CONVOLUTION MEETS LoRA: PARAMETER EFFICIENT FINETUNING FOR SEGMENT ANYTHING MODEL *[ICLR 24]*

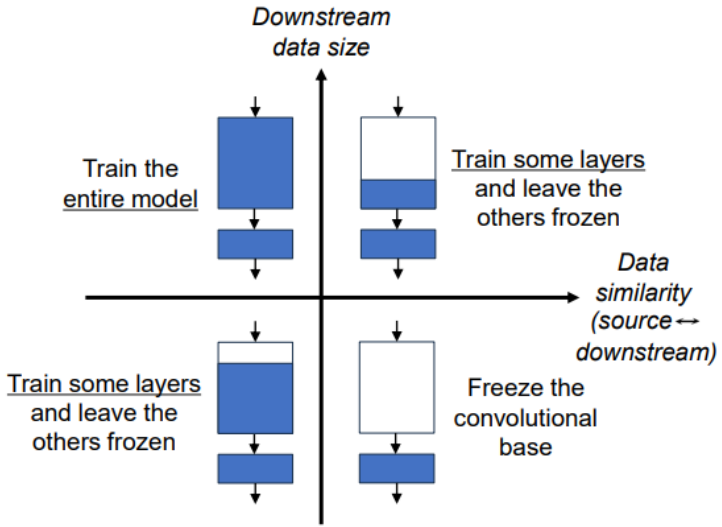
Zihan Zhong, Zhiqiang Tang, Tong He, Haoyang Fang, Chun Yuan
Tsinghua University, Amazon Web Services

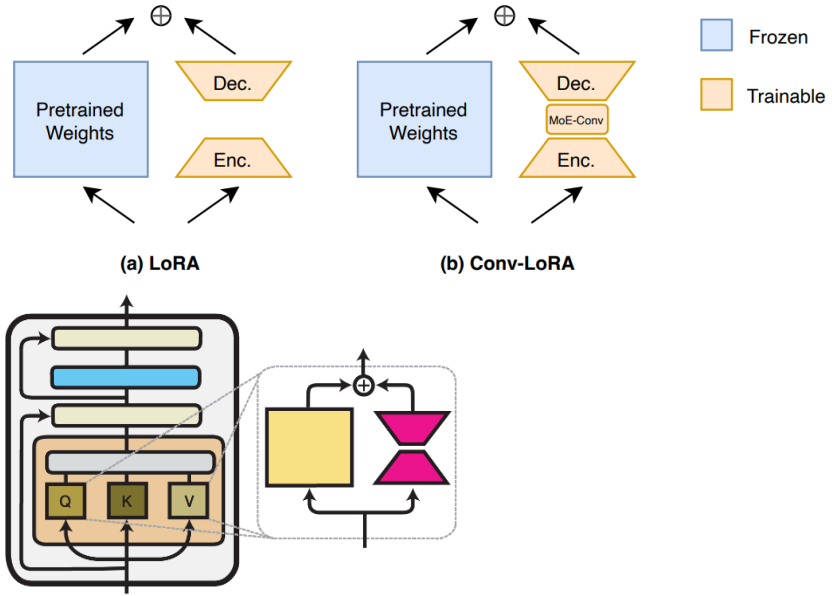
Sunghyun Ahn
skd@yonsei.ac.kr

1 Introduction

Parameter Efficient FineTuning

- **PEFT:** pre-trained 모델의 모든 parameter를 학습시키지 않고, 적은 양의 parameter만 학습하여 사전 지식을 유지하는 기법
- **LoRA:** pre-trained 모델의 weight를 update하지 않고도 fully-tuning한 결과와 비슷하거나 더 좋은 성능을 보인 PEFT 방법
- **ConvLoRA:** LoRA에 Convolution block을 추가하여 vision-specific inductive bias를 추가한 방법





1 Introduction

LoRA *[ICLR 21]*

- **Low Rank Decomposition:** 큰 차원의 Matrix 연산을 낮은 차원으로 분해하는 것
- **Training:** 사전 학습한 모델의 weight들은 업데이트 하지 않고, LoRA의 rank decomposition matrices의 weight들만 업데이트 (기존 크기의 0.01%)
- **Testing:** pretrained weights와 trainable weights를 더하므로, 추가적인 Inference Latency가 발생하지 않음 ($W = W_0 + BA$)

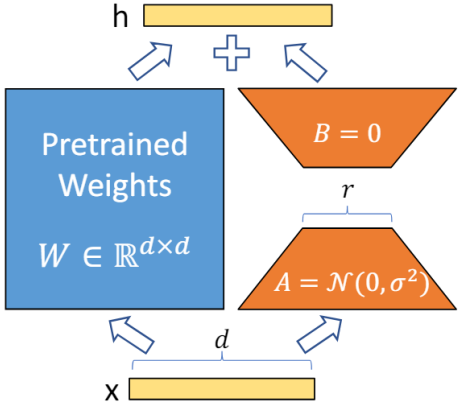


Figure 1: Our reparametrization. We only train A and B.

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

$$W_0 \in \mathbb{R}^{d\times k}, B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k}, r \ll \min(d, k)$$

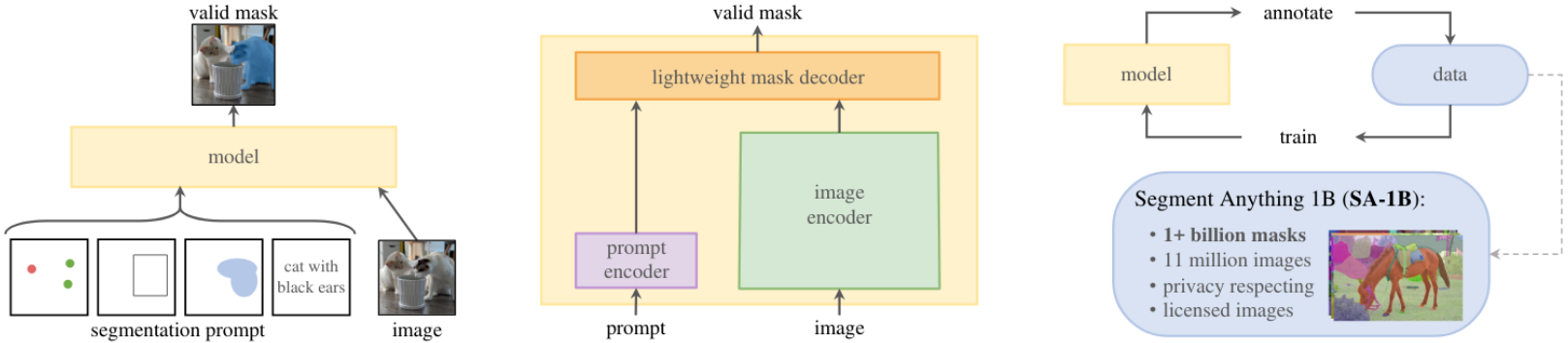
$$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(P_\Phi(y_t|x, y_{<t})\right)$$

$$\max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t|x, y_{<t})\right)$$

1 Introduction

SAM *[ICCV 23]*

- **Task:** zero-shot segmentation을 위한 promptable segmentation 제안 (image + points/bbox/mask/text -> valid mask)
- **Model:** prompt와 image를 입력으로 받아 실시간으로 mask를 예측하는 모델 제안 (encoder: pretrained MAE, CLIP)
- **Data:** SA-1B (Large Instance Segmentation Dataset) 제공 (human-annotated data + model-annotated data)



(a) **Task**: promptable segmentation (b) **Model**: Segment Anything Model (**SAM**) (c) **Data**: data engine (top) & dataset (bottom)

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.

1 Introduction

Conv-LoRA

Key 1: 기존 LoRA layer 사이에 lightweight한 convolution block 삽입

Convolution의 고유 특징인 vision-specific inductive bias를 활용 가능

Key 2: MoE(Mixture-of-Experts) 컨셉 구현

여러 experts를 두고, 각 expert가 다른 scale의 image feature를 확인 ->
다양한 scale의 이미지 특징을 추출하므로, 다양한 객체를 탐지 가능

Key 3: SAM의 decoder에 다중 클래스 예측 MLP 추가

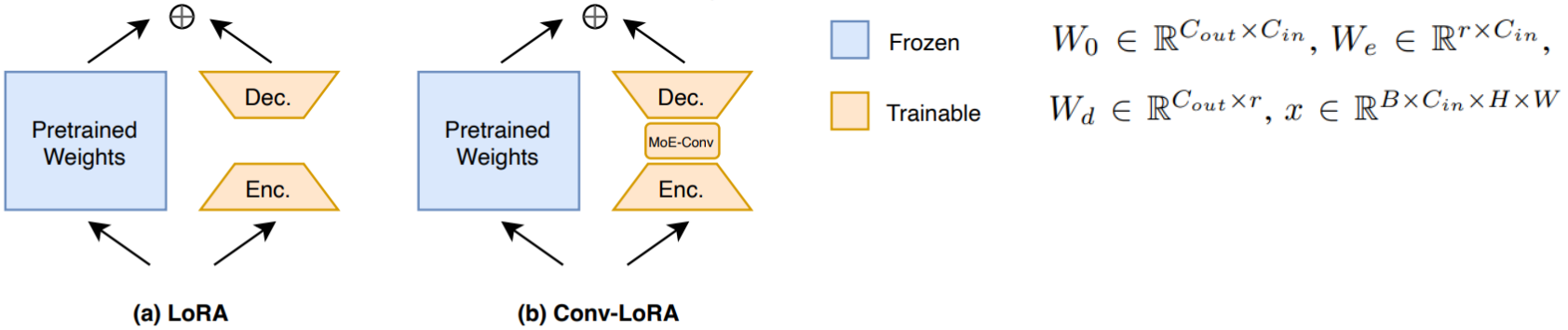
SAM을 통해 Semantic Segmentation을 가능하게 함

2 Method

Conv-LoRA

- LoRA의 중간부에 Convolution operations 삽입
- 단순한 Conv block 대신, MoE-Conv block을 제안

$$h = W_0 x + W_d W_e x \qquad h = W_0 x + W_d(\sum_i^n G(W_e x)_i E_i(W_e x))$$



(a) LoRA (b) Conv-LoRA

Figure 2: LoRA vs. Conv-LoRA. Both LoRA and Conv-LoRA add an extra trainable encoder-decoder structure parallel to the frozen pre-trained weights. Inside the bottleneck of LoRA, Conv-LoRA inserts lightweight convolution operations managed by MoE with negligible extra parameters.

2 Method

MoE-Conv

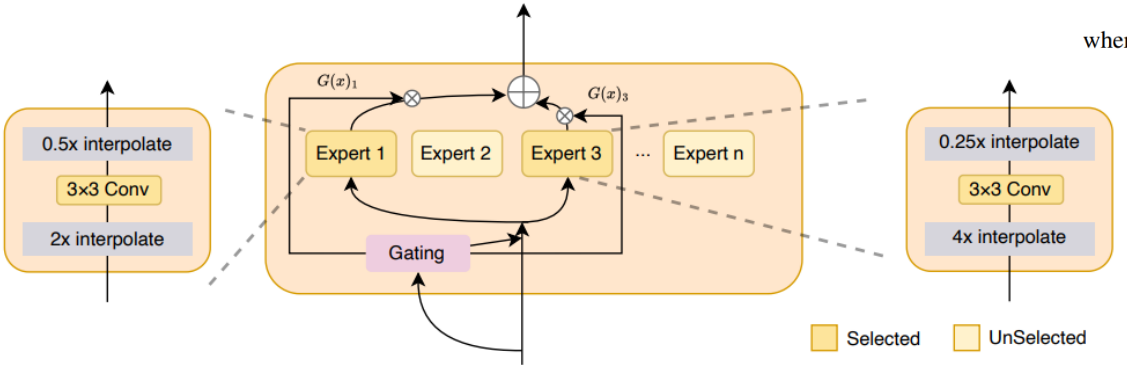
- N개의 expert networks와 gating module로 구성됨 (expert: 다른 스케일의 feature 확인, gating: dynamic하게 expert 선택)
- 다양한 scale의 이미지 특징을 추출하므로, 다양한 크기의 객체를 탐지할 수 있음
- Multi-scale 방식(i.e. Swin)에 비해 computational cost가 적어 효율적임

$$x_h = \text{Reshape}(\text{AvgPool}(x), (B, r))$$
$$H(x)_i = (x_h \cdot W_g)_i + \text{StandardNormal}() \cdot \text{Softplus}((x_h \cdot W_{noise})_i)$$

$$G(x) = \text{Softmax}(\text{KeepTopK}(H(x), k))$$
$$\text{where KeepTopK}(v, k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ -\infty & \text{otherwise.} \end{cases}$$

$$h = W_0 x + W_d(\sum_i^n G(W_e x)_i E_i(W_e x))$$

$$x_h \in \mathbb{R}^{B \times r} \quad W_g \in \mathbb{R}^{r \times n}$$



$$E_i(x) = \text{Interpolate}(\text{Conv}_{3\times3}(\text{Interpolate}(x, s_i)), 1/s_i) \quad (3)$$

For instance, if $s_i = 4$, expert E_i would initially upscale the feature maps by a factor of 4x, apply the $\text{Conv}_{3\times3}$ operation, and finally, downscale the feature maps by 4x.

2 Method

Multi-class Segmentation with SAM

- Mask-based method (i.e. Maskformer)를 따라 N개의 binary masks와 N개의 classification scores를 예측함
- Multi-class 예측을 위해 prompt encoder를 제거 후 lightweight한 MLP를 추가함
- SAM의 구조를 변경하여 end-to-end로 학습이 가능해짐

$$\mathcal{L}_{multi-class} = \lambda_{mask}\mathcal{L}_{mask} + \lambda_{cls}\mathcal{L}_{cls} + \lambda_{MoE}\mathcal{L}_{MoE}$$

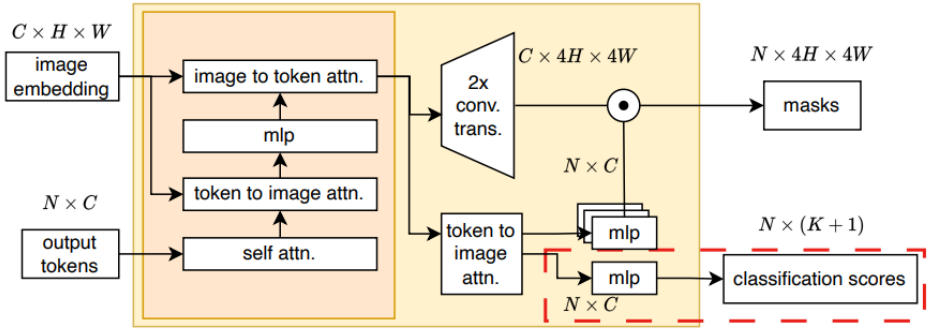


Figure 4: The modified SAM's mask decoder for multi-class semantic segmentation. The classification module (within the red dashed box) is new added compared to original SAM's mask decoder. N is the number of output mask tokens, K is the number of classes, C is the number of channels, H and W indicate the height and width of the feature map (we omit 'batch size' for simplicity).

3 Experiments

Binary class semantic segmentation

- Medical, Natural, Agriculture, Remote sensing 환경에서 실험 진행
- SAM trained from scratch 성능이 가장 안 좋은 이유는 PEFT 방법들과 달리 SAM의 사전 지식을 잃어버렸기 때문
- Vision-specific한 inductive bias가 추가됨으로써 가장 우수한 성능을 보임

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
SAM trained from scratch	641.09 / 100%	78.5	82.4	85.9	91.6	73.8	82.5	61.9	67.0	40.5	5.53	52.1	65.5	55.6	71.1
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 ± 0.46	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	93.7 ± 1.41	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

Table 1: Results on binary semantic segmentation. '# Params (M) / Ratio (%)' represents the number of trainable parameters and its proportion relative to the total number. *Domain Specific* is a placeholder referring to methods that specifically designed for the tasks. 'Underlined' denotes the better results compared to PEFT methods. See appendix D for more details. Compared to LoRA, Conv-LoRA incurs negligible parameter overhead, but delivers a clear performance boost.

3 Experiments

- Multi class semantic segmentation
  - 다양한 클래스를 지닌 multi-class dataset에서도 가장 좋은 성능을 보임
  - Decoder만 학습하면 Encoder가 high-level 정보를 추출하지 못하지만, Encoder를 PEFT로 학습하면 성능이 크게 향상됨

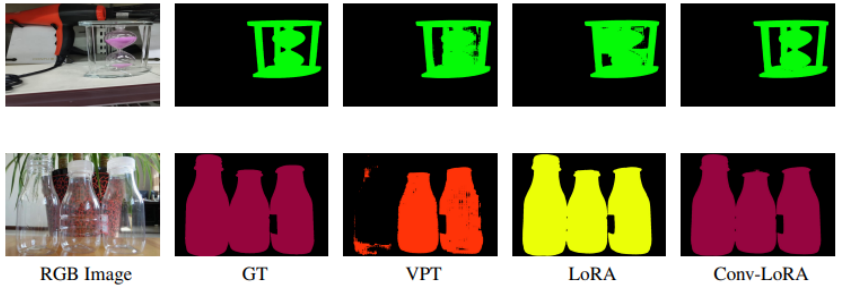


Figure 14: Visualization Results on Multi-class Transparent Object Segmentation.

Method	#Params (M) / Ratio (%)	Easy				Hard			
		Acc ↑	mIoU ↑	MAE ↓	MBER ↓	Acc ↑	mIoU ↑	MAE ↓	MBER ↓
TransLab	42.19 / 100%	95.77	92.23	0.036	3.12	83.04	72.10	0.166	13.30
decoder-only	3.51 / 0.55%	94.68	88.54	0.050	4.24	83.53	68.30	0.186	14.37
VPT	4.00 / 0.62%	98.31	95.73	0.017	1.52	90.42	83.38	0.083	7.21
LoRA	4.00 / 0.62%	98.44	96.26	0.016	1.35	91.94	83.95	0.083	6.35
Conv-LoRA	4.02/ 0.63%	**98.63**	**96.45**	**0.015**	**1.27**	**93.05**	**84.37**	**0.075**	**6.25**

Method	# Params (M) / Ratio (%)	Acc↑	mIoU ↑	Category IoU ↑										
				bg	shelf	jar	freezer	window	door	eyeglass	cup	wall	bowl	bottle	box
TransLab	42.19 / 100%	92.67	69.00	93.90	54.36	64.48	65.14	54.58	57.72	79.85	81.61	72.82	69.63	77.50	56.43
Trans2Seg	56.20 / 100%	94.14	72.15	95.35	53.43	67.82	64.20	59.64	60.56	88.52	86.67	75.99	73.98	82.43	57.17
decoder-only	3.51 / 0.55%	90.66	49.97	93.66	32.75	39.96	35.87	50.70	45.89	57.38	73.16	69.36	54.23	56.58	33.77
VPT	4.00 / 0.62%	94.42	62.81	97.41	29.76	52.82	62.09	55.54	63.61	81.12	83.40	79.61	65.29	72.92	44.77
LoRA	4.00 / 0.62%	94.80	66.01	97.50	42.17	57.82	**64.35**	53.44	64.08	**87.28**	**85.28**	80.43	63.67	77.97	49.56
Conv-LoRA	4.02 / 0.63%	**95.07**	**67.09**	**97.66**	**50.51**	**58.44**	51.70	**55.69**	**65.22**	85.23	84.84	**80.97**	**72.84**	**79.83**	**52.73**

Table 2: Results on multi-class semantic segmentation. The tables are the results for three-class and twelve-class semantic segmentation respectively.

3 Experiments

Mean attention distance

- MAE로 학습된 ViT는 후반부에서 Mean Attention Distance가 길게 나타남 (후반부로 갈수록 fine-grained information이 부족함)
- ConvLoRA로 학습된 SAM의 Encoder는 후반부에서 Mean Attention Distance가 골고루 나타남 (fine-grained detail이 풍부하여 segmentation에 탁월함)

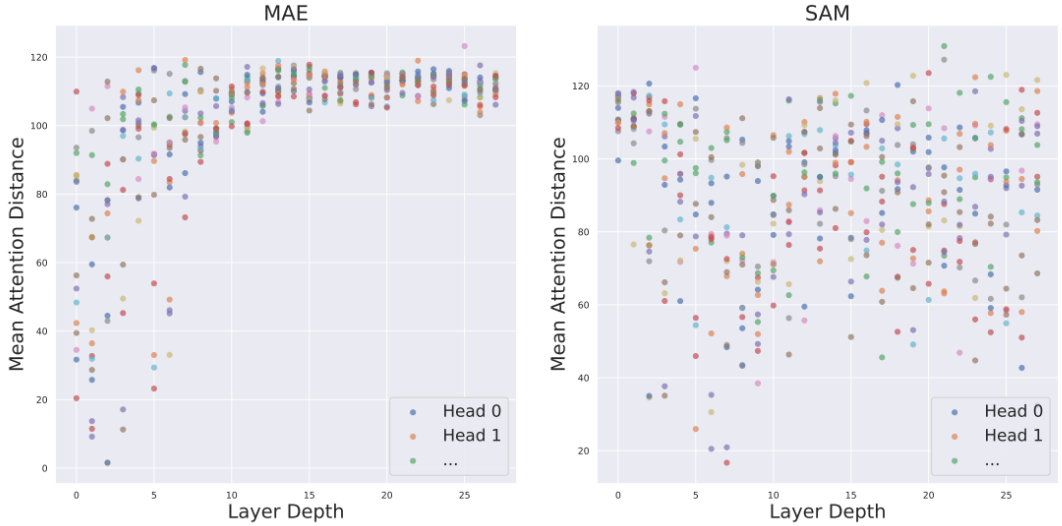


Figure 5: Mean attention distance of each attention head, with each dot indicating the mean distance across images for one of the 16 heads at one layer. In contrast to MAE, SAM retains the ability to incorporate local information even in deeper layers.

3 Experiments

Further Analysis

Method	ISIC 2017			
	Jac ↑	T-Jac ↑	Dice ↑	Acc ↑
Multi-scale	77.4	69.5	85.4	93.7
MoE	**77.9**	**70.3**	**85.9**	**93.9**

Method	Training Speed (Iter / s) ↑	Training Memory (GB) ↓
Multi-scale	0.79	23.4
MoE	**1.22**	**21.7**

Table 3: MoE vs. Multi-scale, the latter fuses features from all scales simultaneously. The performance and the training cost illustrate the effectiveness and efficiency of dynamic selecting the experts, i.e., injecting the local prior into the feature maps of different scales dynamically.

LoRA	#Params (M)	Test	
		Acc ↑	mIoU ↑
$r = 3$	4.00	94.80	66.01
$r = 6$	4.49	95.15	66.24
$r = 12$	5.48	95.23	66.69
$r = 24$	7.44	**95.14**	**67.02**

Table 11: The performance trend when increasing the rank r of LoRA for twelve-class segmentation.

Method	# Params / Ratio(%)	CVC-ColonDB				ETIS			
		S_α ↑	E_ϕ ↑	F_β^ω ↑	MAE ↓	S_α ↑	E_ϕ ↑	F_β^ω ↑	MAE ↓
VPT	4.00 / 0.62%	85.0	88.4	75.0	3.6	86.5	87.4	70.1	1.9
Conv-LoRA	4.03 / 0.63%	85.1	88.8	75.7	**3.4**	86.8	**88.5**	70.2	**1.7**
Conv-LoRA+VPT	4.23 / 0.66 %	**86.0**	**89.0**	**76.8**	3.5	**87.7**	88.1	**70.3**	**1.7**

Table 12: Performance of combining Conv-LoRA and VPT on Polyp Segmentation.

Method	# Params / Ratio(%)	Leaf			ISIC 2017			
		IoU ↑	Dice ↑	Acc ↑	Jac ↑	T-Jac ↑	Dice ↑	Acc ↑
Inception	4.01 / 0.63%	72.1	82.3	95.0	76.9	68.5	84.9	93.3
MoE-Conv	4.01 / 0.63%	**74.0**	**83.8**	**95.6**	**77.4**	**69.5**	**85.5**	**93.9**

Table 13: MoE-Conv vs. Blocks with Various Kernel sizes (indicated as 'Inception').

Thank You