Hierarchical Visual Feature Aggregation for OCR-Free Document Understanding

태그	LLM VLM
URL	https://arxiv.org/abs/2411.05254
날짜	@2025년 2월 19일
세션	☑
작성자	김성현

문제정의

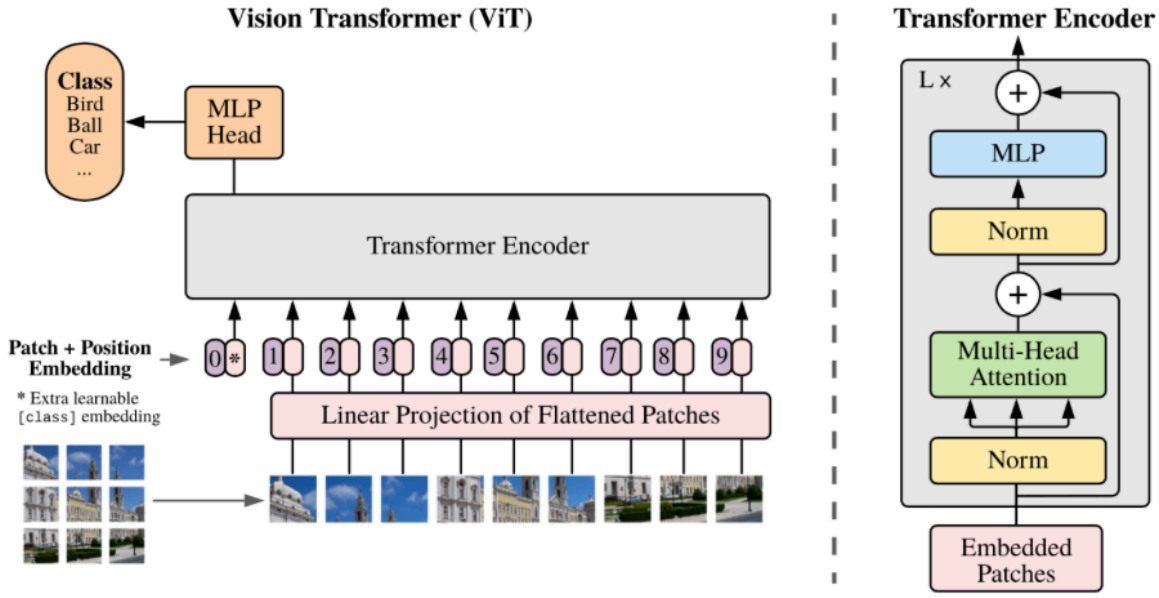
Visual Document Understanding을 잘 하는 모델을 학습하자

오늘의 목표

- VLM이 어떤 종류들이 있는지 훑어보기
- 이미지 이해를 넘어, 문서를 이해하기 위해 어떤 노력들이 있었는지 훑어보기
- 문서를 이해하기 위한 데이터셋은 뭐가 있는지 훑어보기

Background

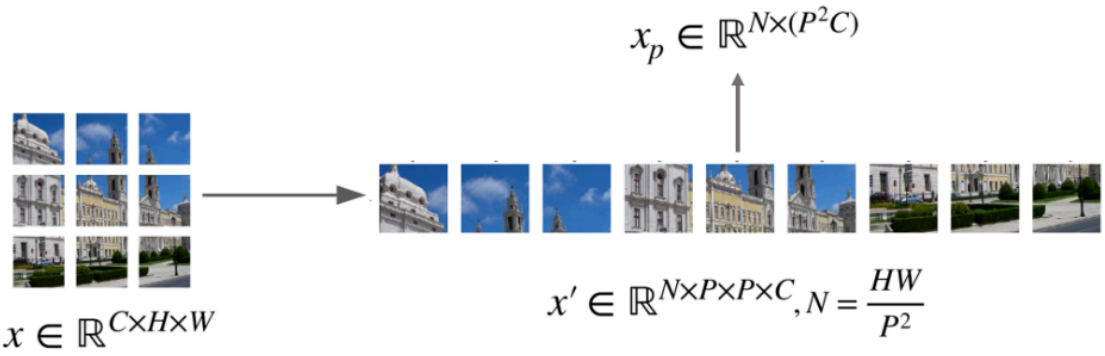
- Vision Transformer (ViT) ← 패치의 개념을 짚고 넘어가봐요



- 학습 데이터: {image - label} paired data
- Step 01: 이미지 쪼게기 (patch)
  - 각 패치는 동일한 사이즈를 가짐
  - 즉, 하나의 이미지를 동일한 사이즈의 patch로 쪼게서 p x p 개의 패치 획득

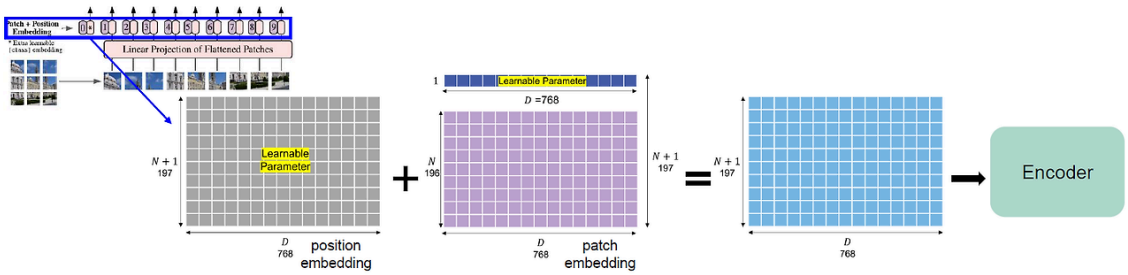
- Step 02: 평탄화 (flatten)
  - 언어모델의 transformers는 token id → word embedding layer → vector화
  - 이미지는 R, G, B로 이루어진 픽셀임
  - 즉, 1개의 패치는 가로길이 x 세로길이 x 3(R, G, B) 만큼의 정보가 나옴

- 즉, 1개의 이미지는 N개의 패치를 가지고, N x 가로길이 x 세로길이 x 3(R, G, B) 만큼의 정보가 나옴



- 1개의 패치를 바라볼 때, P x P x C 로 되어있는 대상을 1D 벡터로 펼침
- 요걸 임베딩 레이어에 입력으로 사용

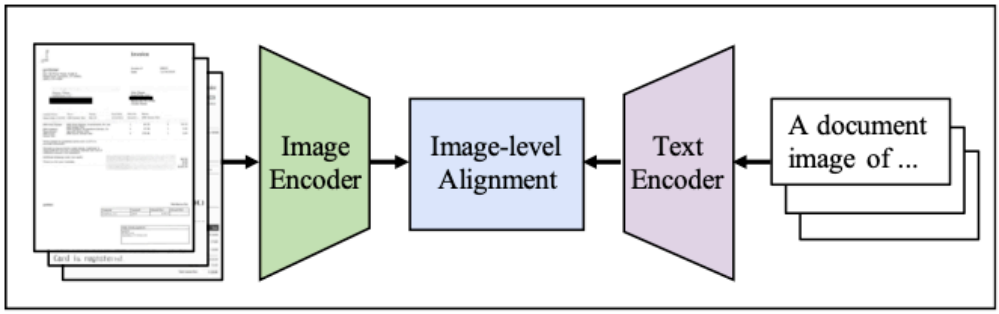
- Step 03: Positional embedding
  - 한 패치 크기를 16으로 가정하고 (P=16)
  - H=W=224
  - 224 × 224 이미지를 16 x 16 패치로 만들면 196개의 패치가 나옴
  - 이걸 768차원으로 임베딩한다면..



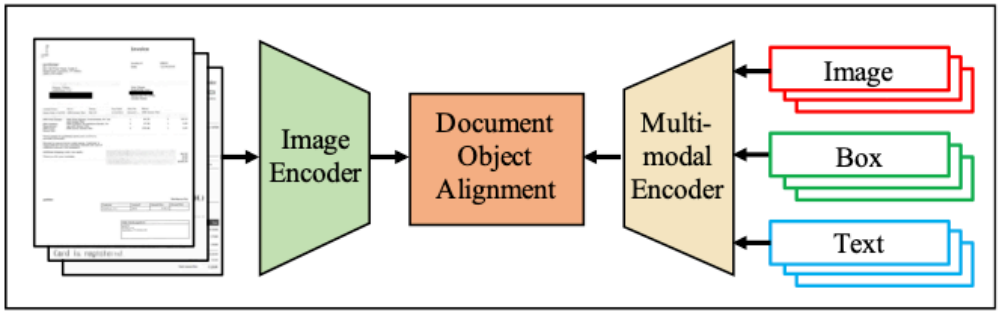
Introduction

- 최근 LVLM (Large Visual-Language Models)가 등장하면서, VDU (Visual Document Understanding) 분야 연구가 활발하게 이루어지고 있음
  - **방법론 1**: Vision 정보를 잘 표현하는 모델을 만들자

- Doco: CLIP 처럼 multi-modal 정보를 하나의 letent space에



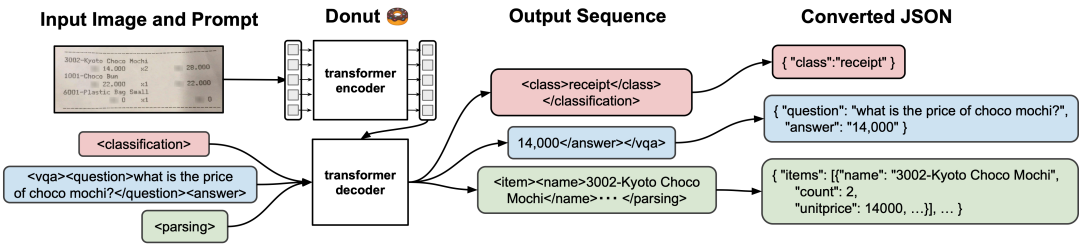
(a) Image-level instance discrimination (CLIP)



(b) Document object discrimination (Our DoCo)

Doco 리뷰

- Donut: 학습 단계에서 OCR 입출력 학습



- 요런 방법론은 은근 좀 적은듯,,,

◦ **방법론 2**: 기존에 학습 된 LLM에 Vision 정보를 먹여서 학습하자

- LLaVa: vision encoder를 사용해서 LLAMA에 학습 → 입력이 embedding layer에 들어가냐.. 어쩌냐,,,

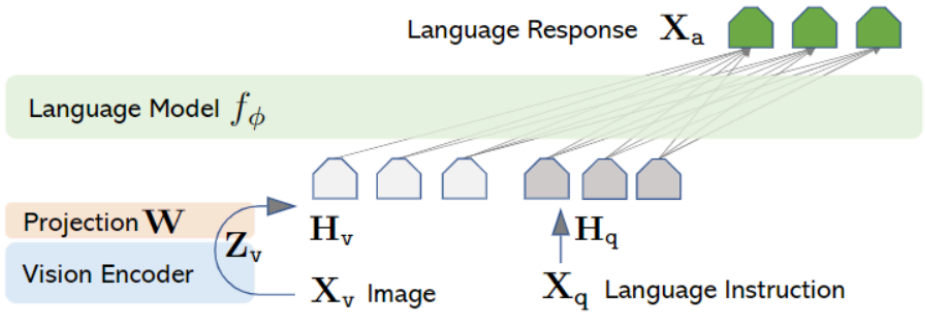
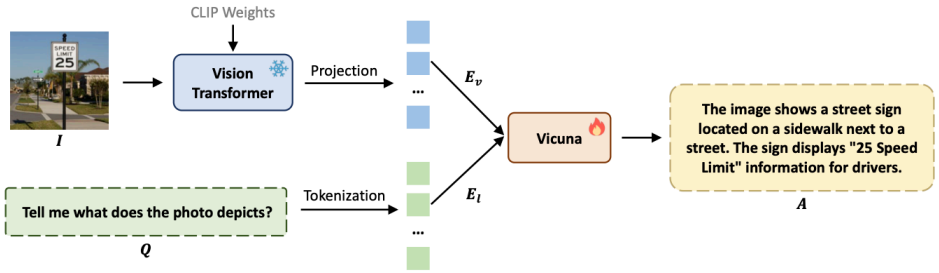
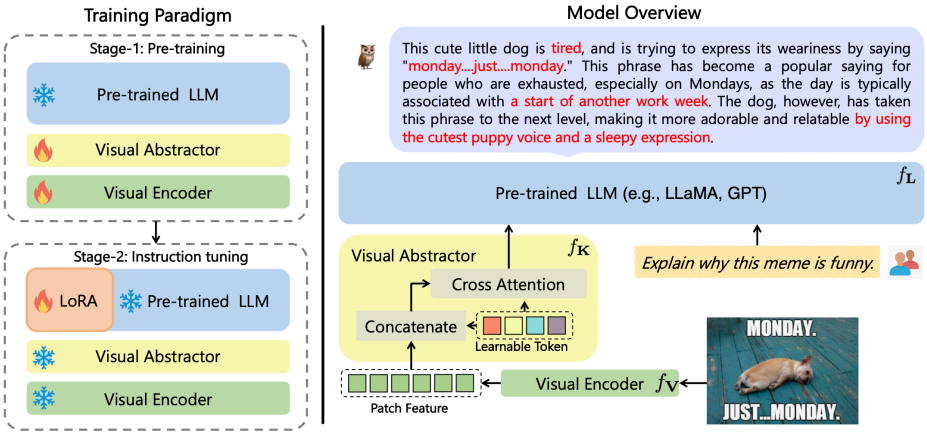


Figure 1: LLaVA network architecture.

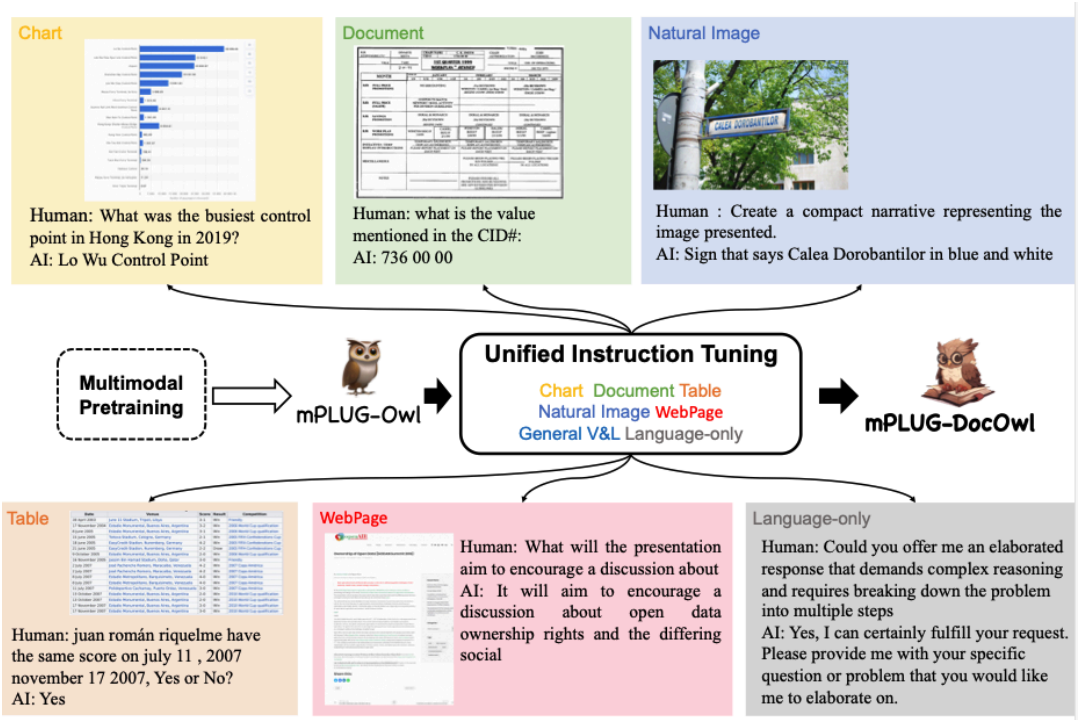
- UniDoc: 다 똑같은 이야기임



- mPLUG-Owl: 똑같..



- mPLUG-DocOwl: document **image comprehension**에 특화 된 데이터셋 구축해서 학습



    - 말고도 넘 많음.. EVLM, GiT, BLIP, Qwen-VL ...

- 하지만 이런 연구도 이미지 내 텍스트의 세부사항을 정확하게 포착하지는 못한다
- 텍스트의 중요 조건은 **(1) 텍스트 내용이 뭔지 알아야하고 (2) 시각적인 구조를 이해해야하고 (3) 각 구조간의 상관관계를 알아야함**
    - 예컨대, 제목은 bold 처리돼서 상단에 위치하고
    - '제목'스러운 스타일의 텍스트로 구성되어 있고
    - 아래 본문은 그 제목을 설명하는 {주제 - 내용} 의 관계성을 포함하고 있음
- 근데 기존 연구의 vision encoder들은.. 일단 이미지 내 텍스트를 디테일하게 분석해내지는 못했고... 보통 CLIP 기반으로 인코더를 만드는데, CLIP은 OCR 수준의 텍스트 분석보다는 이미지가 내포하는 객체 중심으로 학습된 모델이었음
- 그렇기 때문에, 오히려 CLIP 기반의 이미지 인코더를 이용해 VDU로 접근하는게 이상하고, VDU에 필요한 정확하고 세밀한 visual feature를 추출할 수 없음
- 게다가 Vision encoder는 제한 된 (고정 된) 사이즈의 해상도로 이미지를 입력 받음
- 이걸 상황들을 보완하기 위해 온갖 연구에서 OCR 모델을 활용하기 시작했음
    - LayoutLM 시리즈 (v3 까지 나옴)

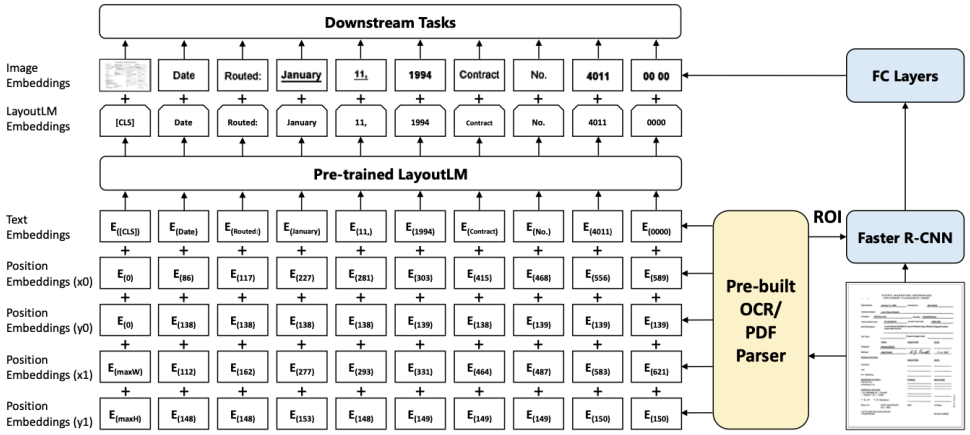
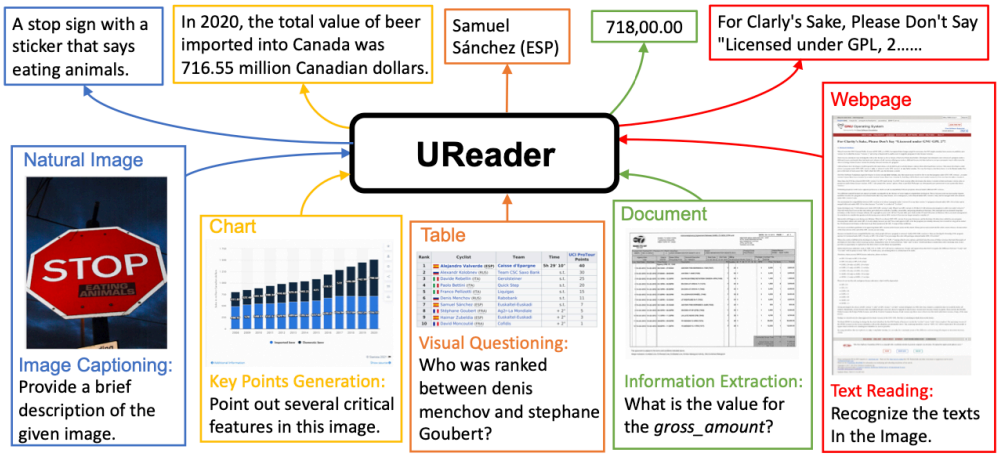


Figure 2: An example of LayoutLM, where 2-D layout and image embeddings are integrated into the original BERT architecture. The LayoutLM embeddings and image embeddings from Faster R-CNN work together for downstream tasks.

  - vision encoder (Faster R-CNN 활용) + OCR 결과 사용
  - Layout 유형 : letter, memo, email, file, file-folder, form, handwritten, invoice, advertisement, budget, news articles, presentation, scientific publication, questionnaire, resume, scientific report, specification

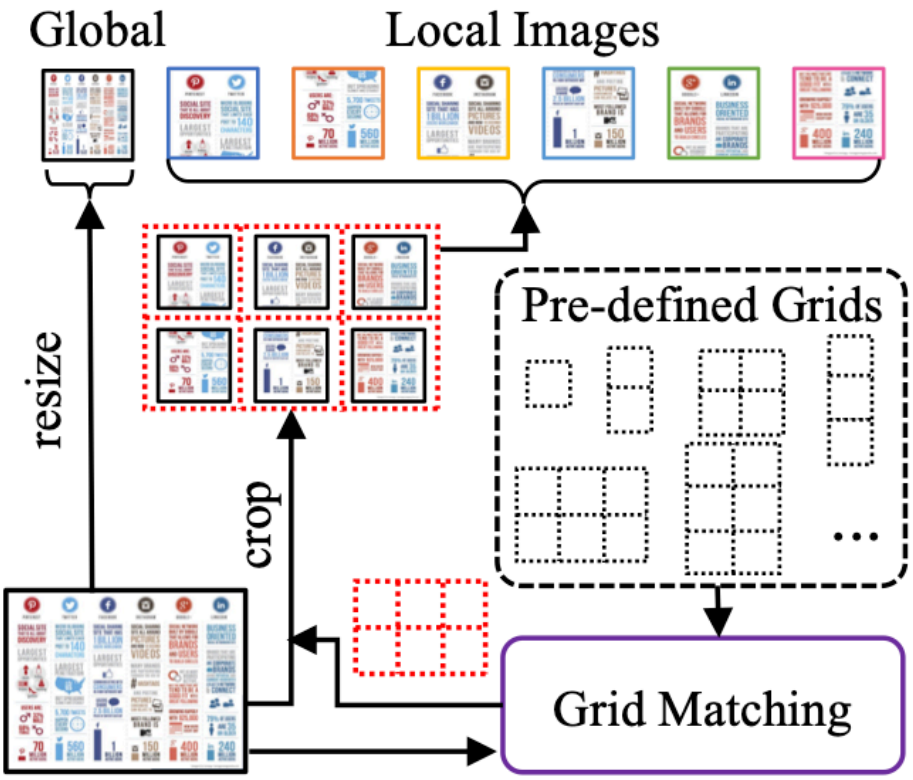
- 근데 이런 연구들, (1) 외부 OCR 엔진이 필요하고 (2) 손글씨에서 인식 성능이 떨어지고 (3) 손상된 문서는 더더욱 떨어지고 (4) OCR 타니까 에러도 전파되면서 연산 비용 증가
- 그래서 등장한 연구들이 OCR-free Document Understanding Models
  - Donut도 그렇고,,
  - UReader: 사전 학습 된 MMLM (mPLUG-Owl)으로 튜닝



  - Flan 처럼.. vision task에 특화 된 instruct 들을 넣어서 더 학습시켰다

- 단순 이미지 캡셔닝 뿐만 아니라, VQA, 키 정보 추출, 표 데이터 분석 등

- 원래는 patch 단위로 입력 받았는데, 이러면 텍스트가 너무 작아지거나 흐려짐
- 그래서 여러 개의 작은 이미지로 분할해서 다양한 크기로 삽입

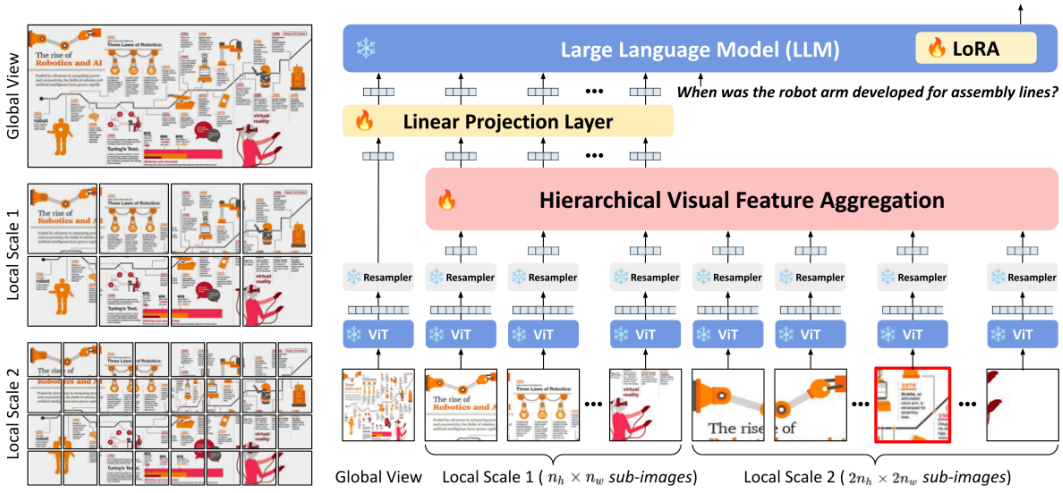


- 분할 기준: Grid matching (Shape-Adaptive Cropping Module)
  - Step 01: 분할을 위한 개수 고르기
    - patch가 지원하는 크기는 고정되어 있음 (e.g. 224 * 224)
    - 원본 해상도를 최대한 유지하는 형태의 grid로 쪼갬
      - 예를 들어, 512 * 1024 이미지를 224 * 224로 축소하면 해상도 차이가 크니까..
      - 대충 3 x 5 정도의 격자 패턴으로 쪼갠다

        - 근데 이게 이미지에 따라 다이나믹하게 골라낸다
        - 그럼에도 224 * 224에 맞추다보니 어느 순간엔 이미지가 찌그러질 수 있음
    - Step 02: 종횡비 유지하기
        - 단순한 크롭 방식은 웹페이지, 표 등이 가진 종횡비를 무시하고 자르고, 정사각형으로 변환 → 텍스트가 찌그러짐
        - 예를 들어 512 * 1024 는 1:2의 종횡비
        - 요런 형태로 비율을 유지하면서 크롭하겠다
- 하지만 이런 연구도 한계점이 있는데
    - patch가 고정 == 텍스트의 인식 해상도도 고정 → 작은 글자나 세부 정보를 포착하지 못할 수 있음
    - 모든 서브 이미지를 다 동일하게 관찰

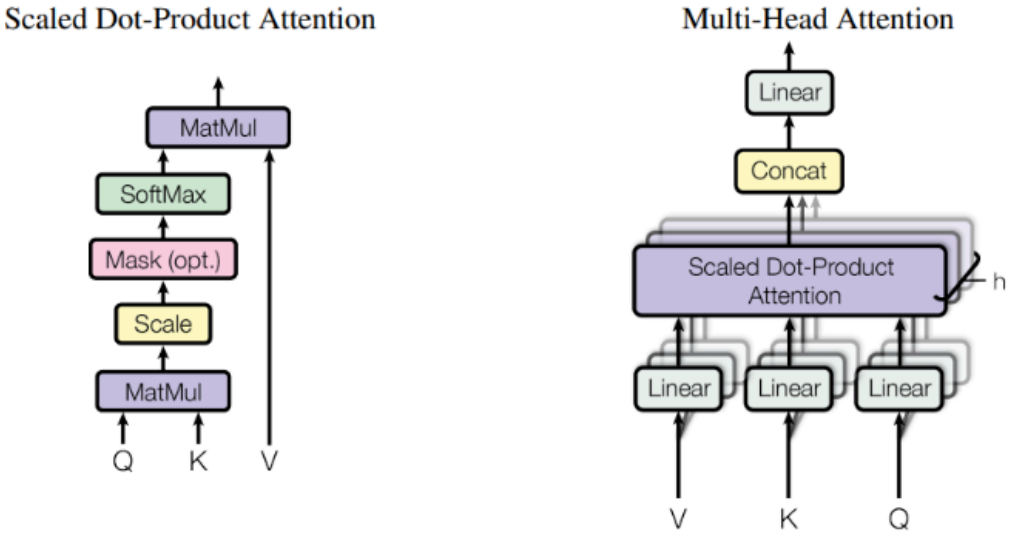
Method

- 그래서.. OCR-free VDU 에서 중요한 핵심은 (1) 텍스트의 char를 잘 이해할 수 있도록 적절하게 cropping 하는게 중요하고 (2) 문서 이해에 특화 된 데이터셋을 사용해야 한다.
- 중간에 special token은 있었나?

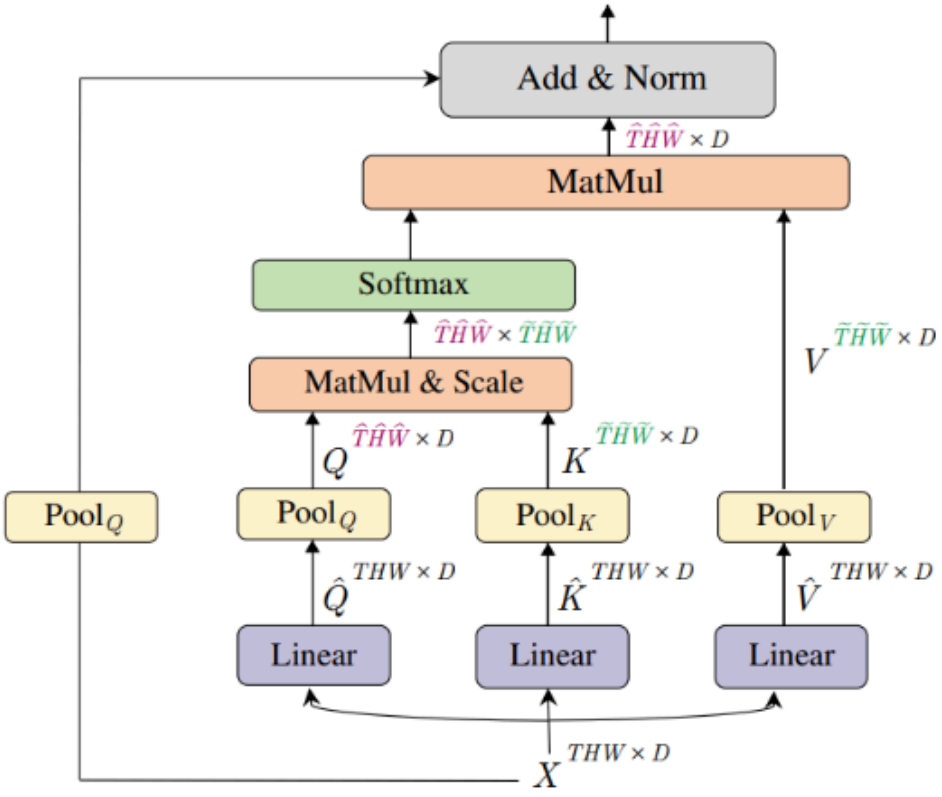


- 이 논문에서의 입력은 크게 3가지로 접근
    - Global view: 문서의 전체적인 맥락 분석

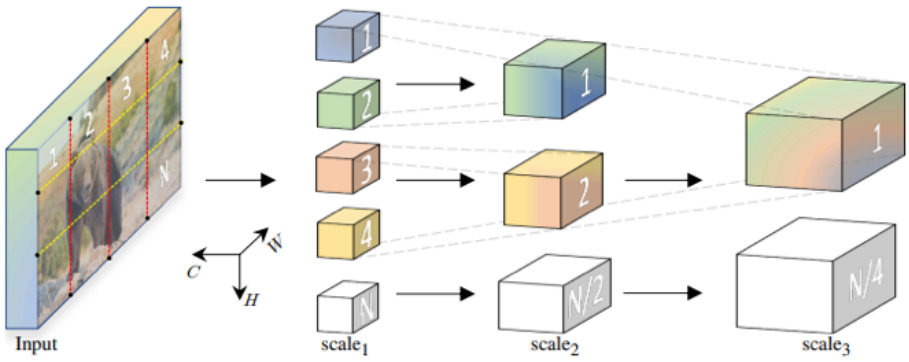
        - patch 사이즈에 맞게 resizing 됨 → 일종의 [CLS] 토큰 역할
    - Local slice 1: 기본적인 patch로 잘린 기본 이미지
        - 동일하게 'Shape-Adaptive Cropping Module' 를 사용한 이미지
    - Local slice 2: 고해상도 서브 이미지 → upscailing 적용한 이미지임
        - 위에서 선정 된 모든 patch를 2배로 upscailing 해서 입력으로 넣음
- 요 입력은 ViT로 가는데.. 논문에서 사용 된 ViT는 MViT (Multiscale vision transformers) 아이디어를 활용함
    - Transformer 구조를 먼저 보면..



    - MViT는...



- Pooling layer가 포함되어 있음
- 개념적으로는 아래 이미지의 컨셉을 유도한 것



- 즉, 고정된 해상도가 아니라, 점점 글로벌 정보를 학습하도록

◦ 좀 더 구체적으로는 아래와 같이 구성을 함

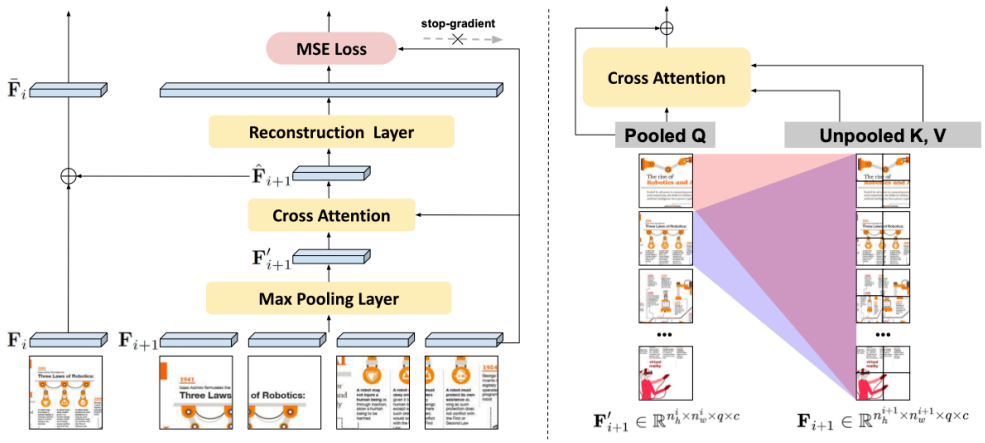


Figure 2: Illustration of the Hierarchical Visual Feature Aggregation (HVFA) module. (Left) HVFA aggregates high-resolution visual features to low-resolution features leveraging feature pyramid structure. (Right) In cross-attentive pooling, each sub-image-feature attends to all of the fine-grained visual features, compressing and preserving more detailed information.

    - 즉, 고해상도 → 맥스 풀링 → 압축된 핵심 정보
        - 물론, 이러면 정보가 손실 될 수 있음.. 그래서 cross attention 과 reconstruction layer 를 이용해서 원본 이미지를 복원하는 형태의 학습을 유도함
        - 즉, 원본을 복원할 수 있을 정도의 핵심 정보를 압축하도록
        - **압축하는 이유: scale 1과 scale 2의 정보를 융합할 수 있어야하며, 연산량도 줄여주고 (압축되니까) 정보는 더 잘 표현하게 됨**
- 자 이제 텍스트를 배워야하는데... 기존 UReader 같은 애들의 문제점
    - 보통 이미지 내에 존재하는 모든 텍스트를 학습하는데 (OCR 사용하던 연구들도 포함)
    - LLM의 길이 제한으로 인해 텍스트가 잘려서 정보 손실이 일어날 수도 있음
    - 모든 텍스트를 다 읽는건 연산량도 크고 비효율적
    - 안중요한 부분까지 학습하게 될 수 있음
- 그래서 제안하는 2가지 방법
    - 텍스트의 일부만 있어도 맥락을 잘 파악할 수 있도록
    - "부분적 텍스트 읽기 (Reading Partial Text, RPT)"
        - 전체 문서를 읽는 것이 아니라, 특정 구간의 텍스트만 선택적으로 읽도록 학습
        - 이를 위해, 전체 텍스트를 "처음(first), 중간(middle), 마지막(last)" 세 가지 유형으로 나누고, 각각의 구간을 읽도록 유도

        - 예를 들어,"이미지 텍스트의 처음 30%를 읽어봐.""이미지 텍스트의 10%에서 55% 사이의 단어를 찾아봐."와 같은 방식으로 훈련 데이터를 생성

유형	예시 질문
처음 (First)	"이미지 텍스트의 처음 30%는 무엇인가?"
중간 (Middle)	"이미지 텍스트의 10%~55%에 해당하는 단어는 무엇인가?"
마지막 (Last)	"이미지 텍스트의 마지막 16%는 어떤 단어인가?"

        - 모델이 텍스트를 무작위로 읽는 것이 아니라, 텍스트의 상대적 위치를 고려하며 읽도록 학습하는 것이 핵심
    - "텍스트 위치 예측 (Predicting Text Position, PTP)"
        - 모델이 특정 텍스트의 문서 내 상대적 위치를 예측하도록 학습
        - 예를 들어,"텍스트 'Invoice No.' 가 문서에서 어느 위치에 있나?"→ "15%~30% 사이에 위치함."이런 식으로 모델이 특정 텍스트가 문서에서 어느 부분(처음, 중간, 끝)에 위치하는지를 예측할 수 있도록 훈련

입력 텍스트	예측 위치
"Invoice No."	"15% ~ 30%"
"Total Amount"	"80% ~ 95%"

학습 데이터셋

- DocVQA
- InfographicsVQA
- DeepForm
- KleisterCharity (KLC)
- WikiTableQuestions (WTQ)
- TabFact
- ChartQA
- VisualMRC
- TextVQA
- TextCaps

평가 지표

데이터셋	평가 지표
DocVQA, InfographicsVQA	ANLS (Average Normalized Levenshtein Similarity) ← 편집거리
DeepForm, KLC	F1 Score
WikiTableQuestions (WTQ), TabFact, TextVQA	Accuracy (정확도)
ChartQA	Relaxed Accuracy ← 5.0% vs 5%
TextCaps, VisualMRC	CIDEr Score ← n-gram 기반

결과

는 논문으로...