Vision Transformer Backbone

유현우

Vision & Display Systems Lab.

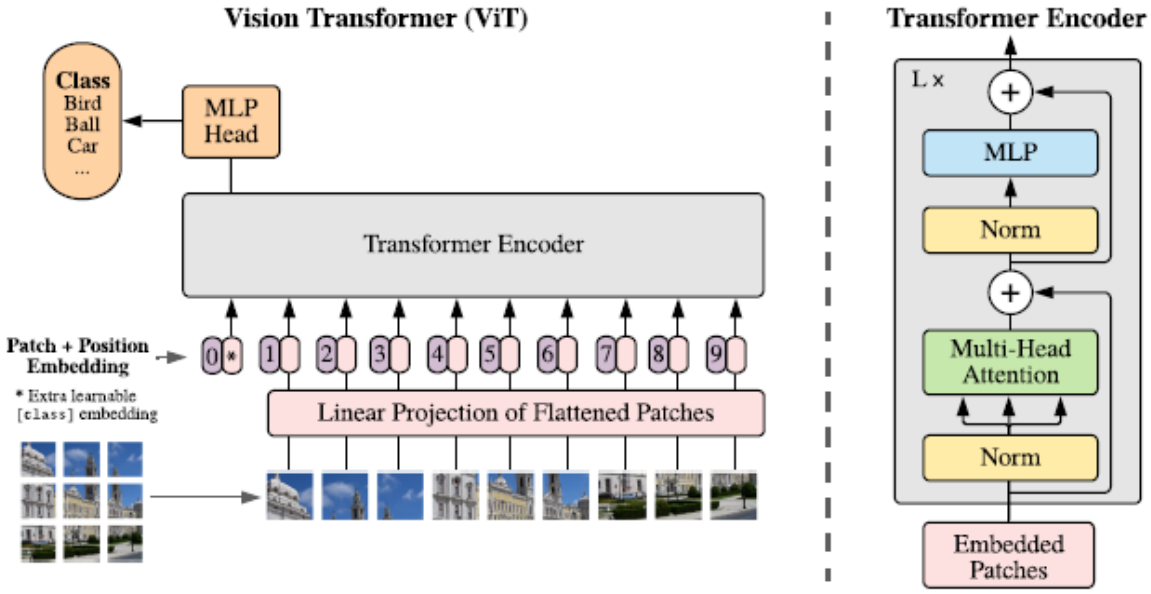
Dept. of Electronic Engineering, Sogang University

과제 개요

- Vision transformer
- Hierarchical structure& Self-attention layer
  - Pyramid vision transformer (PVT)
  - Swin transformer
  - Convolutional vision transformer(CvT)
- Experiments
- Conclusion

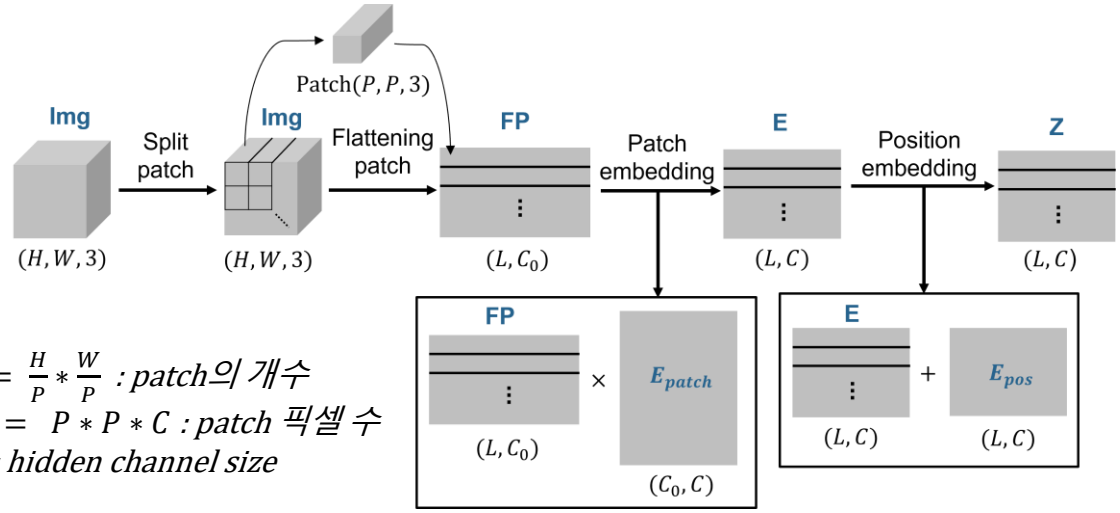
Vision transformer[1]

- 자연어 처리에서 사용되는 transformer를 vision task에 적용한 연구
- Image를 patch 단위로 나누어 embedding한 후 self-attention을 이용해 feature representation을 수행
  - Embedding된 patch간의 관계를 self-attention layer에서 global하게 고려



Vision transformer

- Embedding (patch size에 따라 downsampling이 된다는거 설명)
  - Image patch를 flatten한 후 linear projection을 이용해 patch embedding 수행
    - Patch를 대표하는 feature를 1d sequence 형태의 token으로 표현
  - Linear projection으로 손실된 공간정보를 위해 positional embedding 수행

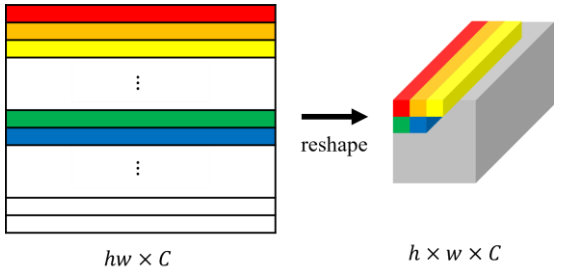


$L = \frac{H}{P} * \frac{W}{P}$: *patch의 개수*

$C_0 = P * P * C$: *patch 픽셀 수*

C : *hidden channel size*

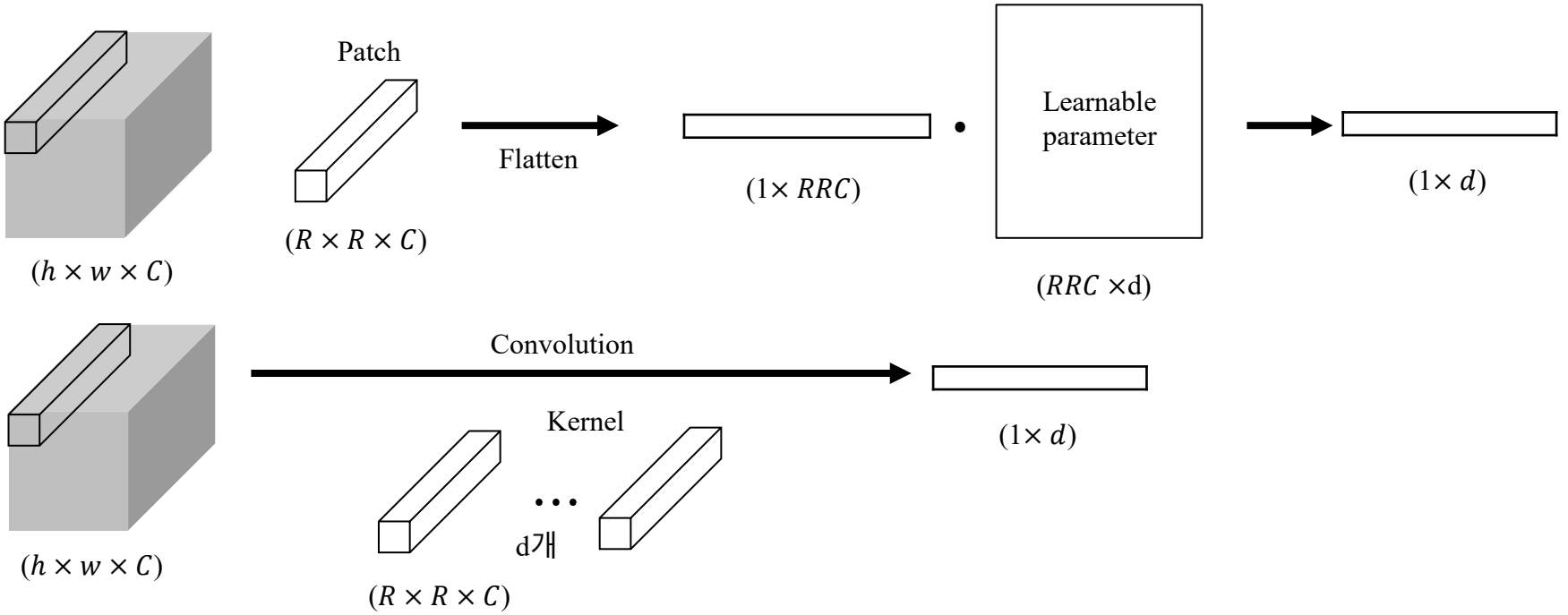
Embedding overview



Feature space

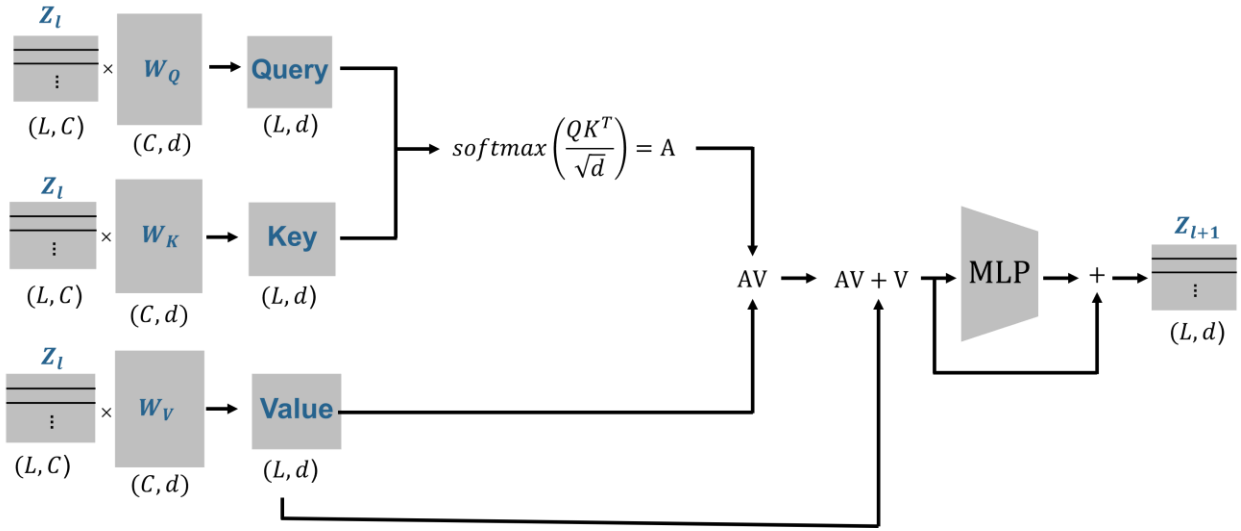
Vision transformer

- Embedding
  - 특정 영역(patch)의 모든 픽셀에 대해 learnable parameter를 취하여 1d sequence 형태의 vector 출력
    - d dimension으로 linear projection해주는 연산과 kernel = R, stride = R이고 출력 channel이 d인 convolution 연산의 역할이 동일



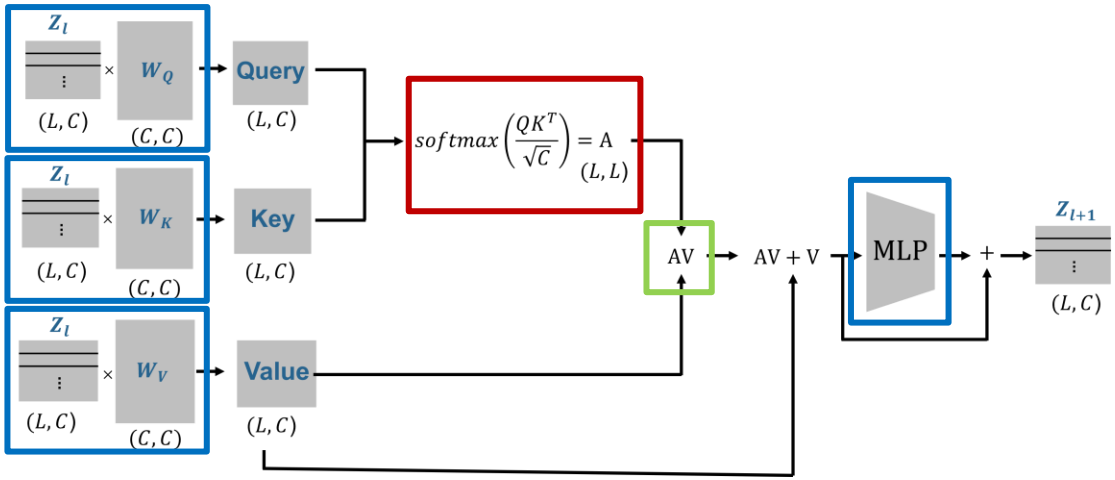
Vision transformer

- Self-attention
  - Learnable parameter로 linear projection을 수행하여 query, key, value 생성
  - Query, key간의 연산(Key · Query.transpose)을 통해 얻은 가중치를 value에 적용
    - Embedding된 patch간의 관계를 global하게 고려하여 feature representation



Vision transformer

- Vision transformer(ViT)
  - Self attention computational complexity of ViT : $4hwC^2 + 2(hw)^2C$
    - hw : patch 개수(L), C : hidden dimension
    - Computational complexity 계산
      - $(hw \times C)\cdot(C \times C)$ 연산 4번 → $4(hw)C^2$
      - $(hw \times C)\cdot(C \times hw)$ 연산 1번 → $(hw)^2C$
      - $(hw \times hw)\cdot(hw \times C)$ 연산 1번 → $(hw)^2C$



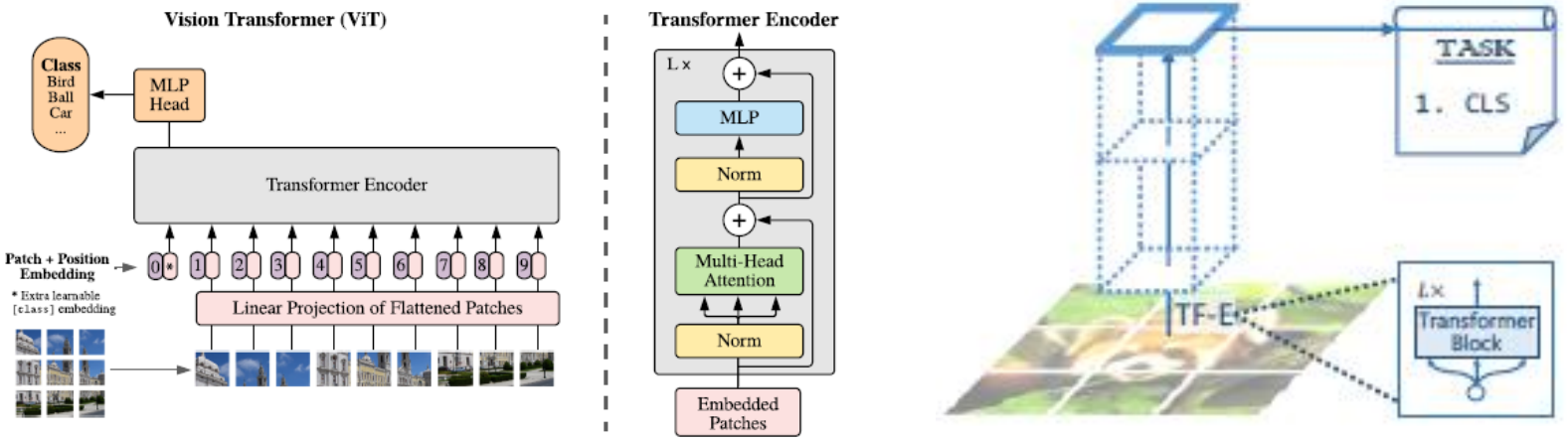
Self-attention layer

Vision transformer

- Problem
  - Unified resolution
    - Equal resolution features for all layers
    - Dense prediction (segmentation, detection…) difficult
  - Computationally inefficient
    - Large model size
    - Slow inference speed
    - Large FLOPs (floating-point operations)

Hierarchical structure

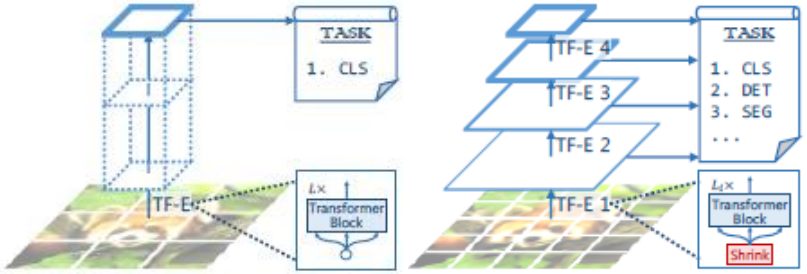
- Pyramid vision transformer(PVT) [1]
  - 백본이 Object detection 및 semantic segmentation 등의 dense prediction task에 적절히 사용되기 위해서는 high resolution 및 multi scale feature를 고려하여 학습해야 함
    - Transformer layer로 구성된 hierarchical structure의 백본 네트워크 제안



Vision transformer

Hierarchical structure

- Pyramid vision transformer(PVT)
  - 백본이 Object detection 및 semantic segmentation 등의 dense prediction task에 사용되기 위해서는 high resolution 및 multi scale feature를 고려하여 학습해야 함
    - Transformer layer로 구성된 hierarchical structure의 백본 네트워크 제안



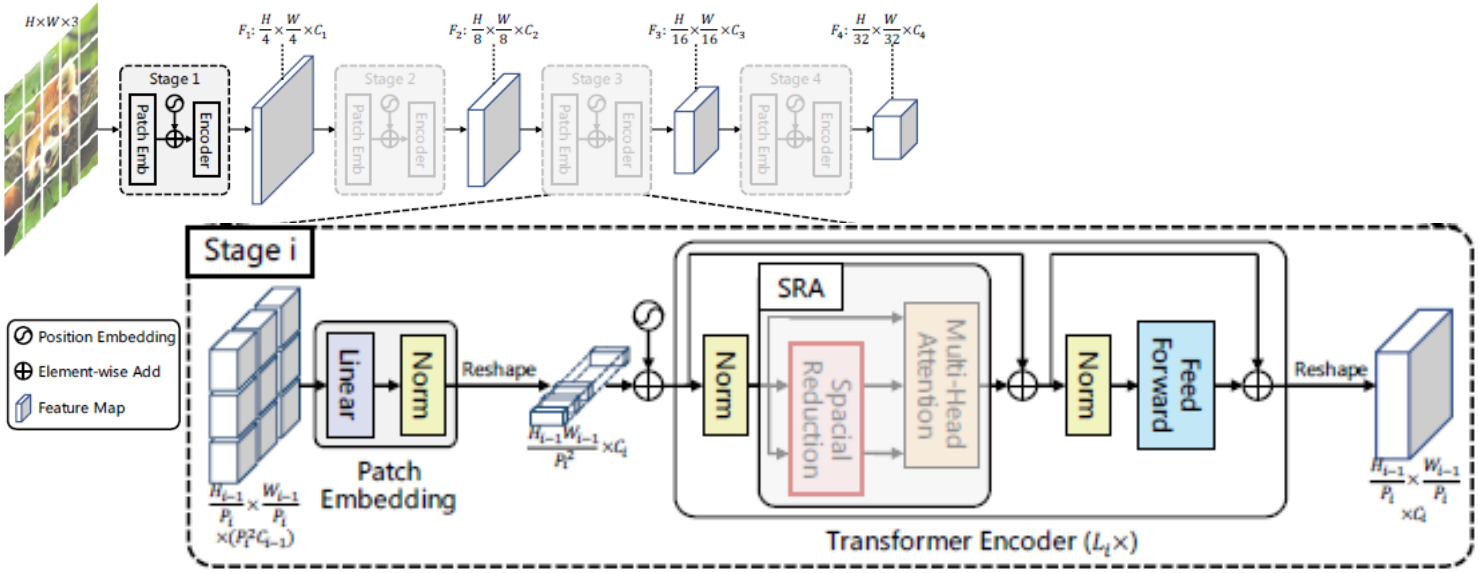
Vision transformer

Pyramid vision transformer

	Output Size
Stage 1	$\frac{H}{4} \times \frac{W}{4}$
Stage 2	$\frac{H}{8} \times \frac{W}{8}$
Stage 3	$\frac{H}{16} \times \frac{W}{16}$
Stage 4	$\frac{H}{32} \times \frac{W}{32}$

Hierarchical structure

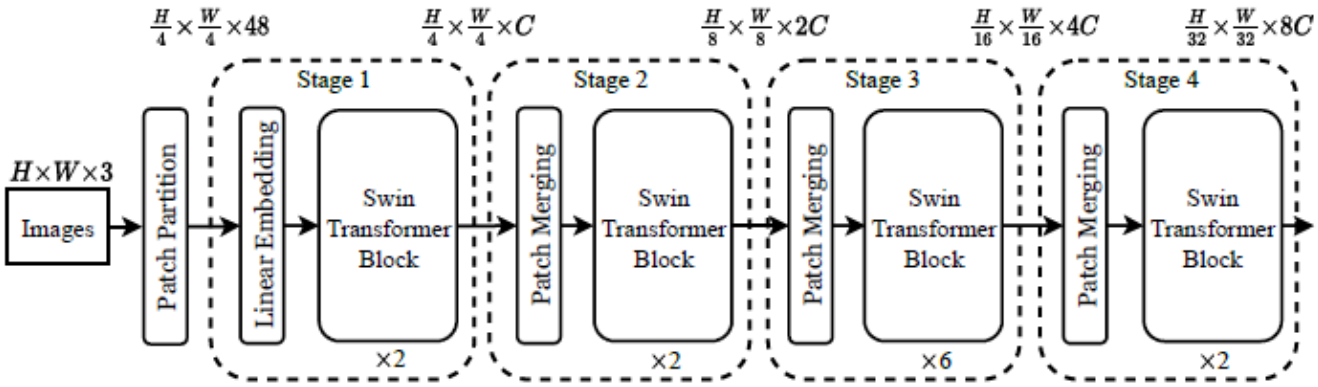
- Pyramid vision transformer(PVT)
  - 기존의 ViT에서 linear projection을 사용한 patch embedding을 사용
    - ViT는 network 첫 단에 16×16 size의 patch embedding을 수행
    - PVT는 Stage에 따라 patch size를 4×4, 2×2, 2×2, 2×2로 설정하여 downsampling 수행



Pyramid vision transformer architecture

Hierarchical structure

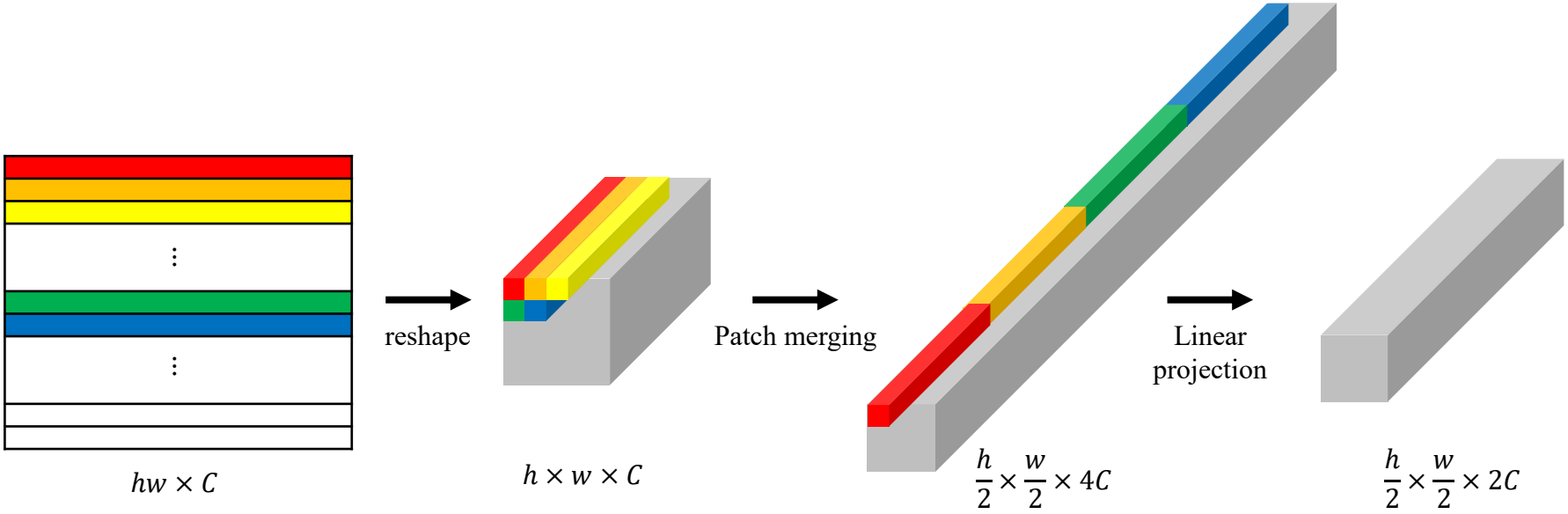
- Swin transformer[1]
  - Hierarchical한 network structure를 구현하기 위해 기존의 linear embedding과 patch merging 기법을 사용
    - Hierarchical한 구조에서의 block별 spatial resolution은 PVT와 동일하게 구성



Swin transformer architecture

Hierarchical structure

- Swin transformer
  - 인접한 patch token들을 channel 축으로 concat하고 linear projection을 적용하여 downsampling 수행
    - 특정 영역에 대해 하나의 token으로 표현하는 맥락에서 기존의 patch embedding과 유사하지만 token을 channel 축으로 concat하여 구현했다는 점에서 차별점을 가짐



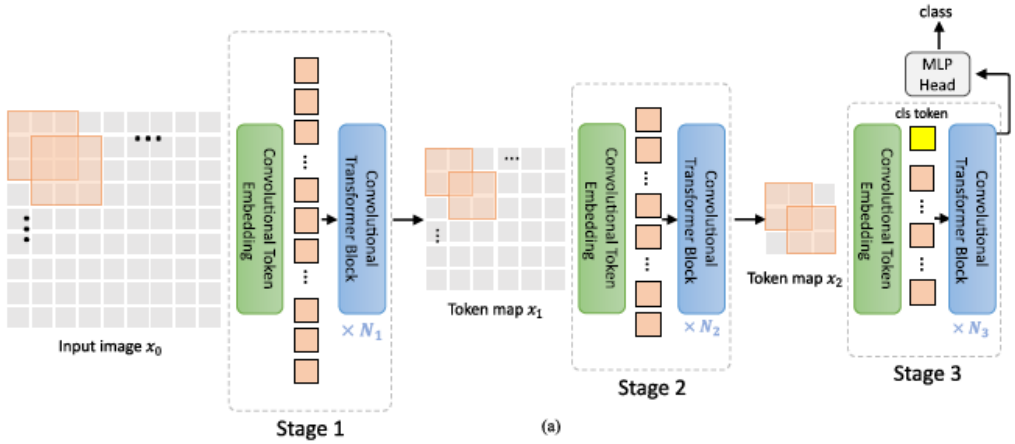
Patch merging overview

Hierarchical structure

- Convolutional vision transformer(CvT)[1]
  - Patch의 영역이 overlapped된 convolution으로 구현되는 convolutional token embedding layer를 적용
    - Locality : Image는 강력한 2D local structure를 갖고 있으며 공간적으로 인접한 픽셀은 일반적으로 높은 상관 관계가 있음
    - 각각의 token들이 분리된 영역을 나타내는 feature가 아니라 overlapped된 영역을 고려해 token을 생성

Method	Conv. Embed.	Pos. Embed.	#Param (M)	ImageNet top-1 (%)
a			19.5	80.7
b		✓	19.9	81.1
c	✓	✓	20.3	81.4
d	✓		20.0	81.6

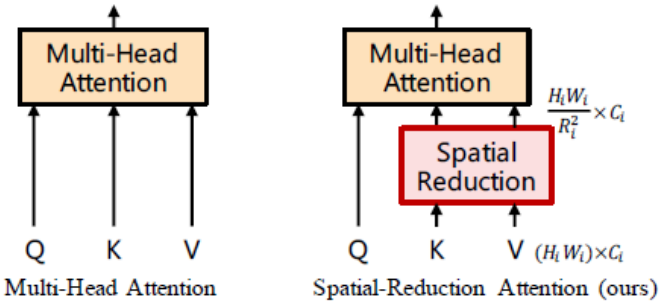
Table 6: Ablations on Convolutional Token Embedding.

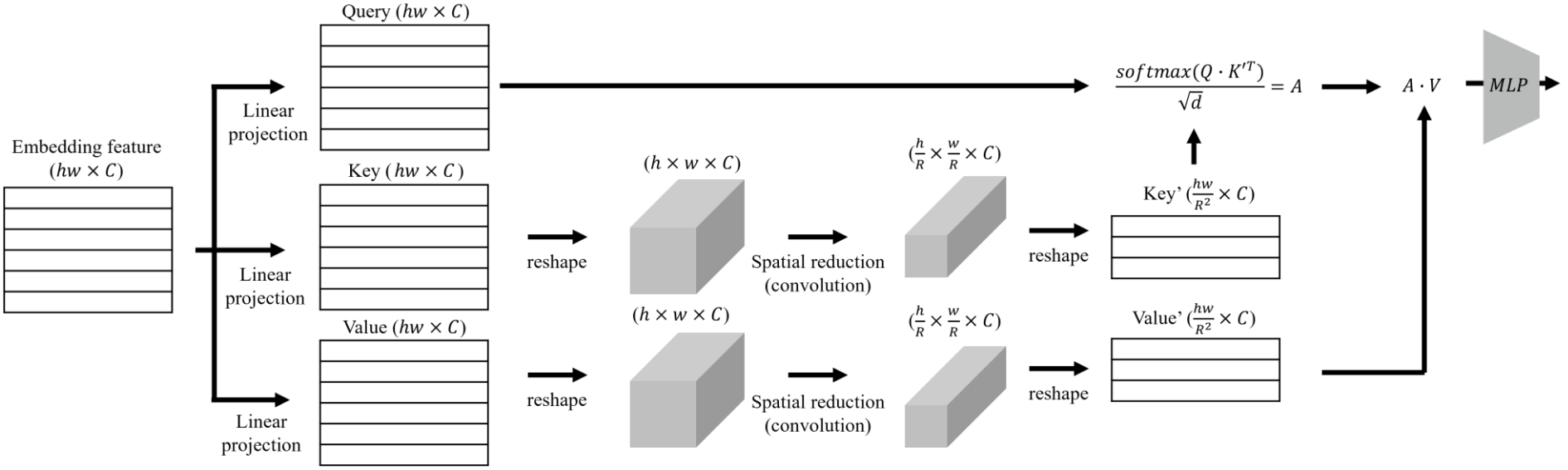


	Output Size	Layer Name	CvT-13	CvT-21
Stage1	56 × 56	Conv. Embed.	7 × 7, 64, stride 4	
	56 × 56	Conv. Proj. MHSA MLP	$\begin{bmatrix} 3\times3, 64 \\ H_1 = 1, D_1 = 64 \\ R_1 = 4 \end{bmatrix} \times 1$	$\begin{bmatrix} 3\times3, 64 \\ H_1 = 1, D_1 = 64 \\ R_1 = 4 \end{bmatrix} \times 1$
Stage2	28 × 28	Conv. Embed.	3 × 3, 192, stride 2	
	28 × 28	Conv. Proj. MHSA MLP	$\begin{bmatrix} 3\times3, 192 \\ H_2 = 3, D_2 = 192 \\ R_2 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 192 \\ H_2 = 3, D_2 = 192 \\ R_2 = 4 \end{bmatrix} \times 4$
Stage3	14 × 14	Conv. Embed.	3 × 3, 384, stride 2	
	14 × 14	Conv. Proj. MHSA MLP	$\begin{bmatrix} 3\times3, 384 \\ H_3 = 6, D_3 = 384 \\ R_3 = 4 \end{bmatrix} \times 10$	$\begin{bmatrix} 3\times3, 384 \\ H_3 = 6, D_3 = 384 \\ R_3 = 4 \end{bmatrix} \times 16$
Head	1 × 1	Linear	1000	
Params			19.98 M	31.54 M
FLOPs			4.53 G	7.13 G

Self-attention

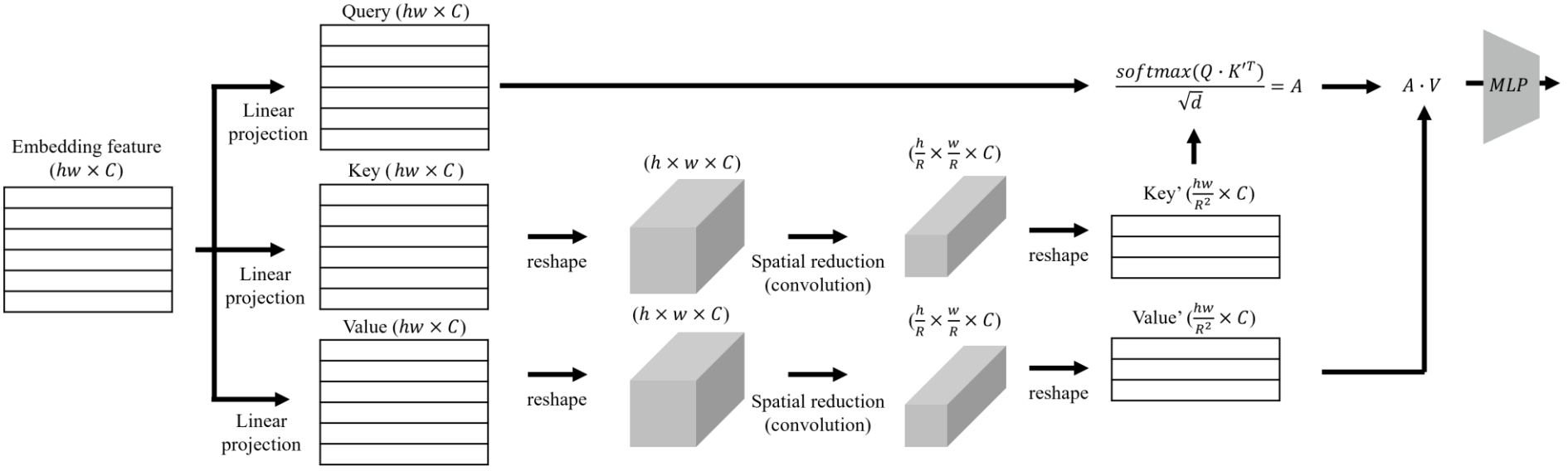
- Pyramid vision transformer(PVT)
  - Self-attention시 key와 value의 spatial reduction을 수행
    - Spatial reduction은 kernel=R, stride=R인 convolution연산으로 진행
      - Key와 value에 대해 더 넓은 영역을 갖는 patch embedding 수행
    - Global한 영역을 고려하지만 이를 sparse하게 고려한다는 뜻





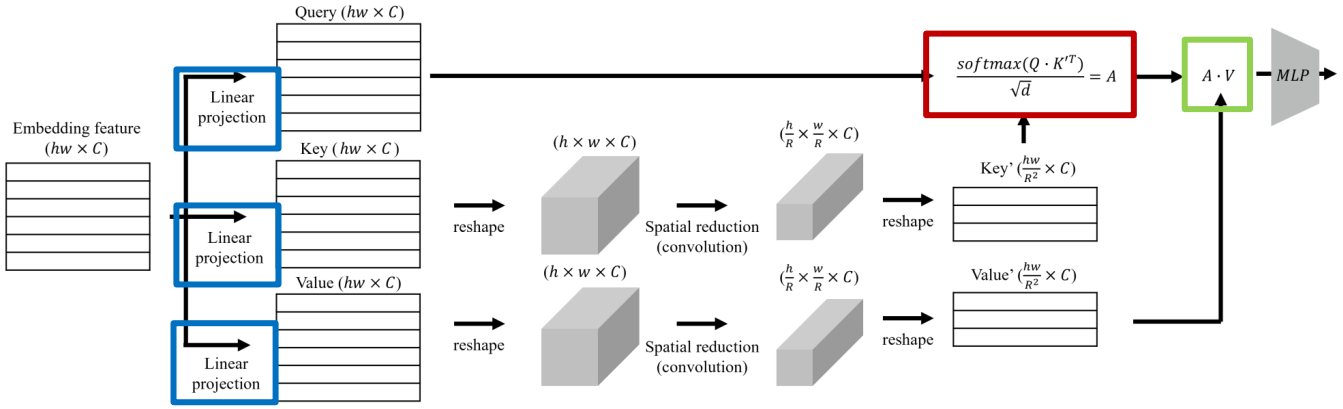
Self-attention

- Pyramid vision transformer(PVT)
  - Query · key.transpose = $A \rightarrow (hw \times C)\cdot(C \times \frac{hw}{R^2}) = (hw \times \frac{hw}{R^2})$
  - A · value = output → $(hw \times \frac{hw}{R^2}) \cdot(\frac{hw}{R^2} \times C) = (hw \times C)$
    - hw : patch 개수(L), C : hidden dimension



Self-attention

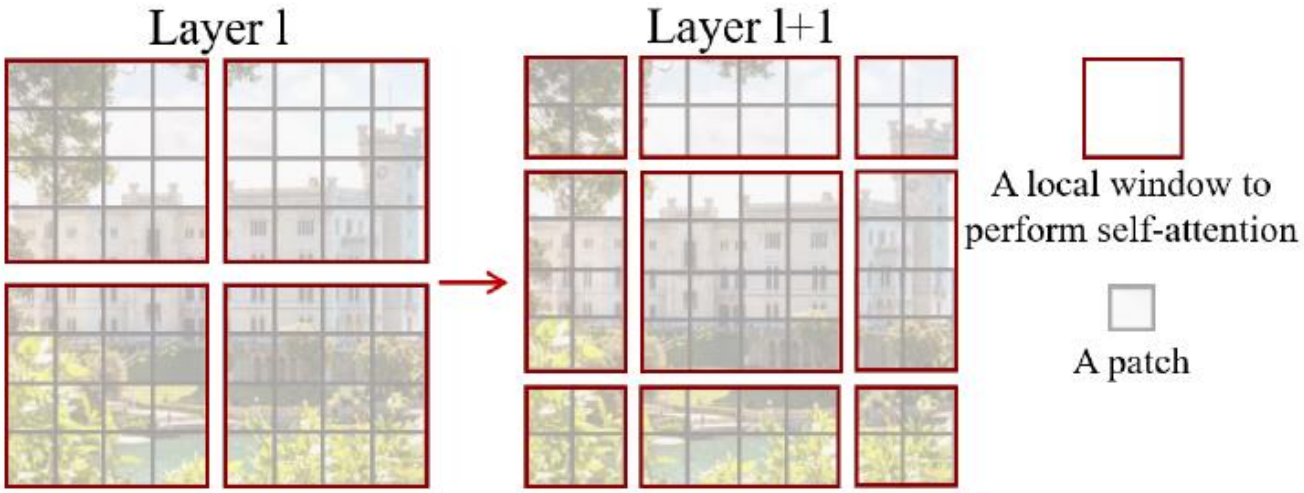
- Pyramid vision transformer(PVT)
  - Self attention computational complexity : $4hwC^2 + 2\frac{(hw)^2}{R^2}C$
    - hw : patch 개수(L), C : hidden dimension
    - Computational complexity 계산
      - $(hw \times C)\cdot(C \times C)$ 연산 4번 $\rightarrow 4(hw)C^2$
      - $(hw \times C)\cdot(C \times \frac{hw}{R^2})$ 연산 1번 $\rightarrow \frac{(hw)^2}{R^2}C$
      - $(hw \times \frac{hw}{R^2})\cdot(\frac{hw}{R^2} \times C)$ 연산 1번 $\rightarrow \frac{(hw)^2}{R^2}C$



Self-attention

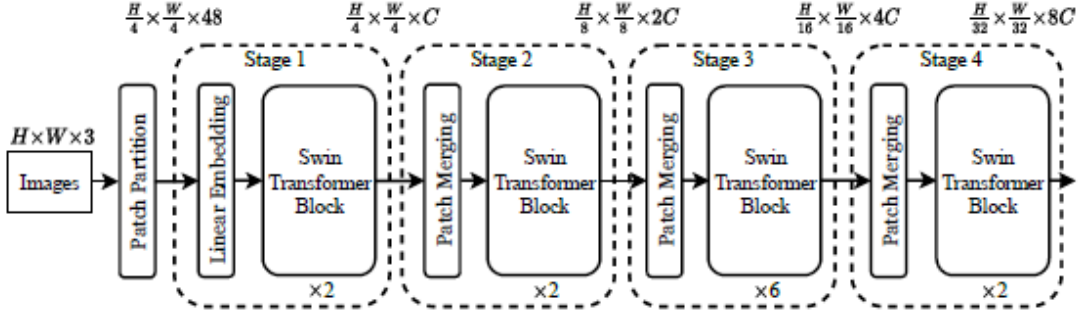
	ImageNet		COCO		ADE20k
	top-1	top-5	AP^{box}	AP^{mask}	mIoU
w/o shifting	80.2	95.1	47.7	41.5	43.3
shifted windows	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**

- Swin transformer
  - Window내의 patch를 대상으로 self attention을 수행하여 computational complexity 개선
  - Shifted window partitioning 기법이 적용된 swin transformer block을 이용
    - Window 기반 self attention은 window간 connection이 부족하여 modeling 능력이 제한
    - Non-overlapping widow 간 connection을 통해 modeling 능력을 향상

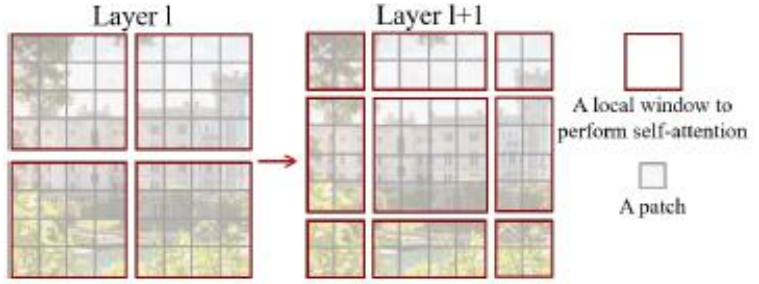


Self-attention

- Swin transformer
  - Self attention computational complexity : $4hwC^2 + 2M^2hwC$
    - hw : patch 개수, C : hidden dimension, M×M : Window 내 patch 개수
    - Computational complexity 계산
      - $(hw \times C)\cdot(C \times C)$ 연산 4번 $\rightarrow 4(hw)C^2$
      - $(M^2 \times C)\cdot(C \times M^2)$ 연산 $\frac{h}{M}\cdot\frac{w}{M}$ 번 $\rightarrow M^2hwC$
      - $(M^2 \times M^2)\cdot(M^2 \times C)$ 연산 $\frac{h}{M}\cdot\frac{w}{M}$ 번 $\rightarrow M^2hwC$
    - Input image의 해상도에 따라 quadratic하지 않고 linear한 computational complexity



Swin transformer architecture

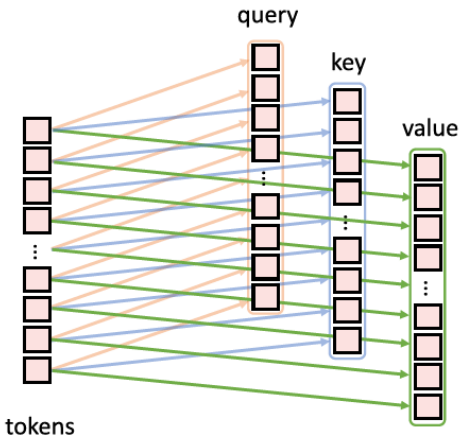


Shifted window partitioning

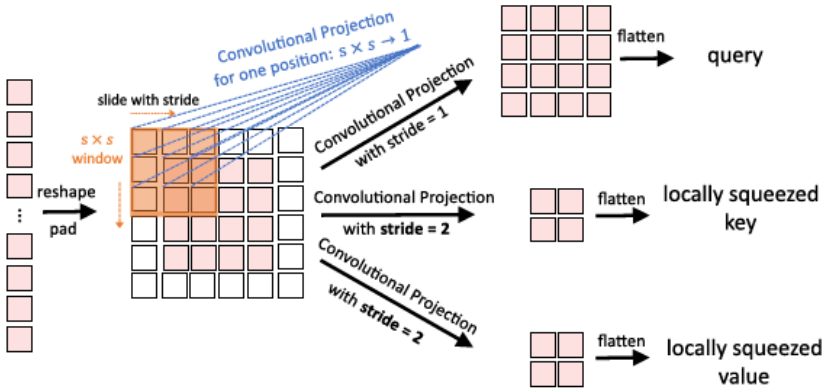
Self-attention

Method	Conv. Projection Stage 1	Stage 2	Stage 3	Imagenet top-1 (%)
a				80.6
b	✓			80.8
c	✓	✓		81.0
d	✓	✓	✓	81.6
#Blocks	1	2	10	

- Convolutional vision transformer(CvT)
  - 인접한 token간의 overlapped convolution 연산을 이용해 query, key, value 생성하는 convolutional projection 사용
    - 인접한 token간의 locality를 고려하여 self-attention 수행
    - Stride를 조정하여 key와 value의 spatial reduction을 수행



Linear projection in ViT



Squeezed convolutional projection

Experiments

- Classification (ImageNet-1k dataset)
  - Locality라는 inductive bias를 취해준 swin transformer 및 CvT에서 비교적 좋은 성능을 보임

Method	#Param (M)	GFLOPs	Top-1 Acc (%)
PVTv2-B0 (ours)	**3.4**	**0.6**	**70.5**
ResNet18 [13]	11.7	1.8	69.8
DeiT-Tiny/16 [29]	**5.7**	**1.3**	72.2
PVTv1-Tiny [31]	13.2	1.9	75.1
PVTv2-B1 (ours)	13.1	2.1	78.7
ResNet50 [13]	25.6	4.1	76.1
ResNeXt50-32x4d [33]	25.0	4.3	77.6
RegNetY-4G [24]	21.0	4.0	80.0
DeiT-Small/16 [29]	22.1	4.6	79.9
T2T-ViT$_t$-14 [35]	22.0	6.1	80.7
PVTv1-Small [31]	24.5	3.8	79.8
TNT-S [10]	23.8	5.2	81.3
Swin-T [21]	29.0	4.5	81.3
CvT-13 [32]	**20.0**	4.5	81.6
CoaT-Lite Small [34]	**20.0**	4.0	81.9
Twins-SVT-S [4]	24.0	**2.8**	81.7
PVTv2-B2-Li (ours)	22.6	3.9	**82.1**
PVTv2-B2 (ours)	25.4	4.0	82.0
ResNet101 [13]	44.7	7.9	77.4
ResNeXt101-32x4d [33]	44.2	8.0	78.8
RegNetY-8G [24]	39.0	8.0	81.7
T2T-ViT$_t$-19 [35]	39.0	9.8	81.4
PVTv1-Medium [31]	44.2	6.7	81.2
CvT-21 [32]	**32.0**	7.1	82.5
PVTv2-B3 (ours)	45.2	**6.9**	**83.2**
ResNet152 [13]	60.2	11.6	78.3
T2T-ViT$_t$-24 [35]	64.0	15.0	82.2
PVTv1-Large [31]	61.4	9.8	81.7
TNT-B [10]	66.0	14.1	82.8
Swin-S [21]	**50.0**	8.7	83.0
Twins-SVT-B [4]	56.0	**8.3**	83.2
PVTv2-B4 (ours)	62.6	10.1	**83.6**
ResNeXt101-64x4d [33]	83.5	15.6	79.6
RegNetY-16G [24]	84.0	16.0	82.9
ViT-Base/16 [7]	86.6	17.6	81.8
DeiT-Base/16 [29]	86.6	17.6	81.8
Swin-B [21]	88.0	15.4	83.3
Twins-SVT-L [4]	99.2	14.8	83.7
PVTv2-B5 (ours)	**82.0**	**11.8**	**83.8**

Experiments

- Object detection (COCO dataset)

Method	mini-val AP^{box}	mini-val AP^{mask}	test-dev AP^{box}	test-dev AP^{mask}	#param.	FLOPs
RepPointsV2* [12]	-	-	52.1	-	-	-
GCNet* [7]	51.8	44.7	52.3	45.4	-	1041G
RelationNet++* [13]	-	-	52.7	-	-	-
SpineNet-190 [21]	52.6	-	52.8	-	164M	1885G
ResNeSt-200* [78]	52.5	-	53.3	47.1	-	-
EfficientDet-D7 [59]	54.4	-	55.1	-	77M	410G
DetectoRS* [46]	-	-	55.7	48.5	-	-
YOLOv4 P7* [4]	-	-	55.8	-	-	-
Copy-paste [26]	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.

Backbone	#Param (M)	Mask R-CNN 1x AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}	Mask R-CNN 3x + MS AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet18 [21]	31.2	34.0	54.0	36.7	31.2	51.0	32.7	36.9	57.1	40.0	33.6	53.9	35.7
PVT-Tiny (ours)	32.9	36.7(+2.7)	59.2	39.3	35.1(+3.9)	56.7	37.3	39.8(+2.9)	62.2	43.0	37.4(+3.8)	59.3	39.9
ResNet50 [21]	44.2	38.0	58.6	41.4	34.4	55.1	36.7	41.0	61.7	44.9	37.1	58.4	40.1
PVT-Small (ours)	44.1	40.4(+2.4)	62.9	43.8	37.8(+3.4)	60.1	40.3	43.0(+2.0)	65.3	46.9	39.9(+2.8)	62.5	42.8
ResNet101 [21]	63.2	40.4	61.1	44.2	36.4	57.7	38.8	42.8	63.2	47.1	38.5	60.1	41.3
ResNeXt101-32x4d [72]	62.8	41.9(+1.5)	62.5	45.9	37.5(+1.1)	59.4	40.2	44.0(+1.2)	64.4	48.0	39.2(+0.7)	61.4	41.9
PVT-Medium (ours)	63.9	42.0(+1.6)	64.4	45.6	39.0(+2.6)	61.6	42.1	44.2(+1.4)	66.0	48.2	40.5(+2.0)	63.1	43.5
ResNeXt101-64x4d [72]	101.9	42.8	63.8	47.3	38.4	60.6	41.3	44.4	64.9	48.8	39.7	61.9	42.6
PVT-Large (ours)	81.0	42.9(+0.1)	65.0	46.6	39.5(+1.1)	61.9	42.5	44.5(+0.1)	66.0	48.3	40.7(+1.0)	63.4	43.7

Table 3: **Object detection and instance segmentation performance on COCO `val2017`.** AP^b and AP^m denote bounding box AP and mask AP, respectively.

Experiments

- Semantic segmentation (ADE20k dataset)

Backbone	Semantic FPN		
	#Param (M)	GFLOPs	mIoU (%)
ResNet18 [21]	15.5	32.2	32.9
PVT-Tiny (ours)	17.0	33.2	35.7(+2.8)
ResNet50 [21]	28.5	45.6	36.7
PVT-Small (ours)	28.2	44.5	39.8(+3.1)
ResNet101 [21]	47.5	65.1	38.8
ResNeXt101-32x4d [72]	47.1	64.7	39.7(+0.9)
PVT-Medium (ours)	48.0	61.0	41.6(+2.8)
ResNeXt101-64x4d [72]	86.4	103.9	40.2
PVT-Large (ours)	65.1	79.6	42.1(+1.9)
PVT-Large* (ours)	65.1	79.6	44.8

Table 4: **Semantic segmentation performance of different backbones on the ADE20K validation set.** "GFLOPs" is calculated under the input scale of 512 × 512. "*" indicates 320K iterations training and multi-scale flip testing.

Method	ADE20K Backbone	val mIoU	test score	#param.	FLOPs	FPS
DANet [23]	ResNet-101	45.2	-	69M	1119G	15.2
DLab.v3+ [11]	ResNet-101	44.1	-	63M	1021G	16.0
ACNet [24]	ResNet-101	45.9	38.5	-		
DNL [71]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [73]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [69]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [73]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [11]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [11]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [81]	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.

Conclusion

- Transformer가 vision task의 백본으로 활용되기 위해 hierarchical structure와 self-attention layer에서의 개선점을 찾는 연구가 활발히 진행
  - Pyramid vision transformer : patch embedding & spatial reduction attention
  - Swin transformer : patch merging & shifted window self attention
  - Convolutional vision transformer : convolutional token embedding & convolutional projection

- Transformer의 global한 feature representation의 장점을 유지하며 locality한 inductive bias를 취해주는 관점에서 성능 개선