Vision AI

2021 arXiv Trends

2021-07

hutom

no.	Paper Title	Correspondence	h-index
1	Swin Transformer: Hierarchical Vision Transformer using Shifted Windows	Han Hu	20
2	Patch AutoAugment	Zhibo Chen	25
3	How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers	Lucas Beyer	15
4	When Vision Transformers Outperform ResNets without Pretraining or Strong Data Augmentations	Boqing Gong	34
5	You only look at one sequence: rethinking transformer in vision through object detection	Wenyu Liu	53

no.	Paper Title	Correspondence	h-index
6	EfficientNetV2: Smaller Models and Faster Training	Mingxing Tan	17
7	Focal Self-attention for Local-Global Interactions in Vision Transformers	Jianfeng Gao	80
8	Attention Bottlenecks for Multimodal Fusion	Chen Sun	30
9	On the Practicality of Deterministic Epistemic Uncertainty	Federico Tombari	44

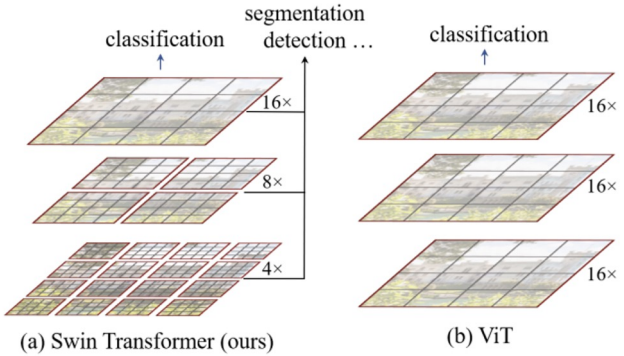
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

https://arxiv.org/pdf/2103.14030.pdf

[Submitted on 25 Mar 2021]

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

Ze Liu[†*] Yutong Lin[†*] Yue Cao[*] Han Hu[*‡] Yixuan Wei[†]
Zheng Zhang Stephen Lin Baining Guo
Microsoft Research Asia
{v-zeliu1,v-yutlin,yuecao,hanhu,v-yixwe,zhez,stevelin,bainguo}@microsoft.com



- Computer Vision Transformer의 General-proposed
- Transformer를 Language에서 Vision영역에 adapt 하는 어려움은 Visual Entities의 scale Variation에 기인
- 이에, Shifted Windows 로 계산되는 Hierarchical Transformer를 도입
- Shifted Windows를 도입함으로써 Self-Attention의 계산을 Non-overlapping local windows로 Limit 하고 Cross-window Connection을 Allow (기존 ViT의 Quadratic한 연산을 Linear하게 줄임)
- 해당 계층구조를 통해
  1. Scale에 대해 Flexibility를 갖고
  2. Image Size에 대해 Linear Computational Complexity 를 갖게 됨.
- Swim Transformer 특성 덕분에
  - Image Classification (86.4 top-1 accuracy)
  - dense Prediction task (58.7 box AP, 51.1 mAP on COCO test-dev) * image의 each pixel에 대한 label을 prediction task
  - Semantic Segmentation (53.5 mIoU on ADE20k val) 를 비롯한 여러 Vision task에 Compatible 하게 사용

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- **Architecture**

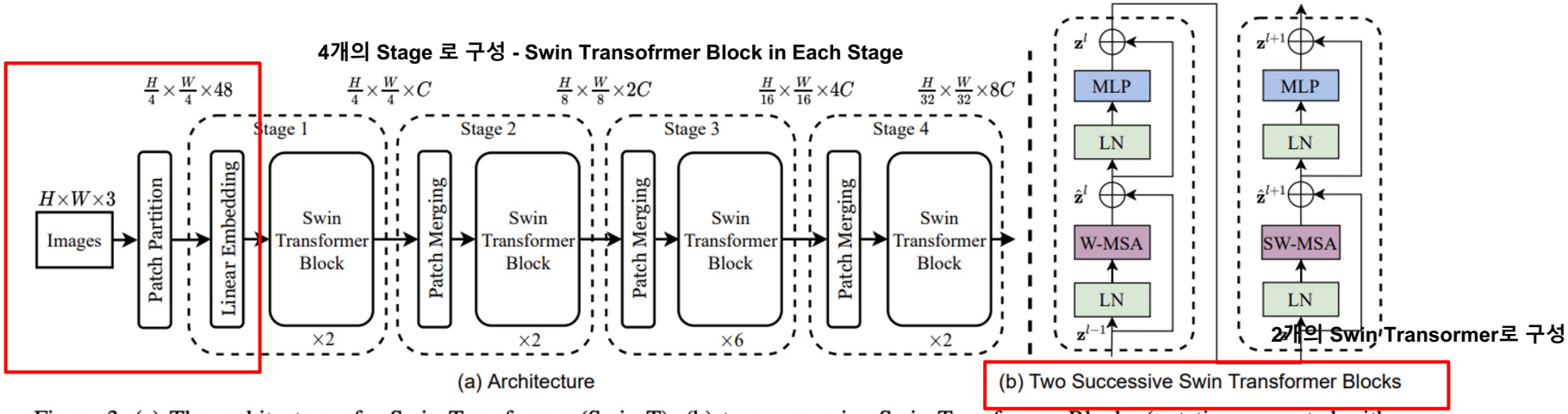
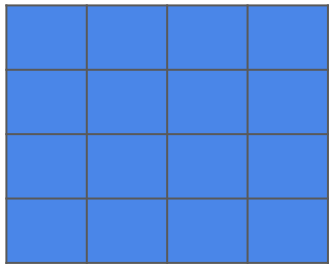
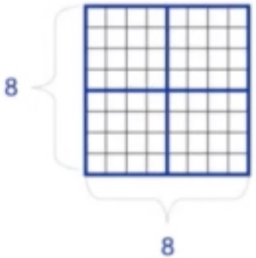


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.



Dimension = 48
(4x4x3 channles)

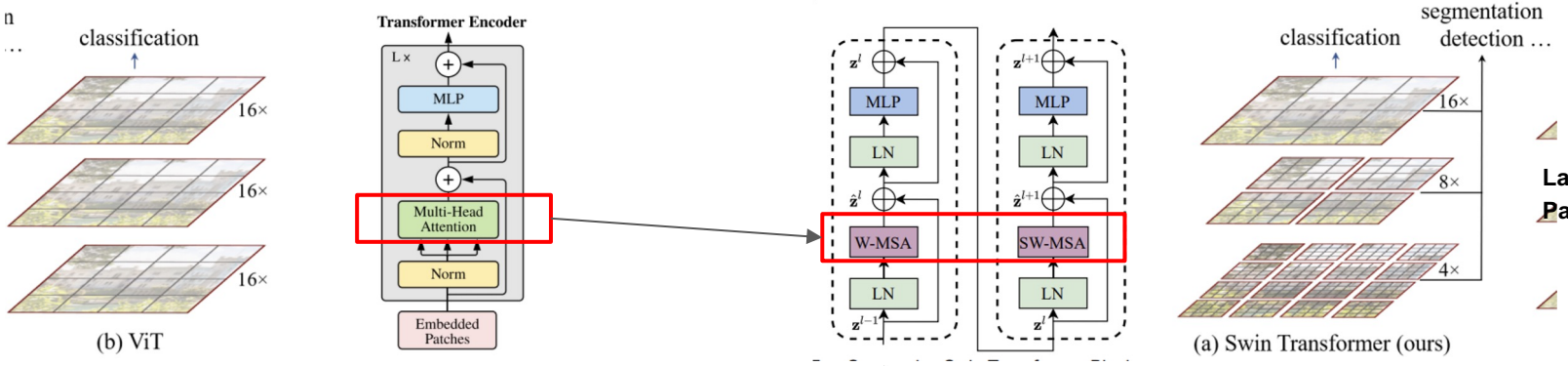


At first, like most computer vision tasks, an RGB image is sent to the network.

This Image is split into Patches, and each patch is treated as a token

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- 핵심 아이디어 - W-MSA, SW-MSA



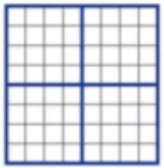
Layer가 깊어질수록 Patch 들이 Merge

$$\Omega(\text{MSA}) = 4hwC^2 + 2(hw)^2C,$$

$$\Omega(\text{W-MSA}) = 4hwC^2 + 2M^2hwC,$$

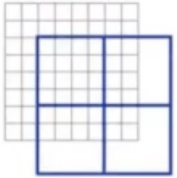
기존 해상도의 Quadratic한 Computation인 반면

선형적인 Computation Complexity



W-MSA

The input image is divided into four windows and compute attention for patches **within the window.**



SW-MSA

The approach **introduces connections** between neighboring non-overlapping windows in the previous layer.

$$\hat{\mathbf{z}}^l = \text{W-MSA}\left(\text{LN}\left(\mathbf{z}^{l-1}\right)\right) + \mathbf{z}^{l-1},$$
$$\mathbf{z}^l = \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^l\right)\right) + \hat{\mathbf{z}}^l,$$
$$\hat{\mathbf{z}}^{l+1} = \text{SW-MSA}\left(\text{LN}\left(\mathbf{z}^l\right)\right) + \mathbf{z}^l,$$
$$\mathbf{z}^{l+1} = \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^{l+1}\right)\right) + \hat{\mathbf{z}}^{l+1},$$

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- **핵심 아이디어 : Cyclic-Shift**

The paper propose an efficient batch computation approach by **cyclic-shifting** toward the top-left direction

After, this shift, a batched window may be composed of several sub-windows that are not adjacent in the feature map, so a **masking mechanism** is employed to limit self-attention computation to within each sub-windows

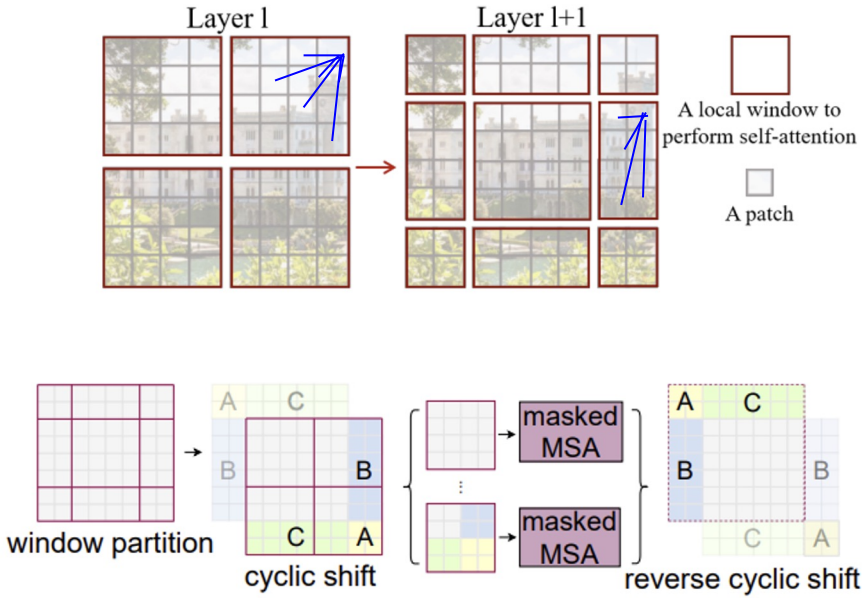


Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

- Swin-T: $C = 96$, layer numbers = $\{2, 2, 6, 2\}$
- Swin-S: $C = 96$, layer numbers =$\{2, 2, 18, 2\}$
- Swin-B: $C = 128$, layer numbers =$\{2, 2, 18, 2\}$
- Swin-L: $C = 192$, layer numbers =$\{2, 2, 18, 2\}$

Architecture Variants

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- **Experiments & Results**

Classification Results
(EfficientNet 이 더 괜찮아보이는데 구지..?)

(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [47]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [47]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [47]	224^2	84M	16.0G	334.7	82.9
EffNet-B3 [57]	300^2	12M	1.8G	732.1	81.6
EffNet-B4 [57]	380^2	19M	4.2G	349.4	82.9
EffNet-B5 [57]	456^2	30M	9.9G	169.1	83.6
EffNet-B6 [57]	528^2	43M	19.0G	96.9	84.0
EffNet-B7 [57]	600^2	66M	37.0G	55.1	84.3
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 [60]	224^2	22M	4.6G	940.4	79.8
DeiT-B [60]	224^2	86M	17.5G	292.3	81.8
DeiT-B [60]	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	83.3
Swin-B	384^2	88M	47.0G	84.7	84.2

(b) ImageNet-22K pre-trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [37]	384^2	388M	204.6G	-	84.4
R-152x4 [37]	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	86.0
Swin-L	384^2	197M	103.9G	42.1	86.4

Object Detection Results

(a) Various frameworks

Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
Cascade Mask R-CNN	R-50	46.3	64.3	50.5	82M	739G	18.0
	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^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	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 AP^{box}	mini-val AP^{mask}	test-dev AP^{box}	test-dev AP^{mask}	#param.	FLOPs
RepPointsV2* [11]	-	-	52.1	-	-	-
GCNet* [6]	51.8	44.7	52.3	45.4	-	1041G
RelationNet++* [12]	-	-	52.7	-	-	-
SpineNet-190 [20]	52.6	-	52.8	-	164M	1885G
ResNeSt-200* [75]	52.5	-	53.3	47.1	-	-
EfficientDet-D7 [58]	54.4	-	55.1	-	77M	410G
DetectoRS* [45]	-	-	55.7	48.5	-	-
YOLOv4 P7* [3]	-	-	55.8	-	-	-
Copy-paste [25]	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 decovolution layers are used to produce hierarchical feature maps. * indicates multi-scale testing.

Semantic Segmentation Results

Method	Backbone	val mIoU	test score	#param.	FLOPs	FPS
DANet [22]	ResNet-101	45.2	-	69M	1119G	15.2
DLab.v3+ [10]	ResNet-101	44.1	-	63M	1021G	16.0
ACNet [23]	ResNet-101	45.9	38.5	-		
DNL [68]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [70]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [66]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [70]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [10]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [10]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [78]	T-Large‡	50.3	61.7	308M	-	-
UperNet	DeiT-S†	44.0	-	52M	1099G	16.2
UperNet	Swin-T	46.1	-	60M	945G	18.5
UperNet	Swin-S	49.3	-	81M	1038G	15.2
UperNet	Swin-B‡	51.6	-	121M	1841G	8.7
UperNet	Swin-L‡	**53.5**	**62.8**	234M	3230G	6.2

Table 3. Results of semantic segmentation on the ADE20K val and test set. † indicates additional deconvolution layers are used to produce hierarchical feature maps. ‡ indicates that the model is pre-trained on ImageNet-22K.

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- **Ablation & Conclusion**

Ablation Study

shift 기법을 사용했을 때와 없을 때 비교한 결과,
shift 기법이 좋은 더 좋은 성능

	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).

Ablation Study

Padding과 Cyclic 을 적용했을 때 Cyclic이 훨씬 빠른 처리 속도

method	MSA in a stage (ms)				Arch. (FPS)		
	S1	S2	S3	S4	T	S	B
sliding window (naive)	122.5	38.3	12.1	7.6	183	109	77
sliding window (kernel)	7.6	4.7	2.7	1.8	488	283	187
Performer [13]	4.8	2.8	1.8	1.5	638	370	241
window (w/o shifting)	2.8	1.7	1.2	0.9	770	444	280
shifted window (padding)	3.3	2.3	1.9	2.2	670	371	236
shifted window (cyclic)	3.0	1.9	1.3	1.0	755	437	278

Table 5. Real speed of different self-attention computation methods and implementations on a V100 GPU.

- **Conclusion**
  - This paper presents Swin Trasnformer, a new vision Transformer which Produces a hireachical feature representations and has **linear computational complexity** with respect to input image size.
  - As a key elements of Swin transformer, the **shifted window based self-attention** is shown to be effiective and efficient on vision problems.
  - Using a similar architecture for both NLP and computer vision could significantly accelerate the research process.

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- **후속연구 - Top Recent 21/6/24 Paper**

https://arxiv.org/pdf/2106.13230v1.pdf

[Submitted on 24 Jun 2021]

Video Swin Transformer

Ze Liu[*12], **Jia Ning**[*13], **Yue Cao**[1†], **Yixuan Wei**[14], **Zheng Zhang**[1], **Stephen Lin**[1], **Han Hu**[1†]
[1]Microsoft Research Asia
[2]University of Science and Technology of China
[3]Huazhong University of Science and Technology
[4]Tsinghua University

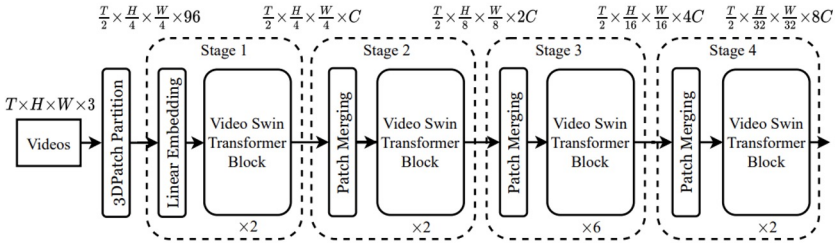
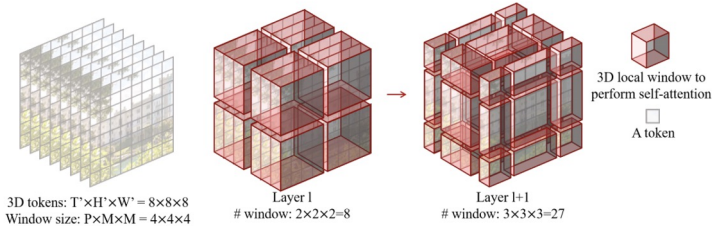


Figure 1: Overall architecture of Video Swin Transformer (tiny version, referred to as Swin-T).



Patch AutoAugment

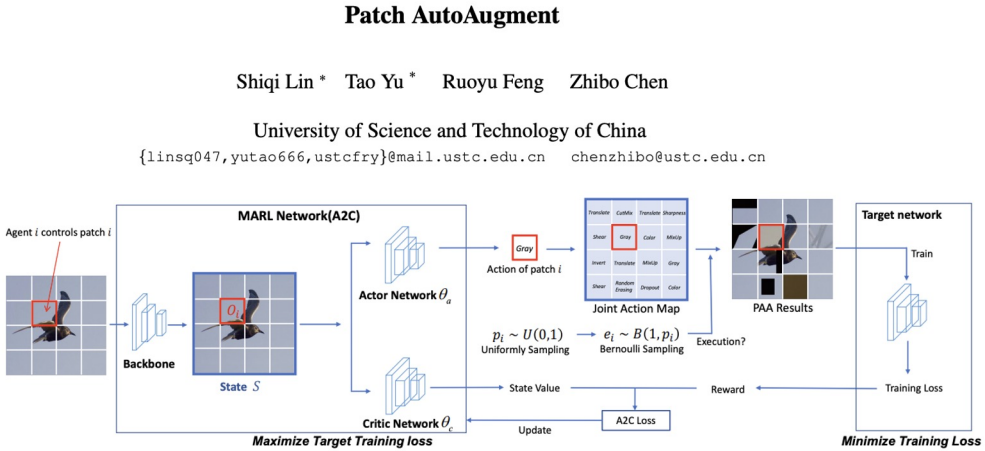


Figure 2. The framework of our PAA. PAA divides the input image into a grid of patches. Each patch augmentation is controlled by an agent. PAA uses Multi-Agent Reinforcement Learning (MARL) to search for the joint optimal policy of patches according to the patch content and the whole image semantics. PAA is co-trained with a target network through adversarial training.

https://arxiv.org/pdf/2103.11099.pdf

Figure 1. Illustration of different automatic augmentation policy. (a) Original images; (b) Results of image-level AutoAugment such as AA [7] and AdvAA [59]; (c) Our patch-level AutoAugment results.

- Data Augmentation (DA)는 Overfit을 줄이고 모델의 일반화를 개선하기 위해 중요한 역할
- 대부분의 Auto DA방법은 이미지의 각 영역에 대한 최적의 정책이 다양할 수 있다는 점을 간과하고 Image-Level 에서 DA정책을 검색.
- 본 논문은 PPA (Patch AutoAugment)라는 Patch-level 자동 Augmentation 알고리즘을 제안.
- PPA는 이미지를 패치 그리드로 나누어 각 패치의 최적의 DA정책을 검색.
- 특히, 각 DA 작업을 Agent에서 제어할 수 있는 MARL (Multi-Agent Reinforcement Learning) 문제로 모델링
- Agent가 팀 단위로 협력하고 전체 이미지의 joint optimal DA Policy를 달성한 데에 대한 team Reward를 공유하는 방식으로 작동
- Image Classification, fine-grained image recognition 의 벤치마크 데이터 셋에서 목표 Network의 성능을 일관되게 향상시킴

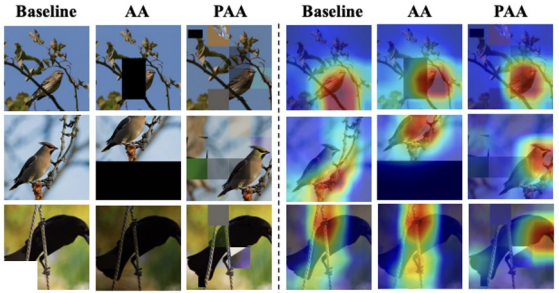
- **Experiments & Results**

Dataset	Model	Baseline	CutOut [11]	AA [7]	FastAA [31]	PAA
CIFAR-10	Wide-ResNet-28-10	96.13	96.92	97.32	97.21	**97.43**±0.03
	ShakeShake(26 2x32d)	96.26	96.54	96.68	96.60	**96.86**±0.04
	ShakeShake(26 2x96d)	97.06	97.44	97.68	97.69	**97.74**±0.02
	ShakeShake(26 2x112d)	97.01	97.43	**97.78**	97.74	97.62±0.05
	PyramidNet+ShakeDrop	97.24	97.69	98.07	98.15	**98.23**±0.03
CIFAR-100	Wide-ResNet-28-10	81.20	82.52	82.72	82.52	**82.74**±0.01
	ShakeShake(26 2x96d)	82.35	83.01	84.06	84.02	**84.11**±0.02
	PyramidNet+ShakeDrop	83.77	85.57	**86.02**	85.00	85.80±0.04

Table 1. The validation top-1 accuracy (%) on CIFAR-10 [28] and CIFAR-100 [28] with different methods. All models are trained from scratch. The results of AdvAA [59] are not reported due to unavailability of official source.

Dataset	GPU hours	AA [7]	FastAA [31]	AdvAA [59]	PAA
CIFAR-10	Search	5000	3.5	**∼0**	**∼0**
	Train	**6**	**6**	-	7.5
	Total	5006	9.5	-	**7.5**
ImageNet	Search	15000	450	**∼0**	**∼0**
	Train	**160**	**160**	1280	270
	Total	15160	610	1280	**270**

Table 2. We train Wide-ResNet-28-10 on CIFAR-10 [28] and ResNet-50 on ImageNet [10]. *Search*: the time of searching for augmentation policys. *Train*: the time of training the target network. *Total*: the total time of searching for augmentation policy and training the target network. The searching time of AdvAA [59] and our PAA is close to zero. The computational cost of PAA is estimated on GeForce GTX 1080 Tis while AA [7], AdvAA [59] are on NVIDIA Tesla P100 and FastAA [31] is on NVIDIA Tesla V100.



e 3. The visualization of features learned by ResNet-50 d with baseline, AA [7] and PAA using Grad-Cam [43]. PAA to help the target network focus on the discriminative areas.

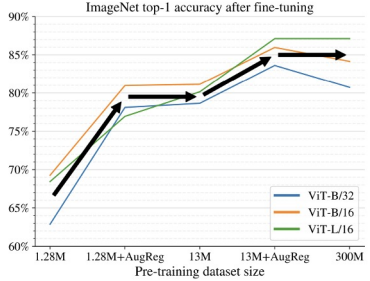
How to train your ViT? Data, Augmentation, and Regularization in Vision Transoformers

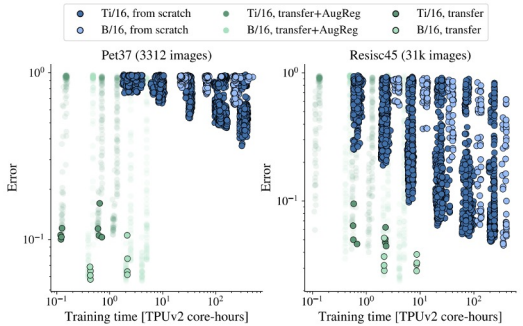
[Submitted on 18 Jun 2021]

https://arxiv.org/pdf/2106.10270.pdf

How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers

Andreas Steiner*, Alexander Kolesnikov*, Xiaohua Zhai*
Ross Wightman†, Jakob Uszkoreit, Lucas Beyer*
Google Research, Brain Team; †independent researcher
{andstein,akolesnikov,xzhai,usz,lbeyer}@google.com, rwightman@gmail.com





- ViT는 Object detection, semantic image segmentation 등 많은 비전 어플리케이션에서 경쟁력 있는 성능을 제공
- CNN과 비교하면 소규모 데이터셋에서

  Transformer's weaker inductive bias는 model의 regularization or data augmentation 에 대한 ("AugReg") 의존도가 증가하는 것으로 확인됨.
- 이에 Training Data 양, AugReg, model의 크기 및 compute budget 사이의 상호작용을 더 잘 이해하기 위해 연구 수행.
- 해당 연구 결과, 컴퓨팅 증가와 "AugReg"의 조합은 보다 큰 규모의 Training Data에 대해 training된 모델과 동일한 성능을 가진 모델을 생산할 수 있음을 알게 됨.
- [illegible] 모델을 공유. 더 큰 규모의 JFT-300M dataset에 대해서도 train. 비슷하거나 outperform

[1]We release more than 50'000 ViT models trained under diverse settings on various datasets. We believe this to be a treasure trove for model analysis. Available at https://github.com/google-research/vision_transformer and https://github.com/rwightman/pytorch-image-models.

How to train your ViT? Data, Augmentation, and Regularization in Vision Transoformers

https://arxiv.org/pdf/2106.10270.pdf

Table 1: Configurations of ViT models.

Model	Layers	Width	MLP	Heads	Params
ViT-Ti [34]	12	192	768	3	5.8M
ViT-S [34]	12	384	1536	6	22.2M
ViT-B [10]	12	768	3072	12	86M
ViT-L [10]	24	1024	4096	16	307M

Table 2: ResNet+ViT hybrid models.

Model	Resblocks	Patch-size	Params
R+Ti/16	[]	8	6.4M
R26+S/32	[2, 2, 2, 2]	1	36.6M
R50+L/32	[3, 4, 6, 3]	1	330.0M

3.3 Regularization and data augmentations

To regularize our models we use robust regularization techniques widely adopted in the computer vision community. We apply dropout to intermediate activations of ViT as in [10]. Moreover, we use the stochastic depth regularization technique [16] with linearly increasing probability of dropping layers.

For data augmentation, we rely on the combination of two recent techniques, namely Mixup [41] and RandAugment [6]. For Mixup, we vary its parameter α, where 0 corresponds to no Mixup. For RandAugment, we vary the magnitude parameter m, and the number of augmentation layers l.

We also try two values for weight decay [23] which we found to work well, since increasing AugReg may need a decrease in weight decay [2].

Overall, our sweep contains 28 configurations, which is a cross-product of the following hyper-parameter choices:

- Either use no dropout and no stochastic depth (e.g. no regularization) or use dropout with probability 0.1 and stochastic depth with maximal layer dropping probability of 0.1, thus 2 configuration in total.
- 7 data augmentation setups for (l, m, α): *none* $(0, 0, 0)$, *light1* $(2, 0, 0)$, *light2* $(2, 10, 0.2)$, *medium1* $(2, 15, 0.2)$, *medium2* $(2, 15, 0.5)$, *strong1* $(2, 20, 0.5)$, *strong2* $(2, 20, 0.8)$.
- Weight decay: 0.1 or 0.03.

3.4 Pre-training

We pre-trained the models with Adam [17], using $\beta_1 = 0.9$ and $\beta_2 = 0.999$, with a batch size of 4096, and a cosine learning rate schedule with a linear warmup (10k steps). To stabilize training, gradients were clipped at global norm 1. The images are pre-processed by Inception-style cropping [31] and random horizontal flipping. On the smaller ImageNet-1k dataset we trained for 300 epochs, and for 30 and 300 epochs on the ImageNet-21k dataset. Since ImageNet-21k is about 10x larger than ImageNet-1k, this allows us to examine the effects of the increased dataset size also with a roughly constant total compute used for pre-training.

3.5 Fine-tuning

We fine-tune with SGD with a momentum of 0.9 (storing internal state as `bfloat16`), sweeping over 2-3 learning rates and 1-2 training durations per dataset as detailed in Table 4 in the appendix. We used a fixed batch size of 512, gradient clipping at global norm 1 and a cosine decay learning rate schedule with linear warmup. Fine-tuning was done both at the original resolution (224), as well as at a higher resolution (384) as described in [35].

Table 4: Finetune details for the pre-trained models.

Dataset	Learning rate	Total steps and warm up steps
ImageNet-1k	$\{0.01, 0.03\}$	$\{(20\,000, 500)\}$
Pets37	$\{0.001, 0.003, 0.01\}$	$\{(500, 100), (2500, 200)\}$
Kitti-distance	$\{0.001, 0.003, 0.01\}$	$\{(500, 100), (2500, 200)\}$
CIFAR-100	$\{0.001, 0.003, 0.01\}$	$\{(2500, 200), (10\,000, 500)\}$
Resisc45	$\{0.001, 0.003, 0.01\}$	$\{(2500, 200), (10\,000, 500)\}$

How to train your ViT? Data, Augmentation, and Regularization in Vision Transoformers

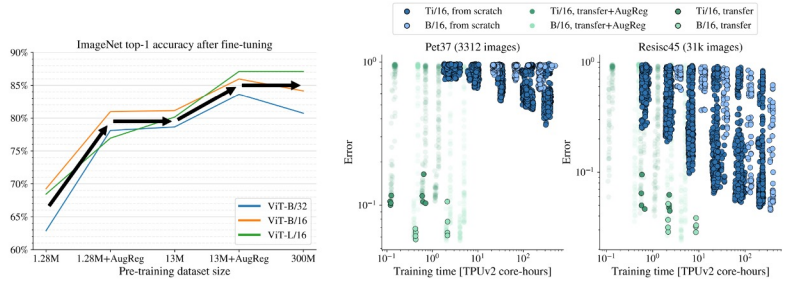


Figure 1: **Left**: Adding the right amount of regularization and image augmentation can lead to similar gains as increasing the dataset size by an order of magnitude. **Right**: For small and mid-sized datasets it is very hard to achieve a test error that can trivially be attained by fine-tuning a model pre-trained on a large dataset like ImageNet-21k – see also Figure 2. With our recommended models (Section 4.5), one can find a good solution with very few trials (bordered green dots). Note that AugReg is not helpful when transferring pre-trained models (borderless green dots).

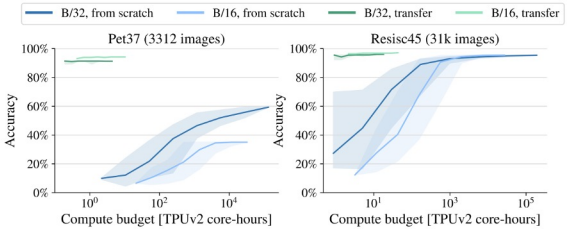


Figure 2: Fine-tuning recommended checkpoints leads to better performance with less compute (green), compared to training from scratch with AugReg (blue) – see Figure 1 (right) for individual runs. For a given compute budget (x-axis), choosing random configurations within that budget leads to varying final performance, depending on choice of hyper parameters (shaded area covers 90% from 1000 random samples, line corresponds to median).

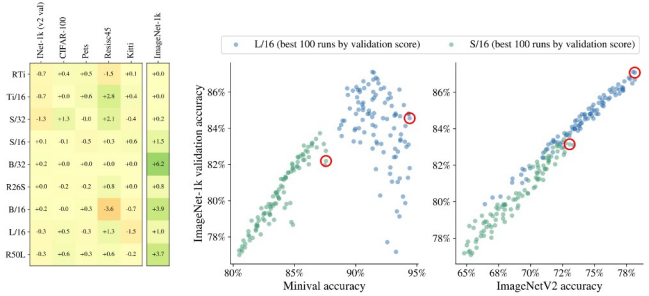


Figure 5: Choosing best models. **Left**: Difference of fine-tuning test scores between models chosen by best validation score on pre-training data vs. validation score on fine-tuning data (negative values mean that selecting models by pre-training validation deteriorates fine-tuning test metrics). **Right**: Correlation between "minival" validation score vs. ImageNetV2 validation score and official ImageNet-1k validation score (that serves as a test score in this study). The red circles correspond to best models by validation score - see Section 4.5 for an explanation.

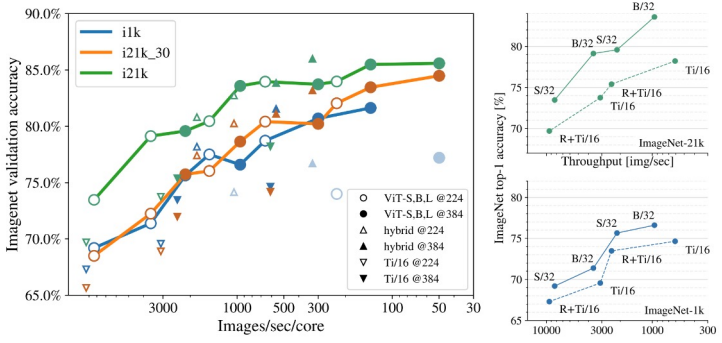


Figure 6: Imagenet transfer. **Left**: For every architecture and upstream dataset, we selected the best model by upstream validation accuracy. Main ViT-S,B,L models are connected with a solid line to highlight the trend, with the exception of ViT-L models pre-trained on i1k, where the trend breaks down. The same data is also shown in Table 3. **Right**: Focusing on small models, it is evident that using a larger patch-size (/32) significantly outperforms making the model thinner (Ti).

How to train your ViT? Data, Augmentation, and Regularization in Vision Transoformers

- **Conclusion**

We conclude that across a wide range of datasets,
even if the downstream data of interest appears to only be weakly related to the data used for pre-training,
transfer learning remains the best available option.

Our analysis also suggests that among similarly performing pre-trained models,
for transfer learning a model with **more training data should likely be preferred over one with more data augmentation.**

We hope that our study will help guide future research on Vision Transformers and will be a useful source of effective training settings for practitioners seeking to optimize their final model performance in the light of a given computational budget.

When Vision Transformers Outperform ResNets without Pertraining or Strong Data Augmentations

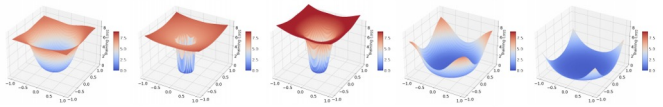
[Submitted on 3 Jun 2021]

https://arxiv.org/pdf/2106.01548.pdf

When Vision Transformers Outperform ResNets without Pretraining or Strong Data Augmentations

Xiangning Chen[1,2*] Cho-Jui Hsieh[2] Boqing Gong[1]

[1]Google Research [2]UCLA



(a) ResNet (b) ViT (c) Mixer (d) ViT-SAM (e) Mixer-SAM

Figure 1: Cross-entropy loss landscapes of ResNet-152, ViT-B/16, and Mixer-B/16. ViT and MLP-Mixer converge to sharper regions than ResNet when trained on ImageNet with the basic Inception-style preprocessing. SAM, a sharpness-aware optimizer, significantly smooths the landscapes.

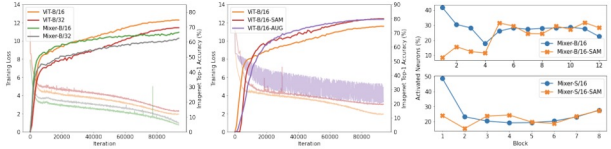


Figure 2: **Left** and **Middle**: ImageNet training error and validation accuracy vs. iteration for ViTs and MLP-Mixers with different patch sizes, vanilla SGD or SAM, and basic pre-processing vs. repeated augmentations. **Right**: Percentage of activated neurons at each block of MLP-Mixers.

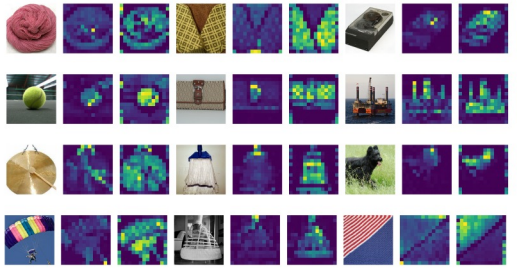


Figure 3: Raw images (**Left**) and attention maps of ViT-S/16 with (**Right**) and without (**Middle**) sharpness-aware optimization. ViT-S/16 with less sharp local optimum contains perceptive segmentation information in its attention maps.

- CNN을 사용하지 않고 Vision Transformer만을 활용해도 CV Task 수행 Good.
- 기존 ViT는 많은 양의 데이터 (Large-Scale Pretraining) 또는
  Repeated Strong data Augmentations 에서 이뤄짐.
- 해당 연구에서 Vision Transformer의 geometry 를 줄이면서, model generalization 성능을 향상시키고자
- Resnet에 비해 ViT, Mixer는 Visualization과 Hessian을 통해 확인한 결과,
  Extremely sharp local minima에 수렴하는 것을 확인할 수 있음
- SAM (Sharpness-aware optimizer) + Smothness Promoting 을 통해
  ViT와 MLP-Mixers의 Accuray 와 Robustness 를 향상.
- Smothness를 추가하였을 때 보통 첫번째, 두번째 Block에서 뉴런이 가장 많이 활성화 되는 것을 알 수 있음.
- 위 2가지 도구를 이용해
  ViT를 Large-scale pretraining 없이, strong data augmentation 없이
  비슷한 사이즈의 Resnet보다 성능이 뛰어남.

You Only Look at one Sequence : Rethinking Transformers in Vision Through Object Detection

[Submitted on 1 Jun 2021 (v1), last revised 21 Jun 2021 (this version, v2)]

https://arxiv.org/pdf/2106.00666.pdf

You Only Look at One Sequence: Rethinking Transformer in Vision through Object Detection

Yuxin Fang [1*] Bencheng Liao [1*] Xinggang Wang [1 ✉] Jiemin Fang [2,1]
Jiyang Qi [1] Rui Wu [3] Jianwei Niu [3] Wenyu Liu [1]

[1] School of EIC, Huazhong University of Science & Technology
[2] Institute of Artificial Intelligence, Huazhong University of Science & Technology
[3] Horizon Robotics
{yxf, bcliao, xgwang}@hust.edu.cn

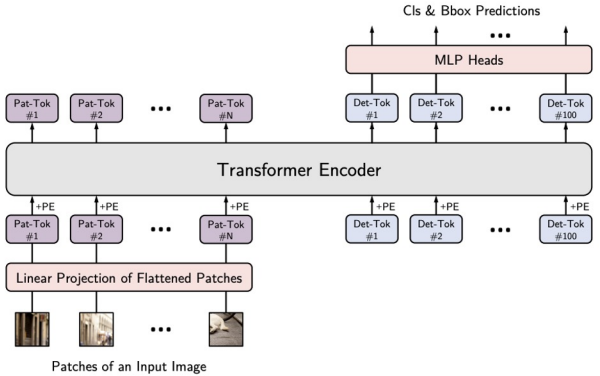


Figure 1: YOLOS overview. "Pat-Tok" refers to patch token, which is the embedding of a flattened image patch. "Det-Tok" refers to [DET] token, which is a learnable embedding for object detection predictions. "PE" refers to positional embedding. During training, YOLOS produces an optimal bipartite matching between predictions from one hundred [DET] tokens and ground truth objects. During inference, YOLOS directly outputs the final set of predictions in parallel. The figure style is inspired by Dosovitskiy et al. [20].

- **Object Detection Task의 새로운 Transformer 구조제안 논문**
- **Transformer**
  1) **2D Spatial Structure에 관한 최소한의 정보 (Minimal Knowledge)**
  2) **Pure Sequence to Sequence 관점에서**
  3) **2D Object-level recognition을 잘 수행할 수 있을까?**
- **해당 질문에 답하기 위해 YOLOS (You Only Look at One Sequence) 제안**
- **naive Vision Transformer 기반의 Inductive Biases와 Modifications을 최소한으로 한 Object Detection models.**
- **ViT와 비교했을 때 YOLOS의 특징**
  1) **YOLOS는 Image classification을 위해서, CLS Token을 Drop하고, DET Token (백개의 초기화된 Detection Tokens)를 추가함**
  2) **Training 에서 YOLOS는 기존의 ViT Image Classification loss를 bipartite matching loss로 대체함.**

EfficientNetV2: Smaller Models and Faster Training

Mingxing Tan [1] Quoc V. Le [1]

https://arxiv.org/pdf/2104.00298.pdf

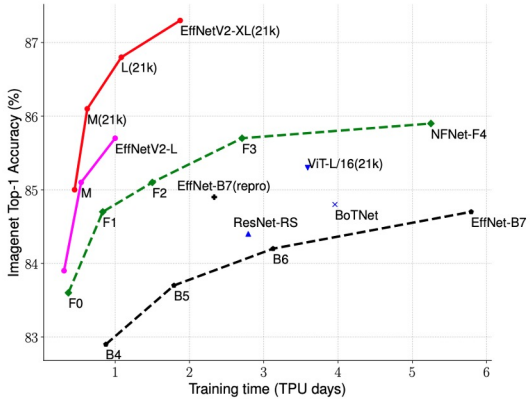
- **Google Research, Brain Team**

- **EfficientNetV2: 기존의 EfficientNet 과 비교해서 학습 속도와 정확도 개선**
  - 작은 수의 파라미터로 동일한 수준의 성능
  - 모델의 학습 속도 또한 빠름
  - 4 stage 로 학습 단계 구분

- **기존 EfficientNet 한계점 (학습 속도 저하 요소)**
  - 큰 이미지로 학습을 하면 학습 속도가 느림
  - 초기 layer 에서 depthwise convolution 은 학습 속도를 느리게 함
  - 모든 stage를 동일한 비율로 scaling up 하는 것은 최적의 선택이 아님

1) **Progressive learning 방법 제안**
   - 이미지 크기에 따라 정규화 조정 => 정확도가 감소하는 문제 해결
     - 학습 시, 이미지의 크기를 점진적으로 증가
     - 입력 이미지가 작으면 약한 정규화, 입력 이미지가 크면 강한 정규화
   - 해당 방법을 통해 학습 속도와 정확도 향상
     - progressive learning 을 다른 모델에 적용해도 성능이 향상됨
   - 다음 세 가지 정규화에 대해서 다룸
     - Dropout: 무작위로 채널 제거
     - RandAugment: 이미지별 다양한 데이터 증가
     - Mixup: 교차 이미지 데이터 증가

   - Adaptive Regularization
     - 이미지의 크기에 따른 적응형 정규화. 모든 구조의 모델 학습에 간단하게 적용하여 모델 학습을 향상시킬 수 있음.



(a) Training efficiency.

	EfficientNet (2019)	ResNet-RS (2021)	DeiT/ViT (2021)	EfficientNetV2 (ours)
Top-1 Acc.	84.3%	84.0%	83.1%	83.9%
Parameters	43M	164M	86M	24M

(b) Parameter efficiency.

Table 5. ImageNet top-1 accuracy. We use RandAug (Cubuk et al., 2020), and report mean and stdev for 3 runs.

	Size=128	Size=192	Size=300
RandAug magnitude=5	**78.3** ±0.16	81.2 ±0.06	82.5 ±0.05
RandAug magnitude=10	78.0 ±0.08	**81.6** ±0.08	82.7 ±0.08
RandAug magnitude=15	77.7 ±0.15	81.5 ±0.05	**83.2** ±0.09

EfficientNetV2: Smaller Models and Faster Training

Mingxing Tan [1] Quoc V. Le [1]

https://arxiv.org/pdf/2104.00298.pdf

2) **Depthwise convolutions** are slow in early layers
- Depthwise convolution은 modern accelerator를 활용하기 때문에 학습 속도를 느리게 함.
- 따라서 stage 1-3 에서는 MBConv 대신에 Fused-MBConv 사용

- **MBConv** : 1*1 conv + 3*3 depthwise conv
- **Fused-MBConv** : 하나의 3*3 conv
  - 모든 stage 에 fused-MBConv를 적용하니, 오히려 학습 속도가 느려졌다고 함
  - 따라서 초기의 stage 만 Fused-MBConv 사용

3) **Equally scaling up every stage is sub-optimal(차선택) (non-uniform scaling strategy)**
- stage 들이 training speed에 기여하는 정도가 다 다름.
- stage가 증가할수록 layer가 증가하는 정도를 높인 것. 증가하는 정도는 heuristic하게 결정.
- 추가적으로 compound scaling 에서 maximum 이미지 사이즈를 작은 값으로 제한.

- Performance
  - Faster 11x training speed, better 6.8x parameter efficiency than SoTA

Table 10. Comparison with the same training settings – Our new EfficientNetV2-M runs faster with less parameters.

	Acc. (%)	Params (M)	FLOPs (B)	TrainTime (h)	InferTime (ms)
V1-B7	85.0	66	38	54	170
V2-M (ours)	85.1	55 (-17%)	24 (-37%)	13 (-76%)	57 (-66%)

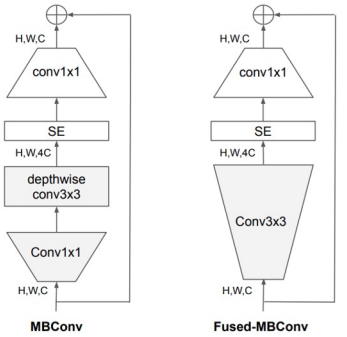


Figure 2. Structure of MBConv and Fused-MBConv.

Table 3. Replacing MBConv with Fused-MBConv. `No fused` denotes all stages use MBConv, `Fused stage1-3` denotes replacing MBConv with Fused-MBConv in stage {2, 3, 4}.

	Params (M)	FLOPs (B)	Top-1 Acc.	TPU imgs/sec/core	V100 imgs/sec/gpu
No fused	19.3	4.5	82.8%	262	155
Fused stage1-3	20.0	7.5	83.1%	362	216
Fused stage1-5	43.4	21.3	83.1%	327	223
Fused stage1-7	132.0	34.4	81.7%	254	206

Table 4. EfficientNetV2-S architecture – MBConv and Fused-MBConv blocks are described in Figure 2.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	272	15
7	Conv1x1 & Pooling & FC	-	1792	1

Focal Self-attention for Local-Global Interactions in Vision Transformers

Jianwei Yang[1] Chunyuan Li[1] Pengchuan Zhang[1] Xiyang Dai[2] Bin Xiao[2]
Lu Yuan[2] Jianfeng Gao[1]
[1]Microsoft Research at Redmond, [2]Microsoft Cloud + AI
{jianwyan,chunyl,penzhan,xidai,bixi,luyuan,jfgao}@microsoft.com

https://arxiv.org/pdf/2107.00641.pdf

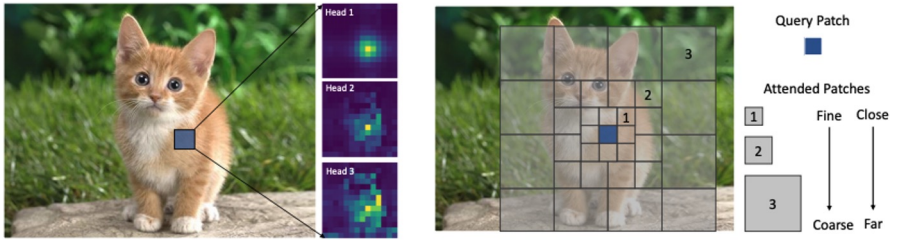


Figure 1: Left: Visualization of the attention maps of the three heads at the given query patch (blue) in the first layer of the DeiT-Tiny model [57]. Right: An illustrative depiction of focal self-attention mechanism. Three granularity levels are used to compose the attention region for the blue query.

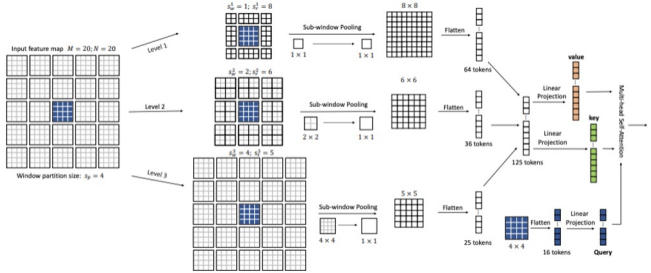


Figure 4: An illustration of our focal self-attention at window level. Each of the finest square cell represents a visual token either from the original feature map or the squeezed ones. Suppose we have an input feature map of size 20×20. We first partition it into 5×5 windows of size 4×4. Take the 4×4 blue window in the middle as the query, we extract its surroundings tokens at multiple granularity levels as its keys and values. For the first level, we extract the 8×8 tokens which are closest to the blue window at the finest grain. Then at the second level, we expand the attention region and pool the surrounding 2×2 sub-windows, which results in 6×6 pooled tokens. At the third level, we attend even larger region covering the whole feature map and pool 4×4 sub-windows. Finally, these three levels of tokens are concatenated to compute the keys and values for the $4 \times 4 = 16$ tokens (queries) in the blue window.

- **Microsoft Research**
- **Vision Transformer**
  - self-attention 을 통한 visual dependencies (short-, long- range) 가 성능에 중요한 영향을 끼침.
  - 그러나 이는, object detection 과 같은 high-resolution vision tasks 에서 quadratic computation overhead challenge 가 있음.
  - 최근 연구에서, **coarse-grained global attentions, fine-grained local attentions 를 적용**함으로써 computational and memory cost를 낮추고, 성능을 향상 시킴.
    - 이러한 접근법은 multi-layer Transformers 의 original self-attention mechanism 의 modeling power 를 저하시킴. (sub-optimal)

- **focal self-attention & Focal Transformer (SoTA)**
  - **fine-grained local 과 coarse-grained global interactions 을 모두 적용한 새로운 메커니즘.**
  - **각 token 은 가장 가까운 주변 token 을 fine granularity (미세하게 세분화) 하고,**
    멀리 떨어진 token 은 coarse granularity (거친 세분화) 하여,
    short- and long- range 의 visual dependencies 를 효과적으로 생성할 수 있음. => 적은 cost 로 더 넓은 영역 cover 가능.

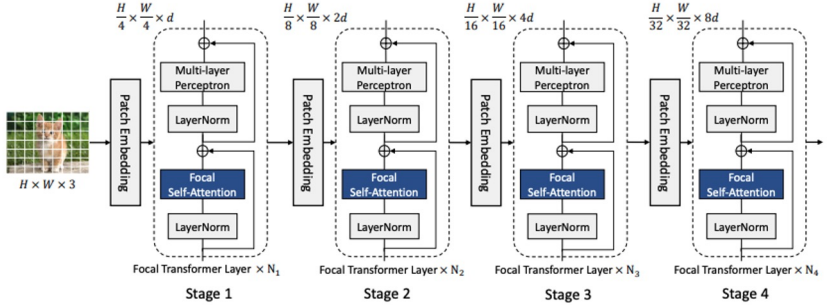


Figure 2: Model architecture for our Focal Transformers. As highlighted in light blue boxes, our main innovation is the proposed focal self-attention mechanism in each Transformer layer.

Method	#Param	FLOPs	mini-val AP^b	mini-val AP^m	test-dev AP^b	test-dev AP^m
X101-64x4d [70]	155M	1033G	52.3	46.0	-	-
EfficientNet-D7 [56]	77M	410G	54.4	-	55.1	-
GCNet* [8]	-	1041G	51.8	44.7	52.3	45.4
ResNeSt-200 [79]	-	-	52.5	-	53.3	47.1
Copy-paste [28]	185M	1440G	55.9	47.2	56.0	47.4
BoTNet-200 [52]	-	-	49.7	-	-	-
SpineNet-190 [24]	164M	1885G	52.6	-	52.8	-
CenterNet2 [90]	-	-	-	-	56.4	-
Swin-L (HTC++) [44]	284M	1470G	57.1	49.5	57.7	50.2
Swin-L (DyHead) [19]	213M	965G	56.2	-	-	-
Swin-L† (HTC++) [44]	284M	-	58.0	50.4	58.7	51.1
Swin-L† (DyHead) [19]	213M	-	58.4	-	58.7	-
Swin-L† (QueryInst) [26]	-	-	56.1	-	56.1	-
Focal-L (HTC++) (Ours)	265M	1165G	57.0	49.9	-	-
Focal-L (DyHead) (Ours)	229M	1081G	56.4	-	-	-
Focal-L† (HTC++) (Ours)	265M	-	58.1	**50.9**	58.4	**51.3**
Focal-L† (DyHead) (Ours)	229M	-	**58.7**	-	**58.9**	-

Table 6: Comparison with state-of-the-art methods on COCO object detection and instance segmentation. The numbers are reported on 5K val set and test-dev. Augmented HTC [13] (denoted by HTC++) and DyHead [19] are used as the detection methods. † means multi-scale evaluation.

Backbone	Method	#Param	FLOPs	mIoU	+MS
ResNet-101	DANet [46]	69M	1119G	45.3	-
ResNet-101	ACNet [27]	-	-	45.9	-
ResNet-101	DNL [73]	69M	1249G	46.0	-
ResNet-101	UperNet [68]	86M	1029G	44.9	-
HRNet-w48 [54]	OCRNet [77]	71M	664G	45.7	-
ResNeSt-200 [79]	DLab.v3+ [14]	88M	1381G	48.4	-
Swin-T [44]	UperNet [68]	60M	945G	44.5	45.8
Swin-S [44]	UperNet [68]	81M	1038G	47.6	49.5
Swin-B [44]	UperNet [68]	121M	1188G	48.1	49.7
Twins-SVT-L [17]	UperNet [68]	133M	-	48.8	50.2
MiT-B5 [69]	SegFormer [69]	85M	-	51.0	51.8
ViT-L/16† [23]	SETR [85]	308M	-	50.3	-
Swin-L‡ [44]	UperNet [68]	234M	3230G	52.1	53.5
ViT-L/16‡ [23]	Segmenter [53]	334M	-	51.8	53.6
Swin-L‡ [44]	K-Net [82]	-	-	-	54.3
Swin-L‡ [44]	PatchDiverse [29]	234M	-	53.1	54.4
VOLO-D5 [76]	UperNet [68]	-	-	-	54.3
Focal-T (Ours)	UperNet [68]	62M	998G	45.8	47.0
Focal-S (Ours)	UperNet [68]	85M	1130G	48.0	50.0
Focal-B (Ours)	UperNet [68]	126M	1354G	49.0	50.5
Focal-L‡ (Ours)	UperNet [68]	240M	3376G	**54.0**	**55.4**

Table 7: Comparison with SoTA methods for semantic segmentation on ADE20K [88] val set. Both single- and multi-scale evaluations are reported at the last two columns. ‡ means pretrained on ImageNet-22K.

- **Focal Transformers (SoTA)**
  - 각 transformer layer에 focal self-attention 메커니즘 적용함.

- Experiments
  - **public image classification and object detection benchmarks 에서 SoTA vision Transformers의 성능을 뛰어넘음.**
  - 83.5% and 83.8% Top-1 accuracy, on ImageNet classification at 224 * 244
  - **Backbones 으로 사용하였을 때, SoTA 인 Swin Transformers (지난 주 현규님께서 소개해 주신 아키텍쳐) 보다 성능 향상**
  - New SoTA on three of the most challenging computer vision tasks
    - 58.7/58.9 box mAPs, 50.9/51.3 mask mAPs on COCO mini-val/test-dev, 55.4 mIoU on ADE20K for semantic segmentation.

Attention Bottlenecks for Multimodal Fusion

Arsha Nagrani Shan Yang Anurag Arnab Aren Jansen
Cordelia Schmid Chen Sun
{anagrani, shanyang, aarnab, arenjansen, cordelias, chensun}@google.com
Google Research

https://arxiv.org/pdf/2107.00135.pdf

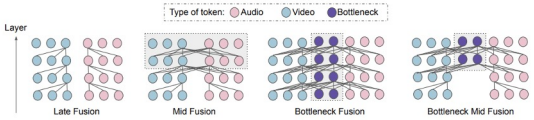


Figure 1: **Cross-modal Fusion**. Unlike late fusion (left), where no cross-modal information is exchanged in the model until after the classifier, we investigate two pathways for the exchange of cross-modal information. The first is via standard pairwise self attention across all hidden units in a layer, but applied only to later layers in the model – mid fusion (middle, left). We also propose the use of 'fusion bottlenecks' (middle, right) that restrict attention flow within a layer through tight latent units. Both forms of restriction can be applied in conjunction (Bottleneck Mid Fusion) for optimal performance (right). We show $B = 2$ bottleneck units and 3 hidden units per modality. Grey boxes indicate tokens that receive attention flow from both audio and video tokens.

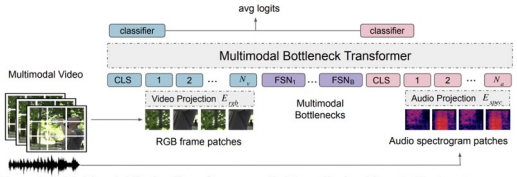


Figure 2: **A Multimodal Fusion Transformer applied to audiovisual inputs.** The input sequence consists of image and spectrogram patches. These are then projected into tokens and appended to special *CLS* (classification) and FSN (fusion bottleneck) tokens. Our transformer encoder then uses self attention to model unimodal information, and restricts cross-modal information flow via cross attention with the bottleneck tokens at multiple layers of the network.

Model	Modalities	Verb	Noun	Action
Damen et al. [13]	A	42.1	21.5	14.8
AudioSlowFast [34]†	A	46.5	22.78	15.4
TSN [57]	V, F	60.2	46.0	33.2
TRN [63]	V, F	65.9	45.4	35.3
TBN [33]	A, V, F	66.0	47.2	36.7
TSM [42]	V, F	**67.9**	49.0	38.3
SlowFast [20]	V	65.6	50.0	38.5
MBT	A	44.3	22.4	13.0
MBT	V	62.0	56.4	40.7
MBT	A, V	64.8	**58.0**	**43.4**

Table 2: **Comparison to the state of the art on Epic Kitchens 100 [13]**. Modalities (Mods) are **A:** Audio, **V:** Visual, **F:** Optical flow.

- **Google Research**
- **Multimodal + Transformer**
  - modality fusion을 위해 'fusion bottlenecks'를 사용하는 transformer 아키텍처 제
  - training experiment 간 Mixup regularization 적용
  - multiple audio-visual classification benchmarks 에서 SoTA 성능 도달 (AudioSet, Epic-Kitches and VGG sound)
- **Fusion Bottlenecks tokens (FSN)**
  - traditional self-attention과 비교하여, 다른 modalities 간의 정보를 소수의 bottleneck에 통과시킴
  - 이는 모델이 각 modality 에서 가장 관련성이 높은 정보를 수집 및 압축하고, 필요한 정보만 공유하도록 함
  - 해당 전략을 통해 fusion 성능 뿐 아니라, computational cost 도 절감
- 모든 코드와 모델 공유, 제안한 논문에 대한 ablation study 도 포함

On the Practicality of Deterministic Epistemic Uncertainty

Janis Postels[*1], **Mattia Segu**[*1 2], **Tao Sun**[1], **Luc Van Gool**[1], **Fisher Yu**[1] **Federico Tombari**[3 4]
[1] ETH Zurich [2] Max Planck ETH CLS [3] Google [4] TUM Munich
{jpostels, segum, taosun47, vangool}@ethz.ch, i@yf.io tombari@google.com

https://arxiv.org/pdf/2107.00649v1.pdf

DUMs		Uncertainty Estimation Method			
		Discriminative		Generative	
		Distance from class centroid	Gaussian Processes	Gaussian Mixture Models	Normalizing Flows
Target Property	Distance awareness	DCS [22], DUQ [23]	SNGP [24], DUE [25]	DDU [26]	-
	Informative representations	-	-	DCU [27], MIR [28]	Invertible networks [29, 30, 31]

Table 1: Taxonomy of DUMs. Methods are grouped according to the target property of the hidden representations (rows), and their uncertainty estimation method (columns).

Method	MNIST → F-MNIST		MNIST → Omniglot		F-MNIST → MNIST		F-MNIST → Omniglot	
	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
Softmax	89.0 ± 1.2	88.5 ± 1.4	95.1 ± 0.3	94.5 ± 0.4	73.2 ± 3.5	75.9 ± 2.6	75.9 ± 1.5	74.1 ± 1.6
Dropout	94.4 ± 0.9	92.0 ± 1.8	94.8 ± 0.2	92.0 ± 0.5	95.8 ± 0.2	93.6 ± 0.6	96.3 ± 0.2	94.3 ± 0.3
Ensemble	95.2± 1.0	91.4 ± 3.6	**97.3 ± 0.4**	95.3 ± 0.1	87.0 ± 1.5	81.4 ± 0.3	90.7 ± 1.1	87.6 ± 2.5
DUE	90.8 ± 2.3	90.9 ± 2.0	94.2 ± 0.2	92.8 ± 0.2	68.2 ± 1.9	65.7 ± 3.0	72.4 ± 1.7	65.8 ± 2.8
DUQ	90.2 ± 3.0	92.2 ± 2.4	93.8 ± 0.3	93.8 ± 0.4	95.1 ± 1.1	95.9 ± 1.0	94.7 ± 0.6	94.3 ± 0.7
DDU	83.9 ± 7.6	83.4 ± 7.9	75.2 ± 6.7	69.4 ± 9.5	90.8 ± 5.5	92.2 ± 4.8	90.6 ± 4.6	90.3 ± 4.6
SNGP	93.2 ± 1.2	94.6 ± 1.3	94.8 ± 0.7	93.9 ± 0.7	89.2 ± 1.1	87.9 ± 1.3	89.8 ± 1.7	85.5 ± 2.7
MIR	**97.0 ± 0.7**	**97.7 ± 0.5**	**97.3 ± 0.6**	**97.4 ± 0.5**	**99.0 ± 0.3**	**99.2 ± 0.2**	**97.9 ± 0.2**	**97.6 ± 0.4**

Table 3: Experiments for OOD detection (AUROC and AUPR, %). The baseline method and all DUMs use a MLP trained on MNIST/FashionMNIST and predict on another dataset. MIR demonstrates strong performance. (F-MNIST = FashionMNIST).

- **Out-Of-Distribution (OOD) & Deterministic Uncertainty Methods (DUMs)**
  - informative representations 을 전제로, deterministic uncertainty methods (DUMs) 은 OOD detecting에 좋은 성능을 보임
  - 그러나, DUMs이 real-world 어플리케이션에 실제 적용이 가능한지, 잘 calibration 되어 있는지는 불분명함
- **본 논문에서는 DUMs 분류 체계 성립과 각 task 에서의 성능을 분석함**
  - image classification tasks
  - semantic segmentation
- **결론) DUMs가 realistic vision tasks에서 OOD detection 을 잘 수행하는 것을 확인하였지만, realistic distributional shifts 하에서는 calibration이 잘 이뤄지지 않아, 현재 방법으로는 실용성이 떨어짐**

End of the Document