AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

개요

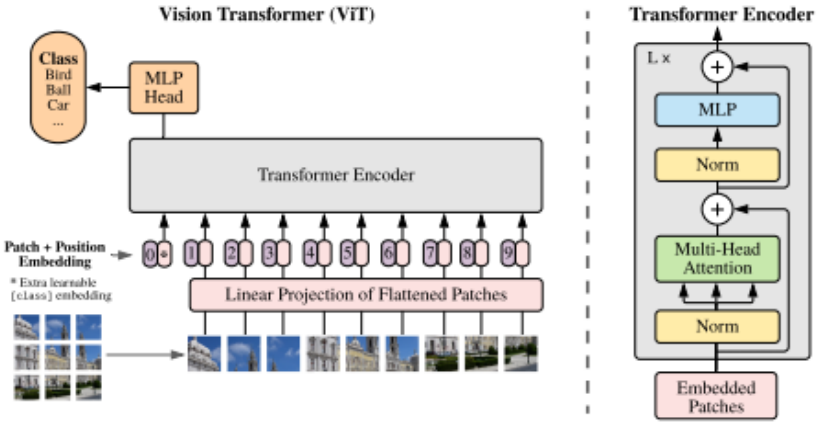
NLP에서는 transformer architecture가 사실상 표준. 그러나 computer vision 분야에서는 CNN에 의존하는 편.

이 논문은 CNN에 의존하지 않고 transformer만을 이용해 Img classification을 할 수 있음을 보여준다.

Pre-trained된 많은 양의 데이터를 여러 중간 크기 또는 작은 이미지recognition benchmark dataset으로 transfer했다.

Vision transformer(Vit)는 CNN에 비해 좋은 성능 & computational 자원 적게 사용함.

관련 작업들



이미지를 고정된 크기의 patch로 쪼갬 > linear하게 embedding > position embedding > transformer ecoder input.

Vision Transformer (ViT)

Input

: 일반 transformer는 token embedding에 대한 1차원 sequence

: 해당 논문에서는 2D 이미지를 다루므로 이미지를 flatten시켜서 2차원 path의 sequence로 변환해서 input으로 이용.

: H*W*C -> N*(P^2*C)

P 패치 크기, N = H*W/P^2 패치개수

: 이미지 패치를 고정된 벡터 크기 D, D차원 벡터로 linear projection

: BERT의 CLS토큰과 비슷하게 임베딩 된 패치의 시퀀스에 Z0=X_class 임베딩을 추가로 붙여넣음 인코더에 넣으면 그 output을 img representation으로 해석해서 classification에 이용."

위치 임베딩

: 각각의 patch embedding에 position embedding을 더해 위치 정보를 활용할 수 있도록 함.

: 학습 가능한 1차원 embedding이용

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{1}$$

$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L \tag{2}$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \qquad \ell = 1 \ldots L \tag{3}$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{4}$$

Hybrid architecture

: 이미지 patch를 그대로 사용하는 대신 CNN의 결과로 나온 feature map을 input sequence로 이용가능

Fine tuning + 높은 해상도 이미지 다루기

ViT 는 대량의 데이터셋에 대해 사전 학습한 후 더 작은 다운스트림 태스크에 fine-tuning 하는 방법을 취한다.

Fine-tuning 시에는 사전 학습된 prediction head 를 제거하고, 0 으로 초기화된 D x K 차원의 FC-Layer 를 연결한다. (K=다운스트림 태스크 카테고리 개수)

이때 fine-tuning 단계에서는 더 높은 해상도에서 학습하는 것이 정확도 향상에 좋다는 것이 알려져 있다.

더 높은 해상도의 이미지를 처리해야 할 경우, 이미지 패치 크기를 동일하게 유지함으로써 더 긴 패치 시퀀스를 사용한다.

ViT 는 더 높은 하드웨어의 메모리가 허용하는 한, 임의의 길이의 시퀀스를 처리할 수 있다.

단, 이 경우 사전 학습된 위치 임베딩이 의미없어진다.

이 경우 사전학습된 위치 임베딩에 원본 이미지에서의 위치에 따라 **2D interpolation** 을 수행한다.

이러한 해상도 조절과 패치 추출 방식은 ViT 에서 이미지의 2 차원 구조에 대한 inductive bias 를 수동적으로 다루는 유일한 포인트들이다.

모델사이즈

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Details of Vision Transformer model variants.

벤치마크 데이터셋 성능

	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

Model Representation

ViT가 이미지 데이터를 어떻게 처리하는 지 이해하기 위해

모델의 representation을 조사하자

- **Embedding Projection**

  : ViT는 펼쳐진 패치를 더 낮은 차원의 공간으로 mapping

- **Position embedding**

  : linear projection 이후 각 patch representation에 position embedding 더해짐.

  : 아래의 시각화 결과를 보면 모델은 이미지 내의 거리 개념을 인코딩하여 위치 임베딩에서 유사성이 나타난다.

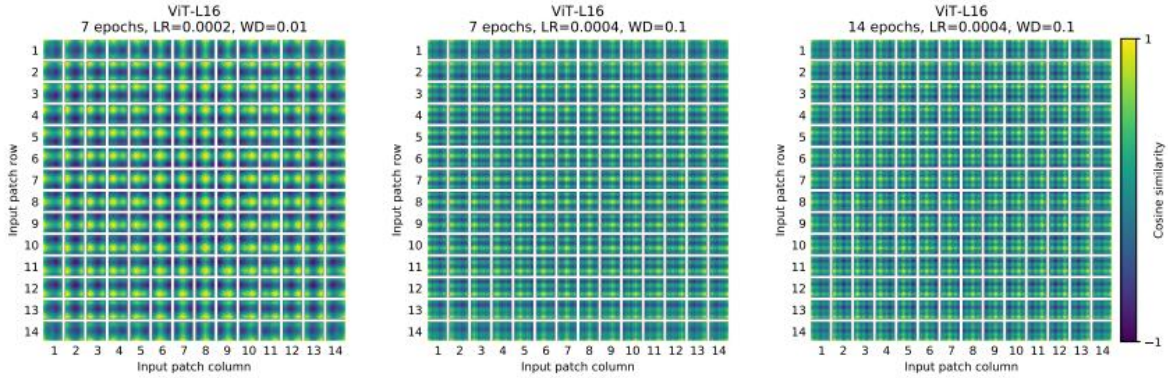


Figure 10: Position embeddings of models trained with different hyperparameters.

- **Self-attention**

  : self-attention은 가장 밑단에 있는 레이어에서부터도 ViT가 전체 이미지에 있는 정보를 통합하도록 돕는다.

: attention weight에 기반해 이미지 공간 상에서 정보가 취합되는 평균거리를 구하면 아래와 같음

: 여기서 attention distance는 CNN에서 receptive field와 비슷하게 해석 가능.

: attention head 중 일부는 가장 낮은 레이어에서부터 대부분의 이미지에 attend학 있고 이렇게 글로벌하게 정보를 통합하는 능력을 모델이 활용하는 것

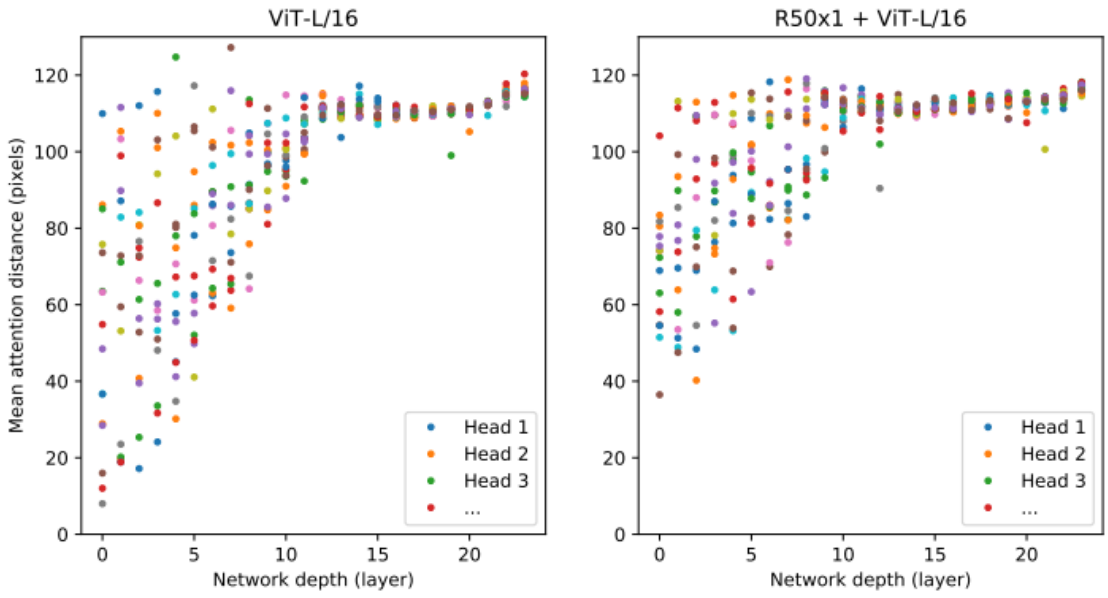
: 또 다른 attention head는 밑단 레이어에서 일관적으로 작은 거리의 패치에 집중하는 모습을 보였는데 이렇게 지역적인 attention은 하이브리드 모델에서는 좀처럼 나타나지 않음. 이런 attention head는 CNN밑단에서 일어나는 것과 비슷한 작용

: attention이 일어나는 거리는 네트워크의 깊이가 깊어질수록 늘어난다.

: 전체적으로 모델은 의미적으로 classification에 필요한 부분에 attention함. 맨 아래사진처럼.

-

-



-

-