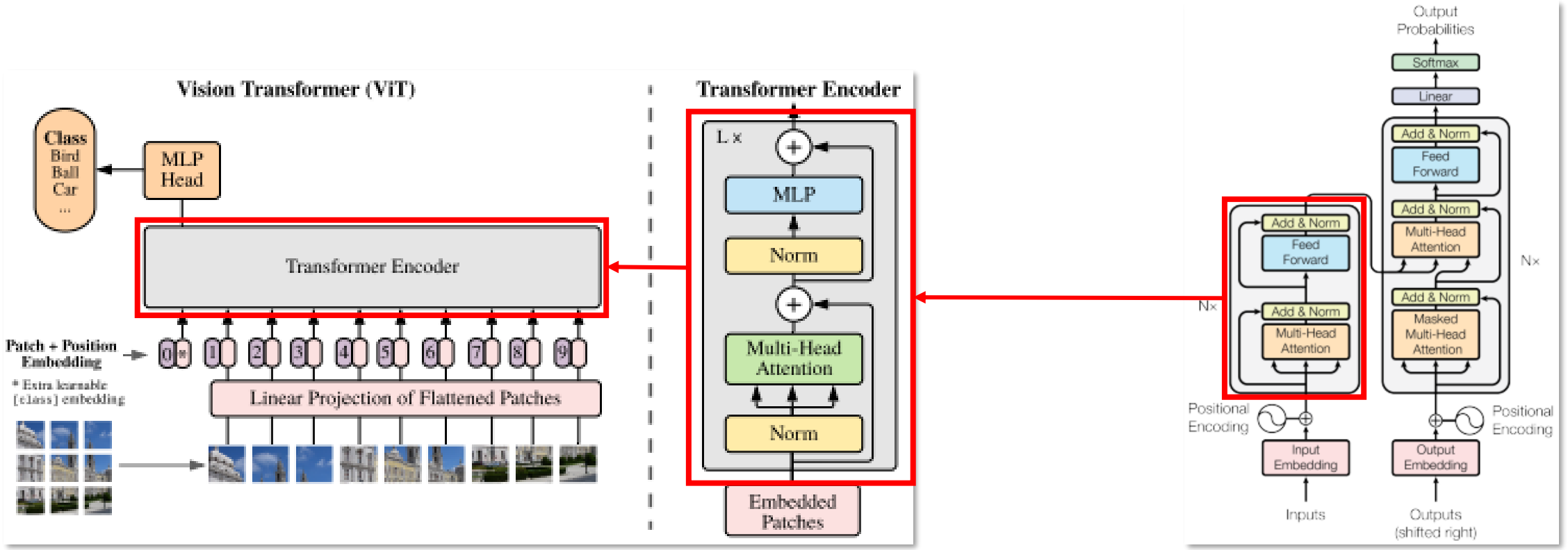
Vision Transformer
병렬 처리 중간 보고서

2025-2 멀티코어 프로그래밍
3조
19011445 조성준
18011650 권혁재

목차

Vision Transformer란?

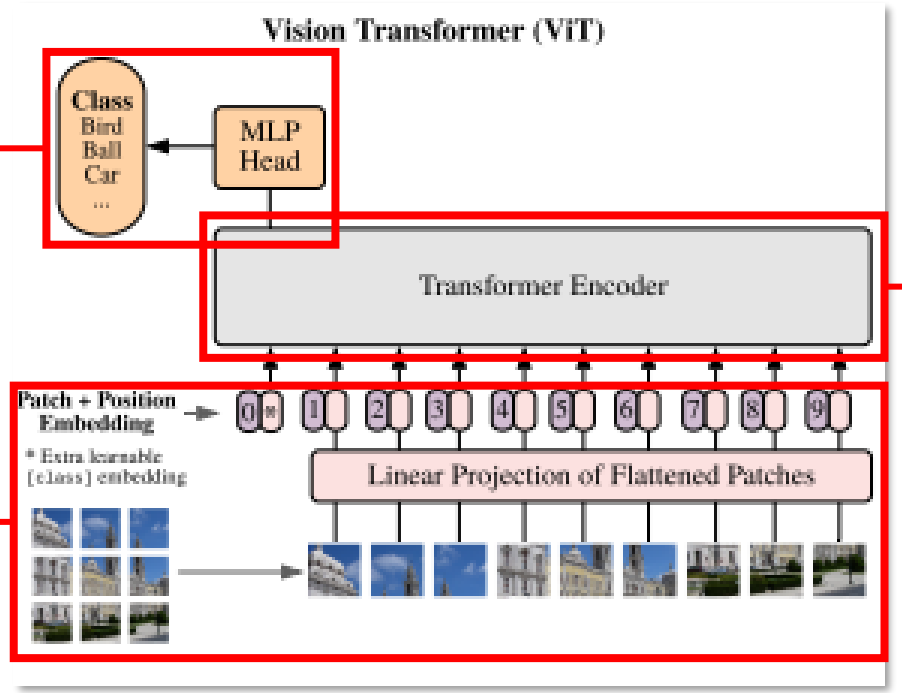
- 이미지 분류 모델
- Transformer 모델의 Encoder부분만을 활용해서 이미지 분류기로 변형



ViT의 실행 과정

- 이미지 전처리 -> 인코더(추론) -> 분류 순서

3. 분류:
인코더에서 나온 최종 결과물의 헤더 정보들을 활성화 함수(소프트맥스)에 넣어서 결과 도출

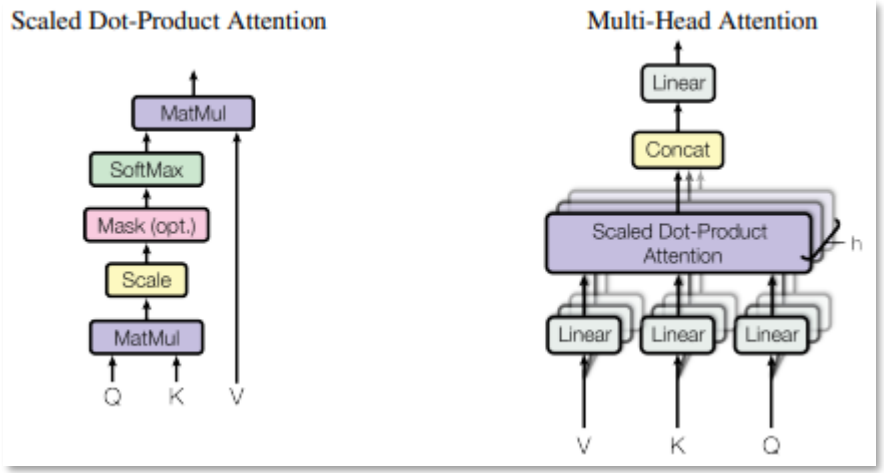


2. 인코더:
멀티헤드어텐션과 완전연결계층으로 이루어진 연산 블록

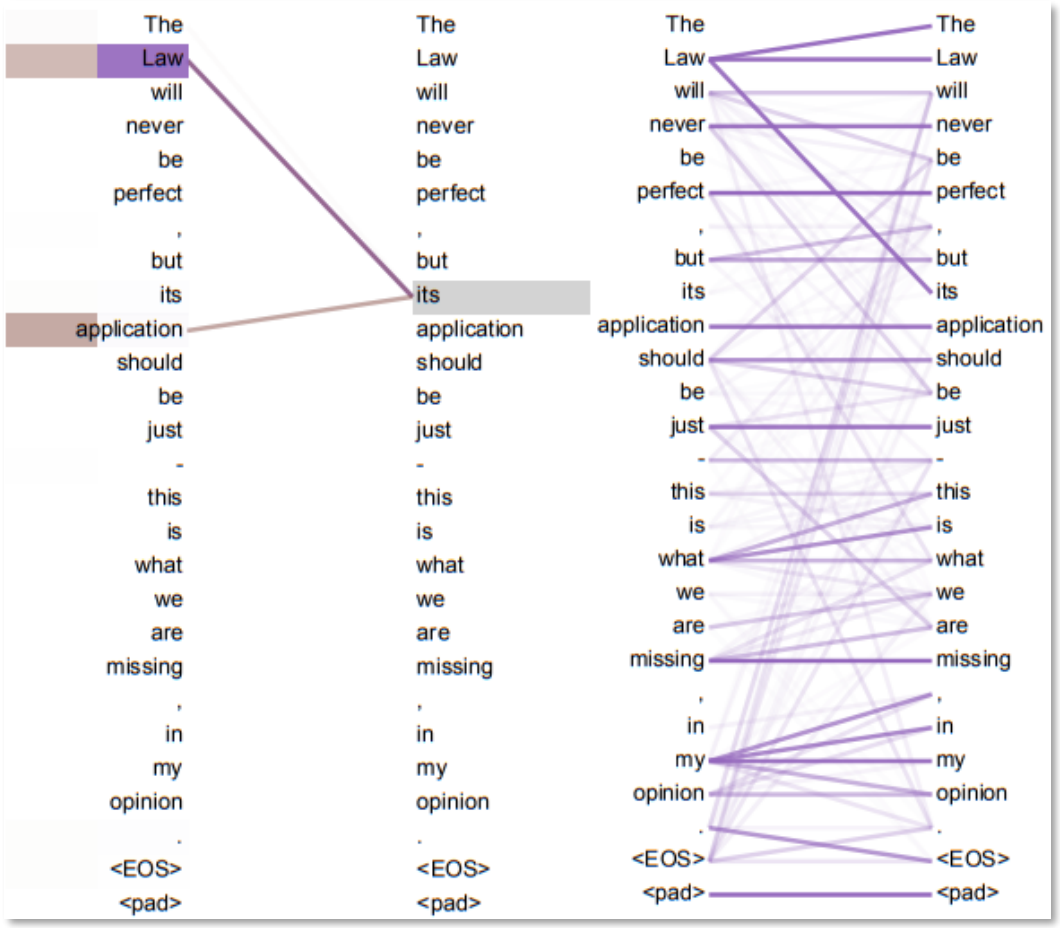
1. 이미지 전처리:
이미지를 여러 패치로 분할 및 형태 변환, 클래스 토큰과 포지션 정보 추가

Encoder 구조 - Multi-Head Attention

- Scaled Dot-Product Attention(Self Attention)을 여러 헤드에 대해 각자 진행한 후 병합
  - Q: 찾고자 하는 정보
  - K: 가지고 있는 정보
  - V: 정보들의 실제 내용
  - Q, K를 셀프어텐션해서 필요한 정보 특정 후 V와 연산해서 결과 도출

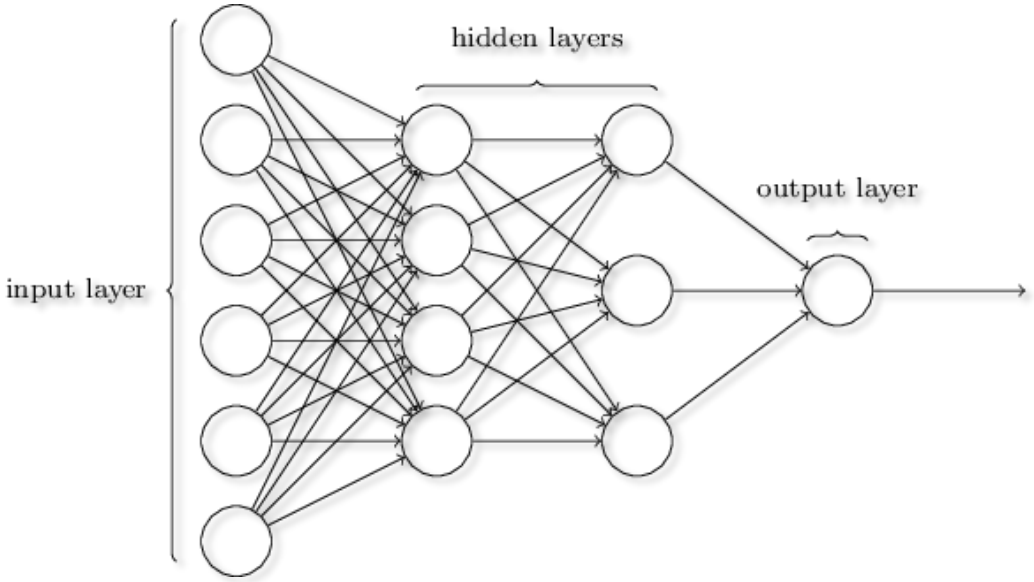


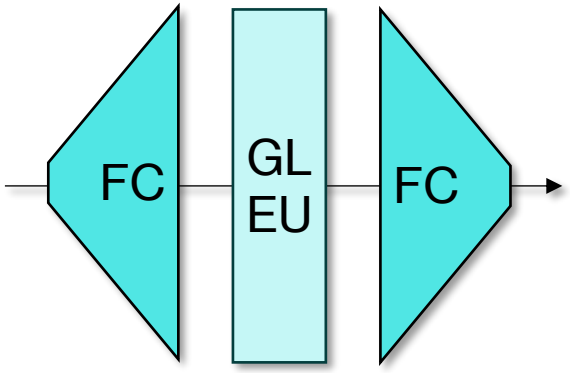
Encoder 구조 - Multi-Head Attention



Encoder 구조 - Multi-Layer Perceptron

- 완전연결계층 – 뇌의 뉴런 구조를 모방하여 인공 뉴런인 퍼셉트론을 만들고, 여러 층으로 쌓아서 연산.
- ViT에서는 FC-GLEU-FC로 구성됨
- MHA로 추출한 특정 정보를 조합하여 비선형적인 고차원 정보로 변환





코드 분석 - 데이터

- /Data/input-100.bin에 입력 이미지 100장이 하나의 파일에 저장되어있음.
  - Network.c의 load_image_data()함수가 이미지 입력 담당
  - 각 이미지에 헤더 정보(n,c,h,w)를 붙이고 ImageData* images 에 나누어서 저장
- /Network 폴더에 이미 학습된 가중치 파일들 저장되어있음. 이걸 Network* network에 저장(152개)
  - 가중치0: 클래스 토큰
  - 가중치1, 2: 컨볼루션
  - 가중치3: 포지션 임베딩
  - 가중치4~147(12*12개): 인코더 내부
  - 가중치148, 149: 인코더 결과
  - 가중치150, 151: 최종 결과(헤드)
- /Data/opencl_result.txt에 입력값들에 대한 정답이 저장되어있음.
  - 결과값 분류는 const char* imagenet_label의 1000개 중 하나로 결정됨
  - 모델 결과값과 정답을 comparator.c의 comparator()함수로 비교하여 결과 확인

코드 분석 - ViT_seq.c 및 병렬화 처리 계획

- 주어진 ViT_seq를 기반으로 병렬화가 가능한 부분들을 찾아서 병렬화 함으로써 성능 향상 도모 예정
- Cov2D() : 다중 for문으로 이루어져있고, 각 패치마다 독립적으로 시행하기 때문에 병렬화
- flatten_transpose() : Cov2D()보단 덜 오래걸리지만 마찬가지 이유로 병렬화 가능은 할 듯
- **multihead_attn**(), **mlp_block**() : 행렬곱 연산들에 대해 타일링 등 기법 적용 예정
- layer_norm(): 2중 for문이고, 인코더 자체가 여러 번 돌기 때문에 성능에 도움이 될 것 같음
- class_token(), pos_emb(), Residual부분 등 단순 벡터합 부분들이나, 병렬화는 가능하지만 연산량이 엄청 많지 않은 부분들은 병렬화 오버헤드에 따라 굳이 할 필요가 없을 수도 있으므로 **성능 테스트가 필요**할 것 같음

실행 시간 측정

- ViT_seq.c를 기반으로 돌렸을 때 소요 시간 측정 결과 (Intel i5-10400F)
  - Cov2D(): 평균 0.11초
  - multihead_attn(): 평균 0.28초
  - mlp_block(): 평균 0.95초
  - 나머지는 거의 미미한 수준
  - 이미지 한 개 당 평균 15초 소요
  - 100개 수행 시 약 26분 정도 소요
- 일단 상술한 3개의 함수를 위주로 먼저 병렬화를 진행
- 특히 multihead_attn()과 mlp_block()는 한 사이클에 12번씩 수행되므로 얼마나 최적화할지가 최종 결과에 있어 중요할 것 같음

```
Cov2D: 0.113000
flatten_transpose: 0.000000
class_token: 0.001000
pos_emb: 0.000000
multihead_attn: 0.282000
mlp_block: 0.960000
Encoder: 1.244000
multihead_attn: 0.284000
mlp_block: 0.956000
multihead_attn: 0.284000
mlp_block: 0.958000
multihead_attn: 0.283000
mlp_block: 0.957000
multihead_attn: 0.283000
mlp_block: 0.955000
multihead_attn: 0.284000
mlp_block: 0.965000
multihead_attn: 0.284000
mlp_block: 0.955000
multihead_attn: 0.283000
mlp_block: 0.963000
multihead_attn: 0.291000
mlp_block: 0.961000
multihead_attn: 0.282000
mlp_block: 0.953000
multihead_attn: 0.286000
mlp_block: 0.962000
multihead_attn: 0.285000
mlp_block: 0.959000
Encoder 12: 14.932000
layer_norm: 0.001000
0 image: 15.049000
```

레퍼런스

- Attention Is All You Need
  - https://arxiv.org/pdf/1706.03762
- AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE
  - https://arxiv.org/pdf/2010.11929
- 다층 퍼셉트론(FFN) 이미지
  - https://dzone.com/articles/deep-learning-via-multilayer-perceptron-classifier