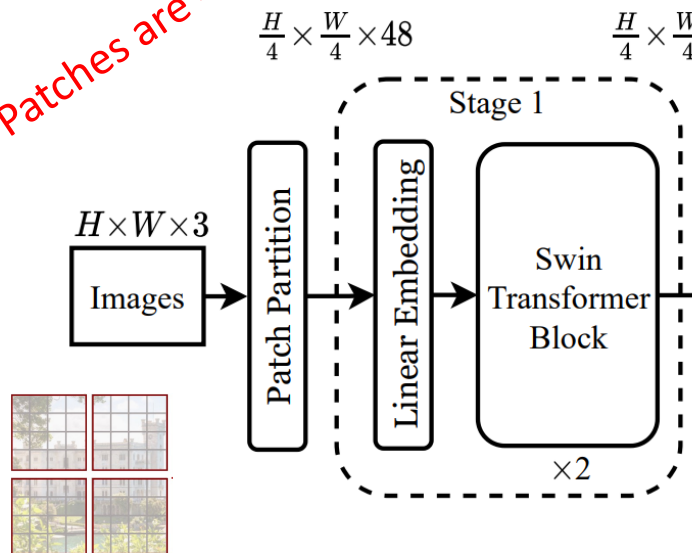
Swin Transformers

Swin Transformer v1

- Motivation:
 - ViT is promising but limited to classification
 - Challenges in using Transformers:
 - large variations in scales of visual entities,
 - more pixels compared to words in text
 - Existing Transformers use fixed token size across layers
- Contributions:
 - Limit self-attention to non-overlapping windows while allowing cross-window attention
 - Change token size across layers

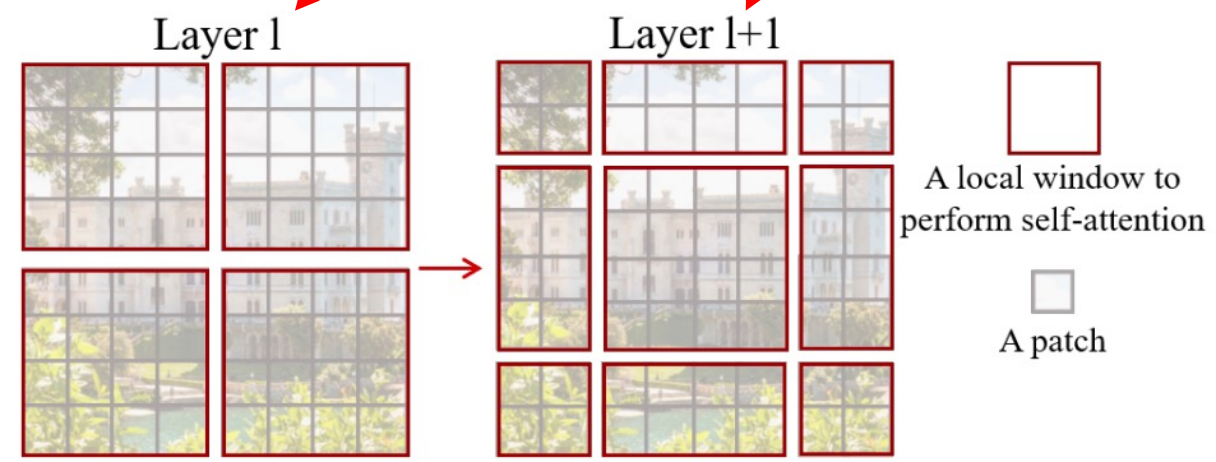


Patches are 4x4!



Two Successive Swin Transformer Blocks

Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.



Patches are 4x4!

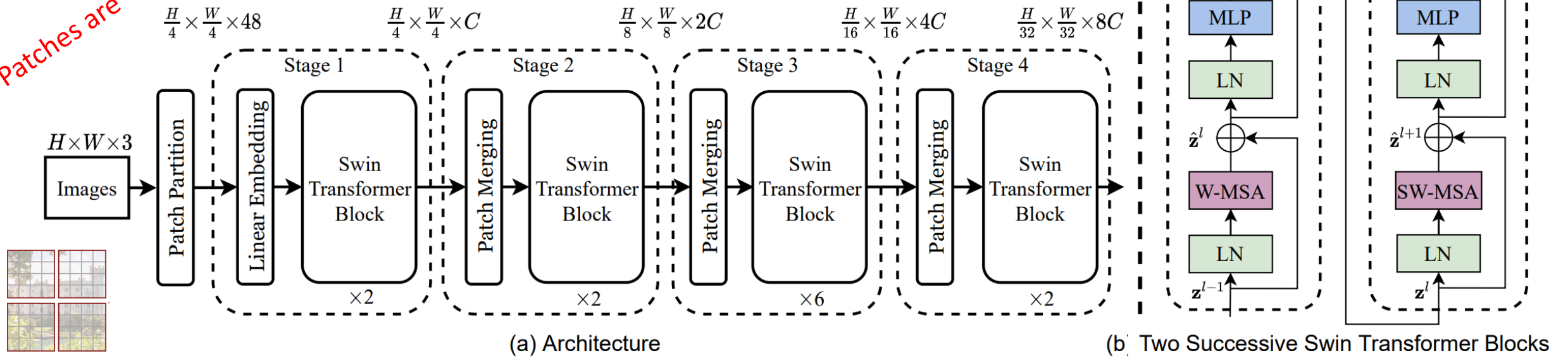
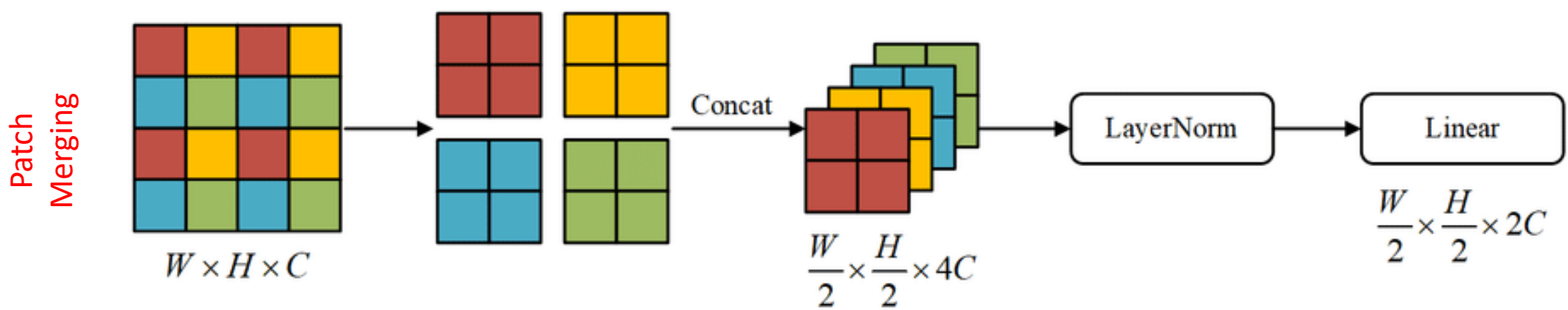


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.



1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

A: Attention in local windows

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

B: Attention in shifted windows (2,2)

19	20	21	22	23	24	17	18
27	28	29	30	31	32	25	26
35	36	37	38	39	40	33	34
43	44	45	46	47	48	41	42
51	52	53	54	55	56	49	50
59	60	61	62	63	64	57	58
3	4	5	6	7	8	1	2
11	12	13	14	15	16	9	10

C: Batched windows after cyclic shift

The corner 2x2 corner patches (B) are stacked together in a single window (C) but they do not attend to each other to preserve locality.

Figure: <https://amaarora.github.io/posts/2022-07-04-swintransformerv1.html>

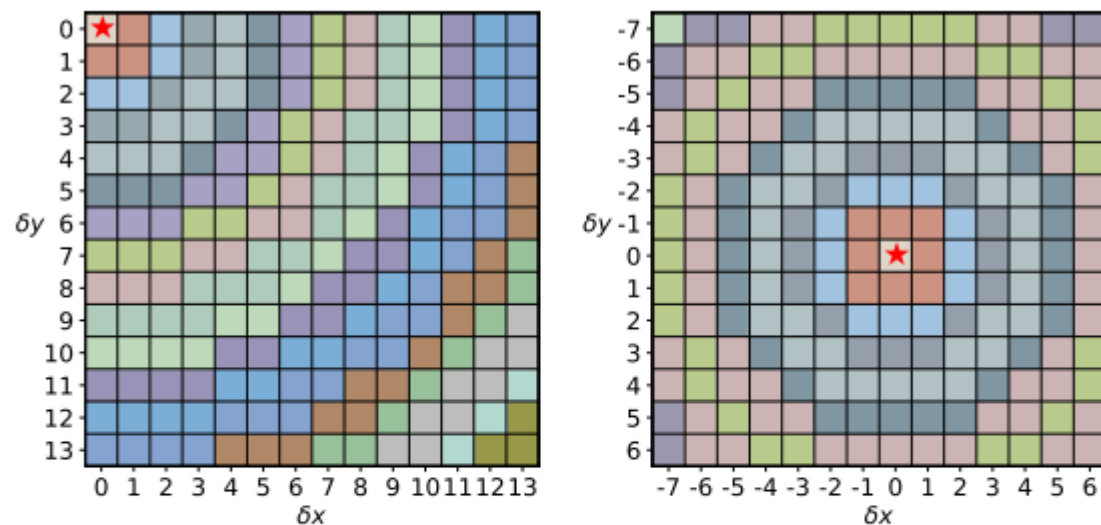
Swin Transformer v1

Relative position bias In computing self-attention, we follow [49, 1, 32, 33] by including a relative position bias $B \in \mathbb{R}^{M^2 \times M^2}$ to each head in computing similarity:

$$\text{Attention}(Q, K, V) = \text{SoftMax}(QK^T / \sqrt{d} + B)V, \quad (4)$$

where $Q, K, V \in \mathbb{R}^{M^2 \times d}$ are the *query*, *key* and *value* matrices; d is the *query/key* dimension, and M^2 is the number of patches in a window. Since the relative position along each axis lies in the range $[-M + 1, M - 1]$, we parameterize a smaller-sized bias matrix $\hat{B} \in \mathbb{R}^{(2M-1) \times (2M-1)}$, and values in B are taken from \hat{B} .

Plot from <https://arxiv.org/pdf/2107.14222>:



(a) top-left

(b) center

Figure 6: Visualization of Euclidean method. The red star \star presents the reference position. Different color means different bucket. The relative positions with the same color share the same encoding.

Swin Transformer v1

Relative position bias In computing self-attention, we follow [49, 1, 32, 33] by including a relative position bias $B \in \mathbb{R}^{M^2 \times M^2}$ to each head in computing similarity:

$$\text{Attention}(Q, K, V) = \text{SoftMax}(QK^T / \sqrt{d} + B)V, \quad (4)$$

where $Q, K, V \in \mathbb{R}^{M^2 \times d}$ are the *query*, *key* and *value* matrices; d is the *query/key* dimension, and M^2 is the number of patches in a window. Since the relative position along each axis lies in the range $[-M + 1, M - 1]$, we parameterize a smaller-sized bias matrix $\hat{B} \in \mathbb{R}^{(2M-1) \times (2M-1)}$, and values in B are taken from \hat{B} .

	ImageNet		COCO		ADE20k
	top-1	top-5	AP ^{box}	AP ^{mask}	mIoU
w/o shifting	80.2	95.1	47.7	41.5	43.3
shifted windows	81.3	95.6	50.5	43.7	46.1
no pos.	80.1	94.9	49.2	42.6	43.8
abs. pos.	80.5	95.2	49.0	42.4	43.2
abs.+rel. pos.	81.3	95.6	50.2	43.4	44.0
rel. pos. w/o app.	79.3	94.7	48.2	41.9	44.1
rel. pos.	81.3	95.6	50.5	43.7	46.1

Table 4. Ablation study on the *shifted windows* approach and different position embedding methods on three benchmarks, using the Swin-T architecture. w/o shifting: all self-attention modules adopt regular window partitioning, without *shifting*; abs. pos.: absolute position embedding term of ViT; rel. pos.: the default settings with an additional relative position bias term (see Eq. (4)); app.: the first scaled dot-product term in Eq. (4).

Swin Transformer v1: Results

(a) Regular ImageNet-1K trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [44]	224 ²	21M	4.0G	1156.7	80.0
RegNetY-8G [44]	224 ²	39M	8.0G	591.6	81.7
RegNetY-16G [44]	224 ²	84M	16.0G	334.7	82.9
ViT-B/16 [19]	384 ²	86M	55.4G	85.9	77.9
ViT-L/16 [19]	384 ²	307M	190.7G	27.3	76.5
DeiT-S [57]	224 ²	22M	4.6G	940.4	79.8
DeiT-B [57]	224 ²	86M	17.5G	292.3	81.8
DeiT-B [57]	384 ²	86M	55.4G	85.9	83.1
Swin-T	224 ²	29M	4.5G	755.2	81.3
Swin-S	224 ²	50M	8.7G	436.9	83.0
Swin-B	224 ²	88M	15.4G	278.1	83.5
Swin-B	384 ²	88M	47.0G	84.7	84.5

(b) ImageNet-22K pre-trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [34]	384 ²	388M	204.6G	-	84.4
R-152x4 [34]	480 ²	937M	840.5G	-	85.4
ViT-B/16 [19]	384 ²	86M	55.4G	85.9	84.0
ViT-L/16 [19]	384 ²	307M	190.7G	27.3	85.2
Swin-B	224 ²	88M	15.4G	278.1	85.2
Swin-B	384 ²	88M	47.0G	84.7	86.4
Swin-L	384 ²	197M	103.9G	42.1	87.3

Table 1. Comparison of different backbones on ImageNet-1K classification. Throughput is measured using the GitHub repository of [62] and a V100 GPU, following [57].

(a) Various frameworks							
Method	Backbone	AP ^{box}	AP ₅₀ ^{box}	AP ₇₅ ^{box}	#param.	FLOPs	FPS
Cascade	R-50	46.3	64.3	50.5	82M	739G	18.0
Mask R-CNN	Swin-T	50.5	69.3	54.9	86M	745G	15.3
ATSS	R-50	43.5	61.9	47.0	32M	205G	28.3
	Swin-T	47.2	66.5	51.3	36M	215G	22.3
RepPointsV2	R-50	46.5	64.6	50.3	42M	274G	13.6
	Swin-T	50.0	68.5	54.2	45M	283G	12.0
Sparse R-CNN	R-50	44.5	63.4	48.2	106M	166G	21.0
	Swin-T	47.9	67.3	52.3	110M	172G	18.4

(b) Various backbones w. Cascade Mask R-CNN									
	AP ^{box}	AP ₅₀ ^{box}	AP ₇₅ ^{box}	AP ^{mask}	AP ₅₀ ^{mask}	AP ₇₅ ^{mask}	param	FLOPs	FPS
DeiT-S [†]	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	50.5	69.3	54.9	43.7	66.6	47.1	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	51.8	70.4	56.3	44.7	67.9	48.5	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	51.9	70.9	56.5	45.0	68.4	48.7	145M	982G	11.6

(c) System-level Comparison						
Method	mini-val		test-dev		#param.	FLOPs
	AP ^{box}	AP ^{mask}	AP ^{box}	AP ^{mask}		
RepPointsV2* [12]	-	-	52.1	-	-	-
GCNet* [7]	51.8	44.7	52.3	45.4	-	1041G
RelationNet++* [13]	-	-	52.7	-	-	-
DetectoRS* [42]	-	-	55.7	48.5	-	-
YOLOv4 P7* [4]	-	-	55.8	-	-	-
Copy-paste [23]	55.9	47.2	56.0	47.4	185M	1440G
X101-64 (HTC++)	52.3	46.0	-	-	155M	1033G
Swin-B (HTC++)	56.4	49.1	-	-	160M	1043G
Swin-L (HTC++)	57.1	49.5	57.7	50.2	284M	1470G
Swin-L (HTC++)*	58.0	50.4	58.7	51.1	284M	-

Table 2. Results on COCO object detection and instance segmentation. [†] denotes that additional deconvolution layers are used to produce hierarchical feature maps. * indicates multi-scale testing.

Swin Transformer v2

Motivation:

- “Large-scale NLP models have been shown to significantly improve the performance on language tasks with no signs of saturation”.
- “They also demonstrate amazing few-shot capabilities like that of human beings”

Contribution:

- “Explore large-scale models in computer vision”
- Three major issues & solutions:
 - **Training stability**: “1) a residual-post-norm method combined with cosine attention to improve training stability”
 - **resolution gaps between pre-training and fine-tuning**: “2) A log-spaced continuous position bias method to effectively transfer models pre-trained using low-resolution images to downstream tasks with high-resolution inputs”
 - **hunger on labelled data**: “3) A self-supervised pretraining method, SimMIM, to reduce the needs of vast labeled images”

Swin Transformer v2

- Issues with v1
 - “An instability issue when scaling up model capacity.”
 - “Degraded performance when transferring models across window resolutions.”

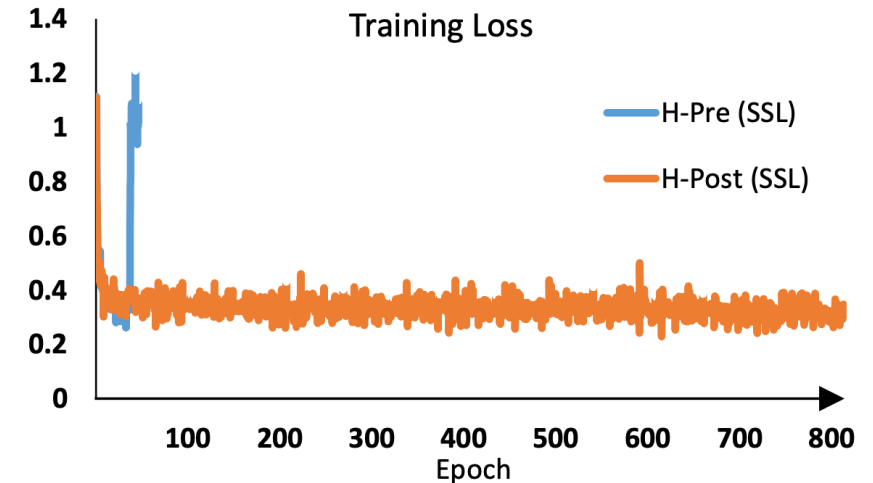


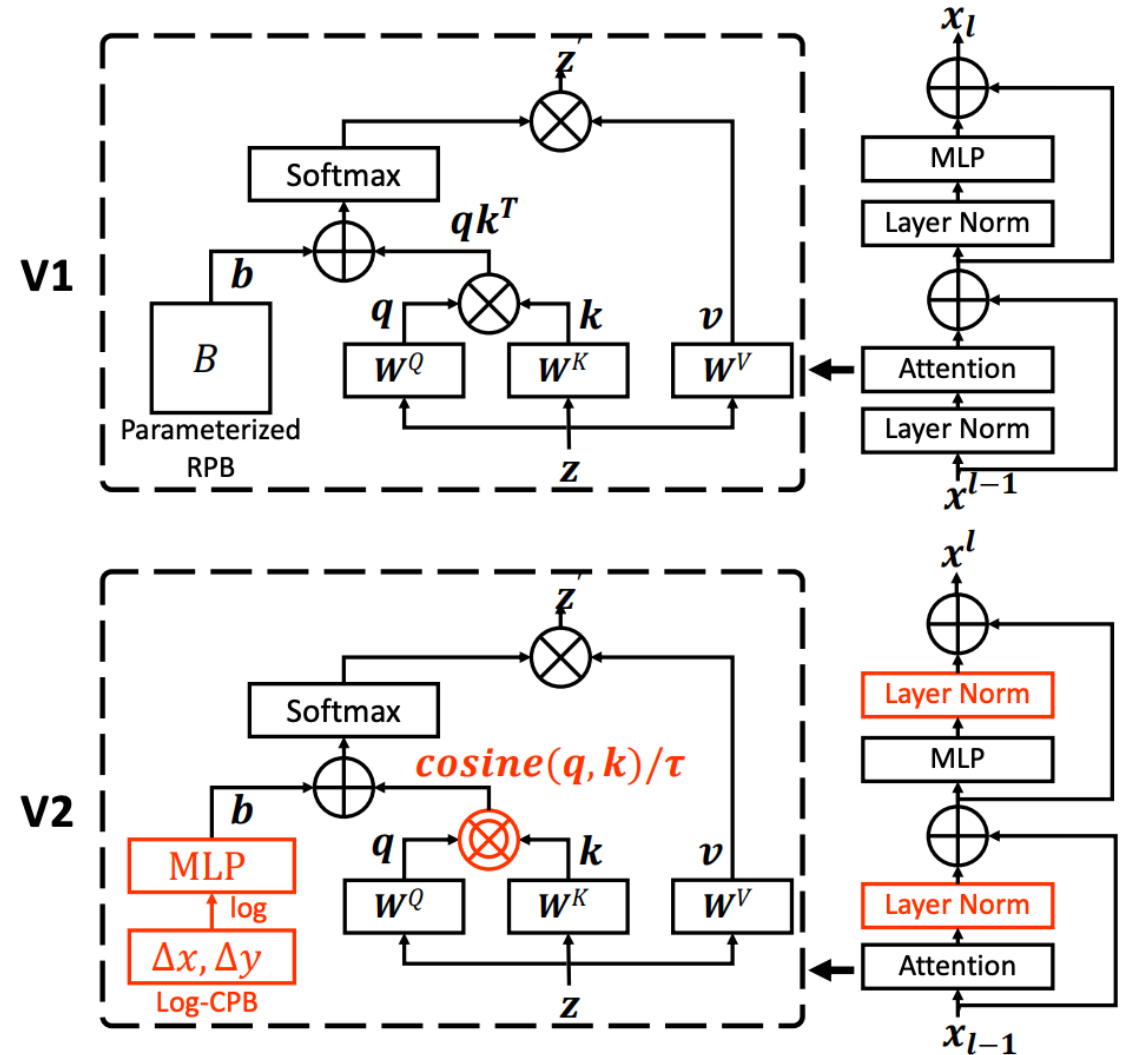
Figure 3. SwinV1-H versus SwinV2-H in training [72].

method	Trained on		Tested/Finetuned on				Finetuned on				
	ImageNet*	ImageNet [†]					COCO		ADE20k		
	W8, I256 top-1 acc	W12, I384 top-1 acc	W16, I512 top-1 acc	W20, I640 top-1 acc	W24, I768 top-1 acc	W16 AP ^{box}	W32 AP ^{box}	W16 mIoU	W20 mIoU	W32 mIoU	
Parameterized position bias [46]	81.7	79.4/82.7	77.2/83.0	73.2/83.2	68.7/83.2	50.8	50.9	45.5	45.8	44.5	
Linear-Spaced CPB	81.7 (+0.0)	82.0/82.9 (+2.6/+0.2)	81.2/83.3 (+4.0/+0.3)	79.8/83.6 (+6.6/+0.4)	77.6/83.7 (+8.9/+0.5)	50.9 (+0.1)	51.7 (+0.8)	47.0 (+1.5)	47.4 (+1.6)	47.2 (+2.7)	
Log-Spaced CPB	81.8 (+0.1)	82.4/83.2 (+3.0/+0.5)	81.7/83.8 (+4.5/+0.8)	80.4/84.0 (+7.2/+0.8)	79.1/84.2 (+10.4/+1.0)	51.1 (+0.3)	51.8 (+0.9)	47.0 (+1.5)	47.7 (+1.9)	47.8 (+3.3)	

Swin Transformer v2

Scaling up model capacity

- Residual post normalization:
Prevents output to diverge
- Scaled cosine attention:
 - learnt attention maps of some blocks and heads are frequently dominated by a few pixel pairs, especially in the res-post-norm configuration => cosine yields values in a smaller range



Scaling Up Window Resolution

- Continuous relative position bias

$$B(\Delta x, \Delta y) = \mathcal{G}(\Delta x, \Delta y), \quad (3)$$

where \mathcal{G} is a small network, e.g., a 2-layer MLP with a ReLU activation in between by default.

The meta network \mathcal{G} generates bias values for arbitrary relative coordinates, and thus can be naturally transferred to fine-tuning tasks with arbitrarily varying window sizes.

Log-spaced coordinates

Log-spaced coordinates When transferring across largely varying window sizes, a large portion of the relative coordinate range needs to be extrapolated. To ease this issue, we propose using log-spaced coordinates instead of the original linear-spaced ones:

$$\begin{aligned} \widehat{\Delta x} &= \text{sign}(x) \cdot \log(1 + |\Delta x|), \\ \widehat{\Delta y} &= \text{sign}(y) \cdot \log(1 + |\Delta y|), \end{aligned} \quad (4)$$

where Δx , Δy and $\widehat{\Delta x}$, $\widehat{\Delta y}$ are the linear-scaled and log-spaced coordinates, respectively.

method	Trained on	Tested/Finetuned on				Finetuned on				
	ImageNet*	ImageNet [†]				COCO		ADE20k		
	W8, I256 top-1 acc	W12, I384 top-1 acc	W16, I512 top-1 acc	W20, I640 top-1 acc	W24, I768 top-1 acc	W16 AP ^{box}	W32 AP ^{box}	W16 mIoU	W20 mIoU	W32 mIoU
Parameterized position bias [46]	81.7	79.4/82.7	77.2/83.0	73.2/83.2	68.7/83.2	50.8	50.9	45.5	45.8	44.5
Linear-Spaced CPB	81.7 (+0.0)	82.0/82.9 (+2.6/+0.2)	81.2/83.3 (+4.0/+0.3)	79.8/83.6 (+6.6/+0.4)	77.6/83.7 (+8.9/+0.5)	50.9 (+0.1)	51.7 (+0.8)	47.0 (+1.5)	47.4 (+1.6)	47.2 (+2.7)
Log-Spaced CPB	81.8 (+0.1)	82.4/83.2 (+3.0/+0.5)	81.7/83.8 (+4.5/+0.8)	80.4/84.0 (+7.2/+0.8)	79.1/84.2 (+10.4/+1.0)	51.1 (+0.3)	51.8 (+0.9)	47.0 (+1.5)	47.7 (+1.9)	47.8 (+3.3)

Swin Transformer v2

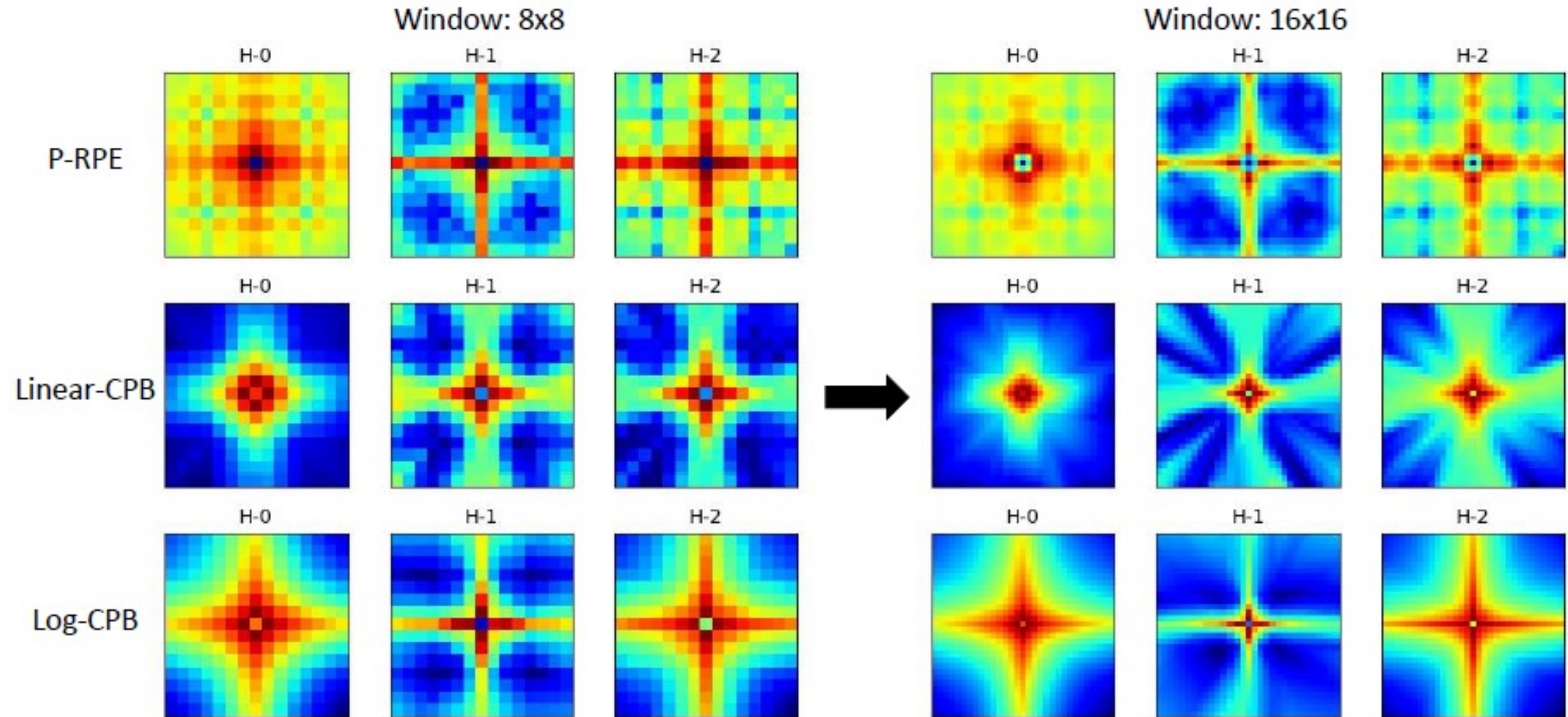


Fig: <https://sh-tsang.medium.com/review-swin-transformer-v2-scaling-up-capacity-and-resolution-401c28b02df8>

Swin Transformer v2

Self-Supervised Pre-training

- Prior work used JFT-3B
- Use SimMIM to train 3B-size Swin Transformer

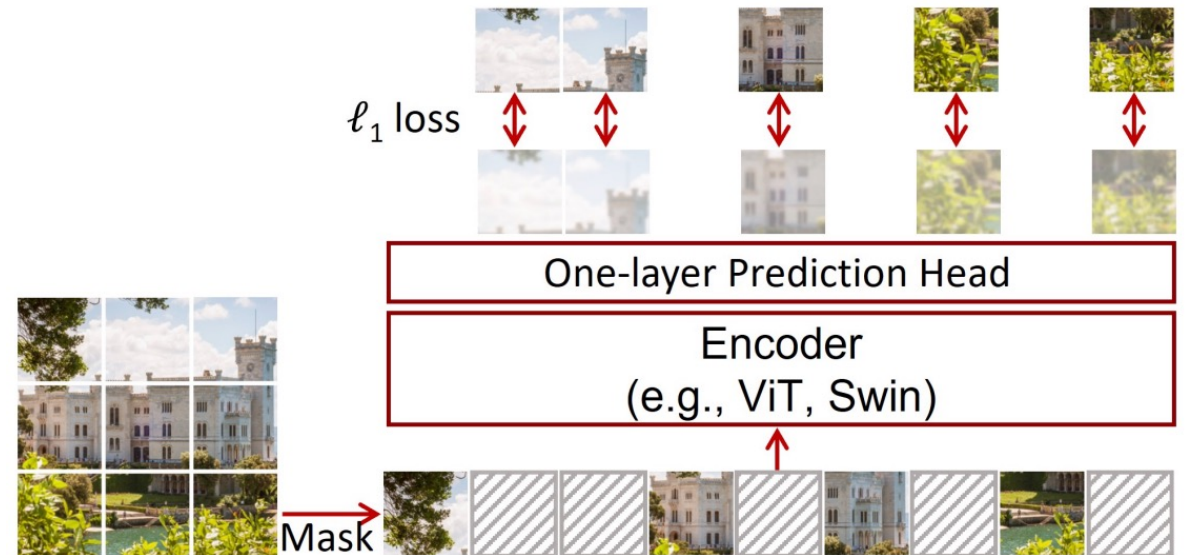


Figure 1. An illustration of our simple framework for masked language modeling, named *SimMIM*. It predicts raw pixel values of the randomly masked patches by a lightweight one-layer head, and performs learning using a simple ℓ_1 loss.

From SimMIM paper.

Method	param	pre-train images	pre-train length (#im)	pre-train im size	pre-train time	fine-tune im size	ImageNet-1K-V1 top-1 acc	ImageNet-1K-V2 top-1 acc
SwinV1-B	88M	IN-22K-14M	1.3B	224 ²	<30 [†]	384 ²	86.4	76.58
SwinV1-L	197M	IN-22K-14M	1.3B	224 ²	<10 [†]	384 ²	87.3	77.46
ViT-G [80]	1.8B	JFT-3B	164B	224 ²	>30k	518 ²	90.45	83.33
V-MoE [56]	14.7B*	JFT-3B	-	224 ²	16.8k	518 ²	90.35	-
CoAtNet-7 [17]	2.44B	JFT-3B	-	224 ²	20.1k	512 ²	90.88	-
SwinV2-B	88M	IN-22K-14M	1.3B	192 ²	<30 [†]	384 ²	87.1	78.08
SwinV2-L	197M	IN-22K-14M	1.3B	192 ²	<20 [†]	384 ²	87.7	78.31
SwinV2-G	3.0B	IN-22K-ext-70M	3.5B	192 ²	<0.5k [†]	640 ²	90.17	84.00

Table 2. Comparison with previous largest vision models on ImageNet-1K V1 and V2 classification. * indicates the sparse model; the “pre-train time” column is measured by the TPUv3 core days with numbers copied from the original papers. † That of SwinV2-G is estimated according to training iterations and FLOPs.

Method	train I(W) size	test I(W) size	views	top-1
ViViT [2]	-(-)	-(-)	4×3	84.8
SwinV1-L [47]	480(12) ² ×16(8)	480(12) ² ×16(8)	10×5	84.9
TokenLearner [57]	256(8) ² ×64(64)	256(8) ² ×64(64)	4×3	85.4
Video-SwinV2-G		320(20) ² ×8(8)	1×1	83.2
	320(20) ² ×8(8)	384(24) ² ×8(8)	1×1	83.4
		384(24) ² ×8(8)	4×5	86.8

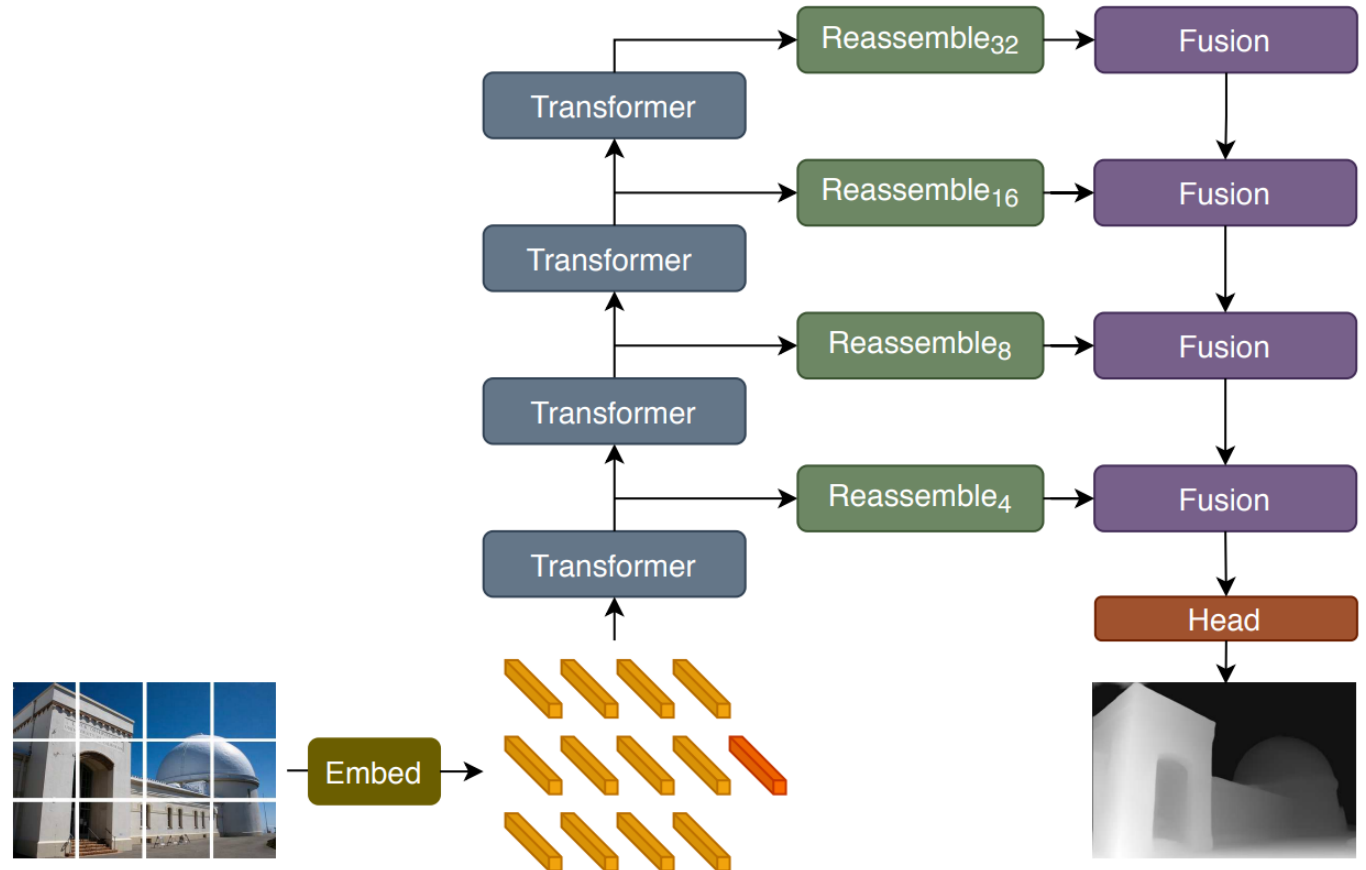
Table 5. Comparison with previous best results on Kinetics-400 video action classification.

Dense Prediction Transformers (DPT)

Ranftl et al., “Vision Transformers for Dense Prediction”, 2021.

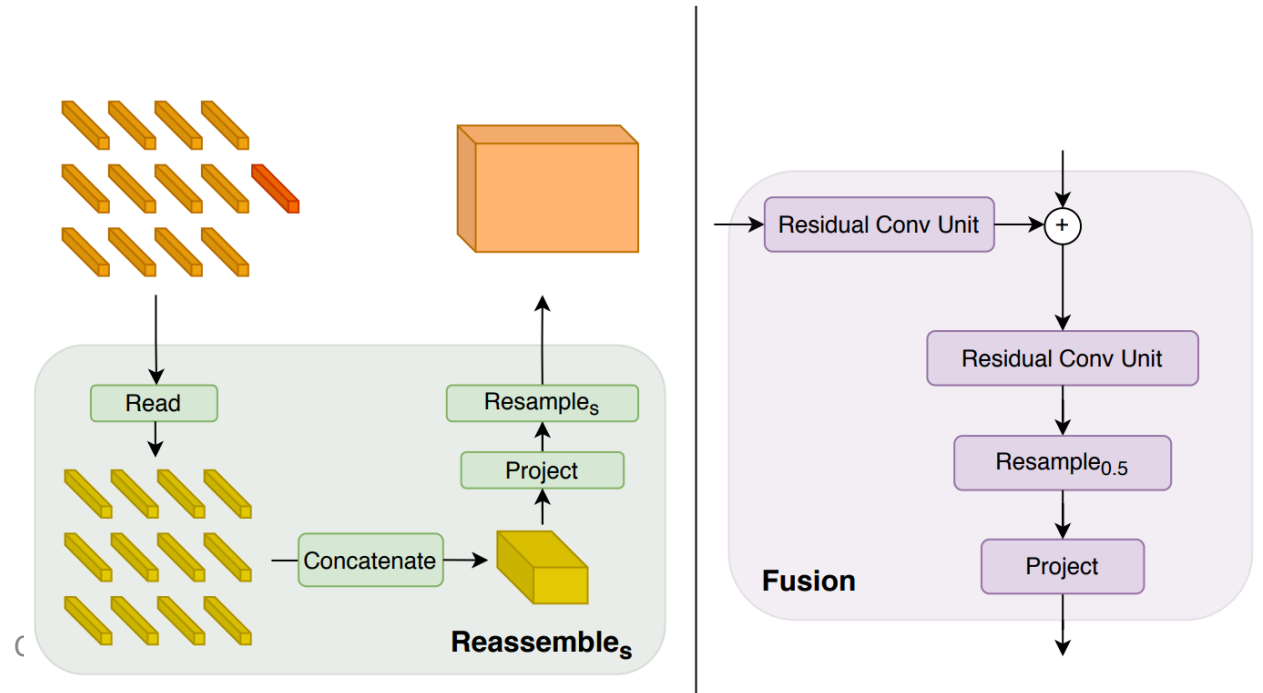
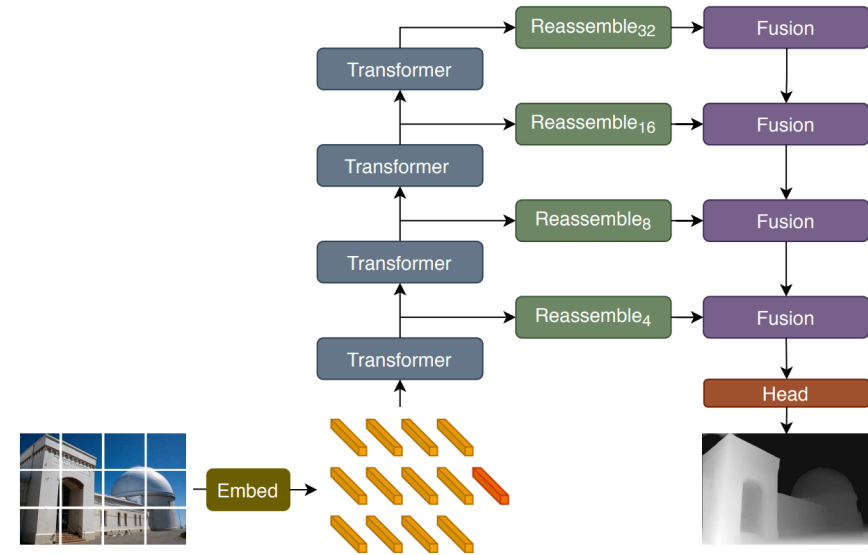
DPT

- Extend a ViT for dense-prediction tasks
- This requires forming a feature hierarchy



DPT

- Extend a ViT for dense-prediction tasks
- Each Reassemble block is a light-weight module that combines tokens and resamples them (1x1 conv)
- Fusion combines features from different scales



ViT-Adapter

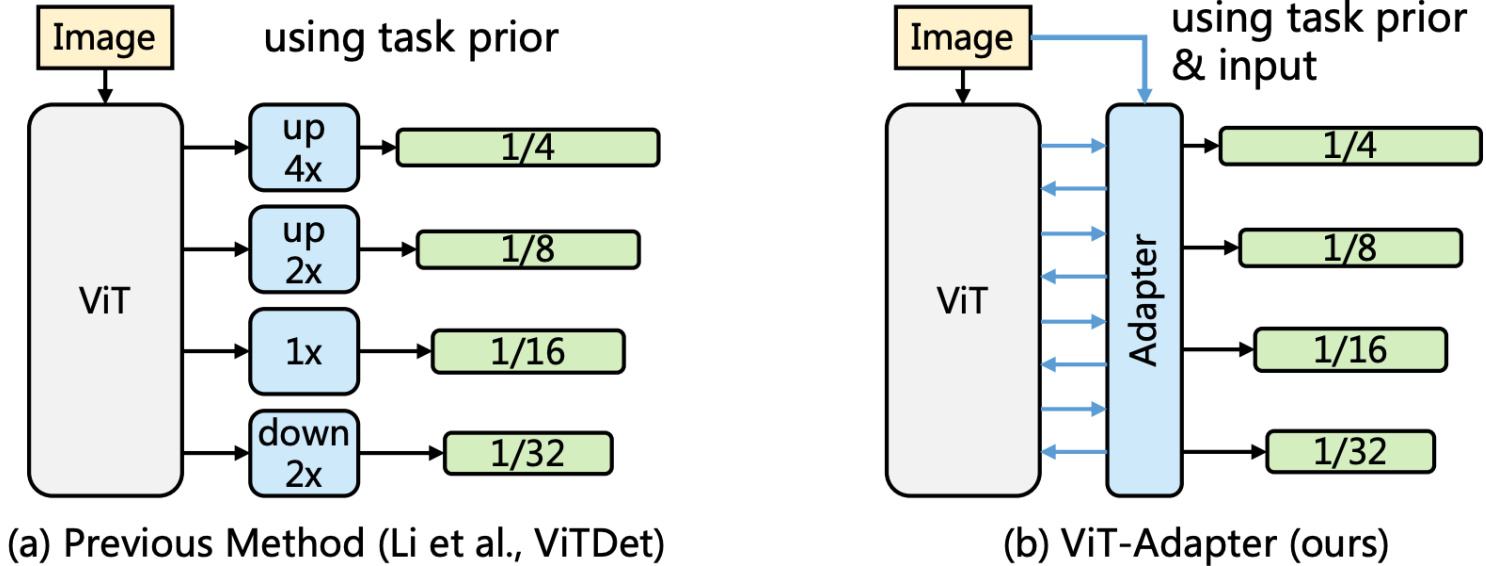


Figure 3: **Overview of ViT-Adapter and two related approaches.** Li et al. (2021b) and ViTDet (Li et al., 2022b) build simple feature pyramid to adapt plain ViT for object detection, which only consider task prior. Differently, our adapter utilizes both task prior and the input image.

ViT-Adapter

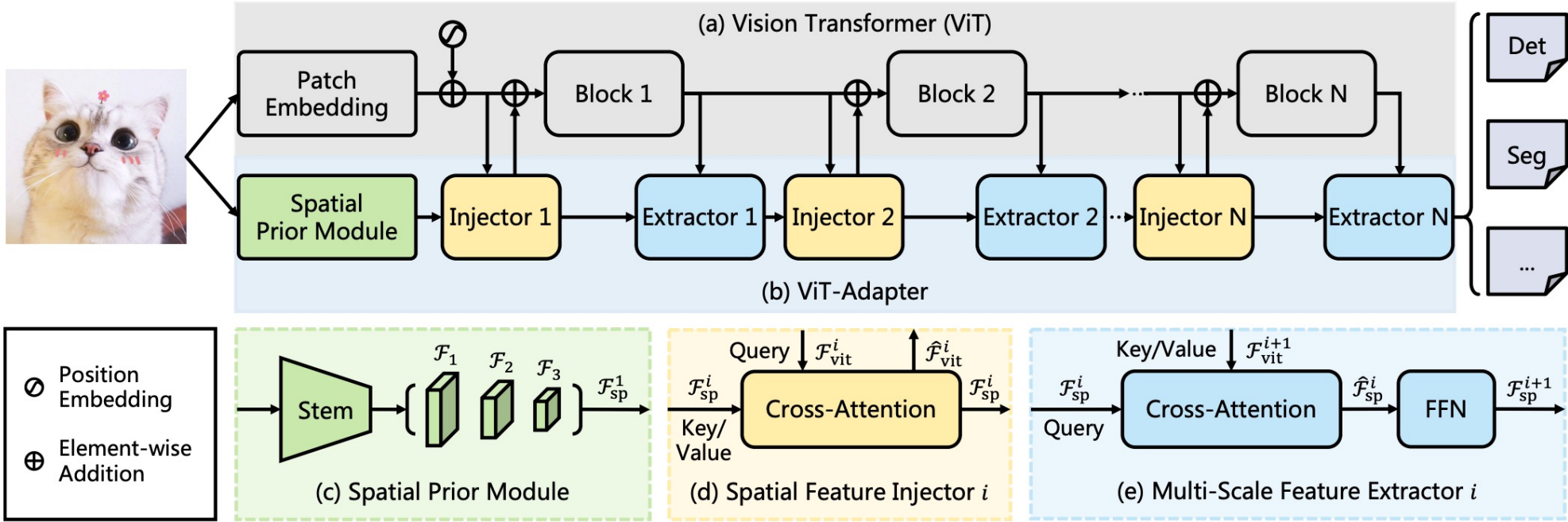


Figure 4: **Overall architecture of ViT-Adapter.** (a) The ViT, whose encoder layers are divided into N (usually $N = 4$) equal blocks for feature interaction. (b) Our ViT-Adapter, which contains three key designs, including (c) a spatial prior module for modeling local spatial contexts from the input image, (d) a spatial feature injector for introducing spatial priors into ViT, and (e) a multi-scale feature extractor for reorganizing multi-scale features from the single-scale features of ViT.

DeIT

- Mimic a teacher CNN model's predictions
- Add a special DIST-token for distillation
- Combine and minimize both
 - CE(GT, CLS-token)
 - CE(Teacher-pred, DIST-token)

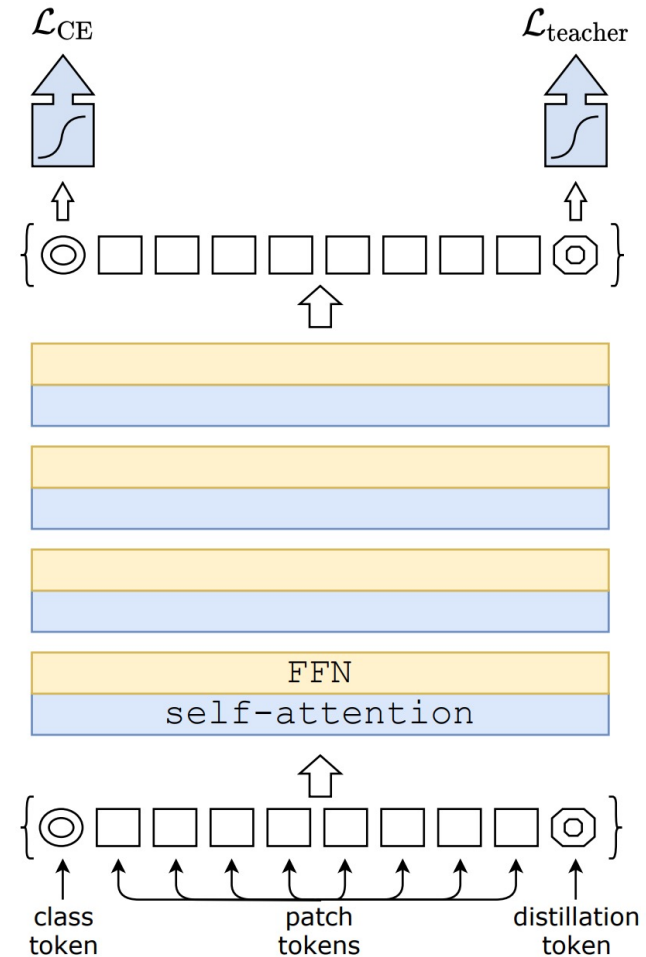


Figure 2: Our distillation procedure: we simply include a new *distillation token*. It interacts with the class and patch tokens through the self-attention layers. This distillation token is employed in a similar fashion as the class token, except that on output of the network its objective is to reproduce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

A ConvNet for the 2020s

Zhuang Liu^{1,2*} Hanzi Mao¹ Chao-Yuan Wu¹ Christoph Feichtenhofer¹ Trevor Darrell² Saining Xie^{1†}

¹Facebook AI Research (FAIR) ²UC Berkeley

Code: <https://github.com/facebookresearch/ConvNeXt>

ConvNeXt

ConvNeXt V2: Co-designing and Scaling ConvNets with Masked Autoencoders

Sanghyun Woo^{1*} Shoubhik Debnath² Ronghang Hu²
Xinlei Chen² Zhuang Liu² In So Kweon¹ Saining Xie^{3†}
¹KAIST ²Meta AI, FAIR ³New York University

Code: <https://github.com/facebookresearch/ConvNeXt-V2>

ConvNeXt v1

- “Modernized” ResNet

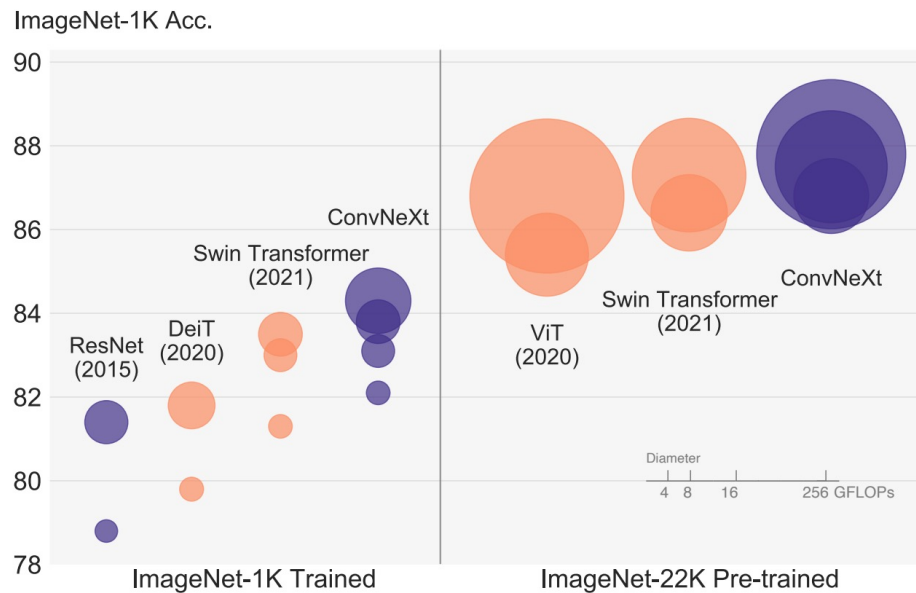
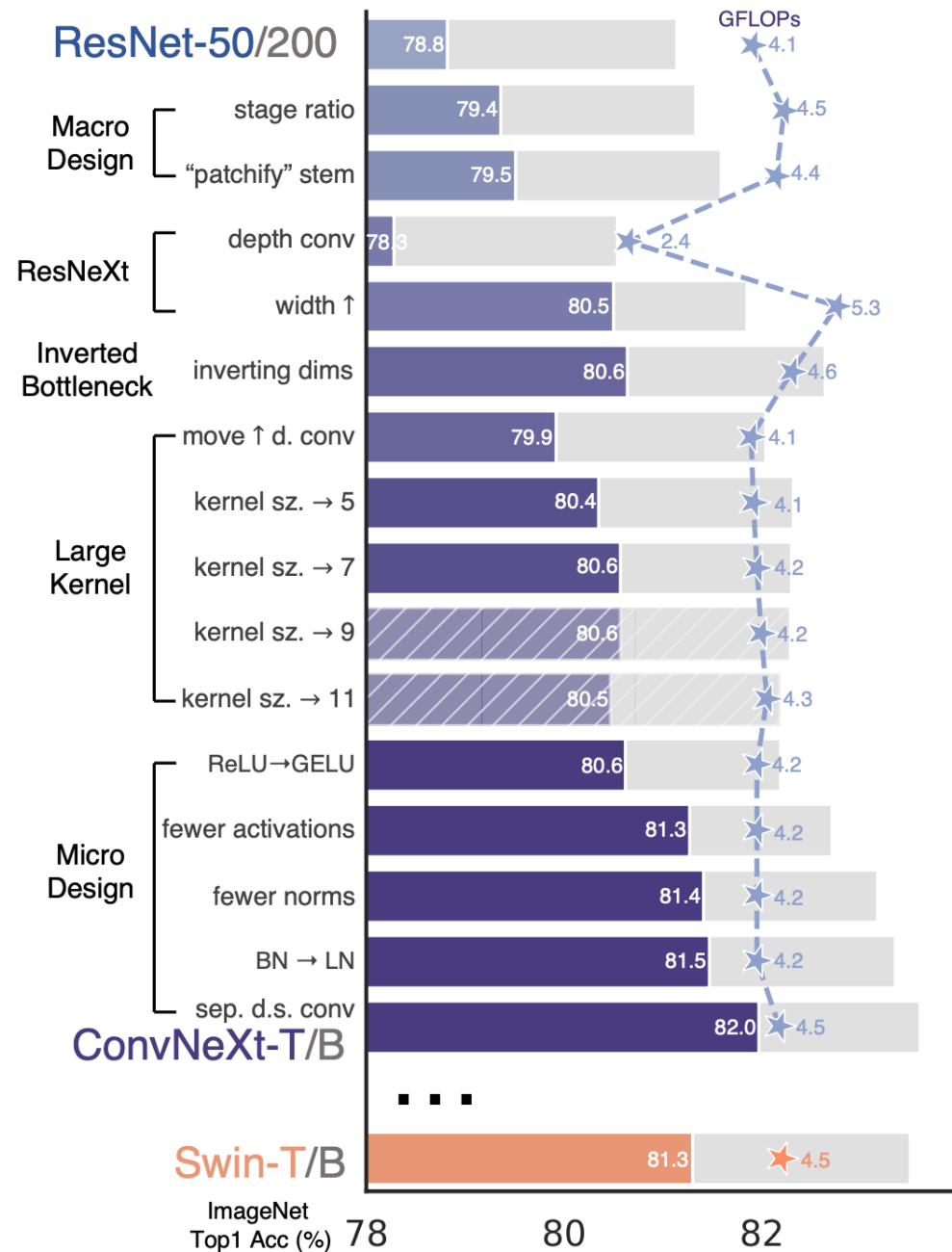


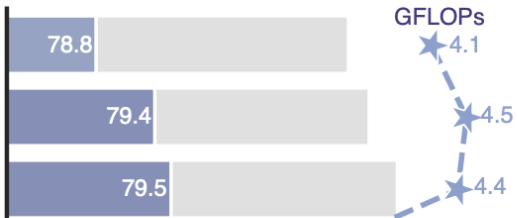
Figure 1. **ImageNet-1K classification** results for • ConvNets and ○ vision Transformers. Each bubble’s area is proportional to FLOPs of a variant in a model family. ImageNet-1K/22K models here take $224^2/384^2$ images respectively. ResNet and ViT results were obtained with improved training procedures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



ResNet-50/200

Macro Design

stage ratio
"patchify" stem



of blocks in each stage from [3,4,6,3] to [3,3,9,3]

7x7 conv with stride 2 + max-pool (4x downsamp) => 4x4 conv with stride 4

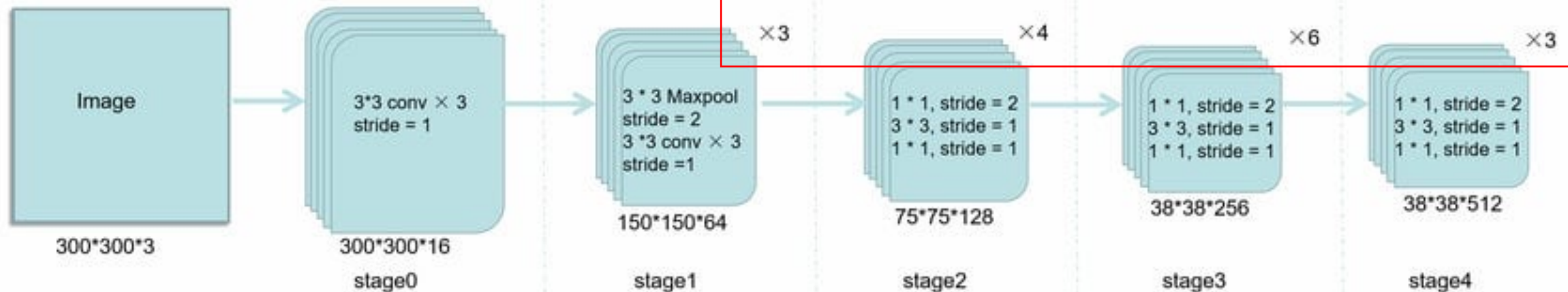
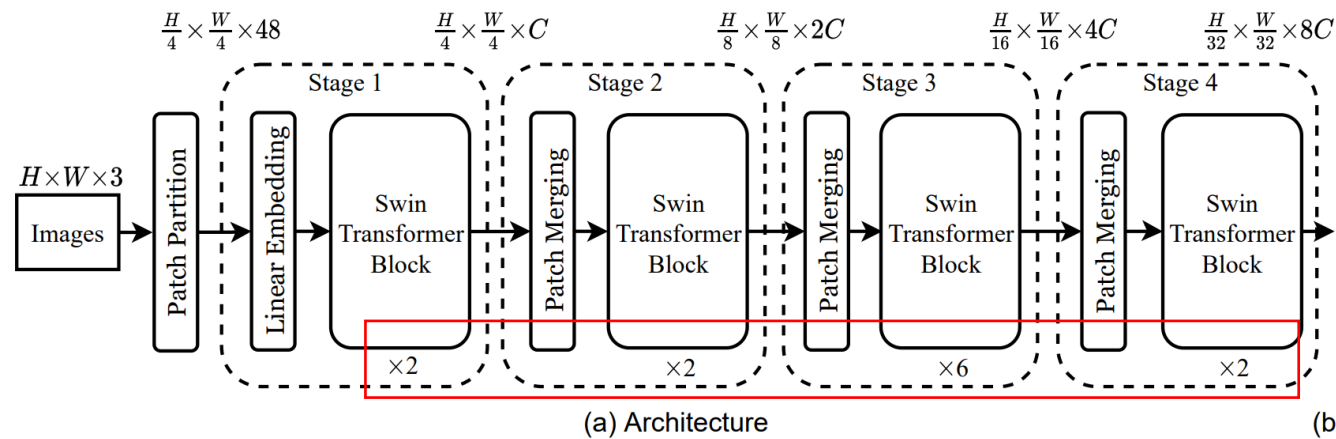
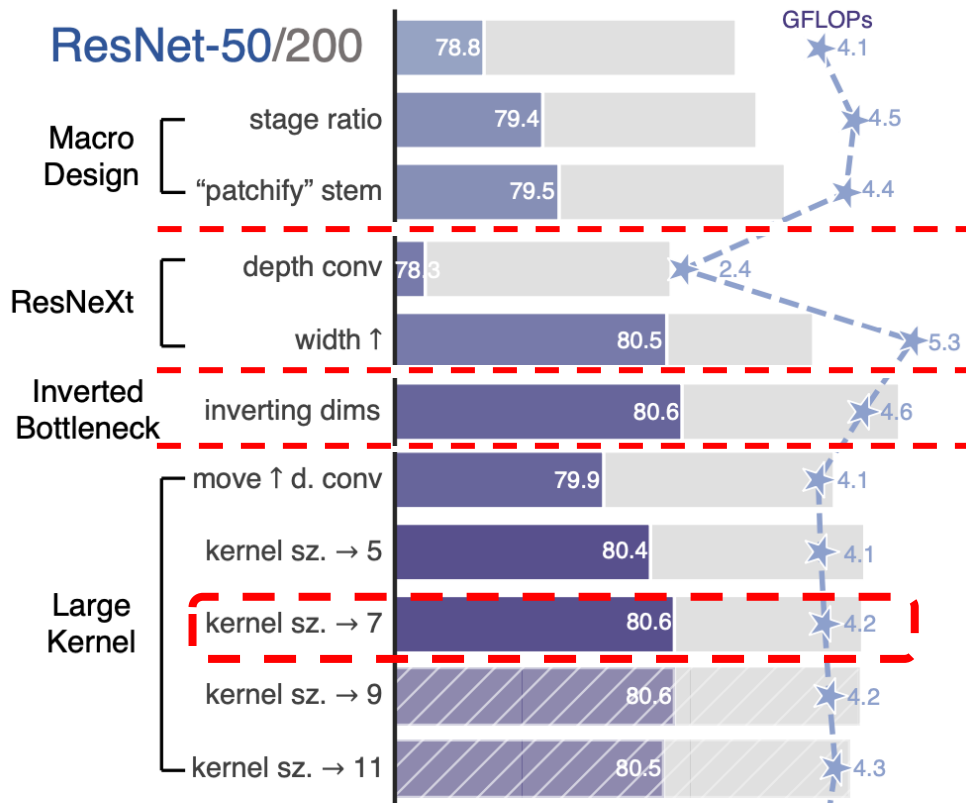


Fig: https://www.researchgate.net/figure/The-structure-of-the-Resnet50-used-in-this-paper-It-includes-5-convolutional-stages_fig1_339940039



(a) Architecture

(b)



of blocks in each stage from [3,4,6,3] to [3,3,9,3]

7x7 conv with stride 2 + max-pool (4x downsamp) => 4x4 conv with stride 4

group convolution of ResNeXt is emulated with depthwise convolution

of channels increased to match Swin-T

Increase embedding dim

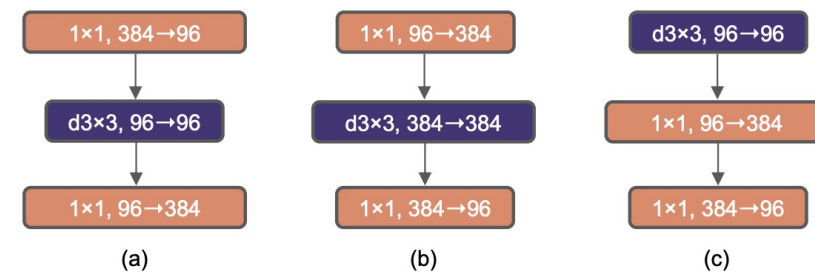
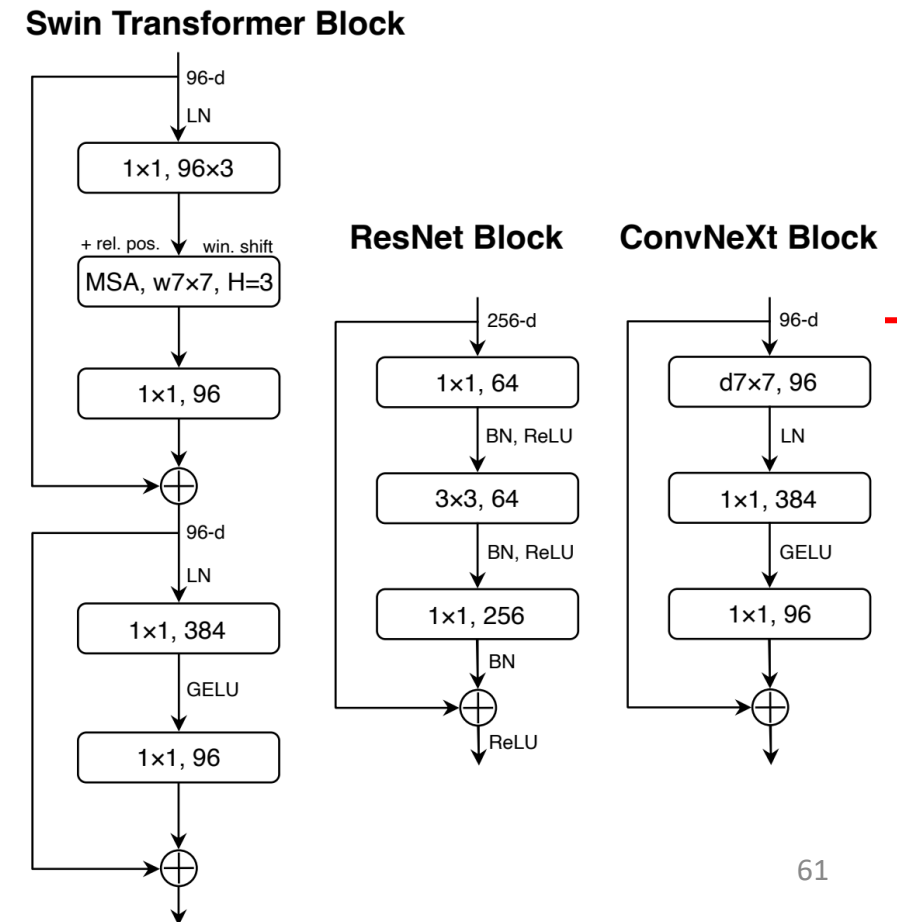
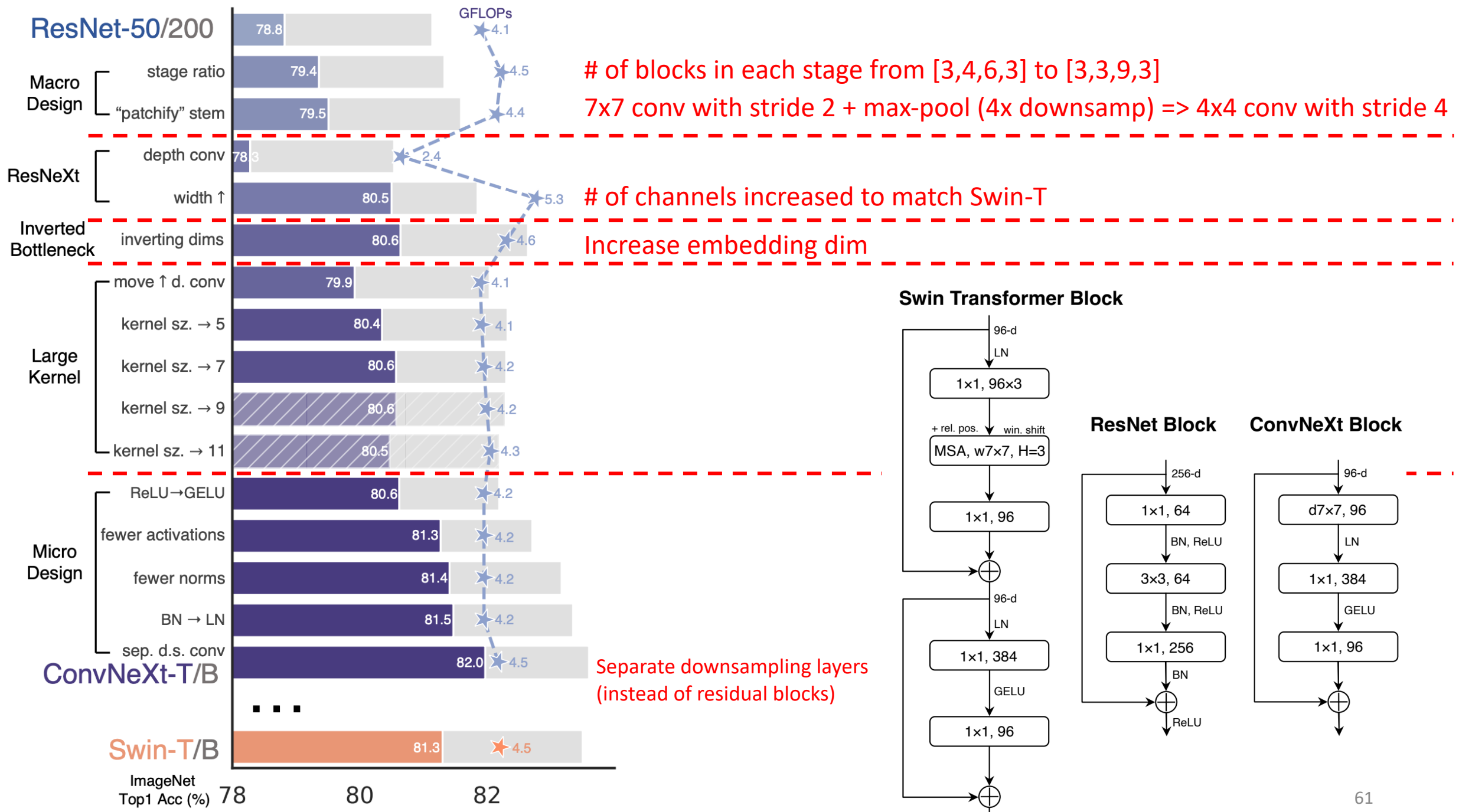


Figure 3. **Block modifications and resulted specifications.** (a) is a ResNeXt block; in (b) we create an inverted bottleneck block and in (c) the position of the spatial depthwise conv layer is moved up.



ConvNeXt v2

- Training ConvNeXt v1 with Masked AE performs poorly
- Introduce
 - fully convolutional masked autoencoder framework and
 - a new Global Response Normalization to enhance inter-channel competition

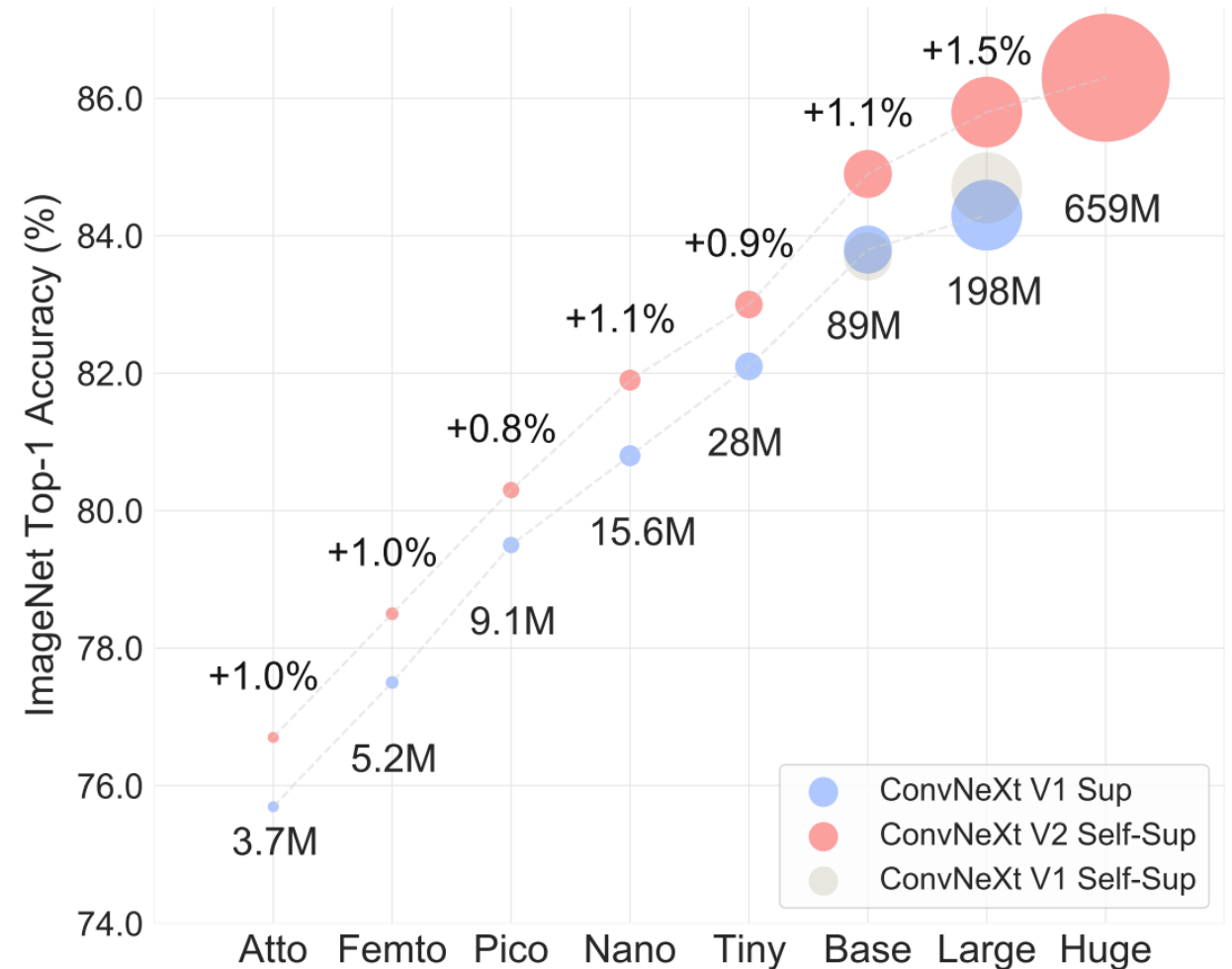


Figure 1. **ConvNeXt V2 model scaling.** The ConvNeXt V2 model, which has been pre-trained using our fully convolutional masked autoencoder framework, performs significantly better than the previous version across a wide range of model sizes.

ConvNeXt v2

Fully Convolutional Masked Autoencoder

- Encoder:
 - Sparse conv to prevent simple copy-paste of input (not a problem in MAE with transformers)
 - Consider image as a 2D array of pixels
- Decoder: ConvNeXt block

w/o Sparse conv.	w/ Sparse conv.
79.3	83.7

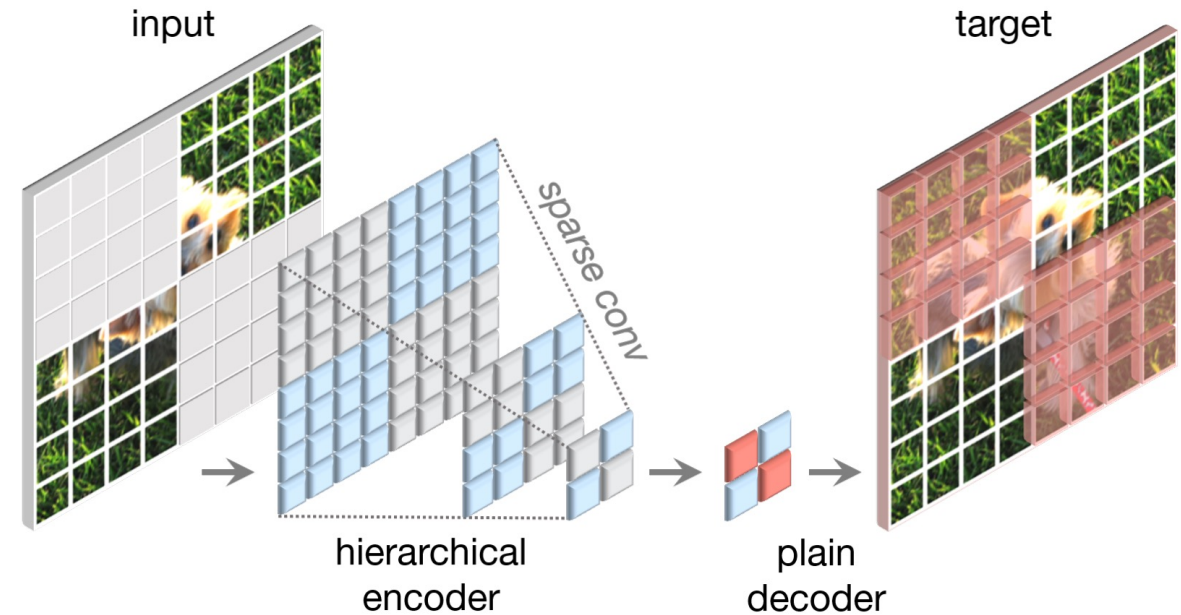
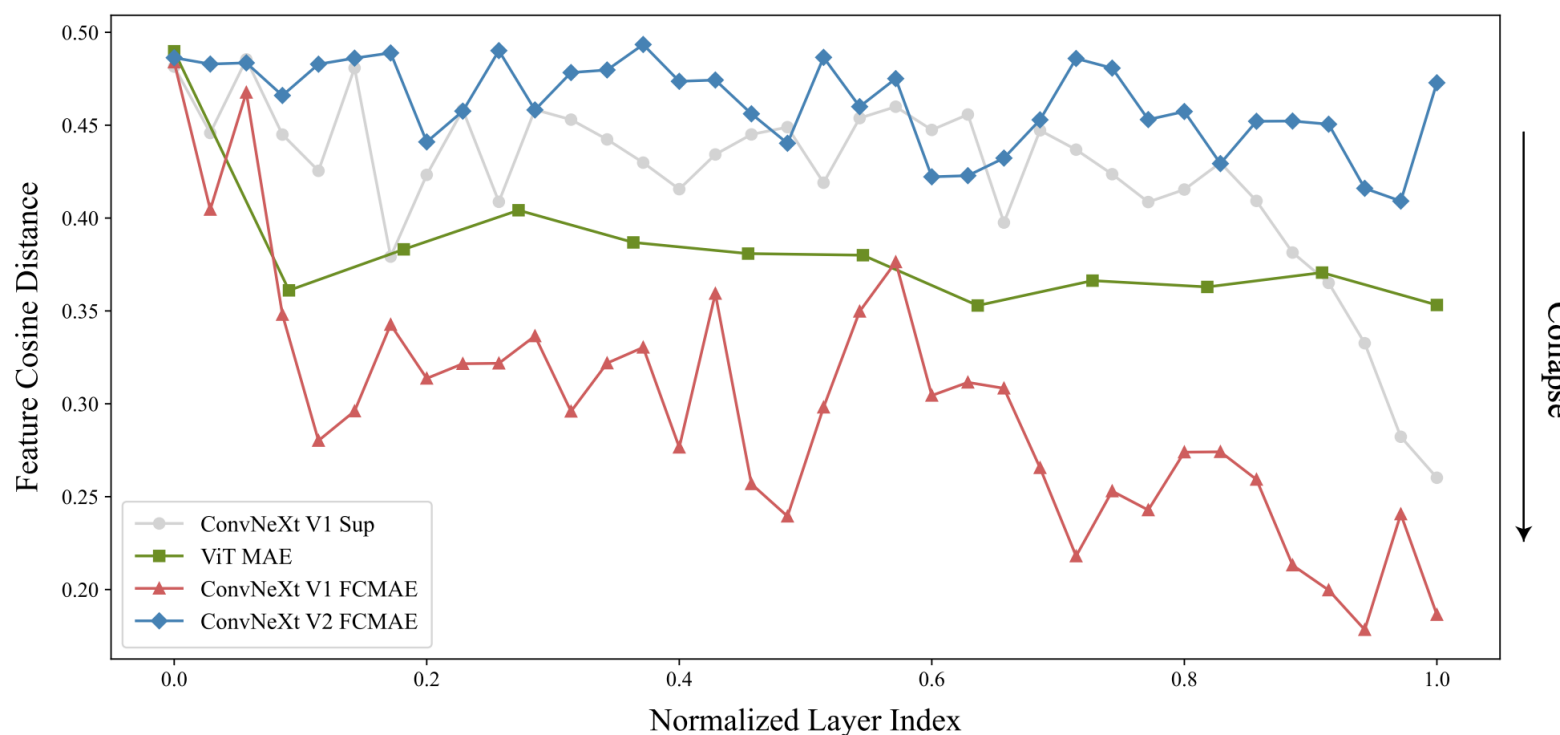
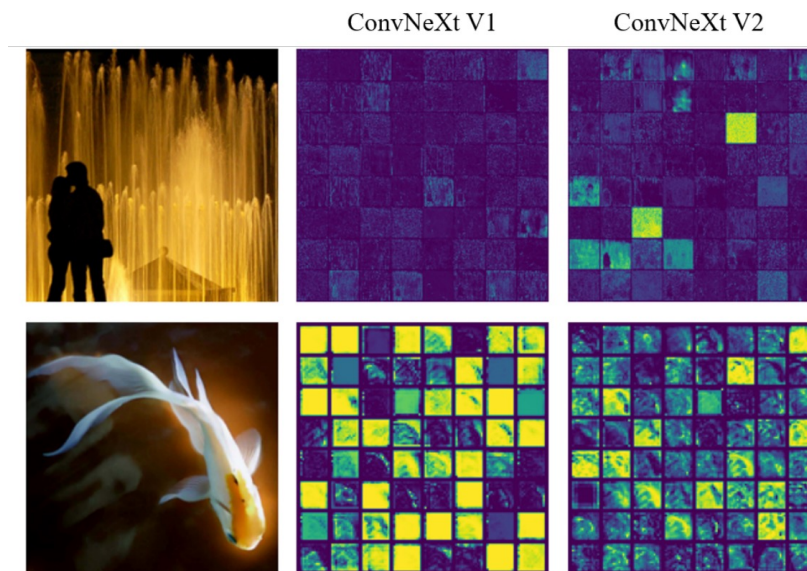


Figure 2. **Our FCMAE framework.** We introduce a fully convolutional masked autoencoder (FCMAE). It consists of a sparse convolution-based ConvNeXt encoder and a lightweight ConvNeXt block decoder. Overall, the architecture of our autoencoder is asymmetric. The encoder processes only the visible pixels,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

ConvNeXt v2

Global Response Normalization

- Feature collapse: many dead/saturated feature maps



ConvNeXt v2

Global Response Normalization

- Inspiration from lateral inhibition in brain
- Three steps:
 1. global feature aggregation,
 2. feature normalization, and
 3. feature calibration.

Algorithm 1 Pseudocode of GRN in a PyTorch-like style.

```
# gamma, beta: learnable affine transform parameters
# X: input of shape (N,H,W,C)

gx = torch.norm(X, p=2, dim=(1,2), keepdim=True)
nx = gx / (gx.mean(dim=-1, keepdim=True)+1e-6)
return gamma * (X * nx) + beta + X
```

First, we aggregate a spatial feature map X_i into a vector gx with a global function $\mathcal{G}(\cdot)$:

$$\mathcal{G}(X) := X \in \mathcal{R}^{H \times W \times C} \rightarrow gx \in \mathcal{R}^C. \quad (1)$$

This can be viewed as a simple pooling layer. We experimented with different functions in Table 2a. Interestingly, global average pooling, a widely used feature aggregator [37, 72], did not perform well in our case. Instead, we found that using norm-based feature aggregation, specifically, using L2-norm, resulted in better performance. This gives us a set of aggregated values $\mathcal{G}(X) = gx = \{\|X_1\|, \|X_2\|, \dots, \|X_C\|\} \in \mathcal{R}^C$ where $\mathcal{G}(X)_i = \|X_i\|$ is a scalar that aggregates the statistics of the i -th channel.

Next, we apply a response normalization function $\mathcal{N}(\cdot)$ to the aggregated values. Concretely, we use a standard divisive normalization as follows,

$$\mathcal{N}(\|X_i\|) := \|X_i\| \in \mathcal{R} \rightarrow \frac{\|X_i\|}{\sum_{j=1, \dots, C} \|X_j\|} \in \mathcal{R}, \quad (2)$$

where $\|X_i\|$ is the L2-norm of the i -th channel.¹ Intu-

Finally, we calibrate the original input responses using the computed feature normalization scores:

$$X_i = X_i * \mathcal{N}(\mathcal{G}(X)_i) \in \mathcal{R}^{H \times W} \quad (3)$$

ConvNeXt v2

Global Response Normalization

case	ft
g.avg.	83.7
L1	84.3
L2	84.6

(a) **Global aggregation** $G(\cdot)$. L2 Norm-based aggregation function produces the best result.

case	ft
$(\ X_i\ - \mu)/\sigma$	84.5
$1/\sum \ X_i\ $	83.8
$\ X_i\ /\sum \ X_i\ $	84.6

(b) **Normalization operator**, $N(\cdot)$. Divisive normalization is an effective channel importance calibrator.

case	ft
w/o skip	84.0
w/ skip	84.6

(c) **Residual connection** helps with GRN optimization and leads to better performance.

case	ft
Baseline	83.7
LRN [26]	83.2
BN [22]	80.5
LN [1]	83.8
GRN	84.6

(d) **Feature normalization**. GRN outperforms other normalizations through global contrasting.

case	ft	#param
Baseline	83.7	89M
SE [19]	84.4	109M
CBAM [48]	84.5	109M
GRN	84.6	89M

(e) **Feature re-weighting**. GRN does effective and efficient feature re-weighting without parameter overhead.

case	ft
Baseline	83.7
drop at ft.	78.8
add at ft.	80.6
both	84.6

(f) **GRN in pre-training/fine-tuning**. To be effective, GRN should be used in both stages.

Table 2. **GRN ablations** with ConvNeXt-Base. We report fine-tuning accuracy on ImageNet-1K. Our final proposal is marked in gray .

ConvNeXt v2

Backbone	Method	#param	FLOPs	Val acc.
ConvNeXt V1-B	Supervised	89M	15.4G	83.8
ConvNeXt V1-B	FCMAE	89M	15.4G	83.7
ConvNeXt V2-B	Supervised	89M	15.4G	84.3 (+0.5)
ConvNeXt V2-B	FCMAE	89M	15.4G	84.6 (+0.8)
ConvNeXt V1-L	Supervised	198M	34.4G	84.3
ConvNeXt V1-L	FCMAE	198M	34.4G	84.4
ConvNeXt V2-L	Supervised	198M	34.4G	84.5 (+0.2)
ConvNeXt V2-L	FCMAE	198M	34.4G	85.6 (+1.3)

Table 3. **Co-design matters.** When the architecture and the learning framework are co-designed and used together, masked image pre-training becomes effective for ConvNeXt. We report the fine-tuning performance from 800 epoch FCMAE pre-trained models. The relative improvement is bigger with a larger model.

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

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

2023

**FastViT: A Fast Hybrid Vision Transformer
using Structural Reparameterization**Pavan Kumar Anasosalu Vasu[†] James Gabriel Jeff Zhu Oncel Tuzel Anurag Ranjan[†]

Apple

**FASTERViT: FAST VISION TRANSFORMERS WITH
HIERARCHICAL ATTENTION****Ali Hatamizadeh, Greg Heinrich, Hongxu Yin, Andrew Tao, Jose M. Alvarez,
Jan Kautz, Pavlo Molchanov**

NVIDIA

{ahatamizadeh, pmolchanov}@nvidia.com

Fast/Faster ViTs

Fast ViT

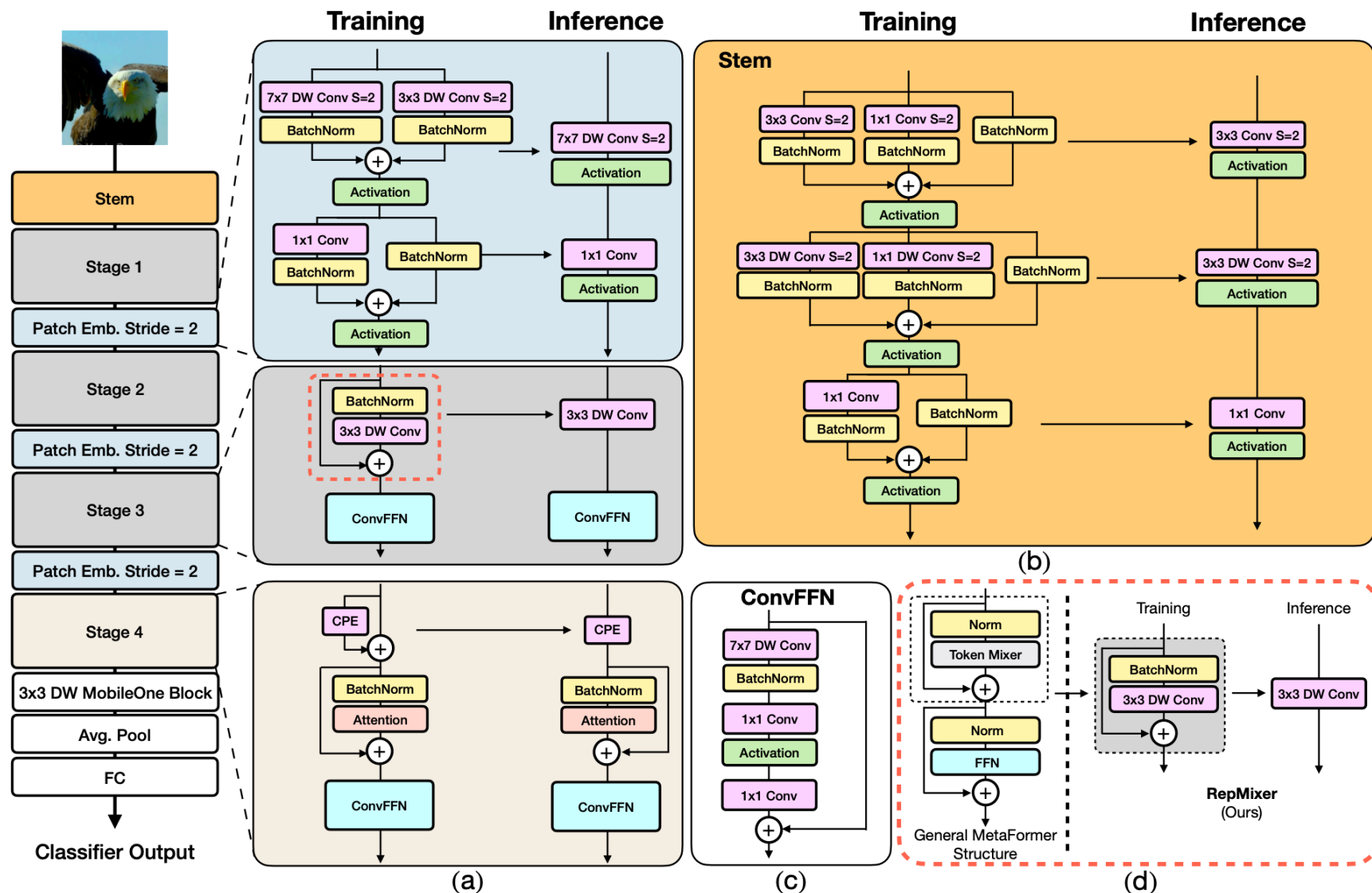


Figure 2: (a) Overview of FastViT architecture which decouples train-time and inference-time architecture. Stages 1, 2, and 3 have the same architecture and uses RepMixer for token mixing. In stage 4, self attention layers are used for token mixing. (b) Architecture of the convolutional stem. (c) Architecture of convolutional-FFN (d) Overview of *RepMixer* block, which reparameterizes a skip connection at inference.

Faster ViT

Figure 2: Visualization of the proposed Hierarchical Attention in the feature space. By performing local window attention and hierarchical attention we can achieve global information propagation at reduced costs.

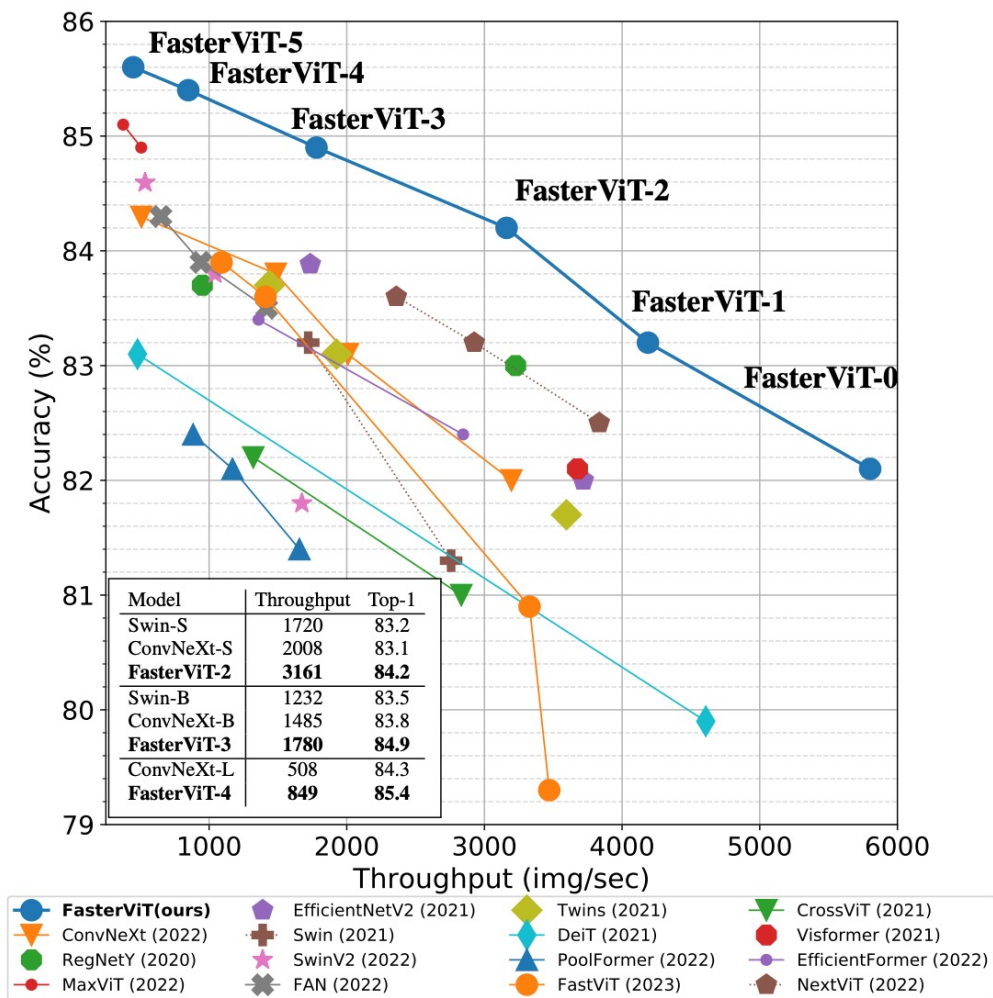
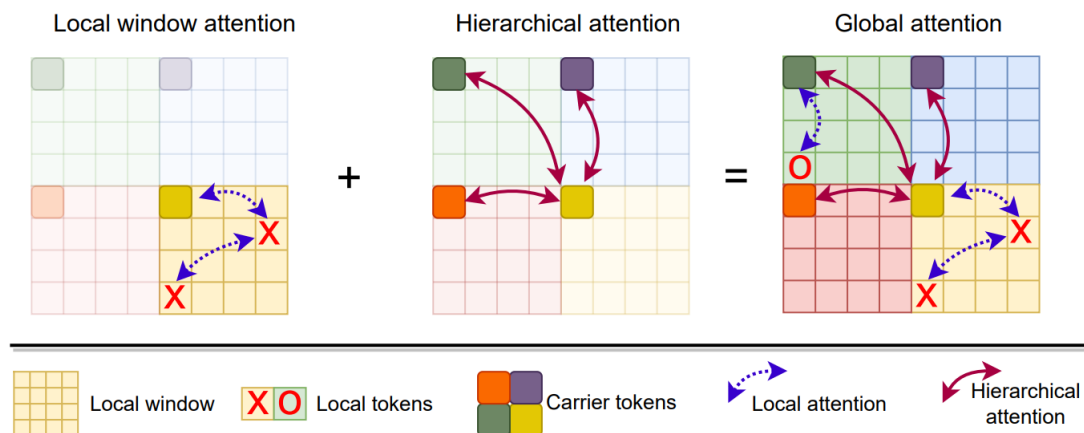


Figure 1: Comparison of image throughput and ImageNet-1K Top-1 accuracy. Throughput is measured on A100 GPU with batch size of 128.

Pretraining Vision Transformers

Masked Autoencoders

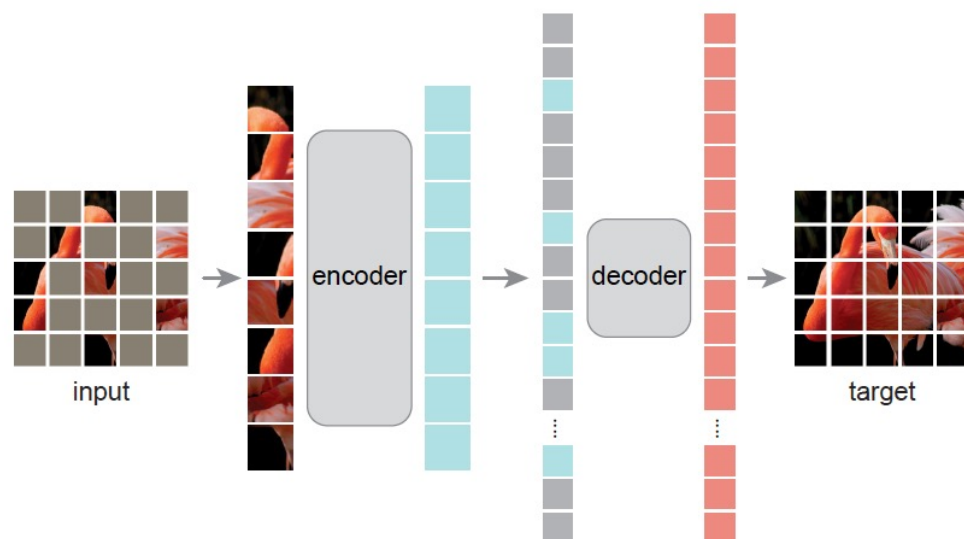


Figure 1. **Our MAE architecture.** During pre-training, a large random subset of image patches (e.g., 75%) is masked out. The encoder is applied to the small subset of *visible patches*. Mask tokens are introduced *after* the encoder, and the full set of encoded patches and mask tokens is processed by a small decoder that reconstructs the original image in pixels. After pre-training, the decoder is discarded and the encoder is applied to uncorrupted images (full sets of patches) for recognition tasks.

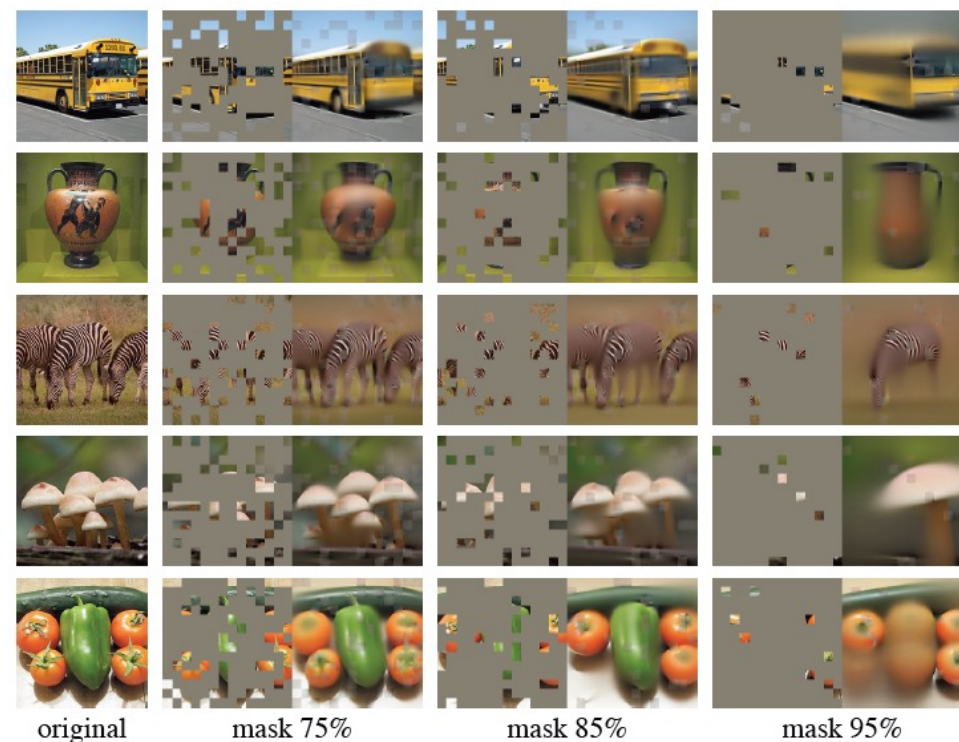


Figure 4. Reconstructions of ImageNet *validation* images using an MAE pre-trained with a masking ratio of 75% but applied on inputs with higher masking ratios. The predictions differ plausibly from the original images, showing that the method can generalize.

MSE loss between pixels for masked tokens only!

Masked Autoencoders

blocks	ft	lin
1	84.8	65.5
2	84.9	70.0
4	84.9	71.9
8	84.9	73.5
12	84.4	73.3

(a) **Decoder depth.** A deep decoder can improve linear probing accuracy.

case	ft	lin
pixel (w/o norm)	84.9	73.5
pixel (w/ norm)	85.4	73.9
PCA	84.6	72.3
dVAE token	85.3	71.6

(d) **Reconstruction target.** Pixels as reconstruction targets are effective.

dim	ft	lin
128	84.9	69.1
256	84.8	71.3
512	84.9	73.5
768	84.4	73.1
1024	84.3	73.1

(b) **Decoder width.** The decoder can be narrower than the encoder (1024-d).

case	ft	lin
none	84.0	65.7
crop, fixed size	84.7	73.1
crop, rand size	84.9	73.5
crop + color jit	84.3	71.9

(e) **Data augmentation.** Our MAE works with minimal or no augmentation.

case	ft	lin	FLOPs
encoder w/ [M]	84.2	59.6	3.3×
encoder w/o [M]	84.9	73.5	1×

(c) **Mask token.** An encoder without mask tokens is more accurate and faster (Table 2).

case	ratio	ft	lin
random	75	84.9	73.5
block	50	83.9	72.3
block	75	82.8	63.9
grid	75	84.0	66.0

(f) **Mask sampling.** Random sampling works the best. See Figure 6 for visualizations.

Table 1. **MAE ablation experiments** with ViT-L/16 on ImageNet-1K. We report fine-tuning (ft) and linear probing (lin) accuracy (%). If not specified, the default is: the decoder has depth 8 and width 512, the reconstruction target is unnormalized pixels, the data augmentation is random resized cropping, the masking ratio is 75%, and the pre-training length is 800 epochs. Default settings are marked in gray.

Masked Autoencoders

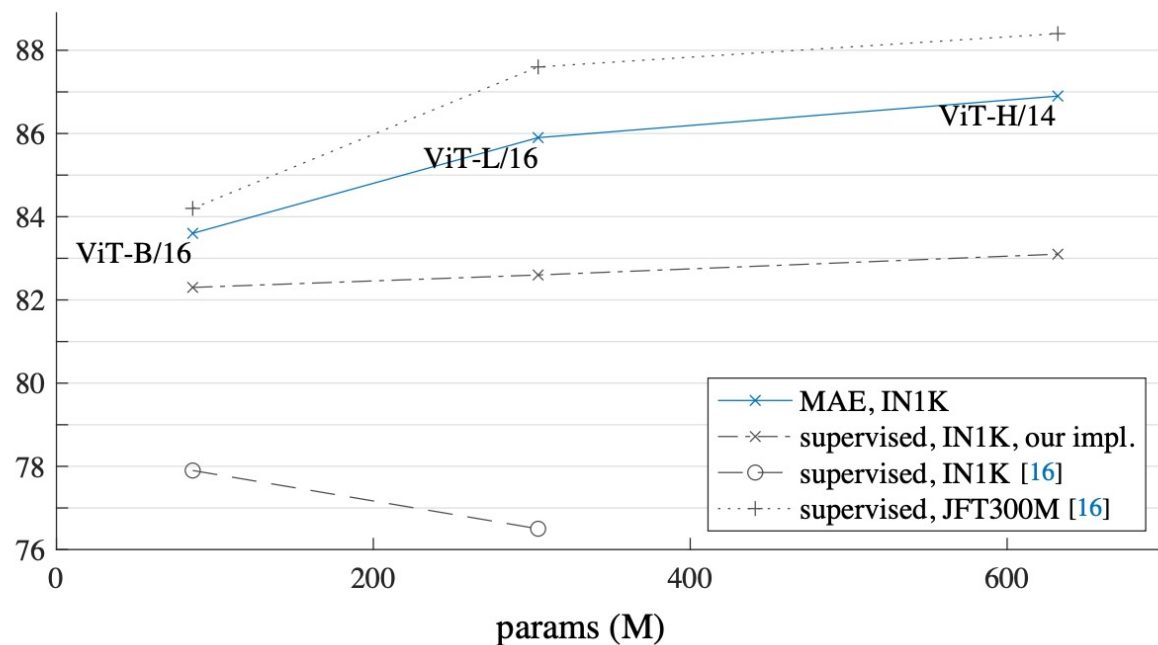


Figure 8. **MAE pre-training vs. supervised pre-training**, evaluated by fine-tuning in ImageNet-1K (224 size). We compare with the original ViT results [16] trained in IN1K or JFT300M.

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

Table 3. **Comparisons with previous results on ImageNet-1K**. The pre-training data is the ImageNet-1K training set (except the tokenizer in BEiT was pre-trained on 250M DALLE data [50]). All self-supervised methods are evaluated by end-to-end fine-tuning. The ViT models are B/16, L/16, H/14 [16]. The best for each column is underlined. All results are on an image size of 224, except for ViT-H with an extra result on 448. Here our MAE reconstructs normalized pixels and is pre-trained for 1600 epochs.

SimMIM

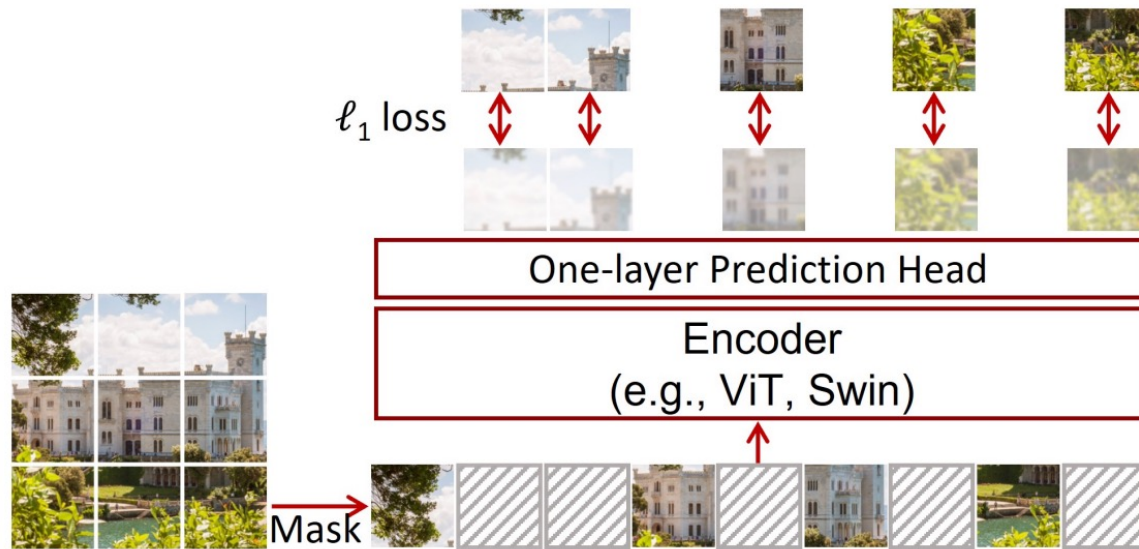


Figure 1. An illustration of our simple framework for masked language modeling, named *SimMIM*. It predicts raw pixel values of the randomly masked patches by a lightweight one-layer head, and performs learning using a simple ℓ_1 loss.

Methods	Input Size	Fine-tuning Top-1 acc (%)	Linear eval Top-1 acc (%)	Pre-training costs
Sup. baseline [44]	224 ²	81.8	-	-
DINO [5]	224 ²	82.8	78.2	2.0×
MoCo v3 [9]	224 ²	83.2	76.7	1.8×
ViT [15]	384 ²	79.9	-	~4.0×
BEiT [1]	224 ²	83.2	56.7	1.5× [†]
Ours	224 ²	83.8	56.7	1.0×

Table 6. System-level comparison using ViT-B as the encoder. Training costs are counted in relative to our approach. [†] BEiT requires an additional stage to pre-train dVAE, which is not counted.

DINO v1 (ICCV'21)

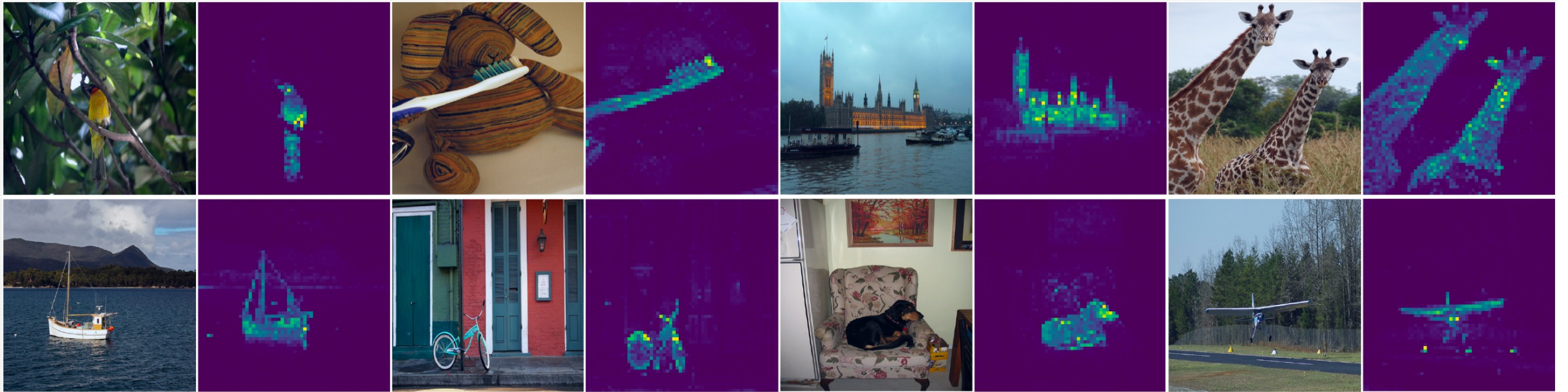


Figure 1: **Self-attention from a Vision Transformer with 8×8 patches trained with no supervision.** We look at the self-attention of the [CLS] token on the heads of the last layer. This token is not attached to any label nor supervision. These maps show that the model automatically learns class-specific features leading to unsupervised object segmentations.

DINO v1 (ICCV'21)

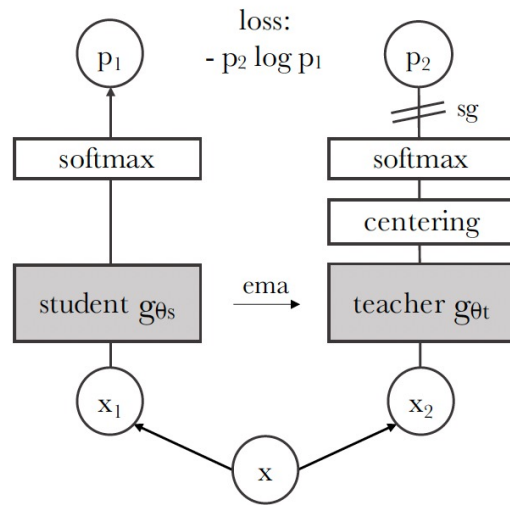


Figure 2: **Self-distillation with no labels.** We illustrate DINO in the case of one single pair of views (x_1, x_2) for simplicity. The model passes two different random transformations of an input image to the student and teacher networks. Both networks have the same architecture but different parameters.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

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

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# 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()
```

DINO v2 (TMLR'24)

- **Patch-Level Loss:** Unlike v1, which looked at the whole image, v2 forced the model to match individual patches. This made it much better at tasks like depth estimation and semantic segmentation.

Patch-level objective (Zhou et al., 2022a). We randomly mask some of the input patches given to the student, but not to the teacher. We then apply the student iBOT head to the student mask tokens. Similarly, we apply the teacher iBOT head to the (visible) teacher patch tokens corresponding to the ones masked in the student. We then apply the softmax and centering steps as above, and obtain the iBOT loss term:

$$\mathcal{L}_{iBOT} = - \sum_i p_{ti} \log p_{si}$$

, where i are patch indices for masked tokens. Similarly to above, we learn the parameters of the student, and build the teacher head through exponential moving average.

DINO v2 (TMLR'24)

- **Data Curation:** Meta developed an automated pipeline to curate a massive, balanced dataset (LVD-142M) from the web, proving that data quality matters as much as quantity.

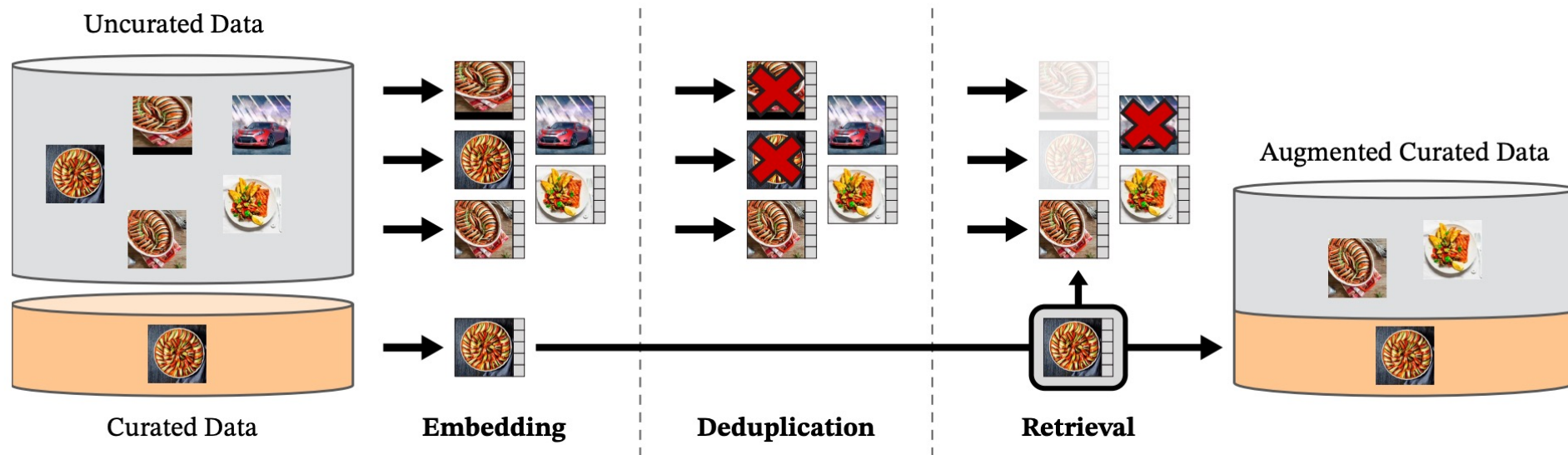


Figure 3: **Overview of our data processing pipeline.** Images from curated and uncurated data sources are first mapped to embeddings. Uncurated images are then deduplicated before being matched to curated images. The resulting combination augments the initial dataset through a self-supervised retrieval system.

DINO v2 (TMLR'24)

- And many other tricks

- **Untying head weights between both objectives.** Both the DINO and the iBOT loss use a learnable MLP projection head. It is applied to the output tokens and the loss is compute atop. In Zhou et al. (2022a), an ablation study shows that sharing parameters between the DINO and iBOT heads leads to better performance. At scale, we observed that the opposite is true, and we therefore use two separate heads in all our experiments.

- **Sinkhorn-Knopp centering (Caron et al., 2020).** Ruan et al. (2023) recommend to replace the teacher softmax-centering step of DINO and iBot by the Sinkhorn-Knopp (SK) batch normalization of SwAV (Caron et al., 2020). We run the Sinkhorn-Knopp algorithm steps for 3 iterations. For the student, we apply the softmax normalization.

- **KoLeo regularizer (Sablayrolles et al., 2019).** The KoLeo regularizer derives from the Kozachenko-Leonenko differential entropy estimator (see Beirlant et al. (1997); Delattre & Fournier (2017)) and encourages a uniform span of the features within a batch. Given a set of n vectors (x_1, \dots, x_n) , it is defined as

$$\mathcal{L}_{\text{koleo}} = -\frac{1}{n} \sum_{i=1}^n \log(d_{n,i}),$$

where $d_{n,i} = \min_{j \neq i} \|x_i - x_j\|$ is the minimum distance between x_i and any other point within the batch. We also ℓ_2 -normalize the features before computing this regularizer.

- **Adapting the resolution (Touvron et al., 2019).** Increasing image resolution is key to pixel-level downstream tasks such as segmentation or detection, where small objects disappear at low resolutions. However, training at high resolution is time and memory demanding, and instead, we increase the resolution of images to 518×518 during a short period at the end of pretraining. This is also similar to UniViT training from Likhomanenko et al. (2021) and FlexiViT training from Beyer et al. (2023).

DINO v3 (2025/6)

- Larger dataset
- Larger model

Table 2: Comparison of the teacher architectures used in DINOv2 and DINOv3 models. We keep the model 40 blocks deep, and increase the embedding dimension to 4096. Importantly, we use a patch size of 16 pixels, changing the effective sequence length for a given resolution.

Teacher model	DINOv2	DINOv3
Backbone	ViT-giant	ViT-7B
#Params	1.1B	6.7B
#Blocks	40	40
Patch Size	14	16
Pos. Embeddings	Learnable	RoPE
Registers	4	4
Embed. Dim.	1536	4096
FFN Type	SwiGLU	SwiGLU
FFN Hidden Dim.	4096	8192
Attn. Heads	24	32
Attn. Heads Dim.	64	128
DINO Head MLP	4096-4096-256	8192-8192-512
DINO Prototypes	128k	256k
iBOT Head MLP	4096-4096-256	8192-8192-384
iBOT Prototypes	128k	96k

DINO v3 (2025/6)

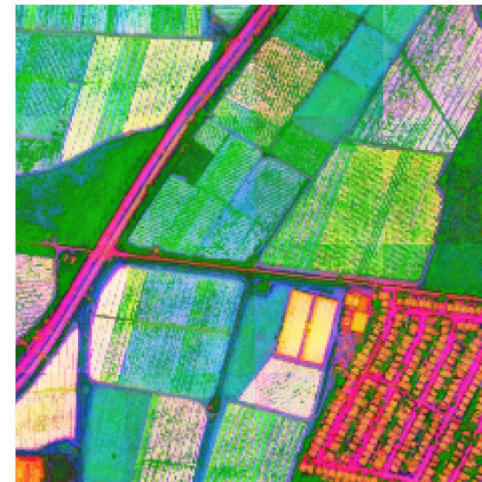
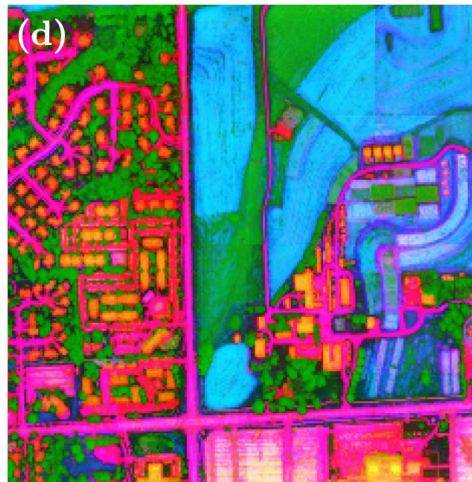
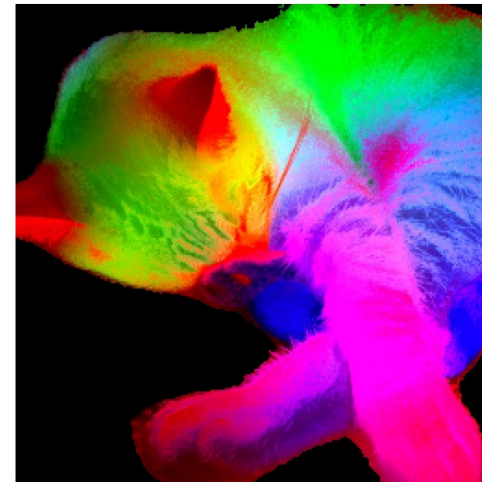
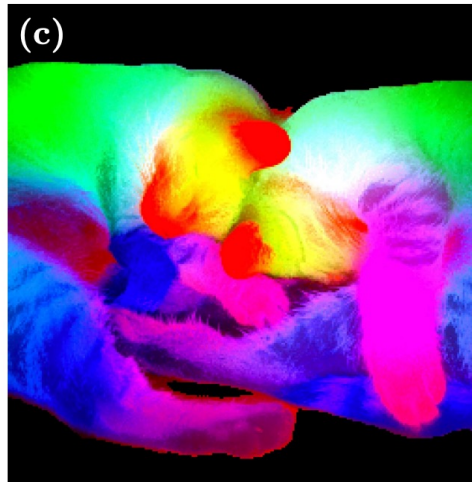
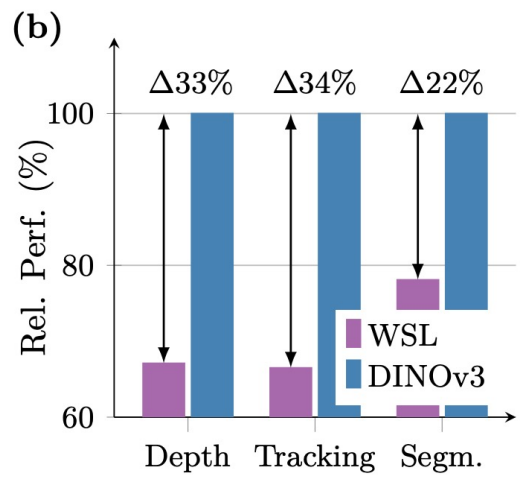
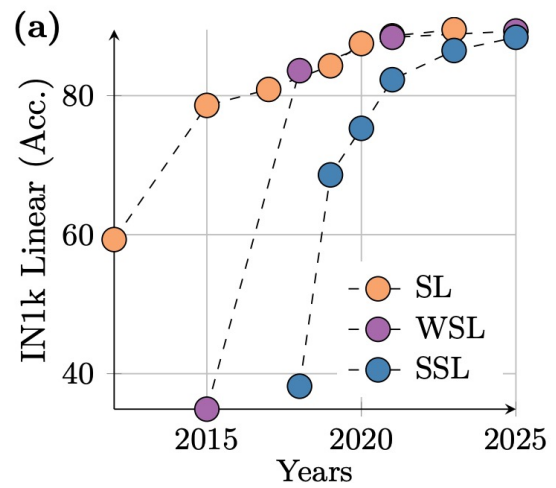
- Gram Anchoring:
 - Why: Feature collapse emerges where local details become noisy over long training runs.
 - Gram Anchoring: a regularization technique that keeps the relationship between image patches stable.

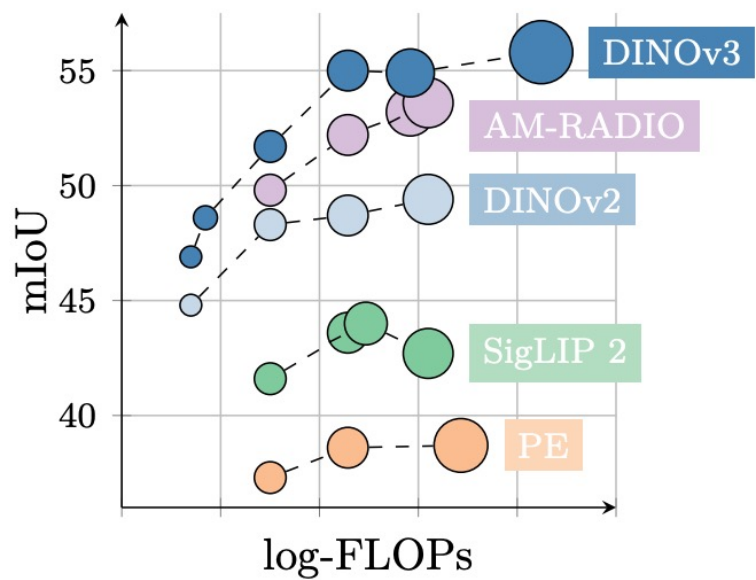
superior dense properties. By operating on the Gram matrix rather than the feature themselves, the local features are free to move, provided the structure of similarities remains the same. Suppose we have an image composed of P patches, and a network that operates in dimension d . Let us denote by \mathbf{X}_S (respectively \mathbf{X}_G) the $P \times d$ matrix of \mathbf{L}_2 -normalized local features of the student (respectively the Gram teacher). We define the loss $\mathcal{L}_{\text{Gram}}$ as follows:

$$\mathcal{L}_{\text{Gram}} = \|\mathbf{X}_S \cdot \mathbf{X}_S^\top - \mathbf{X}_G \cdot \mathbf{X}_G^\top\|_{\text{F}}^2. \quad (2)$$

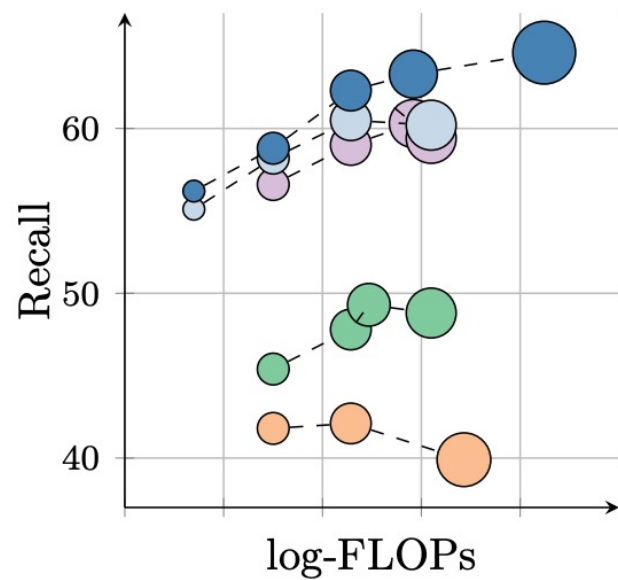
DINO v3 (2025/6)

- High-Resolution Focus: It replaces standard positional embeddings with RoPE (Rotary Positional Embeddings), allowing the model to process images at much higher resolutions (4k+) than previous versions.
- Frozen Performance: DINO v3 is designed to be a universal backbone that is so powerful it can often outperform specialized, human-labeled models even when its weights are completely frozen.

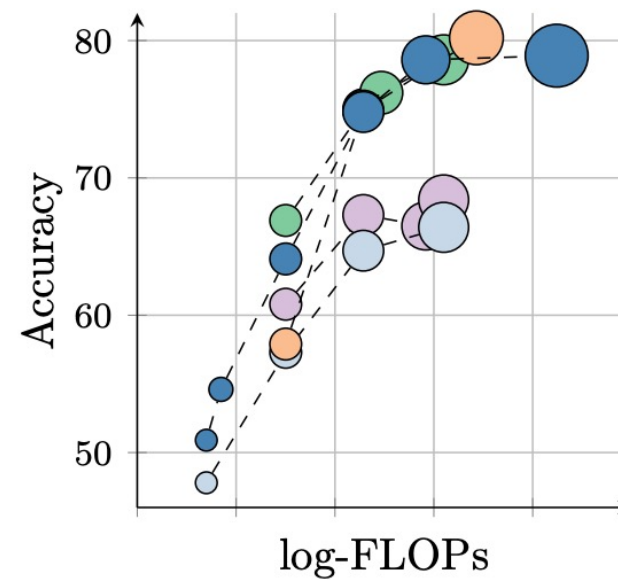




(a) Semantic segmentation (ADE20k)



(b) 3D keypoint matching (NAVI)



(c) OOD classif. (ObjectNet)



Figure 3: High-resolution dense features. We visualize the cosine similarity maps obtained with DINOv3 output features between the patches marked with a red cross and all other patches. Input image at 4096×4096 . *Please zoom in, do you agree with DINOv3?*

Feature	DINO v1 (2021)	DINO v2 (2023)	DINO v3 (2025/26)
Max Model Size	ViT-Base (86M params)	ViT-giant (1.1B params)	ViT-7B (7B params)
Training Data	ImageNet (1.3M images)	LVD-142M (142M images)	LVD-1.7B (1.7B images)
Learning Signal	Global (Image-level)	Global + Local (Patch-level)	Global + Stable Dense Features
Key Innovation	Self-distillation + Sinkhorn-Knopp	Curated data + Register tokens	Gram Anchoring + RoPE
Core Goal	Emergent object segments	Universal "frozen" backbone	Multi-purpose foundation model

Autoregressive Image Modeling (AIM)

<https://machinelearning.apple.com/research/autoregressive-image-models>

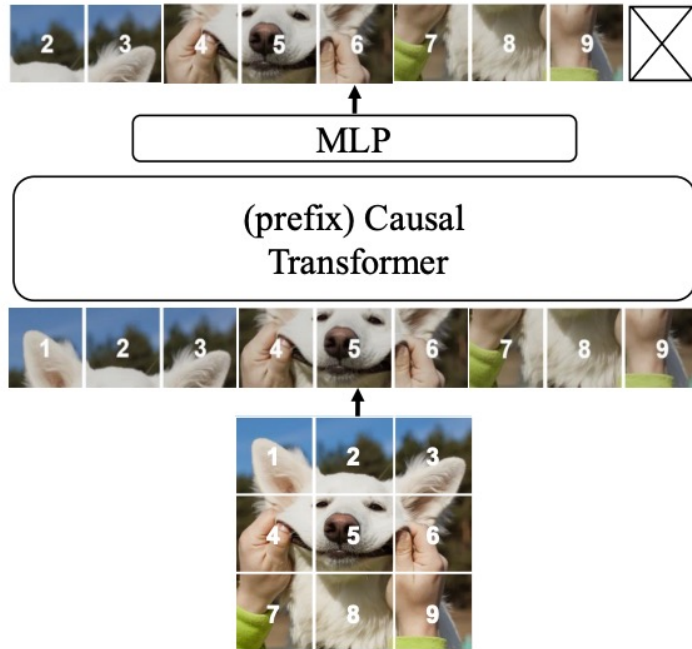


Figure 2. AIM pre-training overview. Input images are split into non-overlapping patches and embedded linearly following Dosovitskiy et al. [29]. The patch features are fed to a transformer in which the self-attention operation is causally masked to prevent attending to preceding positions. Afterward, a heavily parameterized MLP processes each of the patch features independently and finally projects it to pixel space. The targets correspond to the input sequence shifted one position to the left, requiring the model to predict the next patch in raster order.

Model	Arch.	Data	IN-1k	iNAT-18	Cifar10	Cifar100	Food101	DTD	Pets	Cars	iWildCam	Camelyon17	PCAM	RxRX1	EuroSAT	fMoW	Infographic	Avg
DINO [17]	ViT-B/8	IN-1k	80.1	66.0	97.8	87.3	89.5	78.4	92.3	89.2	58.5	93.7	90.2	6.1	98.2	57.0	41.1	75.0
iBOT [88]	ViT-L/16	IN-21k	83.5	70.5	99.2	93.3	93.5	81.6	92.8	90.8	61.8	94.5	90.0	5.9	98.0	60.3	47.7	77.6
DINOv2 [58]	ViT-g/14 ₅₁₆	LVD	86.4	84.5	99.6	95.2	96.3	86.3	96.4	95.6	68.2	96.5	90.7	8.0	98.6	66.7	58.8	81.9
BEiT [5]	ViT-L/14	IN-21k	62.2	44.4	94.4	78.7	79.0	64.0	80.9	69.5	52.0	92.8	88.2	4.2	97.5	47.7	25.9	65.4
MAE [41, 70]	ViT-H/14	IN-1k	80.9	64.6	97.1	85.8	90.2	78.1	95.0	93.7	58.1	94.2	89.8	5.4	98.1	56.9	42.2	75.3
	ViT-2B/14	IG-3B	82.2	70.8	97.5	87.3	93.4	81.2	95.1	94.9	57.8	94.4	90.3	7.3	98.2	60.1	50.2	77.4
AIM-0.6B	ViT-H/14	DFN-2B+	78.5	64.0	97.2	86.8	90.1	80.1	93.0	93.0	57.9	94.3	90.0	7.8	98.4	58.3	45.2	75.6
AIM-1B	ViT-1B/14		80.6	67.2	98.2	88.3	91.6	81.8	93.4	93.9	58.6	94.5	90.0	9.0	98.6	59.8	47.5	76.9
AIM-3B	ViT-3B/14		82.2	69.7	98.4	89.9	92.7	81.9	94.1	93.8	58.8	94.3	90.4	9.7	98.5	60.9	48.9	77.6
AIM-7B	ViT-7B/14		82.4	70.9	98.6	90.0	93.1	82.3	93.8	92.1	59.5	93.6	90.7	10.1	98.6	61.7	49.6	77.8
AIM-7B†	ViT-7B/14		DFN-2B+	84.0	75.5	98.9	91.8	94.1	85.6	95.4	95.0	61.4	94.2	90.5	8.4	98.5	63.5	57.7

Table 6. Downstream evaluation with a frozen trunk. We assess the quality of AIM features by evaluating against a diverse set of 15 image recognition benchmarks. AIM and the baseline methods are evaluated using attentive probing with a frozen trunk. AIM models exhibit a strong performance across all benchmarks, especially the AIM-7B. AIM outperforms all other methods, using joint-embedding or generative approaches, except for DINOv2 which utilizes higher-resolution images, that typically results in a 1-1.5% improvement on ImageNet for instance. †: Extracting features from the 20th layer instead of the last (32nd), see Table 7 for more details.

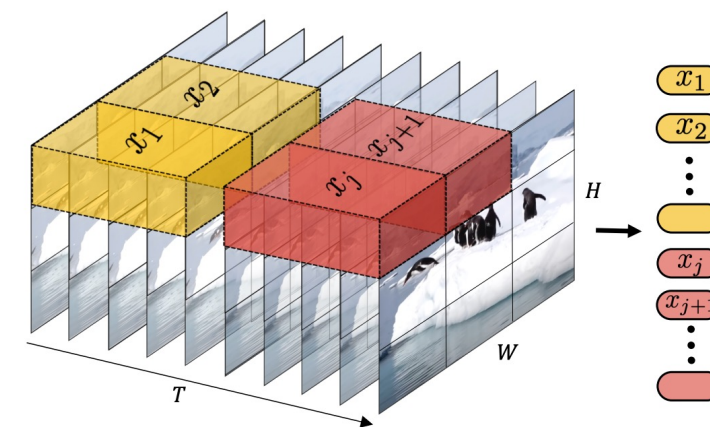
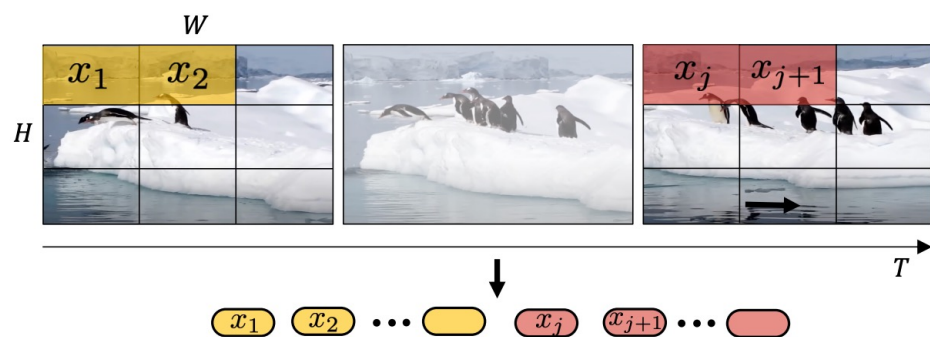
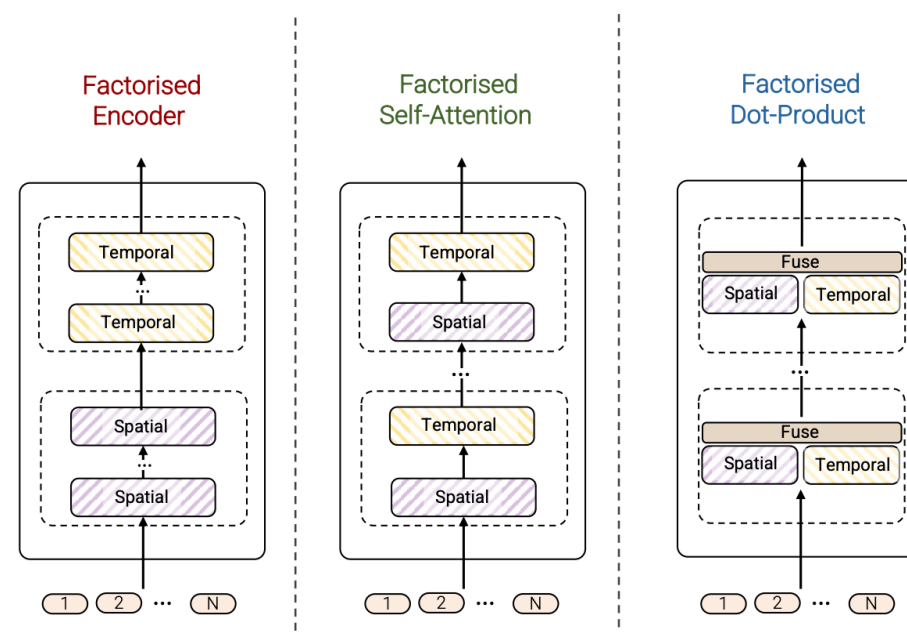
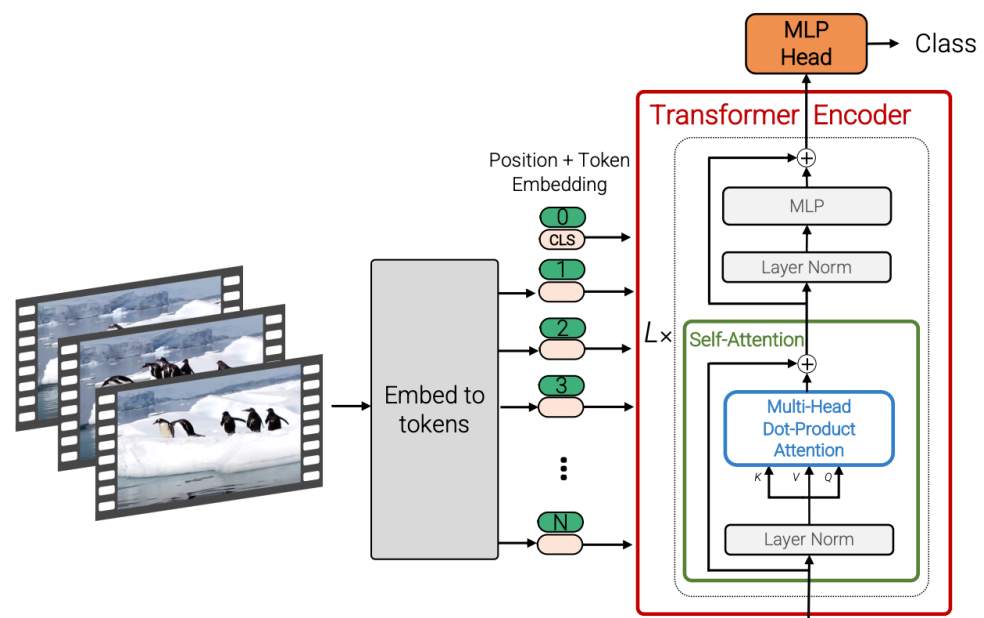


Figure 2: Uniform frame sampling: We simply sample n_t frames, and embed each 2D frame independently following ViT [18].

Figure 3: Tubelet embedding. We extract and linearly embed non-overlapping tubelets that span the spatio-temporal input volume.

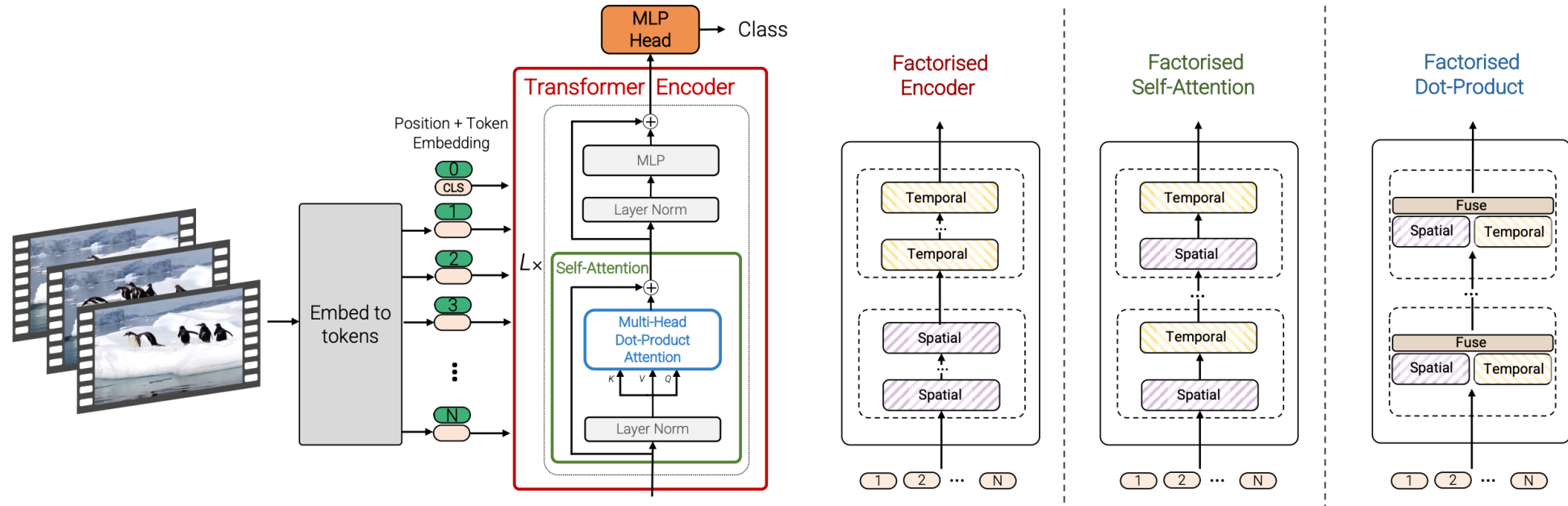


Table 1: Comparison of input encoding methods using ViViT-B and spatio-temporal attention on Kinetics. Further details in text.

	Top-1 accuracy
Uniform frame sampling	78.5
<i>Tubelet embedding</i>	
Random initialisation [25]	73.2
Filter inflation [8]	77.6
Central frame	79.2

Table 2: Comparison of model architectures using ViViT-B as the backbone, and tubelet size of 16×2 . We report Top-1 accuracy on Kinetics 400 (K400) and action accuracy on Epic Kitchens (EK). Runtime is during inference on a TPU-v3.

	K400	EK	FLOPs ($\times 10^9$)	Params ($\times 10^6$)	Runtime (ms)
Model 1: Spatio-temporal	80.0	43.1	455.2	88.9	58.9
Model 2: Fact. encoder	78.8	43.7	284.4	115.1	17.4
Model 3: Fact. self-attention	77.4	39.1	372.3	117.3	31.7
Model 4: Fact. dot product	76.3	39.5	277.1	88.9	22.9
Model 2: Ave. pool baseline	75.8	38.8	283.9	86.7	17.3

VideoMAE

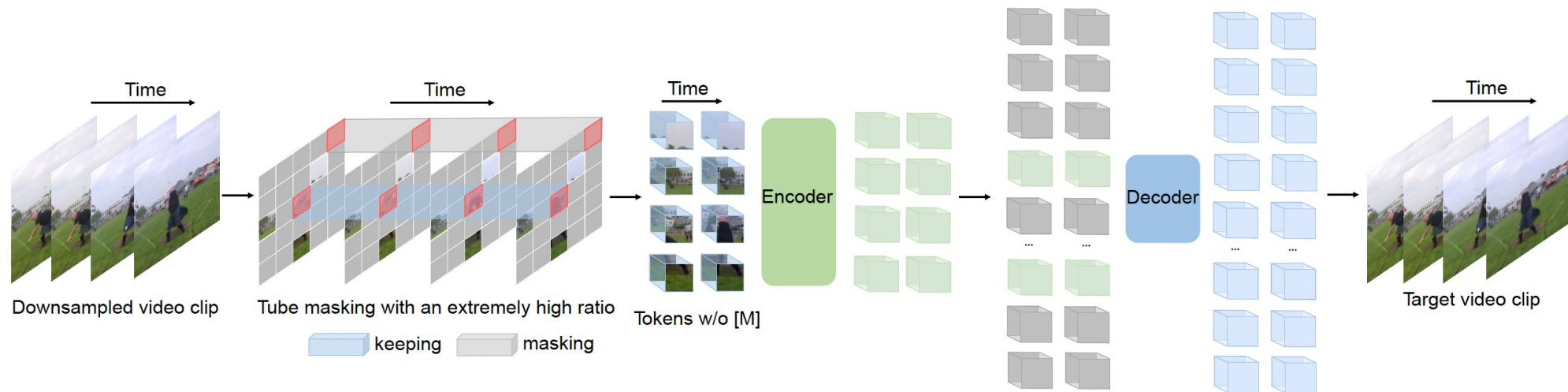


Figure 1: **VideoMAE** performs the task of masking random cubes and reconstructing the missing ones with an asymmetric encoder-decoder architecture. Due to high redundancy and temporal correlation in videos, we present the customized design of tube masking with an extremely high ratio (90% to 95%). This simple design enables us to create a more challenging and meaningful self-supervised task to make the learned representations capture more useful spatiotemporal structures.

Vision-Language Models

Reference: Bordes et al., “An Introduction to Vision-Language Modeling”, 2024.
<https://arxiv.org/pdf/2405.17247>

Searching for Ambiguous Objects in Videos using Relational Referring Expressions

Earlier Attempts:

Hazan Anayurt*
hazan.anayurt@metu.edu.tr
Sezai Artun Ozyegin*

Department of Computer Engineering,
Middle East Technical University,
Ankara, Turkey 2019

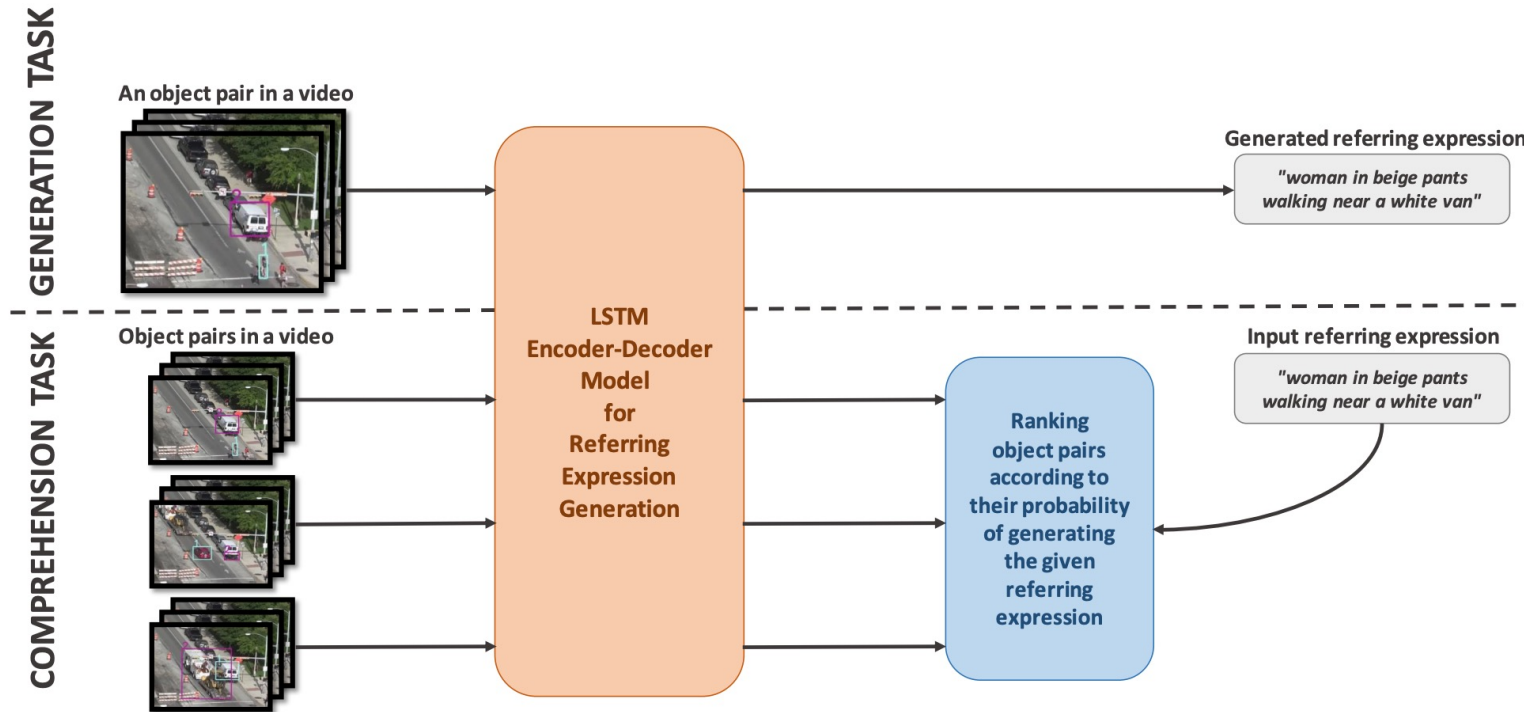


Figure 1: An overview of the generation and comprehension tasks performed by our model.

Earlier Attempts:

ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks



2019

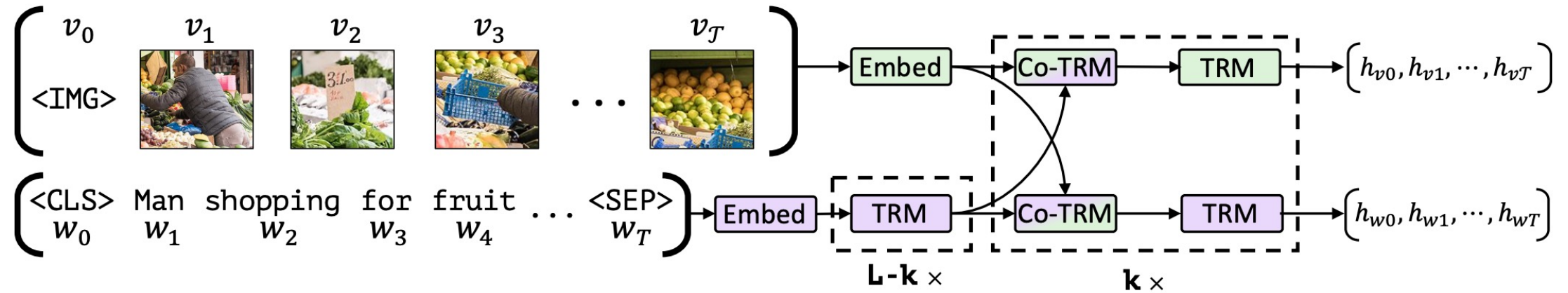


Figure 1: Our ViLBERT model consists of two parallel streams for visual (green) and linguistic (purple) processing that interact through novel co-attentional transformer layers. This structure allows for variable depths for each modality and enables sparse interaction through co-attention. Dashed boxes with multiplier subscripts denote repeated blocks of layers.

Co-TRM: Co-attentional Transformer

Earlier Attempts:

ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks



2019

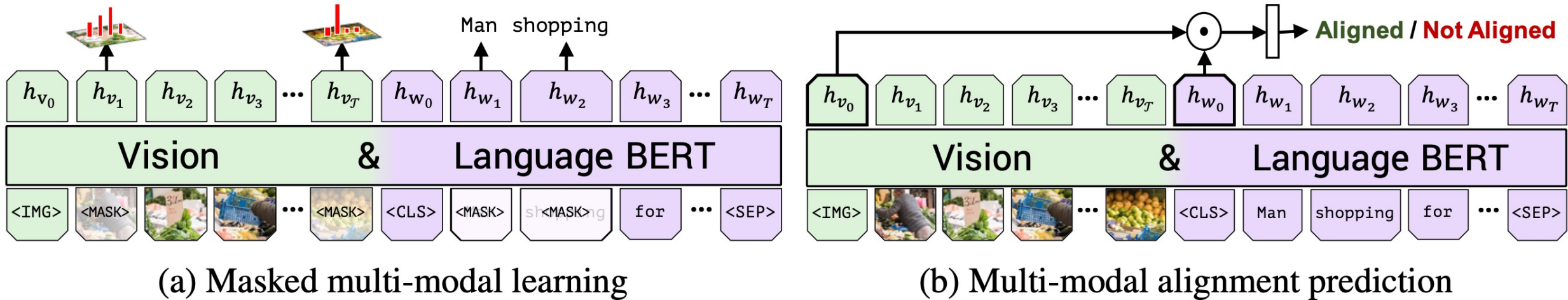


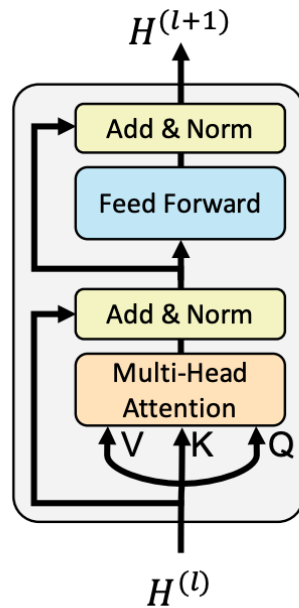
Figure 3: We train ViLBERT on the Conceptual Captions [24] dataset under two training tasks to learn visual grounding. In masked multi-modal learning, the model must reconstruct image region categories or words for masked inputs given the observed inputs. In multi-modal alignment prediction, the model must predict whether or not the caption describes the image content.

Earlier Attempts:

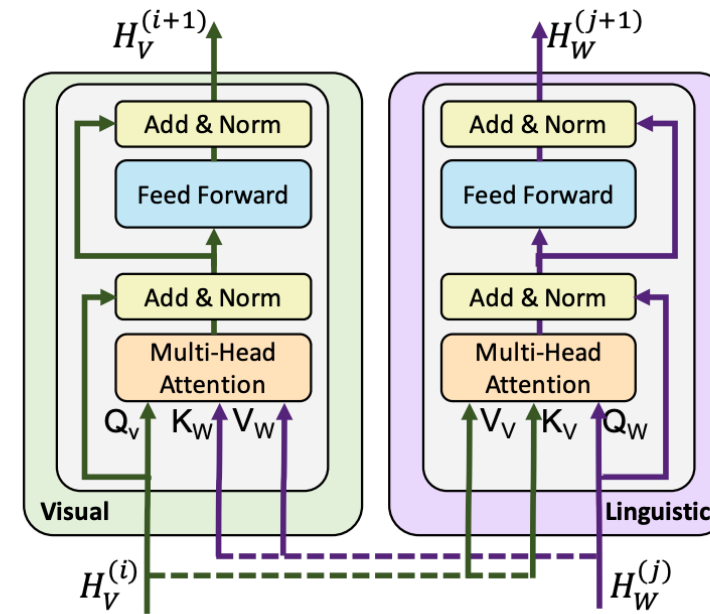
ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks



2019



(a) Standard encoder transformer block



(b) Our co-attention transformer layer

Figure 2: We introduce a novel co-attention mechanism based on the transformer architecture. By exchanging key-value pairs in multi-headed attention, this structure enables vision-attended language features to be incorporated into visual representations (and vice versa).

VISUALBERT: A SIMPLE AND PERFORMANT BASELINE FOR VISION AND LANGUAGE

Earlier Attempts:

Liunian Harold Li[†], Mark Yatskar^{*}, Da Yin[°], Cho-Jui Hsieh[†] & Kai-Wei Chang[†]

[†]University of California, Los Angeles

^{*}Allen Institute for Artificial Intelligence

[°]Peking University

2019



A person hits a ball with a tennis racket

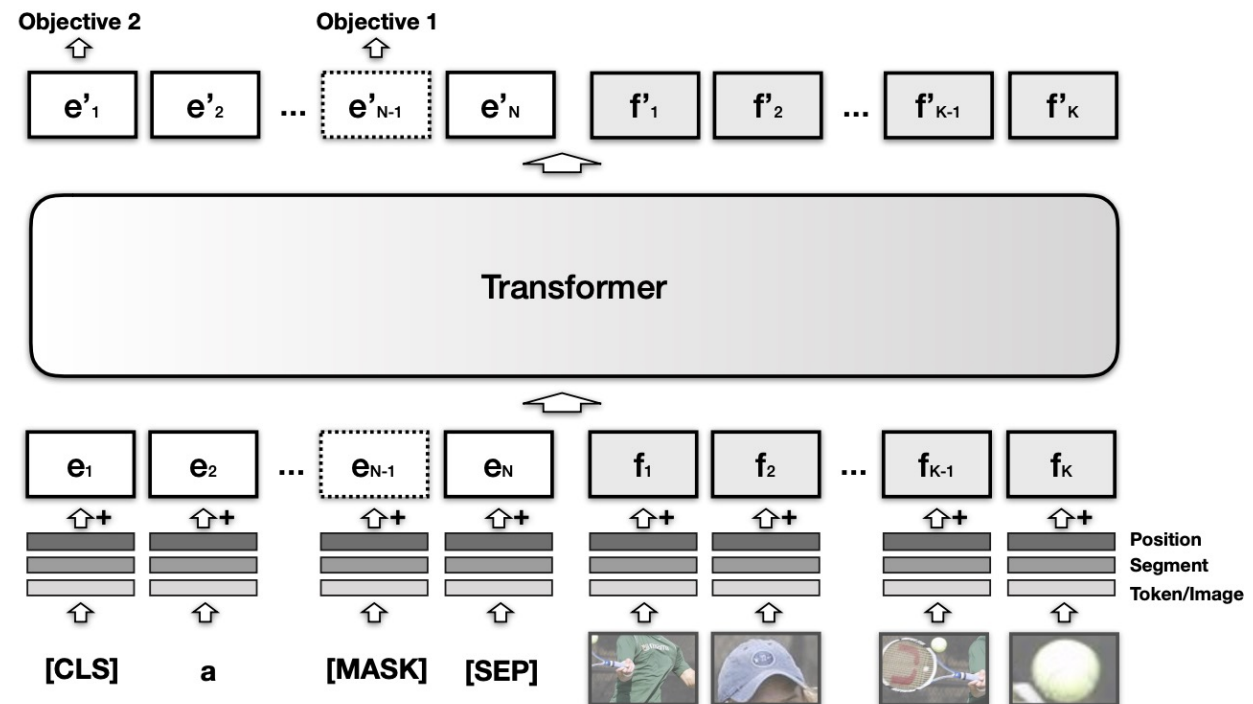


Figure 2: The architecture of VisualBERT. Image regions and language are combined with a Transformer to allow the self-attention to discover implicit alignments between language and vision. It is pre-trained with a masked language modeling (Objective 1), and sentence-image prediction task (Objective 2), on caption data and then fine-tuned for different tasks. See §3.3 for more details.

Overview

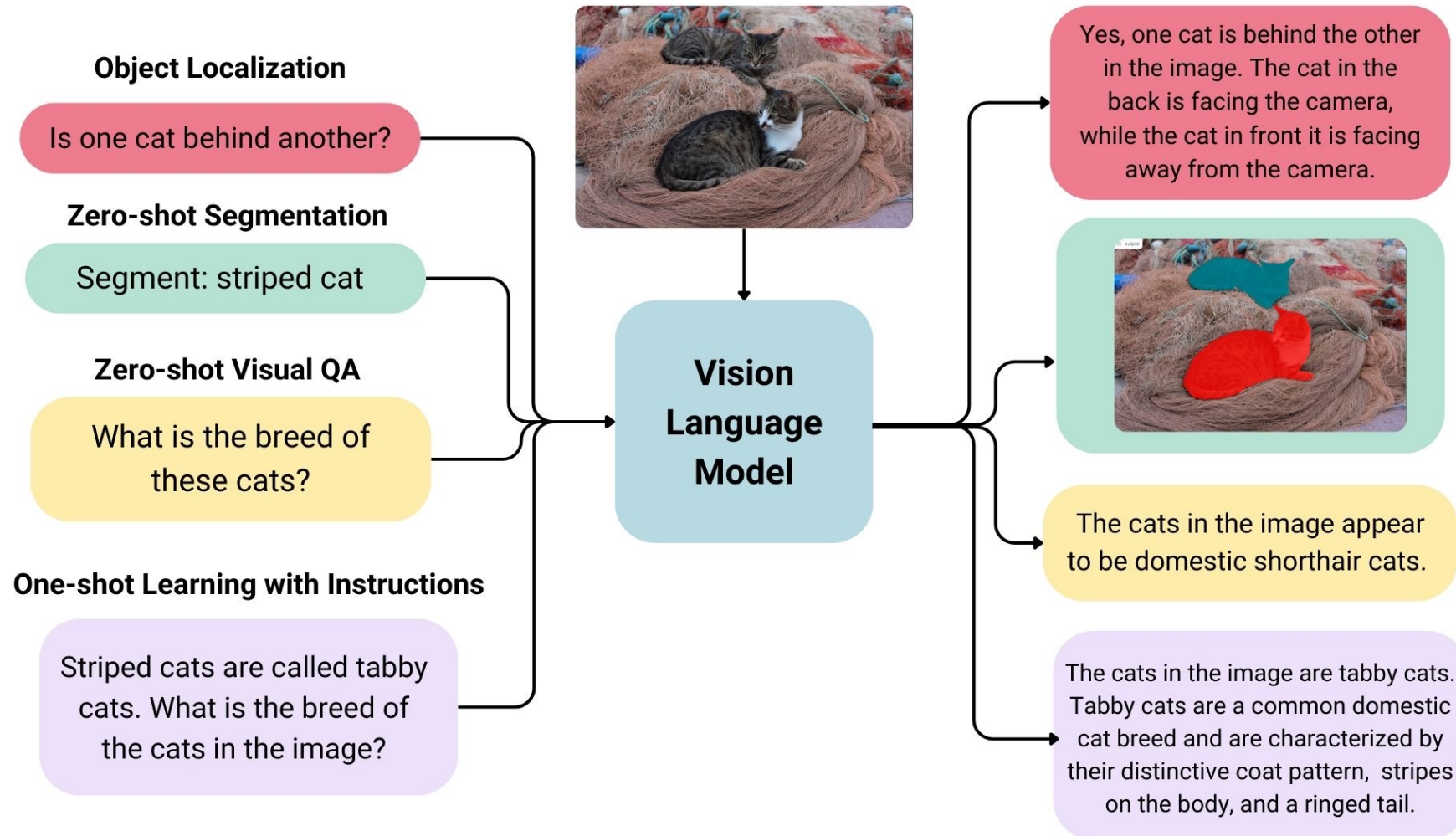
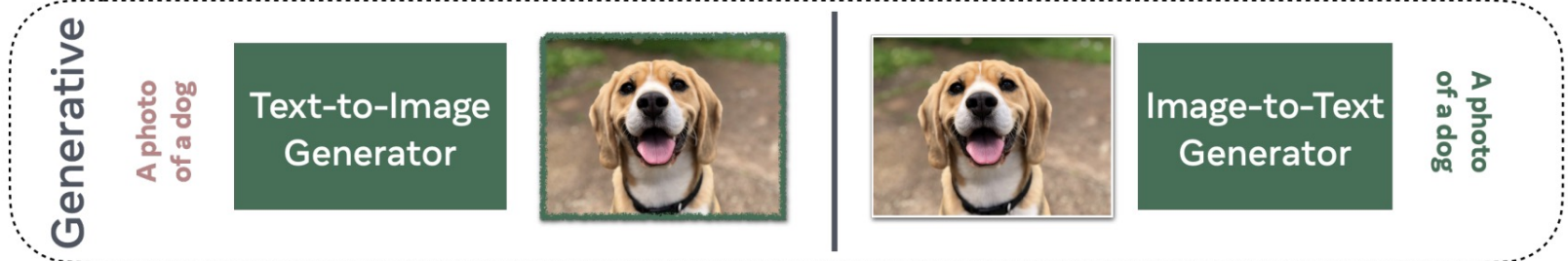
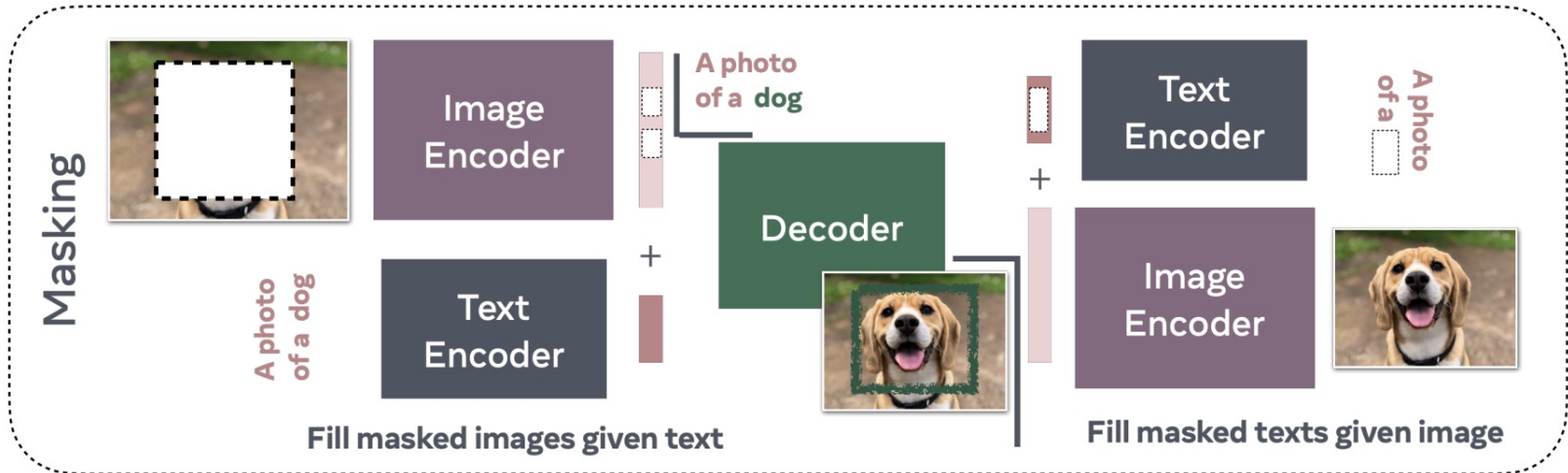
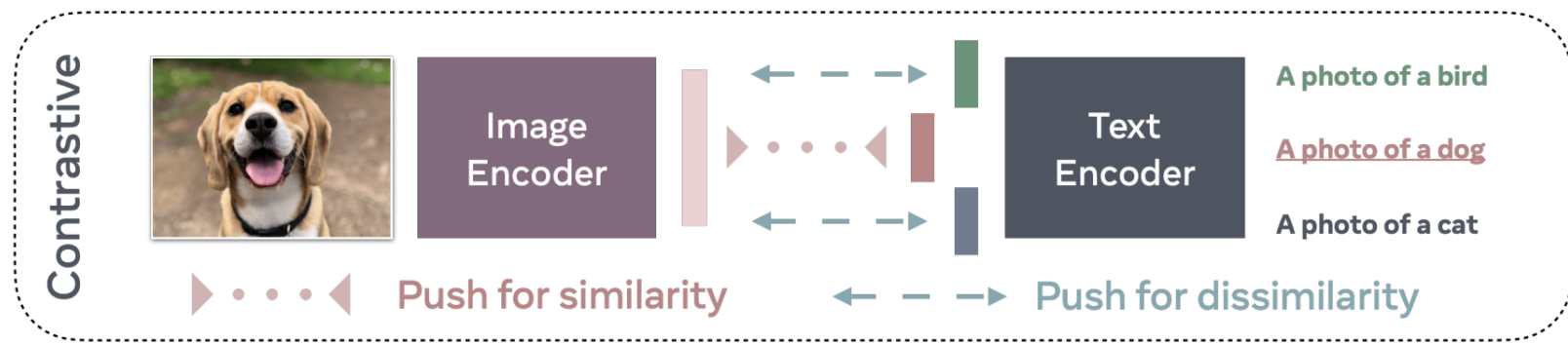


Fig: <https://huggingface.co/blog/vlms>

Overview



Bordes et al., “An Introduction to Vision-Language Modeling”, 2024.
<https://arxiv.org/pdf/2405.17247>

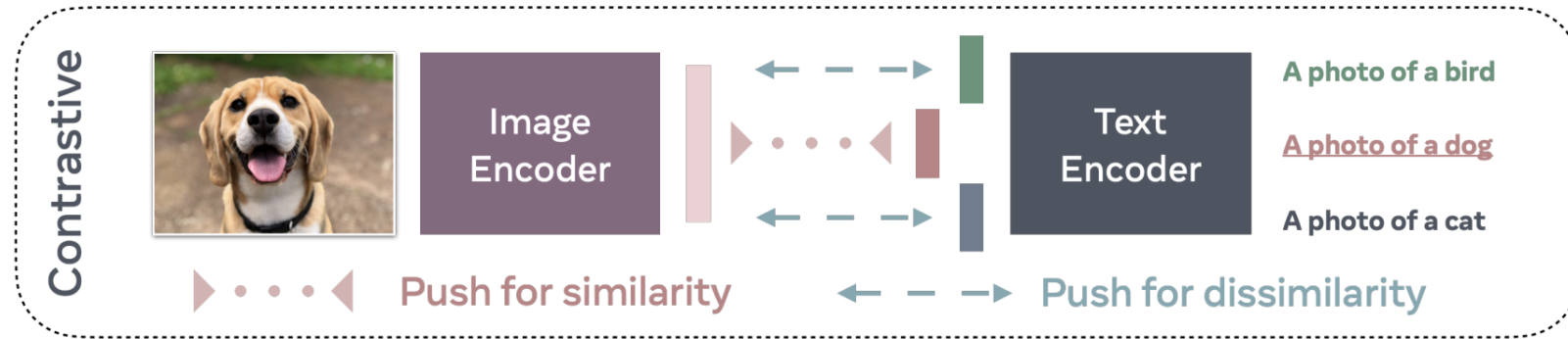


Fig: Bordes et al., "An Introduction to Vision-Language Modeling", 2024.

Contrastive Approaches

CLIP

Alec Radford^{*1} Jong Wook Kim^{*1} Chris Hallacy¹ Aditya Ramesh¹ Gabriel Goh¹ Sandhini Agarwal¹
Girish Sastry¹ Amanda Askell¹ Pamela Mishkin¹ Jack Clark¹ Gretchen Krueger¹ Ilya Sutskever¹

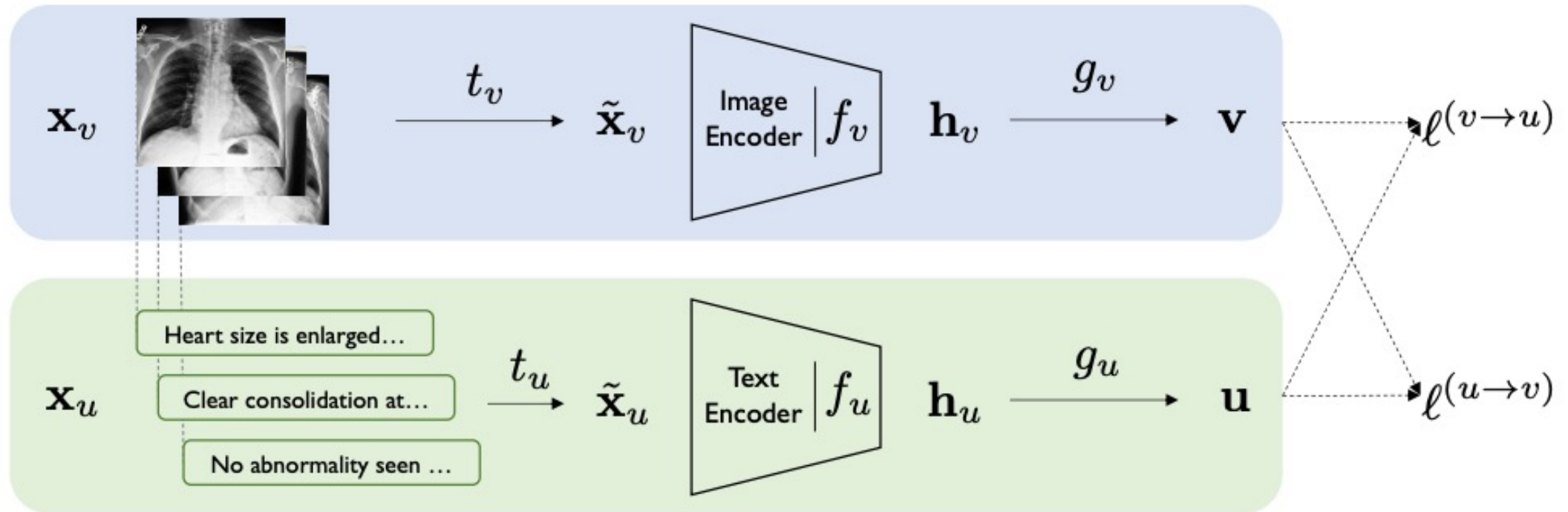
2021

- Motivation:
 - NLP models have benefited significantly from large amounts of web data
 - “Could scalable pre-training methods which learn directly from web text result in a similar breakthrough in computer vision?”
- Contribution:
 - Using natural language supervision for image representation learning at large scale
 - 400M pairs of (image, text)
 - Adapted from ConVIRT (train from scratch)

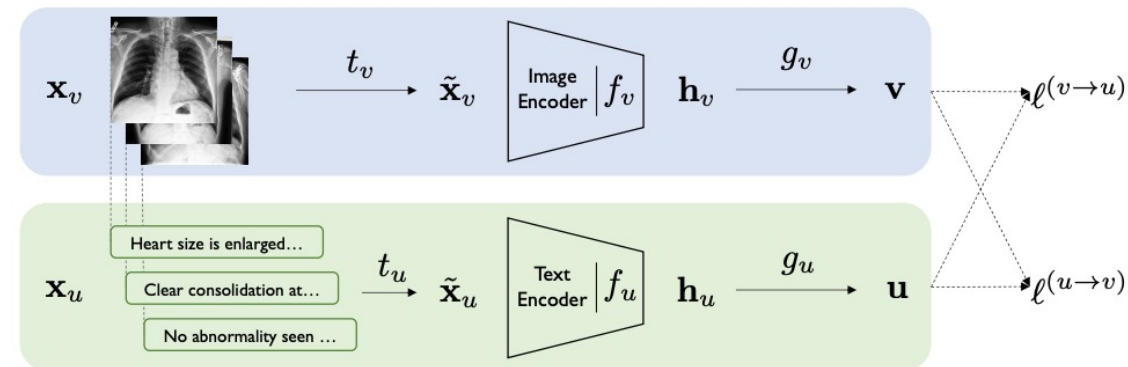
ConVIRT

Contrastive Learning of Medical Visual Representations from Paired Images and Text

Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D. Manning, Curtis P. Langlotz



ConVIRT



The training objective of ConVIRT involves two loss functions. The first loss function is an image-to-text contrastive loss for the i -th pair:

$$\ell_i^{(v \rightarrow u)} = -\log \frac{\exp(\langle \mathbf{v}_i, \mathbf{u}_i \rangle / \tau)}{\sum_{k=1}^N \exp(\langle \mathbf{v}_i, \mathbf{u}_k \rangle / \tau)}, \quad (2)$$

where $\langle \mathbf{v}_i, \mathbf{u}_i \rangle$ represents the cosine similarity, i.e., $\langle \mathbf{v}, \mathbf{u} \rangle = \mathbf{v}^\top \mathbf{u} / \|\mathbf{v}\| \|\mathbf{u}\|$; and $\tau \in \mathbb{R}^+$ represents a temperature parameter. This loss takes the same form as the InfoNCE loss

unlike previous work which use a contrastive loss between inputs of the same modality (Chen et al., 2020a; He et al., 2020), our image-to-text contrastive loss is asymmetric for each input modality. We therefore define a similar text-to-image contrastive loss as:

$$\ell_i^{(u \rightarrow v)} = -\log \frac{\exp(\langle \mathbf{u}_i, \mathbf{v}_i \rangle / \tau)}{\sum_{k=1}^N \exp(\langle \mathbf{u}_i, \mathbf{v}_k \rangle / \tau)}. \quad (3)$$

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \left(\lambda \ell_i^{(v \rightarrow u)} + (1 - \lambda) \ell_i^{(u \rightarrow v)} \right), \quad (4)$$

where $\lambda \in [0, 1]$ is a scalar weight.

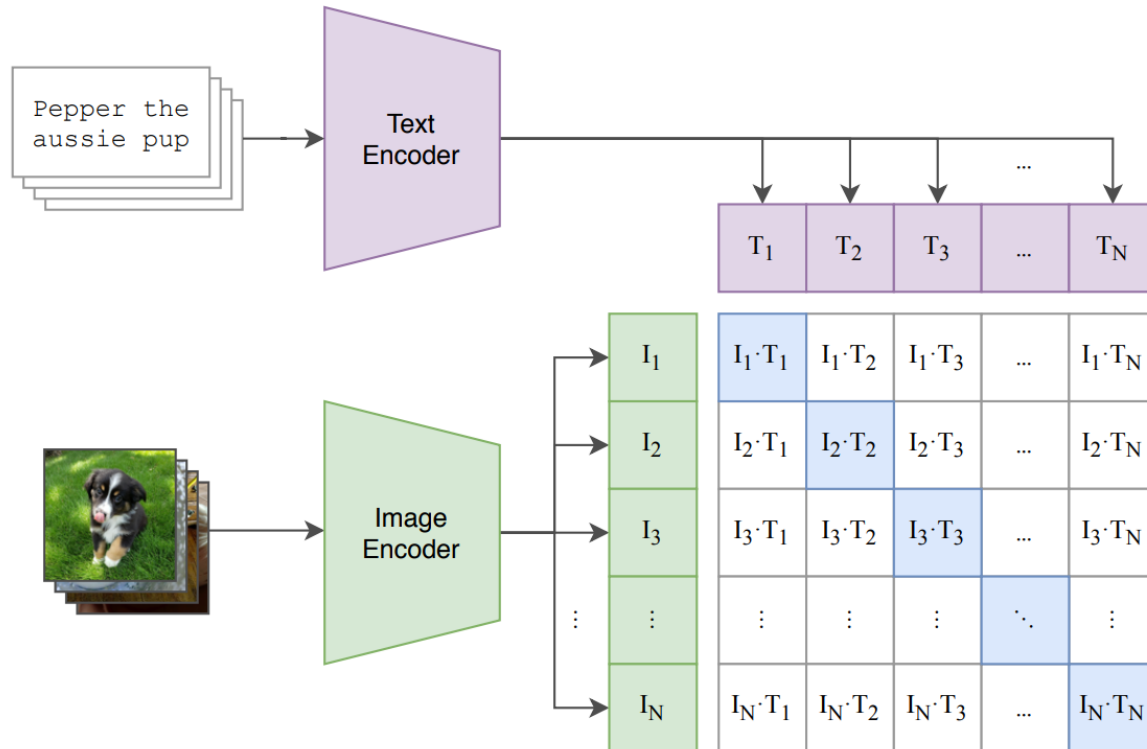
Learning Transferable Visual Models From Natural Language Supervision

CLIP

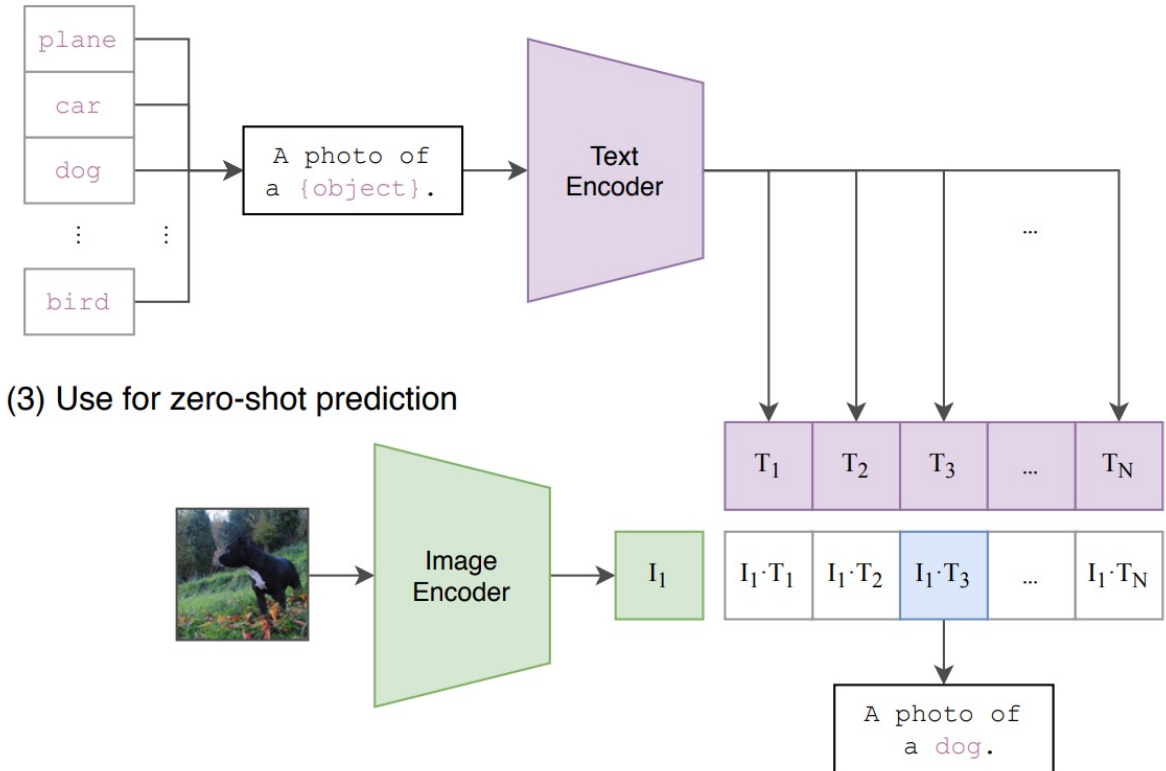
Alec Radford^{*1} Jong Wook Kim^{*1} Chris Hallacy¹ Aditya Ramesh¹ Gabriel Goh¹ Sandhini Agarwal¹
Girish Sastry¹ Amanda Askell¹ Pamela Mishkin¹ Jack Clark¹ Gretchen Krueger¹ Ilya Sutskever¹

2021

(1) Contrastive pre-training



(2) Create dataset classifier from label text



CLIP: Dataset

- MS-COCO, Visual Genome, .. are limited in size
- YFCC100M: Includes 100M images but metadata for each image is sparse and low quality. When filtered, the # of images is around the size of ImageNet
- NL supervision requires a large-scale dataset
- Search (image, text) pairs with text including 500K base queries:
 - “The base query list is all words occurring at least 100 times in the English version of Wikipedia. This is augmented with bi-grams with high pointwise mutual information as well as the names of all Wikipedia articles above a certain search volume. Finally all WordNet synsets not already in the query list are added.”
- To balance, 20K (image, text) pairs are included for each query.
- 400M is similar to the size of WebText used to train GPT-2

CLIP: Efficiency

amounts of compute. Mahajan et al. (2018) required 19 GPU years to train their ResNeXt101-32x48d and Xie et al. (2020) required 33 TPUv3 core-years to train their Noisy Student EfficientNet-L2. When considering that both these systems were trained to predict only 1000 ImageNet classes, the task of learning an open set of visual concepts from natural language seems daunting. In the course of our ef-

Our initial approach, similar to VirTex, jointly trained an image CNN and text transformer from scratch to predict the caption of an image. However, we encountered difficulties efficiently scaling this method. In Figure 2 we show that a 63 million parameter transformer language model, which already uses twice the compute of its ResNet-50 image encoder, learns to recognize ImageNet classes three times slower than a much simpler baseline that predicts a bag-of-words encoding of the same text.

Both these approaches share a key similarity. They try to predict the *exact* words of the text accompanying each image. This is a difficult task due to the wide variety of descriptions, comments, and related text that co-occur with images. Recent work in contrastive representation learning for images has found that contrastive objectives can learn better representations than their equivalent predictive objective (Tian

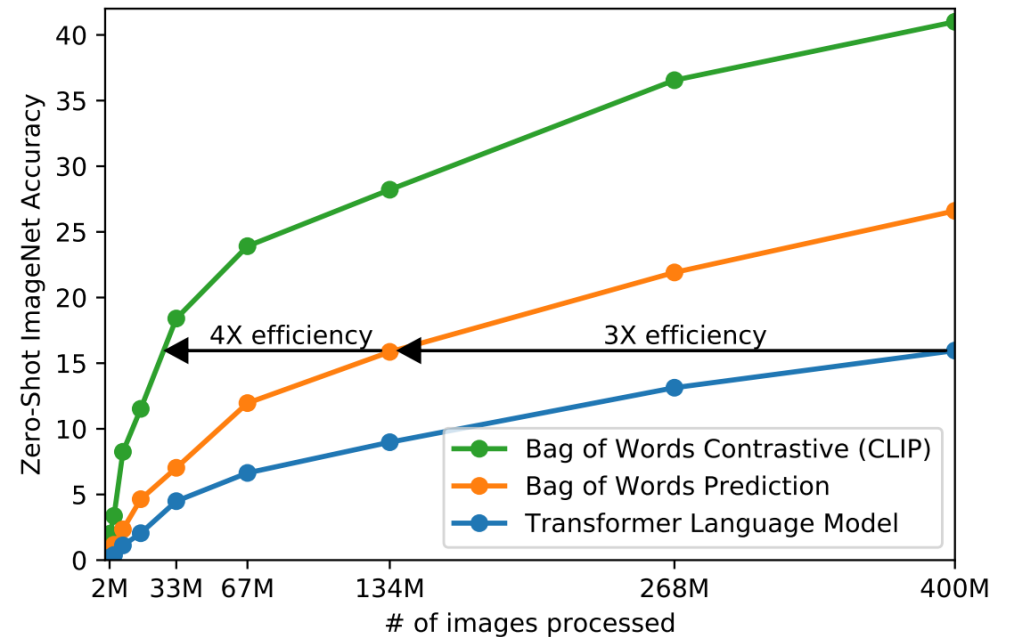


Figure 2. CLIP is much more efficient at zero-shot transfer than our image caption baseline. Although highly expressive, we found that transformer-based language models are relatively weak at zero-shot ImageNet classification. Here, we see that it learns 3x slower than a baseline which predicts a bag-of-words (BoW) encoding of the text (Joulin et al., 2016). Swapping the prediction objective for the contrastive objective of CLIP further improves efficiency another 4x.

CLIP: Method

$$\mathcal{L}_{\text{infoNCE}} = - \sum_{(i,j) \in \mathbb{P}} \log \left(\frac{e^{\text{CoSim}(\mathbf{z}_i, \mathbf{z}_j) / \tau}}{\sum_{k=1}^N e^{\text{CoSim}(\mathbf{z}_i, \mathbf{z}_k) / \tau}} \right),$$

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

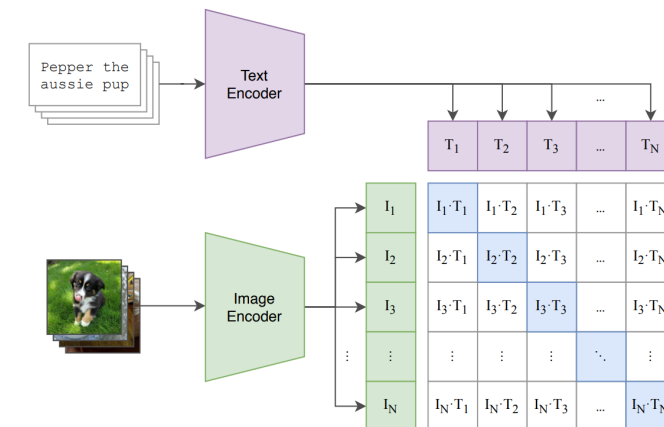
# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t) / 2
```

Figure 3. Numpy-like pseudocode for the core of an implementation of CLIP.

CLIP: Architecture

(1) Contrastive pre-training



- Image Encoder

- Option 1: ResNet-D, some modifications over ResNet-50 + antialiased rect-2 blur pooling + **attention pooling** (instead of global average pooling)
- Option 2: ViT + additional layer norm after combining patch and position embeddings

- Text Encoder

- Transformer (same architecture as GPT-2)
- Different sizes (base version: 63M params, 12 layers, 512-wide model with 8 attention heads)
- Lowercased byte-pair encoding
- Max sequence length: Trimmed at 76
- [SOS] and [EOS] tokens

CLIP: Training

- Train 5 ResNets, 3 ViTs:
 - ResNet-50, ResNet-101, 3 More following EfficientNet-style modifications on ResNet-50
 - ViT-B/32, ViT-B/16, ViT-L/14
- 32 epochs, Adam, weight decay (on all weights except for biases), LR with cosine scheduling
- Hyperparams:
 - Grid search, random search, manual tuning for 1 epoch for ResNet-50-baseline
 - Then based on heuristics adapted for larger models
- Temperature: Learnable, initialized at 0.07 and clipped to prevent explosion
- Batchsize: 32,768.
- “The largest ResNet model, RN50x64, took 18 days to train on 592 V100 GPUs while the largest Vision Transformer took 12 days on 256 V100 GPUs.”

CLIP: Results

	aYahoo	ImageNet	SUN
Visual N-Grams	72.4	11.5	23.0
CLIP	98.4	76.2	58.5

Table 1. Comparing CLIP to prior zero-shot transfer image classification results. CLIP improves performance on all three datasets by a large amount. This improvement reflects many differences in the 4 years since the development of Visual N-Grams (Li et al., 2017).

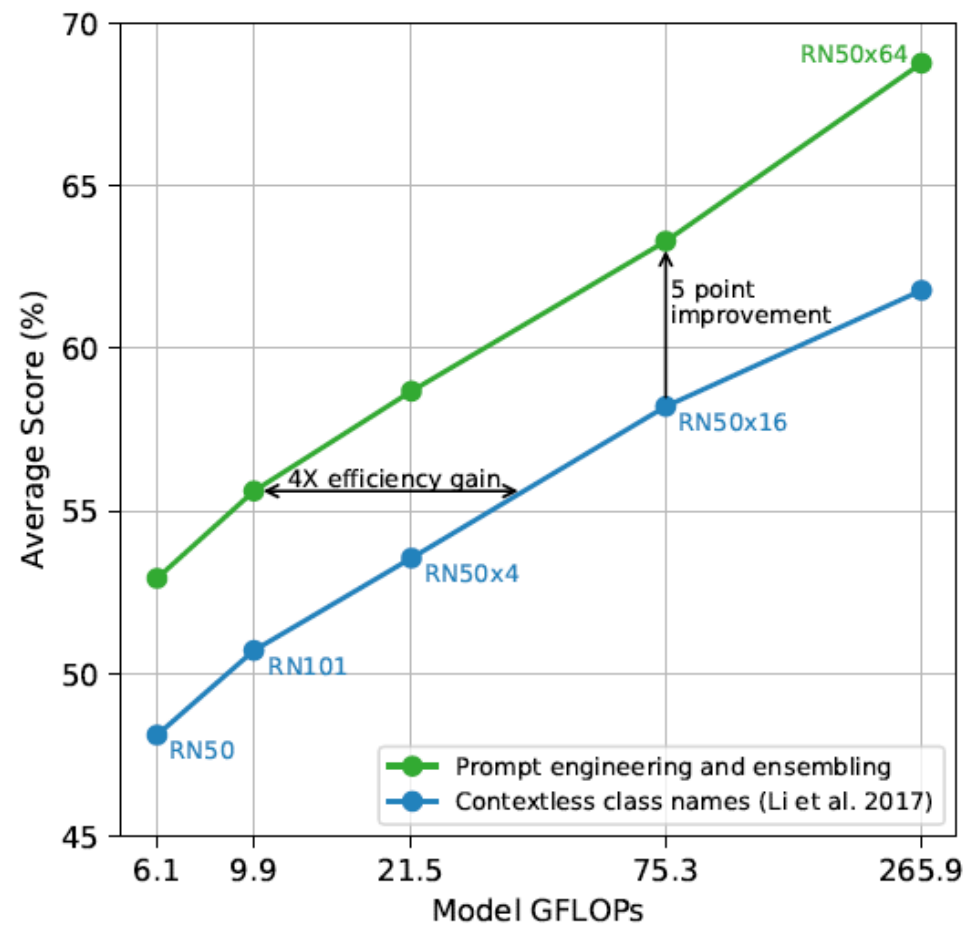


Figure 4. Prompt engineering and ensembling improve zero-shot performance. Compared to the baseline of using contextless class names, prompt engineering and ensembling boost zero-shot classification performance by almost 5 points on average across 36 datasets. This improvement is similar to the gain from using 4 times more compute with the baseline zero-shot method but is “free” when amortized over many predictions.

CLIP: Results

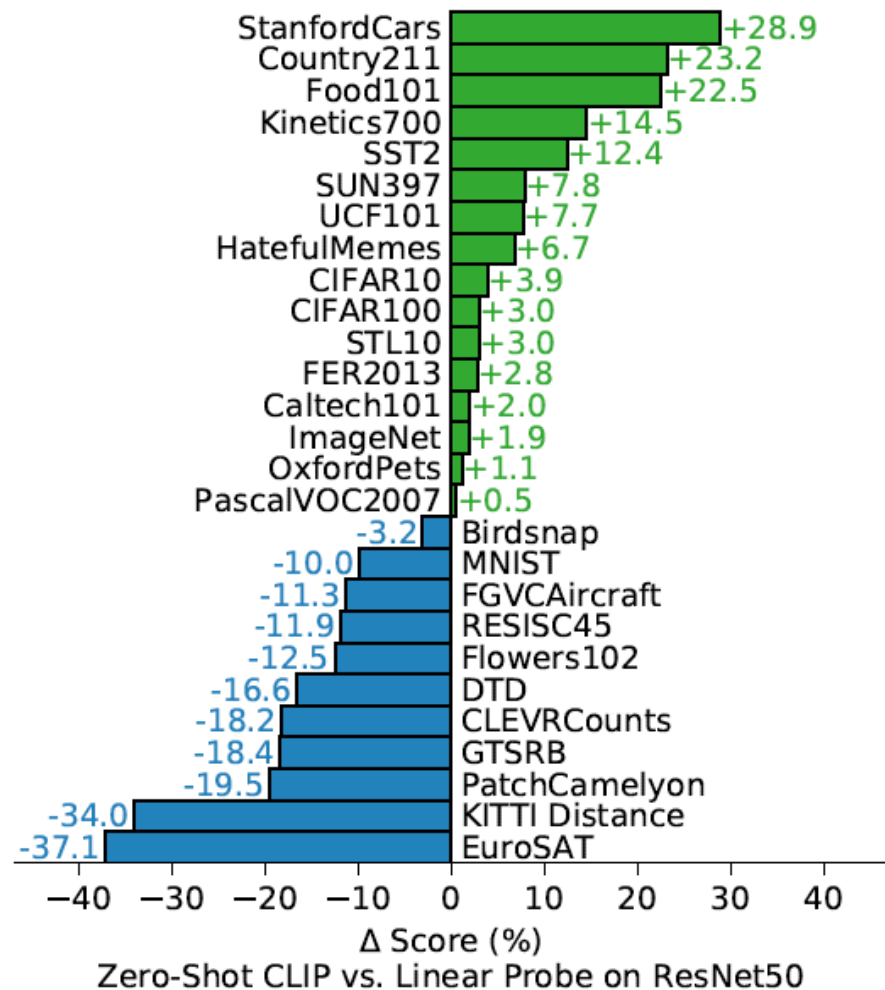


Figure 5. Zero-shot CLIP is competitive with a fully supervised baseline. Across a 27 dataset eval suite, a zero-shot CLIP classifier outperforms a fully supervised linear classifier fitted on ResNet-50 features on 16 datasets, including ImageNet.

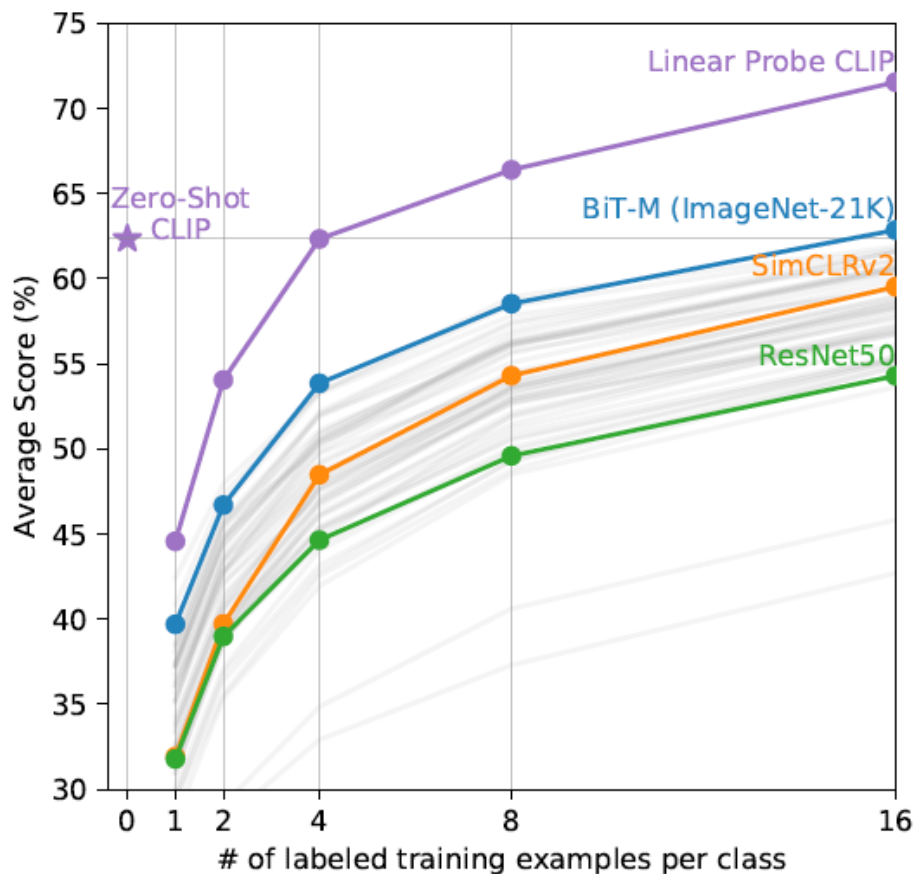


Figure 6. Zero-shot CLIP outperforms few-shot linear probes. Zero-shot CLIP matches the average performance of a 4-shot linear classifier trained on the same feature space and nearly matches the best results of a 16-shot linear classifier across publicly available models. For both BiT-M and SimCLRv2, the best performing model is highlighted. Light gray lines are other models in the eval suite. The 20 datasets with at least 16 examples per class were used in this analysis.

CLIP: Results

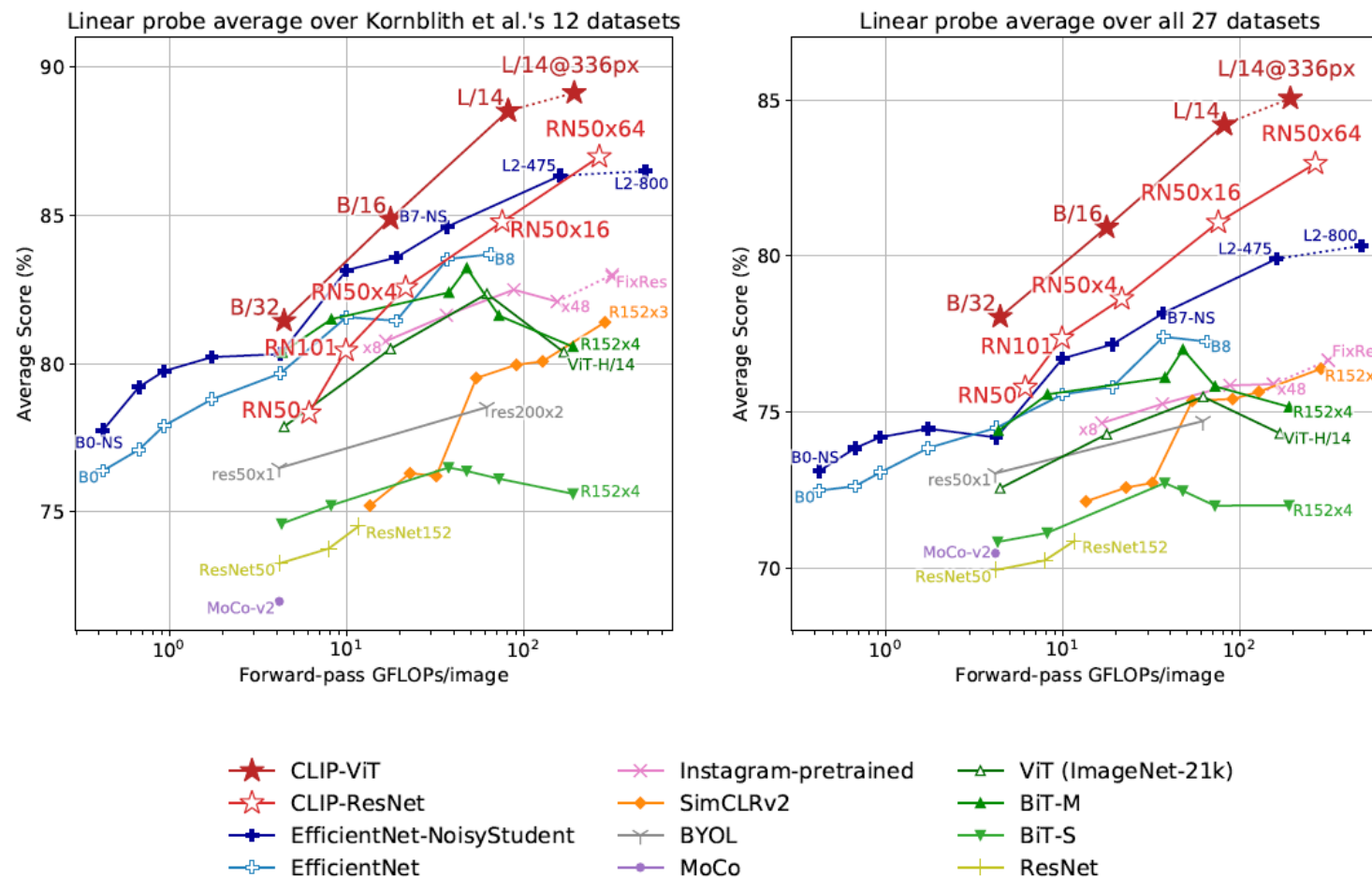


Figure 10. Linear probe performance of CLIP models in comparison with state-of-the-art computer vision models, including EfficientNet (Tan & Le, 2019; Xie et al., 2020), MoCo (Chen et al., 2020d), Instagram-pretrained ResNeXt models (Mahajan et al., 2018; Touvron et al., 2019), BiT (Kolesnikov et al., 2019), ViT (Dosovitskiy et al., 2020), SimCLRv2 (Chen et al., 2020c), BYOL (Grill et al., 2020), and the original ResNet models (He et al., 2016b). (Left) Scores are averaged over 12 datasets studied by Kornblith et al. (2019). (Right) Scores are averaged over 27 datasets that contain a wider variety of distributions. Dotted lines indicate models fine-tuned or evaluated on images at a higher-resolution than pre-training. See Table 10 for individual scores and Figure 20 for plots for each dataset.

CLIP: Results

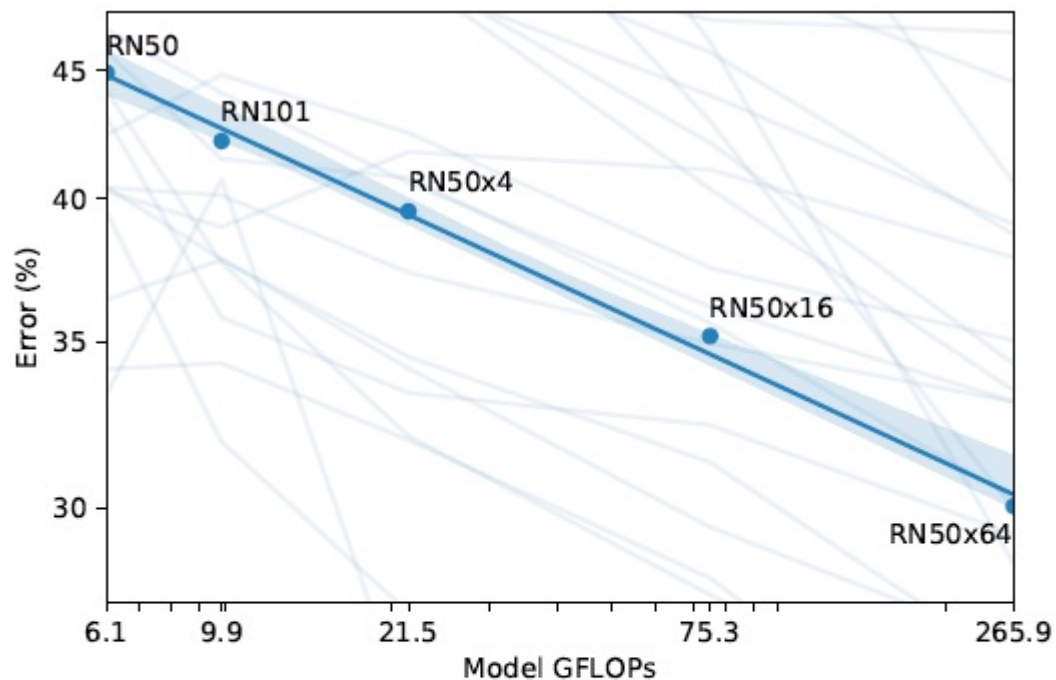


Figure 9. Zero-shot CLIP performance scales smoothly as a function of model compute. Across 39 evals on 36 different datasets, average zero-shot error is well modeled by a log-log linear trend across a 44x range of compute spanning 5 different CLIP models. Lightly shaded lines are performance on individual evals, showing that performance is much more varied despite the smooth overall trend.

CLIP: Extensions

- SigLIP (Zhai et al., 2023)
 - CLIP with NCE Loss with binary CE instead of CLIP's multi-class InfoNCE
 - Better zero-shot performance on smaller batch sizes
- SLIP
 - CLIP combined with Self-Supervised Learning

Add this

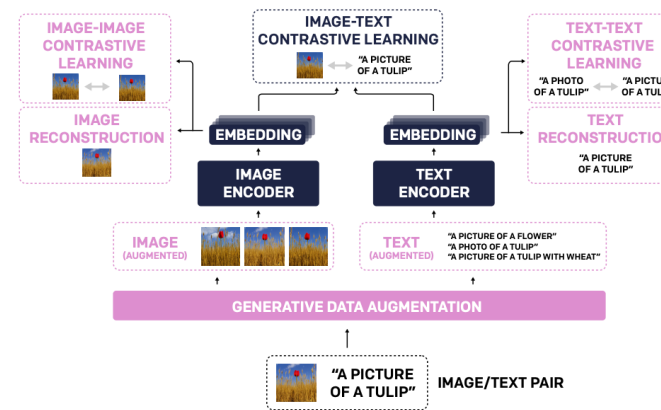
CVJ 19 Mar 2025

TULIP: Towards Unified Language-Image Pretraining

Zineng Tang, Long Lian, Seun Eisaep, XuDong Wang,
Roei Herzig, Adam Yala, Alane Suhr, Trevor Darrell, David M. Chan
University of California, Berkeley

Abstract

Despite the recent success of image-text contrastive models like CLIP and SigLIP, these models often struggle with vision-centric tasks that demand high-fidelity image understanding, such as counting, depth estimation, and fine-grained object recognition. These models, by performing language alignment, tend to prioritize high-level semantics over visual understanding, weakening their image understanding. On the other hand, vision-focused models are great at processing visual information but struggle to understand language, limiting their flexibility for language-



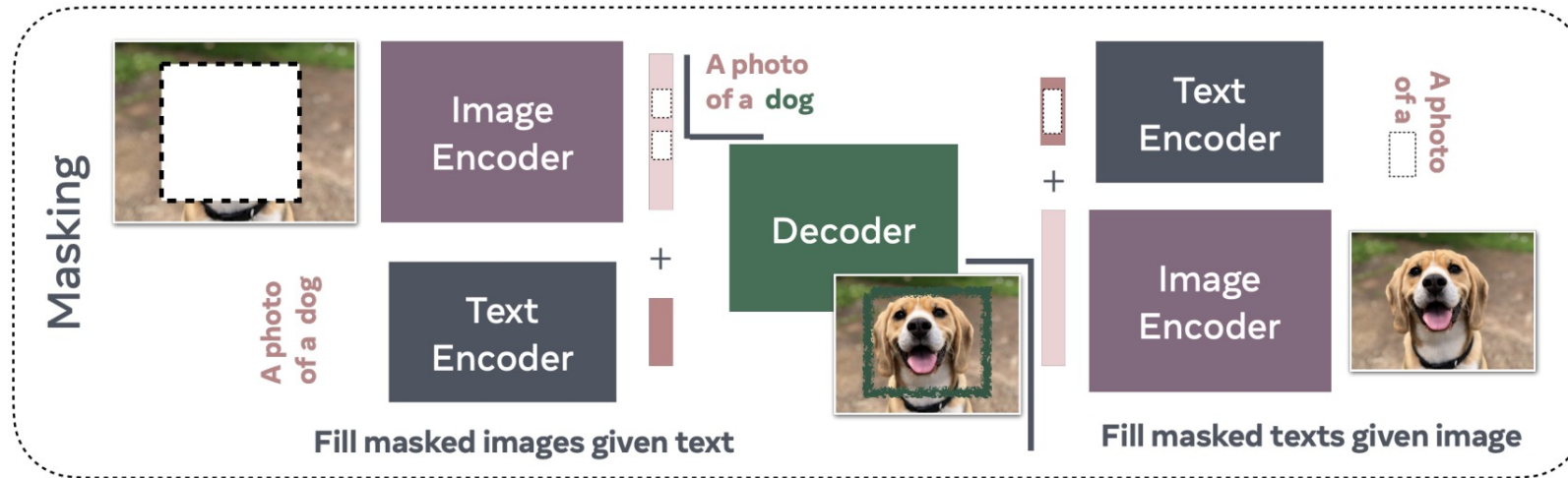


Fig: Bordes et al., "An Introduction to Vision-Language Modeling", 2024.

Masking Approaches

FLAVA

Foundational Language And Vision Alignment (FLAVA)
[Singh et al., 2022].

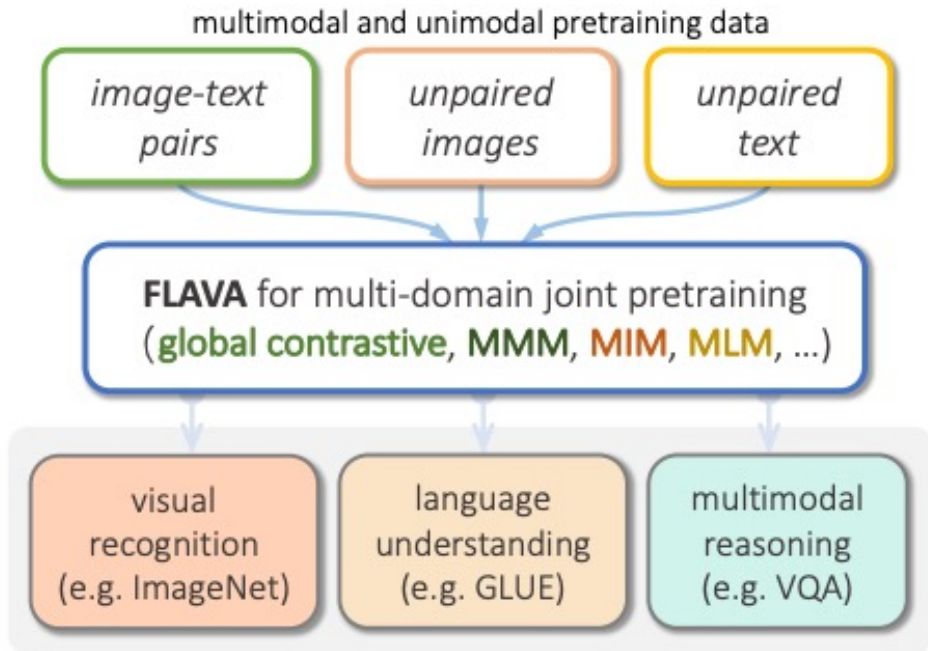


Figure 1. We present FLAVA, a language and vision alignment model that learns strong representations from multimodal (image-text pairs) and unimodal data (unpaired images and text) and can be applied to target a broad scope of tasks from three domains (visual recognition, language understanding, and multimodal reasoning) under a common transformer model architecture.

Method	Multimodal Pretraining data			Pretraining Objectives				Target Modalities			
	public	dataset(s)	size	Contr.	ITM	Masking	Unimodal	V	CV&L	MV&L	L
CLIP [83]	✗	WebImageText	400M	✓	-	-	-	✓	✓	-	-
ALIGN [50]	✗	JFT	1.8B	✓	-	-	-	✓	✓	-	-
SimVLM [109]	✗	JFT	1.8B	-	-	PrefixLM	CLM	*	✓	✓	✓
UniT [43]	-	None	-	-	-	-	-	*	-	✓	✓
VinVL [118]	✓	Combination	9M	✓	-	MLM	-	-	✓	✓	-
ViLT [54]	✓	Combination	10M	-	✓	MLM	-	-	✓	✓	-
ALBEF [62]	✓	Combination	5M	✓	✓	MLM	-	-	✓	✓	-
FLAVA (ours)	✓	PMD (Tbl. 2)	70M	✓	✓	MMM	MLM+MIM	✓	✓	✓	✓

Table 1. Comparison of recent models in different modalities. CV&L and MV&L stands for cross-modal and multi-modal vision-and-language. * means the modality is partially targeted (SimVLM [109] and UniT [43] include ImageNet and object detection, respectively).

FLAVA

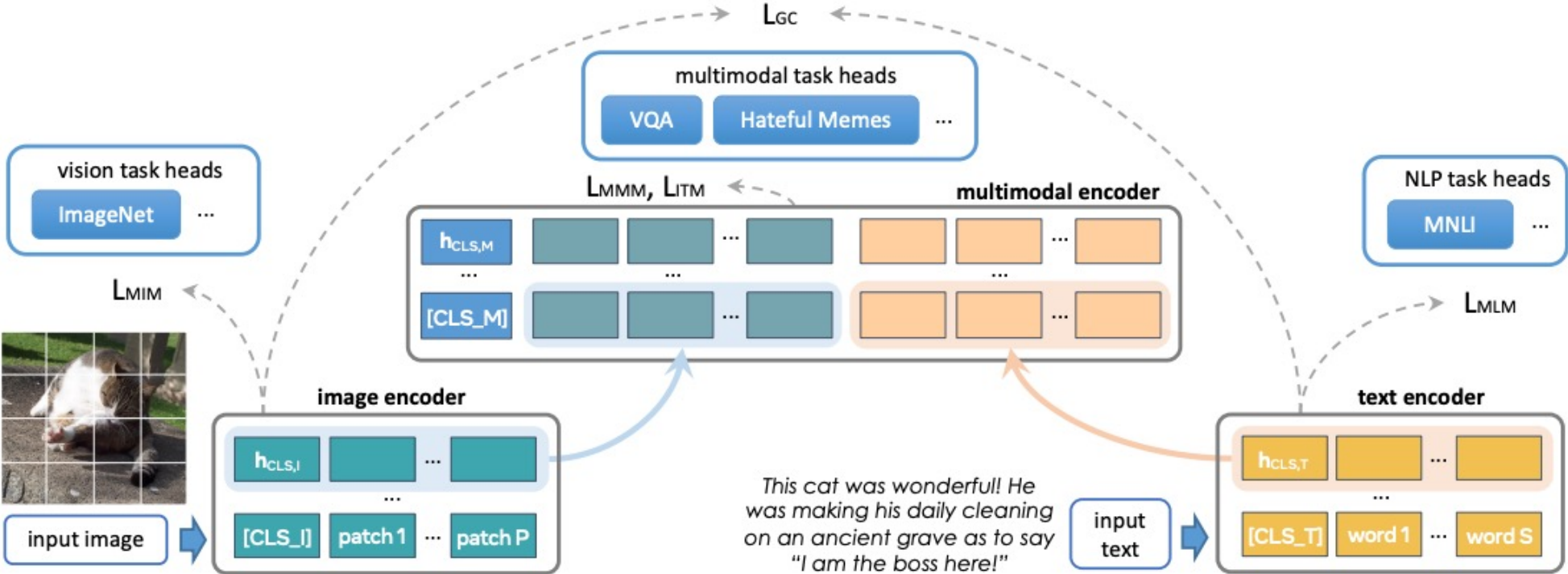


Figure 2. **An overview of our FLAVA model**, with an image encoder transformer to capture unimodal image representations, a text encoder transformer to process unimodal text information, and a multimodal encoder transformer that takes as input the encoded unimodal image and text and integrates their representations for multimodal reasoning. **During pretraining**, masked image modeling (MIM) and mask language modeling (MLM) losses are applied onto the image and text encoders over a single image or a text piece, respectively, while contrastive, masked multimodal modeling (MMM), and image-text matching (ITM) loss are used over paired image-text data. **For downstream tasks**, classification heads are applied on the outputs from the image, text, and multimodal encoders respectively for visual recognition, language understanding, and multimodal reasoning tasks.

FLAVA: Results

Method	Vision Avg.	NLP Avg.	Multi-modal Avg.	Macro Avg.
1 MIM	57.46	–	–	19.15
2 MLM	–	71.55	–	23.85
3 FLAVA _C	64.80	79.14	66.25	70.06
4 FLAVA _{MM}	74.22	79.35	69.11	74.23
5 FLAVA w/o unimodal init	75.55	78.29	67.32	73.72
6 FLAVA	78.19	79.44	69.92	75.85

Table 3. Our full FLAVA pretraining (row 6) achieves the best average scores on vision, language, and multimodal tasks compared to ablations. Row 1 to 4 are pretrained on PMD while row 5 and 6 also involve unimodal IN-1k, CCNews, and BookCorpus datasets.

Perception Encoder

Bolya et al., “Perception Encoder: The best visual embeddings are not at the output of the network”, 2025.

- “Traditionally, vision encoders have relied on a variety of pretraining objectives, each tailored to specific downstream tasks such as classification, captioning, or localization. Surprisingly, after scaling our carefully tuned image pretraining recipe and refining with our robust video data engine, **we find that contrastive vision-language training alone can produce strong, general embeddings for all of these downstream tasks.** There is only one caveat: **these embeddings are hidden within the intermediate layers of the network. To draw them out, we introduce two alignment methods: language alignment for multimodal language modeling, and spatial alignment for dense prediction.**”

Perception Encoder

Bolya et al., “Perception Encoder: The best visual embeddings are not at the output of the network”, 2025.

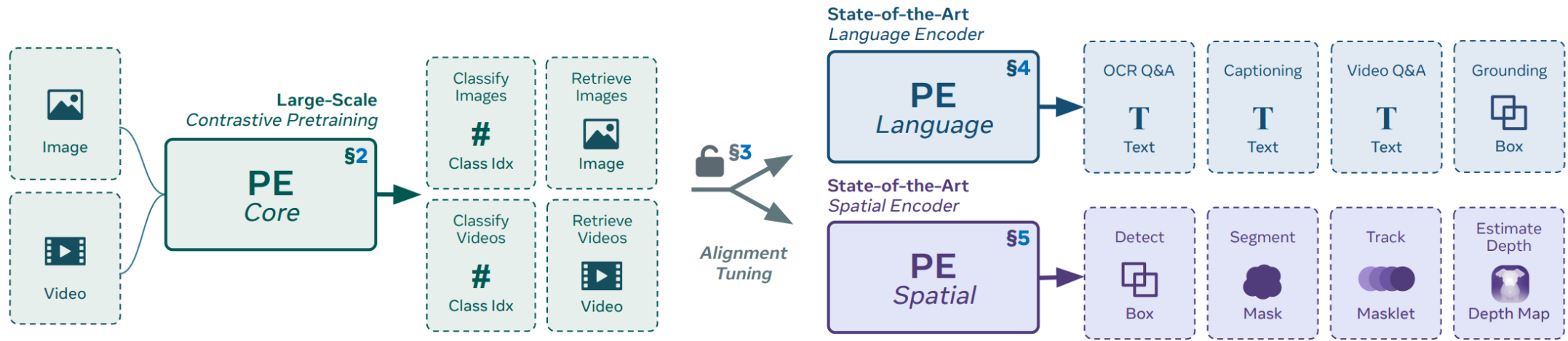


Figure 1 Perception Encoder (PE) is a family of large-scale vision encoder models with state-of-the-art performance on a large variety of vision tasks. By using a robust contrastive pretraining recipe and finetuning on synthetically aligned videos, PE not only outperforms all existing models on classification and retrieval (§2), but it also internally produces strong, general features that *scale* for downstream tasks (§3). PE unlocks the ability for large-scale contrastive pretraining to transfer to downstream tasks with alignment tuning to capitalize on those general features (§4, §5).

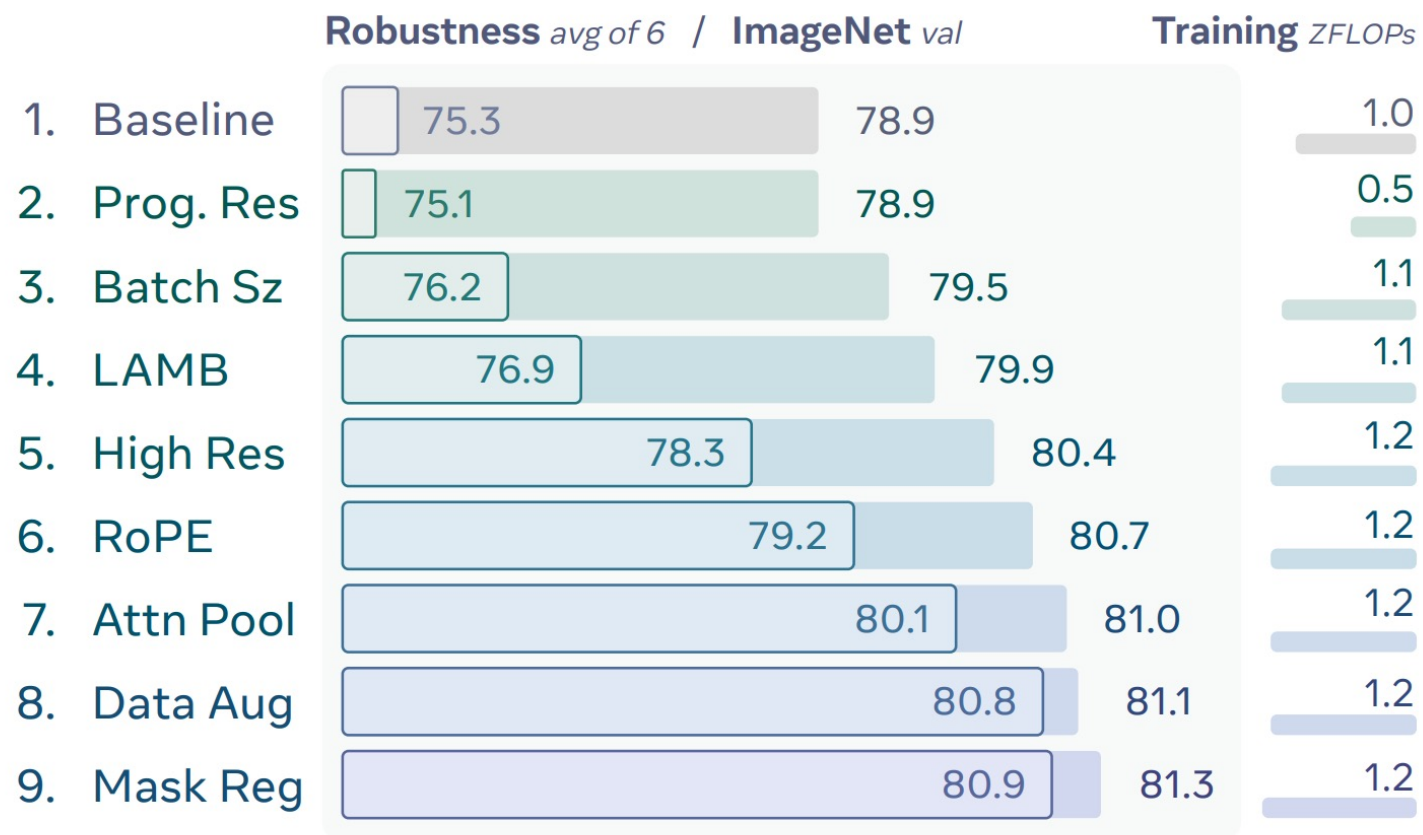


Figure 2 Robust Image Pretraining. We tune our pretraining recipe (§2.1) to maximize performance on a fixed set of data, starting with an OpenCLIP [51] ViT-L/14 model. We report cumulative zero-shot classification results for each modification. The inner bars show robustness evaluation, calculated as the average of 6 robustness benchmarks [4, 26, 46, 47, 112, 143], and the outer bars show ImageNet val [26] alone. Several changes significantly improve robustness, indicating that ImageNet val scales more with data, while robustness can scale with refined training techniques.

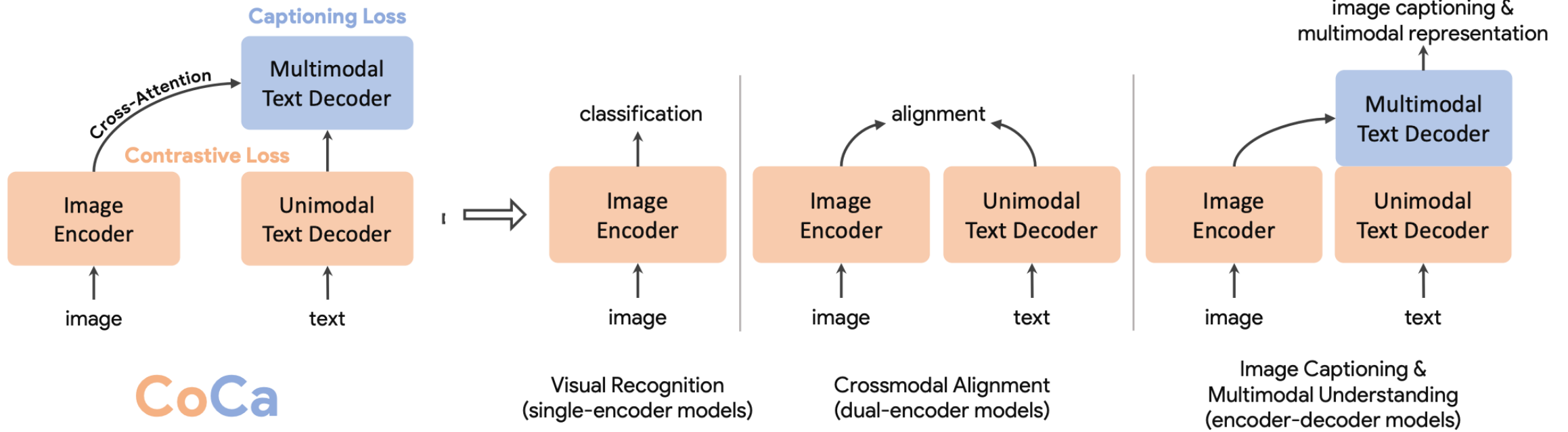


Fig: Bordes et al., "An Introduction to Vision-Language Modeling", 2024.

Generative Approaches

CoCa

Contrastive Captioner (CoCa),
Yu et al., 2022.

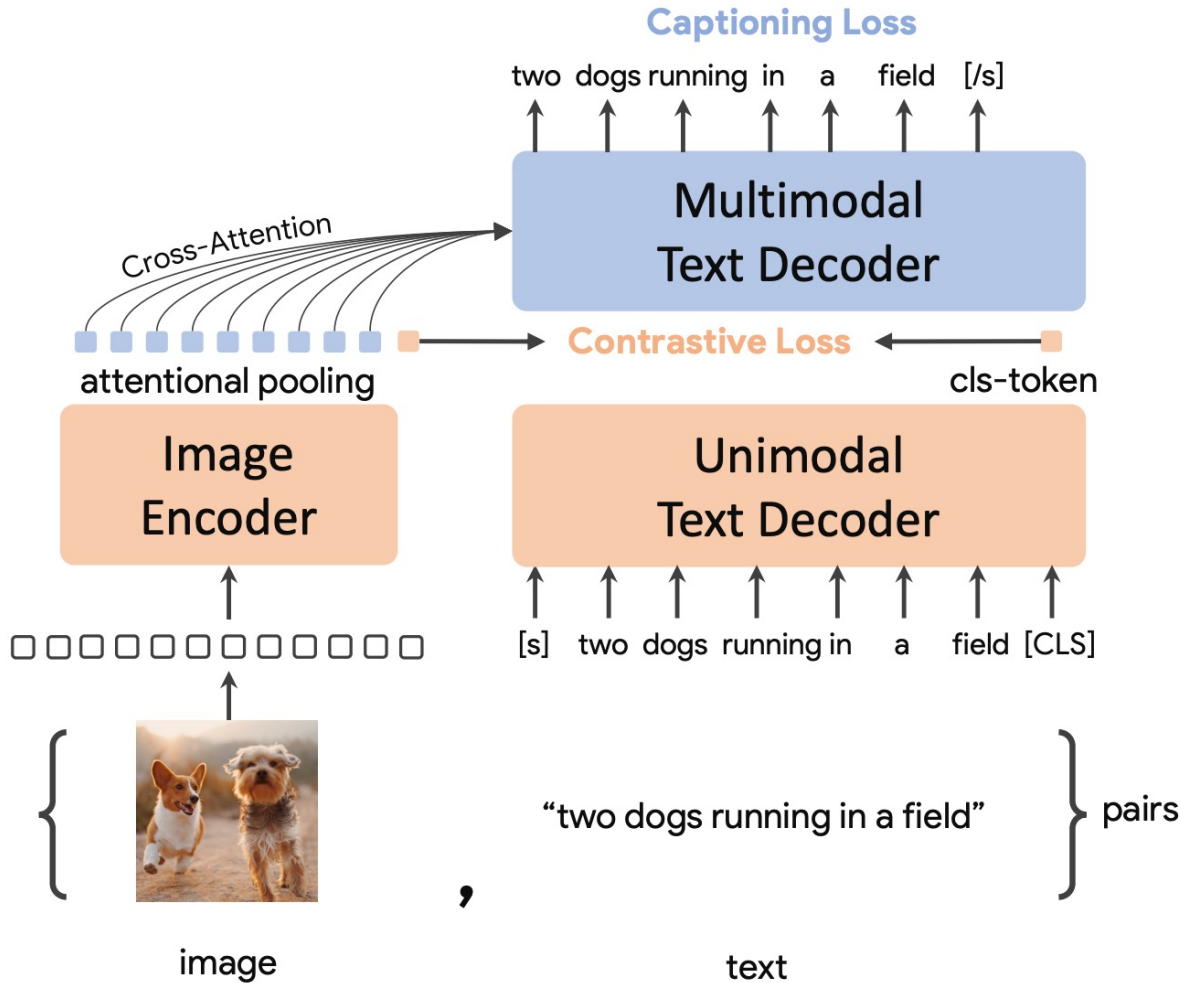


CoCa
Pretraining

Zero-shot, frozen-feature or finetuning

CoCa

Contrastive Captioner (CoCa),
Yu et al., 2022.



Single-Encoder Classification. The classic single-encoder approach pretrains a visual encoder through image classification on a large crowd-sourced image annotation dataset (*e.g.*, ImageNet [9], Instagram [20] or JFT [21]), where the vocabulary of annotation texts is usually fixed. These image annotations are usually mapped into discrete class vectors to learn with a cross-entropy loss as

$$\mathcal{L}_{\text{Cls}} = -p(y) \log q_{\theta}(x), \quad (1)$$

Dual-Encoder Contrastive Learning. Compared to pretraining with single-encoder classification, which requires human-annotated labels and data cleaning, the dual-encoder approach exploits noisy web-scale text descriptions and introduces a learnable text tower to encode free-form texts. The two encoders are jointly optimized by contrasting the paired text against others in the sampled batch:

$$\mathcal{L}_{\text{Con}} = -\frac{1}{N} \left(\underbrace{\sum_i \log \frac{\exp(x_i^{\top} y_i / \sigma)}{\sum_{j=1}^N \exp(x_i^{\top} y_j / \sigma)}}_{\text{image-to-text}} + \underbrace{\sum_i \log \frac{\exp(y_i^{\top} x_i / \sigma)}{\sum_{j=1}^N \exp(y_i^{\top} x_j / \sigma)}}_{\text{text-to-image}} \right), \quad (2)$$

where x_i and y_j are normalized embeddings of the image in the i -th pair and that of the text in the j -th pair. N is the batch size, and σ is the temperature to scale the logits. In addition to the image

Encoder-Decoder Captioning. While the dual-encoder approach encodes the text as a whole, the generative approach (*a.k.a.* captioner) aims for detailed granularity and requires the model to predict the exact tokenized texts of y autoregressively. Following a standard encoder-decoder architecture, the image encoder provides latent encoded features (*e.g.*, using a Vision Transformer [39] or ConvNets [40]) and the text decoder learns to maximize the conditional likelihood of the paired text y under the forward autoregressive factorization:

$$\mathcal{L}_{\text{Cap}} = -\sum_{t=1}^T \log P_{\theta}(y_t | y_{<t}, x). \quad (3)$$

CoCa

Contrastive Captioner (CoCa),
Yu et al., 2022.

Model	Flickr30K (1K test set)						MSCOCO (5K test set)					
	Image → Text			Text → Image			Image → Text			Text → Image		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
CLIP [12]	88.0	98.7	99.4	68.7	90.6	95.2	58.4	81.5	88.1	37.8	62.4	72.2
ALIGN [13]	88.6	98.7	99.7	75.7	93.8	96.8	58.6	83.0	89.7	45.6	69.8	78.6
FLAVA [35]	67.7	94.0	-	65.2	89.4	-	42.7	76.8	-	38.4	67.5	-
FILIP [61]	89.8	99.2	99.8	75.0	93.4	96.3	61.3	84.3	90.4	45.9	70.6	79.3
Florence [14]	90.9	99.1	-	76.7	93.6	-	64.7	85.9	-	47.2	71.4	-
CoCa-Base	89.8	98.8	99.8	76.8	93.7	96.8	63.8	84.7	90.7	47.5	72.4	80.9
CoCa-Large	91.4	99.2	99.9	79.0	95.1	97.4	65.4	85.6	91.4	50.1	73.8	81.8
CoCa	92.5	99.5	99.9	80.4	95.7	97.7	66.3	86.2	91.8	51.2	74.2	82.0

Table 3: Zero-shot image-text retrieval results on Flickr30K [62] and MSCOCO [63] datasets.

Model	ImageNet	ImageNet-A	ImageNet-R	ImageNet-V2	ImageNet-Sketch	ObjectNet	Average
CLIP [12]	76.2	77.2	88.9	70.1	60.2	72.3	74.3
ALIGN [13]	76.4	75.8	92.2	70.1	64.8	72.2	74.5
FILIP [61]	78.3	-	-	-	-	-	-
Florence [14]	83.7	-	-	-	-	-	-
LiT [32]	84.5	79.4	93.9	78.7	-	81.1	-
BASIC [33]	85.7	85.6	95.7	80.6	76.1	78.9	83.7
CoCa-Base	82.6	76.4	93.2	76.5	71.7	71.6	78.7
CoCa-Large	84.8	85.7	95.6	79.6	75.7	78.6	83.3
CoCa	86.3	90.2	96.5	80.7	77.6	82.7	85.7

Table 4: Zero-shot image classification results on ImageNet [9], ImageNet-A [64], ImageNet-R [65], ImageNet-V2 [66], ImageNet-Sketch [67] and ObjectNet [68].

Chameleon

Chameleon: Mixed-Modal Early-Fusion Foundation Models, Meta, 2024.

Prompt: I'm bored. Could you show me some cool, quirky-looking birds? Also tell me about them in 2-3 sentences.



Response: Absolutely! Here are three birds with some unique features that you might find interesting:

Keel-Billed Toucan The Keel-Billed Toucan is a brightly colored bird native to South American rainforests and woodlands. Its most distinctive feature is its colorful beak, which can grow up to seven inches long and has a distinctive, curved shape. The beak is used for plucking fruit from trees and for social displays. Despite its large size, the Keel-Billed Toucan is a skilled flyer and can often be seen flitting through the treetops. Here is an image of this vibrant bird: ****

Puffin Puffins are small, aquatic birds that are found in the northern hemisphere. They have distinctive black and white feathers, as well as a brightly colored beak that is used for attracting mates and defending their territory. Puffins are known for their distinctive "puffin" call, which sounds like a loud, guttural chuckle. They feed on fish, which they catch by diving into the water from the air. Puffins are colonial birds, and they breed in large colonies on rocky cliffs and islands. Here is an image of this adorable bird: ****



Golden Pheasant The Golden Pheasant is a brightly colored bird native to the forests of China. It is a member of the pheasant family and is known for its striking plumage, which features a golden-yellow body, red face, and green tail feathers. The Golden Pheasant is a ground-dwelling bird, and it feeds on a diet of insects, seeds, and fruits. It is a popular game bird, and is often hunted for its meat and feathers. In captivity, the Golden Pheasant is sometimes kept as an ornamental bird, due to its striking appearance. Here is an image of this beautiful bird: ****

I hope these images and descriptions have helped to brighten your day! If you have any other questions or requests, please don't hesitate to ask.

Chameleon

Chameleon: Mixed-Modal Early-Fusion Foundation Models, Meta, 2024.

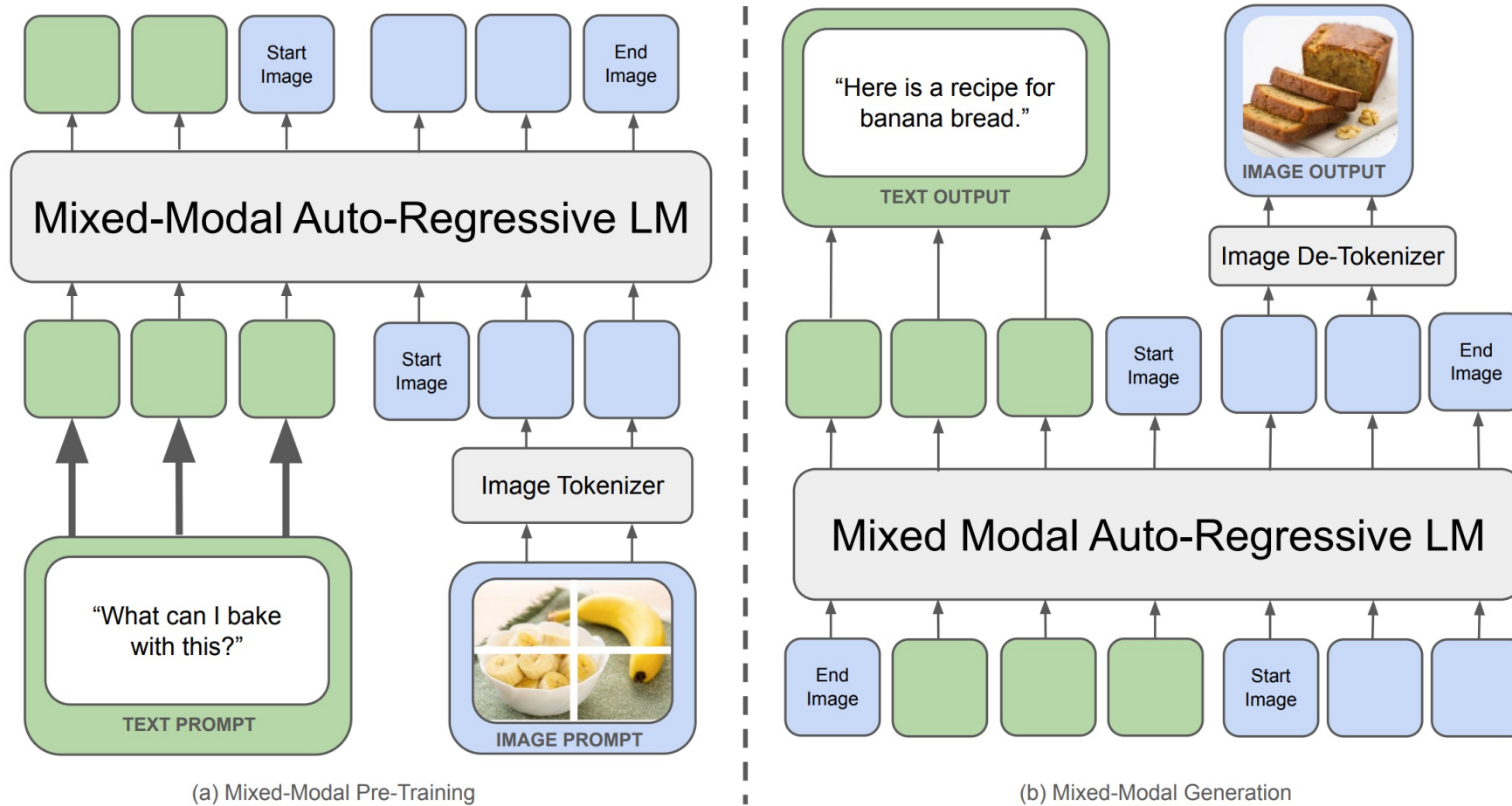


Figure 1 Chameleon represents all modalities — images, text, and code, as discrete tokens and uses a uniform transformer-based architecture that is trained from scratch in an end-to-end fashion on $\sim 10T$ tokens of interleaved mixed-modal data. As a result, Chameleon can both reason over, as well as generate, arbitrary mixed-modal documents. Text tokens are represented in green and image tokens are represented in blue.

Chameleon

2 Pre-Training

Chameleon represents images, in addition to text, as a series of discrete tokens and takes advantage of the scaling properties of auto-regressive Transformers (Ramesh et al., 2021; Aghajanyan et al., 2022, 2023; Yu et al., 2023). We present any ordering of images and text during training ranging from text-only, to single text/image pairs to full interleaved text-image documents.



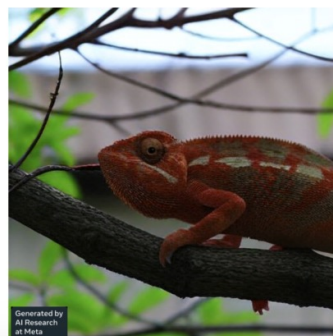
Prompt: `` While chameleons are masters of camouflage in the natural world, let's see how they might fare in a man-made setting.

1. Analyze the colors, patterns, and textures in this image and identify the most challenging spot for a chameleon to blend in.
2. Then generate an image of a chameleon with the optimal pattern to camouflage itself.
3. Then, rate and explain how successful each attempt is on a scale from 1-10.

Response: The image is of a stop sign in a forested area, with some sunlight filtering through the trees. The sign has a red octagonal shape with a white border and the word "STOP" written in white letters. In the background, there are green leaves and trees.

Challenging Spot for a Chameleon: The stop sign would be the most challenging spot for a chameleon to blend in due to its bright red color and bold white letters, which would stand out in the forested environment.

Optimal Pattern for Camouflage: To camouflage itself on the stop sign, a chameleon would need to have a pattern that would break up its outline and make it less conspicuous. A pattern with shades



of brown and red would help the chameleon blend in more effectively. Additionally, a mottled or dappled pattern would help to scatter light and further reduce the chameleon's visibility.

``

Rating and Explanation: A rating of 2/10 would be given for the chameleon's ability to blend in on the stop sign. While the chameleon's green and brown pattern helps to break up its outline, the bright red color of the stop sign will still make it stand out in the forested environment.

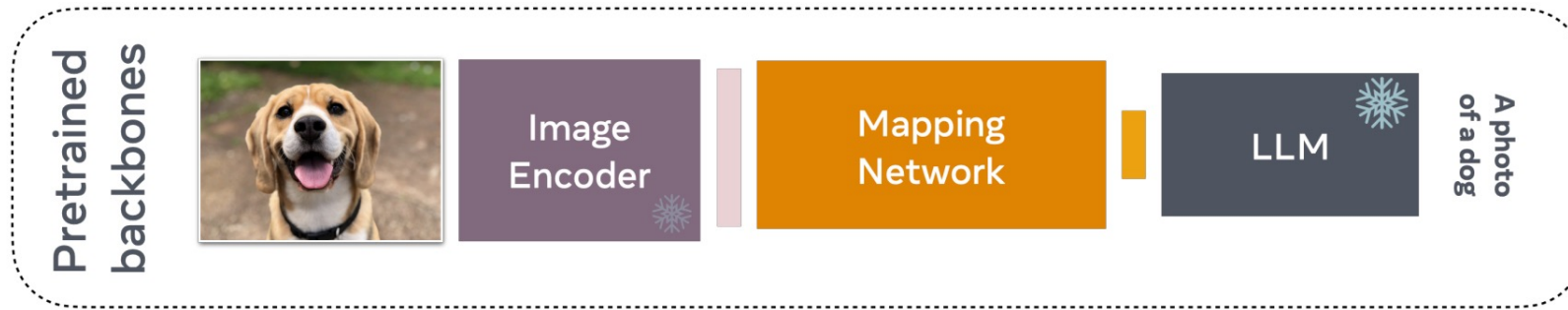


Fig: Bordes et al., "An Introduction to Vision-Language Modeling", 2024.

Approaches Using Pre-trained Backbones

Frozen

Multimodal Few-Shot Learning with Frozen Language Models, 2021



Figure 1: Curated samples with about five seeds required to get past well-known language model failure modes of either repeating text for the prompt or emitting text that does not pertain to the image. These samples demonstrate the ability to generate open-ended outputs that adapt to both images and text, and to make use of facts that it has learned during language-only pre-training.

Frozen

Multimodal Few-Shot Learning with Frozen Language Models, 2021

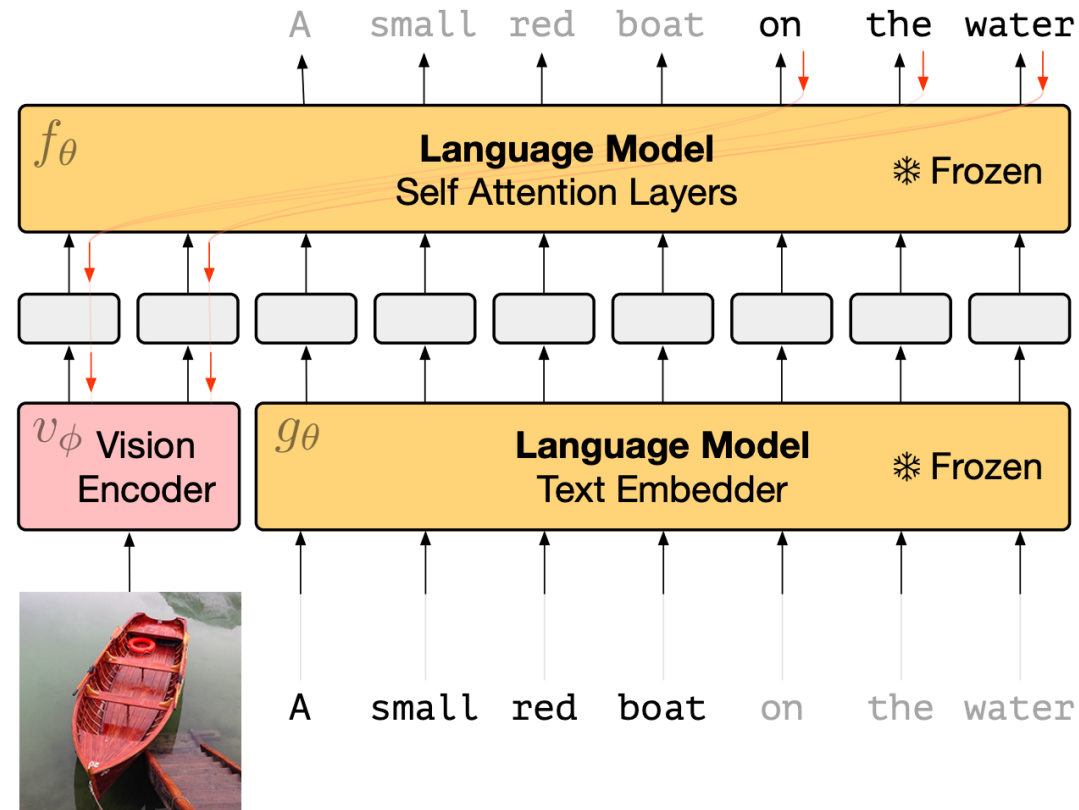


Figure 2: Gradients through a frozen language model's self attention layers are used to train the vision encoder.

Frozen

Multimodal Few-Shot Learning with Frozen Language Models, 2021

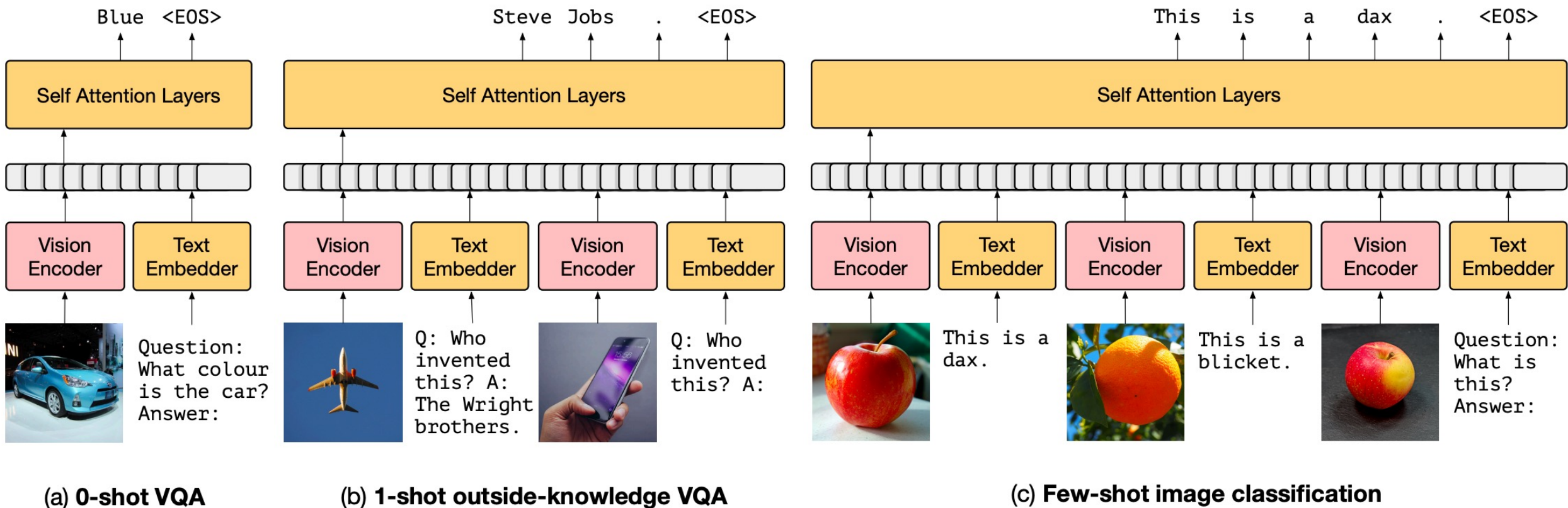


Figure 3: Inference-Time interface for *Frozen*. The figure demonstrates how we can support (a) visual question answering, (b) outside-knowledge question answering and (c) few-shot image classification via in-context learning.

Frozen

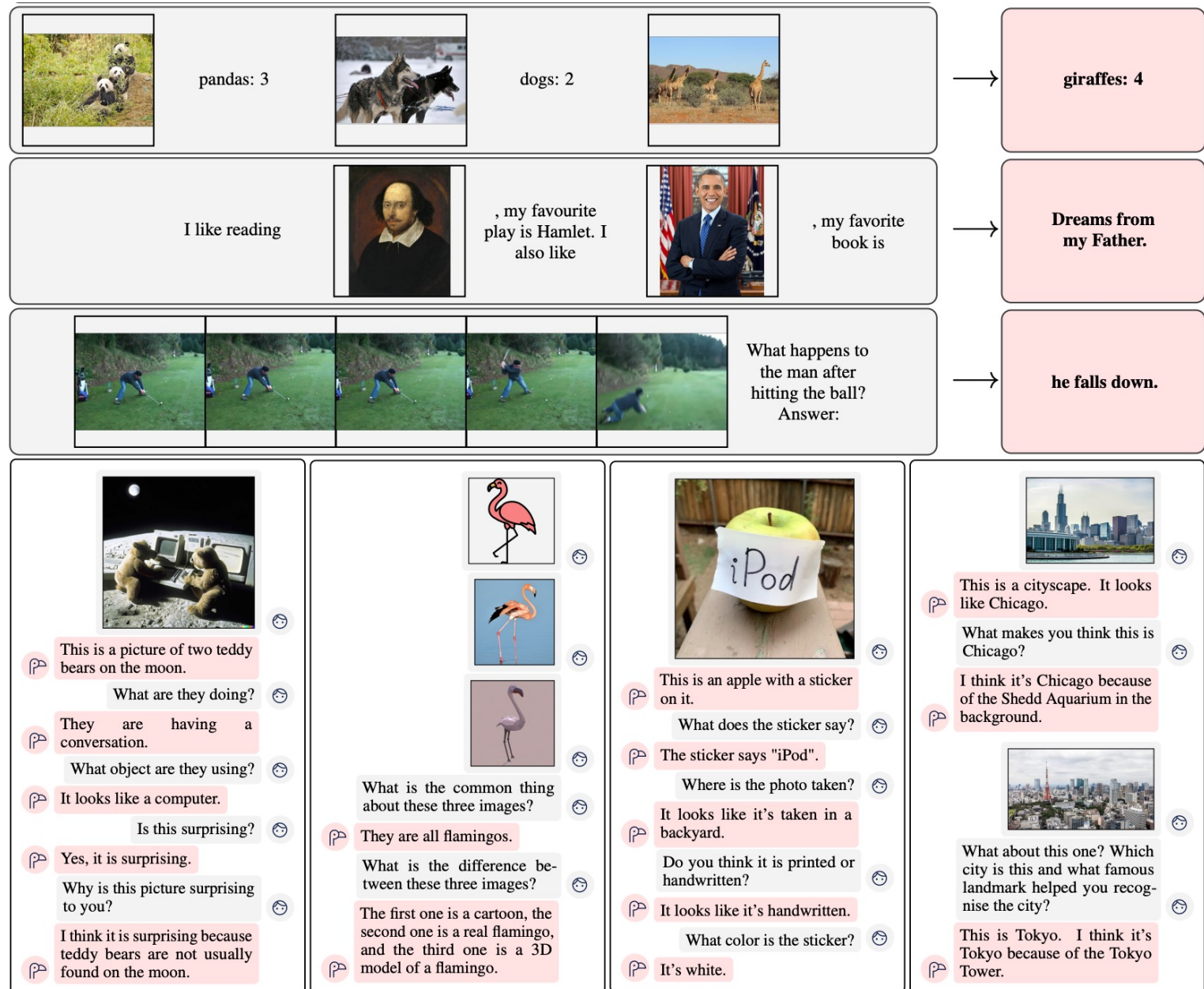
Multimodal Few-Shot Learning with
Frozen Language Models, 2021

Vision Encoder Our vision encoder is based on NF-ResNet-50 [3]. We define v_ϕ as a function that takes a raw image and emits a continuous sequence to be consumed by the transformer. We use the final output vector of the NF-Resnet *after* the global pooling layer.

Visual Prefix One important requirement is to represent images in a form that the transformer *already* understands: a sequence of continuous embeddings, each having the same dimensionality D as a token embedding t_l . We therefore form the visual prefix by linearly mapping the vision encoder's output to $D * n$ channels, and then reshaping the result as a sequence of n embeddings, each with dimensionality D . We call this sequence a *visual prefix* since it plays the same functional role in the transformer architecture as (part of) an embedding sequence of prefix tokens. We experimented using different number of tokens, specifically 1, 2 and 4 and found that 2 performs best, though certainly this would be sensitive to other architectural details. See Appendix for more details on the architecture.

Flamingo

Flamingo: a Visual Language Model for Few-Shot Learning, Deepmind, 2022



Flamingo

Flamingo: a Visual Language Model for Few-Shot Learning, Deepmind, 2022

Vision Encoder: Normalizer-Free ResNet (NFNet)

Perceiver Sampler: Fixed # of queries attend to variable length of visual tokens.

LLM: Chinchilla

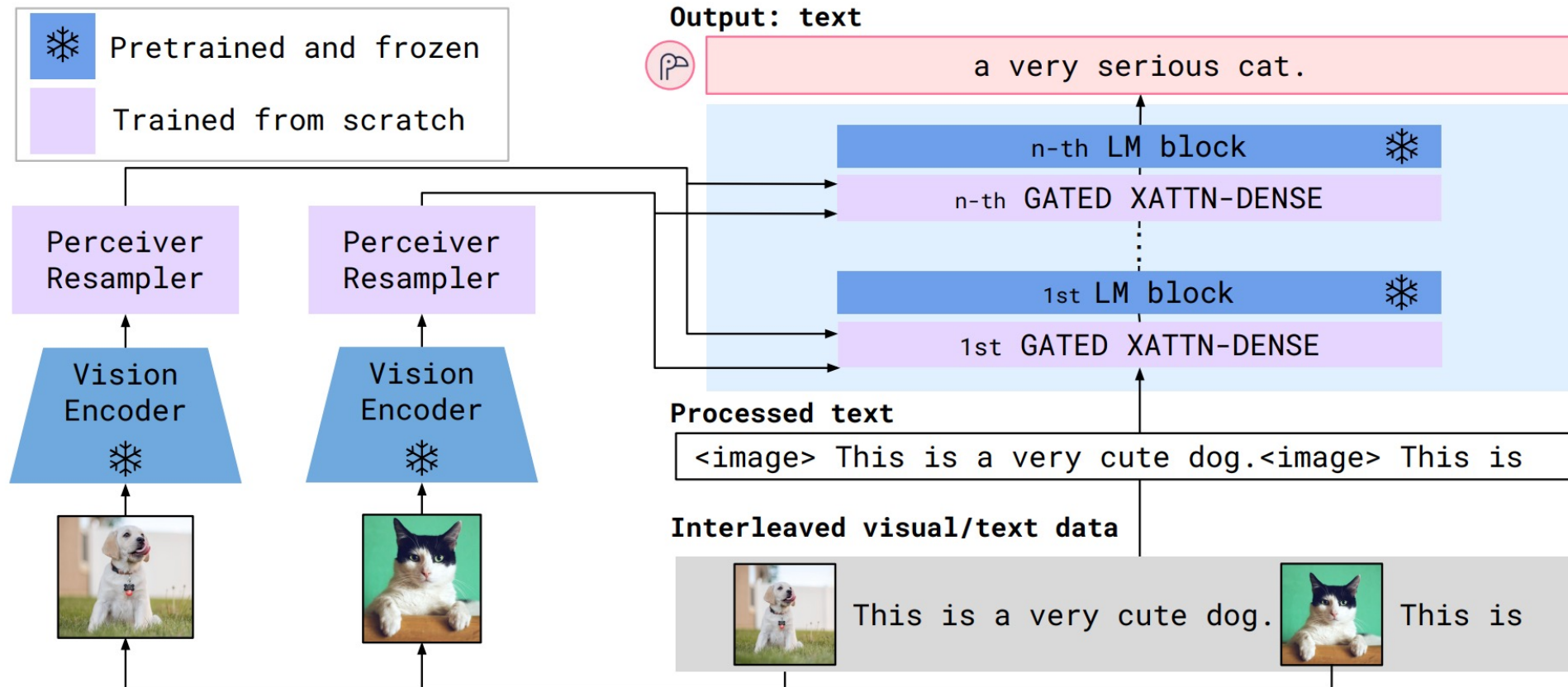


Figure 3: **Flamingo architecture overview.** Flamingo is a family of visual language models (VLMs) that take as input visual data interleaved with text and produce free-form text as output.

Flamingo

Flamingo: a Visual Language Model
for Few-Shot Learning, Deepmind, 2022

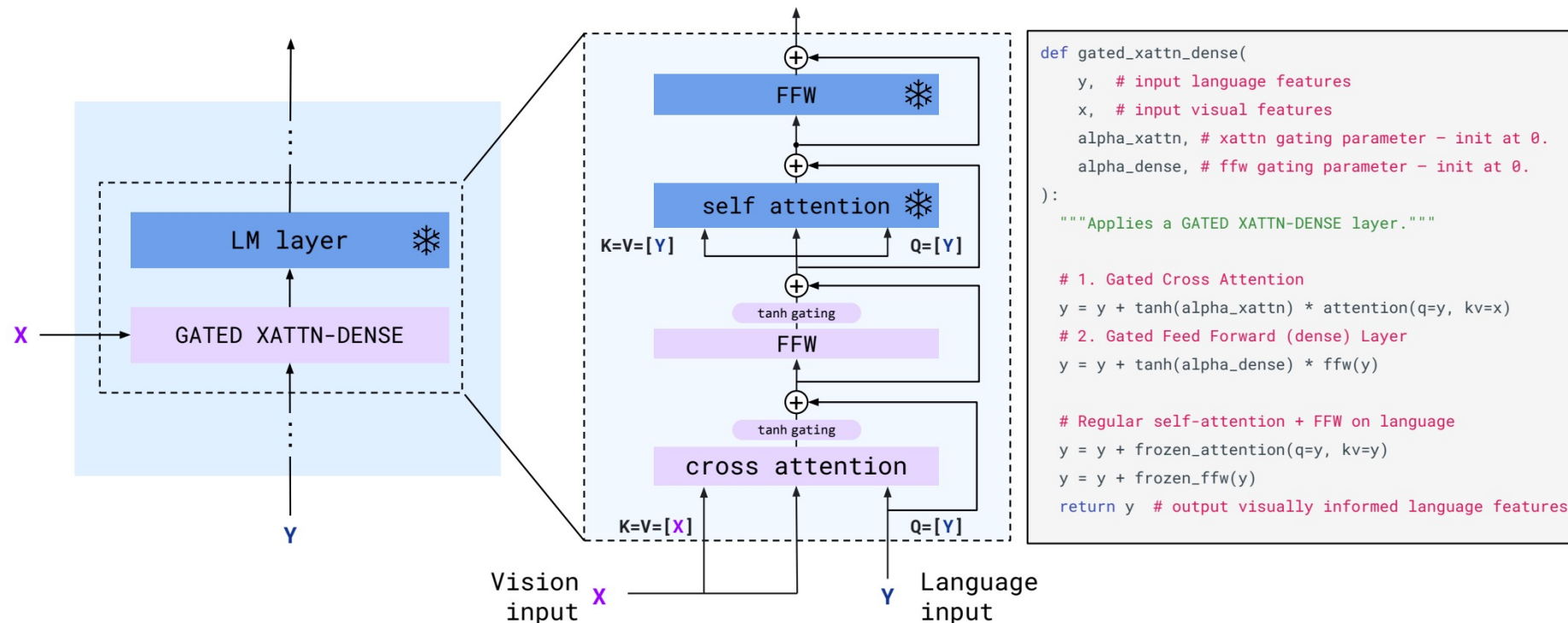


Figure 4: **GATED XATTN-DENSE layers.** To condition the LM on visual inputs, we insert new cross-attention layers between existing pretrained and frozen LM layers. The keys and values in these layers are obtained from the vision features while the queries are derived from the language inputs. They are followed by dense feed-forward layers. These layers are *gated* so that the LM is kept intact at initialization for improved stability and performance.

Flamingo

Flamingo: a Visual Language Model for Few-Shot Learning, Deepmind, 2022

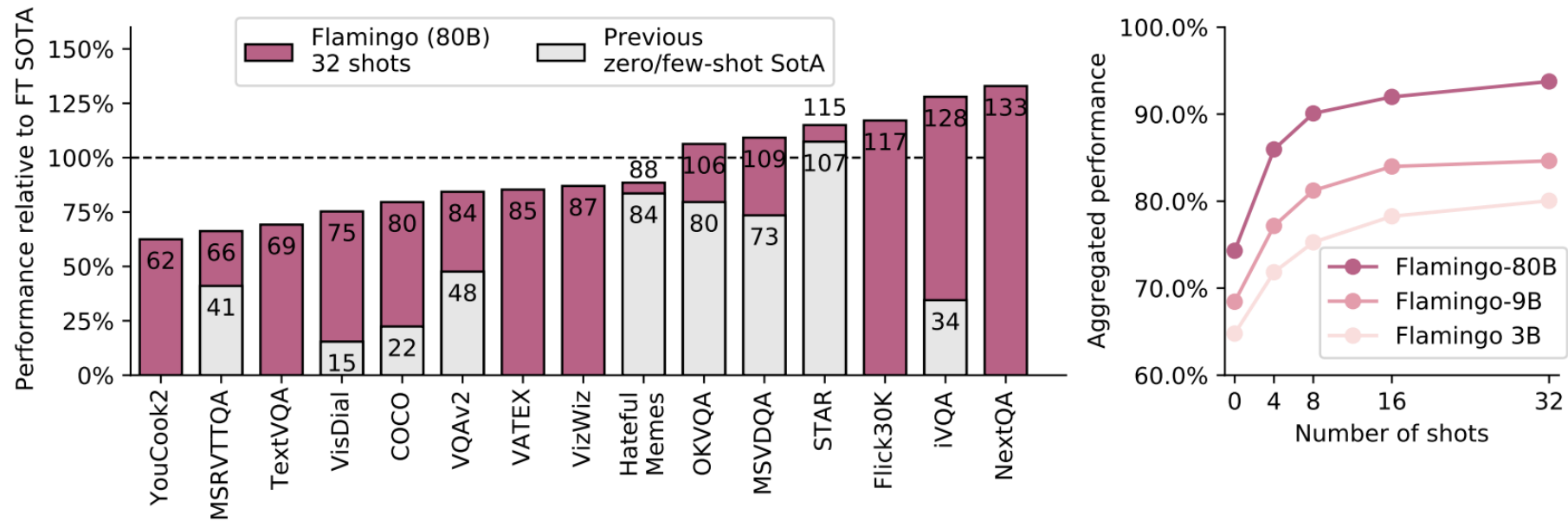


Figure 2: **Flamingo results overview.** *Left:* Our largest model, dubbed *Flamingo*, outperforms state-of-the-art fine-tuned models on 6 of the 16 tasks we consider with no fine-tuning. For the 9 tasks with published few-shot results, *Flamingo* sets the new few-shot state of the art. *Note:* We omit RareAct, our 16th benchmark, as it is a zero-shot benchmark with no available fine-tuned results to compare to. *Right:* Flamingo performance improves with model size and number of shots.

BLIP

BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation, Salesforce, 2022.

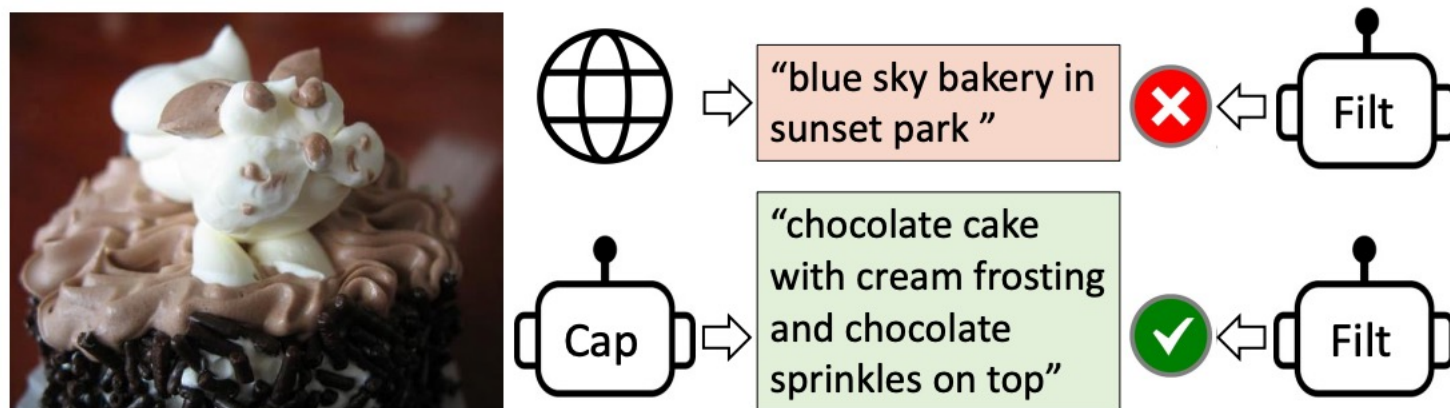


Figure 1. We use a Captioner (Cap) to generate synthetic captions for web images, and a Filter (Filt) to remove noisy captions.

BLIP

BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation, Salesforce, 2022.

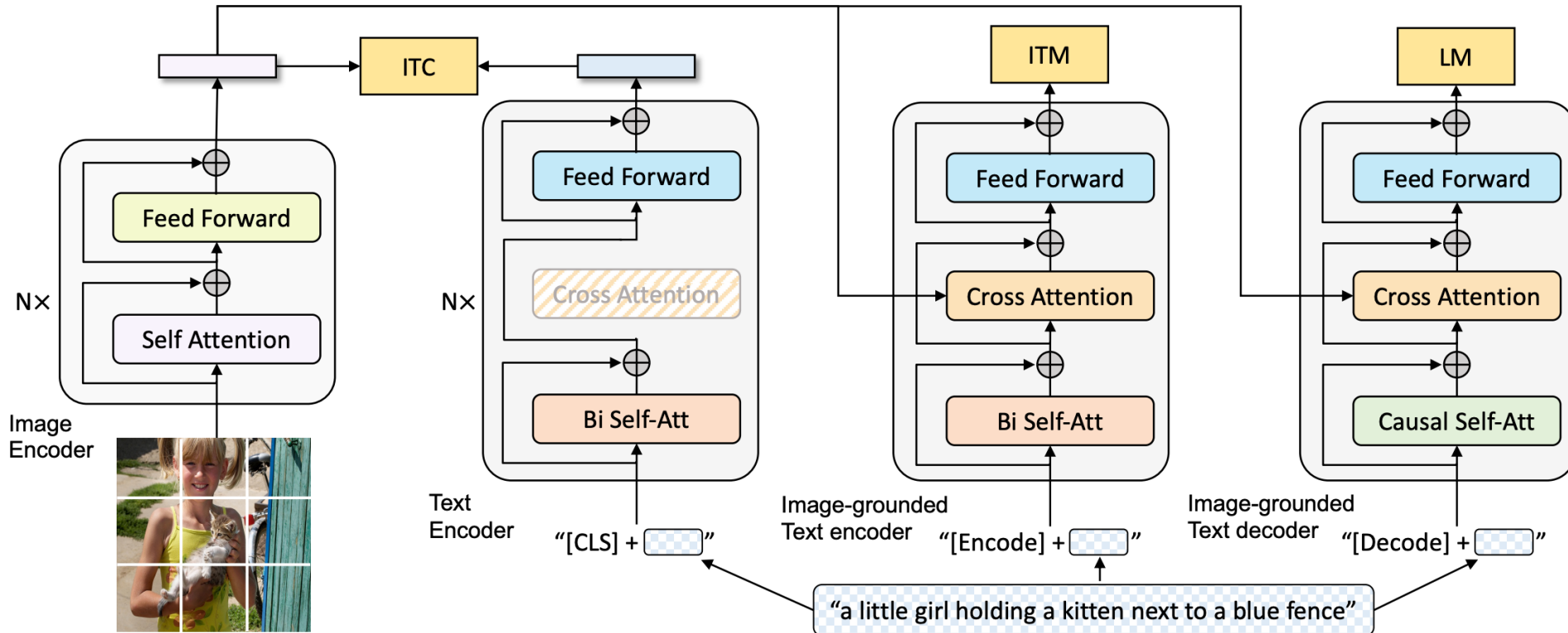


Figure 2. Pre-training model architecture and objectives of BLIP (same parameters have the same color). We propose multimodal mixture of encoder-decoder, a unified vision-language model which can operate in one of the three functionalities: (1) Unimodal encoder is trained with an image-text contrastive (ITC) loss to align the vision and language representations. (2) Image-grounded text encoder uses additional cross-attention layers to model vision-language interactions, and is trained with a image-text matching (ITM) loss to distinguish between positive and negative image-text pairs. (3) Image-grounded text decoder replaces the bi-directional self-attention layers with causal self-attention layers, and shares the same cross-attention layers and feed forward networks as the encoder. The decoder is trained with a language modeling (LM) loss to generate captions given images.

BLIP

BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation, Salesforce, 2022.

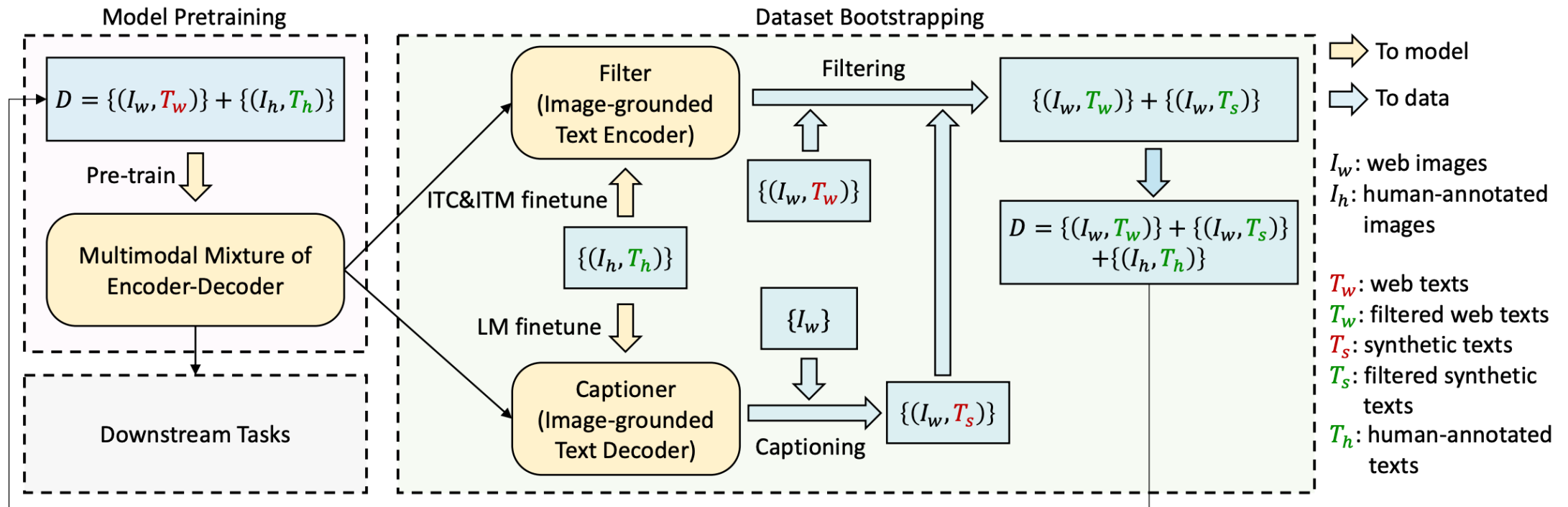


Figure 3. Learning framework of BLIP. We introduce a captioner to produce synthetic captions for web images, and a filter to remove noisy image-text pairs. The captioner and filter are initialized from the same pre-trained model and finetuned individually on a small-scale human-annotated dataset. The bootstrapped dataset is used to pre-train a new model.

BLIP-2

BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models, Salesforce, 2023.

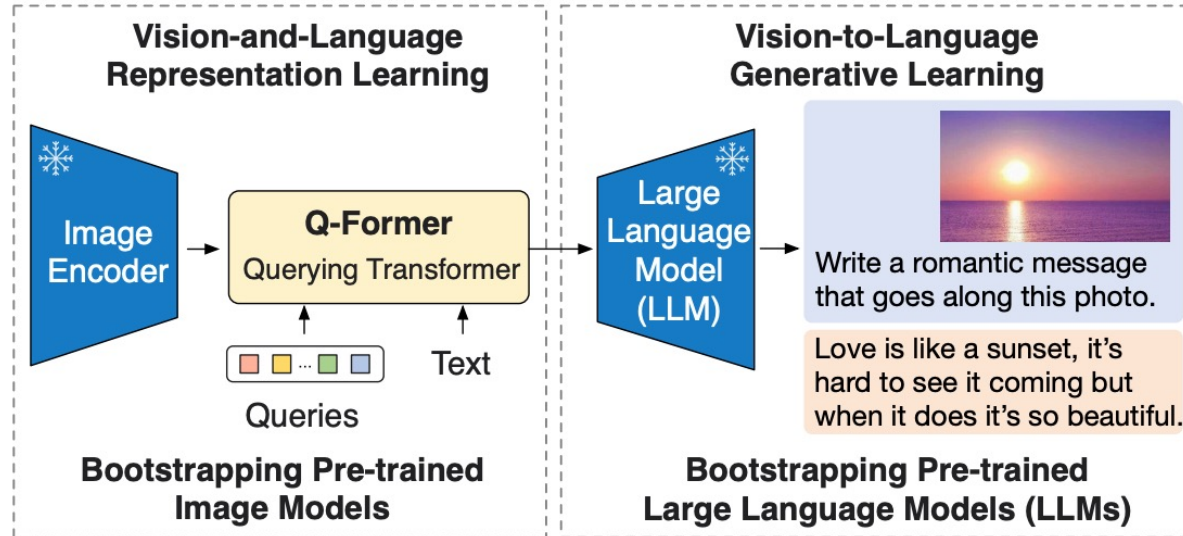


Figure 1. Overview of BLIP-2’s framework. We pre-train a lightweight Querying Transformer following a two-stage strategy to bridge the modality gap. The first stage bootstraps vision-language representation learning from a frozen image encoder. The second stage bootstraps vision-to-language generative learning from a frozen LLM, which enables zero-shot instructed image-to-text generation (see Figure 4 for more examples).

BLIP-2

BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models, Salesforce, 2023.

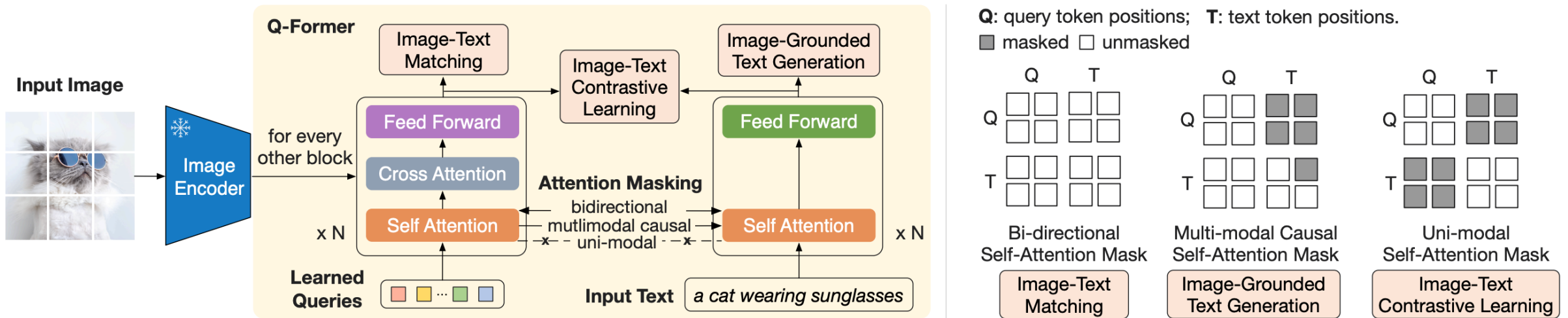


Figure 2. (Left) Model architecture of Q-Former and BLIP-2’s first-stage vision-language representation learning objectives. We jointly optimize three objectives which enforce the queries (a set of learnable embeddings) to extract visual representation most relevant to the text. **(Right)** The self-attention masking strategy for each objective to control query-text interaction.

BLIP-2

BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models, Salesforce, 2023.

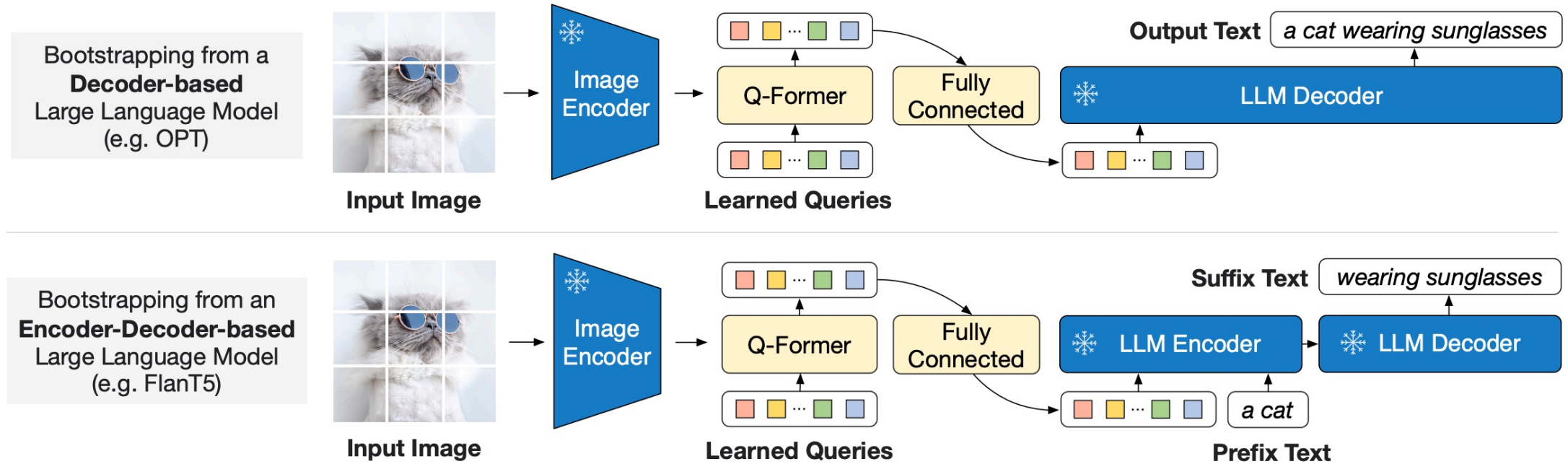


Figure 3. BLIP-2’s second-stage vision-to-language generative pre-training, which bootstraps from frozen large language models (LLMs). **(Top)** Bootstrapping a decoder-based LLM (e.g. OPT). **(Bottom)** Bootstrapping an encoder-decoder-based LLM (e.g. FlanT5). The fully-connected layer adapts from the output dimension of the Q-Former to the input dimension of the chosen LLM.

Misc

Segment Anything Model (SAM) 2023

Segment Anything

Alexander Kirillov^{1,2,4} Eric Mintun² Nikhila Ravi^{1,2} Hanzi Mao² Chloe Rolland³ Laura Gustafson³
Tete Xiao³ Spencer Whitehead Alexander C. Berg Wan-Yen Lo Piotr Dollár⁴ Ross Girshick⁴
¹project lead ²joint first author ³equal contribution ⁴directional lead

Meta AI Research, FAIR

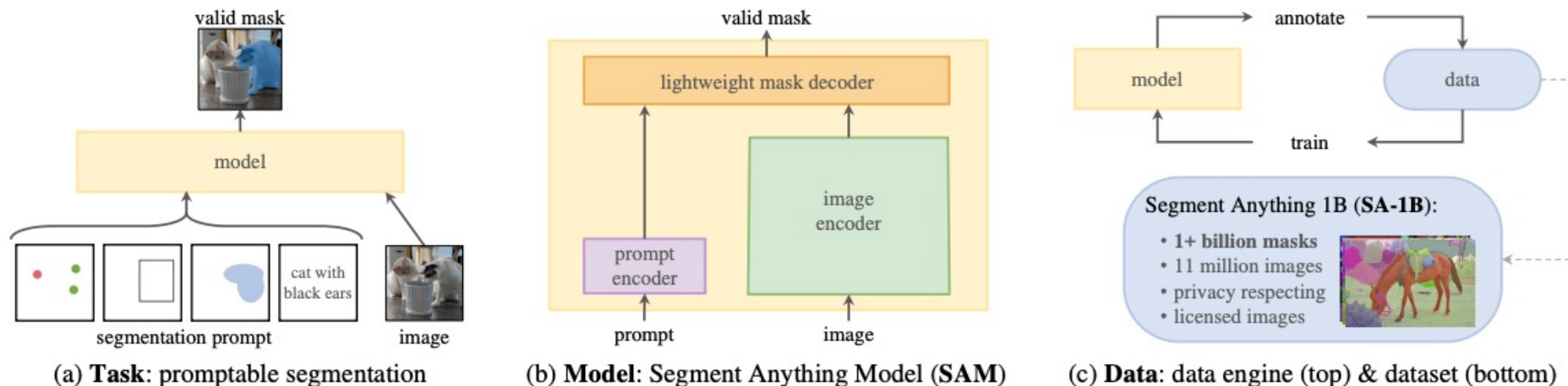


Figure 1: We aim to build a foundation model for segmentation by introducing three interconnected components: a promptable segmentation *task*, a segmentation *model* (SAM) that powers data annotation and enables zero-shot transfer to a range of tasks via prompt engineering, and a *data* engine for collecting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

Segment Anything Model (SAM) v2 2024

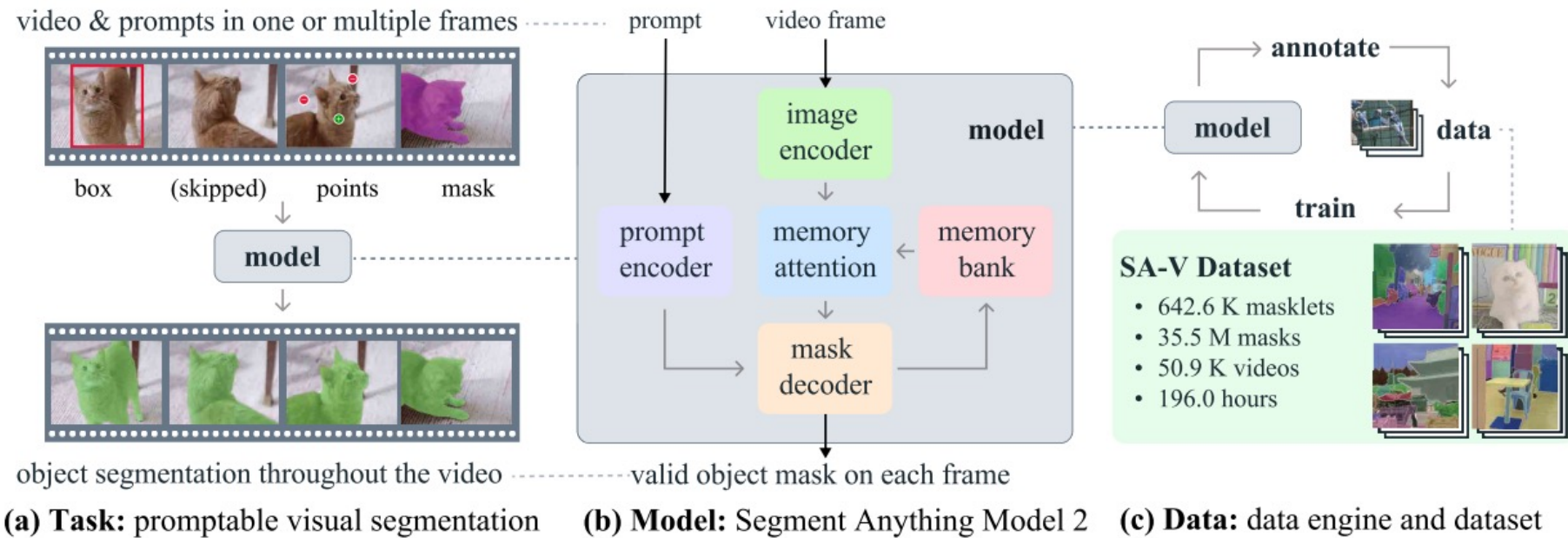


Figure 1 We introduce the Segment Anything Model 2 (SAM 2), towards solving the promptable visual segmentation task (a) with our foundation model (b), trained on our large-scale SA-V dataset collected through our data engine (c). SAM 2 is capable of interactively segmenting regions through prompts (clicks, boxes, or masks) on one or multiple video frames by utilizing a streaming memory that stores previous prompts and predictions.

Segment Everything Everywhere All At Once 2023

Segment Everything Everywhere All at Once

Xueyan Zou^{*§2}, Jianwei Yang^{*†1}, Hao Zhang^{*‡}, Feng Li^{*‡}, Linjie Li[†], Jianfeng Wang[†]
Lijuan Wang[†], Jianfeng Gao^{¶‡}, Yong Jae Lee^{¶§}

§ University of Wisconsin-Madison † Microsoft Research, Redmond ‡ HKUST † Microsoft Cloud & AI

*Equal Contribution ¶ Equal Advisory Contribution 1. Project Lead 2. Main Technical Contribution

{xueyan,yongjaelee}@cs.wisc.edu {jianwyan,jfgao,linjli}@microsoft.com {hzhangcx,fliaiy}@connect.ust.hk

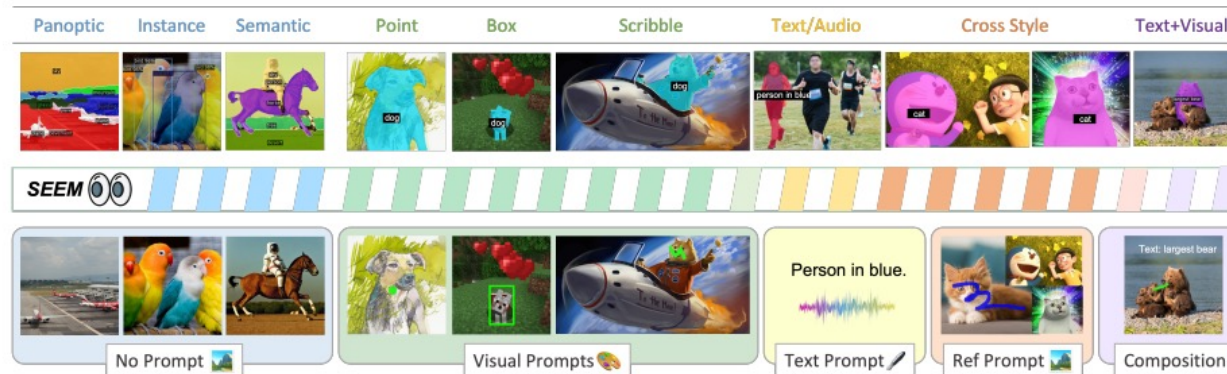


Figure 1: *SEEM* supports generic segmentation tasks—including semantic, instance, and panoptic segmentation—in an open-set fashion when no prompt is provided. *SEEM* also enables the use of visual, textual, and referring region prompts in flexible combinations, making it a promptable and interactive segmentation interface.

INTERNVIDEO2: SCALING FOUNDATION MODELS FOR MULTIMODAL VIDEO UNDERSTANDING

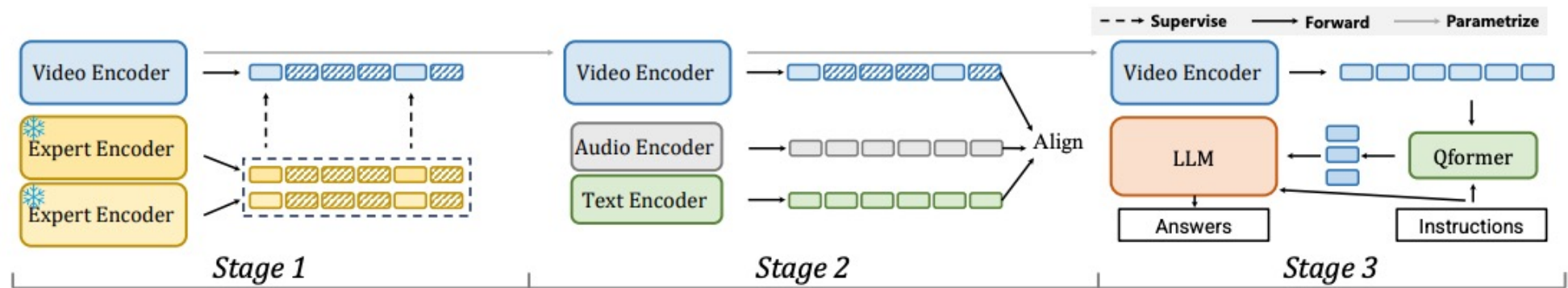


Figure 2: Framework of **InternVideo2**. It consists of three consecutive training phases: unmasked video token reconstruction, multimodal contrastive learning, and next token prediction. In stage 1, the video encoder is trained from scratch, while in stages 2 and 3, it is initialized from the version used in the previous stage.

Stage 1: Reconstructing Unmasked Video Tokens

Stage 2: Aligning Video to Audio-Speech-Text

Stage 3: Predicting Next Token with Video-Centric Inputs

INTERNVIDEO2: SCALING FOUNDATION MODELS FOR MULTIMODAL VIDEO UNDERSTANDING

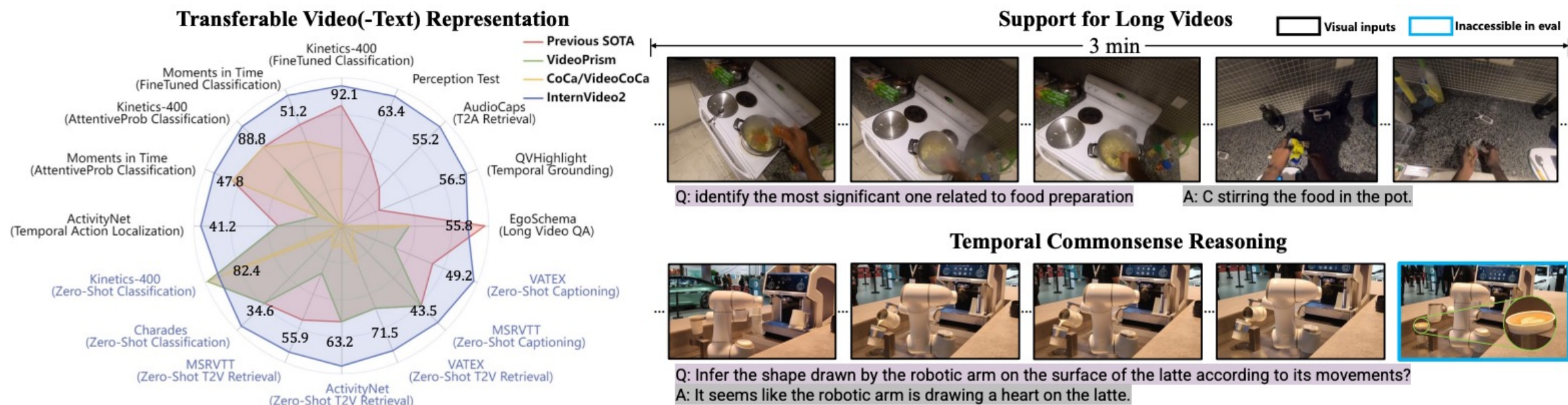


Figure 1: **InternVideo2** yields strong transferable visual and visual-linguistic representations across a total of 70 video understanding tasks, ranging from action recognition, video-text understanding, to video-centric dialogue. It also exhibits capability of long-form video understanding and procedure-aware reasoning.