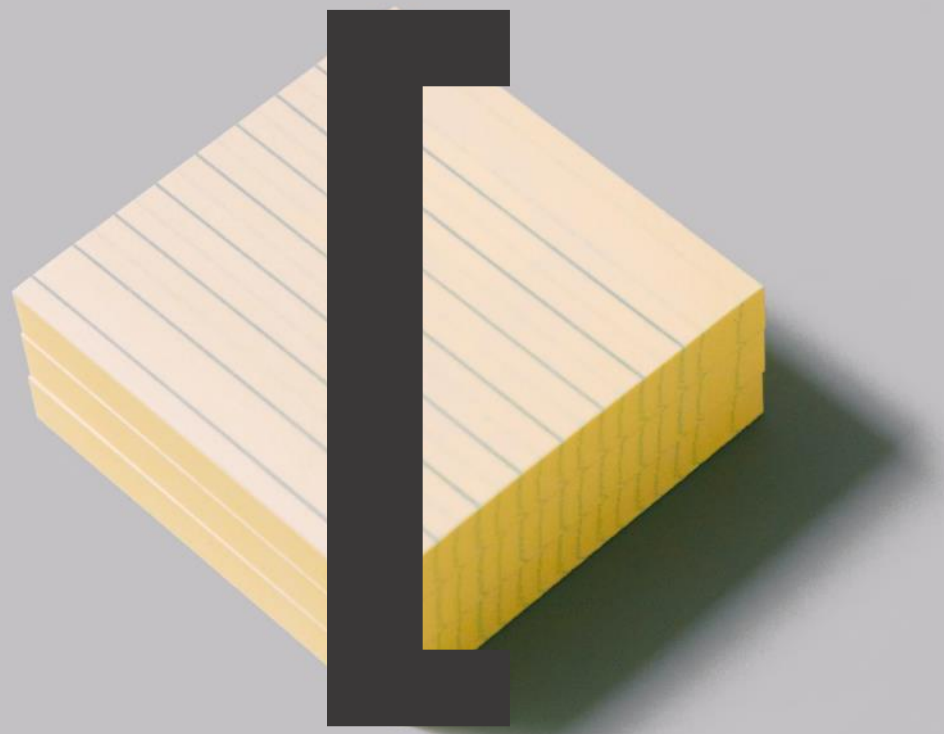


An Image is Worth 16x16 Words : Transformers for Image Recognition at Scale

Inductive Bias
Fine-Tuning
Higher Resolution
Few-shot
자기 지도 학습

Introduction

Part 1

Introduction

Transformer in Computer Vision

- Non-local neural network(Wang et al., 2018)
- Stand-alone self-attention in Vision models(Remachandran et al.,2019)
- Axial-DeepLab(Wang et al., 2020)

⋮

- Vision Transformer(Dosovitskiy et al., 2020)
- Data efficient image Transformer(Touvron et al., 2020)
- TransGAN(Jiang et al., 2021)

⋮

CNN에 self attention을
어떻게 적용할까?

Transformer 모델 자체를
이용해보자!

Part 1

Introduction

Vision Transformer(ViT) 개요

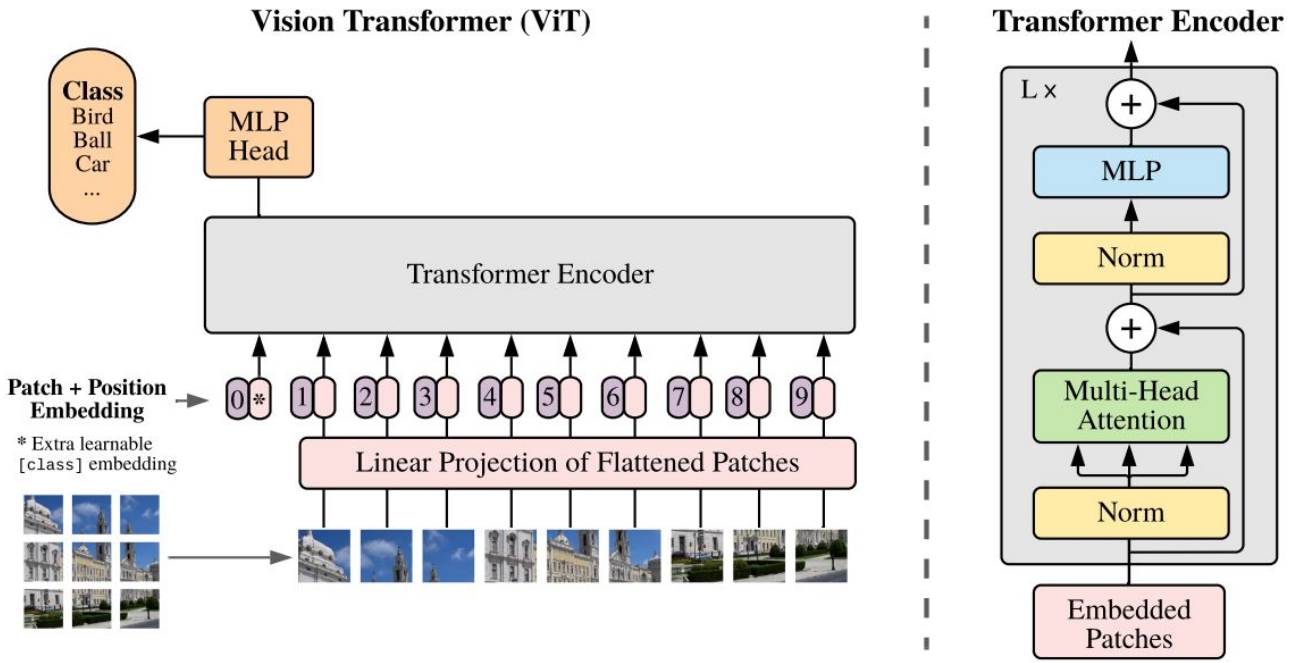
- 본 연구에서는 NLP에서 사용하는 Standard Transformer를 이미지에 그대로 적용하여 이미지 분류에 좋은 성능을 도출한 Vision Transformer(ViT)를 제안함
- ViT는 이미지를 패치로 분할한 후, 이를 NLP의 단어로 취급하여 각 패치의 linear embedding을 순서대로 Transformer의 input으로 넣어 이미지를 분류함
- ViT를 ImageNet-1k에 학습했을 때, 비슷한 크기의 ResNet보다 낮은 정확도를 도출하는 것을 통해 ViT가 CNN보다 Inductive bias가 낮은 것을 알 수 있음
- 반면, ImageNet-21k와 JFT-300M에 pre-training한 ViT를 다른 Image recognition task에 transfer learning했을 때, ViT가 SOTA 성능을 도출하는 것을 통해 large scale 학습이 낮은 Inductive bias로 인해 성능 저하를 해소시키는 것을 알 수 있음

Proposed Method

Part 2

Proposed Method

ViT 모델 구조

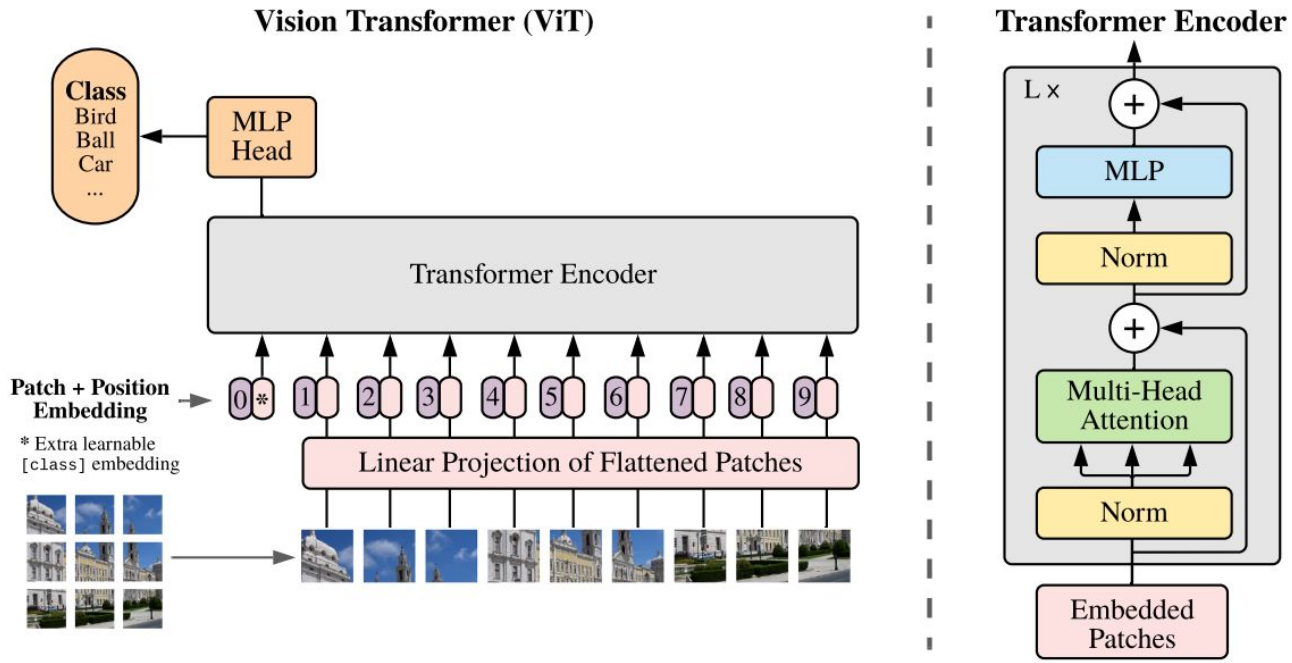


- ViT는 Multi-head Self Attention(MSA)와 MLP block으로 구성되어 있음
- MLP는 2개의 layer를 가지며, GELU activation function을 사용함
- 각 block의 앞에는 Layer Norm(LN)을 적용하고, 각 block의 뒤에는 residual connection을 적용함 x_q

Part 2

Proposed Method

ViT 모델 구조

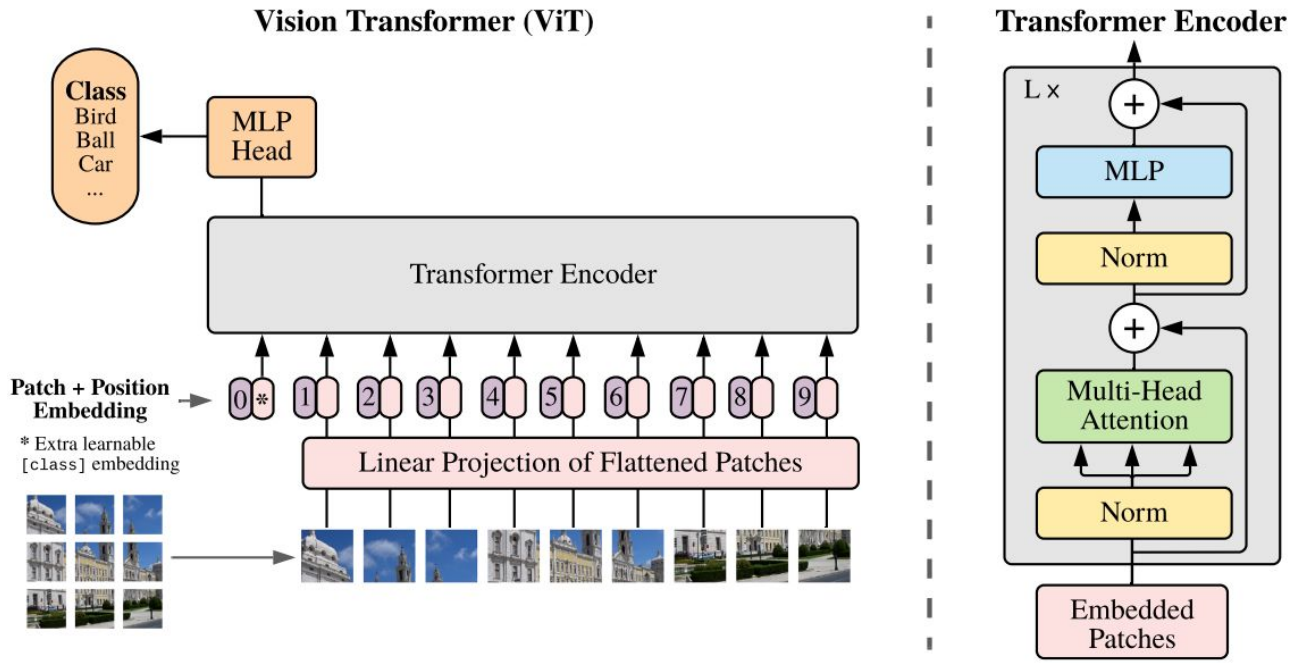


- ViT의 작동 과정은 아래와 같습니다.
  - Step1. 이미지 $x \in \mathbb{R}^{H \times W \times C}$ 가 있을 때, 이미지를 $N(=HW/P^2)$ 개의 $(P \times P)$ 크기의 패치 $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$ 로 분할하여 sequence 를 구축함
  - Step2. Trainable linear projection을 통해 x_q 의 각 패치를 flatten한 벡터를 D차원으로 변환한 후, 이를 패치 임베딩으로 사용함

Part 2

Proposed Method

ViT 모델 구조

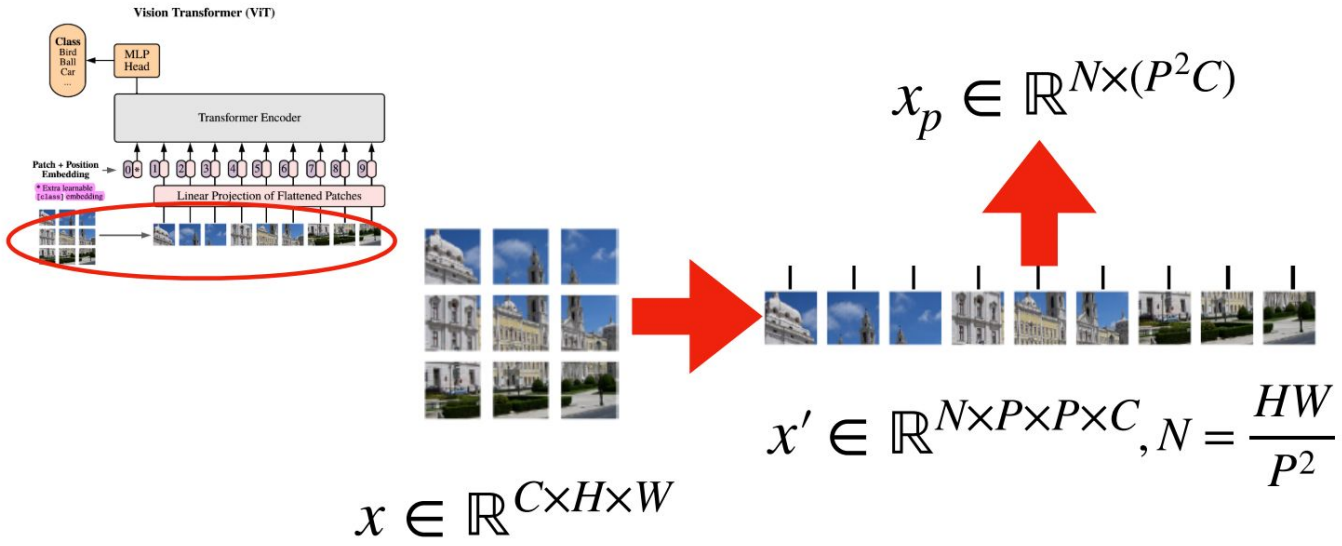


- ViT의 작동 과정은 아래와 같습니다.
  - Step3. Learnable class embedding과 Patch embedding에 learnable position embedding을 더함
  - Step4. embedding을 vanilla Transformer encoder에 input으로 넣어 마지막 layer에서 class embedding에 대한 output인 image representation을 도출함
  - Step 5. MLP에 image representation을 Input으로 넣어 이미지의 class를 분류함

Part 2

Proposed Method

Transformer Encoder

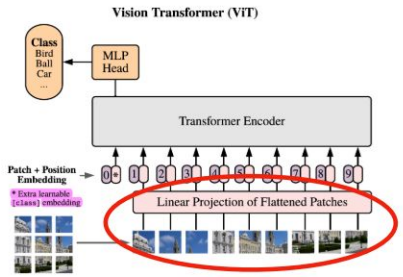


- 위 그림의 기호 중(C,H,W)는 각각 Channel, Height, Width를 의미하며 P는 Patch의 크기를 나타냅니다. 각 패치는 (C, P, P)의 크기를 가지게 되며 이때 N은 나뉘어진 패치의 갯수를 의마합니다.
- 각 패치를 Flatten 과정을 거쳐 벡터로 만들면 벡터의 크기는 가 되고 이 벡터가 N개가 됩니다. 이 N개의 벡터를 합친 것을 라고 합니다.

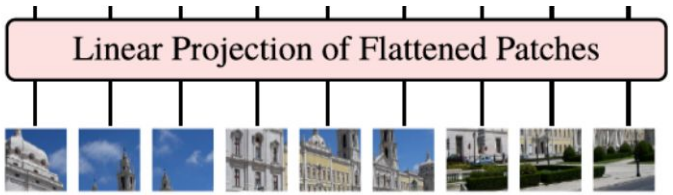
Part 2

Proposed Method

Transformer Encoder



$$[x_p^1 E; x_p^2 E; \cdots; x_p^N E] \in \mathbb{R}^{N \times D}$$



$E : matrix$

$$x_p \in \mathbb{R}^{N \times (P^2 C)}$$

$$x_p^i \in \mathbb{R}^{P^2 C}, E \in \mathbb{R}^{(P^2 C) \times D}$$

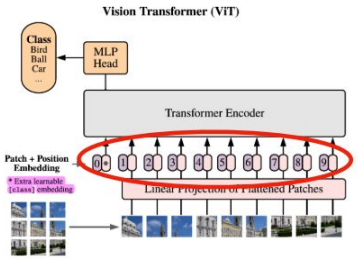
D는 embedding dimension

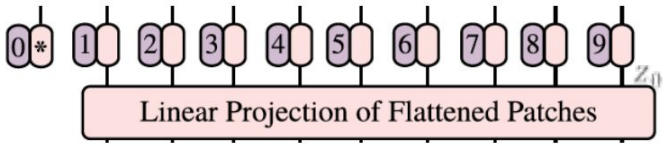
- 앞에서 생성한 x_p 를 Embedding하기 위하여 행렬 E와 연산을 해줍니다. E의 shape은 (, D)가 됩니다. D는 Embedding dimension로 크기의 벡터를 D로 변경하겠다는 의미입니다.
- 따라서 x_p 의 shape은 (N, P^2C) , E의 shape은 (, D)으로 곱연산을 하면 (N,D)의 크기를 가지게 됩니다.
- 배치 사이즈까지 고려하게 된다면 (B, N, D)의 크기를 가지는 텐서를 가지게 됩니다.

Part 2

Proposed Method

Transformer Encoder





$$[x_p^1 E; x_p^2 E; \cdots; x_p^N E] \in \mathbb{R}^{N \times D} \Rightarrow [x_{cls}; x_p^1 E; x_p^2 E; \cdots; x_p^N E] \in \mathbb{R}^{(N+1) \times D}$$

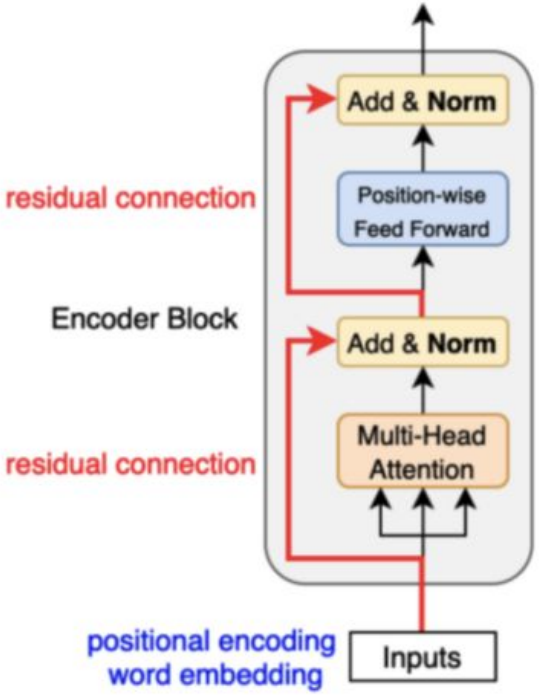
$$\Rightarrow z_0 = [x_{cls}; x_p^1 E; x_p^2 E; \cdots; x_p^N E] + E_{pos} \in \mathbb{R}^{(N+1) \times D}$$

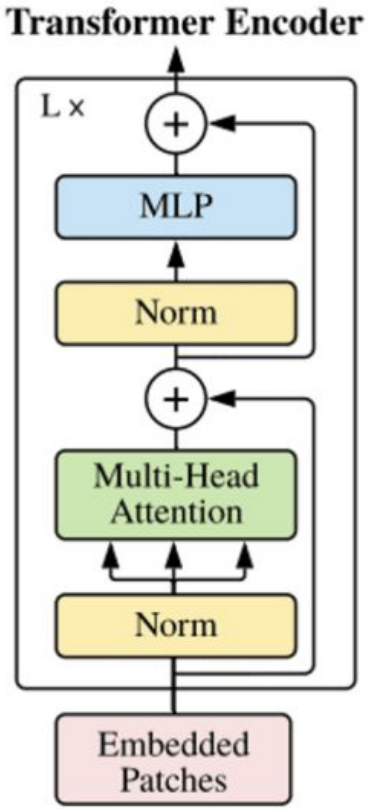
- Embedding한 결과에 클래스 토큰을 위 그림과 같이 추가합니다. 그러면 (N, D) 크기의 행렬이 (N+1, D)의 크기가 됩니다. 클래스 토큰은 학습 가능한 파라미터를 입력해 주어야 합니다.
- 마지막으로 Positional Encoding을 추가하기 위하여 (N+1, D) 크기의 행렬을 더해주면 z_0 입력값 준비가 마무리가 됩니다.

Part 2

Proposed Method

Transformer Encoder



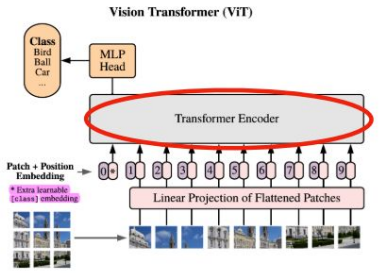


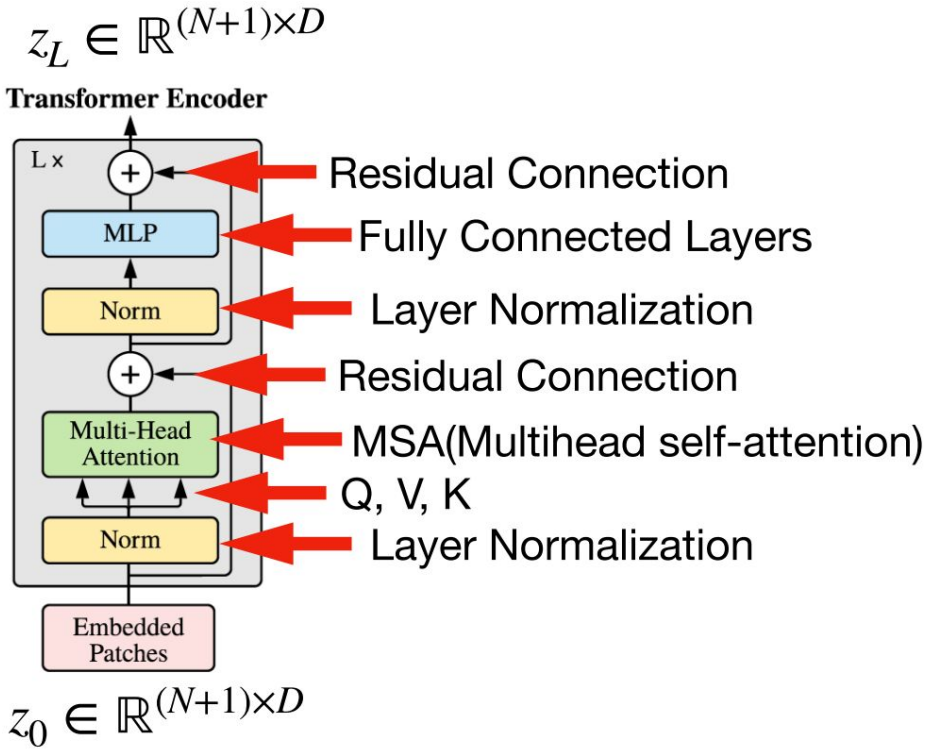
- Transformer의 Encoder는 L번 반복하기 위해 입력과 출력의 크기가 같도록 유지합니다.
- Vision Transformer에서 사용된 아키텍쳐는 기존의 Transformer Encoder와 조금 다르지만 큰 맥락은 유지합니다. 기존의 Transformer Encoder에서는 Multi-Head Attention을 먼저 진행한 다음에 LayerNorm을 진행하지만 순서가 바뀌어 있는 것을 알 수 있습니다.

Part 2

Proposed Method

Transformer Encoder



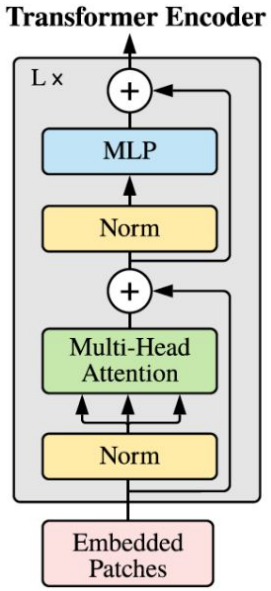


- 입력값 z_0 에서 시작하여 L번 반복 시 이 최종 Encoder의 출력이 됩니다.
- 해당 구조에서 사용된 Multihead Attention은 Self Attention이므로 Multihead Self Attention 즉, MSA로 표현하겠습니다.

Part 2

Proposed Method

Transformer Encoder



$$z'_l = MSA(LN(z_{l-1})) + z_{l-1}$$

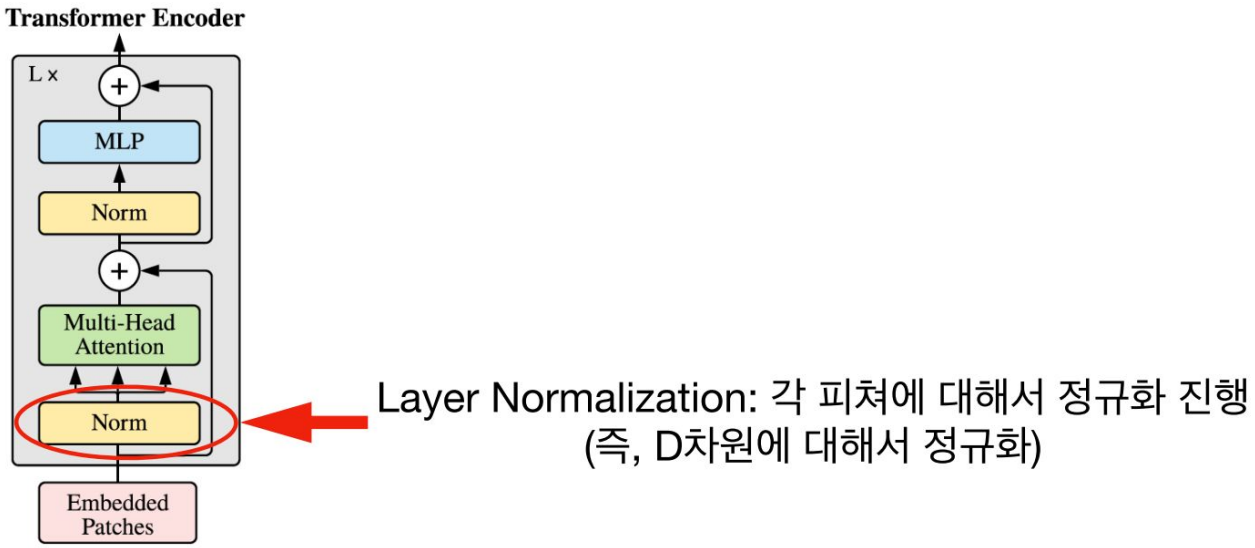
$$z_l = MLP(LN(z'_l)) + z'_l \qquad l = 1,2,\cdots,L$$

- 식과 같이 LM(LayerNorm), MSA, MLP 연산을 조합하면 Transformer Encoder를 구현할 수 있습니다.

Part 2

Proposed Method

Transformer Encoder

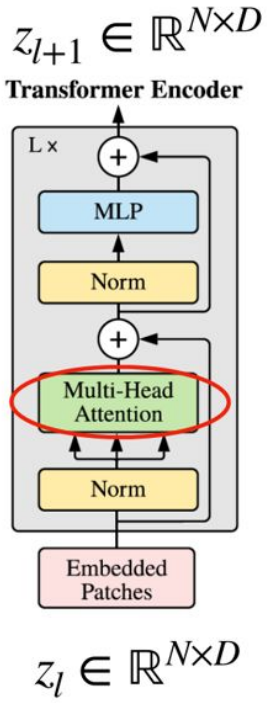


- Layer Normalization은 D차원에 대하여 각 feature에 대한 정규화를 진행합니다.

Part 2

Proposed Method

Transformer Encoder



$$[q, k, v] = zU_{qkv} \qquad U_{qkv} \in \mathbb{R}^{D \times 3D_h}$$

$$q, k, v \in \mathbb{R}^{N \times D_h}$$

$$A = softmax(qk^T / \sqrt{D_h}) \in \mathbb{R}^{N \times N}$$

$$SA(z) = Av \in \mathbb{R}^{N \times D_h}$$

$$MSA(z) = [SA_1(z); SA_2(z); \cdots; SA_k(z)]U_{msa}$$

$$U_{msa} \in \mathbb{R}^{k \cdot D_h \times D}$$

k는 헤드 수

$D_h = D/k$ 로 설정

- Multi-Head Attention에 대하여 알아보도록 하겠습니다.

Part 2

Proposed Method

Transformer Encoder

$$q = z \cdot w_q (w_q \in \mathbb{R}^{D \times D_h})$$

$$k = z \cdot w_k (w_k \in \mathbb{R}^{D \times D_h})$$

$$v = z \cdot w_v (w_v \in \mathbb{R}^{D \times D_h})$$

$$[q,k,v] = z \cdot U_{qkv} (U_{qkv} \in \mathbb{R}^{D \times 3D_h})$$

- Attention 구조에 맞게 q(query), k(Key), v(value)를 가지며 self attention 구조에 맞게 다음 식과 같이 q, k, v가 구성됩니다.

Part 2

Proposed Method

Transformer Encoder

$$\mathrm{A} = \mathrm{softmax}\left(\frac{q \cdot k^T}{\sqrt{D_h}}\right) \in R^{N \times N}$$

$$\mathrm{SA(z)} = A \cdot v \in R^{N \times D_h}$$

$$\mathrm{MSA(z)} = [SA_1(z); SA_2(z); \cdots ; SA_k(z)]U_{msa}$$

- 첫번째 식과 두번째 식을 이용하여 각 head에서의 self attention결과를 뽑고 세번째 식을 이용하여 각 head의 self attention 결과를 묶은 다음에 Linear 연산을 통해 최종적으로 Multi-head Attention의 결과를 얻을 수 있습니다.
- 세번째 식에서 self attention의 결과를 묶은 것의 shape은 (N, D_h , D)이므로 연산의 결과는 (N, D)가 됩니다. 이 과정을 통해 Transformer Encoder의 입력과 같은 shape을 가지도록 조절할 수 있습니다.

Part 2

Proposed Method

Transformer Encoder

$$\text{head1} : q_1 = z \cdot w_q^1, k_1 = z \cdot w_k^1, v_1 = z \cdot w_v^1$$

$$\text{head2} : q_2 = z \cdot w_q^2, k_2 = z \cdot w_k^2, v_2 = z \cdot w_v^2$$

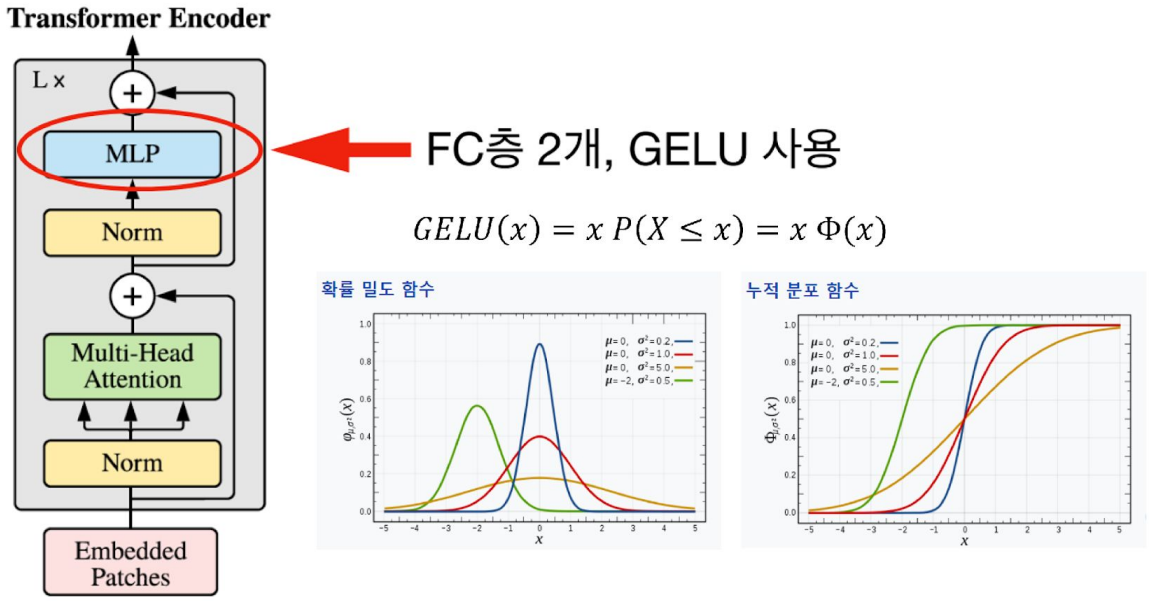
$$\text{Single Head} : \text{q,k,v} \in \mathbb{R}^{N \times D_h} \rightarrow \textit{Multi Head} : q, k, v \in \mathbb{R}^{N \times k \times D_h}$$

- 실제 Multi-Head Attention을 구현할 때, 각 head의 q,k,v에 대한 연산을 따로 하지 않고 한번에 처리할 수 있습니다.
- 첫번째, 두번째 식과 같이 같은 구조의 head에서 weight만 달라지게 되므로 세번째 식처럼 같이 한번에 묶어서 연산할 수 있습니다.

Part 2

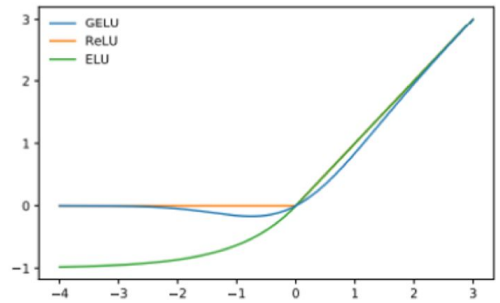
Proposed Method

Transformer Encoder



FC층 2개, GELU 사용

$$GELU(x) = x\,P(X \le x) = x\,\Phi(x)$$



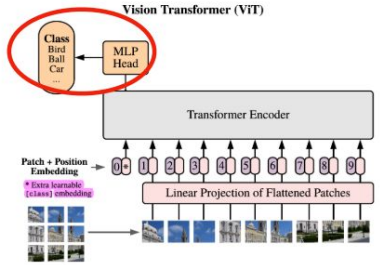
igure 1: The GELU ($\mu = 0, \sigma = 1$), ReLU, and ELU $\alpha = 1$).

- 마지막으로 MLP과정을 거치고, 이 때, GELU Activation을 사용합니다.
- GELU는 입력값과 입력값의 누적 정규 분포의 곱을 사용한 형태입니다.
- 이 함수 또한 모든 점에서 미분 가능하고 단조 증가 함수가 아니므로 Activation 함수로 사용가능하며 입력값 x가 다른 입력에 비해 얼마나 큰지에 대한 비율로 값이 조정되기 때문에 확률적인 해석이 가능해지는 장점이 있습니다.

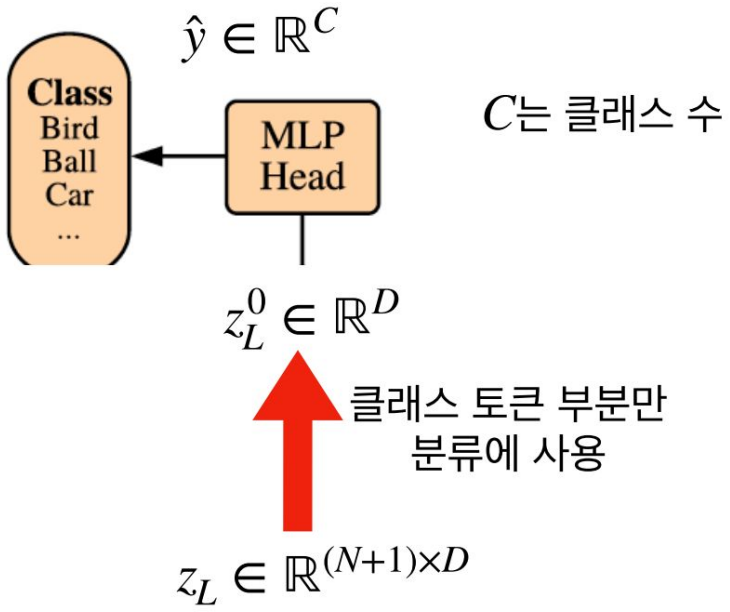
Part 2

Proposed Method

Transformer Encoder



MLP Head: LN을 적용하고 FC를 거쳐 $\hat{y}$을 생성



- L번 반복한 Transformer Encoder의 마지막 출력에서 클래스 토큰 부분만 분류 문제에 사용하게 되며 마지막에 추가적인 MLP를 이용하여 클래스를 분류하게 됩니다.

Part 1

Proposed Method

Positional Embedding

Pos. Emb.	Default/Stem	Every Layer	Every Layer-Shared
No Pos. Emb.	0.61382	N/A	N/A
1-D Pos. Emb.	0.64206	0.63964	0.64292
2-D Pos. Emb.	0.64001	0.64046	0.64022
Rel. Pos. Emb.	0.64032	N/A	N/A

- ViT에서는 4가지 Position embedding을 시도한 후 최종적으로 가장 효과가 좋은 1D position embedding을 ViT에 사용함
  - No positional information : Considering the inputs as a bag of patchs
  - 1-dimensional positional embedding : Considering the inputs as a sequence of patches in the raster order
  - 2-dimensional positional embedding : Considering the inputs as a grid of patches in two dimensions.
  - Relative Positional embeddings : Considering the relative distance between patches to encode the spatial information as instead of their absolute position.

Part 2

Proposed Method

딥러닝 모델들과 Inductive Bias

Component	Entities	Relations	Rel. inductive bias	Invariance
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations

- Inductive Bias는 training에서 보지 못한 데이터에 대해서도 적절한 귀납적 추론이 가능하도록 하기 위해 모델이 가지고 있는 가정들의 집합을 의미함
- DNN의 기본적인 요소들의 inductive bias는 아래와 같음
  - Fully connected: 입력 및 출력 element가 모두 연결되어 있으므로 구조적으로 특별한 relational inductive bias를 가정하지 않음
  - Convolutional: CNN은 작은 크기의 kernel로 이미지를 지역적으로 보며, 동일한 kernel로 이미지 전체를 본다는 점에서 locality와 transitional invariance 특성을 가짐
  - Recurrent: RNN은 입력한 데이터들이 시간적 특성을 가지고 있다고 가정하므로 sequentiality와 temporal invariance 특성을 가짐
- **Transformer는 CNN 및 RNN보다 상대적으로 inductive bias가 낮음**

Part 2

Proposed Method

Inductive Bias 역할

- Inductive Bias는 Hypothesis Space를 결정하는 역할을 합니다.
- Hypothesis Space는 다른 표현으로 설명하자면 '최적의 모델을 찾는 공간'을 의미합니다.
- 전체 탐색 공간의 크기가 크다면 데이터의 일반화 관계를 더 잘 표현하는 Hypothesis를 찾기 위해 상대적으로 많은 데이터가 필요하게 되고 탐색 공간의 크기가 작고 적절한 범위로 제한시켜준다면 상대적으로 적은 데이터로도 최적의 Hypothesis를 찾는 것이 쉬워집니다.
- 하지만 역으로 생각해보면 모델이 갖고 있는 Inductive Bias가 데이터의 상관관계를 충분히 잘 표현할 수 있다면 문제가 되지 않겠지만 그렇지 않다면 오히려 표현해야하는 영역을 아예 표현하지 못하게 될 수 있습니다.

-> ViT는 Inductive Bias가 상대적으로 낮은 모델로서 기존의 CNN이 지역적(local) 특징으로부터 전역적(global) 특징을 찾아갔던 것과 달리, 처음부터 전역적인 특징을 찾으려고 합니다. 즉 편견 없이 문제를 풀려고 한다고 생각하면 됩니다.

Part 2

Proposed Method

Inductive Bias 역할

> Transformers lack some of the inductive biases inherent to CNNs, such as translation equivariance and locality, and therefore do not generalize well when trained on insufficient amounts of data.

- 이미지 데이터를 분석하는데 있어서 CNN은 Parameter sharing을 통한 hierarchical view를 제공하는 반면 ViT에서 파라미터 공유는 MLP Layer가 거의 유일합니다.
- ViT는 구조적으로 패치 간의 상관 관계 해석 방식 조차도 자체적으로 학습해야합니다.(CNN은 픽셀 간 해석 방식을 'hierarchical'이라고 구조적으로 제시)

-> 이미지 해석에 대해 패치 간의 상관관계 자체를 학습하도록 하는 자유를 더 주기 때문에 상대적으로 Inductive Bias가 더 낮고 이를 학습하기 위해 필요한 데이터 수가 더 많다고 이야기할 수 있습니다.

Part 2

Proposed Method

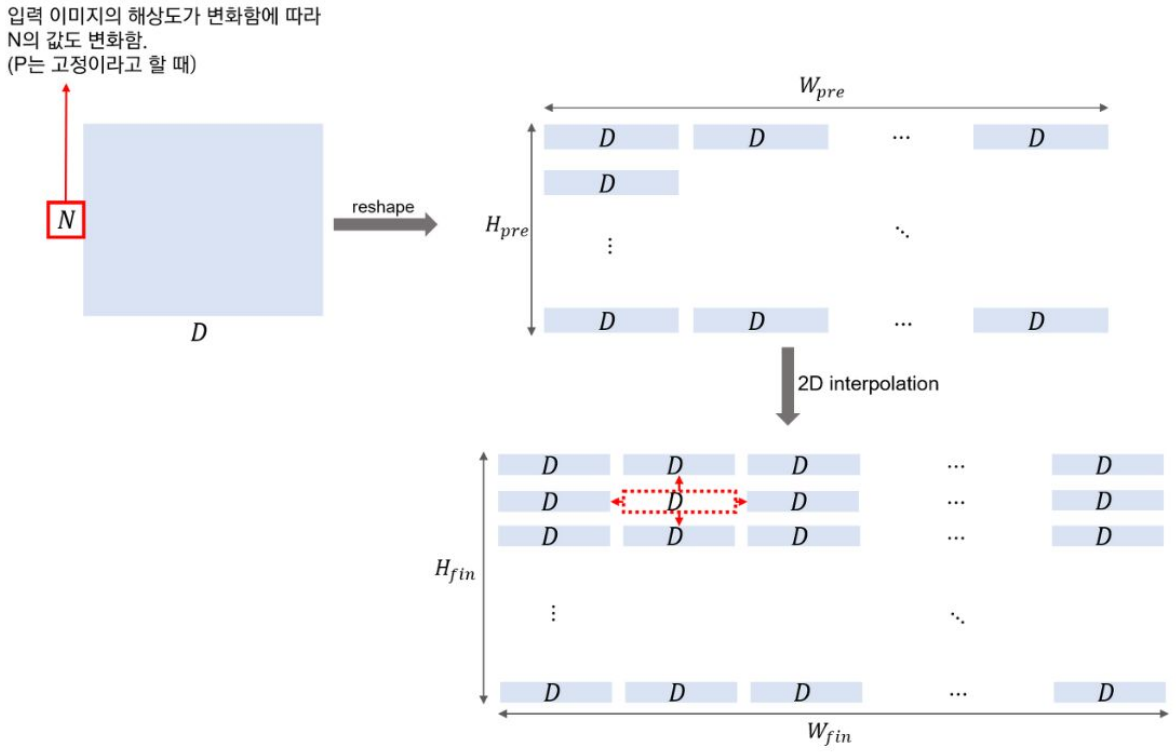
Hybrid Architecture

- ViT는 raw image가 아닌 CNN으로 추출한 raw image의 feature map을 활용하는 hybrid architecture로도 사용할 수 있음
- Feature map은 이미 raw image의 공간적 정보를 포함하고 있으므로 hybrid architecture는 패치 크기를 1x1로 설정해도 됨
- 1x1크기의 패치를 사용할 경우 feature map의 공간 차원을 flatten하여 각 벡터에 linear projection을 적용하면 됨

Part 2

Proposed Method

Fine-Tuning and Higher Resolution



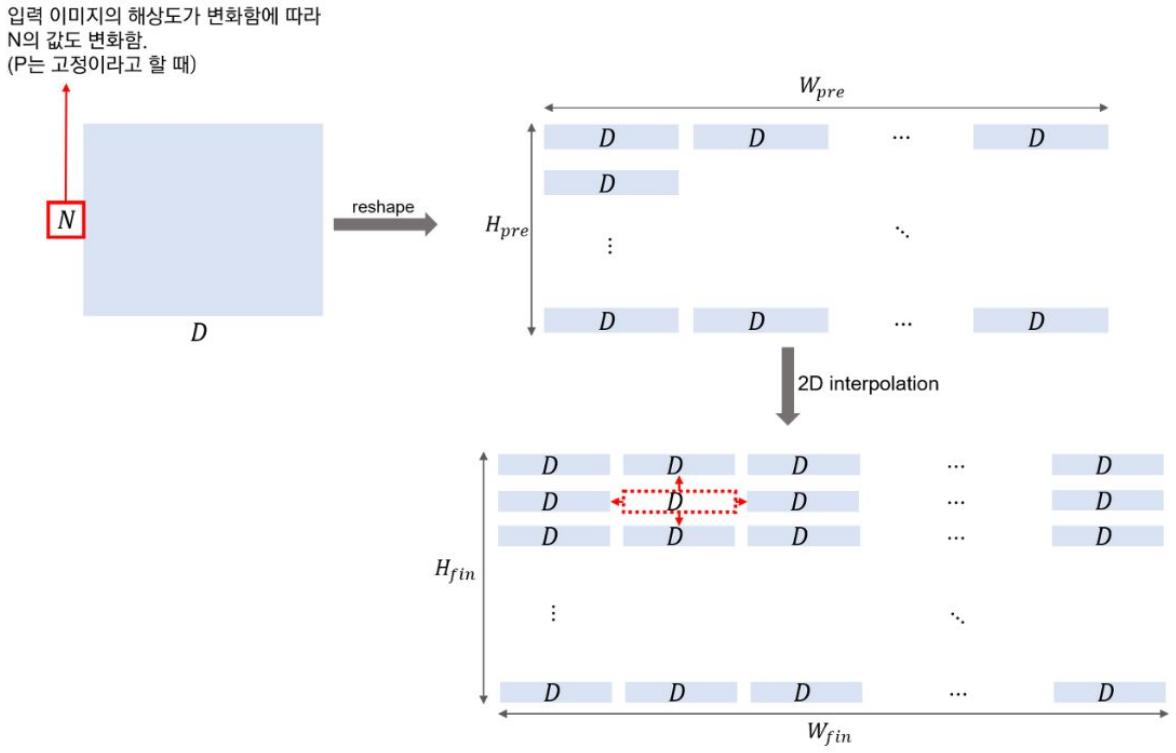
2d interpolation when fine-tuning

- 일반적으로 large-scale dataset에 대해 ViT를 Pre-train하고 downstream에 대해 fine-tuning을 수행한다. 이를 위해 pre-trained prediction head를 제거하고 0으로 초기화된 D x K feedforward layer를 추가한다. 여기서 K는 downstrem class의 개수이다.

Part 2

Proposed Method

Fine-Tuning and Higher Resolution



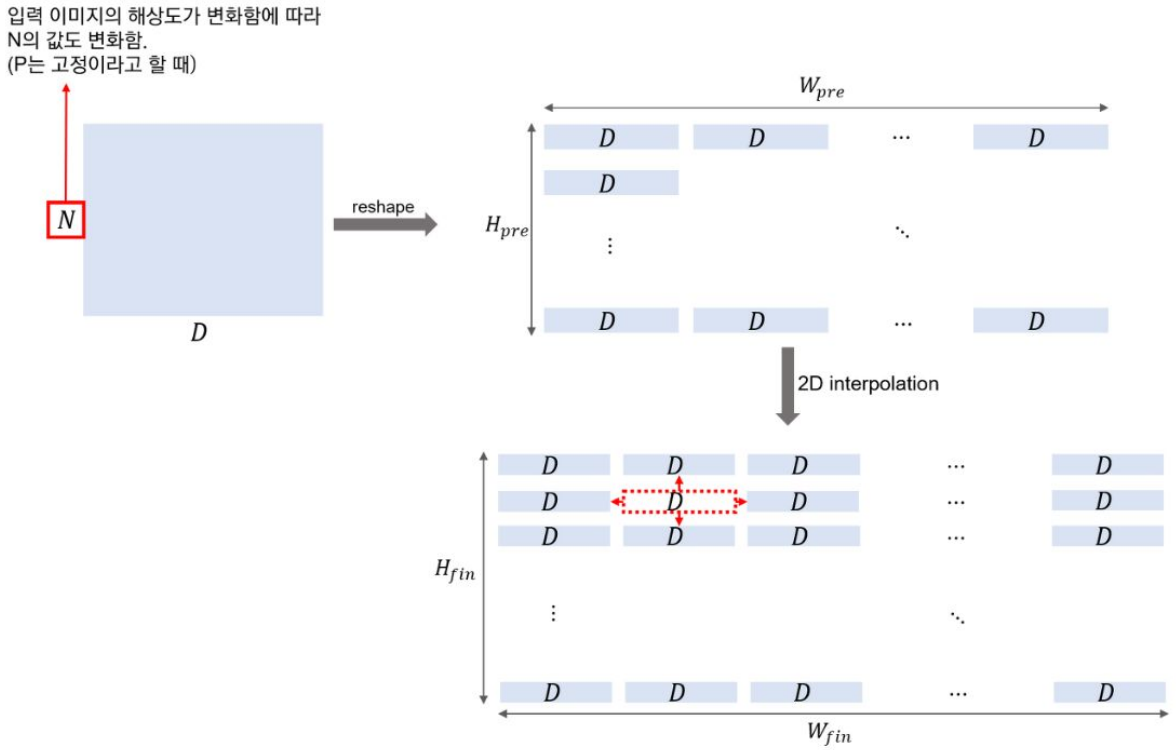
2d interpolation when fine-tuning

- Pre-train보다 높은 resolution으로 fine-tuning하는 것은 종종 도움이된다. 더 높은 resolution의 이미지를 feed할 때 patch크기가 동일하게 유지되므로 sequence length가 더 길어진다. Vision Transformer는 임의의 sequence length를 처리할 수 있지만 pre-trained position embedding은 의미가 없을 수 있다.

Part 2

Proposed Method

Fine-Tuning and Higher Resolution



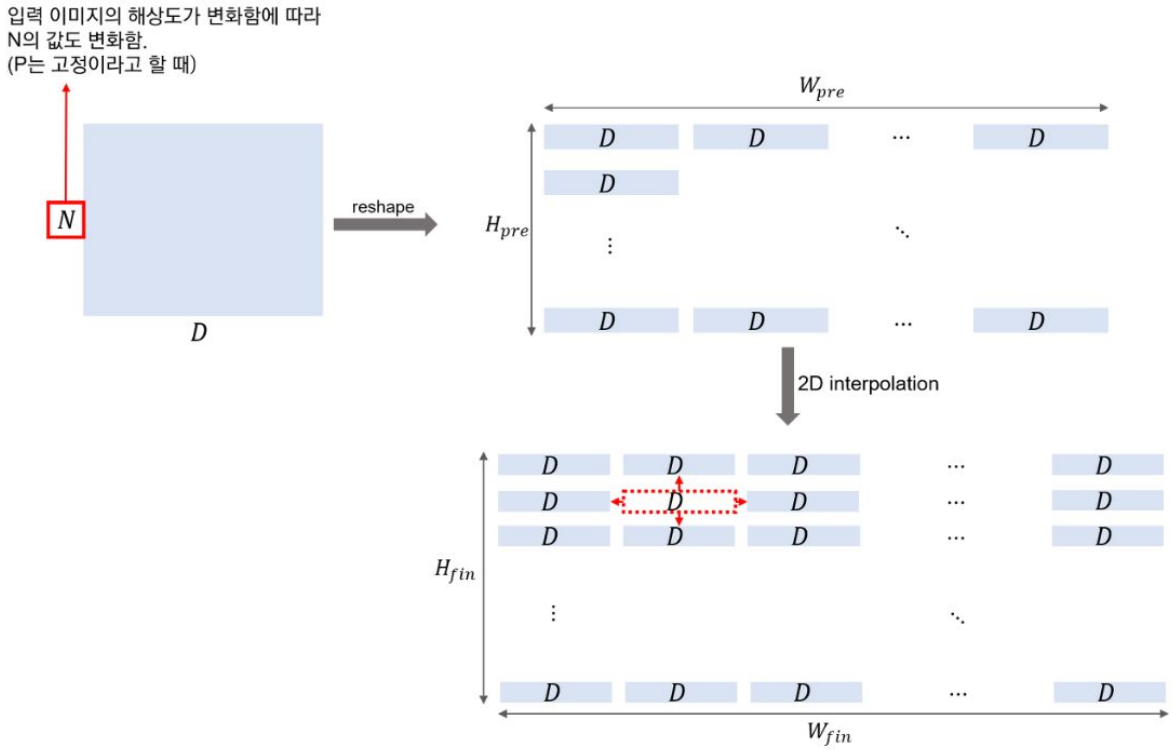
2d interpolation when fine-tuning

- 이를 극복하고자 위치 임베딩이 사전학습에서 학습되고 그것을 fine-tuning에 적용될 때에 그것의 원본 이미지에서의 위치를 기준으로 하는 2D interpolation을 적용하고자 했다.
- 다시말해 위치 임베딩이 fine-tuning시에 변화되기는 하지만 종합적으로 어느 정도의 위치인지 변환시켜줌

Part 2

Proposed Method

Fine-Tuning and Higher Resolution



2d interpolation when fine-tuning

- 이전 같은 해상도 조정 및 Patch extraction이 image의 2차원 구조에 대한 inductive bias가 ViT에 수동적으로 주입되는 유일한 과정입니다.
- 즉, 2D 위치에 대한 inductive bias가 없는 transformer에게 위치에 대한 개념을 주입시켜주는 것입니다.

Experiment

Part 2

Experiment

Datasets

Pre-trained Dataset	# of Classes	# of Images
ImageNet-1k	1k	1.3M
ImageNet-21k	21k	14M
JFT	18k	303M (High resolution)

- ViT는 위와 같이 class와 이미지의 개수가 각각 다른 3개의 데이터셋을 기반으로 pre-train됩니다.
- 아래의 benchmark tasks를 downstream task로 하여 pre-trained ViT의 representation 성능을 검증합니다.
  - ReaL labels, CIFAT-10/100, Oxford-IIIT Pets, Oxford Flowers-102
  - 19-task VTAB classification suite

Part 2

Experiment

Model Variants

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.

- ViT는 위와 같이 총 3개의 volume에 대하여 실험을 진행하였으며, 다양한 패치 크기에 대해 실험을 진행합니다.
- Baseline CNN은 Batch normalization layer를 group normalization으로 변경하고 standardized convolutional layer를 사용하여 transfer learning에 적합한 Big Transformer(BiT)구조의 ResNet을 사용함

Part 2

Experiment

Training & Fine-tuning

- pre-training에는 모든 모델을 Adam optimizer를 활용했으며 하이퍼파라미터 세팅은 β_2 = 0.999, batch_size = 4096로 두었고, weight decay를 적용하였는데 이것이 모든 모델에 대해 transfer시 유용한 도움을 주는것을 발견하였다.
- 학습률 스케쥴로는 linear learning rate warmup and decay를 사용하였다.
- 한편 fine-tuning에는 SGD with momentum 옵티마이저를 사용했고, batch_size = 512으로 두고 훈련하였다.

Part 2

Experiment

Metrics

- 연구진은 성능을 평가하기 위해 downstream 데이터셋에 대한 결과를 few-shot accuracy 혹은 fine-tuning accuracy로 성능 비교를 진행했다. 이때 Fine-tuning acc는 각 모델의 각 데이터셋에 대한 성능을 반영한다고 볼 수 있다.
- Few-shot accuracy는 훈련이미지들의 subset의 representation을 $\{-1,1\}^K$ 의 타겟벡터로 매핑하는 규제된 linear regression 문제의 해를 구하는 과정에서 얻어진다.
- 주로 fine-tuning 성능으로 비교에 집중했으나 fine-tuning이 너무 고비용인 경우 보다 가볍게 평가할 수 있도록 few-shot accuracy를 사용했다.

Part 2

Experiment

4.2 Comparison to state of the art

SOTA CNNs와 ViT largest models과 비교

두개의 비교 포인트 비교 포인트

- 대규모 ResNets 지도 전이 학습된 Big Transfer
  여기에 보고된 다른 데이터셋(ImageNet 및 그 상위 집합, 그리고 JFT 데이터셋)에서 최고 성능
- ImageNet과 레이블이 제거된 JFT300M에서 반지도 학습을 사용하여 훈련된 대규모 EfficientNet인 Noisy Student (당시 ImageNet에서 최고 성능)

Part 2

Experiment

1. 평가지표: accuracy

	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

훈련에 사용된 TPU v3 코어(칩당 2개)의 수를 훈련 시간(일)과 곱한 값 = TPUv3-core-days

- 세번의 fine-tuning을 평균내어 정확도의 평균과 표준편차를 보고
- JFT-300M 데이터셋에서 사전 훈련된 Vision Transformer 모델들이 모든 데이터셋에서 ResNet기반의 기준 모델들을 능가하면서, 사전 훈련에 훨씬 적은 계산 자원을 사용, 더 작은 공개 ImageNet-21k데이터셋에서 사전 훈련된 VIT도 잘 수행됩니다.
- 이는 Vision Transformer 모델이 이미지 분류 작업에서의 최신 기술보다 우수한 성능을 제공한다는 의미입니다.

Part 2

Experiment

2. 평가 지표: VTAB

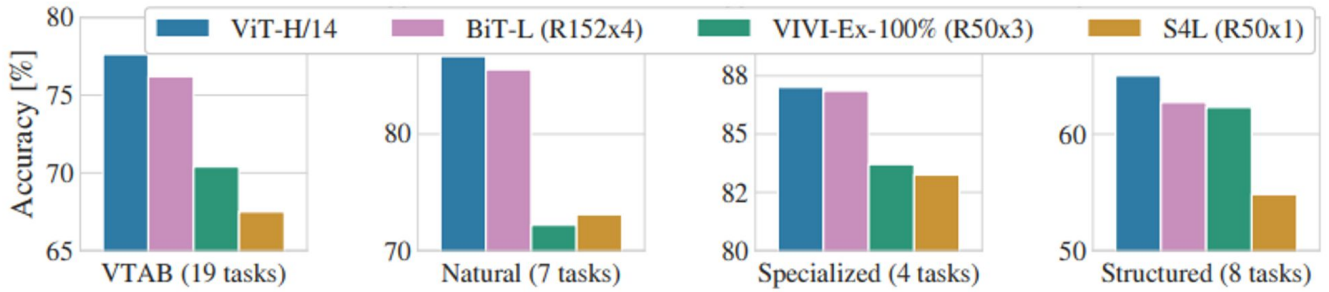


Figure 2: Breakdown of VTAB performance in *Natural*, *Specialized*, and *Structured* task groups.

- VTAB(Visual Task Adaptation Benchmark)은 시각적 작업의 적응성을 평가하기 위한 벤치마크입니다. 이는 모델이 다양한 시각적 작업으로 얼마나 잘 전이(transfer)할 수 있는지를 측정하기 위해 설계, 1000개의 저데이터를 훈련을 하여 평가

Part 2

Experiment

2. 평가 지표: VTAB

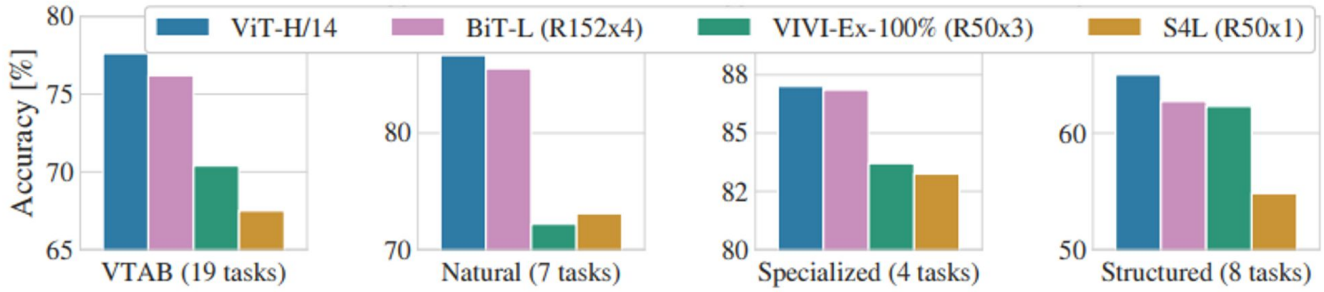


Figure 2: Breakdown of VTAB performance in *Natural*, *Specialized*, and *Structured* task groups.

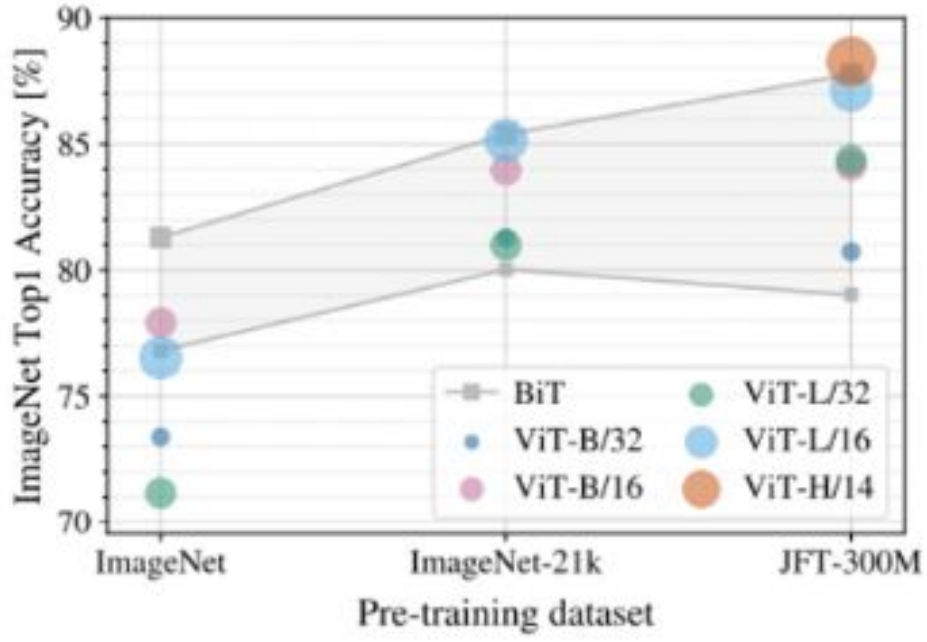
- Natural (자연): 일반적인 시각적 패턴과 객체를 인식하고 분류하는
- Specialized (전문화된):전문화된 이미지 내의 패턴을 학습하고 인식할 수 있는지
- Structured (구조화된): 이미지의 공간적 구성을 이해하고, 그 안에서 객체 간의 관계를 파악할 수 있는지

사전 훈련에 훨씬 적은 계산 지원을 사용하였고, 사전 훈련의 효율성은 모델의 구조뿐만 아니라 훈련 일정, 최적화 알고리즘, 가중치 감소 등 다른 매개변수에 의해서도 영향을 받을 수 있음, 하지만 제어된 연구를통해 위와 같은 결론을 얻었습니다.

Part 2

Experiment

2. 평가 지표: 사전 훈련 데이터셋의 양에 따른 ImageNet Top1 Accuracy

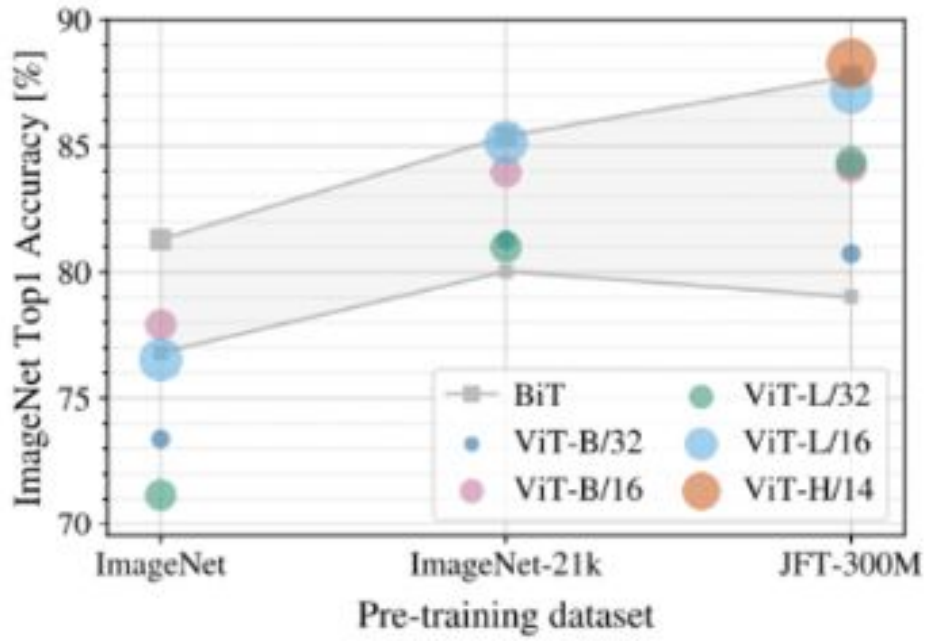


- ResNets와 비교해 봤을 때, ViT가 시각적 작업을 위한 귀납적 편향(inductive biases)이 더 적음에도 불구하고, 데이터셋의 크기가 얼마나 중요한지를 탐구하기 위해 두 가지 실험을 수행함

Part 2

Experiment

2. 평가 지표: 사전 훈련 데이터셋의 양에 따른 ImageNet Top1 Accuracy

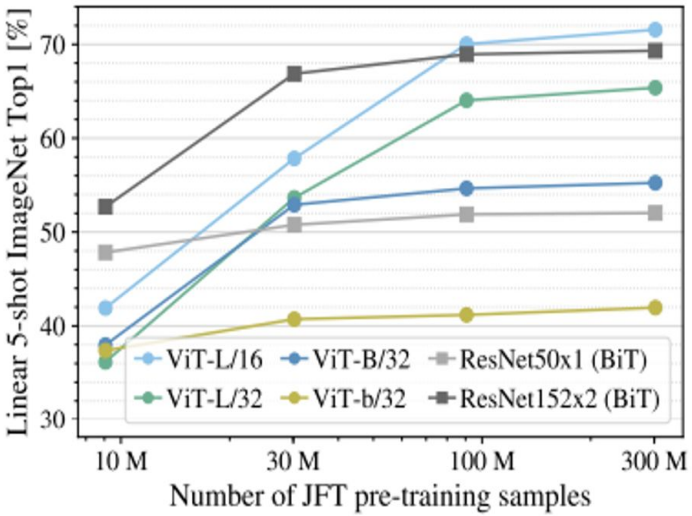


- 첫 번째 실험에서는 다양한 크기의 데이터셋에서 ViT 모델을 사전 훈련
- 더 작은 데이터셋에서의 성능을 향상시키기 위해 세 가지 기본 정규화 매개변수 가중치 감소(weight decay), 드롭아웃(dropout),레이블 평활화 (label smoothing) 통해 최적화
- 가장 작은 데이터셋인 ImageNet에서 사전 훈련될 때, ViT-Large 모델은 적당한 정규화에도 불구하고 ViT-Base 모델에 비해 성능이 떨어짐
- 데이터셋의 크기가 증가함에 따라 더 큰 모델이 더 많은 정보를 학습하고 더 복잡한 패턴을 인식할 수 있는 능력이 향상됨
- 대규모 데이터셋에서는 큰 모델이 그 가치를 발휘할 수 있음

Part 2

Experiment

2. 평가 지표: 사전 훈련 데이터셋의 양에 따른 ImageNet Top1 Accuracy



- 첫 번째 실험에서는 다양한 크기의 데이터셋에서 ViT 모델을 사전 훈련
- 더 작은 데이터셋에서의 성능을 향상시키기 위해 세 가지 기본 정규화 매개변수 가중치 감소(weight decay), 드롭아웃(dropout),레이블 평활화 (label smoothing) 통해 최적화
- 가장 작은 데이터셋인 ImageNet에서 사전 훈련될 때, ViT-Large 모델은 적당한 정규화에도 불구하고 ViT-Base 모델에 비해 성능이 떨어짐
- 데이터셋의 크기가 증가함에 따라 더 큰 모델이 더 많은 정보를 학습하고 더 복잡한 패턴을 인식할 수 있는 능력이 향상됨
- 대규모 데이터셋에서는 큰 모델이 그 가치를 발휘할 수 있음

Qn

감사합니다