Florence-2:
Advancing a Unified Representation for a Variety of Vision Tasks

고병하

Content

About Florence-2

Vision Foundation Model

Florence-2는 Azure AI 팀에서 개발한 차세대 **vision foundation 모델**로, 다양한 컴퓨터 비전 및 비전-언어 작업을 처리할 수 있다.

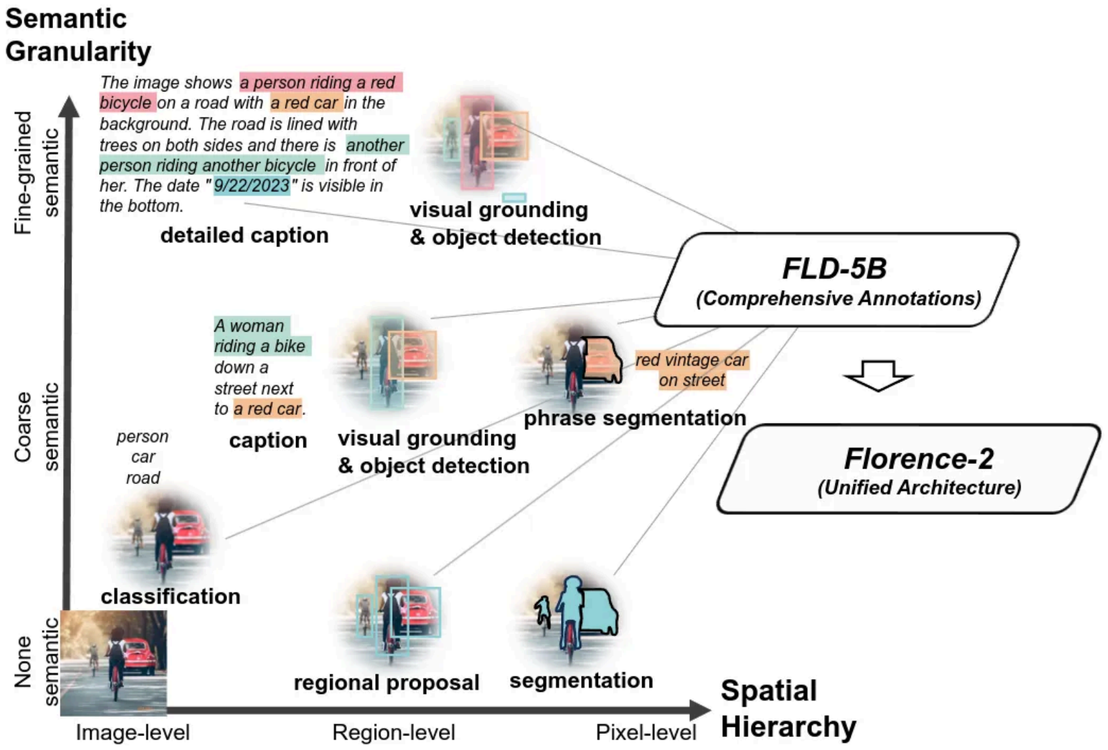
Unified, Prompt-based representation

기존의 대형 비전 모델은 transfer learning에서 우수한 성과를 보였으나, 다양한 작업을 간단한 지시로 수행하는 데는 한계가 있었다.
Florence-2는 여러 downstream 작업을 **통합된 방식**으로 처리할 수 있도록 설계되었다.
또한, **텍스트 프롬프트**를 작업 지침으로 받아들이고, 결과를 **텍스트 형태로 생성**하도록 설계되었다.

FLD-5B Dataset

1억 2천 6백만 개의 이미지와 **54억** 개의 시각적 주석으로 구성된다.
자동화된 Image annotation 및 model refinement의 반복적인 전략을 사용하여 구축되었다.

Florence-2's Capacity



Spatial Hierarchy & Semantic Granularity

공간적 계층 & 의미적 세밀함

목적

NLP와 달리, 컴퓨터 비전은 복잡한 시각 데이터 처리 필요
객체의 위치, 마스크된 윤곽선 등 vision정보를 통해 포괄적으로 이미지를 이해

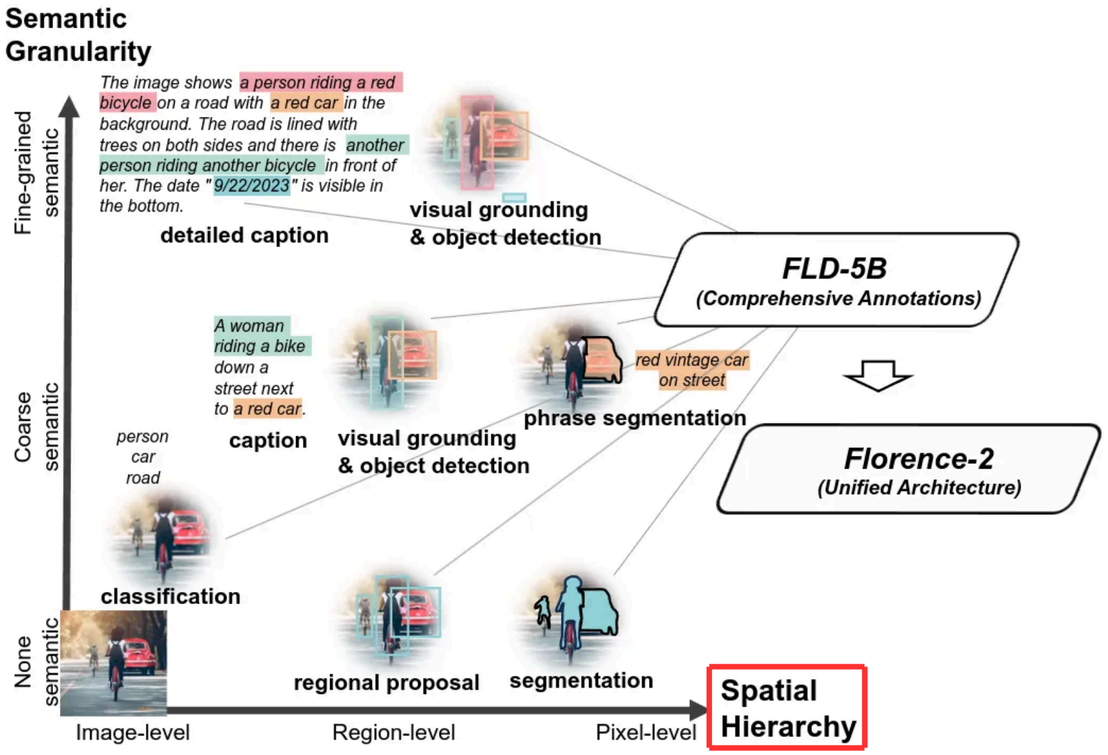
Spatial Hierarchy 공간적 계층

모델은 다양한 크기에서 **공간적 세부 사항**을 구분해야 하며,
이미지 수준 개념에서 세밀한 픽셀 세부 사항까지 모두 이해해야 한다.
이미지 내의 복잡한 공간적 계층을 수용하기 위해서는 **다양한 수준의 세밀함을 처리하는 모델의 능력이 필요**

Semantic Granularity 의미적 세밀함

컴퓨터 비전에서의 보편적 표현은 다양한 의미적 세밀함의 스펙트럼을 포함
모델은 **고급 캡션에서 세부적인 설명까지**,
다양한 **의미에 대한 다재다능한 이해**가 필요

Florence-2's Capacity



Spatial Hierarchy

공간적 계층

Image-level

High-level 의미를 포착하고 언어적 설명을 통해 이미지에 대한 포괄적인 이해를 촉진
이미지의 **전체 맥락**을 이해하고, 언어 영역에서 **의미적 관계와 맥락의 뉘앙스**를 파악 가능
ex) Classification, Captioning, …

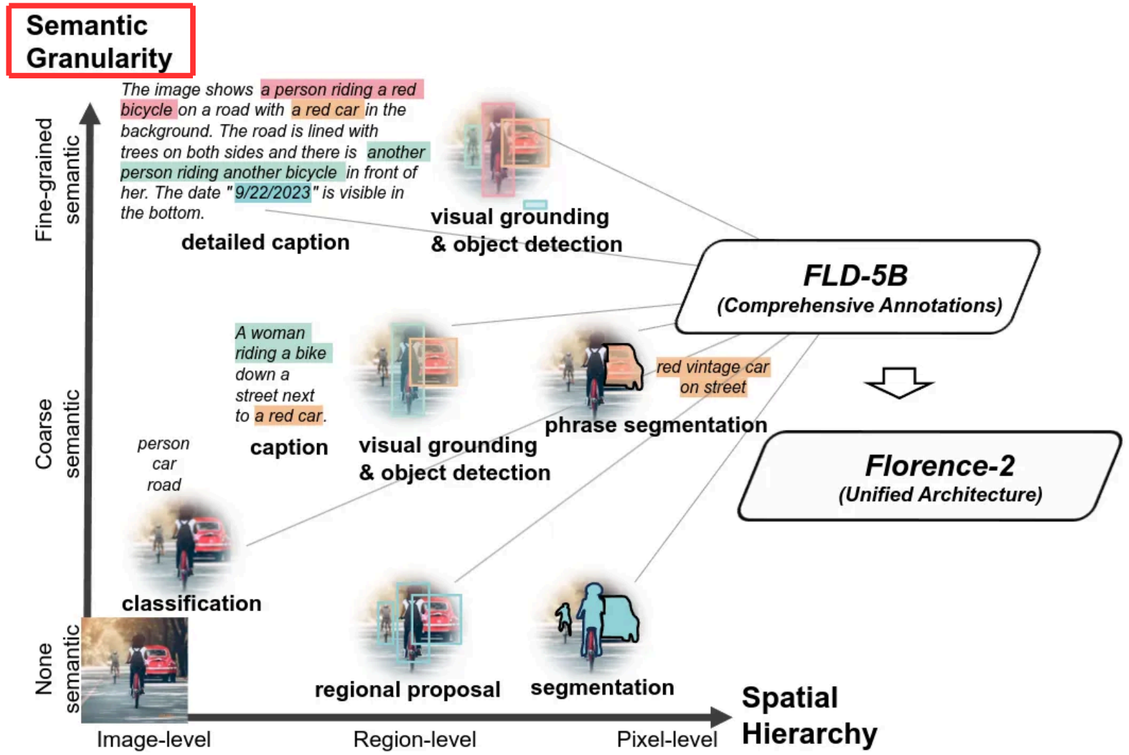
Region-level

이미지 내의 개체 및 엔티티 위치를 자세히 로컬화하여 **객체 간의 관계**와 **그들의 공간적 맥락**을 이해 가능
ex) Region proposal, Visual grounding, Object Detection, …

Pixel-level

시각적 개체의 **로컬 세부 사항**과 그들의 **의미적 맥락**, 그리고 **텍스트와 시각적 요소 간의 상호작용**을 이해 가능
ex) Segmentation, Phrase segmentation, …

Florence-2's Capacity



Semantic Granularity

의미적 세밀함

의미적 세밀함의 정도에 따라 3단계로 나눌 수 있다.

Non-semantic

의미적 정보를 포함하고 있지 **않는다**.
ex) Regional proposal, Segmentation, …

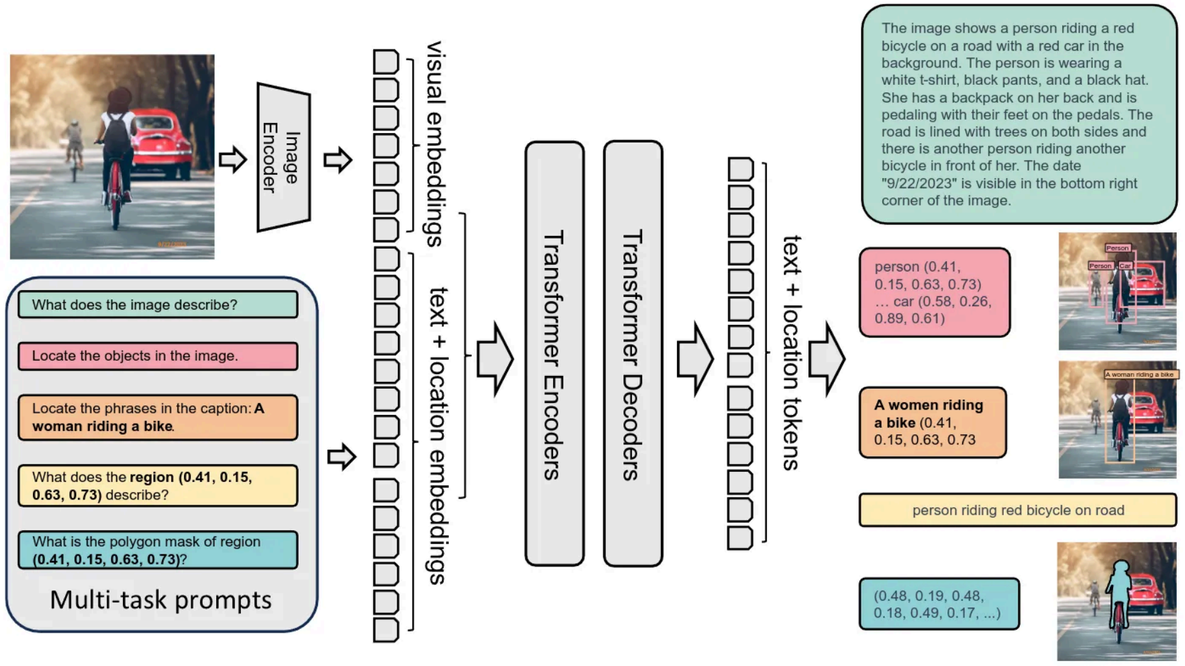
Coarse semantic

의미적 정보를 **어느정도 포함**하고 있다.
ex) Captioning, Visual grounding, Object detection, Phrase segmentation, …

Fine-grained semantic

세밀한 의미 정보를 포함하고 있다.
ex) Detailed Caption, Visual grounding, …

Florence-2 Architecture



Model Architecture

모델 구조

Learning Paradigm

다양한 비전 작업을 통합된 방식으로 다루기 위해 **Seq2Seq** 프레임워크를 채택
-> task들을 번역 문제로 공식화 했기 때문

Vision Encoder

이미지들을 시각적 토큰 임베딩으로 변환하기 위해 Vision Encoder 사용

Multi-modality Encoder Decoder

시각적 토큰 임베딩이 텍스트 임베딩과 **결합**된 후,
transformer 기반의 multi-modality encoder-decoder를 거쳐 응답을 생성

Seq2Seq?

RNN or LSTM or GRU

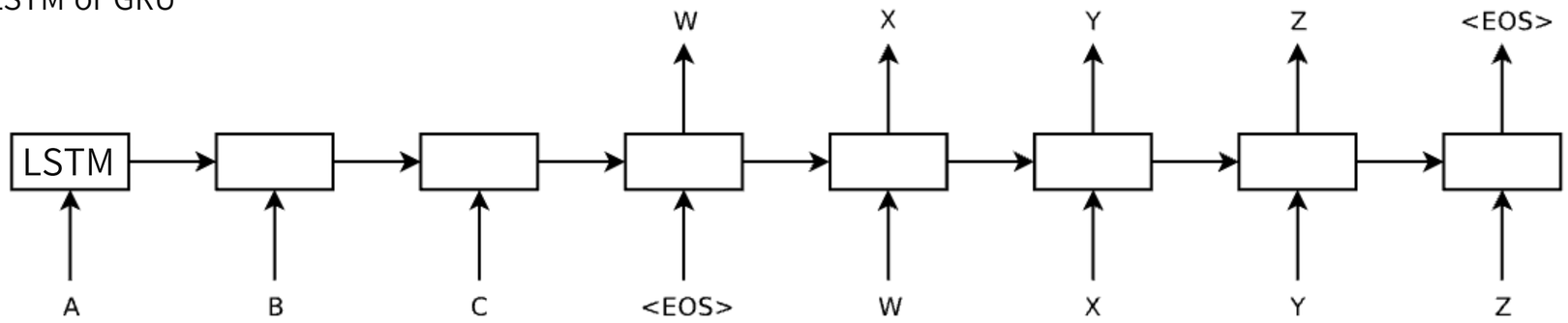
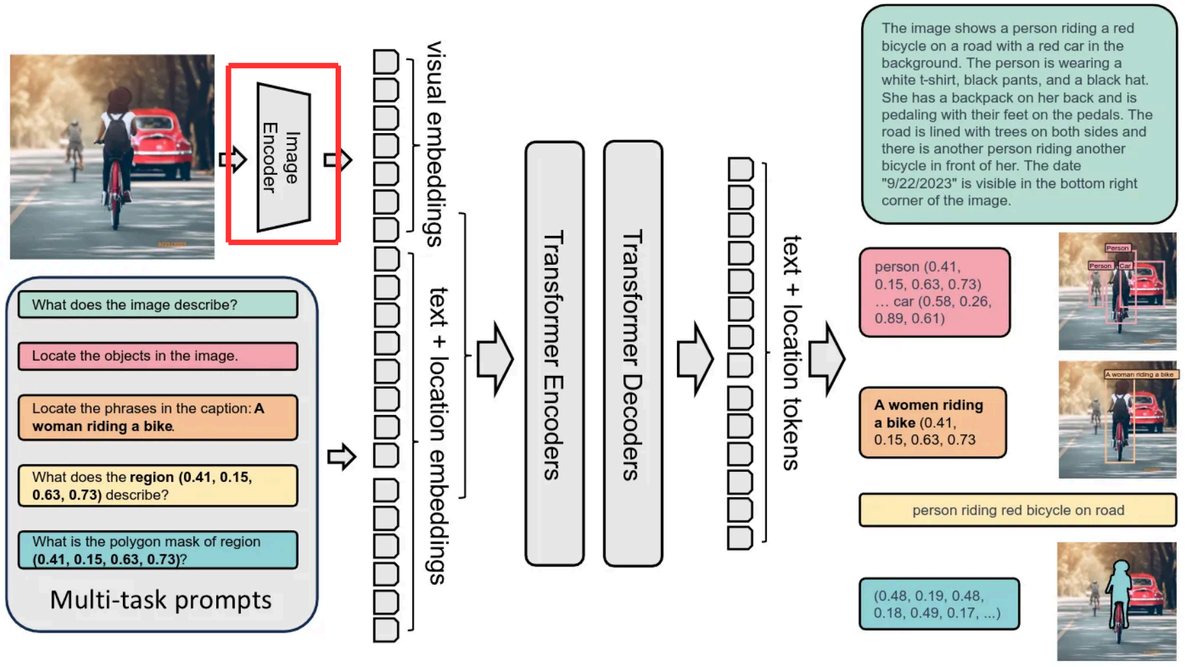


Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

Sequence-to-sequence (Seq2Seq) 학습 방식은 주로 **입력 시퀀스**를 주어진 **출력 시퀀스**로 **변환**하는 작업.
두 개의 데이터 사이에 어떤 **연관 관계**나, 어떤 입력이 들어가면 어떤 출력이 나와야 한다는 **원인과 결과 관계**를 구현하는 학습 방식을 **모두** seq2seq라고 부를 수 있다.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14). MIT Press, Cambridge, MA, USA, 3104–3112.

Florence-2 Architecture



Vision Encoder

Specific model

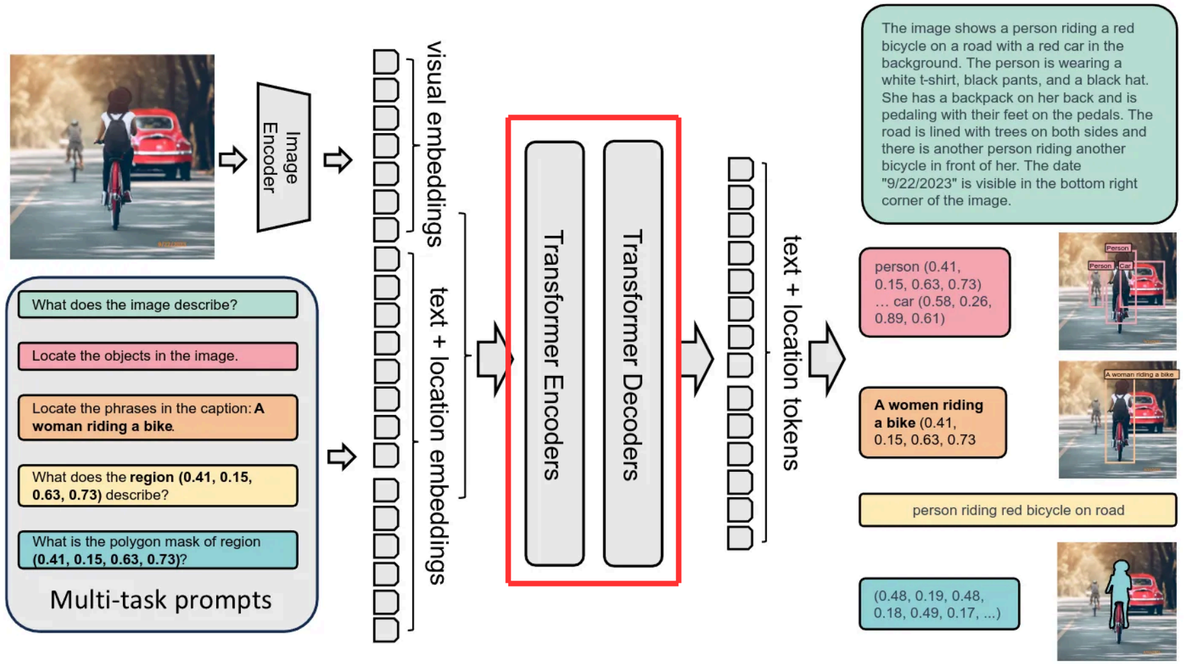
DaViT vision encoder

Image embedding

$$I \in \mathbb{R}^{H \times W \times 3} \rightarrow V \in \mathbb{R}^{N_v \times D_v}$$

Image **I**를 flattened visual token embedding **V**로 변환한다.
H, **W**는 높이와 너비, **N_v** 와 **D_v** 는 vision token의 수와 차원을 나타낸다.

Florence-2 Architecture



Multi-modality Encoder Decoder

Specific model

Standard multi-modality encoder-decoder

Text embedding

$$T_{prompt} \in R^{N_t \times D}$$

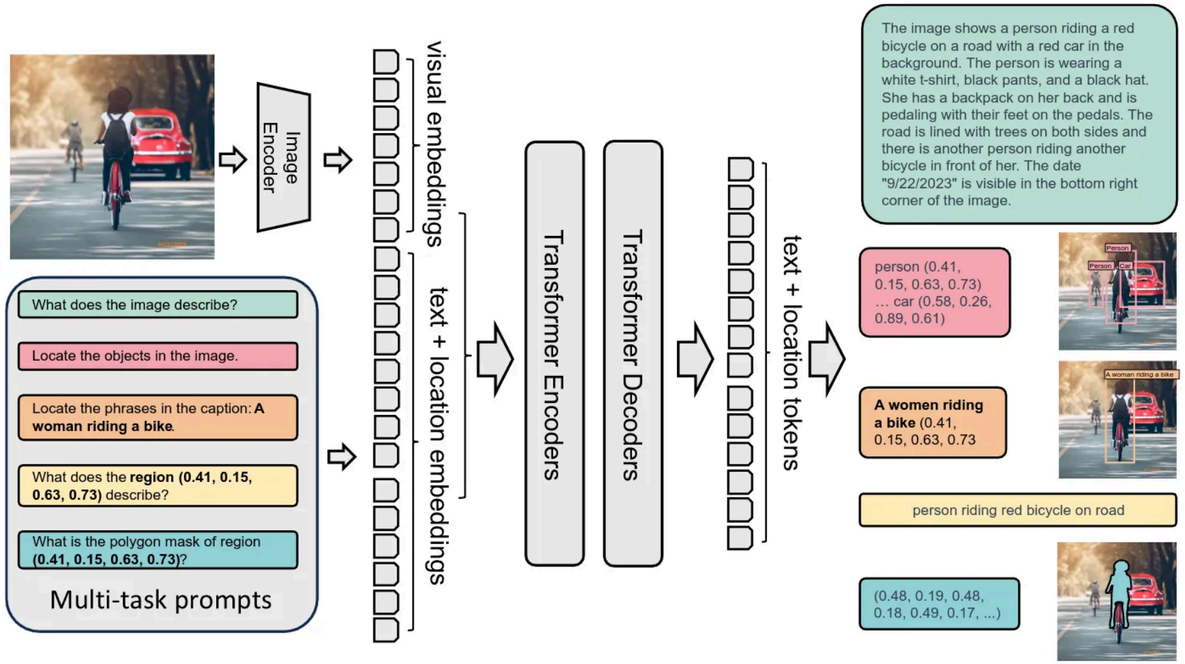
Language tokenizer와 word embedding layer를 통해 **prompt의 text embedding** 도출

Multi-modality Encoder Decoder

$$X = [V', T_{prompt}],\ where\ V' \in \mathbb{R}^{N_v \times D}$$

Visual token **V**와 concat하고 이를 multi-modality encoder module에 입력
V 의 차원 수를 맞추기 위해 **linear projection**과 **LayerNorm** layer 추가

Florence-2 Architecture



Optimization objective (Loss function)

Cross-entropy Loss

Image와 text를 combine한 input **x**, target **y**
Standard language modeling cross-entropy loss 사용

$$\mathcal{L} = -\sum_{i=1}^{|y|} \log P_\theta(y_i | y_{<i}, x)$$

θ는 network parameter고, **|y|** 는 target token의 갯수

Task Formulation > Text

Downstram task들을 **번역 문제**로 공식화
입력 이미지와 작업 특정 프롬프트가 주어지면, 해당하는 출력 응답을 생성
작업에 따라 프롬프트와 응답은 Text나 Text로 표현된 Region일 수 있다.

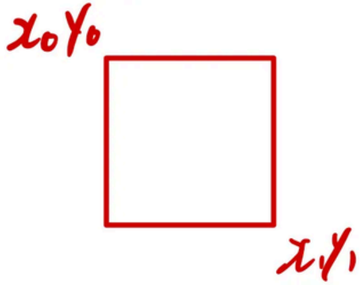
Text

프롬프트 또는 응답이 특별한 형식 없이 **일반 텍스트**일 경우, 최종 Seq2Seq 형식에서 이를 유지한다.

*특별한 형식: <OD>, <CAPTION> 등의 task를 지정하는 텍스트, region을 나타내는 text

Task Formulation > Region

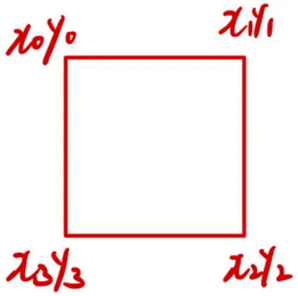
Box representation



Object Detection, Dense Region Captioning 등이 될 수 있으며, **Top-left, bottom-right 순서**이다.

```
# input format:
{"PROMPT": "<REFERRING_EXPRESSION_SEGMENTATION> <loc_702><loc_575><loc_866><loc_772>"

# output format:
{"<TASK_NAME>": {'bboxes':[[x0, y0, x1, y1], [...] ...]}
```

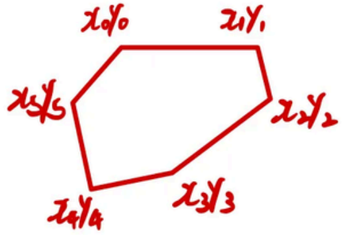
Quad box representation



Text Detection, Recognition tasks 등이 될 수 있으며, **Top-left에서 시작해서 시계방향 순서**이다.

```
# output format:
{"<TASK_NAME>": {'quad_boxes':[[x0, y0, x1, y1, x2, y2, x3, y3], [...] ...]}
```

Polygon representation



Segmentation task들이 될 수 있으며, **시계방향 순서**이다.

```
# output format:
{"<TASK_NAME>": {'polygons':[[x0, y0, x1, y1, ... xn, yn]}
```

Downstream Tasks

Single Tasks

Task	Annotation Type	Prompt Input	Output
Caption	Text	Image, text	Text
Detailed caption	Text	Image, text	Text
More detailed caption	Text	Image, text	Text
Region proposal	Region	Image, text	Region
Object detection	Region-Text	Image, text	Text, region
Dense region caption	Region-Text	Image, text	Text, region
Phrase grounding	Text-Phrase-Region	Image, text	Text, region
Referring expression comprehension	Region-Text	Image, text	Text, region
Open vocabulary detection	Region-Text	Image, text	Text, region
Referring segmentation	Region-Text	Image, text	Text, region
Region to text	Region-Text	Image, text, region	Text
Text detection and recognition	Region-Text	Image, text	Text, region

Cascaded Tasks

Caption + Phrase Grounding

Detailed Caption + Phrase Grounding

More Detailed Caption + Phrase Grounding

Downstream Tasks > Single Task > Captioning

Caption

```
run_example(task_prompt='<CAPTION'>, pil_image=pil_image)
# output:
{'<CAPTION>':
    'A green car parked in front of a yellow building.'}
```

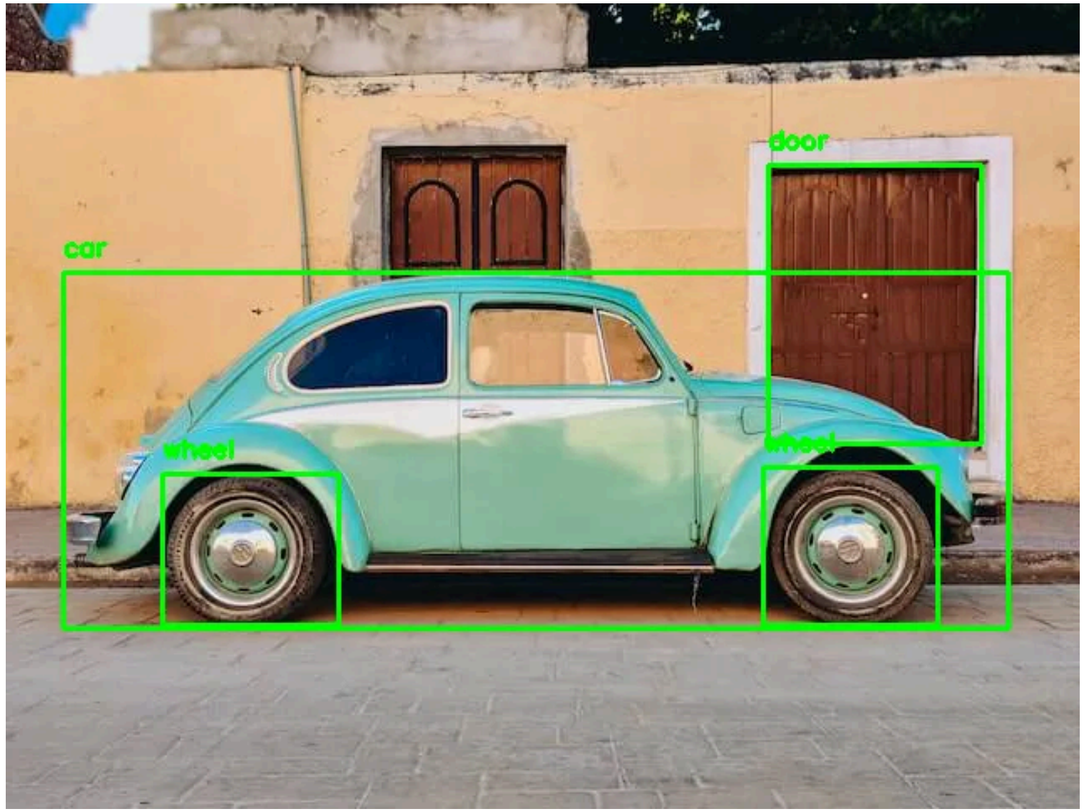
Detailed Caption

```
run_example(task_prompt='<DETAILED CAPTION'>, pil_image=pil_image)
# output:
{'<DETAILED_CAPTION>':
    'The image shows a blue Volkswagen Beetle parked in front of a yellow building with two brown doors, surrounded by trees and a clear blue sky.'}
```

More Detailed Caption

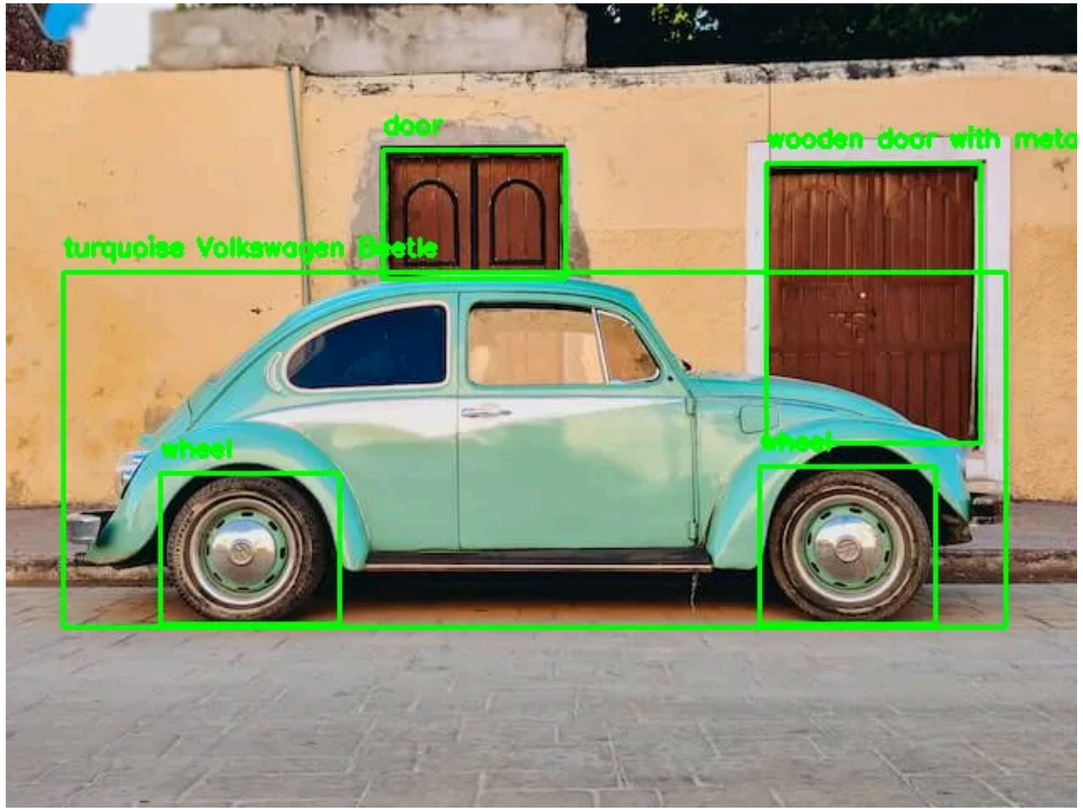
```
run_example(task_prompt='<CAPTION'>, pil_image=pil_image)
# output:
{'<MORE_DETAILED_CAPTION>':
    'The image shows a vintage Volkswagen Beetle car parked on a cobblestone street in front of a yellow building with two wooden doors. The car is painted in a bright turquoise color and has a sleek, streamlined design. It has two doors on either side of the car, one on top of the other, and a small window on the front. The building appears to be old and dilapidated, with peeling paint and crumbling walls. The sky is blue and there are trees in the background.'}
```

Downstream Tasks > Single Task > Detection > Object Detection



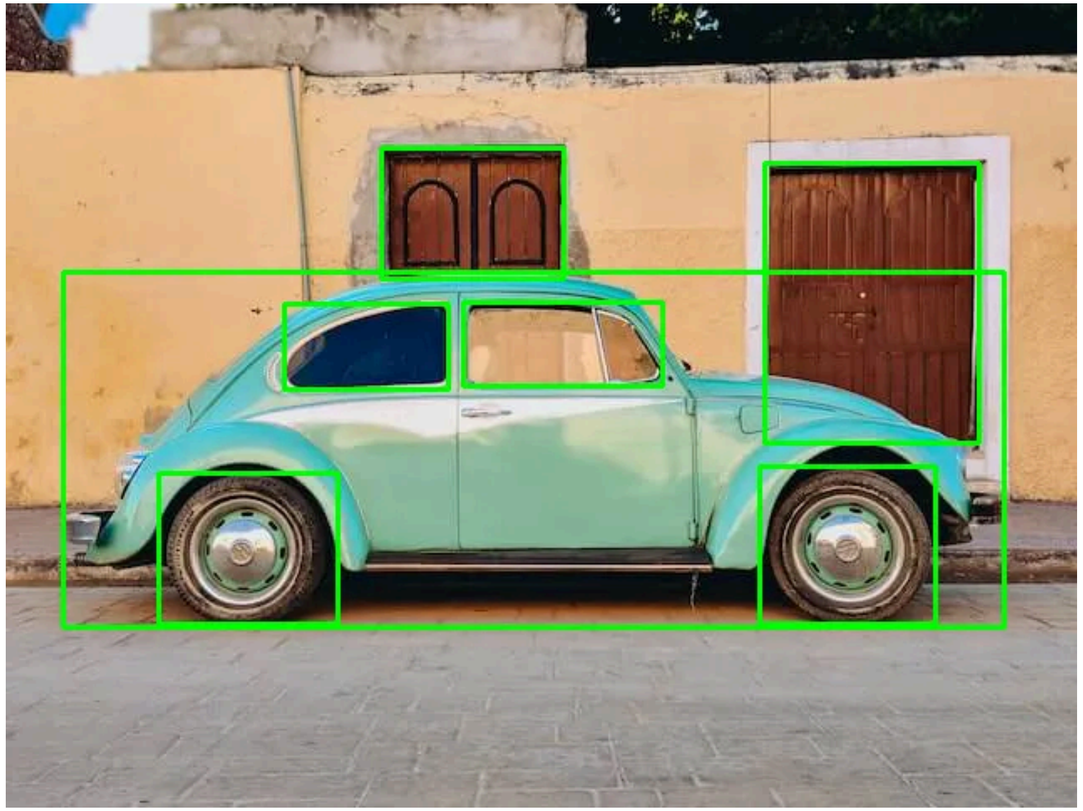

```python
result = run_example(task_prompt="<OD>", pil_image=pil_image)
# result:
{'<OD>': {
    'bboxes': [
        [34.880001068115234, 160.0800018310547, 597.4400024414062, 372.239990234375],
        [454.7200012207031, 96.23999786376953, 581.4400024414062, 262.79998779296875],
        [451.5199890136719, 276.7200012207031, 555.2000122070312, 370.79998779296875],
        [93.1199951171875, 280.55999755859375, 198.72000122070312, 370.79998779296875]],
    'labels': ['car', 'door', 'wheel', 'wheel']}}
image = draw_bboxes(pil_image, bboxes, labels)
```

Downstream Tasks > Single Task > Detection > Dense Region Caption



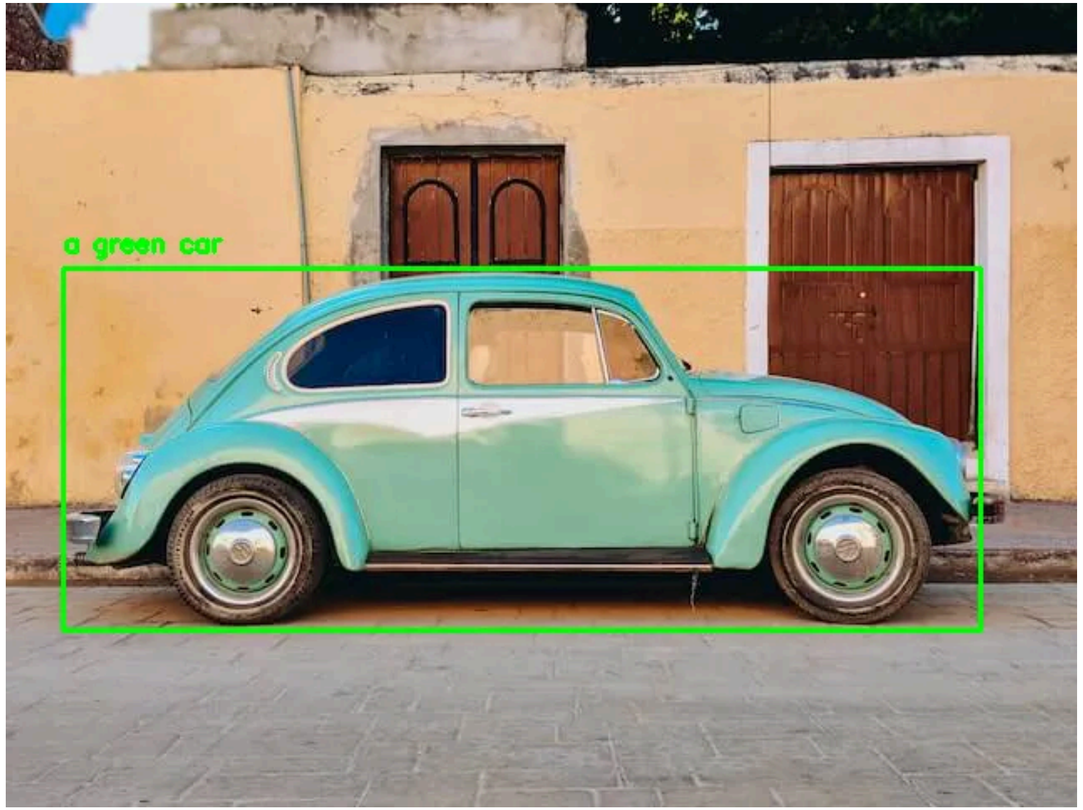

```python
result = run_example(task_prompt="<DENSE_REGION_CAPTION>", pil_image=pil_image)
# result:
{'<DENSE_REGION_CAPTION>': {
    'bboxes': [
        [34.880001068115234, 160.0800018310547, 597.4400024414062, 372.239990234375],
        [454.0799865722656, 95.75999450683594, 582.0800170898438, 262.79998779296875],
        [450.8800048828125, 276.7200012207031, 555.8399658203125, 370.79998779296875],
        [92.47999572753906, 280.55999755859375, 199.36000061035156, 370.79998779296875],
        [225.59999084472656, 87.1199951171875, 334.3999938964844, 164.39999389648438]],
    'labels': [
        'turquoise Volkswagen Beetle',
        'wooden door with metal handle and lock',
        'wheel',
        'wheel',
        'door']
}}

image = draw_bboxes(pil_image, bboxes, labels)
```

Downstream Tasks > Single Task > Detection > Region Proposal



```
result = run_example(task_prompt="<REGION_PROPOSAL>", pil_image=pil_image)
# result:
{'<REGION_PROPOSAL>': {
'bboxes': [
    [34.23999786376953, 160.0800018310547, 596.7999877929688, 372.239990234375],
    [453.44000244140625, 95.75999450683594, 581.4400024414062, 262.79998779296875],
    [450.239990234375, 276.7200012207031, 555.2000122070312, 370.79998779296875],
    [91.83999633789062, 280.55999755859375, 198.0800018310547, 370.79998779296875],
    [224.95999145507812, 86.63999938964844, 333.7599792480469, 164.87998962402344],
    [273.6000061035156, 178.8000030517578, 392.0, 228.72000122070312],
    [166.0800018310547, 179.27999877929688, 264.6399841308594, 230.63999938964844]],
'labels': ['', '', '', '', '', '', '']}}

image = draw_bboxes(pil_image, bboxes, labels)
```

Downstream Tasks > Single Task > Detection > Open Vocabulary Detection




```
result = run_example(task_prompt="<OPEN_VOCABULARY_DETECTION>", text_input="a green car",
pil_image=pil_image)
# result:
{'<OPEN_VOCABULARY_DETECTION>': {
    'bboxes': [
        [34.880001068115234, 158.63999938964844, 582.0800170898438, 374.16000366210941],
    'bboxes_labels': ['a green car'],
    'polygons': [],
    'polygons_labels': []}}

image = draw_bboxes(pil_image, bboxes, labels)
```

Downstream Tasks > Single Task > Segmentation
> Referring Expression Segmentation

```
result = run_example(task_prompt="<REFERRING_EXPRESSION_SEGMENTATION>", text_input="a green car", pil_image=pil_image)

# result:
{'<REFERRING_EXPRESSION_SEGMENTATION>': {
    'polygons': [[[180.8000030517578, 180.72000122070312, 182.72000122070312, 180.72000122070312, 187.83999633789062, 177.83999633789062, 189.75999450683594, 177.83999633789062, 192.95999145507812, 175.9199981689453, 194.87998962402344, 175.9199981689453, 198.0800018310547, 174.0, 200.63999938964844, 173.0399932861328, 203.83999633789062, 172.0800018310547, 207.0399932861328, 170.63999938964844, 209.59999084472656, 169.67999267578125, 214.0800018310547, 168.72000122070312, 217.9199981689453, 167.75999450683594, 221.75999450683594, 166.8000030517578, 226.239990234375, 165.83999633789062, ...]]],
    'labels': ['']}}

image = draw_polygons(pil_image, polygons, labels)
```

Downstream Tasks > Single Task > Segmentation > Region to Segmentation

```
result = run_example(task_prompt="<REGION_TO_SEGMENTATION>", text_input="a green car",
text_input="<loc_702><loc_575><loc_866><loc_772>", pil_image=pil_image)
# result:
{'<REGION_TO_SEGMENTATION>': {
    'polygons': [[[468.79998779296875, 288.239990234375, 472.6399841308594,
285.3599853515625, 475.8399963378906, 283.44000244140625, 477.7599792480469, 282.47998046875,
479.67999267578125, 282.47998046875, 482.8799743652344, ... , 300.239990234375,
459.8399963378906, 298.32000732421875, 460.47998046875, 296.3999938964844, 462.3999938964844,
293.5199890136719, 465.5999755859375, 289.1999816894531]]],
    'labels': ['']}}

image = draw_polygons(pil_image, polygons, labels)
```

Downstream Tasks > Single Task > Region to Text

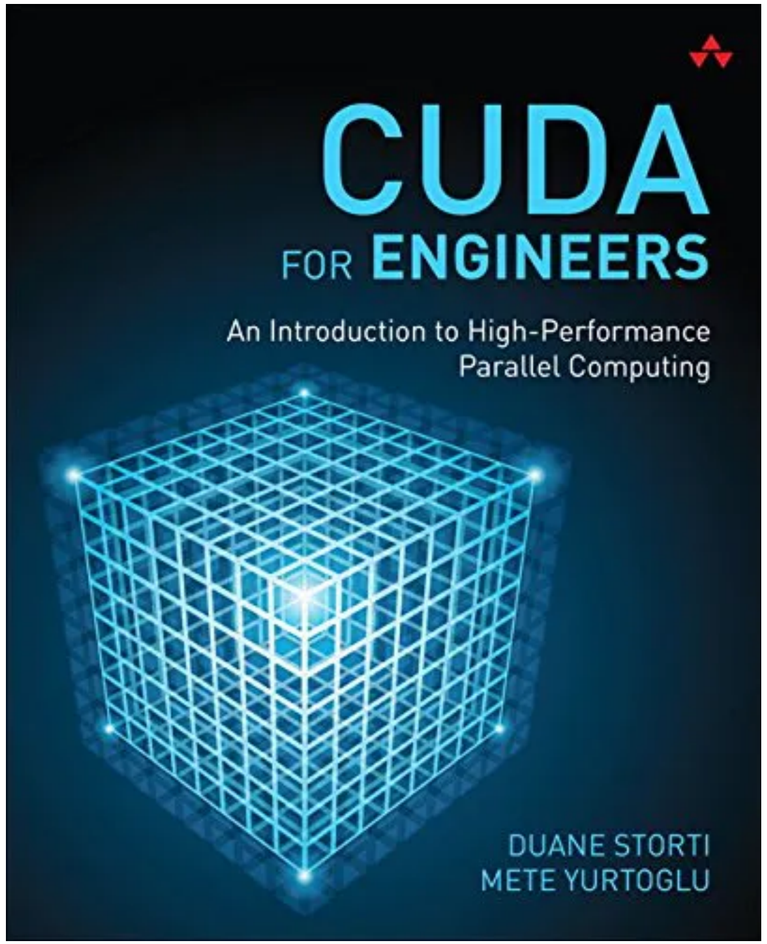
Region to Category

```
result = run_example(task_prompt="<REGION_TO_CATEGORY>", text_input="<loc_52><loc_332>
<loc_932><loc_774>", pil_image=pil_image)

# result:
{'<REGION_TO_CATEGORY>': 'car<loc_52><loc_332><loc_932><loc_774>'}
```

Region to Description

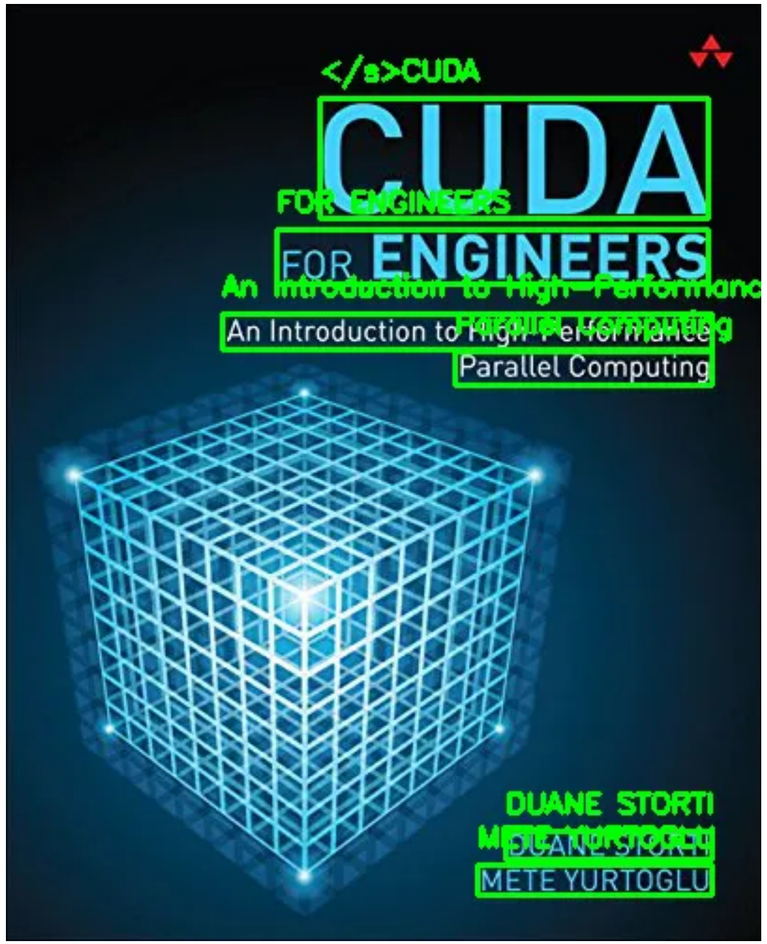
```
result = run_example(task_prompt="<REGION_TO_DESCRIPTION>", text_input="<loc_52><loc_332>
<loc_932><loc_774>", pil_image=pil_image)

# result:
{'<REGION_TO_DESCRIPTION>': 'turquoise Volkswagen Beetle<loc_52><loc_332><loc_932><loc_774>'}
```

Downstream Tasks > Single Task > OCR > OCR



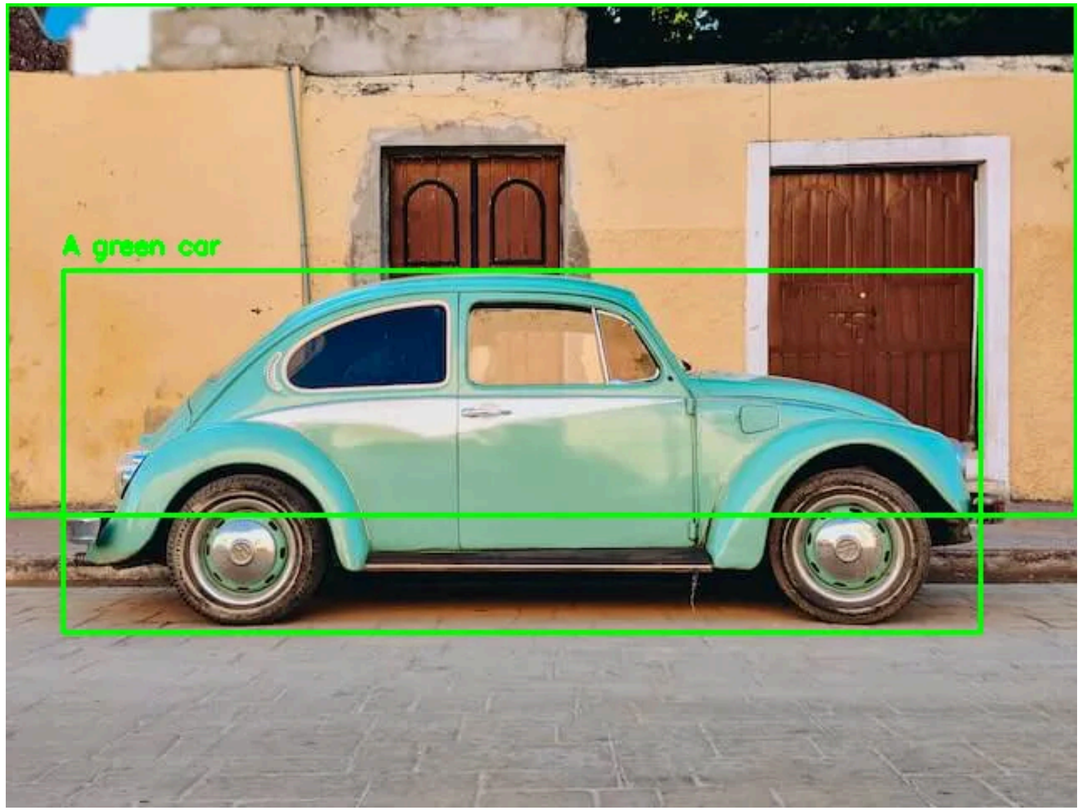

```
result = run_example(task_prompt="<OCR>", pil_image=pil_image)

# result:
{'<OCR>': 'CUDAFOR ENGINEERSAn Introduction to High-PerformanceParallel ComputingDUANE
STORTIMETE YURTOGLU'}
```

Downstream Tasks > Single Task > OCR > OCR with Region



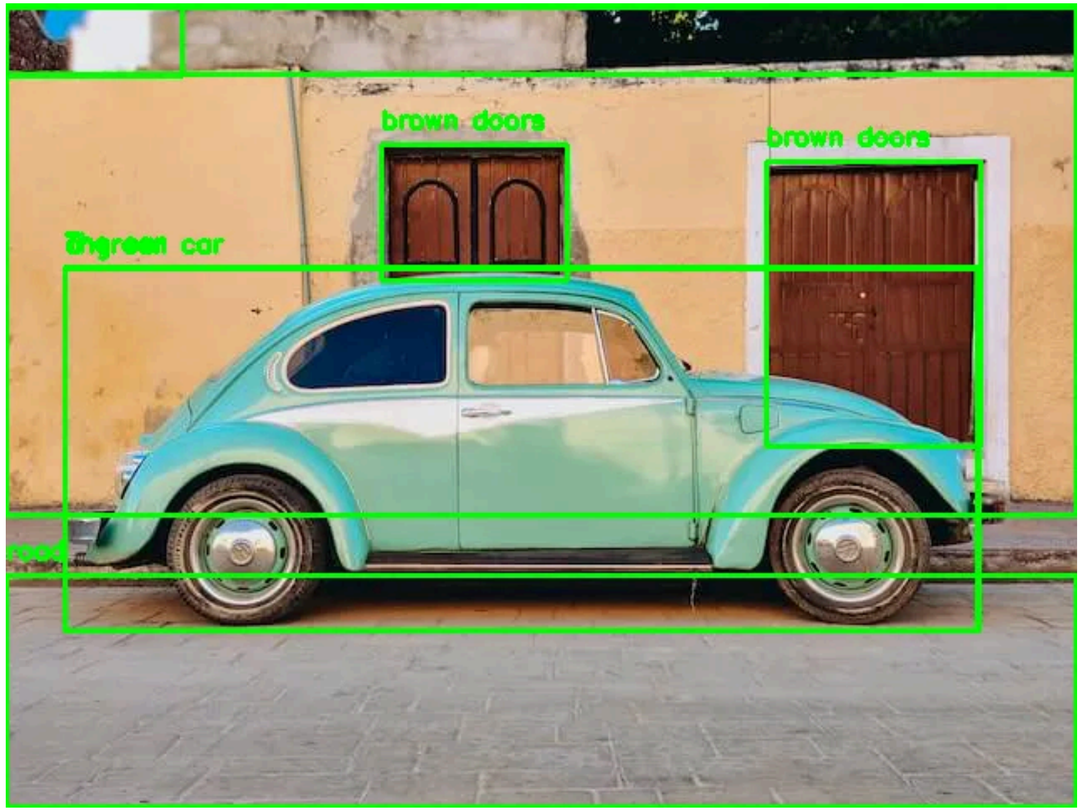

```python
result = run_example(task_prompt="<OCR>", pil_image=pil_image)

# result:
{'<OCR_WITH_REGION>': {
    'quad_boxes': [
        [167.0435028076172, 50.25, 374.9914855957031, 50.25, 374.9914855957031, 114.75, 167.0435028076172, 114.75],
        [144.8784942626953, 120.75, 374.9914855957031, 120.75, 374.9914855957031, 149.25, 144.8784942626953, 149.25],
        [115.86249542236328, 165.25, 376.20050048828125, 166.25, 376.20050048828125, 184.25, 115.86249542236328, 183.25],
        [239.9864959716797, 185.25, 376.20050048828125, 186.25, 376.20050048828125, 203.75, 239.9864959716797, 201.75],
        [266.1814880371094, 441.25, 376.20050048828125, 441.25, 376.20050048828125, 456.25, 266.1814880371094, 456.25],
        [251.67349243164062, 459.75, 376.20050048828125, 459.75, 376.20050048828125, 475.25, 251.67349243164062, 475.25]],
    'labels': [
        '</s>CUDA',
        'FOR ENGINEERS',
        'An Introduction to High-Performance',
        'Parallel Computing',
        'DUANE STORTI',
        'METE YURTOGLU']}}
```

Downstream Tasks > Cascaded Task > Caption + Phrase Grounding



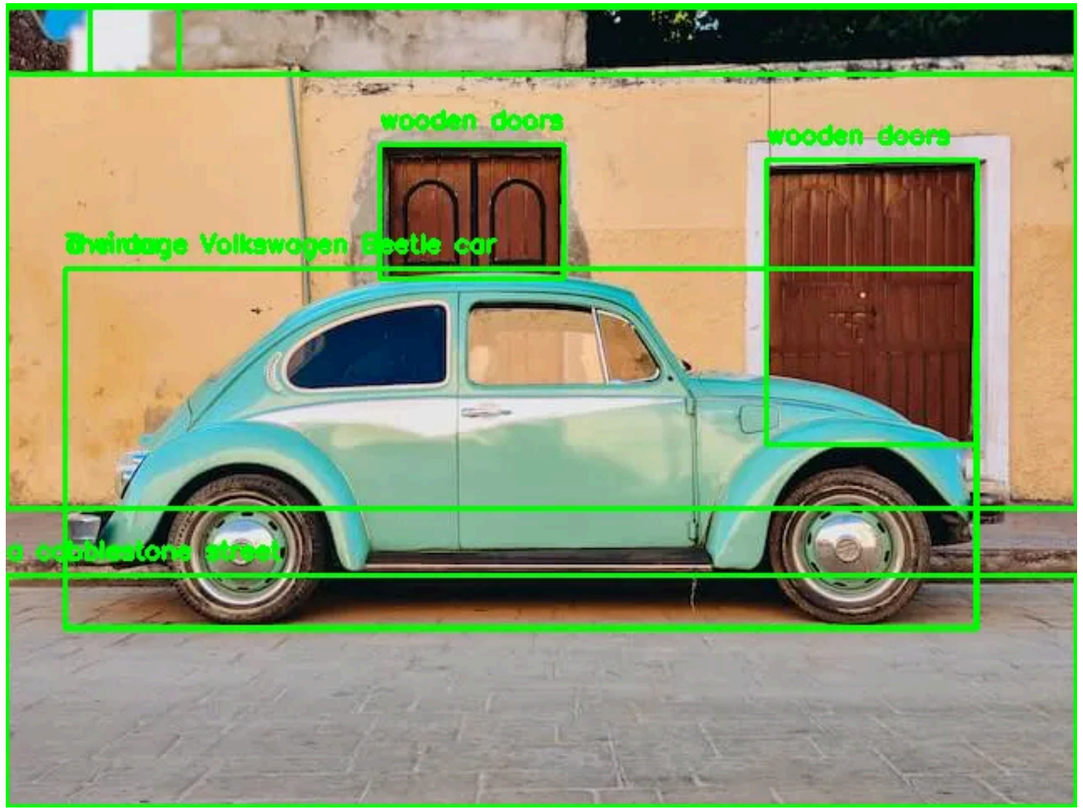

```
task_prompt_0="<CAPTION>"
result_0 = run_example(task_prompt=task_prompt_0, pil_image=pil_image)

task_prompt_1="<CAPTION_TO_PHRASE_GROUNDING>"
result_1 = run_example(task_prompt=task_prompt_1, text_input=result_0[task_prompt_0], pil_image=pil_image)

# result_0:
{'<CAPTION>': 'A green car parked in front of a yellow building.'}
# result_1:
{'<CAPTION_TO_PHRASE_GROUNDING>': {
    'bboxes': [
        [34.880001068115234, 159.1199951171875, 582.719970703125, 375.1199951171875],
        [0.3199999928474426, 0.23999999463558197, 639.0399780273438, 305.5199890136719]],
    'labels': ['A green car', 'a yellow building']}}
```

Downstream Tasks > Cascaded Task > Detailed Caption + Phrase Grounding




```
task_prompt_0="<DETAILED_CAPTION>"
result_0 = run_example(task_prompt=task_prompt_0, pil_image=pil_image)

task_prompt_1="<CAPTION_TO_PHRASE_GROUNDING>"
result_1 = run_example(task_prompt=task_prompt_1, text_input=result_0[task_prompt_0], pil_image=pil_image)

# result_0:
{'<DETAILED_CAPTION>':
    'The image shows a green car parked in front of a yellow building with two brown doors. The car is on the road and the sky is visible in the background.'}

# result_1:
{'<CAPTION_TO_PHRASE_GROUNDING>': {
    'bboxes': [
        [35.52000045776367, 158.16000366210938, 581.4400024414062, 374.1600036621094],
        [0.3199999928474426, 2.6399998664855957, 639.0399780273438, 305.5199890136719],
        [454.0799865722656, 94.31999969482422, 582.0800170898438, 264.7200012207031],
        [224.3199920654297, 84.72000122070312, 335.03997802734375, 165.36000061035156],
        [35.52000045776367, 157.67999267578125, 581.4400024414062, 374.1600036621094],
        [35.52000045776367, 158.16000366210938, 580.7999877929688, 374.1600036621094],
        [0.3199999928474426, 341.03997802734375, 639.0399780273438, 479.2799987792969],
        [0.3199999928474426, 0.23999999463558197, 639.0399780273438, 42.959999084472656],
        [0.3199999928474426, 0.23999999463558197, 105.27999877929688, 43.91999816894531]],
    'labels': [
        'a green car',
        'a yellow building',
        'brown doors',
        'brown doors',
        'The car',
        'The car',
        'road',
        'the sky',
        'the sky']}}
```

Downstream Tasks > Cascaded Task > Detailed Caption + Phrase Grounding




```python
task_prompt_0="<MORE_DETAILED_CAPTION>"
result_0 = run_example(task_prompt=task_prompt_0, pil_image=pil_image)

task_prompt_1="<CAPTION_TO_PHRASE_GROUNDING>"
result_1 = run_example(task_prompt=task_prompt_1, text_input=result_0[task_prompt_0], pil_image=pil_image)

# result_0:
{'<MORE_DETAILED_CAPTION>':
    'The image shows a vintage Volkswagen Beetle car parked on a cobblestone street in front of a yellow building with two wooden doors. The car is a light blue color with a white stripe running along the side. It has two large, round wheels with silver rims. The building appears to be old and dilapidated, with peeling paint and crumbling walls. The sky is blue and there are trees in the background.'}


# result_1:
{'<CAPTION_TO_PHRASE_GROUNDING>': {
    'bboxes': [
        [35.52000045776367, 158.16000366210938, 580.7999877929688, 373.1999816894531],
        [0.3199999928474426, 341.03997802734375, 639.0399780273438, 479.2799987792969],
        [454.0799865722656, 93.83999633789062, 580.7999877929688, 263.2799987792969],
        [223.67999267578125, 84.23999786376953, 333.7599792480469, 164.39999389648438],
        [35.52000045776367, 158.63999938964844, 579.5199584960938, 372.239990234375],
        [0.3199999928474426, 2.6399998664855957, 639.0399780273438, 301.67999267578125],
        [0.3199999928474426, 0.23999999463558197, 639.0399780273438, 42.0],
        [0.3199999928474426, 0.23999999463558197, 103.36000061035156, 42.47999954223633],
        [0.3199999928474426, 0.23999999463558197, 50.23999786376953, 42.47999954223633]],
    'labels': [
        'a vintage Volkswagen Beetle car',
        'a cobblestone street',
        'wooden doors',
        'wooden doors',
        'The car',
        'The building',
        'The sky',
        'The sky',
        'trees']}}
```

Data Engine

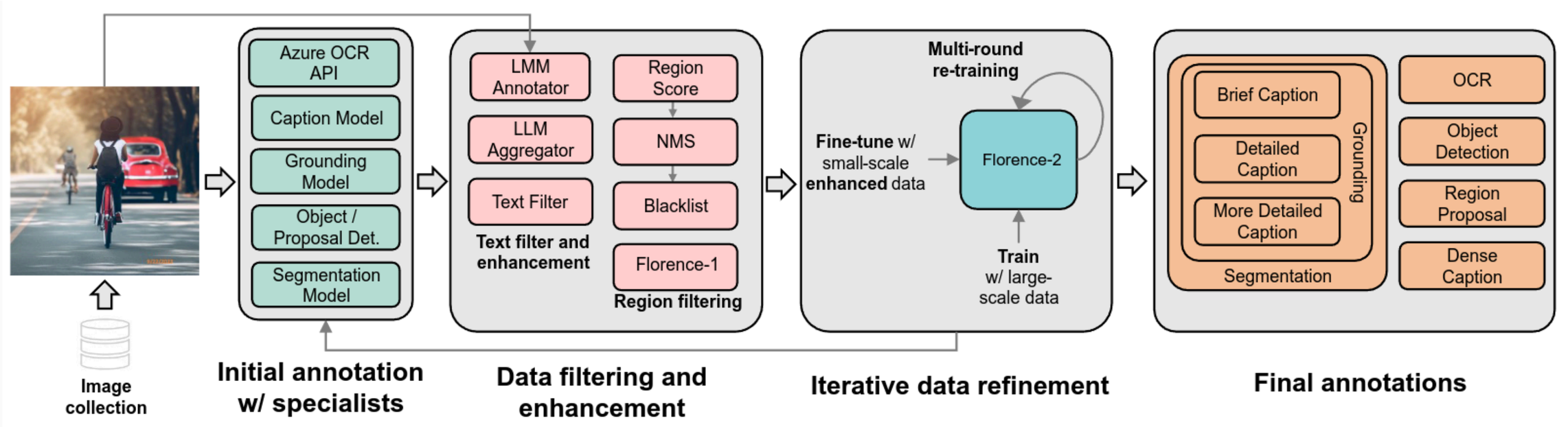
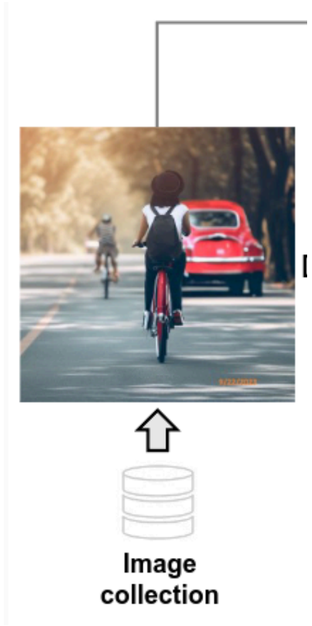


Image: 126M
Text annotation: 500M
Text-region annotaions: 1.3B
Text-phrase-region annotations: 3.6B

1B : 10억

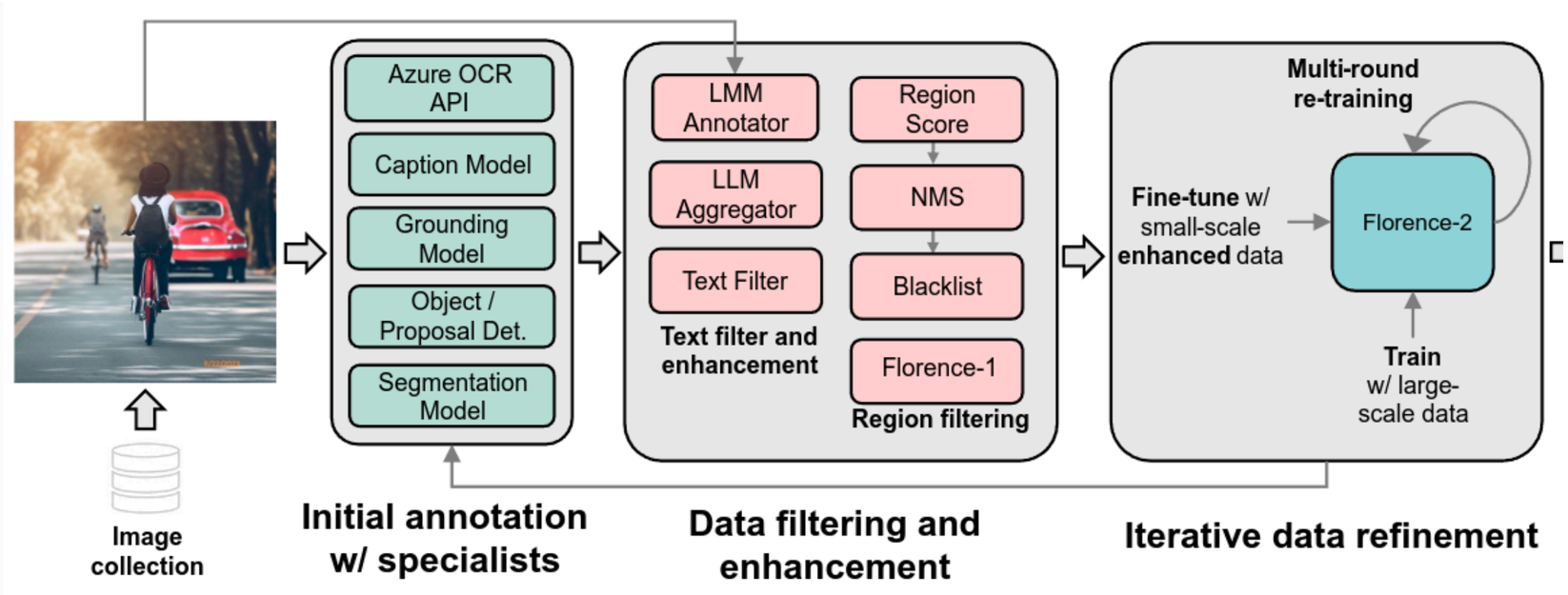
Data Engine > Image Collection



ImageNet-22K, Object 365, Open Images, Conceptual Captions, LAION -> 5개의 데이터셋 결합하여 126M images

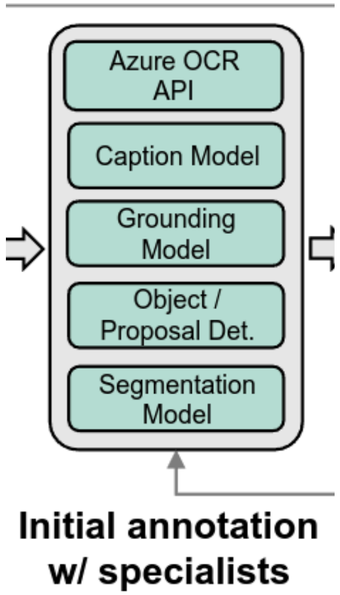
Data Engine > Data Annotation

Data Annotation 작업은 세가지 단계로 구성되어 있으며, 각 단계는 annotation의 정확성과 품질을 보장한다



1. 전문가 모델을 이용한 **초기 주석 (Initial annotation)**
2. 오류 수정 및 관계없는 주석 제거를 위한 **데이터 필터링 (Data filtering and enhancement)**
3. 데이터 개선을 위한 **반복 프로세스 (Iterative data refinement)**

Data Engine > Data Annotation > Initial annotation

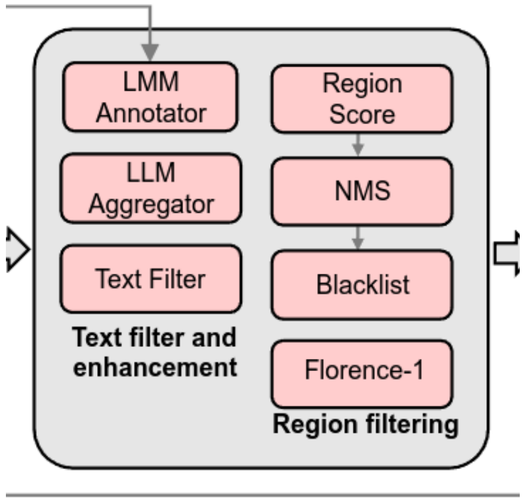


Specialist 모델에서 얻은 합성 레이블을 사용
Specialist 모델은 공개 데이터셋에서 훈련된 오프라인 모델과 클라우드 플랫폼에서 호스팅되는 온라인 서비스의 조합

세부 설명과 같은 특정 주석은 **상당히 작은 크기의 데이터셋**으로 표현된다.
-> 고성능 specialist 모델을 얻는 데 본질적으로 어려움을 초래함
-> 따라서 초기 주석 단계에서 이러한 작업을 건너뛰기로 선택한다.
이러한 작업에 대한 주석은 Iterative data refinement 중에 생성됩니다.

엄격한 초기 주석 절차를 통해 126M 개의 이미지가 대다수의 주석 유형에 대해 포괄적으로 레이블이 지정된 데이터셋 보장

Data Engine > Data Annotation > Data filtering and enhancement



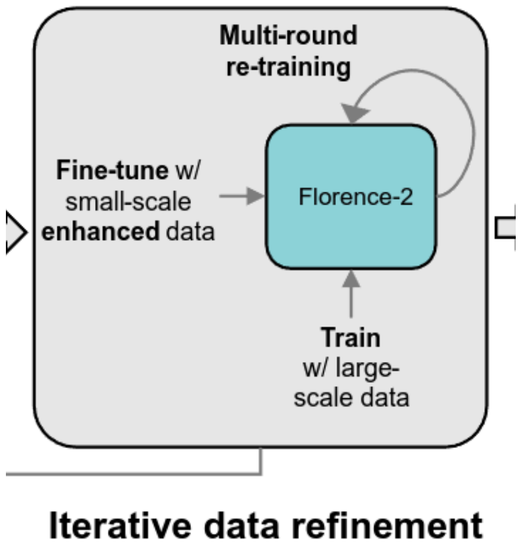
Data filtering and enhancement

초기 주석은 포괄적이지만, **잡음과 부정확성에 취약**
→ 원치 않는 주석을 정제하고 제거하기 위한 **다면적** 필터링 프로세스를 구현
필터링 프로토콜은 주석의 두 가지 데이터 유형, 즉 **텍스트**와 **영역** 데이터를 주로 대상으로 한다.

- **Text annotation**
  - SpaCy를 기반으로 한 파싱 도구를 개발하여 객체, 속성 및 행동을 추출
  - **과도한 객체가 포함된 텍스트**를 필터링
  - 이러한 텍스트는 **잡음을 초래**할 수 있으며, 해당 이미지의 **실제 내용을 정확하게 반영하지 못할 수 있다**
  - 행동 및 객체의 복잡성을 분석하여 의존성 파싱 트리에서의 노드 수를 측정함으로써 이를 평가
  - 이미지에서 **시각적 개념의 풍부함을 보장**하기 위해 **특정 최소 행동 및 객체 복잡성을 가진 텍스트**를 유지

- **Region annotation**
  - 경계 상자와 관련하여 **신뢰 점수 임계값 아래의 잡음 상자를 제거**한다.
  - 이를 보완하기 위해, 중복되거나 겹치는 경계 상자를 줄이기 위해 **비최대 억제**를 적용

Data Engine > Data Annotation > Iterative data refinement



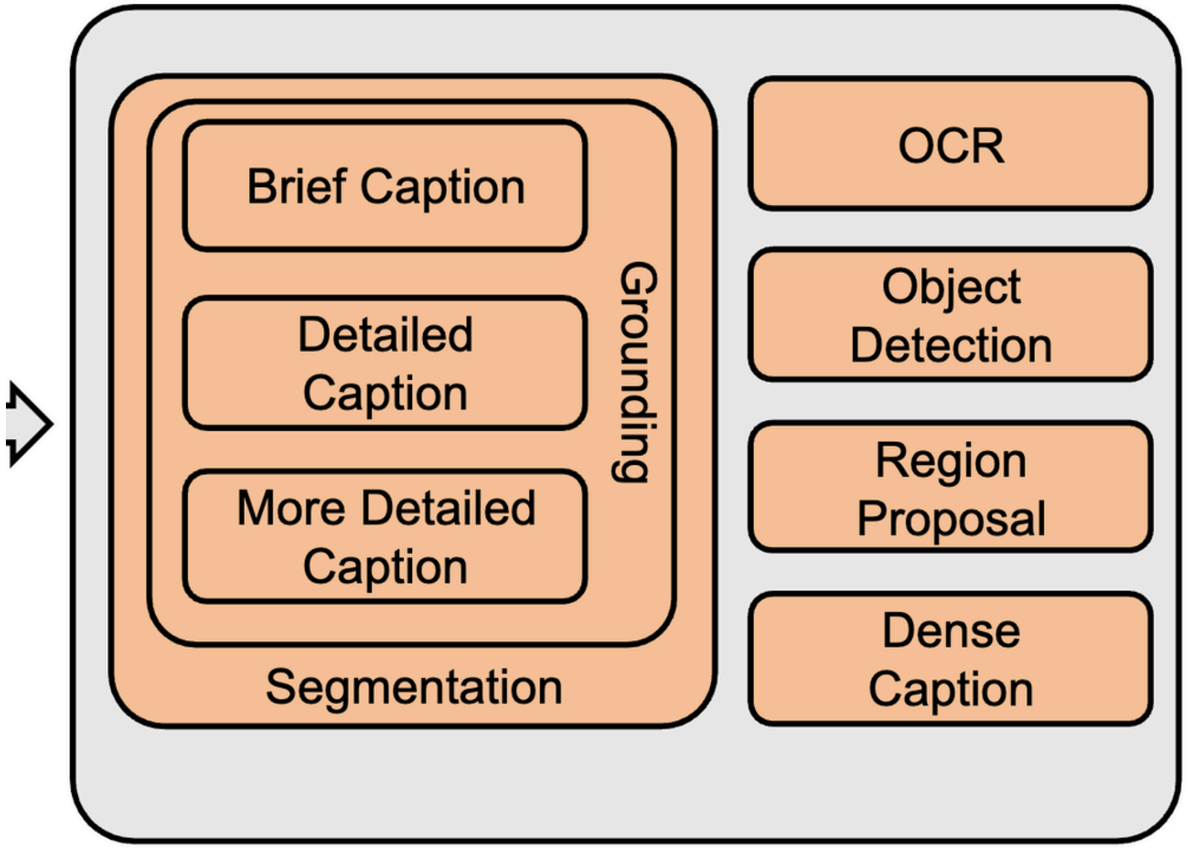
Iterative data refinement

이전 2단계를 통해 **업데이트된 주석을 원래 주석과 통합**하고 모델을 **다시** 훈련시키는 과정
이는 점진적으로 훈련 데이터셋의 품질을 향상시킨다.

처음의 specialist 모델 교육을 위한 데이터가 부족하여 우회한 작업의 경우,
-> 반복적으로 훈련된 모델을 pre-train 목적으로 활용
이 사전 훈련된 모델을 희소 데이터셋으로 후속 fine tuning한 결과, 동일한 데이터에서 처음부터 훈련된 모델보다 우수한 성능을 보였다.

fine tuning된 모델을 126M 이미지를 포함하는 방대한 데이터셋 주석을 위한 전문가로 활용하여 종합적인 주석을 완성했다.

Data Engine > Data Annotation > Iterative data refinement



Final annotations

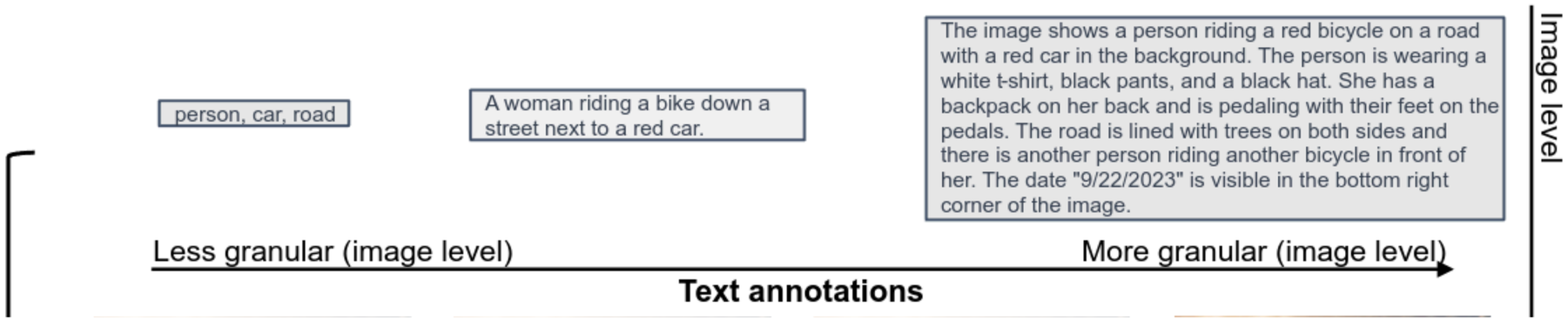
Data Engine > Annotation-specific Variations

Three discrete annotation categories :

- **Text**
- **Region-text pairs**
- **Text-phrase-region triplets**

Data Engine > Annotation-specific Variations

Text



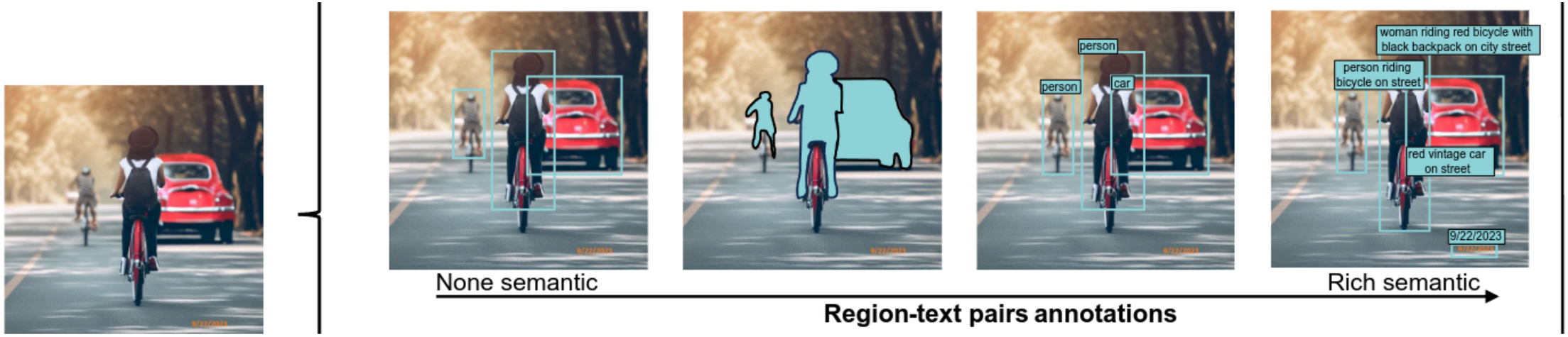
3가지 종류의 세분성: **Brief**, **Detailed**, **More Detailed**

Brief : 공개적으로 사용 가능한 이미지 캡션 및 이미지-텍스트 데이터셋에서 전문 모델로 훈련된 Florence-2 모델을 사용하여 초기 주석을 위한 이미지-텍스트 모델을 생성. 반복적인 개선을 통해 이러한 텍스트의 노이즈를 최소화

Detailed & More Detailed : Brief text 및 region-text 주석과 같은 기존 이미지 주석을 포함하는 프롬프트가 **대형 언어 모델(LLM)** 또는 **대형 다중 모달 모델(LMM)**에 **일부** 제공되어 포괄적인 설명을 생성
-> 캡션 specialist 모델을 fine-tuning하여 추가 주석을 위한 자세한 설명 specialist 모델을 개발하는 데 사용된다.

Data Engine > Annotation-specific Variations

Region-text pairs



이미지의 semantic region에 대해 설명하는 text 주석 (Semantic region에는 **visual 객체**와 **text 영역** 모두 포함)
Visual 객체와 text 영역은 서로 다르게 주석이 달린다.

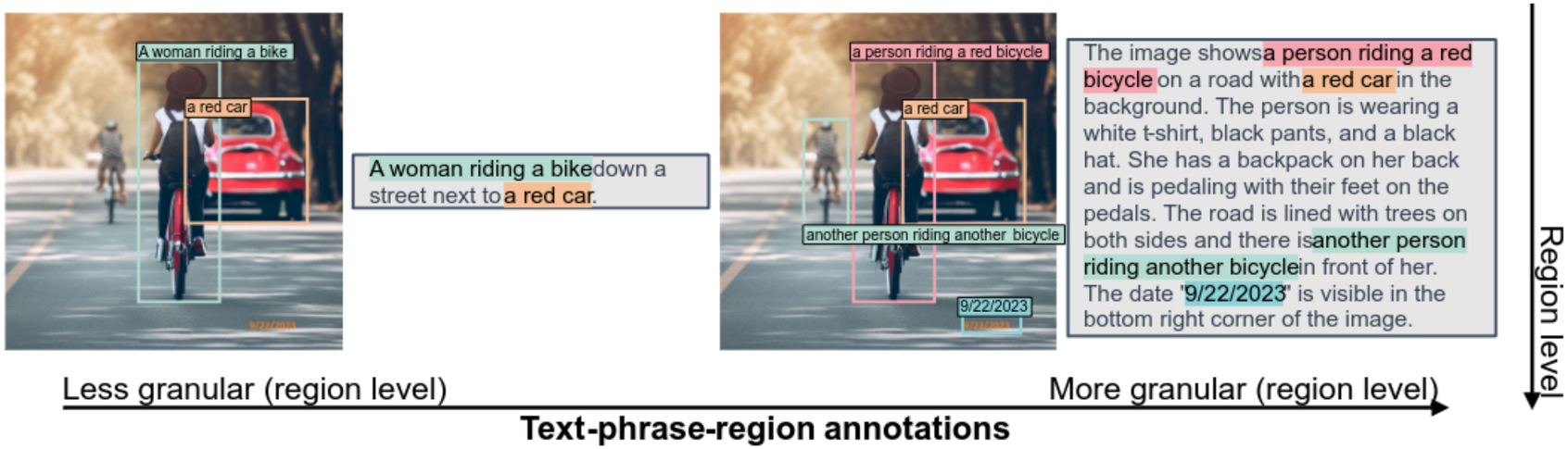
Visual : 공공 데이터셋에서 훈련된 DINO object detector

- 데이터 필터링, 즉 신뢰도 임계값 설정 및 비최대 억제를 통해 노이즈 박스가 제거
- 시각적 객체 영역에 대한 텍스트 주석은 crop된 이미지 지역에서 생성된 간단한 텍스트로 더욱 풍부해진다.
- 각 지역에 세 가지 텍스트 주석 부여: (1)**객체 범주의 구절**, (2)**간단한 텍스트**, (3)**간단한 텍스트의 명사 구절 청크**
  ((1)**phrase from object category**, (2)**brief text** , (3)**noun phrase chunks from the brief text**)

Text : Azure AI 서비스의 OCR API를 사용하여 라벨링

Data Engine > Annotation-specific Variations

Text-phrase-region triplets



(1)**이미지의 설명적인 텍스트** , (2)이 **텍스트**와 관련된 **이미지 객체의 명사 구절** , (3)이러한 **객체의 지역 주석**으로 구성
((1)**Descriptive text of the image**, (2)**noun phrases** in this text related to image objects, (3)**region annotations** for these objects)

텍스트는 이전에 생성된 **Brief**, **Detailed**, 및 **More Detailed** 텍스트를 포함

- 각 텍스트에 대해 Grounding DINO 모델이 명사 구절을 식별하고 이를 위한 **경계 상자를 생성**
- SAM 모델은 각 상자에 대한 세분화 마스크를 생성하여 **더 정확한 객체 위치 지정**을 제공
- 명사 구절 및 경계 상자에 신뢰도 점수 임계값이 적용되어 **관련성을 보장**
- 대명사 및 추상 개념과 같은 관련 없는 명사 구절을 제외하기 위해 **블랙리스트 사용**

Data Analysis > Annotation statistics

Image: 126M
Text annotation: 500M
Text-region annotaions: 1.3B
Text-phrase-region annotations: 3.6B

Dataset	Rep. Model	#Images	#Annotations	Spatial hierarchy	Semantics granularity
JFT300M [21]	ViT	300M	300M	Image-level	Coarse
WIT [64]	CLIP	400M	400M	Image-level	Coarse
SA-1B [32]	SAM	11M	1B	Region-level	Non-semantic
GrIT [60]	Kosmos-2	91M	137M	Image & Region-level	Fine-grained
M3W [2]	Flamingo	185M	43.3M*	Multi-image-level	Fine-grained
FLD-5B (ours)	*Florence-2* (ours)	126M	5B	Image & Region-level	Coarse to fine-grained

▲ 다른 모델들과 비교

500M text annotation

- COCO와 유사한 token수를 가진 Brief에 비해 Detailed와 More Detailed는 각각 **4배**, **9배**의 token 수를 가지고 있다.

1.3B text-region annotation

- OpenImages및 Object 365와 같은 객체 탐지 데이터 셋보다 **30배** 이상 많다.

3.6B text-phrase-region annotation

- Brief 주석은 평균 4.27개의 phrase-region 쌍을 가지며, Detailed 주석은 10개 이상의 쌍을 가진다.
- 이는 **Detailed 주석이 더 많은 객체와 그에 해당하는 구들을 포괄함**을 나타낸다.

Annotation Type	Text Type	#Image Annotations	#Avg Tokens	#Regions	#Avg Regions	#Avg Regional Tokens
Text	Brief	235M	7.95	-	-	-
	Detailed	**500M** 126M	31.65	-	-	-
	More detailed	126M	70.53	-	-	-
Region-Text	Phrase	126M	-	681M	**1.3B** 5.42	1.19
	Brief	126M	-	681M	5.42	2.55
Text-Phrase-Region	Brief	235M	7.95	1007M	4.27	1.93
	Detailed	126M	31.65	1289M	**3.6B** 10.25	1.49
	More detailed	126M	70.53	1278M	10.17	1.35

▲ Annotation 통계

1007 / 235 = 4.28
1289 / 126 = 10.23
1278 / 126 = 10.14

Data Analysis > Semantic coverage

의미적 범위를 다루기 위해 SpaCy를 활용하여 토큰화 및 구문 분석을 진행했으며,
이 과정은 품사(part-of-speech, POS) 태그 및 토큰 간 의존 파싱 트리를 생성한다.

- **의미적 요소 유형**으로 분류: 토큰을 **객체**, **속성**, **행동** 및 **고유 명사**로 분류
- **토큰 복잡성** 개념: 의존 파싱 트리가 비방향 그래프 형태로 처리되었을 때 토큰의 총 차수. 이 복잡성은 의미적 연결의 풍부함을 반영한다.

표는 평균 **의미적 요소 수**와 해당 **복잡성**에 대한 통계를 제시한다.
텍스트 주석에 **더 많은 세부정보가 포함될수록 모든 측정값이 증가함**을 나타낸다.

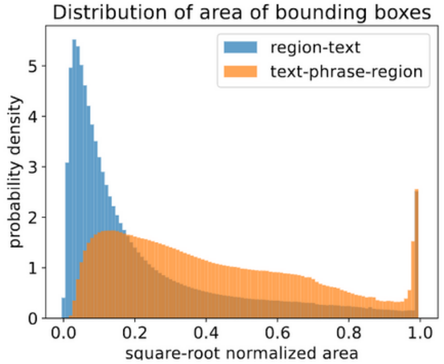
- **평균 행동(#Avg Actions)**
  - Detailed 텍스트와 More Detailed 텍스트는 Brief 텍스트에 비해 각각 **7배** 및 **15배**의 가장 큰 증가
  - 이는 이미지 행동을 설명하는 데 있어 **전통적인 간략 텍스트 주석의 한계**가 있음을 확인할 수 있음 -> (COCO annotation의 한계)

- **고유 명사(#Proper Nouns)**
  - 증가폭은 상대적으로 낮은 이유는 specialist model들이 **특정 고유 명사**를 사용하는 것보다 더 **일반적**으로 객체를 설명할 가능성이 높기 때문

- **복잡성 측정(Avg Complexity)**
  - **Object**와 **Action** 모두 **More detailed** 텍스트 주석에서 더 많은 의미적 연결을 보임
  - 특히 **Action**의 복잡성은 향상 수준이 높아지며, 이는 행동 수 증가에 대한 우리의 관찰과 일치합니다.

Text Type	Brief	Detailed	More detailed
#Image Annotations	235M	126M	126M
#Avg Tokens	7.95	31.65	70.53
#Avg Objects	3.23	13.31	28.06
#Avg Attributes	2.80	7.27	16.25
#Avg Actions	0.58	4.21	8.76
#Proper Nouns	1.10	2.40	2.41
Avg Object Complexity	2.80	4.00	4.02
Avg Action Complexity	1.14	3.63	4.38

Data Analysis > Spatial coverage



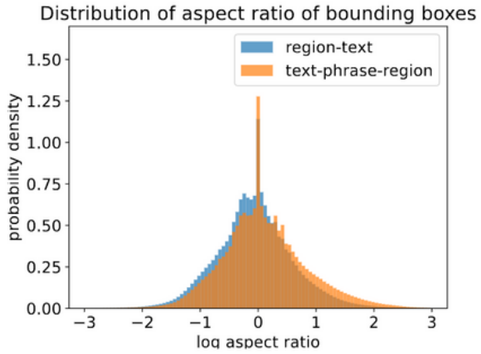
Distribution of area of bounding boxes

Bounding box 면적의 분포

Region-text pair에서 더 많은 **작은 box**가 발생하고, **text-phrase-region triplet**에서는 **균일한 box** 크기 분포를 보여준다.

이 차이는 **region-text pair**을 위한 객체 탐지기와 **text-phrase-region triplet**를 위한 기초 모델의 기원 차이에서 비롯된다.

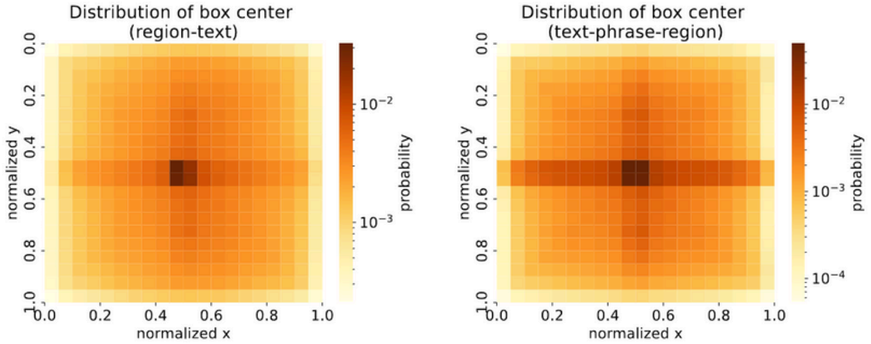
기초 모델은 지역화된 시각적 개념을 나타내는 텍스트 구문에 상자를 정렬한다.



Distribution of aspect ratio of bounding boxes

Bounding box 종횡비의 로그 형식 분포

region-text pair와 **text-phrase-region triplet**는 유사한 대칭 분포를 보여주며, 다양한 종횡비를 포괄



Distribution of box center

중심 편향: 주석 유형에 대한 상자 중심의 히트맵

region-text pair(left)가 **text-phrase-region triplet(right)**보다 더 균일한 분포를 보인다.

Experiments

Model size

Florence-2-Base : 232 million parameters (**0.22B**)
Florence-2-Large : 771 million parameters (**0.77B**)

Initialize

Image Encoder : **UniCL**
Multi-modality Encoder-Decoder : **BART**

Optimizer

AdamW with **cosine** learning rate decay
Deepspeed, **mixed precision**
Max learning rate : **1e-4(base), 1e-5(large)**
Linear warm-up to the maximum learning rate : during the **first 5,000 optimization steps**

Batch

Mini batch size: **2048/3072(base/large)**
Image size: **384×384**
30억 개의 유효 훈련 샘플에 도달할 때까지 진행
500M/100M(base/large) sample에 대해 이미지 크기 **768×768**으로 고해상도 조정을 추가로 수행

Experiments

Zero-shot Evaluation Across Tasks

여러 작업에 대한 **zero-shot 성능**을 평가하여 특정 작업 데이터에 대한 추가 fine-tuning 없이 **하나의 generalist 모델**을 사용하여 여러 작업을 처리하는 본래의 능력을 보여준다

Generalist Model with Public Supervised Data

추가적인 supervised 데이터로 **하나의 generalist 모델**을 더 훈련시켜 다양한 작업에서 경쟁력 있는 최첨단 성능을 달성함으로써 우리 방법의 **적응성**을 보여준다

Downstream Tasks Fine-tuning

Downstream 작업에서 학습된 시각적 표현의 성능을 backbone으로 검토하여 **우리의 pre-train 방법이 이전 접근 방식보다 우수함**을 보여준다.

- Object detection and segmentation
- Semantic segmentation

Ablation Studies

- Multitask transfer
- Scaling (Model & Data)
- Training settings

Experiments > Zero-shot Evaluation Across Tasks

Zero-shot Evaluation Across Tasks

Method	#params	COCO Cap.	NoCaps	TextCaps	COCO Det.	Flickr30k	Refcoco			Refcoco+			Refcocog		Refcoco RES
		test	val	val	val2017	test	val	test-A	test-B	val	test-A	test-B	val	test	val
		CIDEr	CIDEr	CIDEr	mAP	R@1	Accuracy			Accuracy			Accuracy		mIoU
Flamingo [2]	80B	84.3	-	-	-	-	-	-	-	-	-	-	-	-	-
Kosmos-2 [60]	1.6B	-	-	-	-	78.7	52.3	57.4	47.3	45.5	50.7	42.2	60.6	61.7	-
Florence-2-B	0.23B	133.0	118.7	70.1	34.7	83.6	53.9	58.4	49.7	51.5	56.4	47.9	66.3	65.1	34.6
Florence-2-L	0.77B	135.6	120.8	72.8	37.5	84.4	56.3	61.6	51.4	53.6	57.9	49.9	68.0	67.0	35.8

Table 4. **Zero-shot** performance of generalist vision foundation models. The models do not see the training data of the evaluation tasks during training. *Florence-2* models are pre-trained on *FLD-5B* dataset. Karpathy test split is used for COCO caption evaluation.

Experiments > Zero-shot Evaluation Across Tasks

What is CIDEr (Consensus-based Image Description Evaluation)?

CIDEr(Consensus-based Image Description Evaluation)은 **이미지 캡션의 품질**을 평가하기 위한 대표적인 지표
모델이 생성한 캡션이 사람에 의해 작성된 캡션과 얼마나 유사한지를 측정하는 데 사용
단순히 단어의 겹침을 비교하는 기존의 평가 지표와는 달리, CIDEr는 여러 사람이 작성한 캡션들 간의 **합의(consensus)**를 기반으로 더 세밀하게 캡션의 품질을 평가

How it works

- **n-그램 매칭**
  - n-그램(n개의 연속된 단어들)을 사용하여, 모델이 생성한 캡션과 여러 개의 사람이 작성한 참조 캡션을 비교하여 **n-그램이 얼마나 많이 겹치는지**를 확인
- **TF-IDF 가중치 부여**
  - 각 n-그램에 **TF-IDF (Term Frequency-Inverse Document Frequency) 가중치**를 부여
  - 이는 **참조 캡션에서 자주 등장**하지만, **전체 캡션 집합에서는 덜 자주 등장**하는 단어와 구문에 **더 높은 중요도**를 부여
  - ex)"a", "the", "is"와 같은 흔한 단어의 영향이 줄어듦

Advantages

- 단순한 단어 겹침을 기반으로 한 지표들(BLEU, ROUGE 등)보다 더 정교한 평가가 가능
- 언어의 다양성을 잘 반영. **드문 표현을 잘 사용하는 캡션은 더 높은 점수**를 받고, 너무 **일반적인 캡션은 페널티**를 받는다.

Experiments > Generalist Model with Public Supervised Data

Simple design for strong performance

특별한 설계 없이 **Standard** multi-modality Transformer Encoder-Decoder로 강력한 성능을 보인다.
Region-level에서 특히 더 강하다.

Competitive performance with fewer parameters

LLM을 필요로 하지 않으면서 경쟁력 있는 성능을 달성, 다양한 작업을 처리하는 데 **효율성**을 보여준다.

Adaptable generalization across task levels

Image-level, **pixel-level** 및 **region-level** 작업에서 경쟁력 있는 성능을 보여준다.

Vision 및 NLP 분야에서 **적응성**과 **효율성**을 강조

Method	#params	COCO Caption Karpathy test CIDEr	NoCaps val CIDEr	TextCaps val CIDEr	VQAv2 test-dev Acc	TextVQA test-dev Acc	VizWiz VQA test-dev Acc
			Specialist Models				
CoCa [92]	2.1B	143.6	122.4	-	82.3	-	-
BLIP-2 [44]	7.8B	144.5	121.6	-	82.2	-	-
GIT2 [78]	5.1B	145	126.9	148.6	81.7	67.3	71.0
Flamingo [2]	80B	138.1	-	-	82.0	54.1	65.7
PaLI [15]	17B	149.1	127.0	160.0△	84.3	58.8 / 73.1△	71.6 / 74.4△
PaLI-X [12]	55B	149.2	126.3	147 / 163.7△	86.0	71.4 / 80.8△	70.9 / 74.6△
			Generalist Models				
Unified-IO [55]	2.9B	-	100	-	77.9	-	57.4
Florence-2-B	0.23B	140.0	116.7	143.9	79.7	63.6	63.6
Florence-2-L	0.77B	143.3	124.9	151.1	81.7	73.5	72.6

▲ **VQA Task** (△ indicates usage of external OCR as input)

Method	#params	COCO Det. val2017 mAP	Flickr30k test R@1	Refcoco val	Refcoco test-A	Refcoco test-B	Refcoco+ val	Refcoco+ test-A	Refcoco+ test-B	Refcocog val	Refcocog test	Refcoco RES val
						Accuracy			Accuracy		Accuracy	mIoU
						Specialist Models						
SeqTR [99]	-	-	-	83.7	86.5	81.2	71.5	76.3	64.9	74.9	74.2	-
PolyFormer [49]	-	-	-	90.4	92.9	87.2	85.0	89.8	78.0	85.8	85.9	76.9
UNINEXT [84]	0.74B	60.6	-	92.6	94.3	91.5	85.2	89.6	79.8	88.7	89.4	-
Ferret [90]	13B	-	-	89.5	92.4	84.4	82.8	88.1	75.2	85.8	86.3	-
						Generalist Models						
UniTAB [88]		-	-	88.6	91.1	83.8	81.0	85.4	71.6	84.6	84.7	-
Florence-2-B	0.23B	41.4	84.0	92.6	94.8	91.5	86.8	91.7	82.2	89.8	82.2	78.0
Florence-2-L	0.77B	43.4	85.2	93.4	95.3	92.0	88.3	92.9	83.6	91.2	91.7	80.5

▲ **Region-level tasks (Reigion proposal, object detection, ...)**

Experiments > Downstream Tasks Fine-tuning > Object Detection

First

Backbone	Pretrain	Mask R-CNN		DINO
		AP_b	AP_m	AP
ViT-B [46]	MAE, IN-1k	51.6	45.9	55.0
Swin-B [51]	Sup IN-1k	50.2	-	53.4
Swin-B [51]	SimMIM [83]	52.3	-	-
FocalAtt-B [86]	Sup IN-1k	49.0	43.7	-
FocalNet-B [85]	Sup IN-1k	49.8	44.1	54.4
ConvNeXt v1-B [52]	Sup IN-1k	50.3	44.9	52.6
ConvNeXt v2-B [81]	Sup IN-1k	51.0	45.6	-
ConvNeXt v2-B [81]	FCMAE	52.9	46.6	-
DaViT-B [20]	*Florence-2*	53.6	46.4	59.2

Table 7. **COCO object detection and instance segmentation results** using Mask-RCNN framework, and **COCO object detection results** using DINO-4scale framework. All the entries use a base size model to ensure a fair comparison. For Mask-RCNN experiments, our method utilizes 1× schedule (12 epochs), ViT-B use 100 epochs, all others use 3× (36 epochs). For DINO experiments, all the entries use 1× schedule except for ViT-B which uses 50 epochs.

Base model인 **DaViT**가 다른 접근방식에 비해 강한 성능 향상을 이루었다.

Second

Pretrain	Frozen stages	Mask R-CNN		DINO	UperNet
		AP_b	AP_m	AP	mIoU
Sup IN1k	n/a	46.7	42.0	53.7	49
UniCL [87]	n/a	50.4	45.0	57.3	53.6
Florence-2	n/a	53.6	46.4	59.2	54.9
Florence-2	[1]	53.6	46.3	59.2	54.1
Florence-2	[1, 2]	53.3	46.1	59.0	54.4
Florence-2	[1, 2, 3]	49.5	42.9	56.7	49.6
Florence-2	[1, 2, 3, 4]	48.3	44.5	56.1	45.9

Table 8. Downstream task fine-tuning on COCO and ADE20K dataset. **COCO object detection** using Mask R-CNN and DINO. **ADE20K semantic segmentation** using UperNet. All entries use DaViT-B with 80M parameters as the backbone and standard 1× schedule.

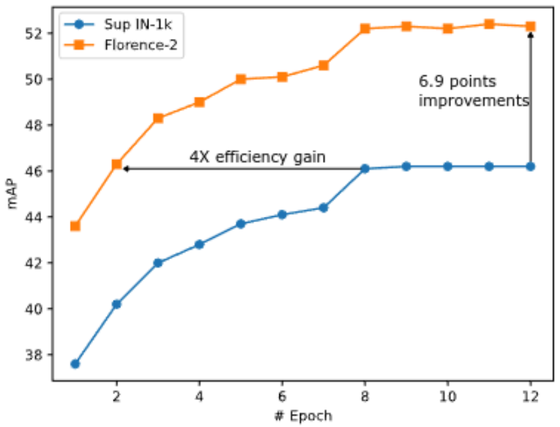
비싼 **fine-tuning없이**도 pre-training은 **일반화된 표현**을 학습할 수 있었다.

Experiments > Downstream Tasks Fine-tuning > Object Detection

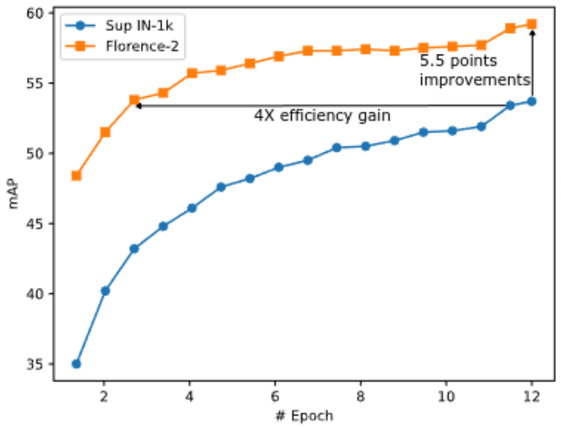
Third

Pretrain	Frozen stages	Mask R-CNN		DINO	UperNet
		AP_b	AP_m	AP	mIoU
Sup IN1k	n/a	46.7	42.0	53.7	49
UniCL [87]	n/a	50.4	45.0	57.3	53.6
Florence-2	n/a	53.6	46.4	59.2	54.9
Florence-2	[1]	53.6	46.3	59.2	54.1
Florence-2	[1, 2]	53.3	46.1	59.0	54.4
Florence-2	[1, 2, 3]	49.5	42.9	56.7	49.6
Florence-2	[1, 2, 3, 4]	48.3	44.5	56.1	45.9

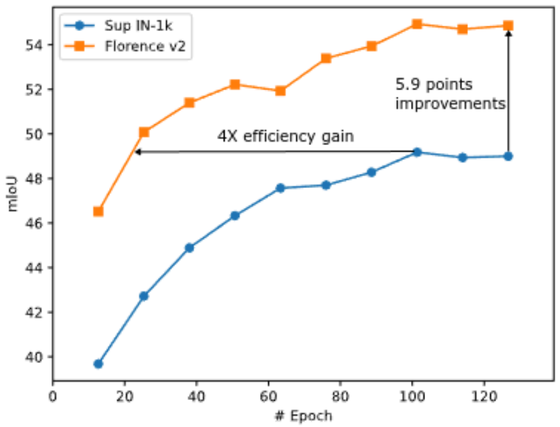
Table 8. Downstream task fine-tuning on COCO and ADE20K dataset. **COCO object detection** using Mask R-CNN and DINO. **ADE20K semantic segmentation** using UperNet. All entries use DaViT-B with 80M parameters as the backbone and standard 1× schedule.



(a) Mask-RCNN on COCO detection.



(b) DINO on COCO detection.



(c) UpperNet on ADE20K.

Pre-training은 더 좋은 훈련 **효율**을 가져왔다.

Experiments > Downstream Tasks Fine-tuning > Semantic segmentation

Backbone	Pretrain	mIoU	ms-mIoU
ViT-B [24]	Sup IN-1k	47.4	-
ViT-B [24]	MAE IN-1k	48.1	-
ViT-B [4]	BEiT	53.6	54.1
ViT-B [59]	BEiTv2 IN-1k	53.1	-
ViT-B [59]	BEiTv2 IN-22k	53.5	-
Swin-B [51]	Sup IN-1k	48.1	49.7
Swin-B [51]	Sup IN-22k	-	51.8
Swin-B [51]	SimMIM [83]	-	52.8
FocalAtt-B [86]	Sup IN-1k	49.0	50.5
FocalNet-B [85]	Sup IN-1k	50.5	51.4
ConvNeXt v1-B [52]	Sup IN-1k	-	49.9
ConvNeXt v2-B [81]	Sup IN-1k	-	50.5
ConvNeXt v2-B [81]	FCMAE	-	52.1
DaViT-B [20]	*Florence-2*	54.9	55.5

Table 9. **ADE20K semantic segmentation results** using UperNet. The input size is 512×512 for all the entries, except for models with BEiT pre-trained, which use the input size of 640×640.

Object detection과 동일한 흐름을 가진다.

Experiments > Ablation Studies > Multitask transfer

Multitask transfer

- **Image level** Model: pre-trained on **image**-level tasks only
- **Image-Region level** Model: pre-trained on **image**-level and **region**-level tasks
- **Image-Region-Pixel** Model: pre-trained on **image**-level, **region**-level, and **pixel**-level tasks

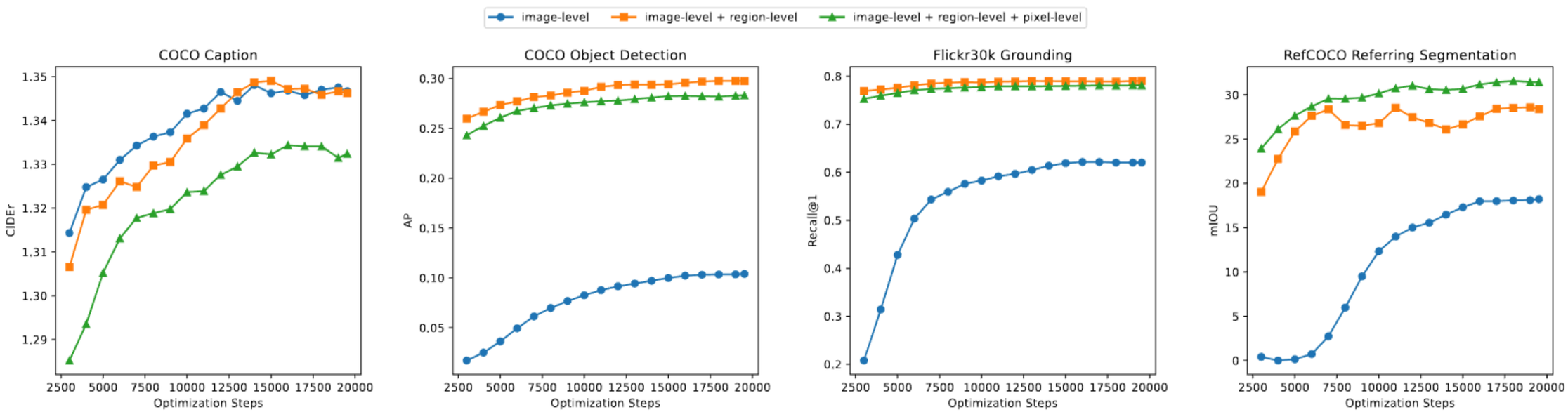


Figure 7. **Multitask transfer.** We conduct experiments with three different versions of *Florence-2* models, each trained on a different level of image annotation: image level, image and region level, and image, region, and pixel level. We then evaluate the transfer learning performance of these models on four downstream tasks: COCO caption, COCO object detection, Flickr30k grounding, and Refcoco referring segmentation.

Experiments > Ablation Studies > Scaling

Model scaling

Model	Caption	Detection	Grounding	RES	
	CIDEr	AP	Recall@1	mIOU	oIOU
Base	118.7	19.7	76.3	18.6	17.8
Large	**124.4**	**22.6**	**78.2**	**21.5**	**19.1**

Table 10. **Model scaling.** Zero-shot performance on COCO caption and COCO object detection, Flickr30k grounding, RefCOCO referring expression segmentation(RES).

모델 사이즈에 따른 네 가지 하위 작업에 대한 제로샷 성능 비교.
Large 모델은 다양한 하위 작업에서 기본 모델보다 확실히 더 나은 성능을 나타낸다.

Data scaling

Data size	Caption	Detection	Grounding	RES	
	CIDEr	AP	Recall@1	mIOU	oIOU
0.12M	102.8	16.1	74.0	15.9	16.6
0.36M	114.3	18.7	75.8	16.6	16.4
1.2M	118.1	18.9	76.3	**19.3**	**18.4**
12M	**118.7**	**19.7**	**76.3**	18.6	17.8

Table 11. **Data scaling.** Zero-shot performance on COCO caption, COCO object detection, Flickr30k grounding, COCORef referring segmentation.

데이터 사이즈에 따른 네 가지 하위 작업에 대한 제로샷 성능 비교.
Large 모델은 다양한 하위 작업에서 기본 모델보다 확실히 더 나은 성능을 나타낸다.

Experiments > Ablation Studies > Training settings

Training settings

V Pre	L Pre	Caption CIDEr	Detection AP	Grounding Recall@1	RES mIOU	RES oIOU
		Freeze Vision Encoder				
✓	✓	120.0	6.9	66.3	9.9	13.6
		Unfreeze Vision Encoder				
	✓	81.3	4.9	69.0	15.3	15.6
✓		117.4	19.6	75.2	**21.5**	**19.3**
✓	✓	**118.7**	**19.7**	**76.3**	18.6	17.8

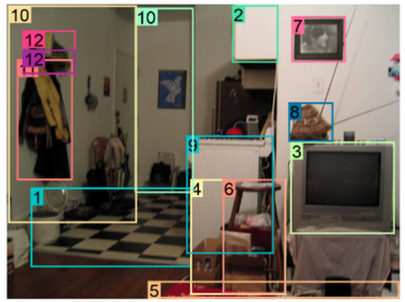
Table 12. **Basic components.** Zero-shot performance on COCO caption, COCO object detection, Flickr30k grounding, and COCORef referring segmentation. V Pre and L Pre indicate that using vision and language pre-training initialization, respectively.

Experiments > Qualitative evaluation

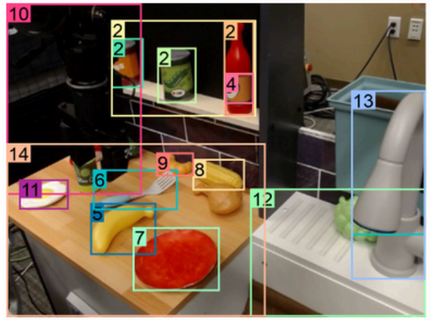
Visual Grounding

Prompt: Locate the phrases in the caption: {caption}

The image shows a group of five cartoon monsters. On the left side, there is a brown monster[1] with horns and a big smile on its face. Next to it, there are two smaller monsters[2], one black and one green. The black monster[3] has two large horns on its head and is standing in the center of the group. The green monster[4] on the right side is a green monster with big eyes and a long antennae. It is standing on its hind legs with its arms stretched out to the sides. In the middle of the image, there appears to be a small blue monster[5] with a round head and two antennae on its back. The background is light beige with small green circles scattered around.



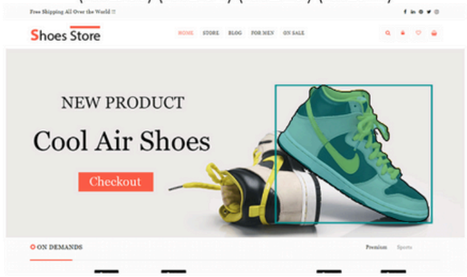
The image shows a cluttered room with a black and white checkered floor[1]. On the right side of the image, there is a small white cabinet[2] with a televi sion[3] on top of it. Next to the cabinet, there are several items[4] scattered on the floor, including a red blanket[5], a wooden stool[6], and a pile of trash. On top of the cabinet is a picture frame[7] and a hat[8]. In the center of the room is a white refrigerator[9] with a few items on top. The walls[10] are painted white and there are a few clothes[11] hanging on a rack[12] on the left wall. The room appears to be in disarray, with some items strewn about and others scattered around.



The image shows a kitchen countertop with various kitchen items on it. On the left side of the countertop, there is a microscope with a black body and a white lens[1]. Next to the microscope, there are two bottles of condiments[2] - one with a red label[3] and the other with green. On top of the microscope is a yellow banana[4], a blue spatula[5], a red plate[6], and a yellow corn[7][8] on the cob. In the center of the image, there appears to be a frying pan[10] with a fried egg[11] on it, and on the right side is a white sink[12] with a white faucet[13]. The countertop[14] is made of wood and has a gray tile backsplash.

Region to Segmentation

Prompt: What is the polygon mask of region ⟨loc_586⟩⟨loc_294⟩⟨loc_929⟩⟨loc_814⟩

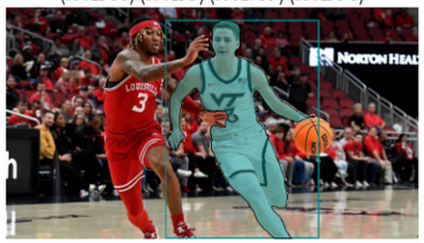


Prompt: What is the polygon mask of region ⟨loc_317⟩⟨loc_314⟩⟨loc_893⟩⟨loc_904⟩

Prompt: What is the polygon mask of region ⟨loc_541⟩⟨loc_266⟩⟨loc_692⟩⟨loc_627⟩

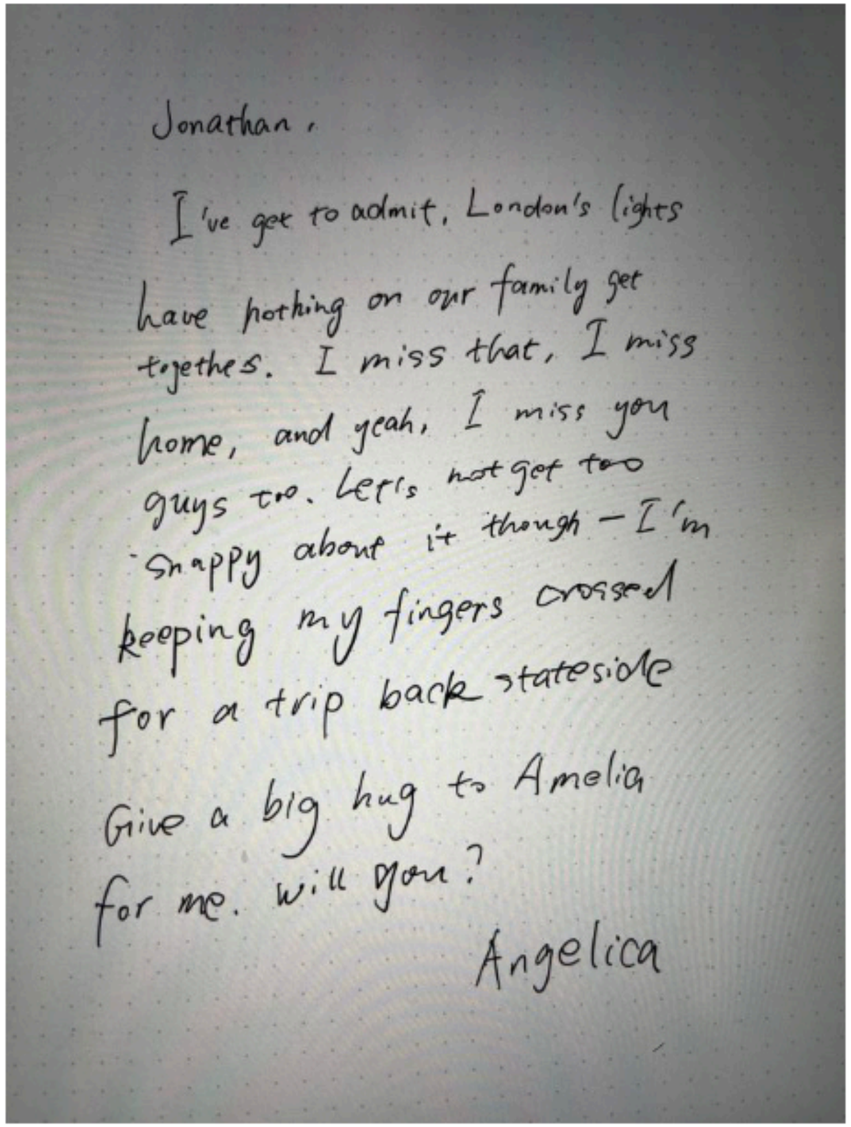
Prompt: What is the polygon mask of region ⟨loc_583⟩⟨loc_66⟩⟨loc_794⟩⟨loc_331⟩

Prompt: What is the polygon mask of region ⟨loc_386⟩⟨loc_53⟩⟨loc_759⟩⟨loc_998⟩



Prompt: What is the polygon mask of region ⟨loc_102⟩⟨loc_7⟩⟨loc_375⟩⟨loc_648⟩

Experiments > Qualitative evaluation



Florence-2: The image is a photograph of a handwritten note from a person named Angelica. The note is written in black ink on a white background. The handwriting is neat and legible, and it appears to be a handwritten letter. The person's name, Jonathan, is written at the top of the note in a cursive font. Below the name, there is a note that reads, "I've got to admit, London's lights have nothing on our family get together. I miss that, I miss you guys too. Let's not get too happy about it though I'm keeping my fingers crossed for a trip back stateside. Give a big hug to Amelia for me. Will you?"

GPT-4V[a]: The image shows a handwritten note addressed to Jonathan. The note conveys a personal message from the sender, Angelica, who admits to missing family gatherings and the feeling of home, contrasting it with the lights of London. Angelica expresses a longing to return stateside and asks Jonathan to give Amelia a big hug on her behalf. The tone of the note is casual and warm, with a sense of fondness and a hint of homesickness. The note is written informally, suggesting a close relationship between the sender and the recipients.

LLaVA[b]: The image is a handwritten letter on a piece of paper, written by a person named Jonathan. The letter is addressed to someone named Angela, and it appears to be a heartfelt message expressing love and affection. The letter is written in cursive, giving it a personal and intimate touch.

MiniGPT-4[c]: The image is a handwritten letter from an unknown person to someone named Angel. The letter is written in black ink on a piece of white paper. The letter is addressed to Angel and begins with the words, \Dear Angel,". The letter goes on to describe the writer's feelings about a recent event or situation, but the details are unclear. The letter ends with the words, \Sincerely, [Writer's Name]."

Experiments > Qualitative evaluation

Florence-2: The image shows a highway with multiple lanes. There are several vehicles on the road, including a black van and a red truck. On the left side of the image, there is a white semi-truck with the word "Martini" written on it. Above the highway, there are several bridges with cars and trucks on them. The sky is blue and there are trees and grass on the right side. In the center of the highway is a green sign that reads "Exit 94, Chamblee Tucker Rd, Mercer Univ."

GPT-4V[a]: The image shows a highway scene with several lanes of traffic. There are overpasses with cars and trucks traveling over them, and the highway itself appears to be busy with vehicles, including a prominent black van in the foreground.
A road sign indicates that Exit 94 for Chamblee-Tucker Rd and Mercer University is 1 1/4 miles ahead. The presence of the sign suggests that drivers in the right lanes will soon have the opportunity to take the exit to reach these destinations.
The infrastructure suggests a well-traveled interstate or highway system, possibly in an urban or suburban area given the overpass complexity and road signage for local roads and institutions. The vehicles on the road are diverse, indicating a mix of commercial and personal transport.

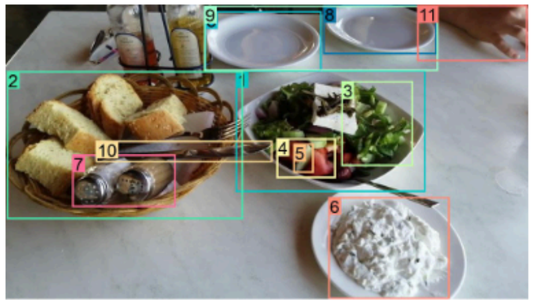
LLaVA[b]: The image depicts a busy highway with multiple lanes of traffic, including cars and trucks. There is a large truck driving on the highway, and several other trucks are also visible in the scene. The highway is filled with cars, and there is a traffic sign above the lanes, indicating the direction to Tucker Road. The scene captures the hustle and bustle of a typical day on a highway filled with vehicles.

MiniGPT-4[c]: The image shows a highway with multiple lanes of traffic moving in both directions. There are several large bridges spanning the highway, and in the distance, there are mountains and a blue sky. There are also several large trucks and cars on the highway, including a black van in the center of the image. The road surface is smooth and well-maintained.

Experiments > Qualitative evaluation

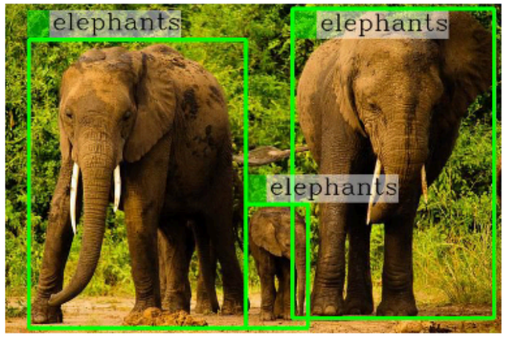
The image features a white dining table with a variety of food items on it. On the table, there is a bowl of bread, a bowl with a salad, and a plate with a side of yogurt. In addition to the food, there are two cups placed on the table. One cup is located near the left side of the table and the other cup is situated on the right side. The table is set with a fork and a knife, ready for a meal.

(c) Result from Kosmos-2.



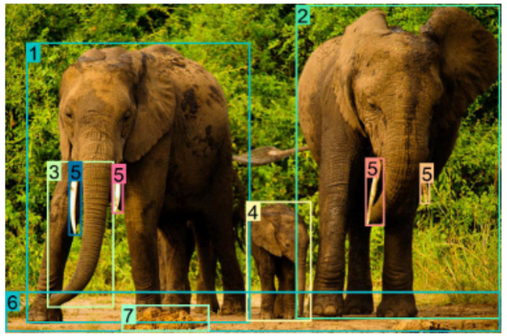
The image shows a table with a basket of bread and a plate of salad[1]. The basket[2] is made of woven straw and has several slices of bread in it. Next to the basket, there is a small bowl[1] of salad with a variety of vegetables, including lettuce[3], tomatoes[4], cucumbers[5], and feta cheese[6]. There are also two salt[7] and pepper shakers[7] on the table. On the right side of the table, there are two white plates[8,9] with a dollop of white sauce on them. The table is covered with a white tablecloth and there are a few other dishes[9] and utensils[10] scattered around. A person"s hand[11] can be seen in the top right corner of the image.

(d) Result from *Florence-2*.



The image features a family of elephants walking together in a lush green forest. The elephants are walking in a line, with one of them walking in front of the rest of the family. The family consists of a mother and her two calves, with the mother and the two calves walking side by side. The forest is filled with green trees, adding to the natural beauty of the scene.

(e) Result from Kosmos-2.



The image shows a group of three elephants standing in a dirt field with trees and bushes in the background. The elephants[1] are standing close together, with the largest elephant in the center and two smaller ones on either side. The largest elephant[1,2] on the left is standing with its trunk[3] extended, while the smaller one[4] is standing next to it. All three elephants[1] have tusks[5] and appear to be in their natural habitat. The ground[6] is covered in dirt and there is a small pile of dirt[7] in front of them. The overall mood of the image is peaceful and serene.

(f) Result from *Florence-2*.

Demo code > modules

```
import cv2
import numpy as np
from PIL import Image as PILImage
import torch
from transformers import AutoProcessor, AutoModelForCausalLM
```

- **cv2**: OpenCV 라이브러리로, 비디오 캡처 및 이미지 처리를 담당
- **numpy**: 배열 및 행렬 연산을 위한 라이브러리
- **PILImage**: PIL(Pillow) 라이브러리의 이미지 클래스
- **torch**: PyTorch 라이브러리로, 딥러닝 모델을 실행하는 데 사용
- **transformers**: Hugging Face의 트랜스포머 모델을 사용하기 위한 라이브러리

Demo code > Camera Setting

```
### Set Camera ###
cap = cv2.VideoCapture(-1)

cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
```

- **cv2.VideoCapture(0 or -1)**: 기본 카메라 장치에서 비디오 스트림을 캡처
- **cap.set(cv2.CAP_PROP_FRAME_###, ###)**: 비디오 프레임 너비와 높이를 설정

Demo code > Init device & models

```python
### Init Device ###
device = "cuda:0" if torch.cuda.is_available() else "cpu"
torch_dtype = torch.float16 if torch.cuda.is_available() else torch.float32

### Prepare Models ###
model = AutoModelForCausalLM.from_pretrained("microsoft/Florence-2-base",
                                             torch_dtype=torch_dtype,
                                             trust_remote_code=True).to(device)
processor = AutoProcessor.from_pretrained("microsoft/Florence-2-base", trust_remote_code=True)
```

- **device**: GPU가 사용 가능한 경우 "cuda:0"을, 그렇지 않으면 "cpu"를 사용
- **torch_dtype**: GPU가 사용 가능한 경우 float16을, 그렇지 않으면 float32를 사용
- **AutoModelForCausalLM**: 사전 학습된 언어 모델을 로드
- **AutoProcessor**: 입력 데이터를 전처리하는 프로세서를 로드

Demo code > functions

```
### Running Florence-2 function ###
def run_example(task_prompt, text_input=None, pil_image=None):

    if text_input is None:
        prompt = task_prompt
    else:
        prompt = task_prompt + text_input

    inputs = processor(text=prompt, images=pil_image, return_tensors="pt").to(device, torch_dtype)
    generated_ids = model.generate(input_ids=inputs["input_ids"],
                                   pixel_values=inputs["pixel_values"],
                                   max_new_tokens=1024,
                                   num_beams=3)
    generated_text = processor.batch_decode(generated_ids,
                                            skip_special_tokens=False)[0]
    parsed_answer = processor.post_process_generation(generated_text,
                                                      task=task_prompt,
                                                      image_size=(pil_image.width, pil_image.height))

    return parsed_answer
```

- **task_prompt**: task 프롬프트 입력
- **text_input**: 추가 text 프롬프트 입력
- **pil_image**: input 이미지 객체
- **inputs**: 프로세서를 통해 전처리된 입력 데이터
- **generated_ids**: 모델이 생성한 텍스트의 ID
- **generated_text**: 생성된 텍스트
- **parsed_answer**: 후처리된 결과

Demo code > functions

```python
### Running Florence-2 function ###
def run_example(task_prompt, text_input=None, pil_image=None):

    if text_input is None:
        prompt = task_prompt
    else:
        prompt = task_prompt + text_input

    inputs = processor(text=prompt, images=pil_image, return_tensors="pt").to(device, torch_dtype)
    generated_ids = model.generate(input_ids=inputs["input_ids"],
                                   pixel_values=inputs["pixel_values"],
                                   max_new_tokens=1024,
                                   num_beams=3)
    generated_text = processor.batch_decode(generated_ids,
                                            skip_special_tokens=False)[0]
    parsed_answer = processor.post_process_generation(generated_text,
                                                      task=task_prompt,
                                                      image_size=(pil_image.width, pil_image.height))

    return parsed_answer
```

- **task_prompt**: ex) <CAPTION>, <OD>, <DENSE_REGION_CAPTION>,<REGION_PROPOSAL>, ...
- **text_input**: ex) None, “a green car”, “<loc 100><loc 200><loc 150><loc 350>”, ...

Demo code > functions

inputs: 프로세서를 통해 전처리된 입력 데이터

```
{'input_ids': tensor([[    0,   574, 22486,     5,  8720,    19,  4120,   766,    11,     5,
          2274,     4,     2]], device='cuda:0'),
 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]], device='cuda:0'),
 'pixel_values': tensor([[[[-1.1416, -0.7480, -0.9189,  ..., -1.9297, -1.9463, -1.9639],
          [-0.7651, -0.7139, -0.9189,  ..., -1.9297, -1.9463, -1.9639],
          [-0.2683, -0.7310, -0.9707,  ..., -1.9463, -1.9639, -1.9805],
          ...,
          [ 0.5708,  0.5708,  0.5708,  ...,  0.3481,  0.3481,  0.3652],
          [ 0.5708,  0.5708,  0.5708,  ...,  0.3823,  0.3823,  0.3652],
          [ 0.5708,  0.5708,  0.5708,  ...,  0.4165,  0.3994,  0.3823]],

         [[-1.3877, -0.9854, -1.1602,  ..., -1.6504, -1.6680, -1.6855],
          [-1.0029, -0.9502, -1.1602,  ..., -1.6504, -1.6680, -1.6855],
          [-0.4951, -0.9678, -1.2129,  ..., -1.6680, -1.6855, -1.7031],
          ...,
          [ 0.6079,  0.6079,  0.6079,  ...,  0.3977,  0.3977,  0.4153],
          [ 0.6079,  0.6079,  0.6079,  ...,  0.4329,  0.4329,  0.4153],
          [ 0.6079,  0.6079,  0.6079,  ...,  0.4678,  0.4502,  0.4329]],

         [[-1.1768, -0.7759, -0.9502,  ..., -1.4902, -1.5078, -1.5254],
          [-0.7935, -0.7412, -0.9502,  ..., -1.4902, -1.5078, -1.5254],
          [-0.2881, -0.7588, -1.0029,  ..., -1.5078, -1.5254, -1.5430],
          ...,
          [ 0.8271,  0.8271,  0.8271,  ...,  0.5659,  0.5659,  0.5835],
          [ 0.8271,  0.8271,  0.8271,  ...,  0.6006,  0.6006,  0.5835],
          [ 0.8271,  0.8271,  0.8271,  ...,  0.6357,  0.6182,  0.6006]]]],
       device='cuda:0', dtype=torch.float16)}
```

generated_ids

```
tensor([[    2,     0,  5901, 50323, 50602, 51202, 51044, 11219, 50979, 50469,
         51177, 50816, 13630, 50974, 50845, 51136, 51041, 50414, 50853, 50579,
         51041,     2]], device='cuda:0')
```

generated_text

```
</s><s>
car<loc_54><loc_333><loc_933><loc_775>
door<loc_710><loc_200><loc_908><loc_547>
wheel<loc_705><loc_576><loc_867><loc_772>
     <loc_145><loc_584><loc_310><loc_772>
</s>
```

parsed_answer

```
{'<OD>': {
    'bboxes': [
        [34.880001068115234, 160.0800018310547, 597.4400024414062, 372.239990234375],
        [454.7200012207031, 96.23999786376953, 581.4400024414062, 262.79998779296875],
        [451.5199890136719, 276.7200012207031, 555.2000122070312, 370.79998779296875],
        [93.1199951171875, 280.55999755859375, 198.72000122070312, 370.79998779296875]],
    'labels': ['car', 'door', 'wheel', 'wheel']}}
```

Demo code > functions

```python
### Drawing bounding boxes ###
def draw_bboxes(image, bboxes, labels):
    for bbox, label in zip(bboxes, labels):
        x_min, y_min, x_max, y_max = map(int, bbox)
        cv2.rectangle(image, (x_min, y_min), (x_max, y_max), (0, 255, 0), 2)
        label_text = label
        cv2.putText(image, label_text, (x_min, y_min-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,255,0),2)
    return image
```

- **image**: 입력 이미지
- **bboxes**: 바운딩 박스 좌표
- **labels**: 바운딩 박스에 대한 레이블
- **cv2.rectangle**: 이미지에 사각형 추가
- **cv2.putText**: 이미지에 텍스트 추가

Demo code > Object Detection

```
### Main Loop ###
while True:
    ret, frame = cap.read()

    pil_image = PILImage.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

```
    ### Object Detection ###
    result = run_example(task_prompt="<OD>", pil_image=pil_image)
    bboxes = result['<OD>']['bboxes']
    labels = result['<OD>']['labels']
    print(labels)
```

```
    annotated_image = draw_bboxes(frame.copy(), bboxes, labels)

    cv2.imshow("display", annotated_image)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

- **cap.read()**: 비디오 프레임을 읽습니다.
- **PILImage.fromarray(...)**: OpenCV 이미지를 PIL 이미지로 변환합니다.

Demo code > Caption + Phrase Grounding

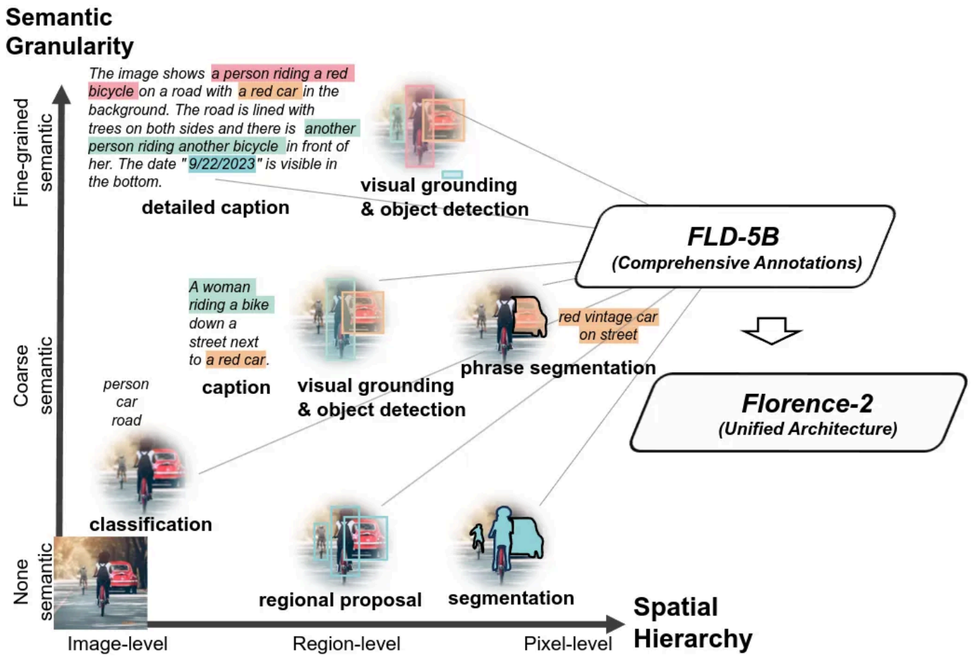
```
### Main Loop ###
while True:
    ret, frame = cap.read()

    pil_image = PILImage.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

```
    ### Caption to Phase Grounding ###
    # '<CAPTION>' / '<DETAILED_CAPTION>' / '<MORE_DETAILED_CAPTION>'
    task_caption = '<CAPTION>'
    result = run_example(task_prompt=task_caption, pil_image=pil_image)
    text_input = result[task_caption]
    result = run_example(task_prompt='<CAPTION_TO_PHRASE_GROUNDING>', text_input=text_input, pil_image=pil_image)
    result[task_caption] = text_input


    bboxes = result['<CAPTION_TO_PHRASE_GROUNDING>']['bboxes']
    labels = result['<CAPTION_TO_PHRASE_GROUNDING>']['labels']
    print(labels)
```

```
    annotated_image = draw_bboxes(frame.copy(), bboxes, labels)

    cv2.imshow("display", annotated_image)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

- 가능한 **task_caption**
  - <CAPTION>
  - <DETAILED_CAPTION>
  - <MORE_DETAILED_CAPTION>

Summary


Semantic Granularity
Fine-grained semantic
Coarse semantic
None semantic
The image shows a person riding a red bicycle on a road with a red car in the background. The road is lined with trees on both sides and there is another person riding another bicycle in front of her. The date "9/22/2023" is visible in the bottom.
detailed caption
visual grounding & object detection
FLD-5B
(Comprehensive Annotations)
A woman riding a bike down a street next to a red car.
caption
visual grounding & object detection
red vintage car on street
phrase segmentation
Florence-2
(Unified Architecture)
person
car
road
classification
regional proposal
segmentation
Image-level
Region-level
Pixel-level
Spatial Hierarchy

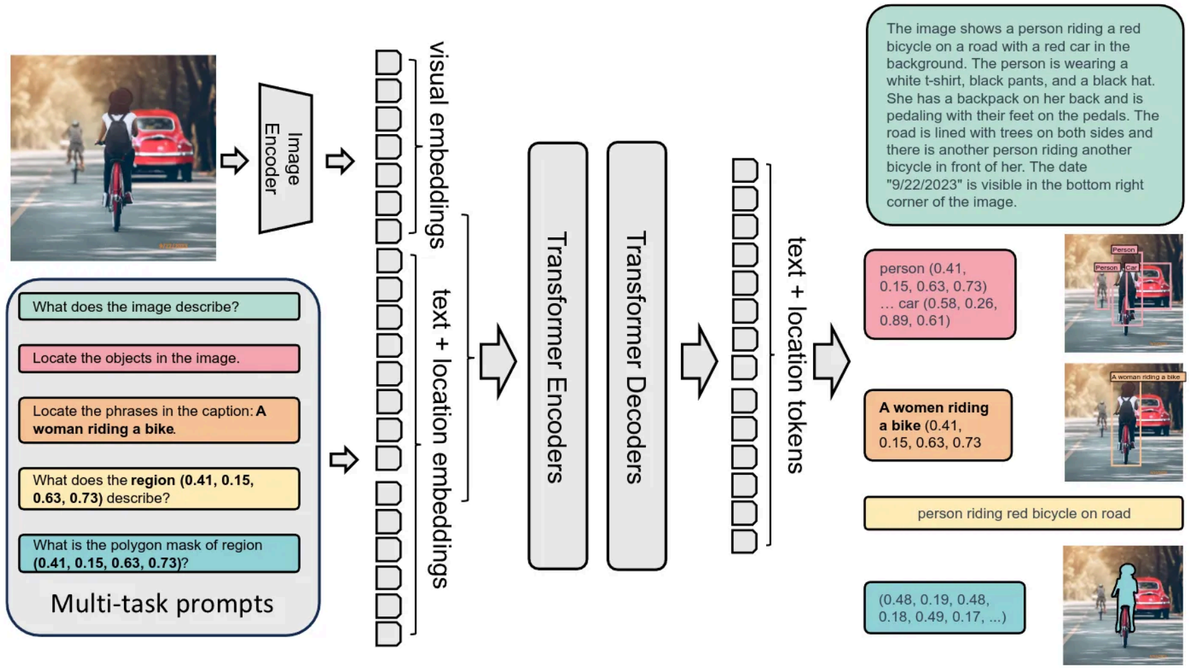


Image Encoder
visual embeddings
text + location embeddings
What does the image describe?
Locate the objects in the image.
Locate the phrases in the caption: A woman riding a bike.
What does the region (0.41, 0.15, 0.63, 0.73) describe?
What is the polygon mask of region (0.41, 0.15, 0.63, 0.73)?
Multi-task prompts
Transformer Encoders
Transformer Decoders
text + location tokens
The image shows a person riding a red bicycle on a road with a red car in the background. The person is wearing a white t-shirt, black pants, and a black hat. She has a backpack on her back and is pedaling with their feet on the pedals. The road is lined with trees on both sides and there is another person riding another bicycle in front of her. The date "9/22/2023" is visible in the bottom right corner of the image.
person (0.41, 0.15, 0.63, 0.73) … car (0.58, 0.26, 0.89, 0.61)
A women riding a bike (0.41, 0.15, 0.63, 0.73
person riding red bicycle on road
(0.48, 0.19, 0.48, 0.18, 0.49, 0.17, ...)

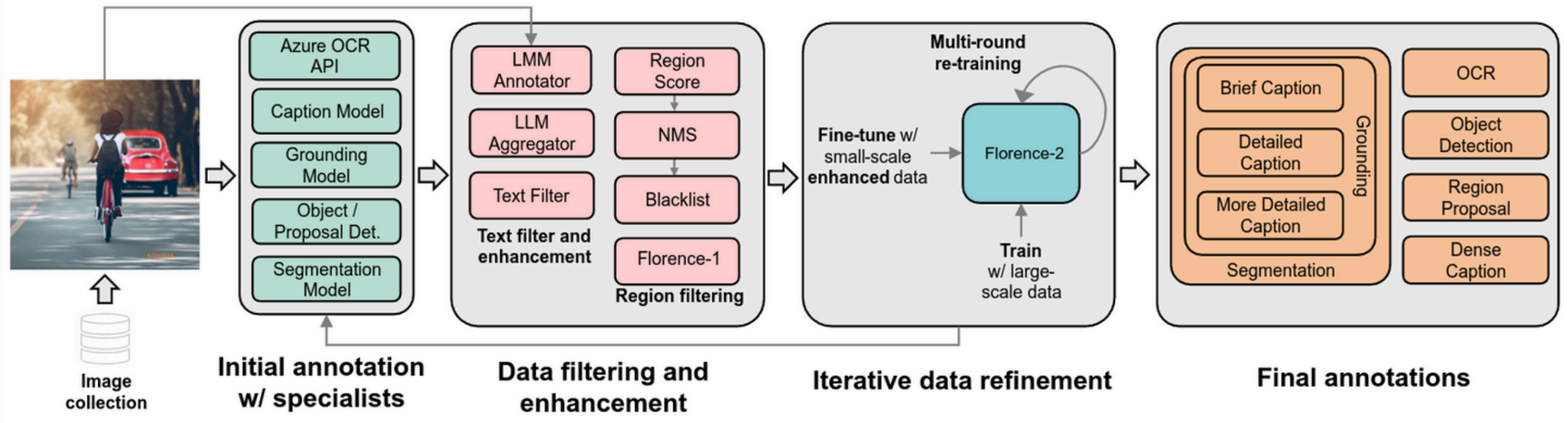


Image collection
Azure OCR API
Caption Model
Grounding Model
Object / Proposal Det.
Segmentation Model
Initial annotation w/ specialists
LMM Annotator
LLM Aggregator
Text Filter
Text filter and enhancement
Region Score
NMS
Blacklist
Florence-1
Region filtering
Data filtering and enhancement
Multi-round re-training
Fine-tune w/ small-scale enhanced data
Florence-2
Train w/ large-scale data
Iterative data refinement
Brief Caption
Detailed Caption
More Detailed Caption
Grounding
Segmentation
OCR
Object Detection
Region Proposal
Dense Caption
Final annotations