Vision Transformer final-report

ByeongGeun Shin[1],
[1]Pusan National University.

1 An Image is Worth 16x16 Words: Transformers for image Recognition at scale(ViT) Paper Summary

1.1 Abstract

Transformer[2] 구조는 NLP 분야에서 주목받고 있지만, 컴퓨터 비전 분야에서는 여전히 CNN이 주로 사용되었다. 이전까지는 Transformer를 컴퓨터 비전에 적용할 때, 전체적인 CNN 구조를 유지하면서 일부 컴포넌트만을 Transformer로 변형하는 경우가 대부분이었다. 전체 구조를 Transformer로 전환한 사례는 드물었다.

본 논문에서는 컴퓨터 비전 분야에 Transformer를 전면적으로 적용하는 새로운 접근 방식을 소개한다. 이미지를 패치의 시퀀스로 변환하고 이를 Transformer 구조에 적용하여 이미지 분류 작업에 활용한다. 방대한 양의 데이터로 사전 학습을 수행하고, 작은 크기의 이미지 인식 작업에 전이 학습을 적용했을 때 이 방법은 뛰어난 성능을 보여준다. 뿐만 아니라, 기존의 CNN 방식에 비해 훨씬 적은 계산 자원을 사용한다.

1.2 Introduction

이전에는 Transformer가 주로 대규모 데이터 세트를 사전 학습(pre-train)하는 데 사용되었고, 특정 작업에 대해 더 작은 데이터 세트로 미세 조정(fine-tune)하여 우수한 성능을 달성했다. 그러나 이러한 Transformer의 장점은 CNN에는 완전히 활용되지 못했다. 여전히 ResNet과 같은 CNN 기반 백본(backbone) 모델들이 이미지 인식 분야에서 최고의 성능(State-of-the-Art, Sota)을 보여주고 있었다.

ViT는 이러한 한계를 극복하고자 이미지를 패치로 분할하고 이들을 선형 시퀀스로 변환하여 Transformer 구조에 입력하는 방법을 제안한다. 이 방법은 이미지 패치들을 NLP에서의 토큰처럼 다루며, 이를 통해 이미지 분류 작업에 적용한다.

1.3 Method

1.3.1 ViT Architecture

아키텍처 그림의 왼쪽 부분에서는 입력 이미지를 Encoder에 넣어주기 전에 Transformer가 연산할 수 있도록 데이터를 변환해주는 모습을 표현해주고 있다. 이렇게 변환된 데이터를 Input Embedding이라고 부른다. 이 Input Embedding은 본격적인 연산을 위해 Transformer Encoder로 들어가게 된다. Vision Transformer에서는 Transformer의 Encoder 부분을 그대로 가져와서 사용한다. 오른쪽 부분을 살펴보면 Transformer와 동일하게 Multi Head Self Attention을 주요한 모듈로 하여 구성된 모습을 볼 수 있다.

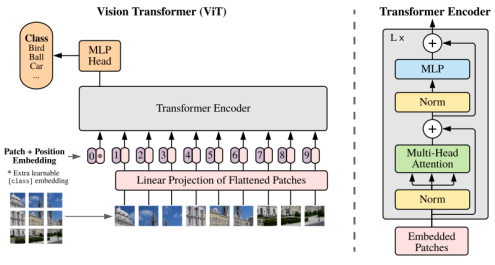

Figure 1: ViT Architecture.

1.3.2 Patch Embedding

본 논문에서는 표준 Transformer 모델의 접근 방식을 따르며, 이 모델은 원래 1차원 토큰 임베딩 시퀀스를 입력으로 받는다. 이에 착안하여, 논문에서는 2차원 이미지를 Transformer 모델에 적용하기 위해 먼저 이미지를 2차원 패치들로 변환하는 방식을 제시한다.

Transformer 모델의 특징 중 하나는 모든 입력과 출력이 동일한 차원을 가진다는 것이다. 이를 기반으로, 논문에서 제안된 방식은 2차원 이미지 패치들을 먼저 시퀀스 형태로 Flatten 하고, 이를 D차원으로 매핑한다. 이 과정을 통해 생성된 D차원의 임베딩은 기존 Transformer에서 사용하는 학습 가능한 임베딩과 동일한 형태를 가지므로, 이를 '패치 임베딩'이라고 한다.

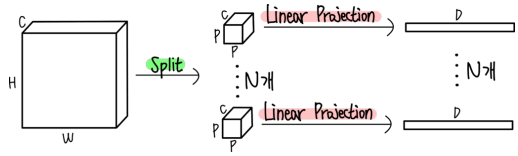

Figure 2: Patch Embedding Struct.

위의 이미지 패치에 더 나아가, learnable embedding vector인 class token을 만들고, 이를 다른 token들에 concat해준다(10xD). Transformer모델에 특성상 transformer layer를 거친 output이 input과 동일한 shape를 가지는데 이렇게 나온 Transformer Encoder의 최종 output에서 class token에 해당하는 vector만을 classification header의 input으로 사용한다.

1.3.3 Position Embedding

위치의 정보가 없는 Transformer의 특성에 따라 본 논문은 위치 정보를 패치 임베딩에 추가한다. 2D Position embeddings이 성능 향상이 그다지 없었기 때문에 1D Position embeddings을 사용한다.

패치 임베딩 과정에서는 이미지의 해상도가 커지더라도 패치의 크기는 동일하게 유지되어, 메모리의 한계 내에서 긴 이미지 시퀀스를 생성할 수 있다. 그러나 이미지의 해상도에 따라 변하는 시퀀스는 사전 학습된 Position embeddings의 의미를 잃어버릴 수 있다. 이를 해결하기 위해 위치 정보에 2차원 보간(2D interpolation)을 적용하여 위치 임베딩을 조정한다.

1.3.4 Inductive Biases

위와 같이 self-attention으로만 이루어진 vision transformer는 CNN 구조에 비해서 더 작은 image-specific inductive bias를 가진다. 대표적인 inductive bias로는 첫번째로 어떠한 픽셀은 가까이 있는 neighborhood 픽셀들에 영향을 많이 받고 거리가 멀어질 수록 그 영향도가 떨어진다는 "locality"가 있고, 두번째로는 object가 x, y축으로 이동하거나 회전을 주더라도 같은 object로 인식하는 "translation invariant"가 있다.

inductive bias가 적다는 것은 그만큼 contraint 없이 이미지 전체에서 정보를 얻을 수 있다는 장점이 있지만 optimal parameter를 찾기 위한 space 또한 커져버리기 때문에 데이터가 충분하게 많지 않으면 학습이 잘 안되는 문제가 발생한다. 때문에 ViT는 large datasets(ex. 14M-300M images)로 ViT를 pretrained 시키고 이를 사용하여 specific task with fewer datapoints에 transfer learning을 한다.

1.4 Experiment

사전학습 데이터 셋은 ImageNet-1k(1.3M), ImageNet-21k(14M), JFT-18k(303M)을 사용하였고 평가 데이터 셋으로 19-Task VTAB - 3가지 하위 요소 Natural, Sepcialized, Structred를 사용하였다.

	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

Figure 3: Comparison to state of the Art.

논문 당시 state-of-art 모델인 BiT[1]와 Noisy student 모델과 성능 비교를 한 표이다. ViT 모델의 / 옆에 있는 숫자는 이미지 패치 사이즈이다. Vit-Huge 모델을 썼을 때 대부분의 task에서 가장 좋은 성능을 얻었고, ViT에 사전 훈련한 모델들이 기존 모델의 성능을 능가하는 것과 학습이 더 빠른 것을 볼 수 있다.

Vit를 pretraining할 때 사용하는 데이터셋의 크기에 따라서 결과가 어떻게 달라지는지에 대한 실험도 진행했는데, 작은 데이터셋(ex. imagenet)의 경우, VIT가 BiT보다 under-perform하고 데이터셋의 크기가 커지면서 BIT보다 성능이 높아지는 것을 볼 수 있었다.

마지막으로, 모델들의 scale을 맞춘 후에 성능을 비교하는 실험에서도 VIT가 ResNet(BIT) 대비 performance/compute trade off가 좋고 Hybrid가 작은 computation scale을 가질 때 VIT의 성능을 outperform 했다.

2 An Image is Worth 16x16 Words: Transformers for image Recognition at scale(ViT) Practice

2.1 Introduction

Vision Transformer(ViT)는 자연어 처리에서 성공을 거둔 트랜스포머 모델을 이미지 처리에 도입한 혁신적인 모델이다. 이미지를 작은 패치 단위로 나누고, 이를 트랜스포머 입력으로 처리함으로써 CNN의 국소적 특징 학습을 넘어 긴 거리의 의존성을 학습할 수 있다. ViT는 데이터 병렬 처리 성능이 뛰어나며, 사전 학습된 가중치를 활용할 경우 다양한 데이터셋에서 탁월한 성능을 발휘한다. 본 실험에서는 사전 학습된 ViT 모델을 활용하여 이미지 분류 작업을 수행하고, ViT의 핵심 구조와 작동 방식을 이해하는 것을 목표로 한다.

2.2 Practice Method

- 환경 설정: PyTorch와 timm 라이브러리를 사용하여 ViT 모델을 불러온다. 모델 학습 및 추론을 GPU 환경에서 실행한다.
- 모델 선택: 사전 학습된 vit_base_patch16_224 모델을 사용한다.
- 데이터 처리: 224x224 크기의 입력 이미지를 16x16 패치로 나누어 처리한다.

ViT 모델의 작동 과정은 크게 네 단계로 나뉜다.

2.2.1 이미지 패치로 분할 및 임베딩

224x224 크기의 입력 이미지는 16×16 크기의 패치로 나뉘며, Conv2D 연산을 통해 고정된 차원의 벡터로 변환된다. 이를 통해 14×14개의 패치가 생성되며, 각 패치는 768차원의 임베딩 벡터로 매핑된다.

$$\text{Conv2d}(3, 768, \text{kernel\_size} = (16, 16), \text{stride} = (16, 16))$$

2.2.2 Position Embedding 추가

패치 간 위치 정보를 학습하기 위해 학습 가능한 위치 임베딩(Position Embedding)을 각 패치 임베딩에 더한다. 이는 패치 간 상호작용 학습에 기여하며, 모델이 이미지 구조를 이해하도록 돕는다.

$$\text{PatchEmbedding} + \text{PositionEmbedding}$$

2.2.3 Transformer Encoder 처리

입력된 패치 벡터는 트랜스포머의 다중 헤드 어텐션(Multi-Head Attention)과 완전연결층(Fully Connected Layer)을 통과하며 인코딩된다. 이 과정은 총 12개의 Transformer Encoder 레이어로 구성된다.

$$\text{TransformerEncoder}(x) = \text{MultiHeadAttention}(x) + \text{FeedForward}(x)$$

2.2.4 MLP(Classification) Head를 통한 최종 예측

Transformer Encoder의 출력 중 첫 번째 값(Class Token)을 사용하여 최종 분류를 수행한다. MLP 헤드가 클래스별 신뢰도를 계산하며, 가장 높은 신뢰도를 가진 클래스를 출력한다.

2.3 Dataset

- 데이터셋: ImageNet ImageNet 데이터셋에는 총 1,000개의 클래스가 포함되어 있다. - 테스트 데이터: 실험에서는 Santorini 이미지 하나를 사용하여 모델이 "church, church_building"으로 올바르게 분류하는지 확인하였다.

2.4 Mission별 구현 방법

2.4.1 이미지를 패치로 나누기

224x224 크기의 이미지를 16x16 크기의 패치로 나누기 위해 Conv2D 연산을 사용하였다.

```
self.img_size = img_size
self.patch_size = patch_size
self.grid_size = img_size // patch_size
self.num_patches = self.grid_size ** 2

self.proj = nn.Conv2d(in_chans, embed_dim, kernel_size=
  ↪ patch_size, stride=patch_size)
```

2.4.2 Position Embeddings

각 패치의 위치를 알려주기 위해, 학습 가능한 position embedding을 Patch embedding vector에 더함.position embedding vector는 이미지 내에서 높은 유사성을 보이는 distance를 학습한다.

```
curr_patch = pos_embed[0, i].unsqueeze(0)
all_patches = pos_embed[0, 1:]
sim = cos(curr_patch, all_patches)
sim = sim.view(14, 14).cpu().detach().numpy()

transformer_input = torch.cat((model.cls_token, patches)
  ↪ , dim=1) + pos_embed
```

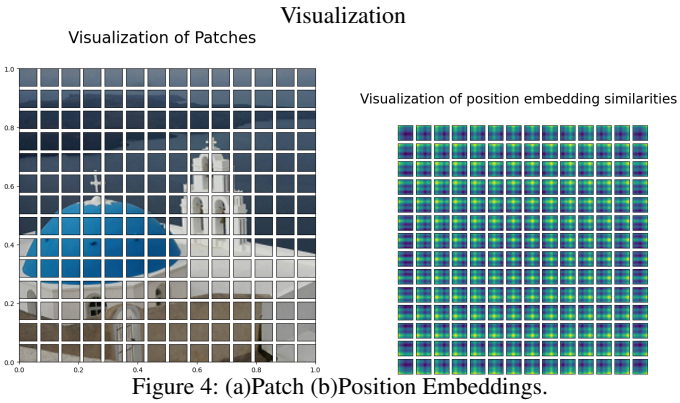

Figure 4: (a)Patch (b)Position Embeddings.

2.4.3 Transformer Encoder

Transformer Encoder는 다중 헤드 어텐션을 통해 각 패치 간 관계를 학습한다. 입력 벡터는 FC 레이어를 통과하여 Q, K, V로 변환되고, 어텐션 계산을 통해 인코딩된다.

```
print("Input␣tensor␣to␣Transformer␣(z0):␣",
  ↪ transformer_input.shape)
x = transformer_input.clone()
for i, blk in enumerate(model.blocks):
   print("Entering␣the␣Transformer␣Encoder␣{}".format(i)
     ↪ )
   x = blk(x)
x = model.norm(x)
transformer_output = x[:, 0]
print("Output␣vector␣from␣Transformer␣(z12-0):",
  ↪ transformer_output.shape)
# Input tensor to Transformer (z0): torch.Size([1, 197,
  ↪ 768])
# Output vector from Transformer (z12-0): torch.Size([1,
  ↪  768])
```

2.4.4 Attention

Multi-head attantion을 위해 qkv를 여러개의 q, k, v vector들로 나눔. Attention Matrix 시각화를 통해 Vision Transformer 모델의 Attention Head들이 이미지의 특정 부분에 어떻게 주목하는지를 보여줌. 각 Heatmap은 각 Attention Head의 주목도를 시각적으로 나타내며, 밝은 부분은 높은 주목도를, 어두운 부분은 낮은 주목도를 의미함.

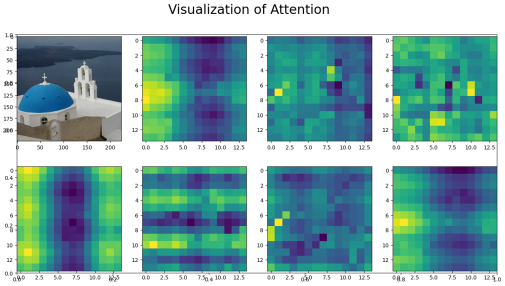

Figure 5: Visualization of Attention.

2.4.5 MLP(Classification) Head

Transformer Encoder의 출력 중 첫 번째 값을 MLP에 입력하여 최종 분류를 수행한다.

```
# MLP Head
print("Classification␣head:␣", model.head)
result = model.head(transformer_output)
result_label_id = int(torch.argmax(result))
plt.plot(result.detach().cpu().numpy()[0])
plt.title("Classification␣result")
plt.xlabel("class␣id")
print("Inference␣result␣:␣id␣=␣{},␣label␣name␣=␣{}".
  ↪ format(
   result_label_id, imagenet_labels[result_label_id]))
```

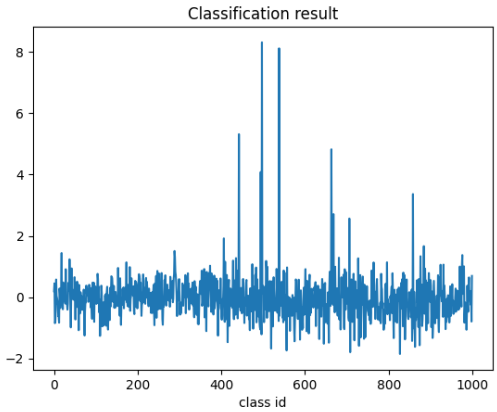

Figure 6: Classification result.

2.5 Training

사전 학습된 ImageNet 모델을 사용하였으며, 추가적인 학습 과정은 수행하지 않았다. 사전 학습된 가중치는 ImageNet 데이터셋을 기반으로 학습되었다.

2.6 Inference

테스트 이미지(Santorini)를 입력하여 ViT 모델의 분류 결과를 확인하였다.모델은 "church, church_building"으로 이미지를 정확히 분류하였으며, 분류 결과는 1000개의 클래스에 대한 신뢰도를 나타내며, 가장 높은 신뢰도를 가지는 클래스 ID는 497이다. 시각화된 결과 그래프는 각 클래스에 대한 모델의 출력 값을 보여준다. 이 결과를 통해 모델이 입력 이미지를 정확히 분류했음을 확인할 수 있다.

2.7 Conclusion

본 실험을 통해 ViT의 구조와 작동 원리를 이해하고, 이를 이미지 분류 작업에 적용하였다. ViT는 CNN 모델과 달리 패치 간의 관계를 효과적으로 학습하며, Attention Matrix 시각화를 통해 이미지 내 중요한 영역에 주목하는 과정을 확인할 수 있었다. ViT는 CNN의 한계를 극복하며, 이미지 처리 분야에서 새로운 가능성을 제시하는 모델임을 확인하였다.

[1] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 2020.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.