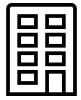
2023 하계 세미나

Convolution in 2020s and Self-Supervised Vision Learners



Sogang University
Vision & Display Systems Lab, Dept. of Electronic Engineering

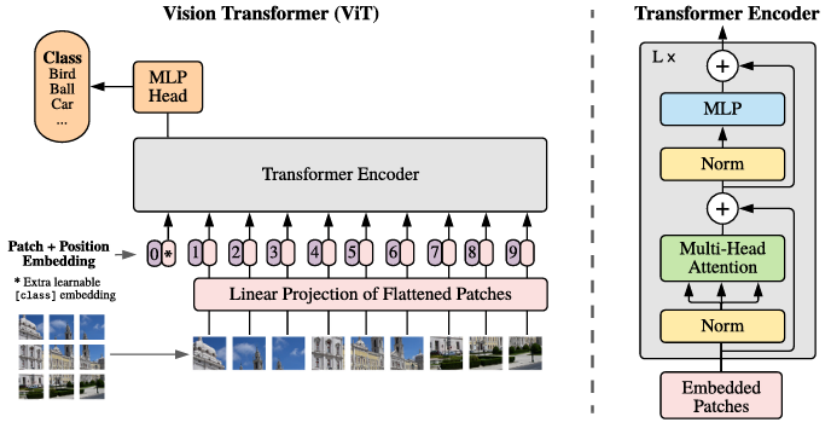
Presented By
JoonKyu Kim

Index

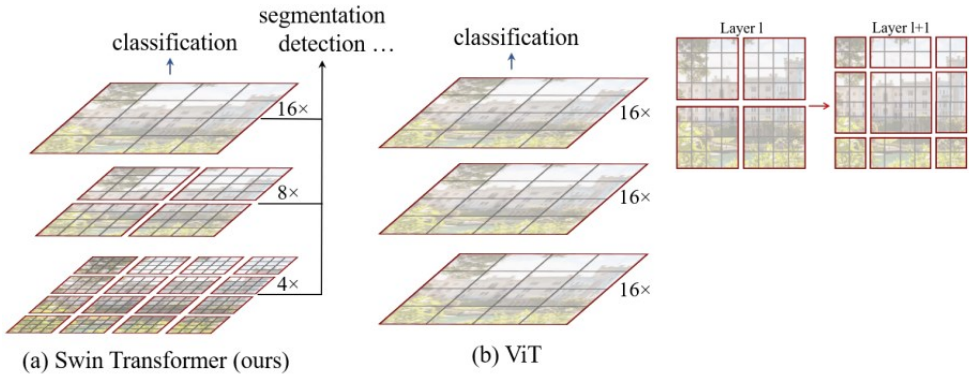
- Recent Transformers in the field of Image Restoration
  - Transformers in the field of High Dynamic Range Imaging
- ConvNeXt
  - A convnet for the 2020s (CVPR 2022)
- Masked Autoencoder
  - Masked Autoencoders Are Scaleable Vision Learners (CVPR 2022)
  - Learning Prior Feature and Attention Enhanced Image Inpainting (ECCV 2022)
- ConvNeXt V2
  - Convnext V2: Co-designing and scaling convnets with masked autoencoders (CVPR 2023)

Background

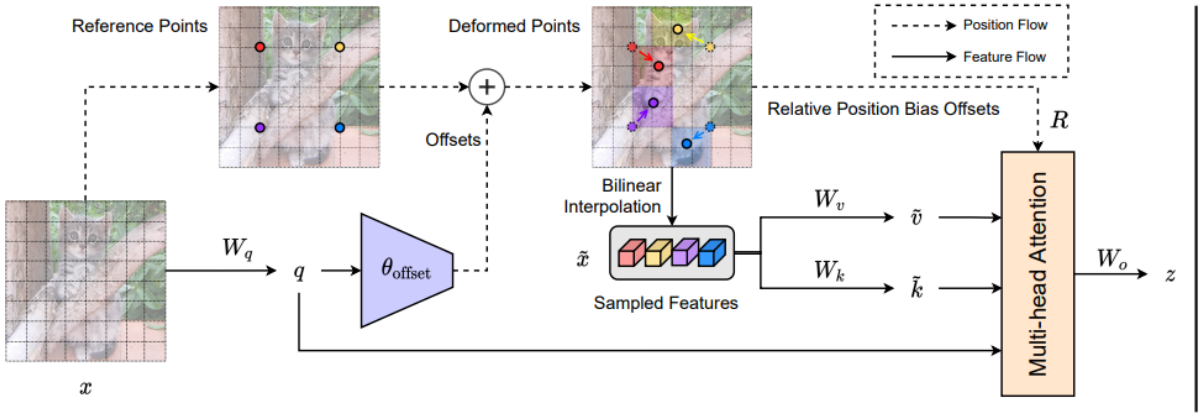
- ViT [1]

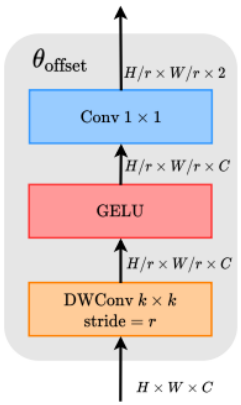


- Swin Transformer [2]



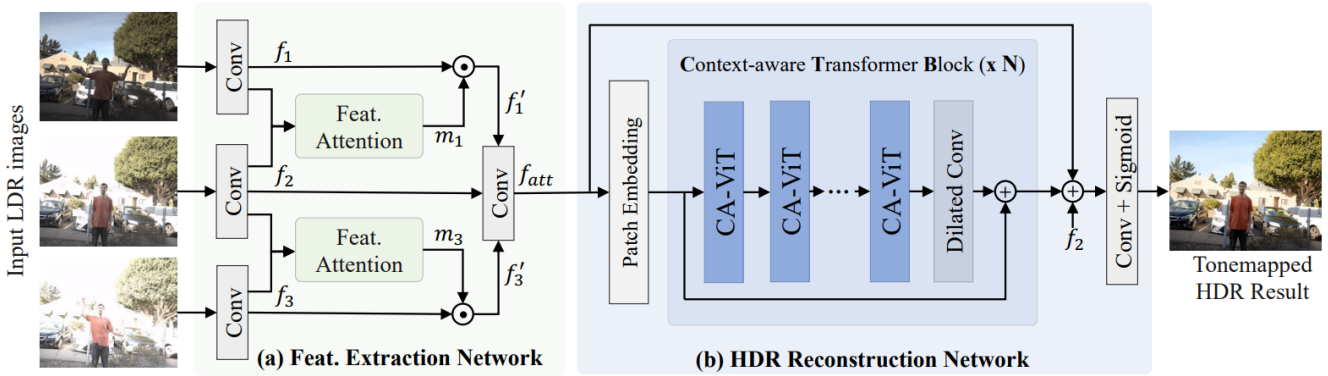
- DAT [3] (Deformable Attention Transformer)

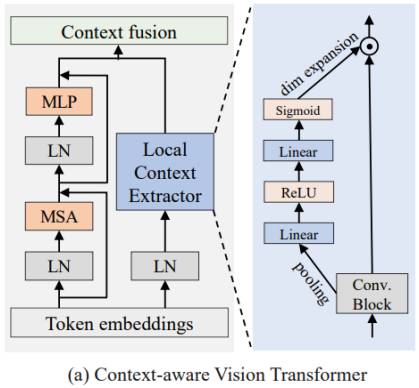




In HDR…

- Ghost-free High Dynamic Range Imaging with Context-aware Transformer (ECCV 2022)



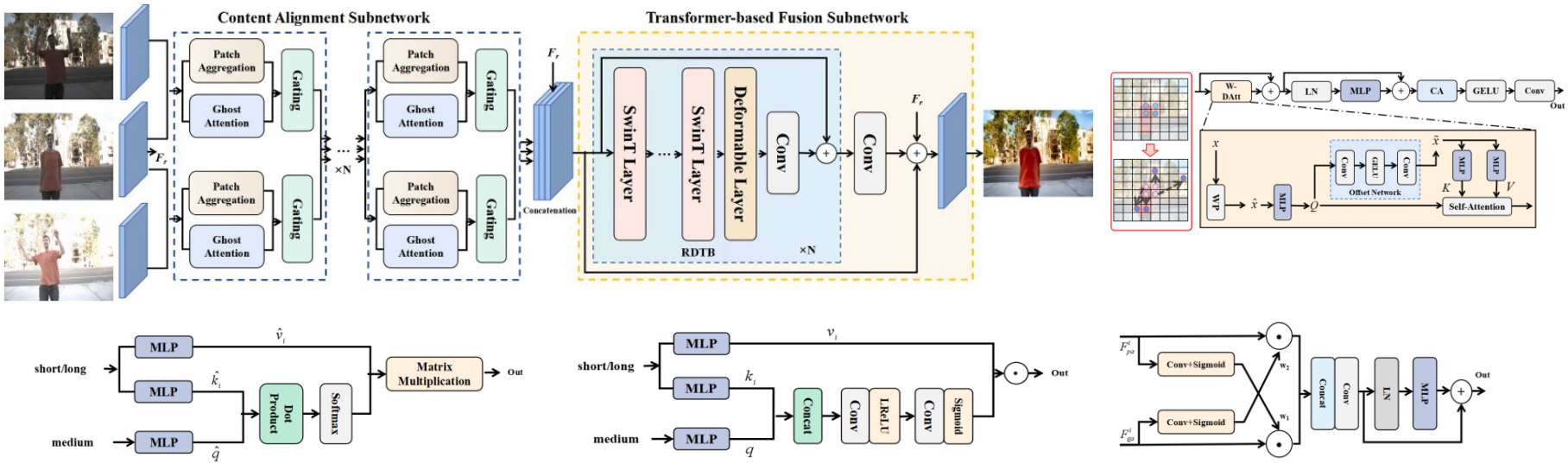


(a) Context-aware Vision Transformer

- Batch size : 16 with four NVIDIA 2080Ti GPU (48GB / 16)
- Patch size : ‘8’
- Trained for 100 epochs

In HDR…

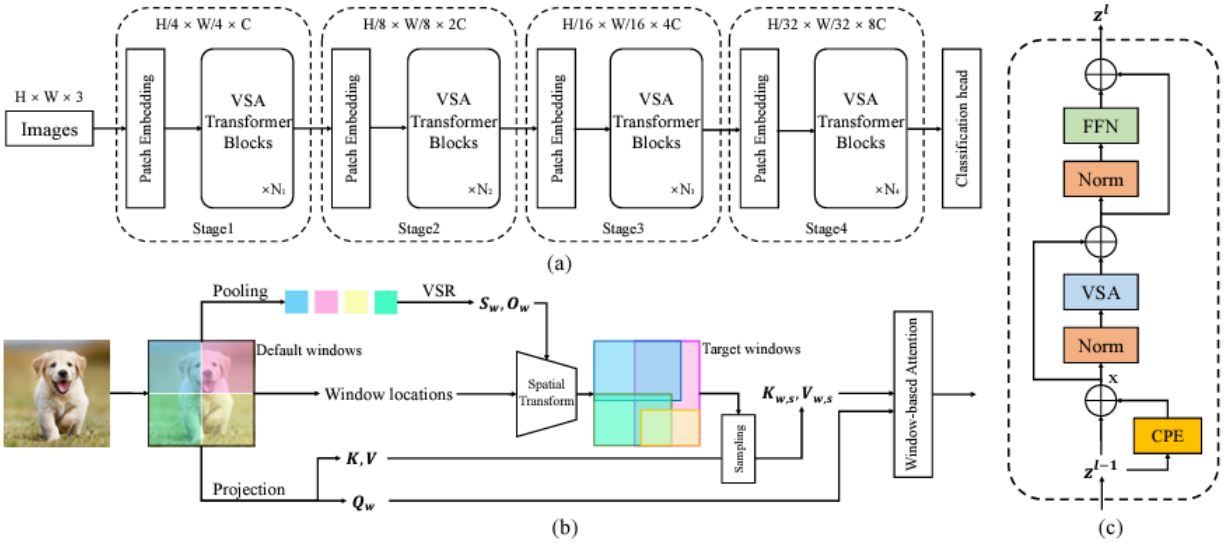
- A Unified HDR Imaging Method with Pixel and Patch Level (CVPR 2023)



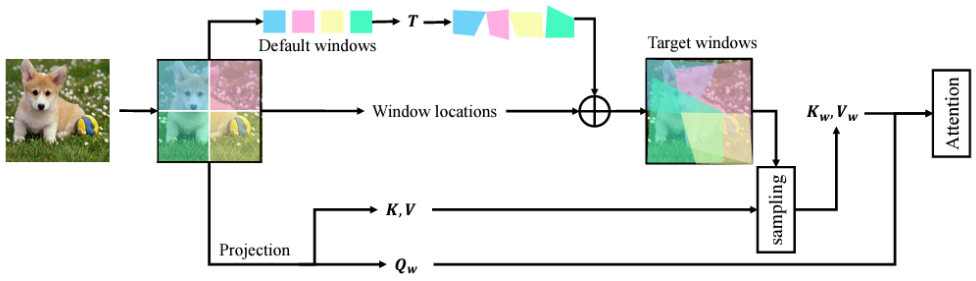
  - Batch size : 4 with two NVIDIA RTX 3090 GPU (48GB / 4)
  - Patch size : ‘8’
  - Trained for 150 epochs
- What’s next?
  - Another Transformers… More Params… More FLOPs…

Candidates

- ViTAE [1]

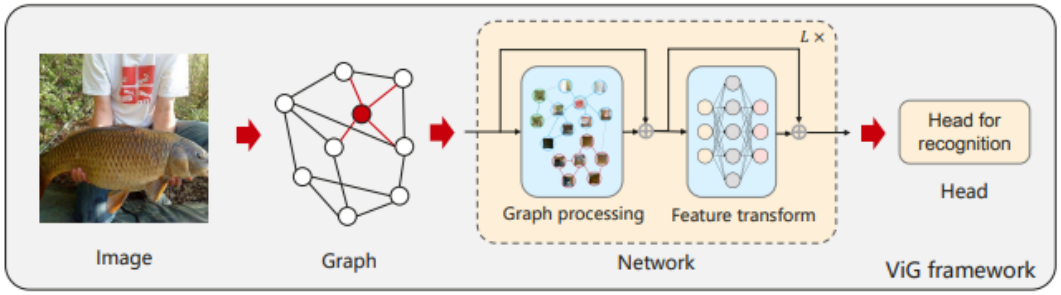


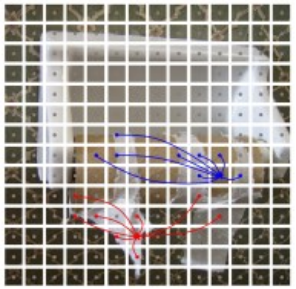
- QFormer [2]



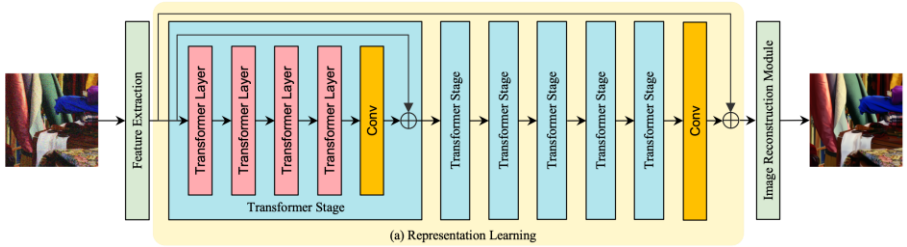
Candidates

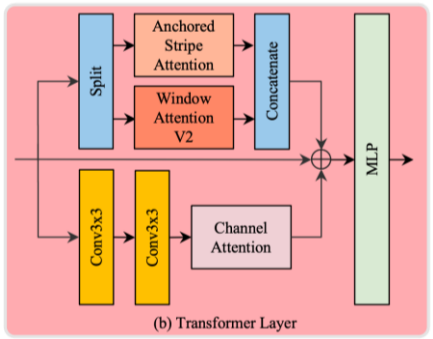
- ViG [1]

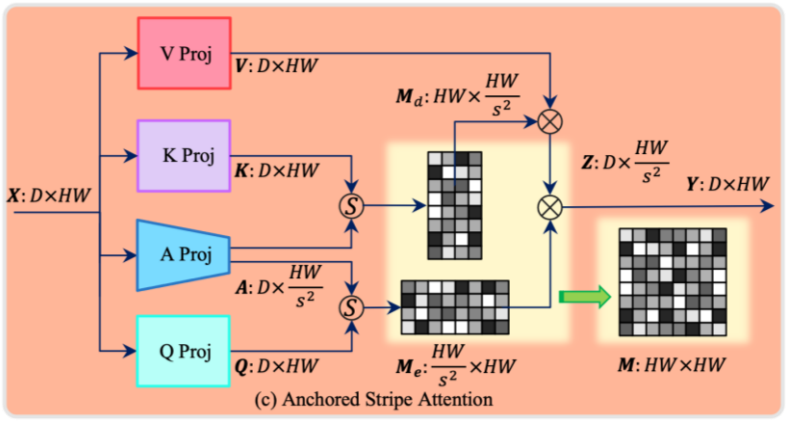




- GRL [2]

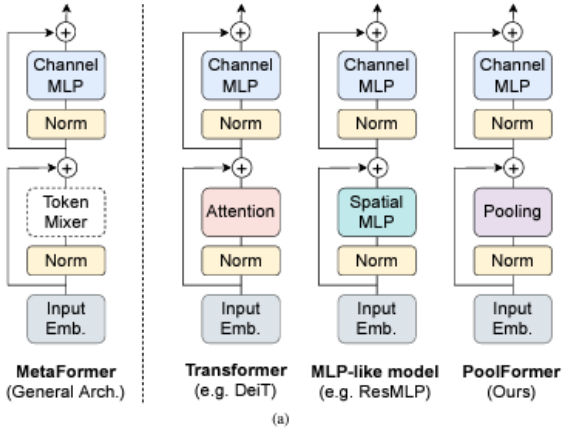






ConvNeXt [1]

- A ConvNet for the 2020s (CVPR 2022)
  - Convolutional Neural Network
    - Inductive bias
    - Computationally efficient
    - Stack multiple layers for wider receptive field
  - Transformers
    - Outperform standard ResNet[1]
    - Quadratic complexity w.r.t the input size
    - Large receptive field
      - Receptive field를 제한하는 Swin Transformer 기반의 방법들을 중심으로 발전하였음
        - ✓ 이는 결국 CNN의 'local한 부분을 본다'라는 개념이 포함된 Swin Transformer가 좋은 성능을 보일 수 있었던 것
        - ✓ 결국 CNN이 meaningless하지 않다는 것을 보여줌
  - 반대로 Transformer 구조를 ResNet [1]에 적용하여 성능 개선이 가능하지 않을까?
    - 'Convolutional network만 사용해서 어디까지 갈 수 있을까'

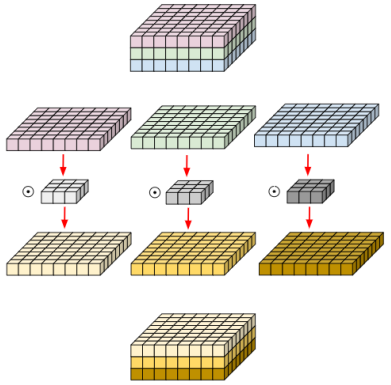


< PoolFormer[3] >

ConvNeXt [1]

- ResNet-50 : 76.1%
  - Training epochs
    - 기존 ResNet의 90 epochs 학습을 300 epochs 학습으로 변경
    - AdamW optimizer 사용
    - Data augmentation (Mixup, Cutmix, RandAugment, Random Erase)
    - Regularization (Stochastic Depth, Label Smoothing)
    - **76.1% → 78.8% (+2.7%)**
  - Stage compute ratio 조정
    - Swin-T의 각 stage에서의 block 수는 1:1:3:1
    - 기존 ResNet의 stage별 block 구성 개수 변경
      - C = (96, 102, 384, 768), B = (3, 4, 6, 3) → C = (96, 102, 384, 768), B = (3, 3, 9, 3)
    - **78.8% → 79.4% (+0.6%)**

ConvNeXt [1]



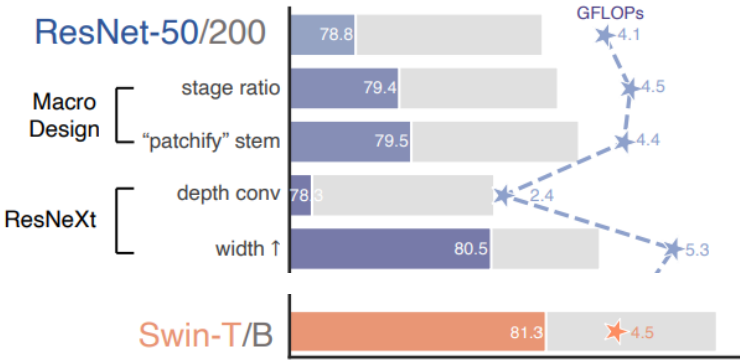
< Depthwise Conv >

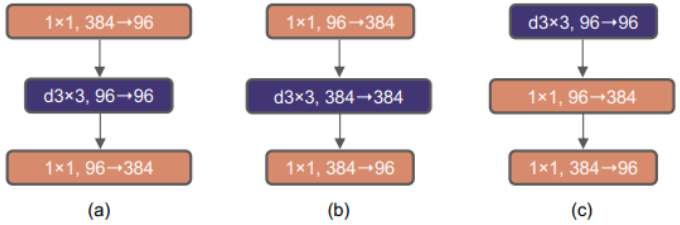
- Patchify stem
  - ResNet : 7x7 conv (stride 2) 이후 max pooling 수행
    - Downsampling (4)
  - ViT : non-overlapping convolution으로 patchify 수행
    - ViT : 14 or 16 정도의 큰 kernel size로 patchify 수행
    - Swin Transformer : patch size 4정도의 작은 kernel size
  - ResNet의 conv-maxpool을 4x4, stride 4의 convolution layer로 변경
  - **79.4% → 79.5% (+0.1%)**

- ResNeXt-ify
  - Channel width를 증가시키고, convolution group의 개수를 늘림
    - Group의 개수를 width까지 증가시켜 depthwise convolution을 사용함
    - Channel, Group의 개수를 기존 64에서 Swin-T와 동일하게 96까지 증가시킴
  - **79.5% → 80.5% (+0.6%)**

ConvNeXt [1]

- Depthwise Conv 적용으로 FLOPs 감소 / Channel width 증가로 FLOPs 증가
  - 목표는 Swin-T의 4.5GFLOPs와 동일한 수준으로 만드는 것





- Inverted Bottleneck
  - ResNet의 bottleneck 구조는 dim을 4배 줄였다가 다시 확장하는 구조 (a)
  - 이를 반대로 수행하여 Transformer에서 MLP 구조와 같이 4배 확장 후 다시 감소하는 형태로 변경하여 수행 (b)
  - MLP 구조에서의 Linear layer와 1x1 conv는 동일한 연산임
    - MLP 구조에서는 Linear layer가 연속해서 두 번 등장함
    - 따라서 (b) 구조에서 Dconv의 위치를 위로 바꿔서 연산 수행
      - ✓ Dconv를 수행하는 channel size가 384에서 96으로 변경되면서 연산량 감소
      - ✓ 성능은 소폭 하락 (-0.6%)

ConvNeXt [1]

- Kernel size
  - Swin Transformer의 window size가 최소 7x7 크기임
    - ResNet은 3x3 kernel size를 가짐
  - 따라서 kernel size를 3, 5, 7, 9, 11로 늘려가며 실험 진행
    - Kernel size가 7일 때부터 성능 향상 폭이 saturation됨
  - **79.9% → 80.6% (+0.7%)**

- Activation function
  - ReLU to GELU
  - Fewer activation function
    - Transformer 구조를 살펴보면 activation function이 많지 않음
      - ✓ Embedding (x), self-attention (x), MLP 쪽에만 activation function을 사용함
    - 동일하게 두 개의 1x1 conv 사이에만 GELU를 하나 사용하고 나머지는 모두 제거
  - **80.6% → 81.3% (+0.7%)**

ConvNeXt [1]

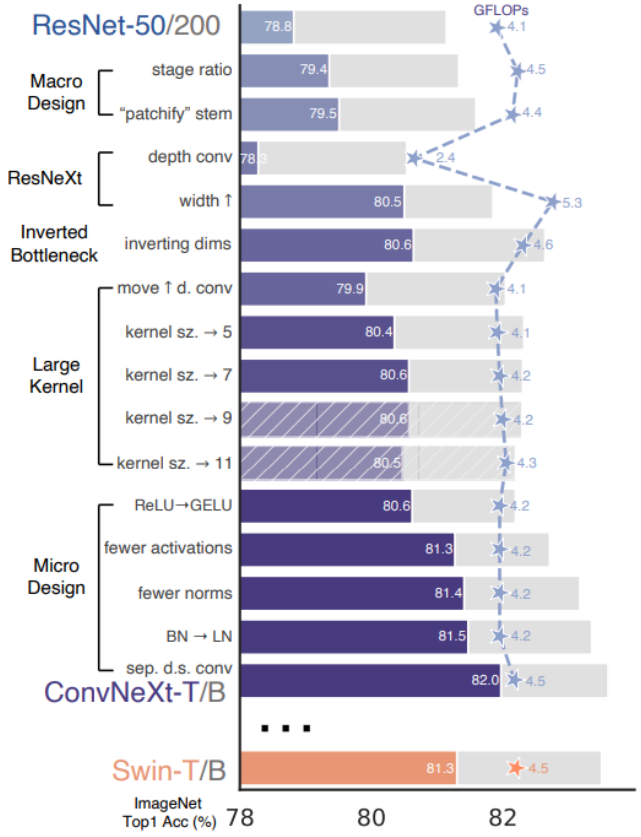
- Normalization layers
  - Activation layer와 동일하게 transformer 구조와 비슷하게 normalization layer 또한 줄임
    - 1x1 conv 직전에 한 번의 BN만 수행
  - **81.3% → 81.4% (+0.1%)**
  - BatchNorm 대신에 LayerNorm 사용
    - 일반적인 convolution layer에서 BatchNorm 대신 LayerNorm으로 바꾸면 보통 성능이 크게 하락함
    - 하지만 이전에 적용한 변경사항들에 의해서 성능이 하락하지 않고 학습이 안정적으로 진행될 수 있었음
  - **81.4% → 81.5% (+0.1%)**

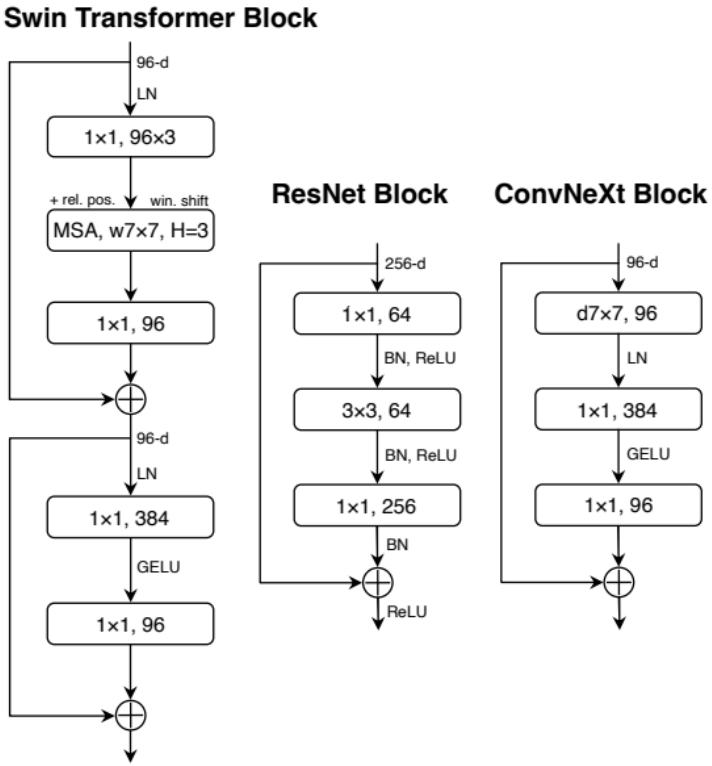
ConvNeXt [1]

- Separate Downsampling Layers
  - 기존 ResNet의 downsampling은 첫 번째 layer를 변경하여 3x3 conv, stride 2로 구성하고, shortcut connection은 1x1 conv, stride 2로 구성하여 수행하였음
  - 반면 Swin Transformer를 보면 downsampling layer가 따로 구성되어 있음
  - 따라서 Swin Transformer와 동일하게 따로 downsampling layer를 추가하였음
    - 2x2 conv, stride 2 layer를 추가하여 사용
      - ✓ 학습 시 stablize가 되지 않아 conv 앞에 normalization layer를 추가함
  - **81.5% → 82.0% (+0.5%)**

ConvNeXt [1]

- Swin-T와 동일한 FLOPs, params, throughput, memory use를 가지면서도 더 좋은 성능을 보임
  - 특히, MSA, S-MSA 와 같은 특별한 block이 필요하지 않다는 점이 의미있음





ConvNeXt [1]

- Image classification (ImageNet-1K)

model	image size	#param.	FLOPs	throughput (image / s)	IN-1K top-1 acc.
ImageNet-1K trained models					
• RegNetY-16G [54]	224^2	84M	16.0G	334.7	82.9
• EffNet-B7 [71]	600^2	66M	37.0G	55.1	84.3
• EffNetV2-L [72]	480^2	120M	53.0G	83.7	85.7
○ DeiT-S [73]	224^2	22M	4.6G	978.5	79.8
○ DeiT-B [73]	224^2	87M	17.6G	302.1	81.8
○ Swin-T	224^2	28M	4.5G	757.9	81.3
• ConvNeXt-T	224^2	29M	4.5G	774.7	**82.1**
○ Swin-S	224^2	50M	8.7G	436.7	83.0
• ConvNeXt-S	224^2	50M	8.7G	447.1	**83.1**
○ Swin-B	224^2	88M	15.4G	286.6	83.5
• ConvNeXt-B	224^2	89M	15.4G	292.1	**83.8**
○ Swin-B	384^2	88M	47.1G	85.1	84.5
• ConvNeXt-B	384^2	89M	45.0G	95.7	**85.1**
• ConvNeXt-L	224^2	198M	34.4G	146.8	**84.3**
• ConvNeXt-L	384^2	198M	101.0G	50.4	**85.5**

- Object detection/segmentation (COCO)

backbone	FLOPs	FPS	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
Mask-RCNN 3× schedule								
○ Swin-T	267G	23.1	46.0	68.1	50.3	41.6	65.1	44.9
• ConvNeXt-T	262G	25.6	**46.2**	67.9	50.8	**41.7**	65.0	44.9
Cascade Mask-RCNN 3× schedule								
• ResNet-50	739G	16.2	46.3	64.3	50.5	40.1	61.7	43.4
• X101-32	819G	13.8	48.1	66.5	52.4	41.6	63.9	45.2
• X101-64	972G	12.6	48.3	66.4	52.3	41.7	64.0	45.1
○ Swin-T	745G	12.2	50.4	69.2	54.7	43.7	66.6	47.3
• ConvNeXt-T	741G	13.5	**50.4**	69.1	54.8	**43.7**	66.5	47.3
○ Swin-S	838G	11.4	51.9	70.7	56.3	45.0	68.2	48.8
• ConvNeXt-S	827G	12.0	**51.9**	70.8	56.5	**45.0**	68.4	49.1
○ Swin-B	982G	10.7	51.9	70.5	56.4	45.0	68.1	48.9
• ConvNeXt-B	964G	11.4	**52.7**	71.3	57.2	**45.6**	68.9	49.5
○ Swin-B‡	982G	10.7	53.0	71.8	57.5	45.8	69.4	49.7
• ConvNeXt-B‡	964G	11.5	**54.0**	73.1	58.8	**46.9**	70.6	51.3
○ Swin-L‡	1382G	9.2	53.9	72.4	58.8	46.7	70.1	50.8
• ConvNeXt-L‡	1354G	10.0	**54.8**	73.8	59.8	**47.6**	71.3	51.7
• ConvNeXt-XL‡	1898G	8.6	**55.2**	74.2	59.9	**47.7**	71.6	52.2

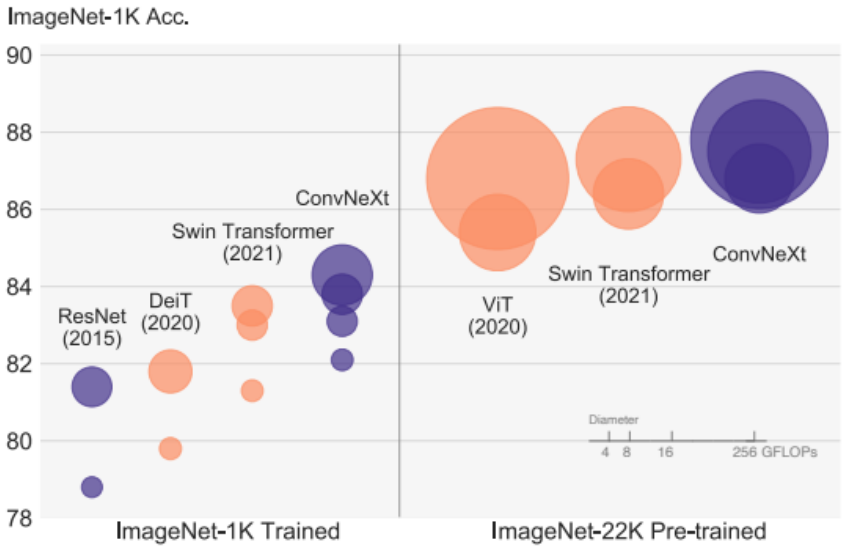
ConvNeXt [1]

- Image segmentation (ADE20K)

backbone	input crop.	mIoU	#param.	FLOPs
	ImageNet-1K pre-trained			
○ Swin-T	512^2	45.8	60M	945G
● ConvNeXt-T	512^2	**46.7**	60M	939G
○ Swin-S	512^2	49.5	81M	1038G
● ConvNeXt-S	512^2	**49.6**	82M	1027G
○ Swin-B	512^2	49.7	121M	1188G
● ConvNeXt-B	512^2	**49.9**	122M	1170G
	ImageNet-22K pre-trained			
○ Swin-B‡	640^2	51.7	121M	1841G
● ConvNeXt-B‡	640^2	**53.1**	122M	1828G
○ Swin-L‡	640^2	53.5	234M	2468G
● ConvNeXt-L‡	640^2	**53.7**	235M	2458G
● ConvNeXt-XL‡	640^2	**54.0**	391M	3335G

- Isotropic style (No downsampling)

model	#param.	FLOPs	throughput (image / s)	training mem. (GB)	IN-1K acc.
○ ViT-S	22M	4.6G	978.5	4.9	79.8
● ConvNeXt-S (*iso.*)	22M	4.3G	1038.7	4.2	79.7
○ ViT-B	87M	17.6G	302.1	9.1	81.8
● ConvNeXt-B (*iso.*)	87M	16.9G	320.1	7.7	82.0
○ ViT-L	304M	61.6G	93.1	22.5	82.6
● ConvNeXt-L (*iso.*)	306M	59.7G	94.4	20.4	82.6

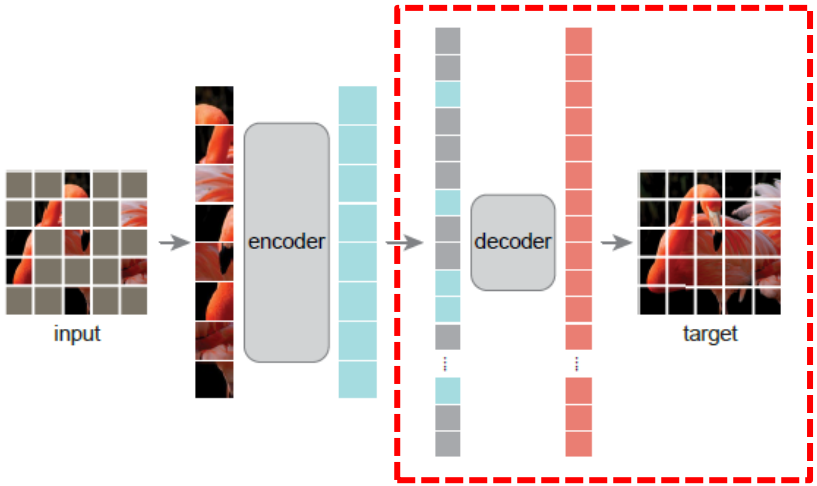


Masked Autoencoders [1]

- Self-Supervised Learning
  - 시간이 지날수록 model 학습에 더 많은 data를 필요로 함
    - 하지만 data에 접근할 수 없거나, labeling에 많은 노력이 필요함
    - Language processing 분야에서는 BERT, GPT와 같은 model들이 이러한 더 많은 데이터를 학습하는데 있어 큰 성과를 보임
      - 데이터의 일부를 masking하고 제거된 부분을 예측하도록 학습을 수행함
  - 하지만 vision 분야에서는 이러한 시도가 잘 이루어지지 못함
    - Architecture의 차이가 존재
      - NLP는 RNN, Transformer와 같은 구조로, masking이 용이하지만 vision에서는 CNN이 주류를 이루어 masking이 용이하지 않았음
        - ✓ ViT가 등장하며 이러한 부분이 해결되었음
    - Information density의 차이가 존재함
      - Language는 density가 높아 단어 하나하나의 의미가 큰 반면, image는 각 patch의 중요도가 크지 않음
        - ✓ Large ratio masking을 수행

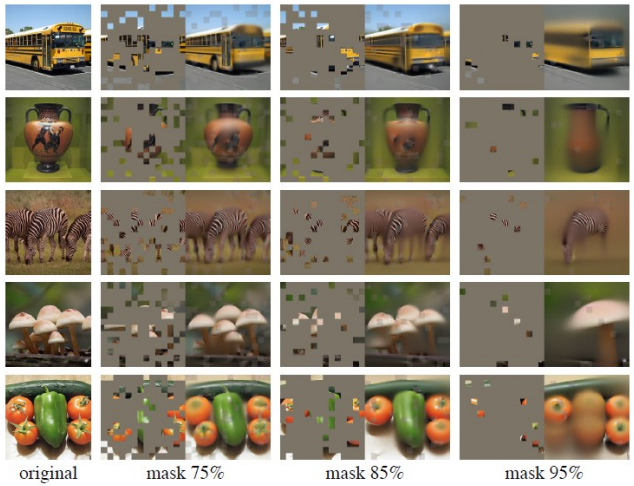
Masked Autoencoders [1]

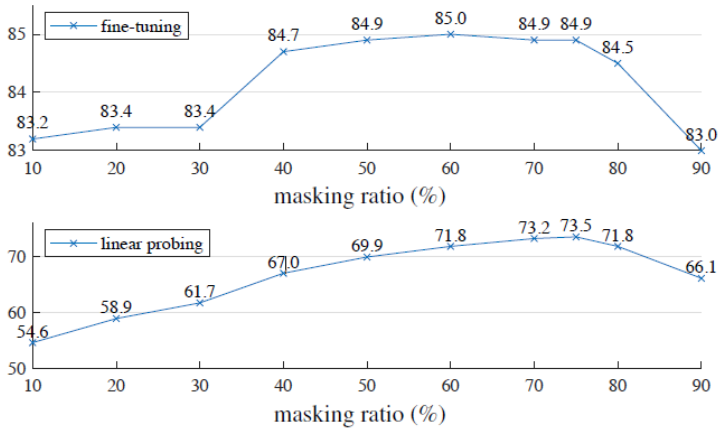
- MAE pretraining
  - 입력 이미지에 uniform한 random masking을 수행 (image patch의 75%)
    - Patch를 random shuffle하고 뒤쪽 75%를 버림
    - Masking된 부분을 reconstruction (L2 loss)하는 방법으로 pretraining 수행
      - Masking된 부분에 대해서만 Loss를 적용함
    - Masking된 patch는 encoder에 입력하지 않음
    - Decoder에서는 masking된 patch도 입력하여 출력 이미지와 동일한 length의 patch를 입력
    - Decoder 부분은 pretraining에서만 사용, pretrain 이후에 task에 맞게 decoder를 붙여 학습



Masked Autoencoders [1]

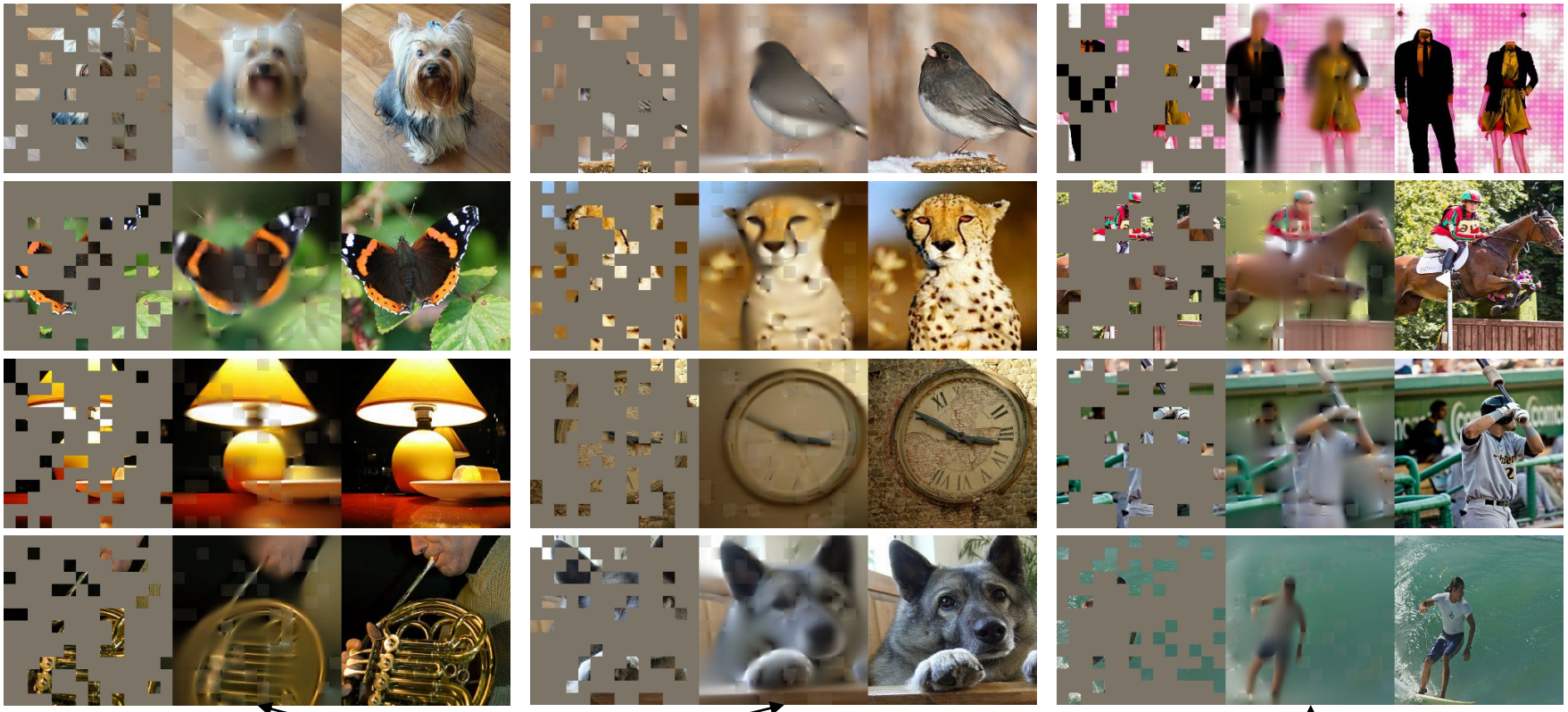
- High Masking Ratio
  - High masking ratio를 사용하기 때문에 단순히 masking된 patch와 인접한 patch를 사용하여 채워넣는 redundancy를 제거할 수 있음
    - High ratio 자체만으로도 좋은 data augmentation이 가능하여 추가적인 어떠한 augmentation 기법을 사용하지 않음
  - Masking된 patch는 encoder에 입력하지 않기 때문에 training time, memory가 크게 감소함 (×3 or more)
  - Language와 다르게 image는 각 patch의 information density가 높지 않음
    - 15%정도의 masking을 수행하는 BERT와 다르게 높은 masking 비율을 사용 가능함





Masked Autoencoders [1]

- Experimental results
  - Pretrained MAE에 validation set reconstruction test
    - 동일하게 75% masking 수행 후 reconstruction을 수행



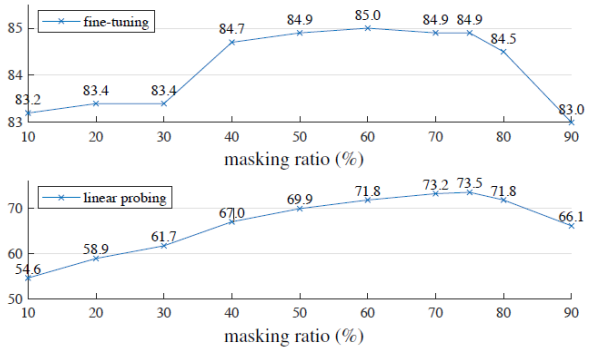
Pretrained on ImageNet-1K,
Tested on ImageNet-1K validation images

Pretrained on ImageNet-1K,
Tested on COCO validation images

Masked Autoencoders [1]

- Experimental results
  - For ImageNet-1K

scratch, original [16]	scratch, our impl.	baseline MAE
76.5	82.5	84.9



  - Finetuing 방법을 사용하면 scratch부터 학습한 결과보다 항상 더 좋음
  - Decoder
    - Finetuing과 Linear probing이 uncorrelate된 경향을 보임
      - Linear probing에서는 decoder의 depth가 중요함
      - 반면 Finetuning에서는 중요하지 않은 것으로 보임
      - 이는 autoencoder의 마지막 몇 개 layer는 recognition이 아닌 reconstruction에 특화되어 있는데, finetuning에서는 이들이 recognition task에 adapt할 수 있는 반면, linear probing에서는 그럴 수 없기 때문인 것으로 보임
    - 8개의 block, 512 dim을 선택하였는데, 이는 ViT-L의 9%인 FLOPs를 갖음

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

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

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

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

Image Inpainting via MAE[1]

- Image inpainting 분야에 MAE를 도입
  - MAE를 통해 pretrain한 prior들을 사용하여 이미지의 masking된 부분을 복원
    - Mask에 인접한 meaningless한 texture 복사가 아닌 prior를 활용하여 mask된 부분 복원
    - 논문에서 주된 비교 대상으로 사용하고 있는 LaMa는 Fast Fourier Convolution을 사용하여 반복되는 패턴에 대해 강점을 갖고 있음
      - ∴ 하지만 detailed prior가 없기 때문에 blurry한 결과 / 반복적이기만 한 inpainting 결과

(a) Masked image (b) MAE (c) LaMa (d) Ours

Image Inpainting via MAE[1]

- Model architecture

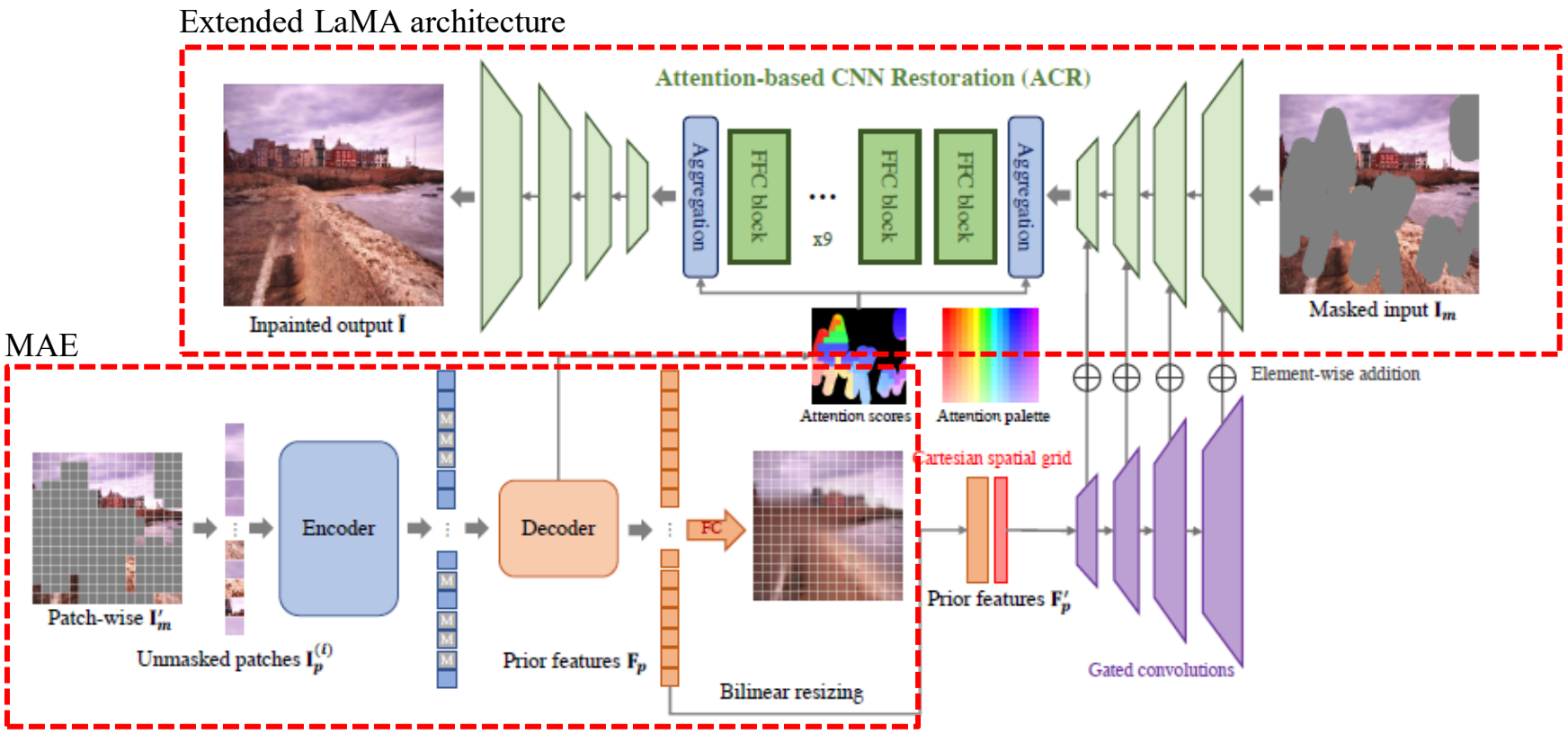
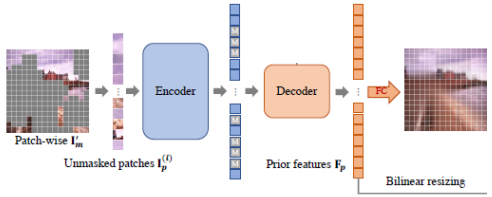


Image Inpainting via MAE[1]



- MAE로 prior 학습
  - 기존 MAE와 다르게 inpainting 분야에서는 mask가 연속적으로 붙어 있음
    - ∴ 따라서 기존 mask와 연속적인 mask를 결합하여 75%의 masking 비율을 만듦

(a) Ground truth image (b) Random masked image (75%) (c) Continues mask (d) Resized continuous mask (e) Hybird masked image (75%)

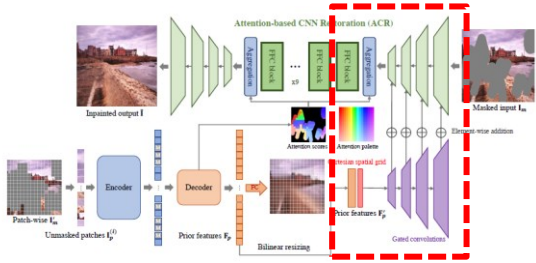
  - MAE는 finetuing을 통해 decoder를 따로 학습하여 task에 적용함
  - 반면, inpainting은 masked 영역에 생성되는 patch의 정보를 필요로 함
    - ∴ 따라서 encoder보다는 decoder에서 feature를 가져와야 함
      - ✓ 따라서 가벼웠던 기존의 decoder를 변경하여 encoder와 균형 있게 decoder를 설정함
      - ✓ Encoder : 12 layers / Deocder : 8 layers
    - ∴ MAE의 linear projection 직전의 feature를 가져옴

Image Inpainting via MAE[1]

- MAE pretraining
  - Places2, FFHQ dataset에 대해 학습
    - MAE 논문에서 ImageNet-1K에 학습된 model을 사용하여 ablation 비교
      - 해당 논문에서는 classification 성능을 위해 patch별로 normalization을 수행하였는데 image inpainting에서는 global normalization을 수행해야 함
    - Pretrained된 Encdoer + Decoder를 사용하고, 마지막 linear projection layer를 추가하여 finetuning을 수행하여 학습하였음

(a) Origin input (b) Masked input (c) MAE w/o ft (d) MAE w/o ft+ACR (e) MAE with ft (d) MAE with ft+ACR

Image Inpainting via MAE[1]



- Attention-based CNN Restoration (ACR)
  - Masked input는 4개의 downsampling layer
  - Prior feature는 4개의 upsampling를 각각 통과하여 feature를 더해 주었음
    - Arbitrary resolution에 대해 robust한 성능을 보장하기 위함
  - 이를 위해 MAE로부터 얻은 prior feature를 1/8배 bilinear interpolation을 수행함
    - 이후 cartesial spatial grid를 더해 continuous position을 embedding하여 주었음
  - 두 feature를 더하기 전에 prior feature의 각 level에 $\alpha_1, \alpha_2 \alpha_3, \alpha_4$를 곱함
    - Trainable parameter로, 0으로 initialize
- Prior Attentions
  - LaMa에 contextual attention을 양 끝에 추가하였음
    - Masked patch와 unmasked patch의 attention score를 곱해 masked patch의 feature를 aggregate하여 더함
      - 하지만 추가적인 성능 향상을 볼 수 없었음
      - 저자는 CNN의 limited capacity를 이유로 보고 있음

Image Inpainting via MAE[1]

- Prior Attentions
  - 따라서, decoder의 각 layer에서 계산된 attention score를 사용함

$$\mathbf{R}^{(l)}_{u,m} = \text{softmax}(\frac{\mathbf{Q}^{(l)}\mathbf{K}^{(l)^T}}{\sqrt{d}} - inf \cdot \mathbf{M}),$$

  - Masked attention score를 구했기 때문에, 모든 attention score는 unmasked region에만 주어짐
  - 8개의 decoder layer에서 나온 8개의 score를 평균을 내어 attention score로 사용함

Masked input | MAE output | MAE attention | Contextual attention | Masked input | MAE output | MAE attention | Contextual attention

MAE mask type	attention type	PSNR↑	SSIM↑	FID↓	LPIPS↓
mixed	no attention	24.34	0.860	26.84	0.117
mixed	trainable CA	24.13	0.859	26.99	0.123
random	prior attention	24.39	0.861	26.25	0.117
mixed	prior attention	**24.51**	**0.864**	**25.49**	**0.113**

Image Inpainting via MAE[1]

- Experimental results

(A) 256x256 FFHQ

(a) Masked input (b) Co-Mod (c) LaMa (d) Ours

(B) 1024x1024 results

(a) Masked input (b) MAE output (c) LaMa (d) Ours

	FFHQ			Places2				
	Co.	La.	Ours	EC	Co.	La.	CT.	Ours
P.↑	25.2	26.6	**26.8**	23.3	22.5	24.3	23.4	**24.5**
S.↑	.889	.903	**.906**	.839	.843	.869	.835	**.871**
F.↓	**5.85**	6.38	6.15	6.21	1.49	1.63	11.2	**1.31**
L.↓	.085	.078	**.074**	.149	.246	.155	.185	**.101**

< 256x256 FFHQ, Places2 >

	P.↑	S.↑	F.↓	L.↓
Hi.	20.1	.764	65.4	.291
Co.	22.0	.843	30.0	.166
La.	24.1	.877	27.8	.149
Ours	**24.3**	**.880**	**25.3**	**.119**

< 512x512 Places2>

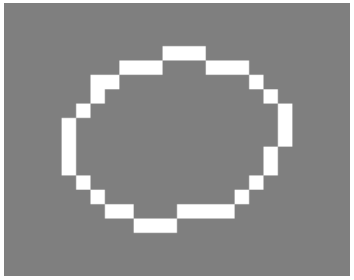
Image Inpainting via MAE[1]

- Experimental results

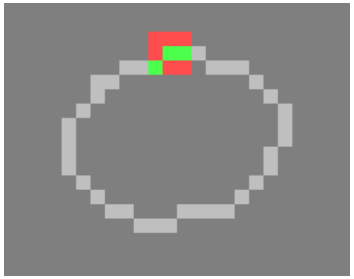
(a) Masked input (b) HiFill (c) Co-Mod (d) LaMa (e) Ours (f) MAE

ConvNeXt V2

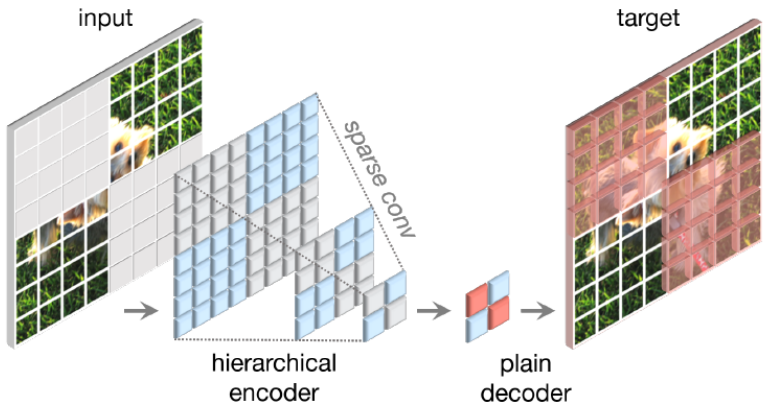
- ConvNeXt + MAE
  - 간단하게 앞서 설명된 내용을 융합하여 학습에 사용
    - ViT와 다르게 Convolution에 masking을 수행하는것은 매우 까다로움
    - 저자는 3D Pointcloud에서 영감을 받아 submainfold sparse convolution으로 이를 극복
      - Mask에 해당하는 부분만 convolution연산에 사용함
    - Masking은 downsampling의 마지막 단계에서 60% pixel에 대해 수행
      - Fine tuning 단계에서는 original conv로 대체하여 학습



< Normal Conv >



< Submainfold Sparse Convolution >



ConvNeXt V2

- TPU, AI accelerator friendly method
  - Sparse Conv 대신 dense convolution 수행 이전, 이후에 binary mask를 곱함
    - 동일한 연산을 하면서 특정 기기에서 빠르게 수행할 수 있는 방법 또한 제안함

- Decoder design
  - 실험적으로 가볍고 단순한 decoder 설계가 finetuning하였을 때 좋은 결과를 보임

dec. type	ft	hours	speedup
UNet w/ skip	**83.7**	12.9	-
UNet w/o skip	83.5	12.9	-
Transformer [31]	83.4	8.5	1.5×
ConvNeXt block	**83.7**	**7.7**	**1.7**×

(a) **Decoder design**. A simple convolutional block outperforms more complex decoder designs.

blocks	ft
1	**83.7**
2	83.5
4	**83.7**
8	83.6
12	83.3

(b) **Decoder depth**. A single block yields competitive fine-tuning performance.

dim	ft
128	83.5
256	**83.7**
512	**83.7**
768	83.6
1024	83.5

(c) **Decoder width**. A decoder width of 256 or 512 achieves the best performance.

- Experimental Results

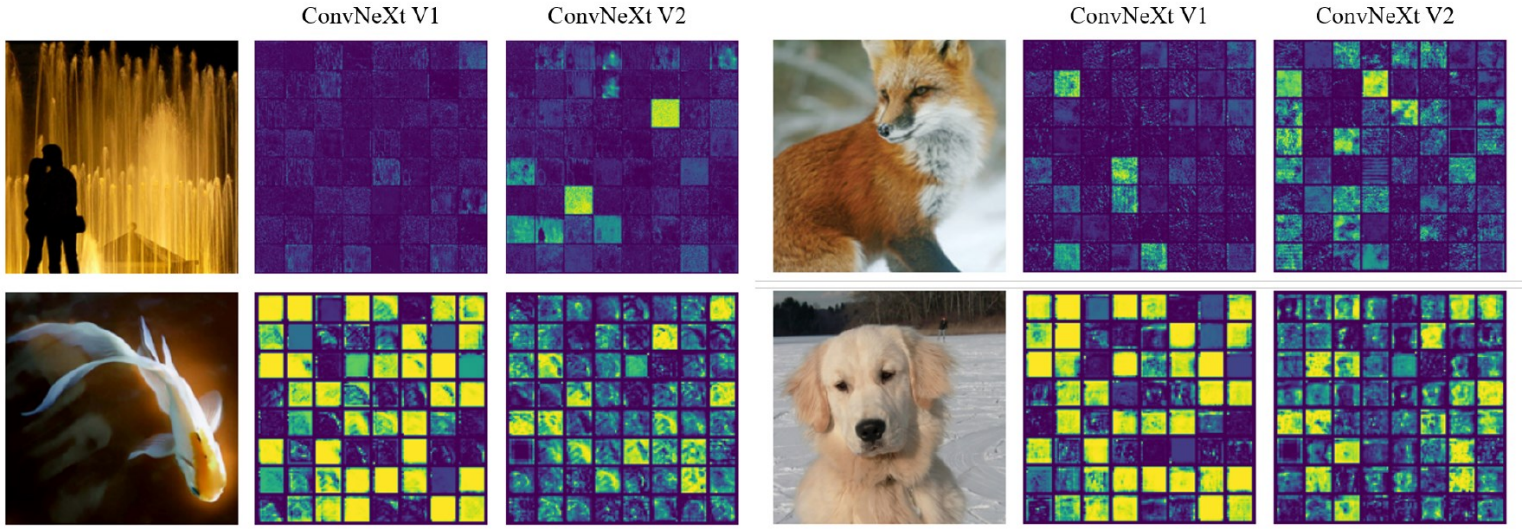
w/o Sparse conv.	w/ Sparse conv.
79.3	83.7

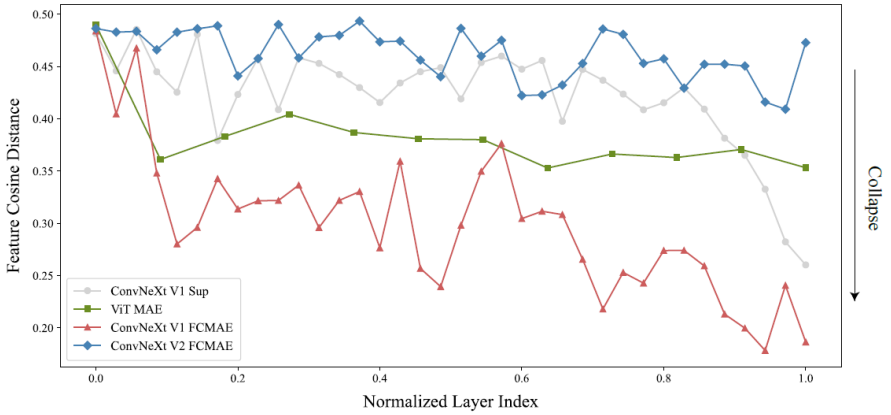
ConvNeXt V1

Sup, 100ep	Sup, 300ep. [52]	FCMAE
82.7	83.8	83.7

ConvNeXt V2

- Global Response Normalization (GRN)
  - ConvNeXt V1의 feature activation map 을 visualize한 결과, collapse가 쉽게 일어남





ConvNeXt V2

- Global Response Normalization (GRN)
  - 뇌의 뉴런에서 영감을 얻어, 이웃에 있는 세포를 억제하는 lateral inhibition을 수행
  - 총 3 단계로 구성됨
    - Global feature Aggregation
      - 각 채널의 feature를 pooling하여 scalar화함
      - Pooling 방식은 실험 결과 L2 norm을 사용함

case	ft
g.avg.	83.7
L1	84.3
L2	**84.6**

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

    - Feature normalization
      - Pooling된 scalar 값들에 normalization을 수행
      - Normalization은 실험 결과 divisive normalization을 사용
      - 상호 억제를 통해 채널 간에 feature를 pooling한 scalar에 대해 경쟁을 발생시킴

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

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

    - Feature Calibration
      - 계산된 normalization score를 input response에 곱함

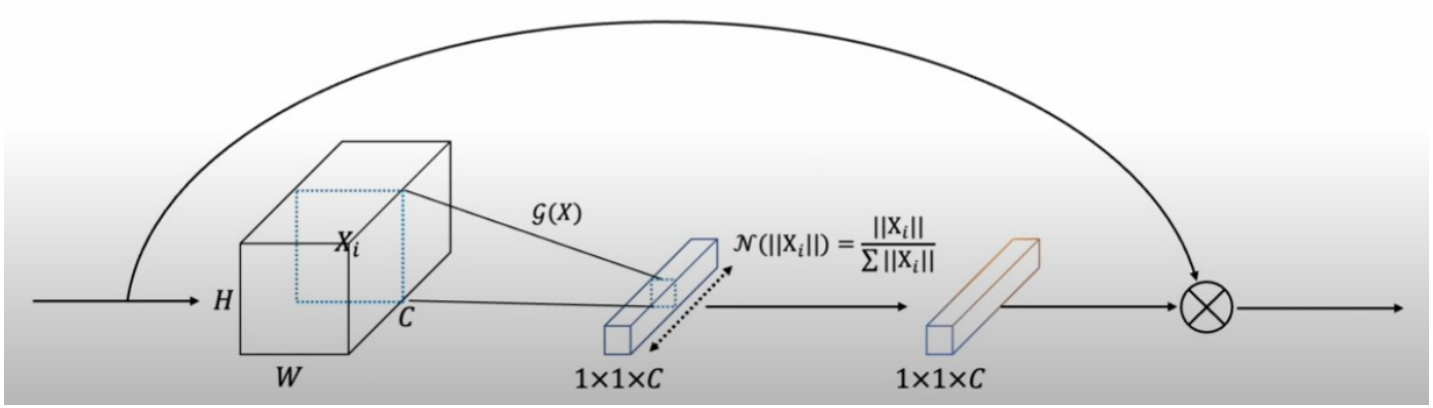
$$X_i = X_i * \mathcal{N}(\mathcal{G}(X)_i) \in \mathcal{R}^{H \times W}$$

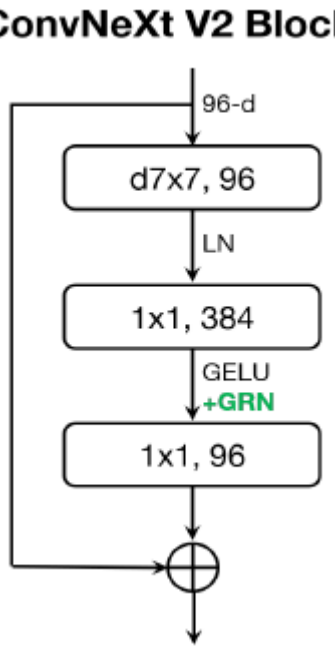
$$\gamma * X_i * \mathcal{N}(\mathcal{G}(X)_i) + \beta + X_i.$$

        - ✓ 최적화를 위해 learnable parameter를 추가하고 0으로 초기화
        - ✓ Residual Conncetion 추가

ConvNeXt V2

- Global Response Normalization (GRN)





- Experimental Results

case	ft
Baseline	83.7
LRN [45]	83.2
BN [41]	80.5
LN [2]	83.8
GRN	**84.6**

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

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

ConvNeXt V2

- Experimental Results

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

  - Base model : pretrain 800 epochs + finetuning 100 epochs

감사합니다