ICLR

International Conference On Learning Representations

ICLR 2024 Potpourri

Mert Kilickaya

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

Pre-filtering based on the title and the abstract: ~**200** papers

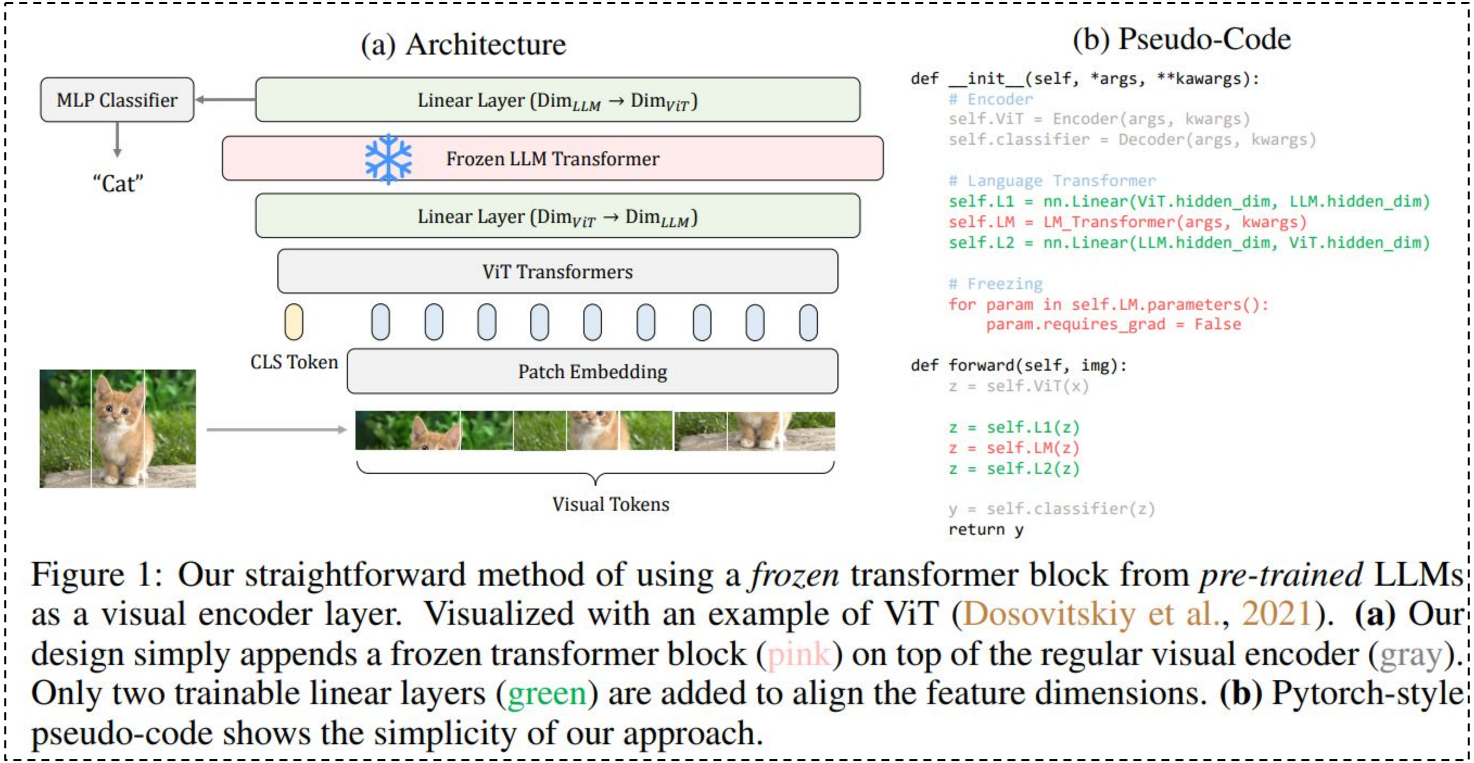
Post-filtering based on content: **20** papers

Four Topics: LLM & Efficiency & ViT & Industry

a) LLM for Vision

1

FROZEN TRANSFORMERS IN LANGUAGE MODELS ARE EFFECTIVE VISUAL ENCODER LAYERS


(a) Architecture
MLP Classifier
Linear Layer ($\mathrm{Dim}_{LLM} \rightarrow \mathrm{Dim}_{ViT}$)
Frozen LLM Transformer
"Cat"
Linear Layer ($\mathrm{Dim}_{ViT} \rightarrow \mathrm{Dim}_{LLM}$)
ViT Transformers
CLS Token
Patch Embedding
Visual Tokens
(b) Pseudo-Code

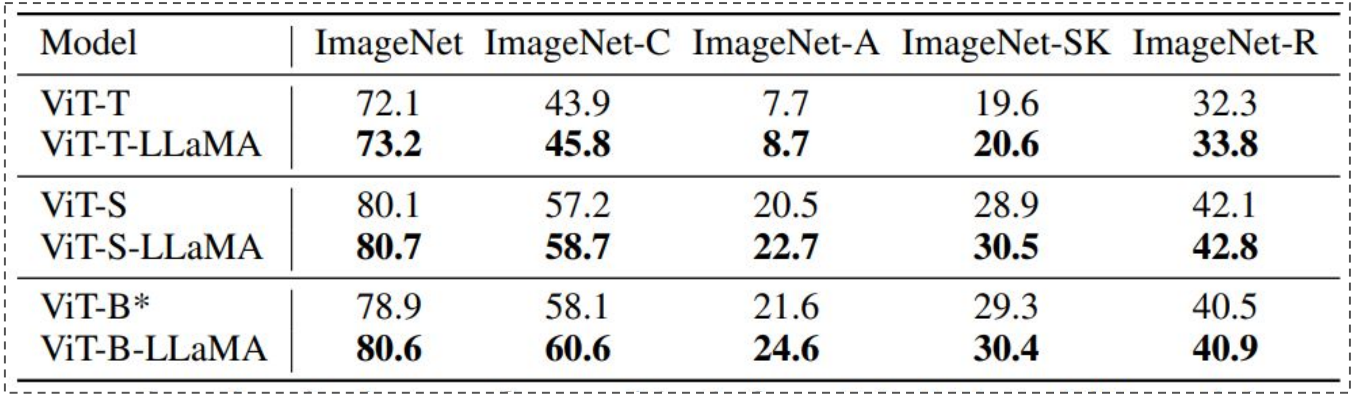
```
def __init__(self, *args, **kawargs):
    # Encoder
    self.ViT = Encoder(args, kwargs)
    self.classifier = Decoder(args, kwargs)

    # 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

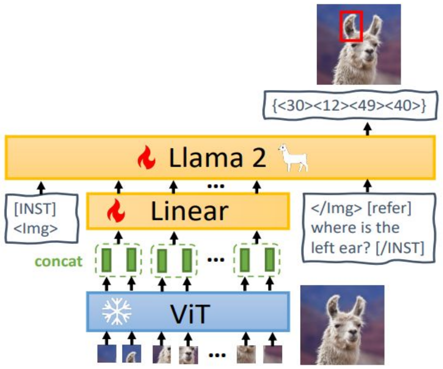
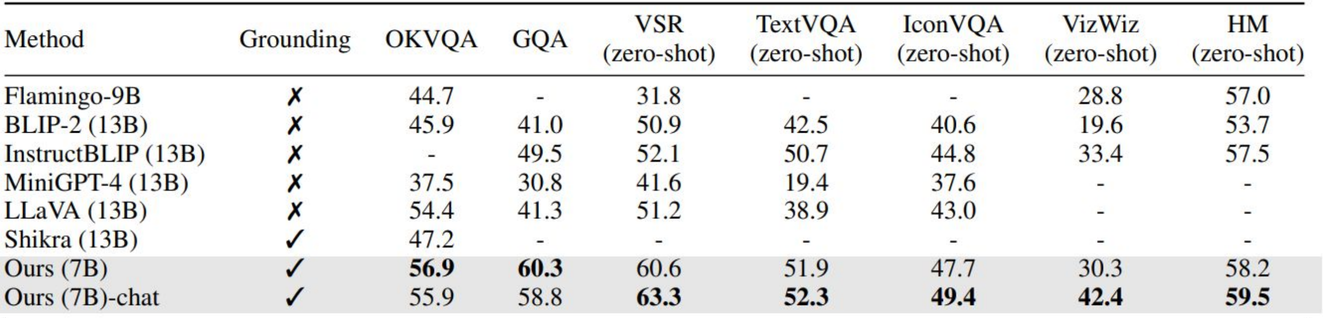


Figure 2: **Architecture of MiniGPT-v2.** The model takes a ViT visual backbone, which remains frozen during all training phases. We concatenate four adjacent visual output tokens from ViT backbone and project them into LLaMA-2 language model space via a linear projection layer.

Results on Visual Question Answering



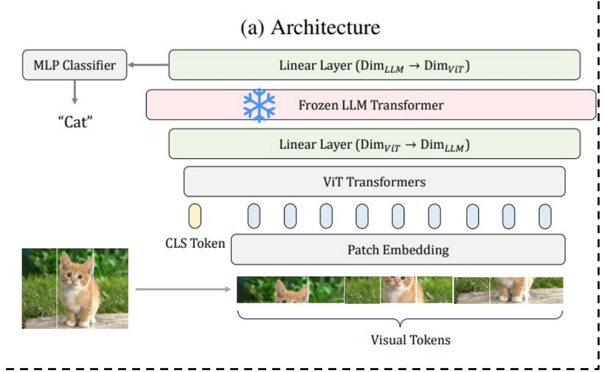
Method	Grounding	OKVQA	GQA	VSR (zero-shot)	TextVQA (zero-shot)	IconVQA (zero-shot)	VizWiz (zero-shot)	HM (zero-shot)
Flamingo-9B	✗	44.7	-	31.8	-	-	28.8	57.0
BLIP-2 (13B)	✗	45.9	41.0	50.9	42.5	40.6	19.6	53.7
InstructBLIP (13B)	✗	-	49.5	52.1	50.7	44.8	33.4	57.5
MiniGPT-4 (13B)	✗	37.5	30.8	41.6	19.4	37.6	-	-
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

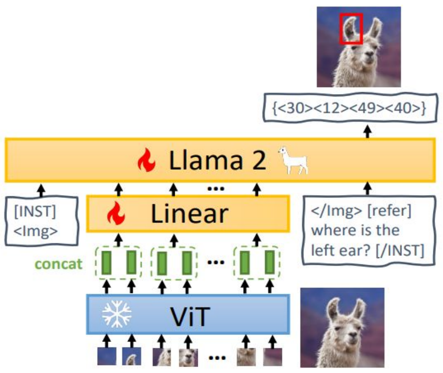
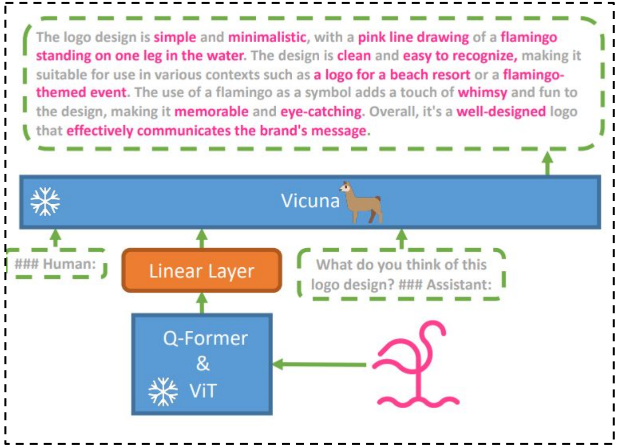


Figure 2: **Architecture of MiniGPT-v2.** The model takes a ViT visual backbone, which remains frozen during all training phases. We concatenate four adjacent visual output tokens from ViT backbone and project them into LLaMA-2 language model space via a linear projection layer.

3

MiniGPT-V4



The logo design is simple and minimalistic, with a pink line drawing of a flamingo standing on one leg in the water. The design is clean and easy to recognize, making it suitable for use in various contexts such as a logo for a beach resort or a flamingo-themed event. The use of a flamingo as a symbol adds a touch of whimsy and fun to the design, making it memorable and eye-catching. Overall, it's a well-designed logo that effectively communicates the brand's message.

4

PLANTING A SEED OF VISION IN LARGE LANGUAGE MODEL

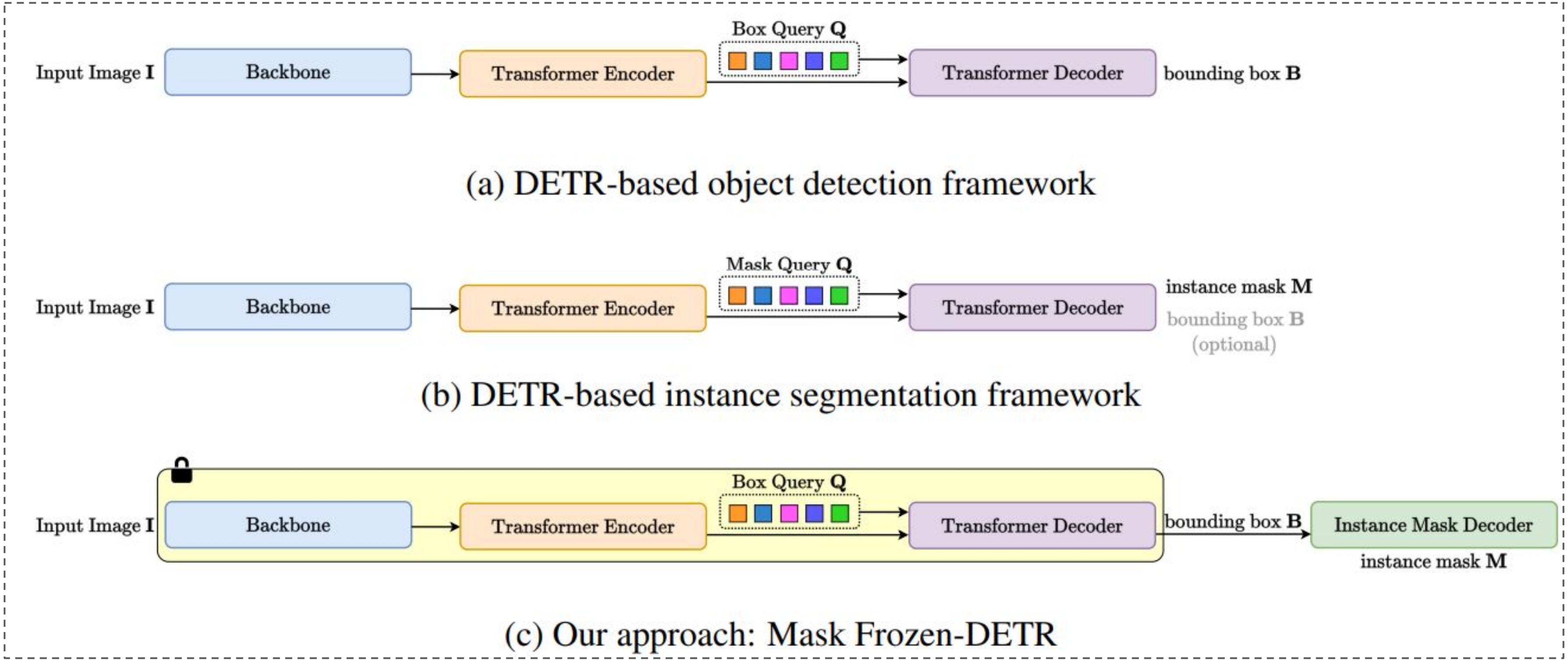
5

LINGUISTIC IMAGE UNDERSTANDING

b) Efficient Training

1

[MASK FROZEN-DETR: HIGH QUALITY INSTANCE SEGMENTATION WITH ONE GPU]

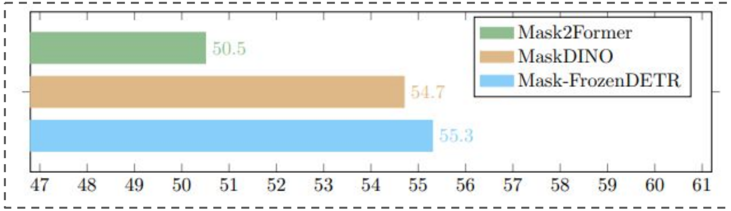


i. Take a pre-trained detector **ii.** Freeze it **iii.** Fine-tune the mask head **iv.** SOTA segmenter in few epochs

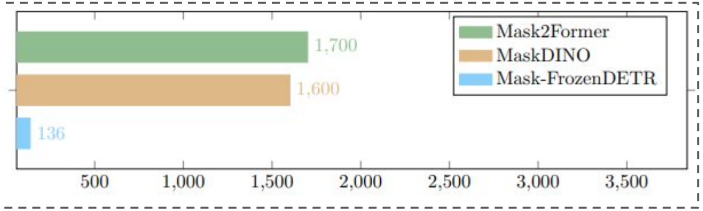
1

MASK FROZEN-DETR: HIGH QUALITY INSTANCE SEGMENTATION WITH ONE GPU

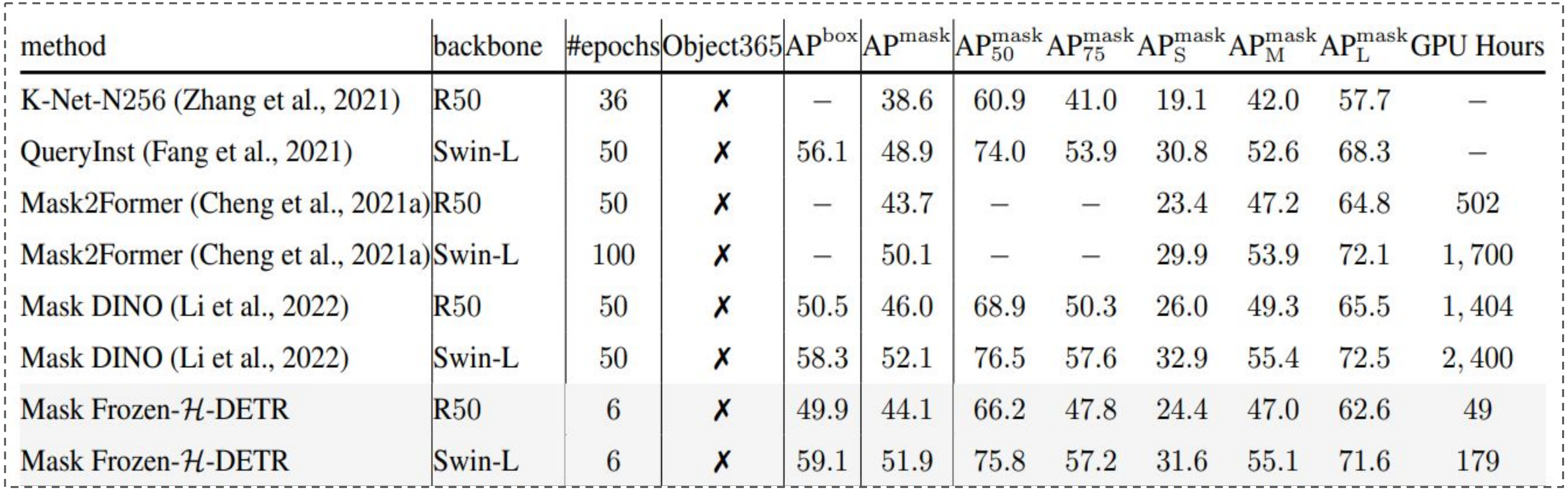
Mask AP



GPU Hours



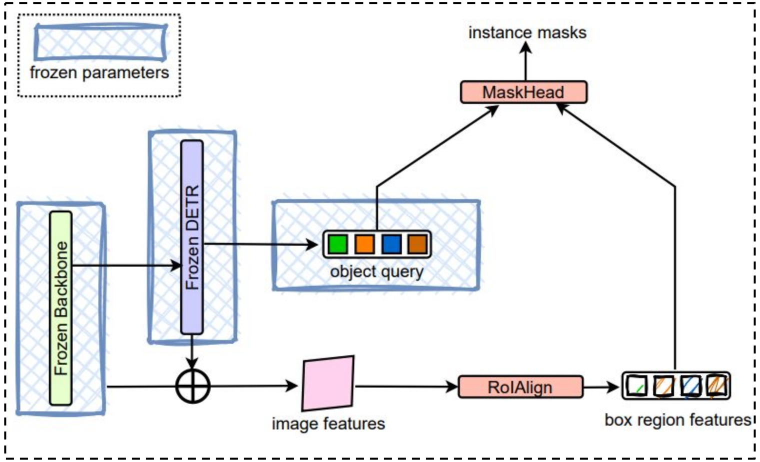
Extensive Comparison



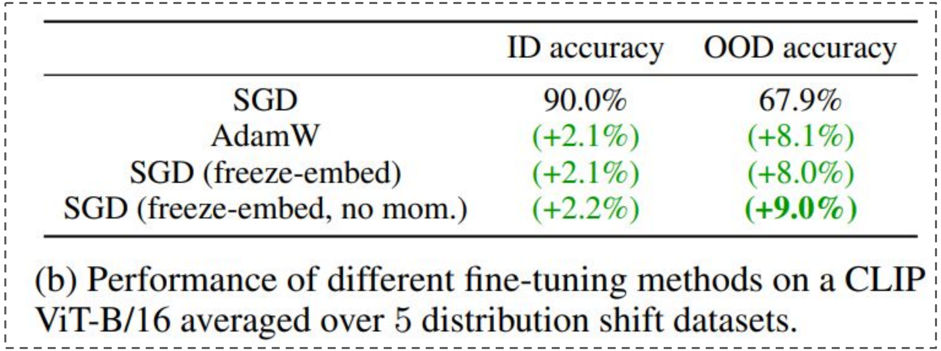
method	backbone	#epochs	Object365	AP^{box}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	AP^{mask}_{S}	AP^{mask}_{M}	AP^{mask}_{L}	GPU Hours
K-Net-N256 (Zhang et al., 2021)	R50	36	✗	–	38.6	60.9	41.0	19.1	42.0	57.7	–
QueryInst (Fang et al., 2021)	Swin-L	50	✗	56.1	48.9	74.0	53.9	30.8	52.6	68.3	–
Mask2Former (Cheng et al., 2021a)	R50	50	✗	–	43.7	–	–	23.4	47.2	64.8	502
Mask2Former (Cheng et al., 2021a)	Swin-L	100	✗	–	50.1	–	–	29.9	53.9	72.1	1, 700
Mask DINO (Li et al., 2022)	R50	50	✗	50.5	46.0	68.9	50.3	26.0	49.3	65.5	1, 404
Mask DINO (Li et al., 2022)	Swin-L	50	✗	58.3	52.1	76.5	57.6	32.9	55.4	72.5	2, 400
Mask Frozen-$\mathcal{H}$-DETR	R50	6	✗	49.9	44.1	66.2	47.8	24.4	47.0	62.6	49
Mask Frozen-$\mathcal{H}$-DETR	Swin-L	6	✗	59.1	51.9	75.8	57.2	31.6	55.1	71.6	179

MaskFrozen outperforms MaskDINO, while training much more efficiently (i.e. >10x less GPU hours)

1 Mask-Frozen DETR



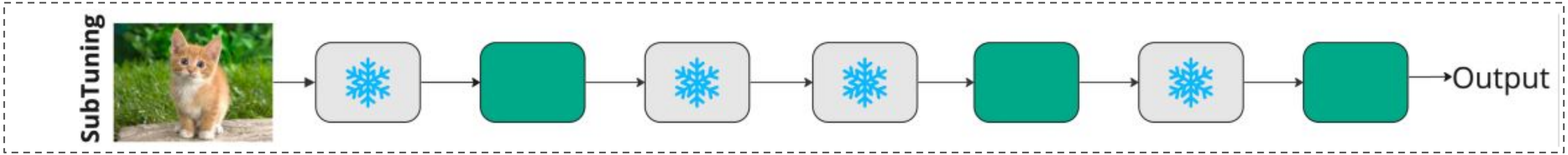
2 HOW TO FINE-TUNE VISION MODELS WITH SGD



	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.

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



c) Vision-Transformers

1

FASTERVIT: FAST VISION TRANSFORMERS WITH HIERARCHICAL ATTENTION

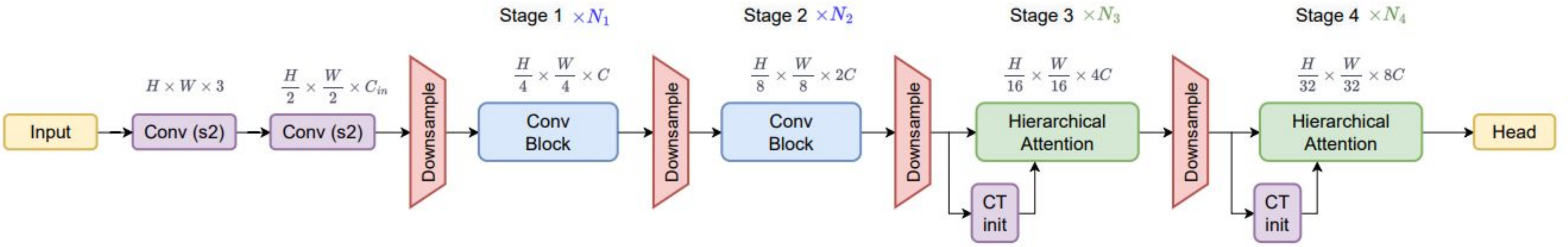


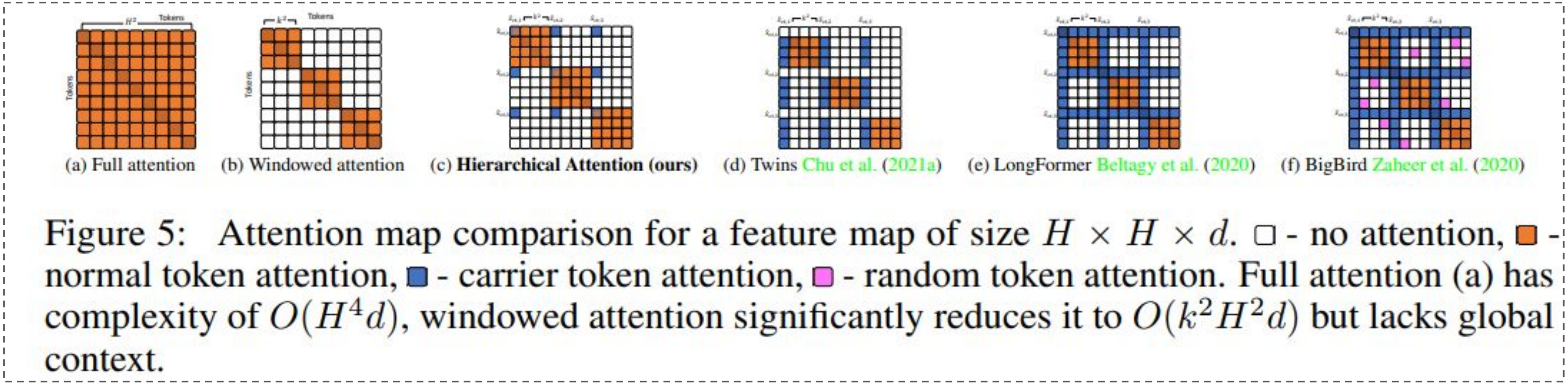
Figure 3: Overview of the FasterViT architecture. We use a multi-scale architecture with CNN and transformer-based blocks in stages 1, 2 and 3, 4, respectively. Best viewed in color.

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



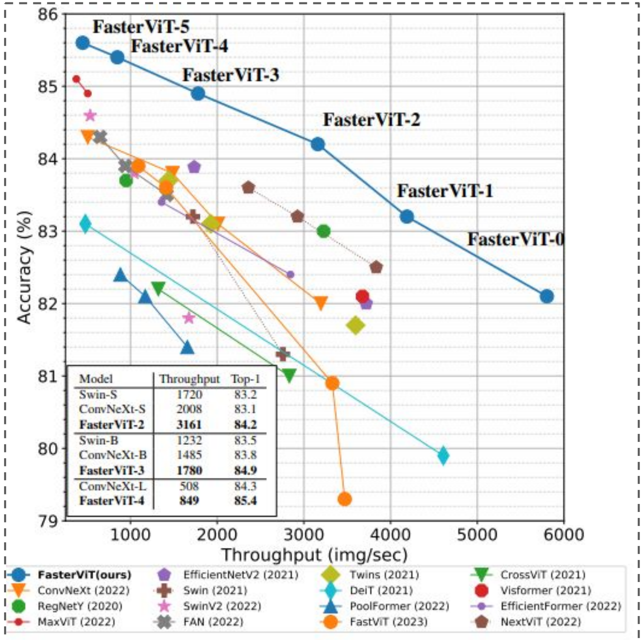
(a) Full attention (b) Windowed attention (c) **Hierarchical Attention (ours)** (d) Twins Chu et al. (2021a) (e) LongFormer Beltagy et al. (2020) (f) BigBird Zaheer et al. (2020)

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 carrie tokens to pool from distinct windows.

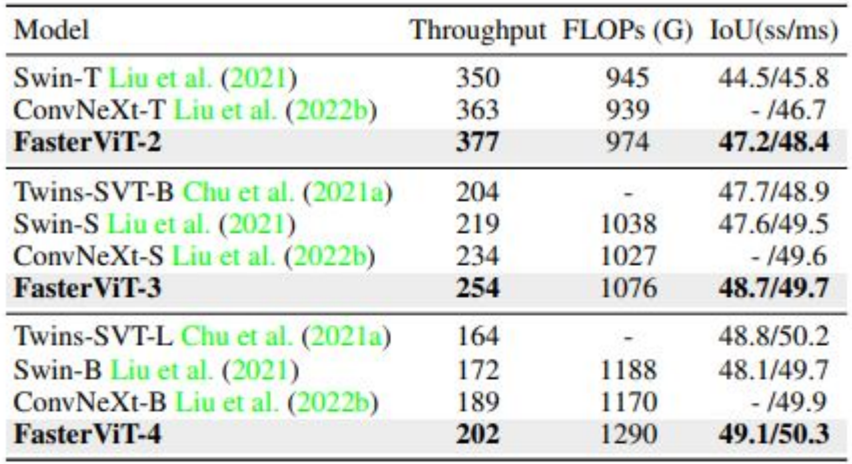
1

FASTERVIT: FAST VISION TRANSFORMERS WITH HIERARCHICAL ATTENTION



Model	Throughput	Top-1
Swin-S	1720	83.2
ConvNeXt-S	2008	83.1
FasterViT-2	**3161**	**84.2**
Swin-B	1232	83.5
ConvNeXt-B	1485	83.8
FasterViT-3	**1780**	**84.9**
ConvNeXt-L	508	84.3
FasterViT-4	**849**	**85.4**

igure 1: Comparison of image throughput and mageNet-1K Top-1 accuracy. For all models, hroughput is measured on A100 GPU with batch ize of 128.



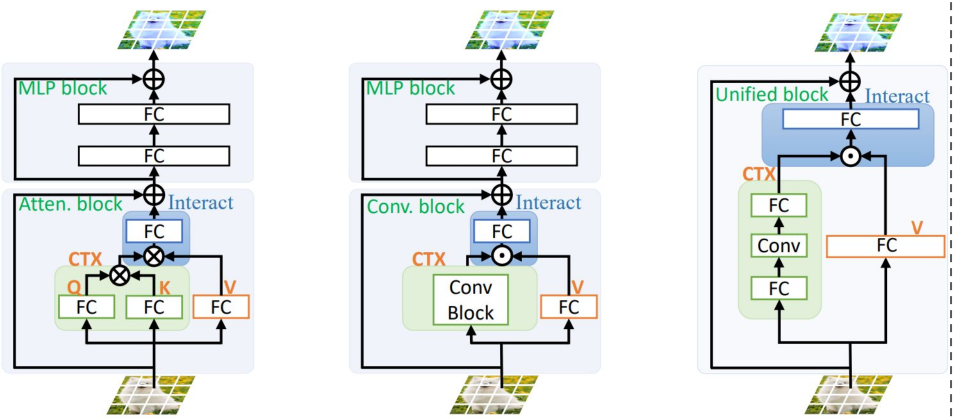
Model	Throughput	FLOPs (G)	IoU(ss/ms)
Swin-T Liu et al. (2021)	350	945	44.5/45.8
ConvNeXt-T Liu et al. (2022b)	363	939	- /46.7
FasterViT-2	**377**	974	**47.2/48.4**
Twins-SVT-B Chu et al. (2021a)	204	-	47.7/48.9
Swin-S Liu et al. (2021)	219	1038	47.6/49.5
ConvNeXt-S Liu et al. (2022b)	234	1027	- /49.6
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).

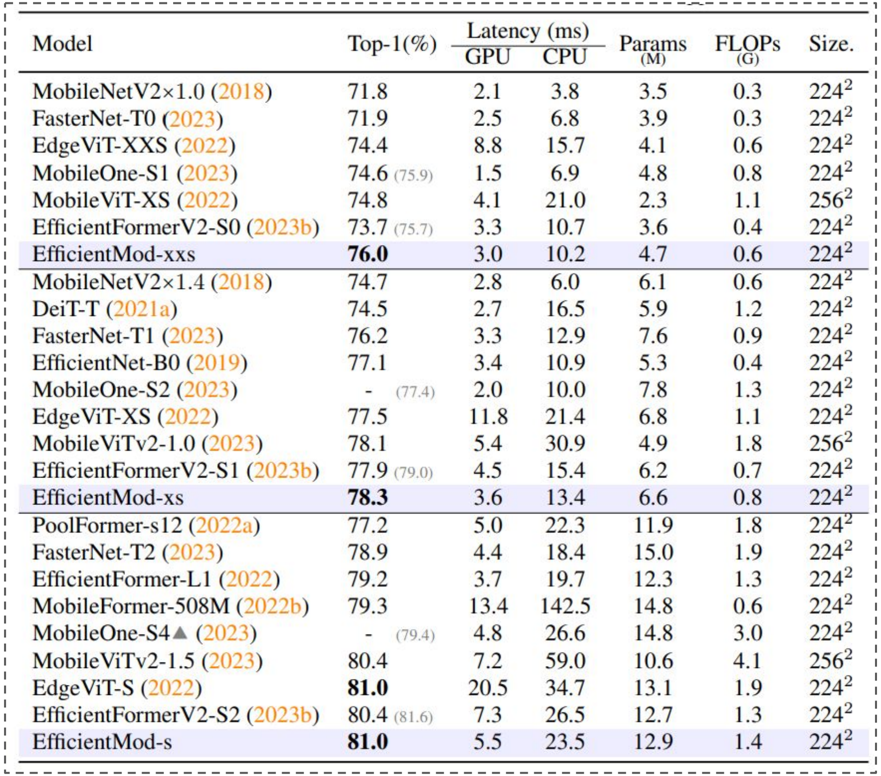
FasterViT yields the highest accuracy-efficiency tradeoff for classification/segmentation tasks (ADE20K).

2

EFFICIENT MODULATION FOR VISION NETWORKS



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



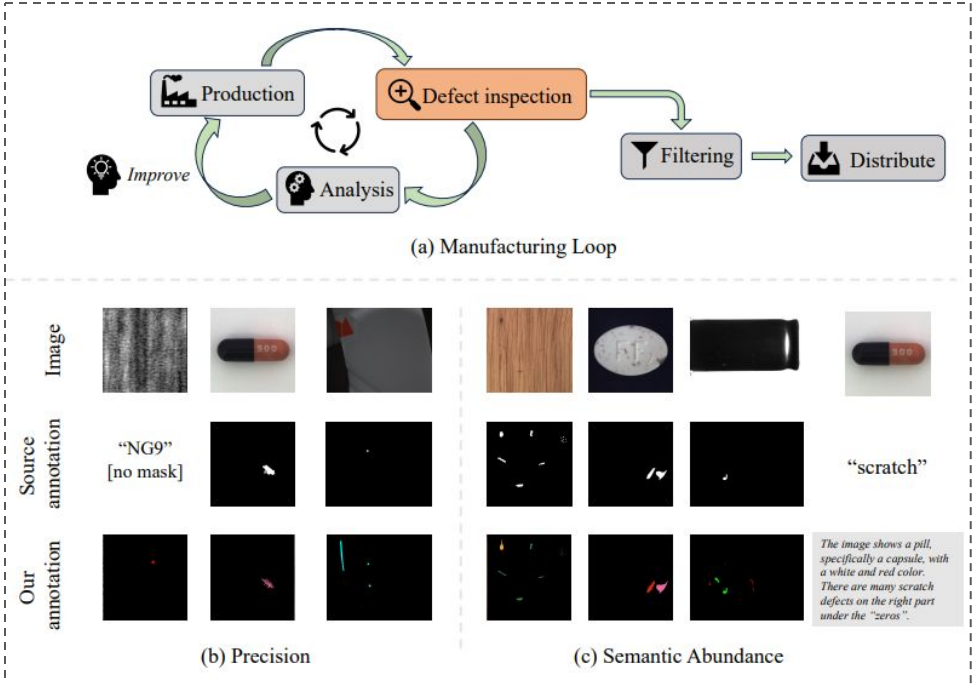
Model	Top-1(%)	Latency (ms) GPU	Latency (ms) CPU	Params (M)	FLOPs (G)	Size.
MobileNetV2×1.0 (2018)	71.8	2.1	3.8	3.5	0.3	224^2
FasterNet-T0 (2023)	71.9	2.5	6.8	3.9	0.3	224^2
EdgeViT-XXS (2022)	74.4	8.8	15.7	4.1	0.6	224^2
MobileOne-S1 (2023)	74.6 (75.9)	1.5	6.9	4.8	0.8	224^2
MobileViT-XS (2022)	74.8	4.1	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.

d) Industrial Vision

1

DEFECT SPECTRUM: A GRANULAR LOOK OF LARGE-SCALE DEFECT DATASETS WITH RICH SEMANTICS



The image shows a pill, specifically a capsule, with a white and red color. There are many scratch defects on the right part under the "zeros".

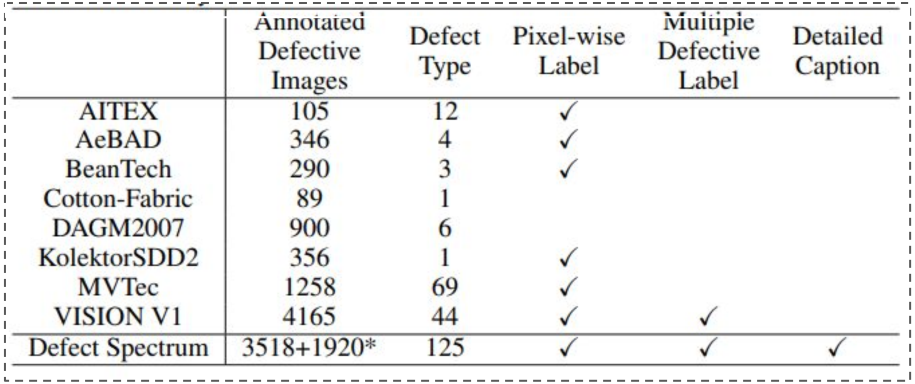
Figure 1: (a) The closed-loop system in industrial manufacturing. Defect inspection plays a pivotal role. (b, c) An overview of our improvements in annotations, in the aspect of precision and semantics abundance. **Best viewed in color.**

The authors present: **1)** Fine-grained defect annotations, **2)** Synthetic defect generator via diffusion models.

1

DEFECT SPECTRUM: A GRANULAR LOOK OF LARGE-SCALE DEFECT DATASETS WITH RICH SEMANTICS

Dataset Annotation Comparison



	Annotated Defective Images	Defect Type	Pixel-wise Label	Multiple Defective Label	Detailed Caption
AITEX	105	12	✓		
AeBAD	346	4	✓		
BeanTech	290	3	✓		
Cotton-Fabric	89	1			
DAGM2007	900	6			
KolektorSDD2	356	1	✓		
MVTec	1258	69	✓		
VISION V1	4165	44	✓	✓	
Defect Spectrum	3518+1920*	125	✓	✓	✓

Training w/ synthetic data improves performance

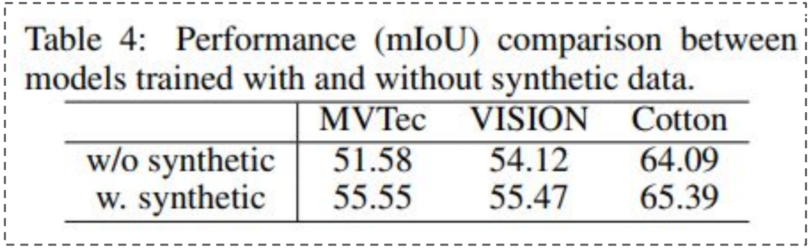


Table 4: Performance (mIoU) comparison between models trained with and without synthetic data.

	MVTec	VISION	Cotton
w/o synthetic	51.58	54.12	64.09
w. synthetic	55.55	55.47	65.39

Comparison of inference time across models. **HR**Netw18small is the fastest.

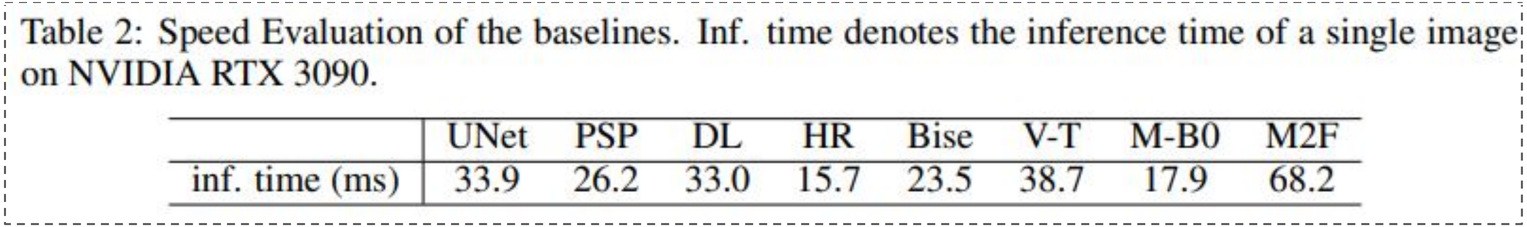


Table 2: Speed Evaluation of the baselines. Inf. time denotes the inference time of a single image on NVIDIA RTX 3090.

	UNet	PSP	DL	HR	Bise	V-T	M-B0	M2F
inf. time (ms)	33.9	26.2	33.0	15.7	23.5	38.7	17.9	68.2

2 SLIME: SEGMENT LIKE ME

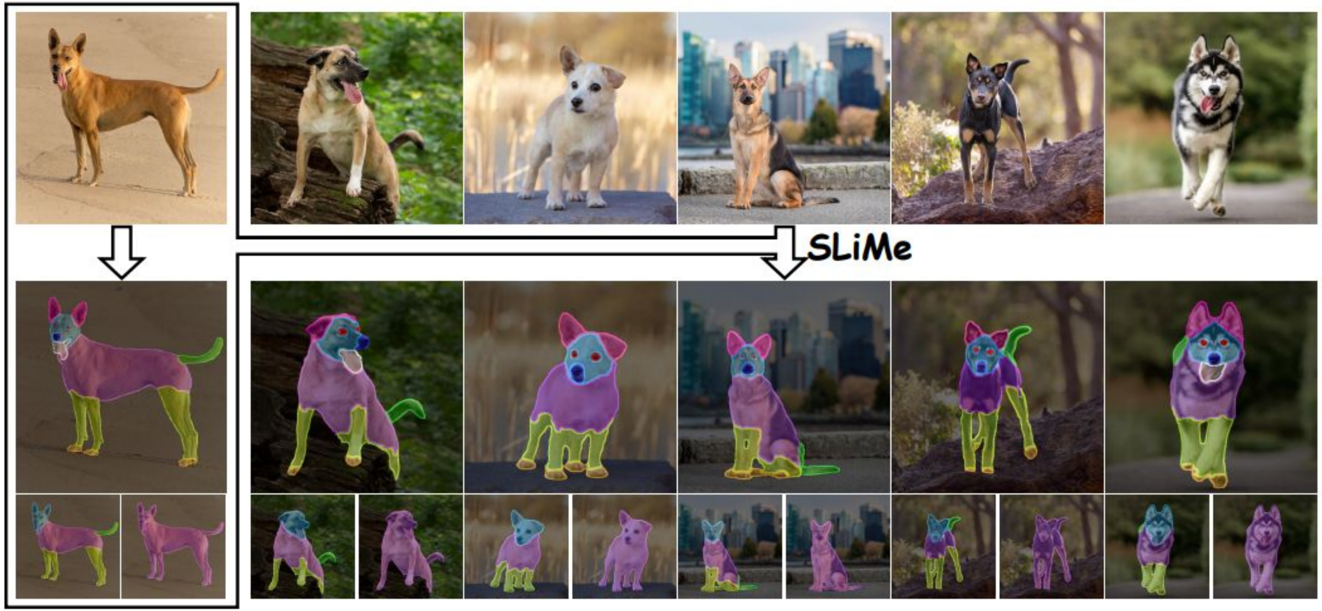
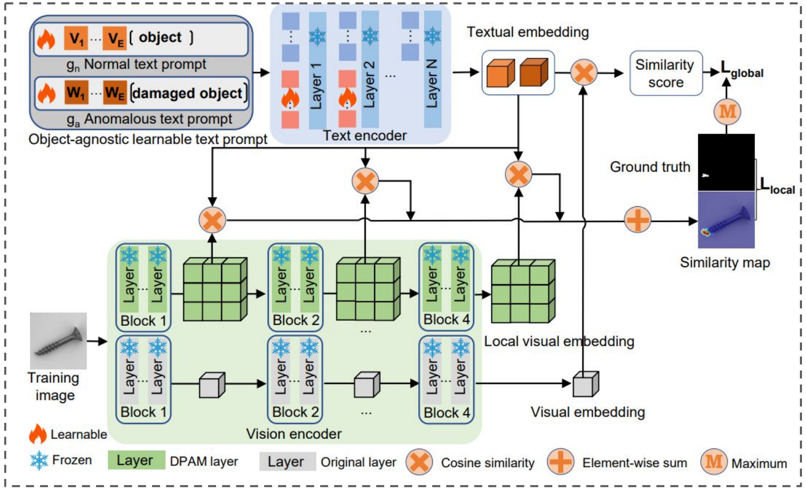


Figure 1: ***SLiMe.*** Using just one user-annotated image with various granularity (as shown in the leftmost column), *SLiMe* learns to segment different unseen images in accordance with the same granularity (as depicted in the other columns).

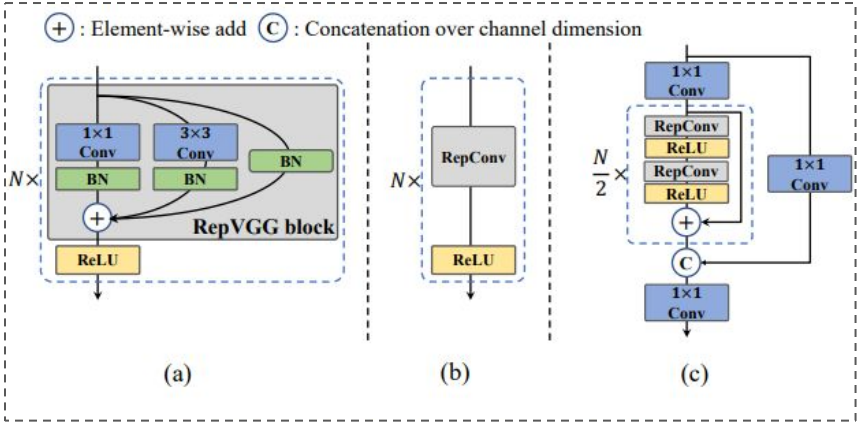
SLIME can learn to segment objects at different granularity from a single image, interactively.

3 ANOMALYCLIP: ZERO-SHOT ANOMALY DETECTION



Prompt CLIP to determine defect regions.

4 YOLOV6: OBJECT DETECTION FOR INDUSTRY



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

Discussion

~Model Interactivity~

Soon, will we all integrate an LLM into our model to interact with & chat and improve it?

Can all annotations turn into 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?