ICLR
International Conference On
Learning Representations

ICLR 2024 Potpourri

Mert Kilickaya

Website

ICLR 2024 received **7500** submissions on 28th September

Pre-filtering: **~200** papers Post-filtering: **~20** papers

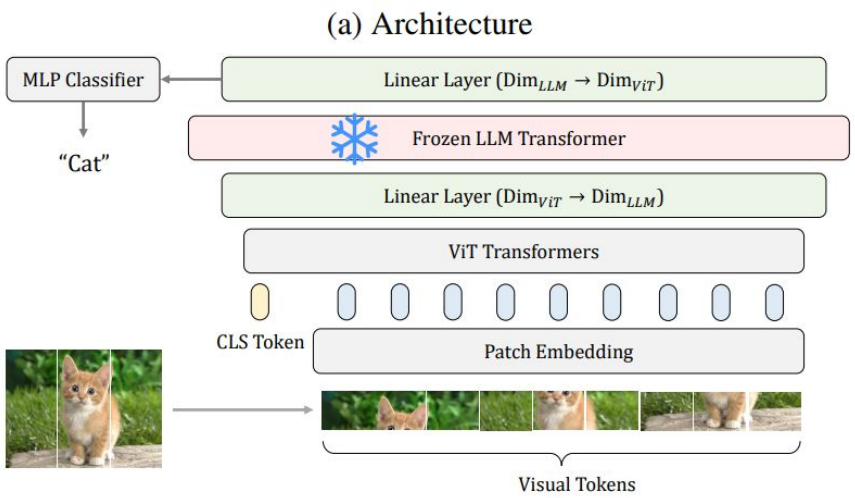
Not depth-first, but breadth-first discussion

Three Topics: LLM & Efficiency & Foundation Models

a) LLM

1

FROZEN TRANSFORMERS IN LANGUAGE MODELS ARE EFFECTIVE VISUAL ENCODER LAYERS



(b) Pseudo-Code

```
def __init__(self, *args, **kawargs):
    # Encoder
    self.ViT = Encoder(args, kwargs)
    self.classifier = Decoder(args, kwargs)

    # Language Transformer
    self.L1 = nn.Linear(ViT.hidden_dim, LLM.hidden_dim)
    self.LM = LM_Transformer(args, kwargs)
    self.L2 = nn.Linear(LLM.hidden_dim, ViT.hidden_dim)

    # Freezing
    for param in self.LM.parameters():
        param.requires_grad = False

def forward(self, img):
    z = self.ViT(x)

    z = self.L1(z)
    z = self.LM(z)
    z = self.L2(z)

    y = self.classifier(z)
    return y
```

Figure 1: Our straightforward method of using a *frozen* transformer block from *pre-trained* LLMs as a visual encoder layer. Visualized with an example of ViT (Dosovitskiy et al., 2021). **(a)** Our design simply appends a frozen transformer block (pink) on top of the regular visual encoder (gray). Only two trainable linear layers (green) are added to align the feature dimensions. **(b)** Pytorch-style pseudo-code shows the simplicity of our approach.

Encode image with ViT + Linear layer → Process the features with (frozen) LLM → Classify & Detect

1

FROZEN TRANSFORMERS IN LANGUAGE MODELS ARE EFFECTIVE VISUAL ENCODER LAYERS

Model	ImageNet	ImageNet-C	ImageNet-A	ImageNet-SK	ImageNet-R
ViT-T	72.1	43.9	7.7	19.6	32.3
ViT-T-LLaMA	**73.2**	**45.8**	**8.7**	**20.6**	**33.8**
ViT-S	80.1	57.2	20.5	28.9	42.1
ViT-S-LLaMA	**80.7**	**58.7**	**22.7**	**30.5**	**42.8**
ViT-B*	78.9	58.1	21.6	29.3	40.5
ViT-B-LLaMA	**80.6**	**60.6**	**24.6**	**30.4**	**40.9**

Inclusion of LLM robustifies the model -**But-** the improvement is not significant (~1-2%)

2 MiniGPT-V2

Method

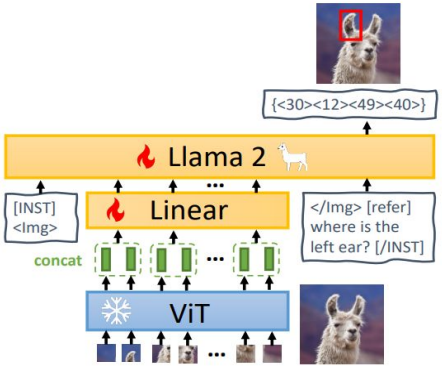


Figure 2: **Architecture of MiniGPT-v2.** The model takes a ViT visual backbone, which remains frozen during all training phases. We concatenate four adjacent visual output tokens from ViT backbone and project them into LLaMA-2 language model space via a linear projection layer.

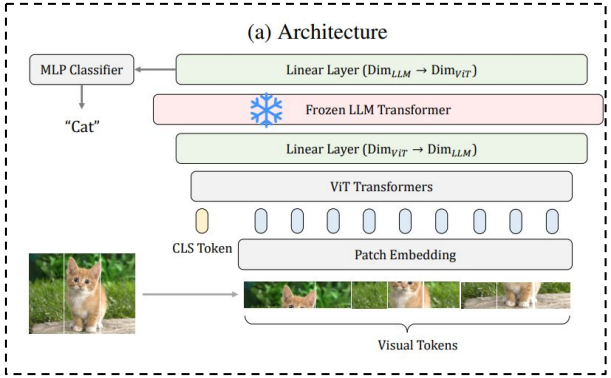
Results on Visual Question Answering

Method	Grounding	OKVQA	GQA	VSR (zero-shot)	TextVQA (zero-shot)	IconVQA (zero-shot)	VizWiz (zero-shot)	HM (zero-shot)
Flamingo-9B	✗	44.7	-	31.8	-	-	28.8	57.0
BLIP-2 (13B)	✗	45.9	41.0	50.9	42.5	40.6	19.6	53.7
InstructBLIP (13B)	✗	-	49.5	52.1	50.7	44.8	33.4	57.5
MiniGPT-4 (13B)	✗	37.5	30.8	41.6	19.4	37.6	-	-
LLaVA (13B)	✗	54.4	41.3	51.2	38.9	43.0	-	-
Shikra (13B)	✓	47.2	-	-	-	-	-	-
Ours (7B)	✓	**56.9**	**60.3**	60.6	51.9	47.7	30.3	58.2
Ours (7B)-chat	✓	55.9	58.8	**63.3**	**52.3**	**49.4**	**42.4**	**59.5**

Table 3: **Results on multiple VQA tasks.** We report top-1 accuracy for each task. Grounding column indicates whether the model incorporates visual localization capability. The best performance for each benchmark is indicated in **bold**.

The model leads to a significant boost across many vision-and-language tasks.

1 [Frozen-LLM]



2 [MiniGPT-V2]

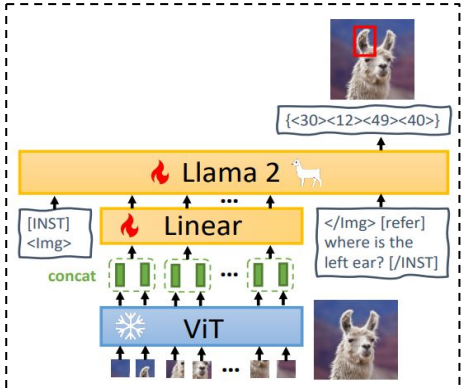
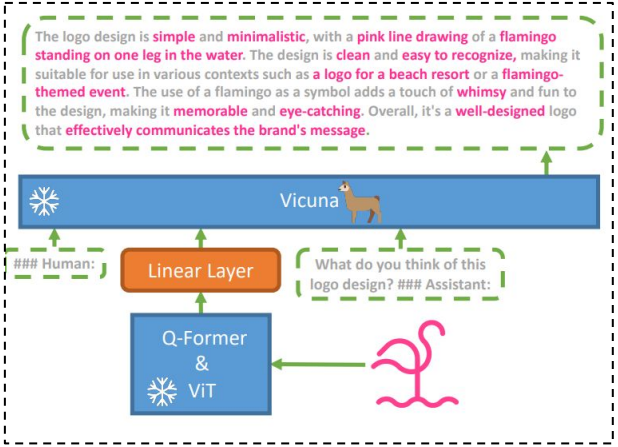


Figure 2: **Architecture of MiniGPT-v2.** The model takes a ViT visual backbone, which remains frozen during all training phases. We concatenate four adjacent visual output tokens from ViT backbone and project them into LLaMA-2 language model space via a linear projection layer.

3 [MiniGPT-V4]



4 [PLANTING A SEED OF VISION IN LARGE LANGUAGE MODEL]

5 [LINGUISTIC IMAGE UNDERSTANDING]

b-1) Efficient ViT

1 FASTERVIT: FAST VISION TRANSFORMERS WITH HIERARCHICAL ATTENTION

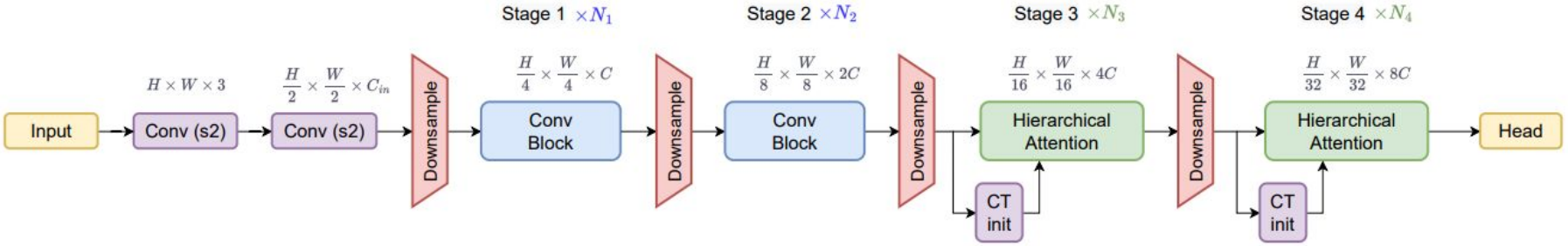


Figure 3: Overview of the FasterViT architecture. We use a multi-scale architecture with CNN and transformer-based blocks in stages 1, 2 and 3, 4, respectively. Best viewed in color.

CT Init: Carrier token initialization

FasterViT combines Convolutional blocks (stage 1-2) with Transformer blocks (stage 3-4).

1 FASTERVIT: FAST VISION TRANSFORMERS WITH HIERARCHICAL ATTENTION

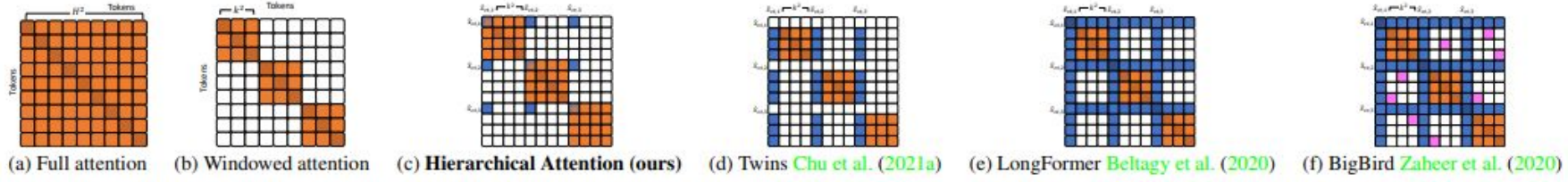


Figure 5: Attention map comparison for a feature map of size $H \times H \times d$. ▢ - no attention, ■ - normal token attention, ■ - carrier token attention, ■ - random token attention. Full attention (a) has complexity of $O(H^4 d)$, windowed attention significantly reduces it to $O(k^2 H^2 d)$ but lacks global context.

Hierarchical attention extends windowed attention via carrier tokens to pool from distinct windows.

1 FASTERVIT: FAST VISION TRANSFORMERS WITH HIERARCHICAL ATTENTION

Classification

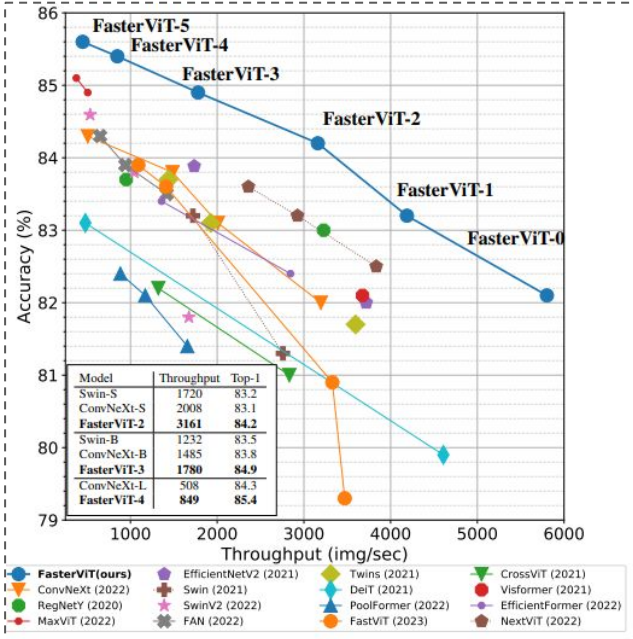


Figure 1: Comparison of image throughput and ImageNet-1K Top-1 accuracy. For all models, throughput is measured on A100 GPU with batch size of 128.

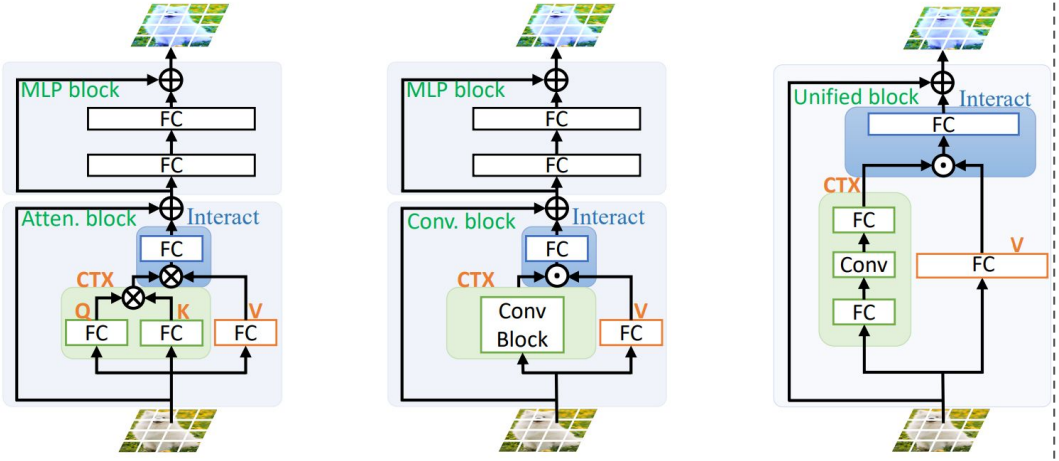
Segmentation

Model	Throughput	FLOPs (G)	IoU(ss/ms)
Swin-T Liu et al. (2021)	350	945	44.5/45.8
ConvNeXt-T Liu et al. (2022b)	363	939	- /46.7
FasterViT-2	**377**	974	**47.2/48.4**
Twins-SVT-B Chu et al. (2021a)	204	-	47.7/48.9
Swin-S Liu et al. (2021)	219	1038	47.6/49.5
ConvNeXt-S Liu et al. (2022b)	234	1027	- /49.6
FasterViT-3	**254**	1076	**48.7/49.7**
Twins-SVT-L Chu et al. (2021a)	164	-	48.8/50.2
Swin-B Liu et al. (2021)	172	1188	48.1/49.7
ConvNeXt-B Liu et al. (2022b)	189	1170	- /49.9
FasterViT-4	**202**	1290	**49.1/50.3**

Table 4: Semantic segmentation on **ADE20K** Zhou et al. (2017) with UPerNet Xiao et al. (2018).

FasterViT yields the highest accuracy-efficiency tradeoff for classification/segmentation tasks (ADE20K).

2 EFFICIENT MODULATION FOR VISION NETWORKS



(a) Transformer block (b) Modulation design (c) Our EfficientMod block

Model	Top-1(%)	Latency (ms) GPU	Latency (ms) CPU	Params (M)	FLOPs (G)	Size.
MobileNetV2×1.0 (2018)	71.8	2.1	3.8	3.5	0.3	224^2
FasterNet-T0 (2023)	71.9	2.5	6.8	3.9	0.3	224^2
EdgeViT-XXS (2022)	74.4	8.8	15.7	4.1	0.6	224^2
MobileOne-S1 (2023)	74.6 (75.9)	1.5	6.9	4.8	0.8	224^2
MobileViT-XS (2022)	74.8	4.1	21.0	2.3	1.1	256^2
EfficientFormerV2-S0 (2023b)	73.7 (75.7)	3.3	10.7	3.6	0.4	224^2
EfficientMod-xxs	**76.0**	3.0	10.2	4.7	0.6	224^2
MobileNetV2×1.4 (2018)	74.7	2.8	6.0	6.1	0.6	224^2
DeiT-T (2021a)	74.5	2.7	16.5	5.9	1.2	224^2
FasterNet-T1 (2023)	76.2	3.3	12.9	7.6	0.9	224^2
EfficientNet-B0 (2019)	77.1	3.4	10.9	5.3	0.4	224^2
MobileOne-S2 (2023)	- (77.4)	2.0	10.0	7.8	1.3	224^2
EdgeViT-XS (2022)	77.5	11.8	21.4	6.8	1.1	224^2
MobileViTv2-1.0 (2023)	78.1	5.4	30.9	4.9	1.8	256^2
EfficientFormerV2-S1 (2023b)	77.9 (79.0)	4.5	15.4	6.2	0.7	224^2
EfficientMod-xs	**78.3**	3.6	13.4	6.6	0.8	224^2
PoolFormer-s12 (2022a)	77.2	5.0	22.3	11.9	1.8	224^2
FasterNet-T2 (2023)	78.9	4.4	18.4	15.0	1.9	224^2
EfficientFormer-L1 (2022)	79.2	3.7	19.7	12.3	1.3	224^2
MobileFormer-508M (2022b)	79.3	13.4	142.5	14.8	0.6	224^2
MobileOne-S4▲ (2023)	- (79.4)	4.8	26.6	14.8	3.0	224^2
MobileViTv2-1.5 (2023)	80.4	7.2	59.0	10.6	4.1	256^2
EdgeViT-S (2022)	**81.0**	20.5	34.7	13.1	1.9	224^2
EfficientFormerV2-S2 (2023b)	80.4 (81.6)	7.3	26.5	12.7	1.3	224^2
EfficientMod-s	**81.0**	5.5	23.5	12.9	1.4	224^2

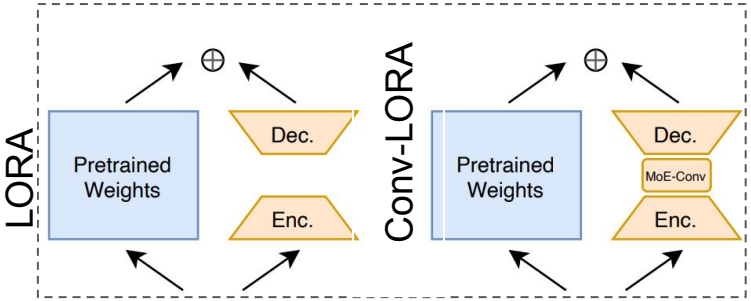
Replace Query-Key subnets with an FC-Conv-FC block → Low latency across GPU & CPU.

b-2) Efficient Fine-tuning

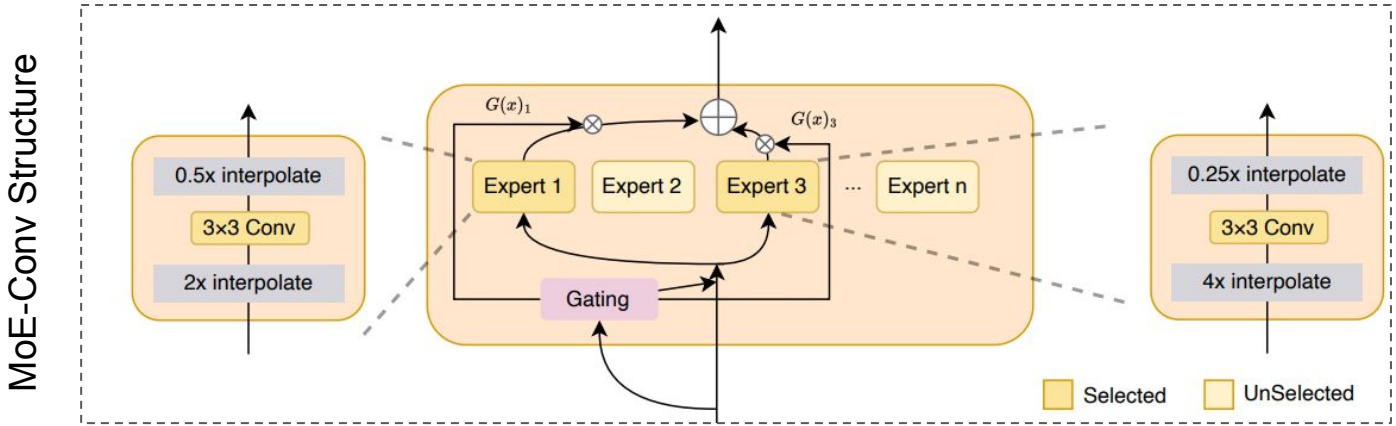
1

CONVOLUTION MEETS LORA: PARAMETER EFFICIENT FINETUNING FOR SAM

The authors freeze SAM, adds few trainable modules (Conv-LORA) for efficient fine-tuning.



MoE-Conv consists of few experts, each processing different feature scales.

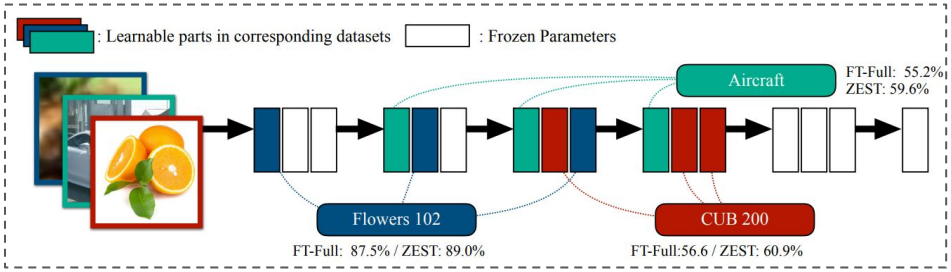


1 CONVOLUTION MEETS LORA: PARAMETER EFFICIENT FINETUNING FOR SAM

Method	#Params (M) / Ratio (%)	Medical						Natural Images				Agriculture		Remote Sensing	
		Kvasir		CVC-612		ISIC 2017		CAMO			SBU	Leaf		Road	
		$S_\alpha \uparrow$	$E_\phi \uparrow$	$S_\alpha \uparrow$	$E_\phi \uparrow$	Jac ↑	Dice ↑	$S_\alpha \uparrow$	$E_\phi \uparrow$	$F_\beta^\omega \uparrow$	BER ↓	IoU ↑	Dice ↑	IoU ↑	Dice ↑
Domain Specific	* / 100%	90.9	94.4	92.6	95.5	80.1	87.5	80.8	85.8	73.1	3.56	62.3	74.1	59.1	73.0
decoder-only	3.51 / 0.55%	86.5	89.5	85.5	89.9	69.7	79.5	78.5	83.1	69.8	14.58	50.8	63.8	48.6	65.1
BitFit	3.96 / 0.62%	90.8 ± 0.57	93.8 ± 0.98	89.0 ± 0.40	91.6 ± 0.98	76.4 ± 0.45	84.7 ± 0.35	86.8 ± 0.33	90.7 ± 0.28	81.5 ± 0.19	3.16 ± 0.128	71.4 ± 1.15	81.7 ± 1.01	60.6 ± 0.15	75.2 ± 0.11
Adapter	3.92 / 0.61%	91.2 ± 0.23	94.0 ± 0.16	89.3 ± 0.43	92.0 ± 0.63	76.7 ± 0.66	85.0 ± 0.56	87.7 ± 0.10	91.3 ± 0.40	82.8 ± 0.35	2.84 ± 0.093	72.1 ± 0.47	82.4 ± 0.36	61.5 ± 0.11	75.9 ± 0.12
VPT	4.00 / 0.62%	91.5 ± 0.23	94.3 ± 0.06	91.0 ± 0.94	76.9 ± 0.94	76.9 ± 0.94	85.1 ± 0.75	87.4 ± 0.60	91.4 ± 0.68	82.1 ± 0.75	2.70 ± 0.055	73.6 ± 0.26	83.8 ± 0.26	60.2 ± 1.87	74.9 ± 1.50
LST	11.49 / 1.77%	89.7 ± 0.25	93.3 ± 0.37	89.4 ± 0.37	92.4 ± 0.54	76.4 ± 1.05	84.9 ± 0.79	83.3 ± 0.28	88.0 ± 0.23	77.1 ± 0.02	3.18 ± 0.012	70.2 ± 0.87	81.1 ± 0.82	60.2 ± 0.26	74.9 ± 0.22
SAM-Adapter	3.98 / 0.62%	89.6 ± 0.24	92.5 ± 0.10	89.6 ± 0.22	92.4 ± 1.06	76.1 ± 0.45	84.6 ± 0.37	85.6 ± 0.26	89.6 ± 0.55	79.8 ± 0.89	3.14 ± 0.063	71.4 ± 0.20	82.1 ± 0.10	60.6 ± 0.06	75.2 ± 0.04
SSF	4.42 / 0.69%	91.3 ± 0.87	93.9 ± 1.49	89.6 ± 0.37	91.9 ± 0.79	76.6 ± 0.19	85.0 ± 0.14	87.5 ± 0.11	91.4 ± 0.16	82.6 ± 0.12	3.19 ± 0.046	71.5 ± 0.63	81.8 ± 0.44	61.6 ± 0.03	76.0 ± 0.02
LoRA	4.00 / 0.62%	91.2 ± 0.28	93.8 ± 0.22	90.7 ± 0.04	92.5 ± 0.41	76.6 ± 0.23	84.9 ± 0.22	88.0 ± 0.24	91.9 ± 0.42	82.8 ± 0.16	2.74 ± 0.079	73.7 ± 0.20	83.6 ± 0.13	62.2 ± 0.21	76.5 ± 0.18
Conv-LoRA	4.02 / 0.63%	**92.0** ± 0.15	**94.7** ± 0.16	**91.3** ± 0.69	**94.0** ± 0.78	**77.6** ± 0.57	**85.7** ± 0.36	**88.3** ± 0.40	**92.4** ± 0.31	**84.0** ± 0.34	**2.54** ± 0.081	**74.5** ± 0.39	**84.3** ± 0.34	**62.6** ± 0.36	**76.8** ± 0.27

Conv-LORA leads to some improvement over the strong competitor LORA on few-labeled tasks.

2 ZEST: ZEROSHOT SPARSE FINE-TUNING

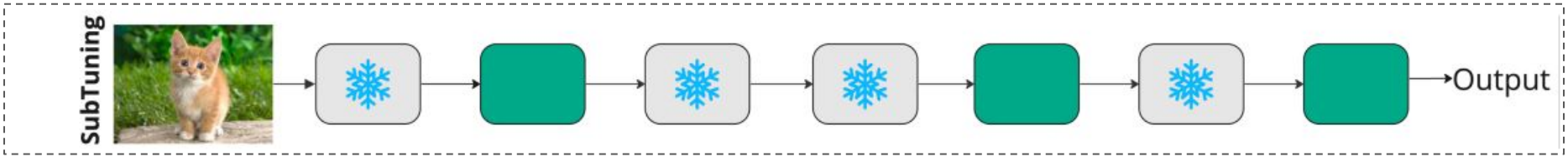


3 HOW TO FINE-TUNE VISION MODELS WITH SGD

	ID accuracy	OOD accuracy
SGD	90.0%	67.9%
AdamW	(+2.1%)	(+8.1%)
SGD (freeze-embed)	(+2.1%)	(+8.0%)
SGD (freeze-embed, no mom.)	(+2.2%)	**(+9.0%)**

(b) Performance of different fine-tuning methods on a CLIP ViT-B/16 averaged over 5 distribution shift datasets.

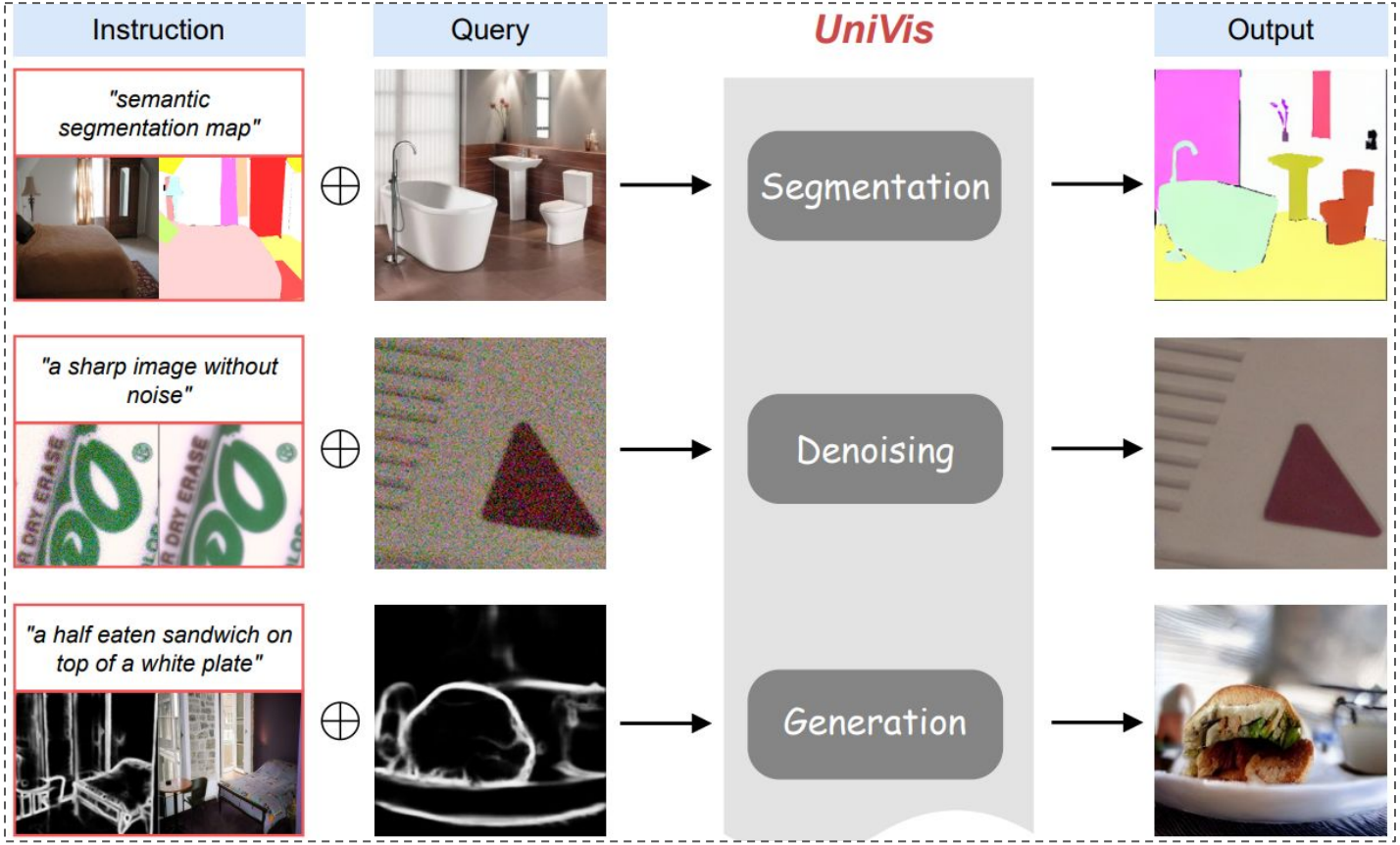
4 LESS IS MORE: SELECTIVE LAYER FINE-TUNING WITH SUBTUNING



c) Foundation Models

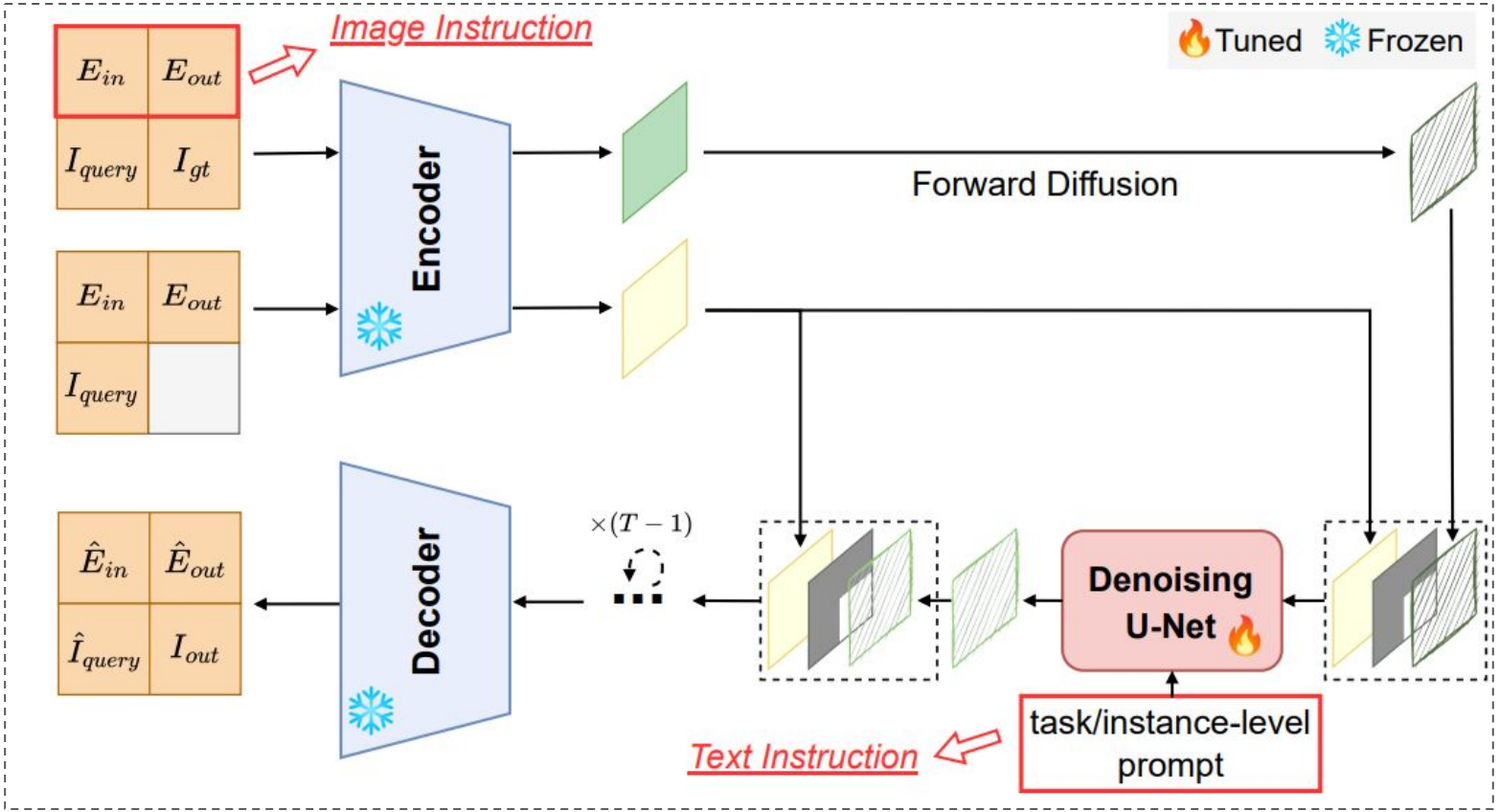
1

UNIVIS: A UNIVERSAL FRAMEWORK FOR COMPUTER VISION TASKS



Perform many different vision tasks (segmentation, denoising, generation) by differing the instruction.

1 UNIVIS: A UNIVERSAL FRAMEWORK FOR COMPUTER VISION TASKS



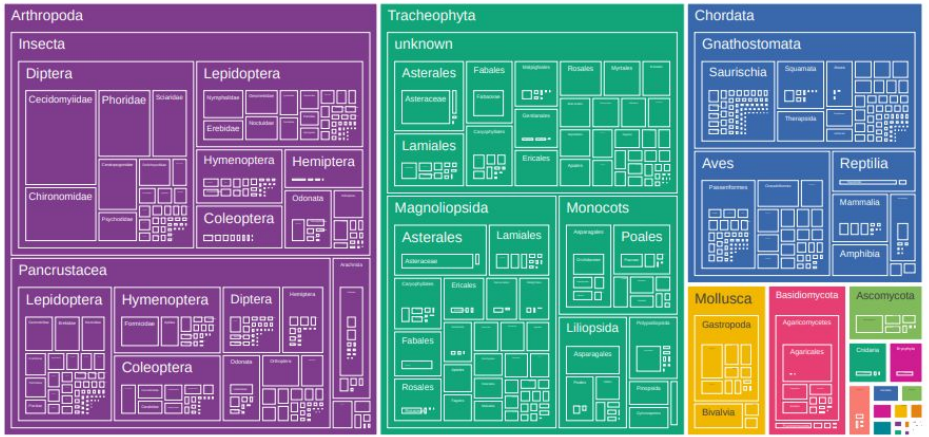
The model builds upon Stable Diffusion, and generates unified input-output prompts for different tasks.

2 SLIME: SEGMENT LIKE ME

Figure 1: ***SLiMe.*** Using just one user-annotated image with various granularity (as shown in the leftmost column), *SLiMe* learns to segment different unseen images in accordance with the same granularity (as depicted in the other columns).

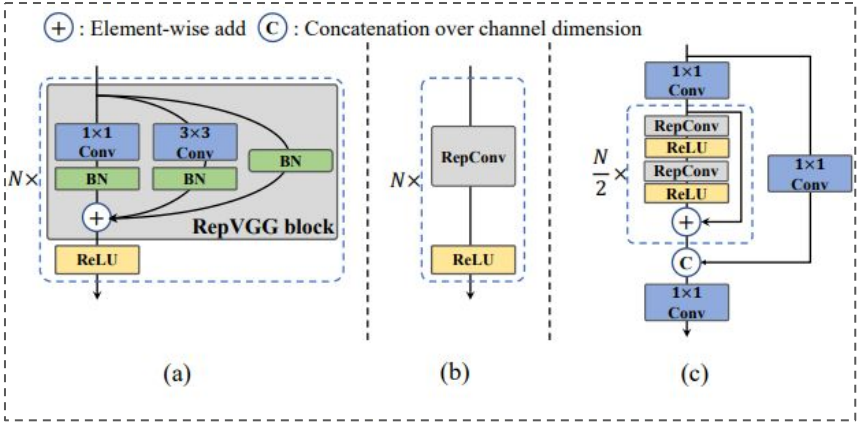
SLIME can learn to segment objects at different granularity from a single image, interactively.

3 [BIOCLIP: A VISION FOUNDATION MODEL FOR THE TREE OF LIFE]



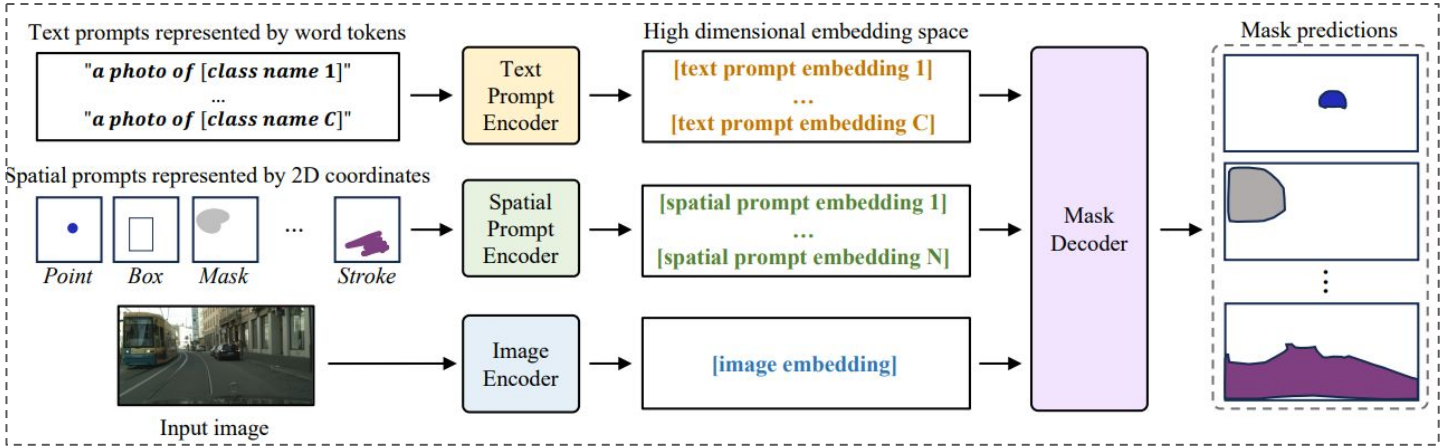
(a) Treemap of the different phyla in TREEOFLIFE-10M. Different colors are different phyla; nested boxes represent classes, orders, etc.

4 [YOLOV6: OBJECT DETECTION FOR INDUSTRY]

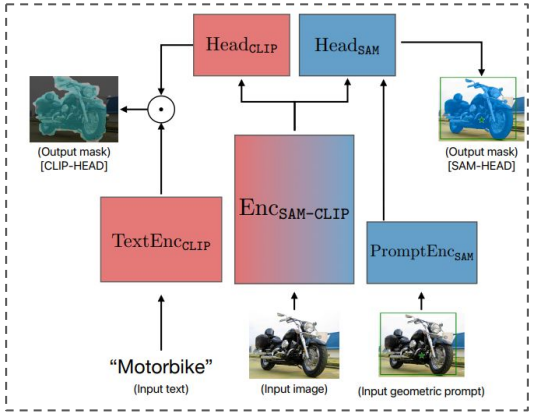


Train with (a) RepVGG & Infer with (c) RepConv.

5 LEARNING TO PROMPT SAM



6 SAM-CLIP: MERGING CLIP and SAM



Other Interesting Papers

1. THE ALL-SEEING PROJECT: TOWARDS PANOPTIC VISUAL RECOGNITION AND UNDERSTANDING OF THE OPEN WORLD
2. IN DEFENSE OF SEMI-SUPERVISED LEARNING: SELF SUPERVISION IS NOT ALL YOU NEED
3. SIMPLIFYING SELF-SUPERVISED OBJECT DETECTION PRETRAINING
4. LARGE LANGUAGE MODELS AS OPTIMIZERS
5. IMAGE COMPRESSION IS AN EFFECTIVE OBJECTIVE FOR VISUAL REPRESENTATION LEARNING
6. AUTOVP: AN AUTOMATED VISUAL PROMPTING FRAMEWORK AND BENCHMARK

Code Repositories

Paper	Code
FROZEN TRANSFORMERS IN LANGUAGE MODELS ARE EFFECTIVE VISUAL ENCODER LAYERS	Link
MiniGPT-V2	Link
MiniGPT-V4	Link
PLANTING A SEED OF VISION IN LARGE LANGUAGE MODEL	Link
FASTERVIT: FAST VISION TRANSFORMERS WITH HIERARCHICAL ATTENTION	Link
EFFICIENT MODULATION FOR VISION NETWORKS	Link
SLIME: SEGMENT LIKE ME	Link
YOLOV6: OBJECT DETECTION FOR INDUSTRY	Link

Discussion

~Model Interactivity~

Soon, will most models become interactive (with the help of an LLM)?

Can vision annotations turn into human language form rather than spatial (e.g. bounding box)?

~Model Efficiency~

Replace FC blocks with Convs for faster ViT variants → Can this be automated?

~Training Efficiency~

How to find the best (few) layers to optimize for our downstream tasks and data?

Thanks! Questions?