20기 정규세션

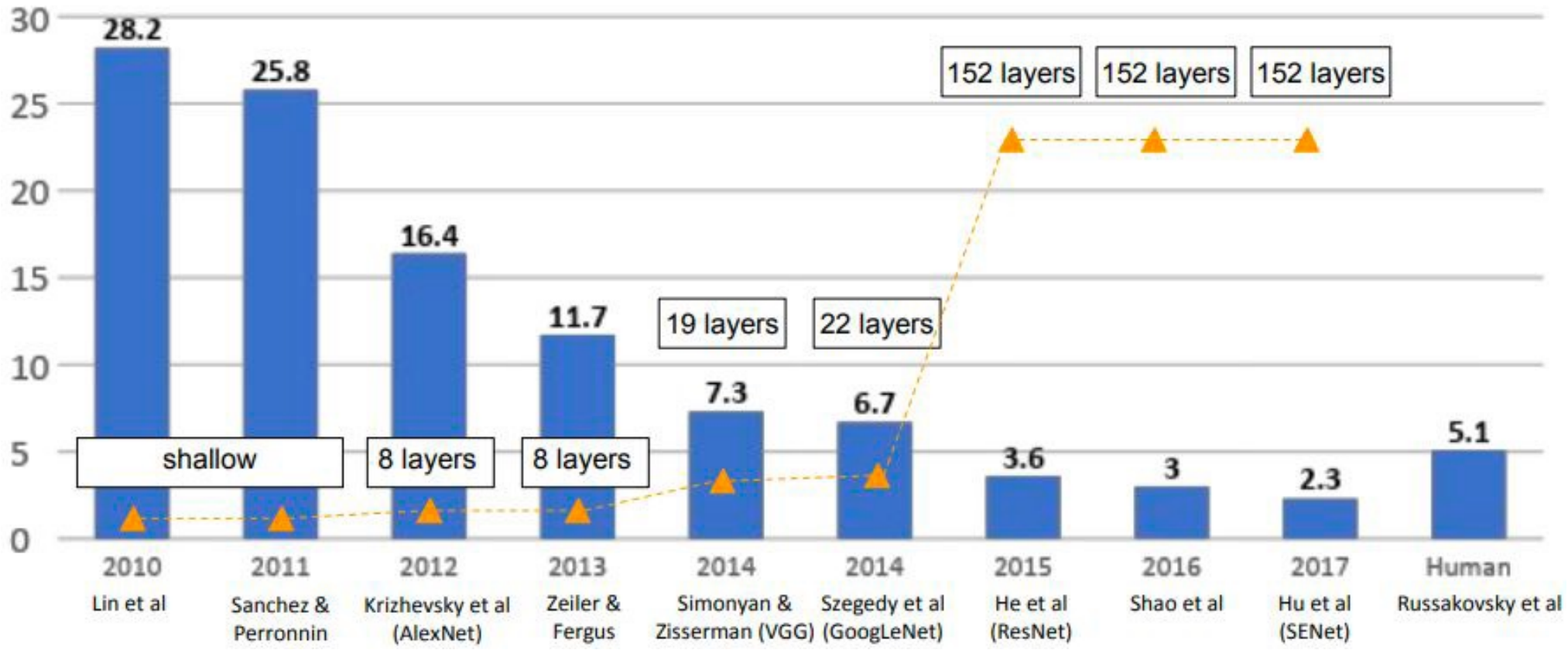
ToBig's 19기 강의자

임서영

VISION ADVANCED

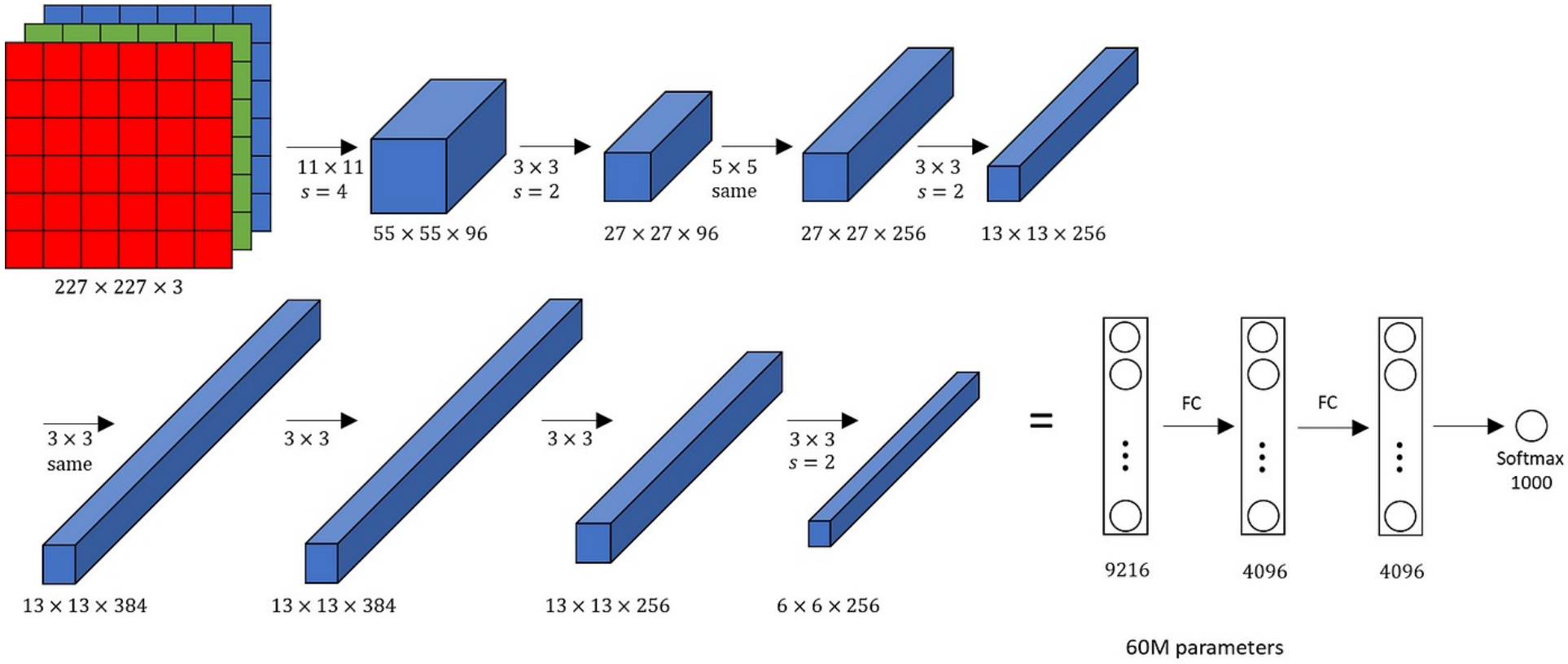
Contents

19기
임서영


30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.6
3
2.3
5.1
shallow
8 layers
8 layers
19 layers
22 layers
152 layers
152 layers
152 layers
2010
Lin et al
2011
Sanchez & Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler & Fergus
2014
Simonyan & Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
2016
Shao et al
2017
Hu et al (SENet)
Human
Russakovsky et al

19기
임서영

1) AlexNet



1) AlexNet

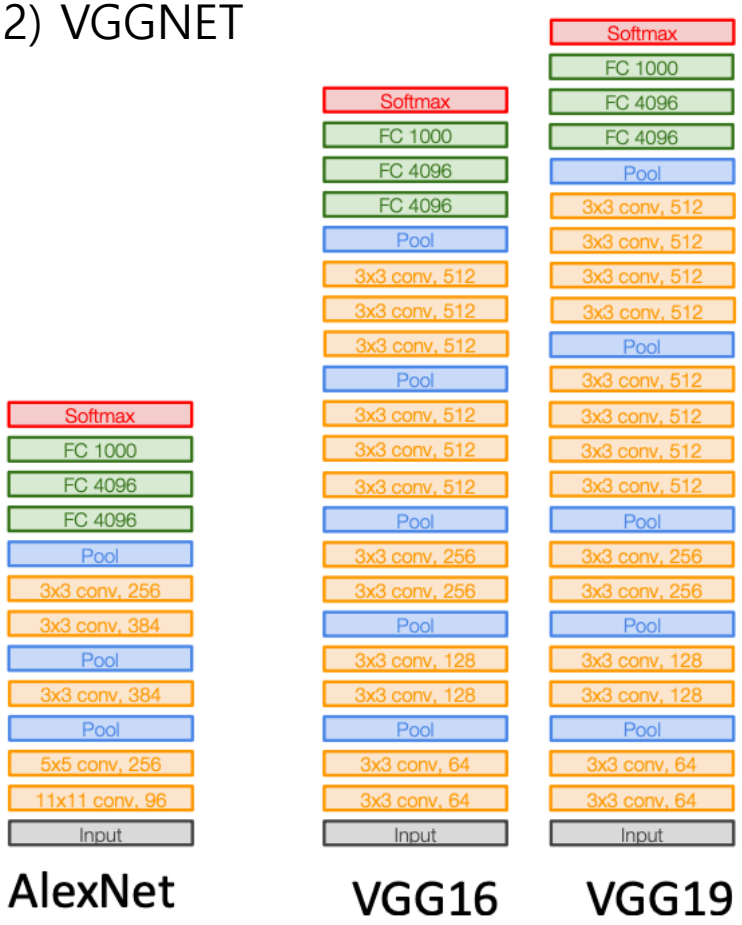
Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Input Image – gray scaled

1) ReLU 사용함
2) Dropout – overfitting 방지
3) Size 줄이기위해 Overlap Pooling 사용함

19기
임서영

2) VGGNET



👉 목적: CNN의 깊이가 image classification 에

어떤 영향을 끼치는지 알아내기 위함

👉 모든 합성곱 연산은

Filter 3x3, Padding 2, Stride 1,

Max Pooling 2 x 2 픽셀씩

👉 산출값의 높이와 넓이는 각 max Pooling마다

1/2씩 줄어들며, Channel의 수는 두 배 혹은 세 배로

늘어난다. (간결함, 수학적으로 깔끔)

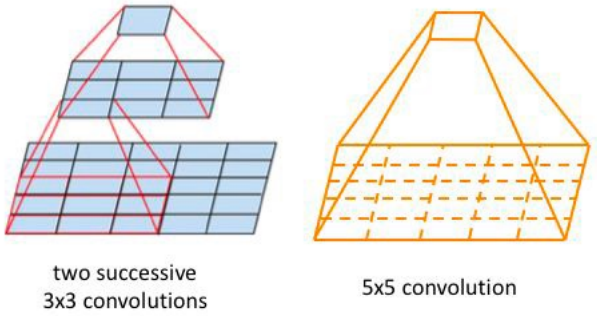
👉 훈련시킬 변수의 개수가 많아

네트워크의 크기가 커진다는 단점

19기
임서영

2) VGGNET 특징

3x3 CONV를 사용하는 이유는?



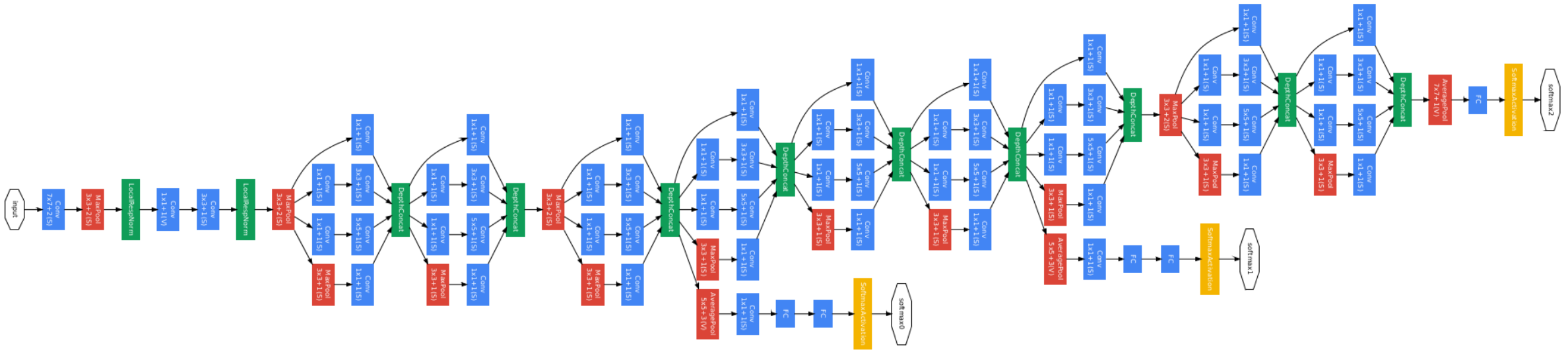
With 1 layer of 7×7 filter 1×(7×7)×C2 = 49C2

With 3 layers of 3×3 filter 3×(3×3)×C2 = 27C2
(where C is the number of channels per layer)

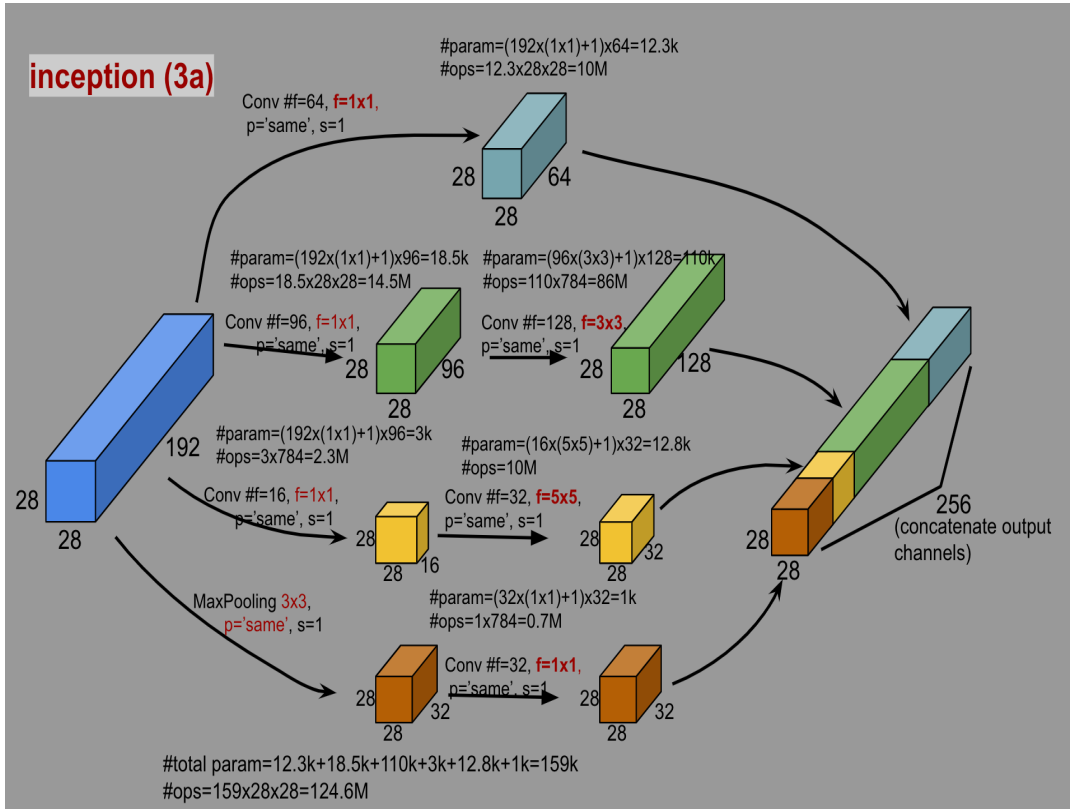
더 non-linearity를 잡을 수 있음!

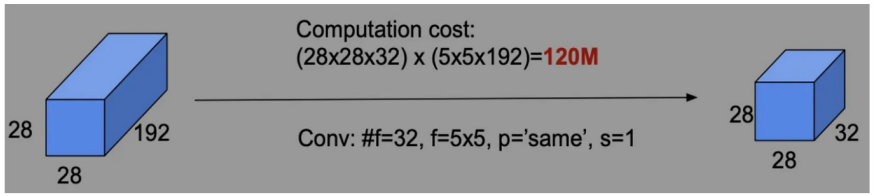
19기
임서영

3) GOOgLeNET

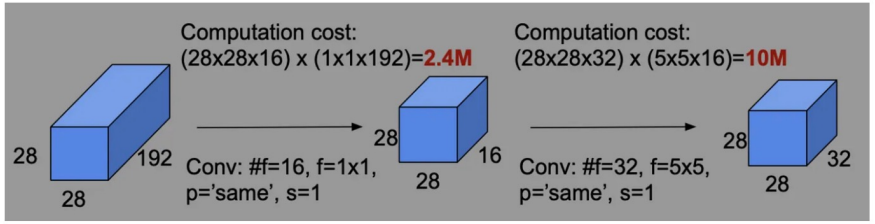


3) GOOgLeNET





Without bottleneck layer, total times of multiplications = 120M

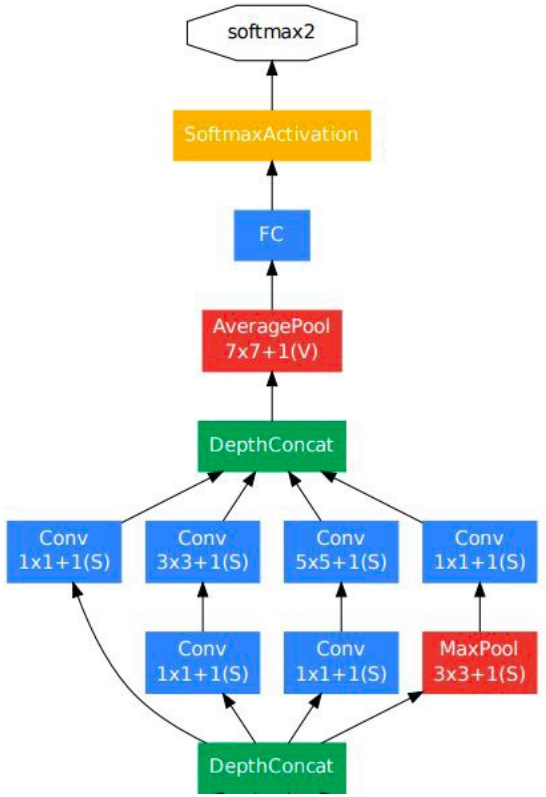


With bottleneck layer, total times of multiplications = 2.4M + 10M = 12.4M
(around 10% of above)

Reference : https://medium.com/@rockyxu399/paper-review-and-model-architecture-for-cnn-classification-94972e40d96a

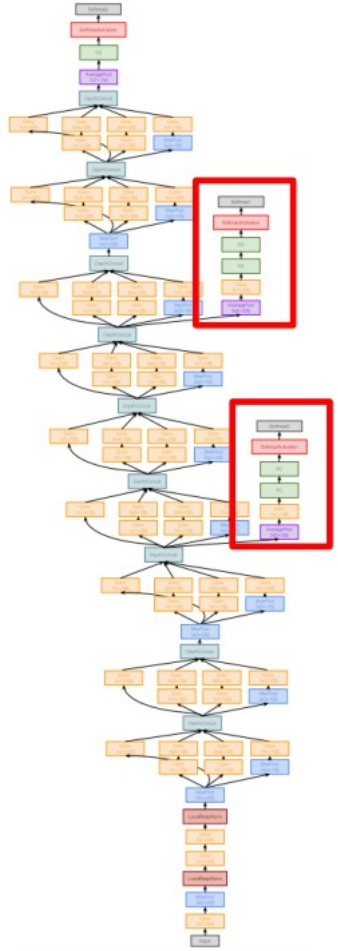
3) GOOgLeNET

- 더이상 FC Layer 가 필요없어짐 (마지막 제외)
- 아키텍쳐가 sparser connections 생김



3) GOOgLeNET

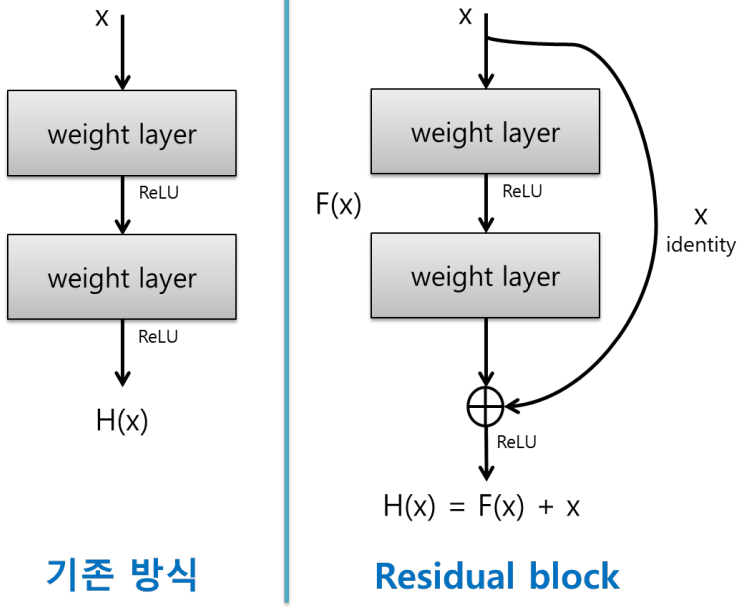
네트워크가 너무 깊어서 gradients가 잘 전달안된다 : “auxiliar
y classifers”달아서 해결



Auxiliary classifier를 사용할때 주의할 점:

Backpropagation시, weight값에 큰 영향을 주는 것을 막기 위해 auxiliary classifier에 0.3을 곱함.
맨마지막 softmax(classifier)는 따로 value를 곱해주지 않으면서 사용.
= Auxiliary classifier는 training시 발생하는 *문제인 vanishing gradient 문제를 해결하기 위해 사용하는 technique*이므로 **inference**하는 과정에서는 중간에 2개의 auxiliary classifier를 모두 제거 후, 제일 마지막 layer의 softmax만을 사용.

4) ResNet

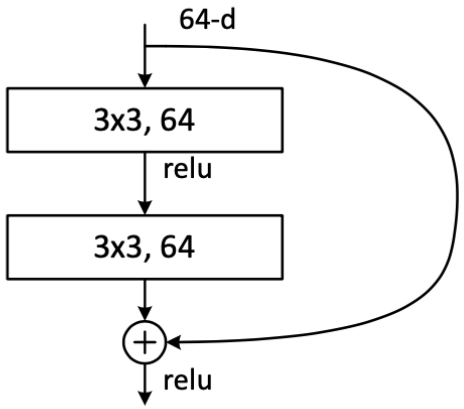


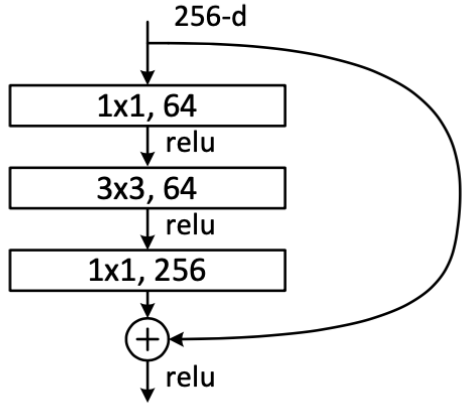
기존 방식

Residual block

더 깊은 layer에서 학습 방식!

Model Cost 때문!





4) ResNet

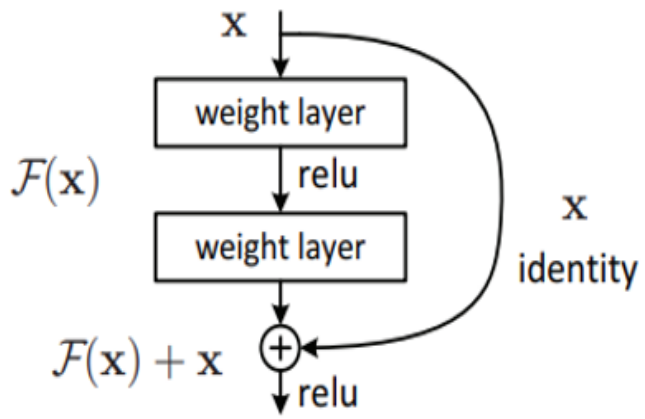
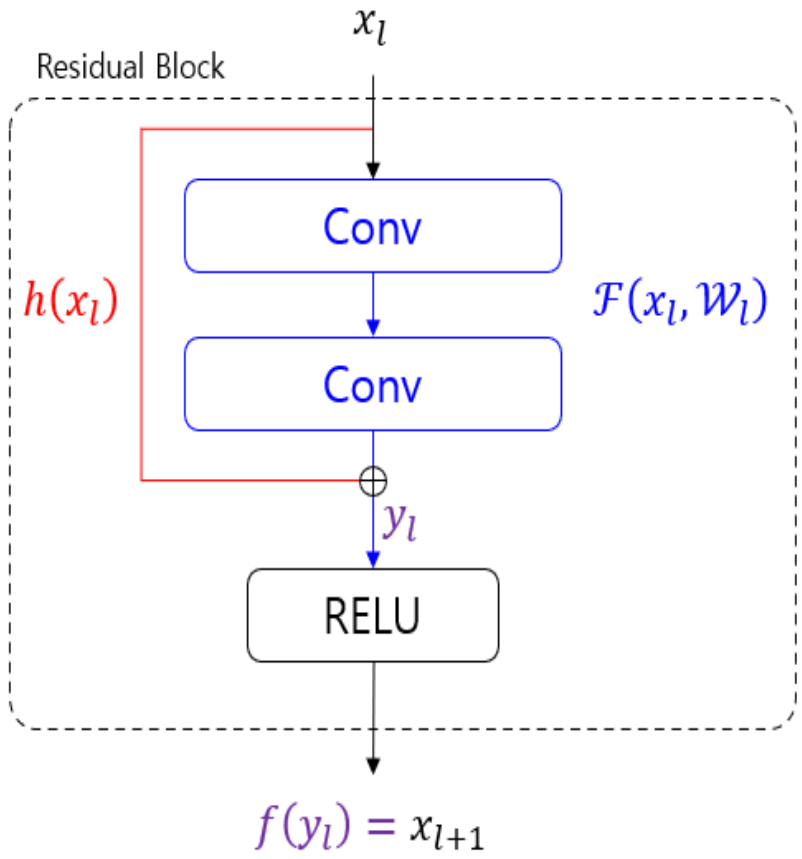
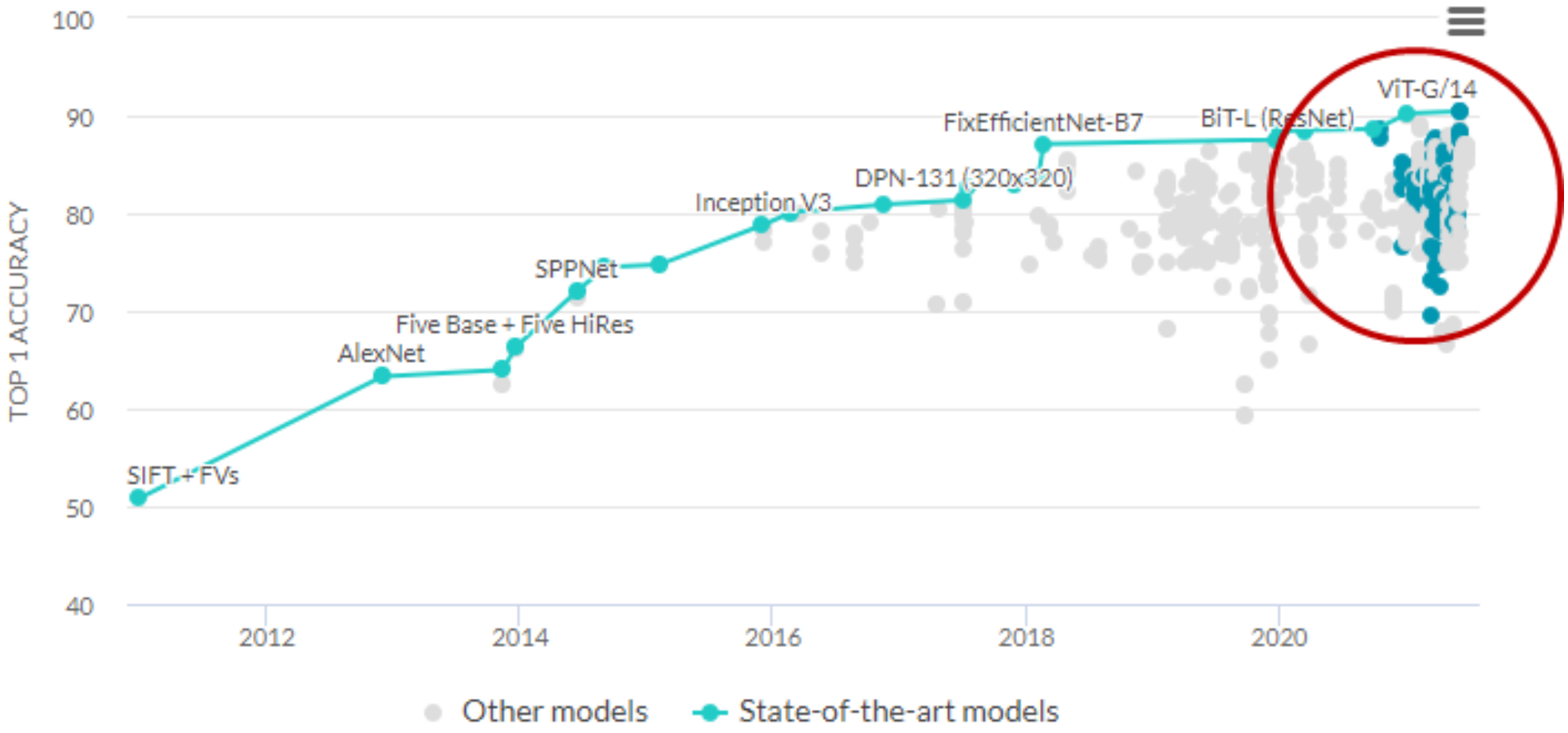


Figure 2. Residual learning: a building block.



$$y_l = \underbrace{h(x_l)}_{\text{Skip-connection}} + \underbrace{\mathcal{F}(x_l, \mathcal{W}_l)}_{\text{Convolutional layer}}$$

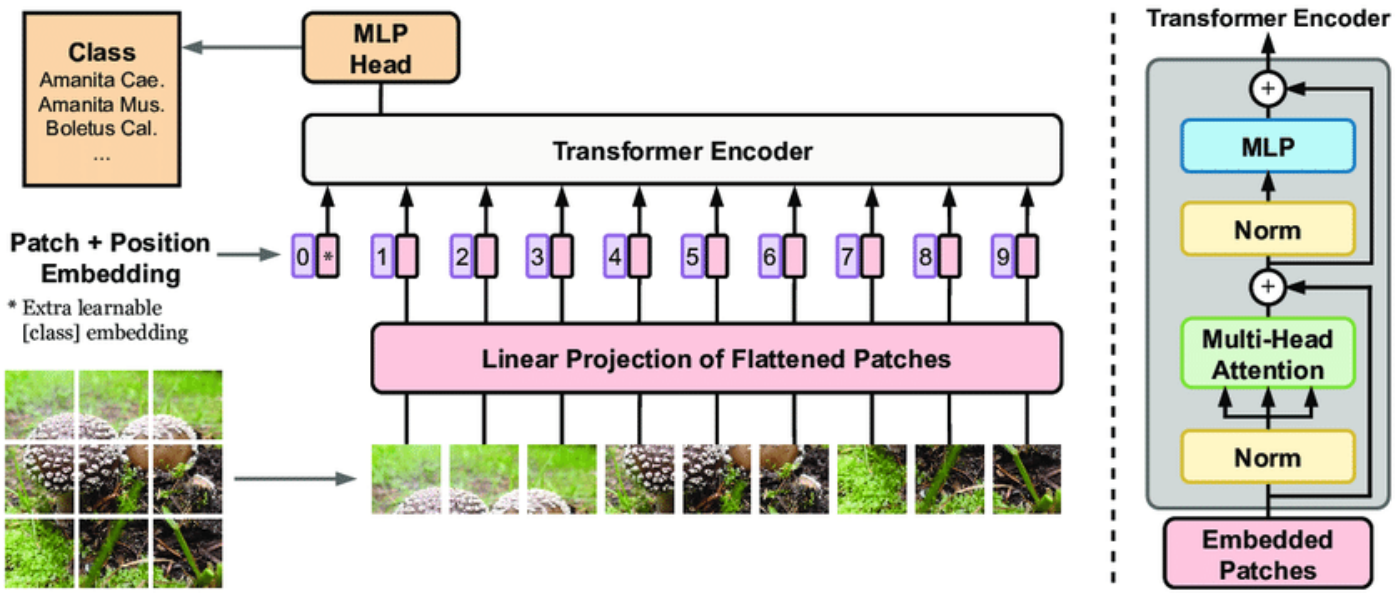
$$x_{l+1} = \underbrace{f(y_l)}_{\text{RELU}}$$

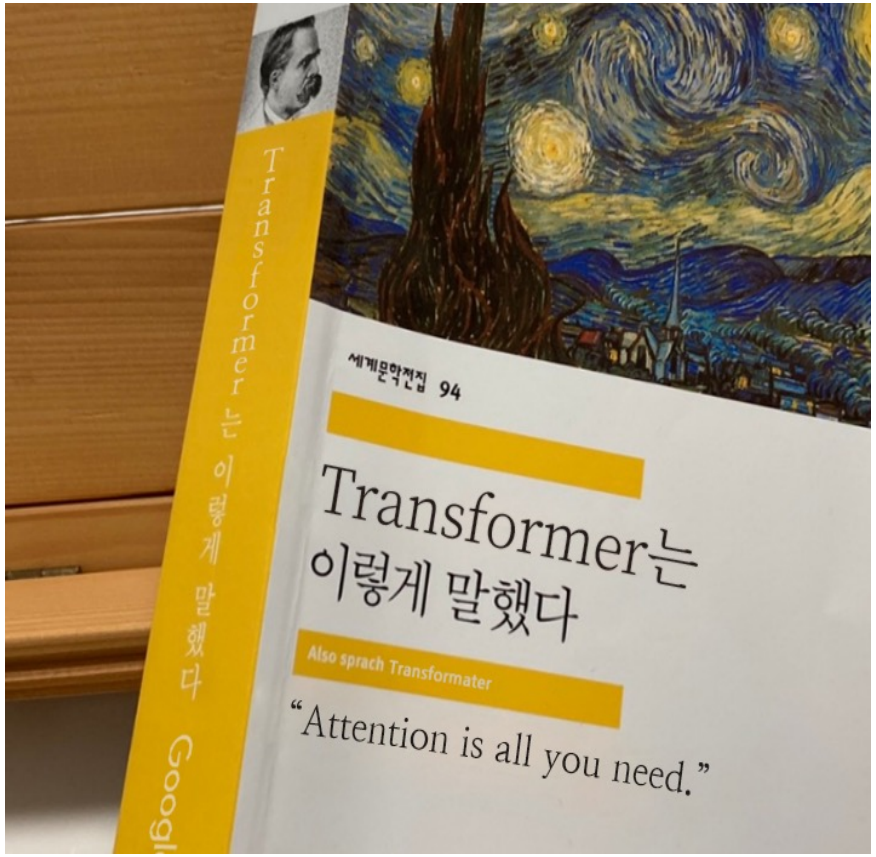

100
90
80
70
60
50
40
TOP 1 ACCURACY
2012
2014
2016
2018
2020
SIFT + FVs
AlexNet
Five Base + Five HiRes
SPPNet
Inception V3
DPN-131 (320x320)
FixEfficientNet-B7
BiT-L (ResNet)
ViT-G/14
Other models
State-of-the-art models

SOTA

19기
임서영

Model Architecture





출처 : https://blog.promedius.ai/transformer/

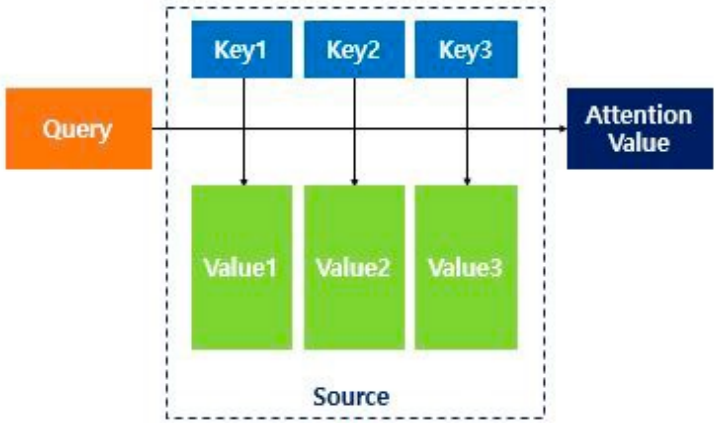
VIT : MAIN IDEA

Self-attention : Each element learns to refine its own representation by attnending its context

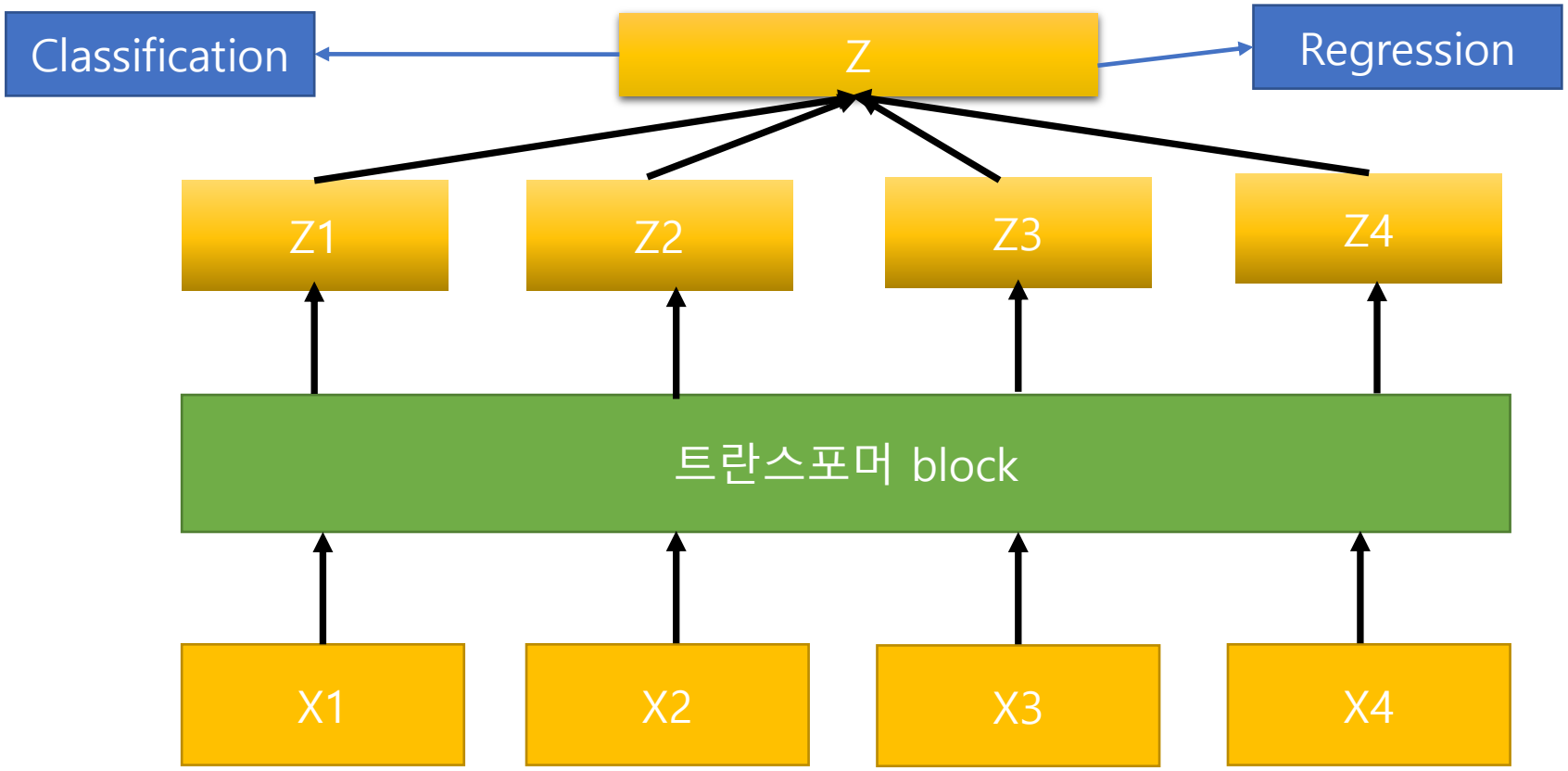
Query , Key, Value 의 개념을 이해해야함

(1) Query : Context (맥락, 주변)
(2) Key-Value : Reference (참고자료)
(3) Attention value is the weighted average of values (합쳐진것)

주의 : Query 와 Key 는 dimension 이 같아야함!
Value 와 Attention Value역시 dimension이 같아야함!



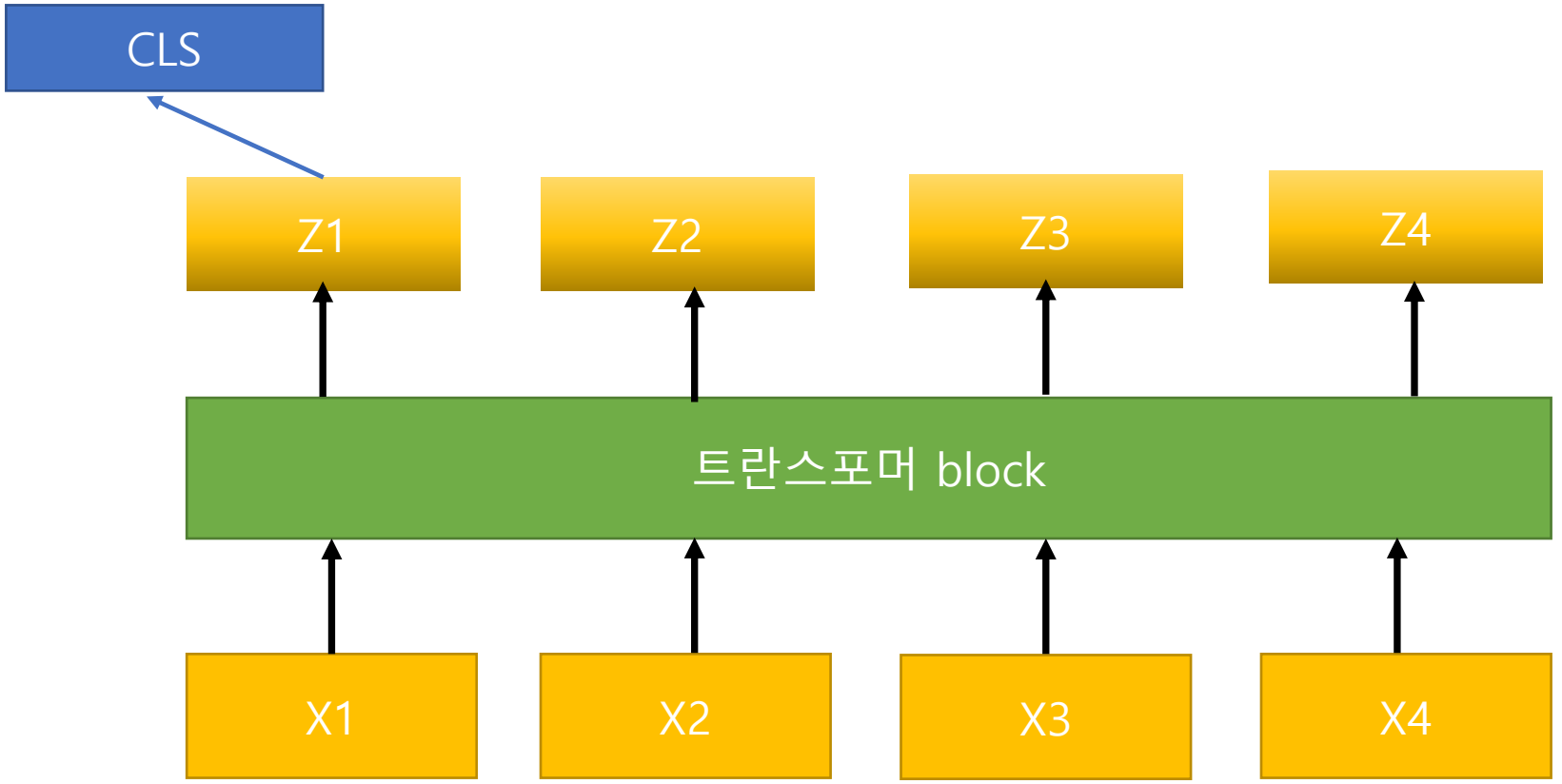
19기
임서영



Token Aggregration

1) 평균을 취하는방법

19기
임서영



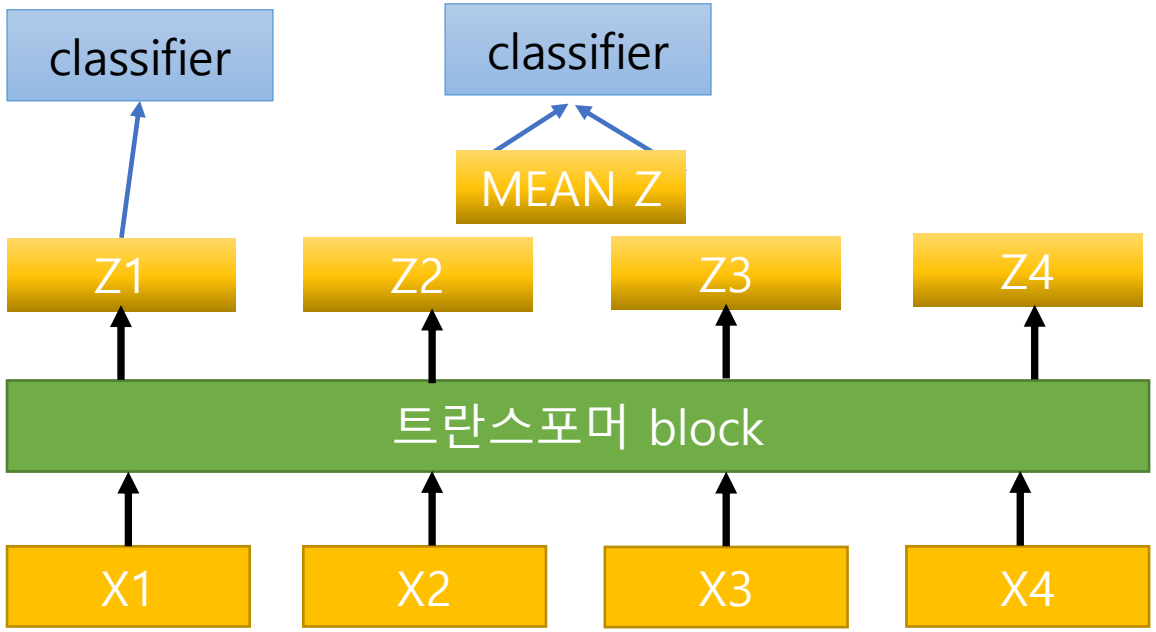
Token Aggregration

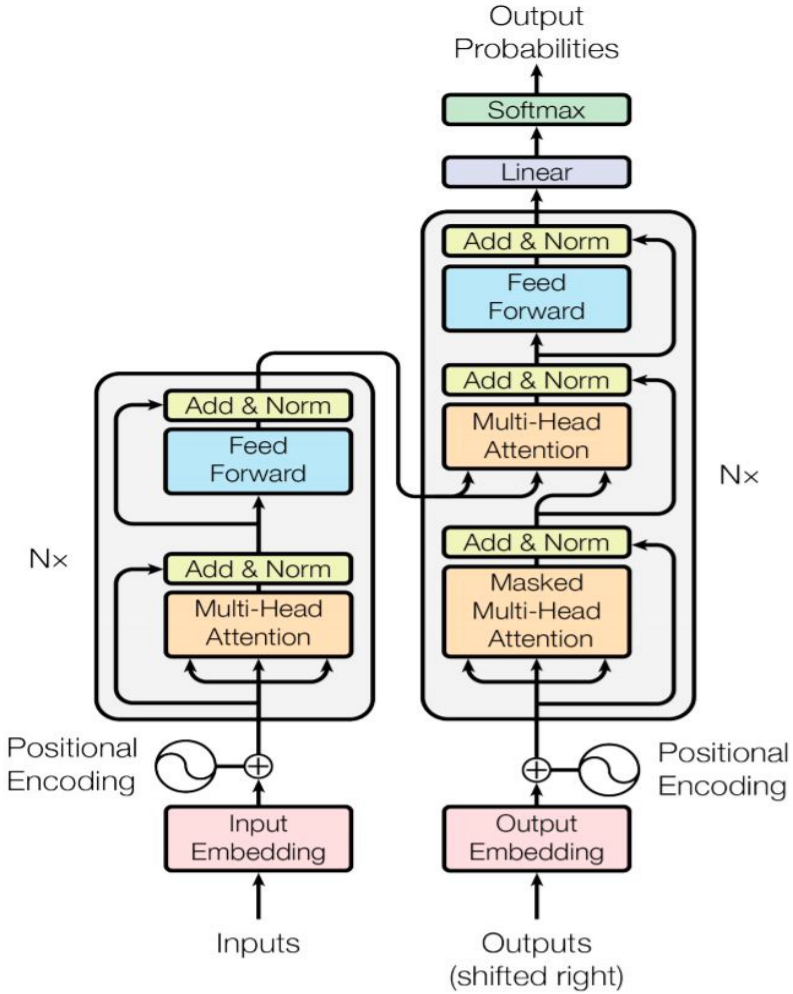
1) 맨 처음 가장 많이 학습된 처음 input z 를 CLS 토큰으로 삼아 classification / regression 할 수 있음

모델이 학습하는것

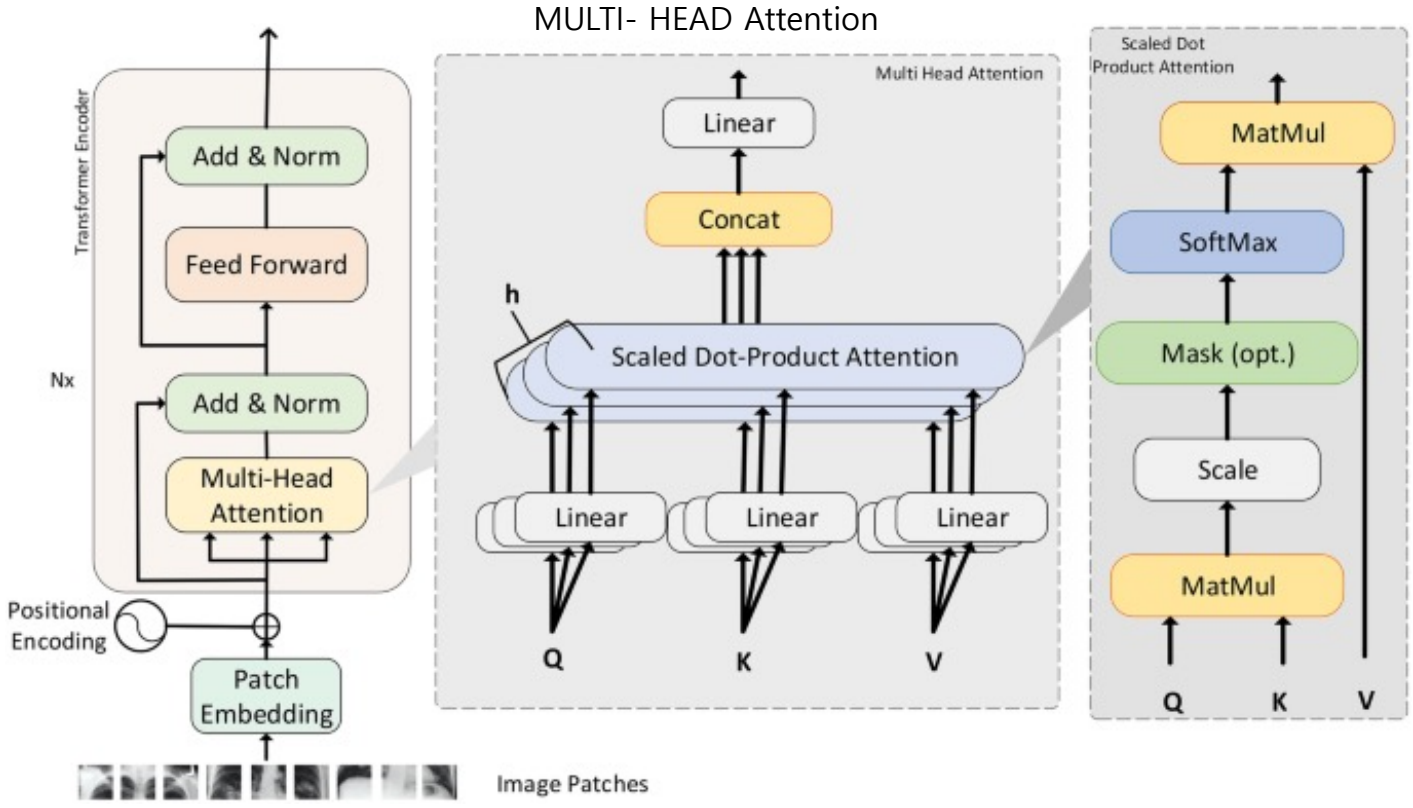
Token-level prediction

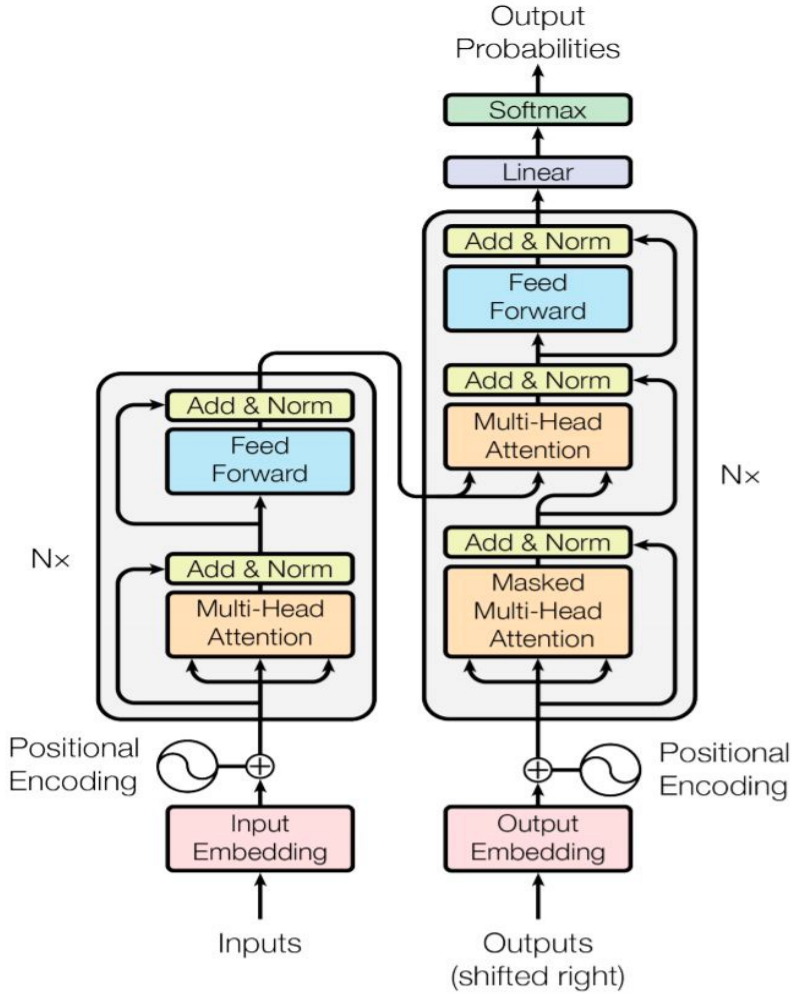
Sequence level prediction





Transformer (ENCODER)

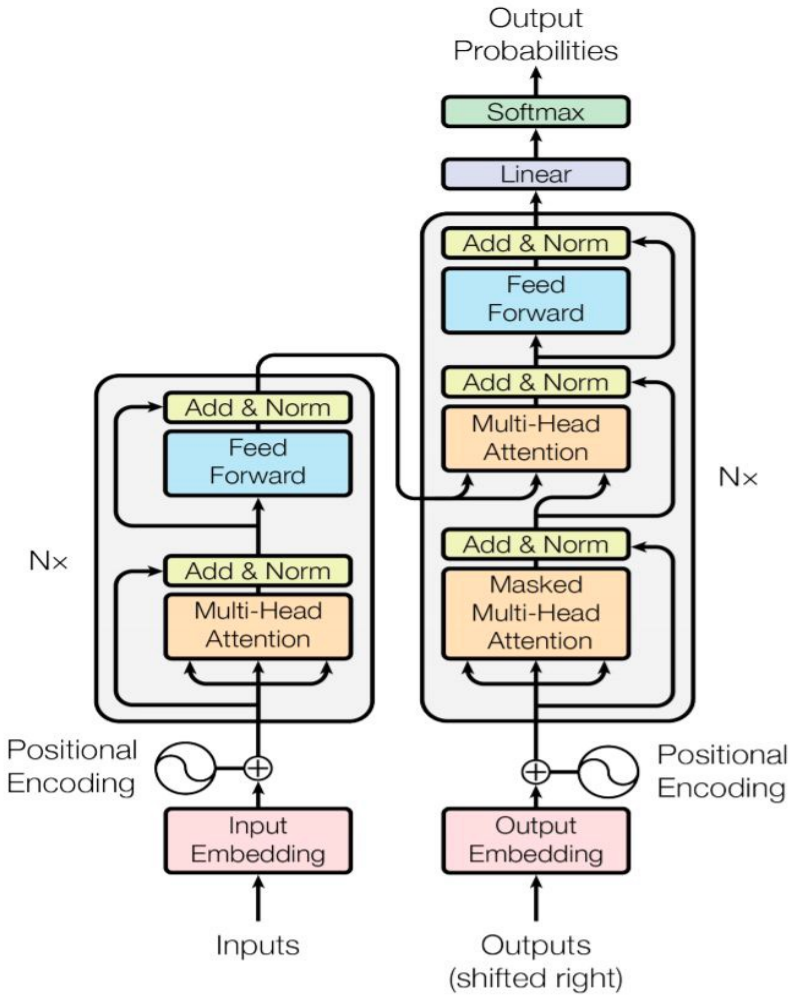




TRANSFORMER – FEED FORWARD

Each contextualized embedding goes through an addictional FC layer

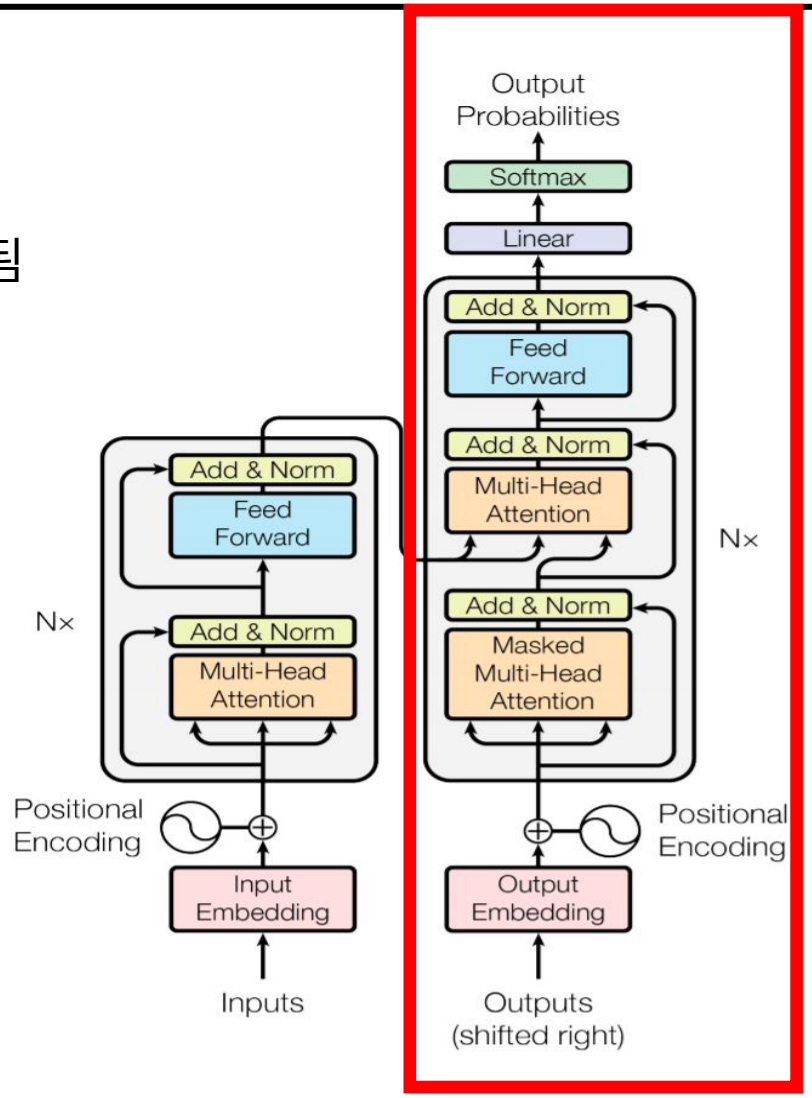
$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$



Transformer - Decoder

1) 아까 변형된 Z 벡터들이 (encoder output) , 순차적으로 들어가게됨

2) 처음 Masked self – attention에서는 그전에 있었던 것을 학습할 수 없으니까 masked out

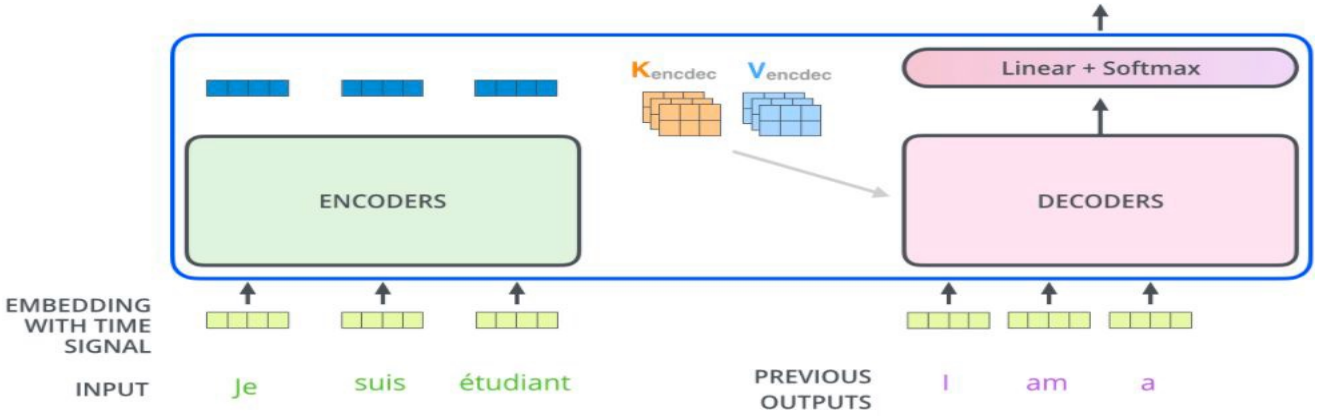


MULTI-head attention

1) 두번째 multi – head attention은 encoder를 참고하여 학습

Query? : decoder 의 쿼리가됨

Key , value : encoder에서 가져옴



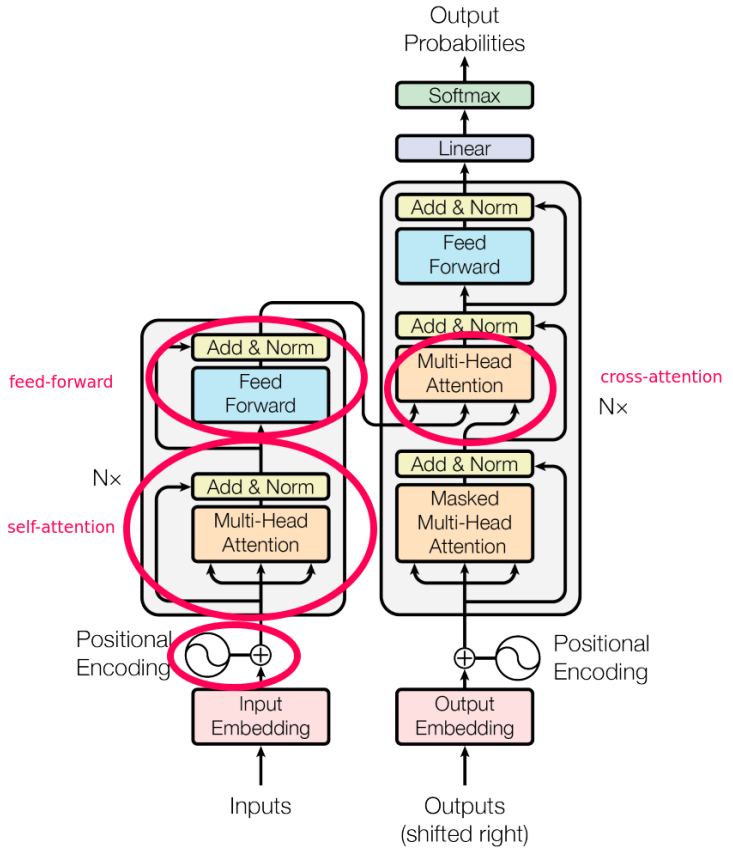


Figure 1: The Transformer - model architecture.

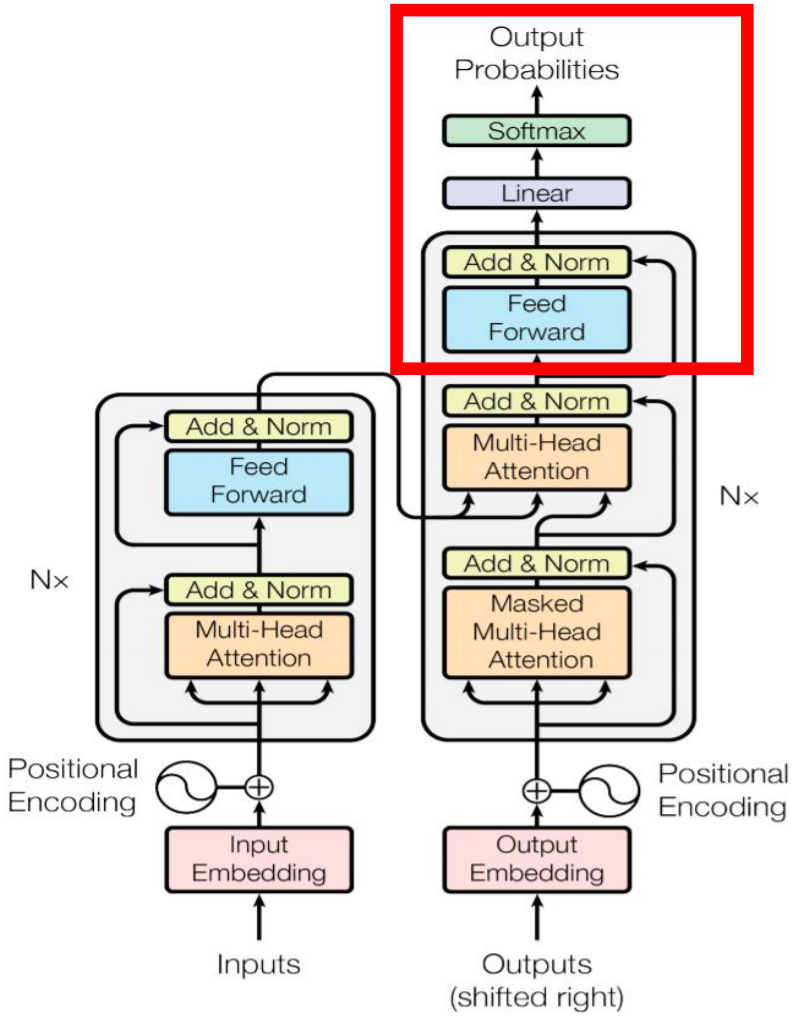
19기
임서영

1) ENCODER랑 똑같이 feed – forward layer

2) Linear Layer

3) Class scores 계산

3) soft max를 활용해 probability로 변환



ViT outperforms previous SOTA (ResNet152)

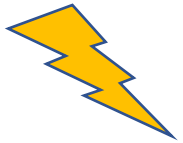
	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Swin Transformer가 해결하고 싶었던것

1) Lack of inductive bias 를 해결하고 싶음
2) 고정된 patches

VIT 의 연산량

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k



TPU v3 costs $8.00 per hour.
- $8 × 24 hr/day × 2500 day =

$480,000 to train this model once!

19기
임서영

Overall Architecture

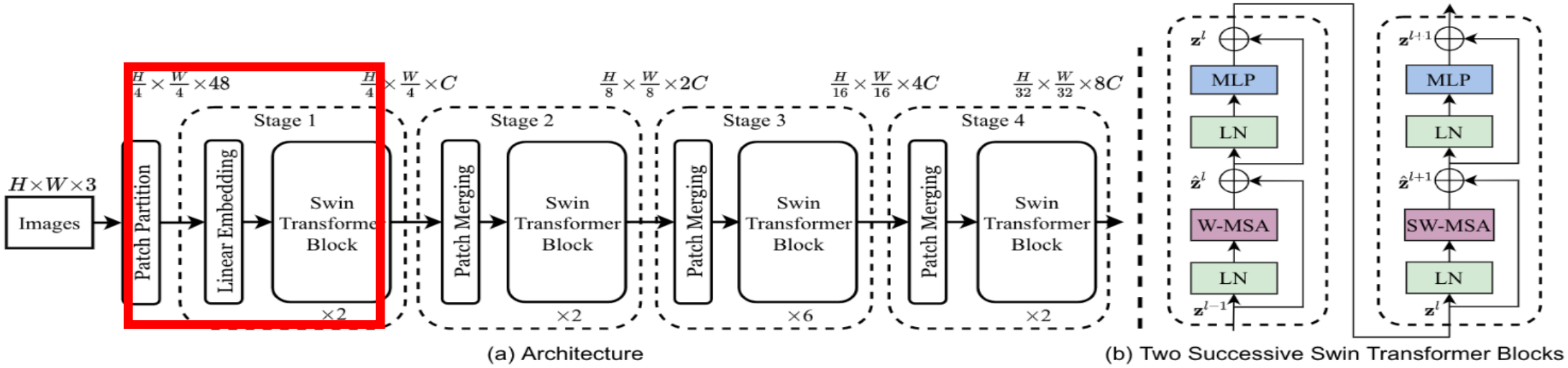
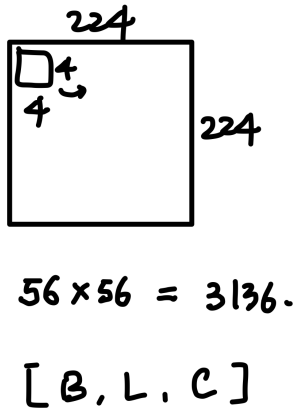


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

이미지를 자를때, 각각 패치들을 토큰으로 다룸



VIT와 비슷하게 convolution의 kernel과 stride를 같게 하여 patch를 뽑기!.

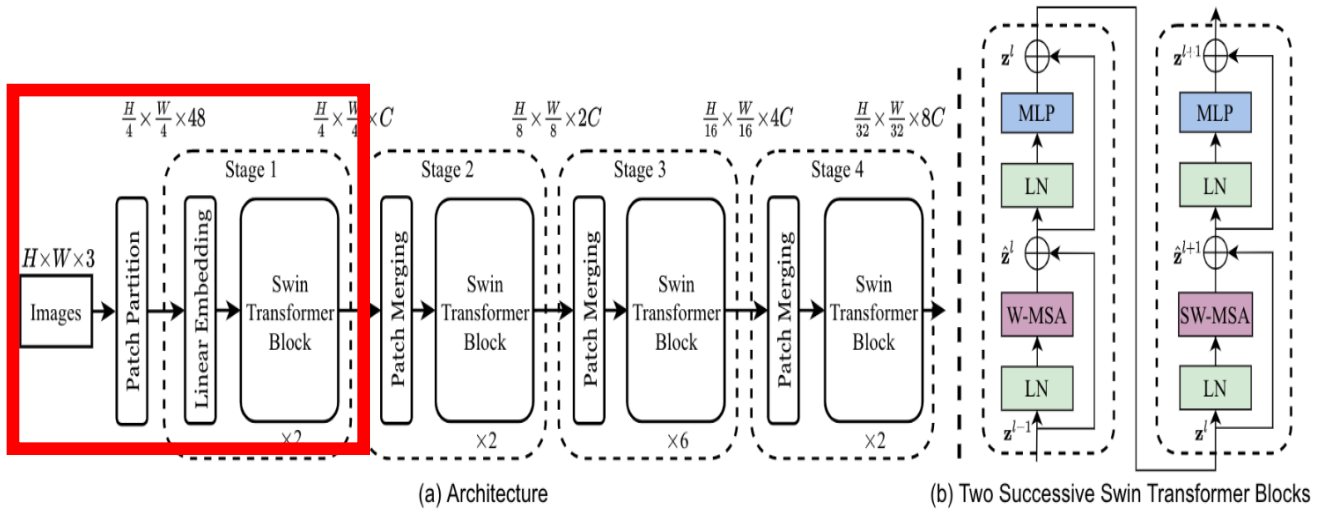
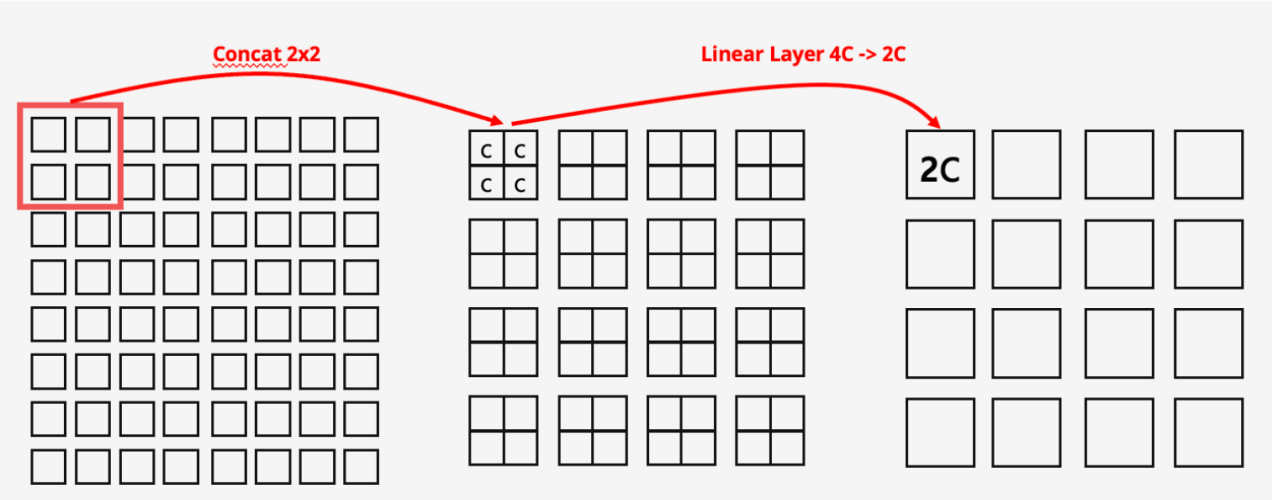
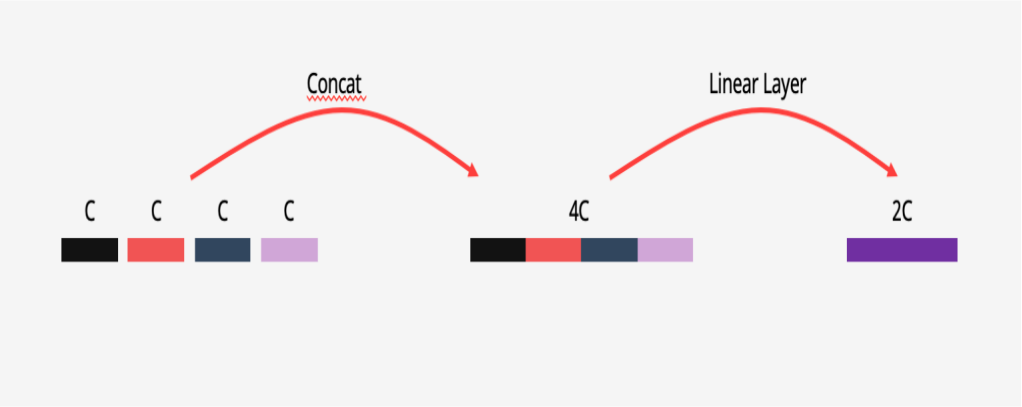


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

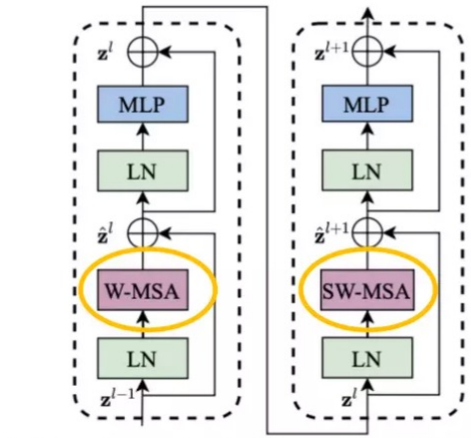
19기
임서영

VIT 극복 1 : Inductive bias is reintroduced by letting each query token refers only to nearby tokens within Window, instead of the entire image



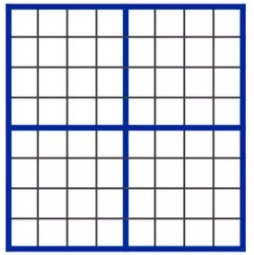


19기
임서영

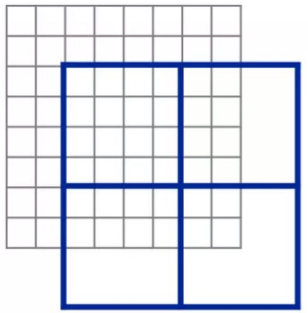


(b) Two Successive Swin Transformer Blocks

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

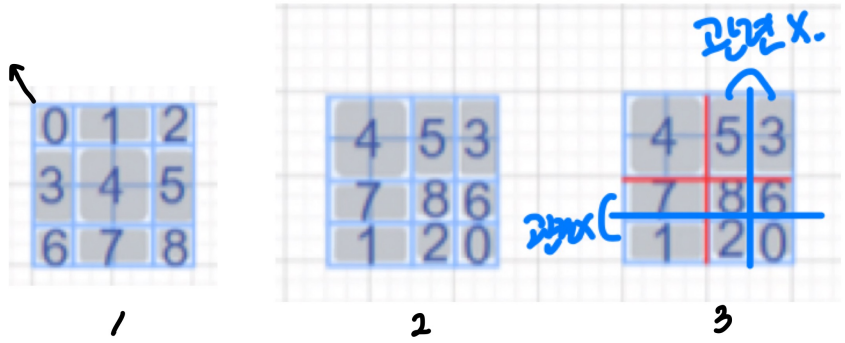


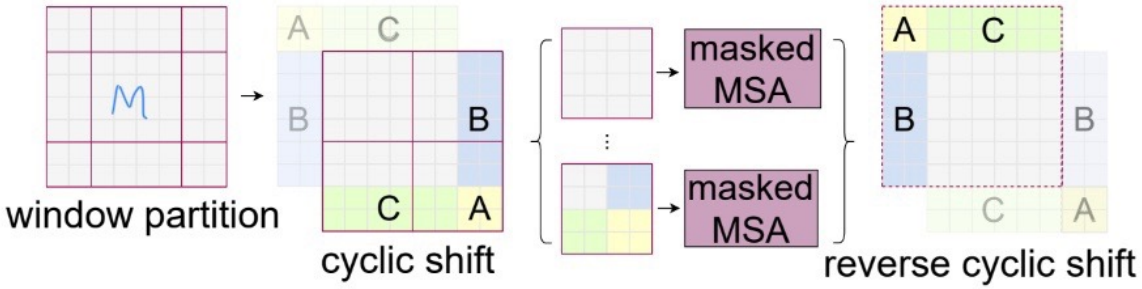
W-MSA
The input image is divided into four windows and compute attention for patches within the window.



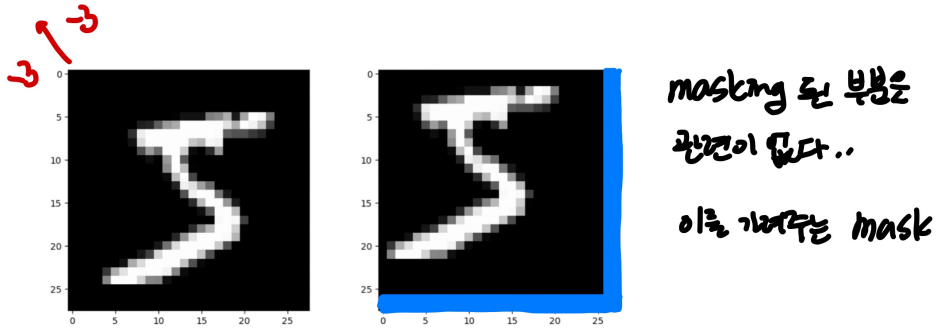
SW-MSA
The approach introduces connections between neighboring non-overlapping windows in the previous layer.

1

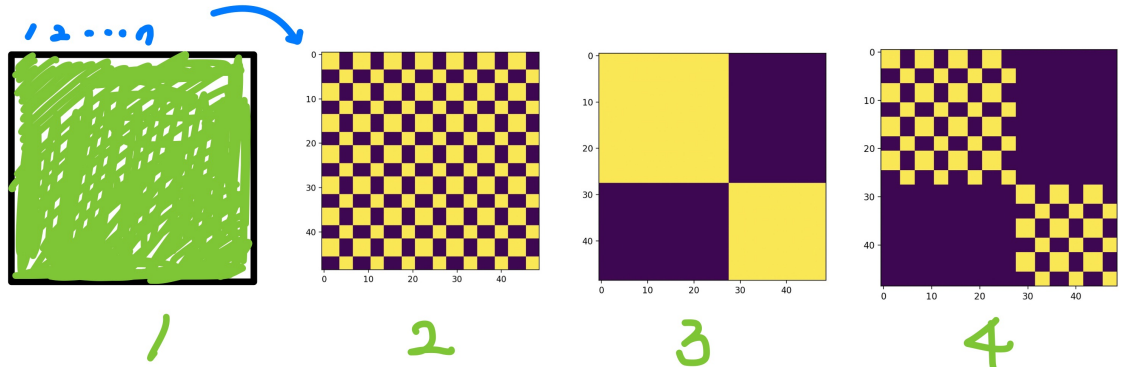




2

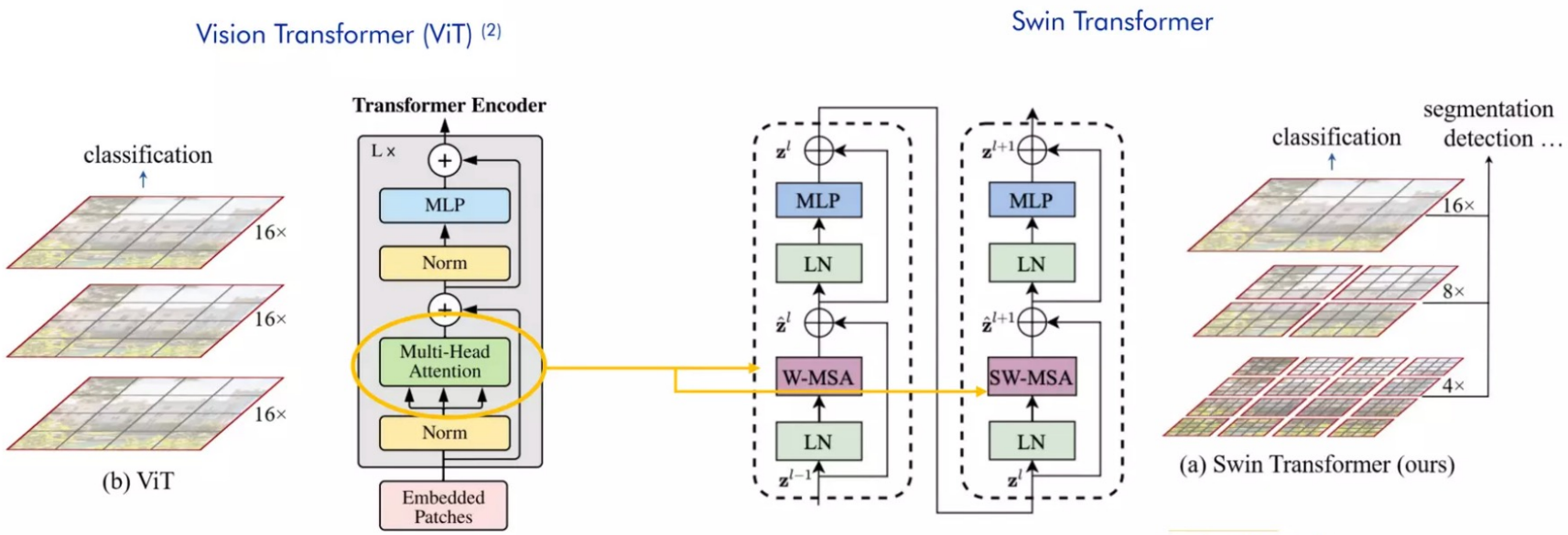


3



참조 : https://csm-kr.tistory.com/86

Q: 왜 연산량이 줄어드는 걸까?



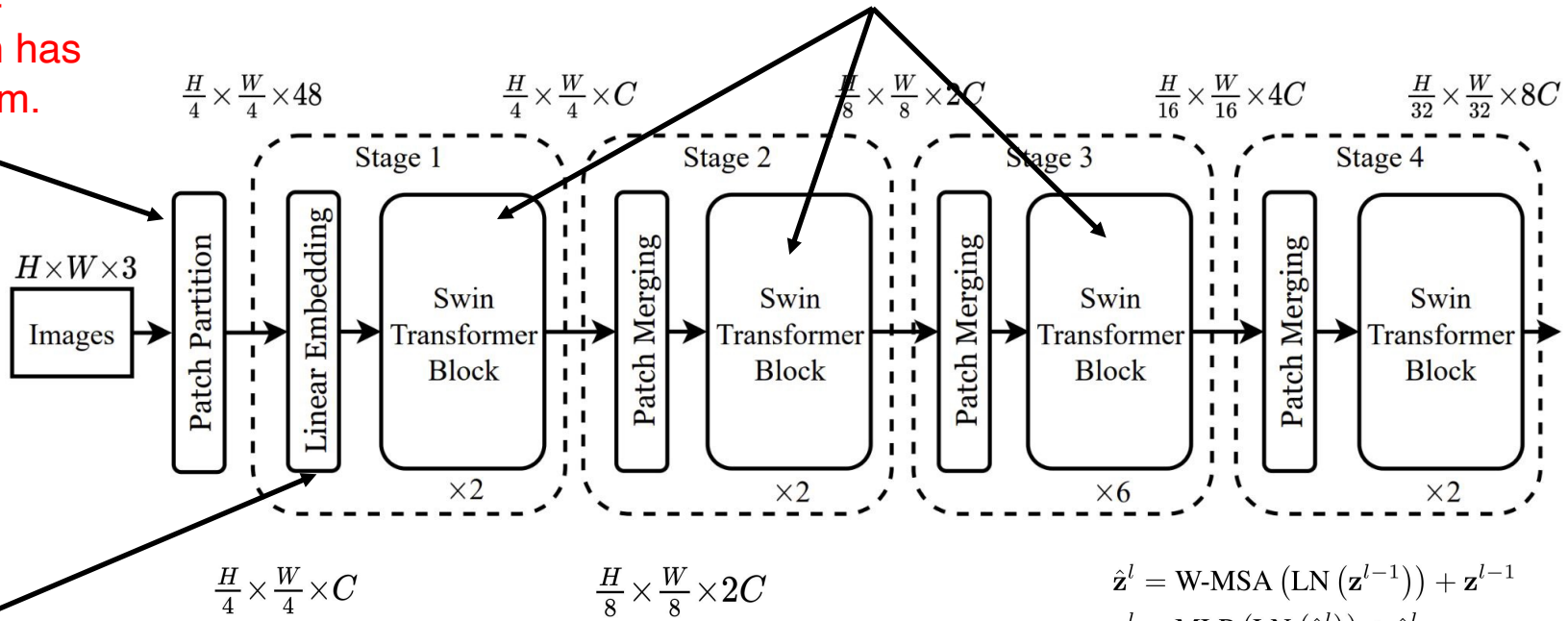
$$\Omega(\text{MSA}) = 4hwC^2 + 2(hw)^2C,$$

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

windows of size M×M

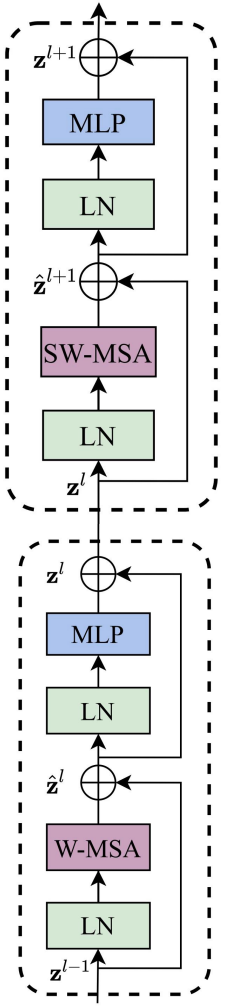
Input image is split to 4×4 patches; thus, each patch has 16 pixels * 3 RGB = 48 dim.

Patches are contextualized by self-attention within a window.



Each patch is linearly mapped to a feature of size C.

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



과제

19기
임서영

1) VISION TRANSFORMER 논문이나 SWIN TRANSFORMER 논문 둘중에 하나를 리뷰해주고,
코드역시 분석해서 blog or 다른 medium , notion 등등 에 올려서 정리하기

-> transformer 기반으로한 다른 vision 논문이여도 됩니다☺

2) Transformer 와 CNN의 차이가 반드시 들어가 있어야함

Vision Transformer : https://arxiv.org/abs/2010.11929 code : https://github.com/google-research/vision_transformer
Swin transformer : https://arxiv.org/abs/2103.14030 code : https://github.com/microsoft/Swin-Transformer

들어주셔서 감사합니다.