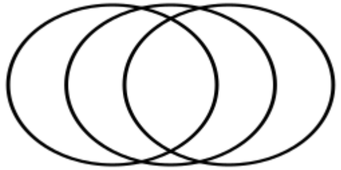
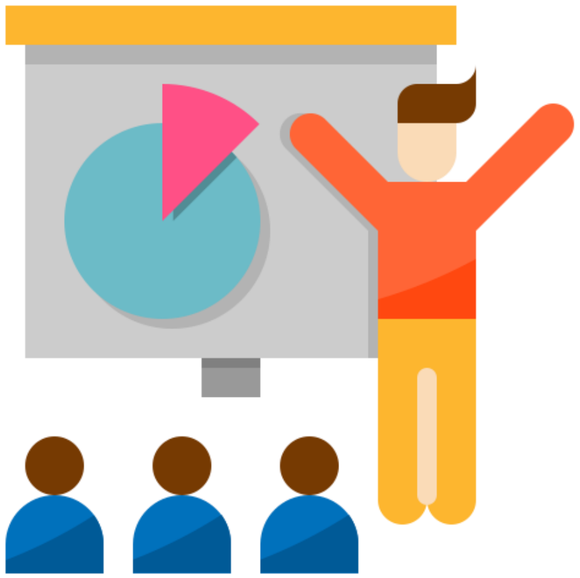
PAPER REVIEW

SAM1 & SAM2



카피바라팀	박현아, 배누리, 김호정, 전사영	2024.12.10

CON-TENTS

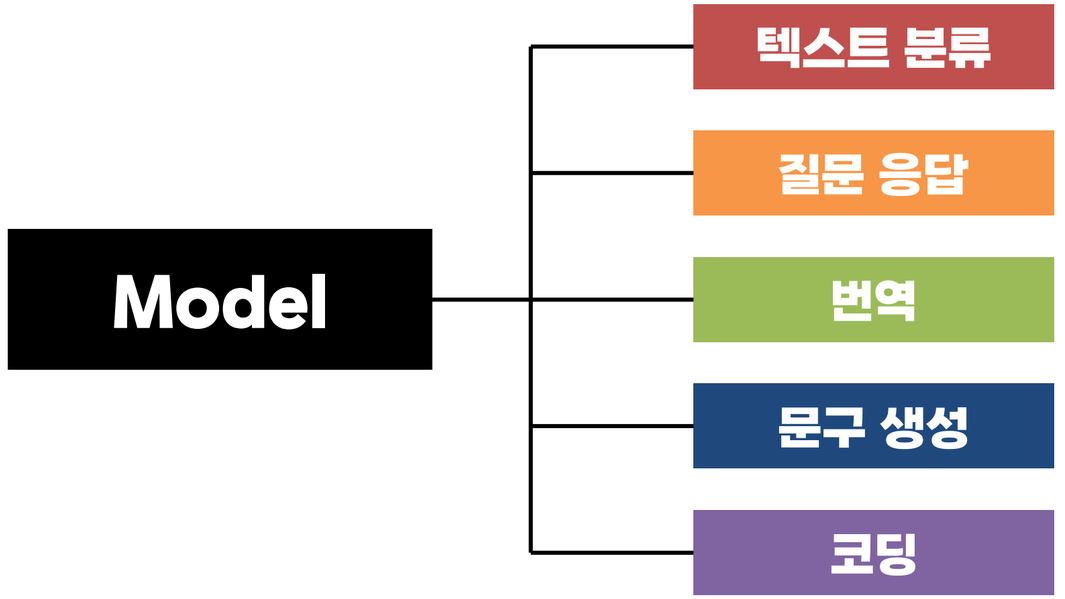


Part.01

Introduction

기존 NLP vs GPT

〈기존 NLP 모델〉

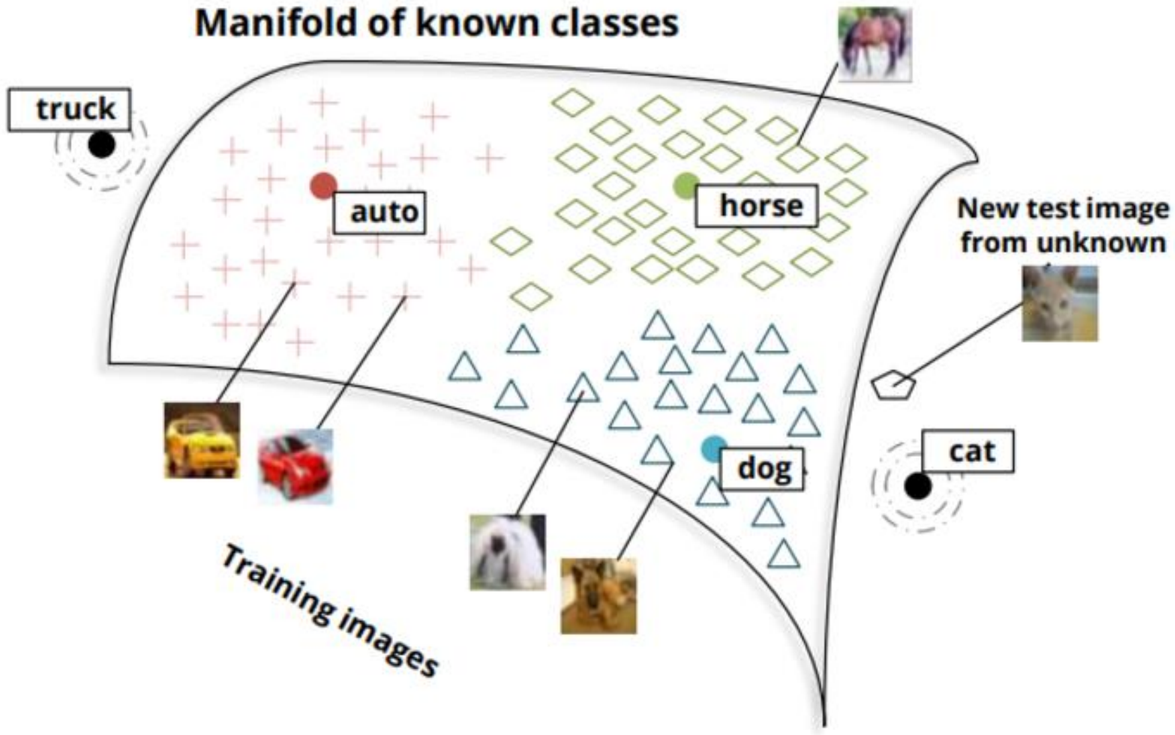


〈GPT〉

기존의 NLP 모델은 특정 태스크에 맞게 설계되고 훈련되었으나, GPT는 **하나의 모델로 여러 NLP 태스크를 수행할 수 있음**

Zero-shot learning

example



HOW?

- 대규모 사전 학습 pre-training
- 공통 표현 학습 semantic embedding
- 프롬프트 활용

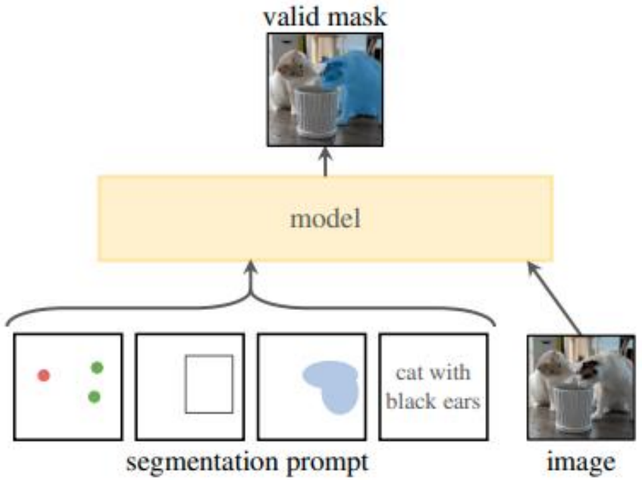
모델이 학습하지 않은 태스크나 클래스를 수행할 수 있는 능력.

즉, 모델이 훈련된 데이터에 포함되지 않은 새로운 카테고리를 이해하고 분류할 수 있게 함.

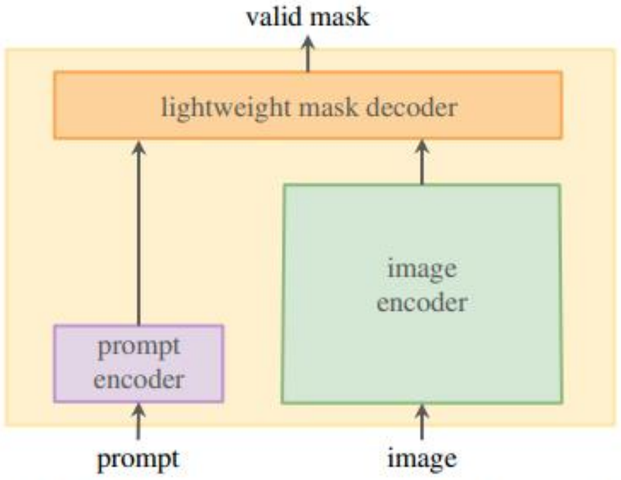
Part.02

SAM1

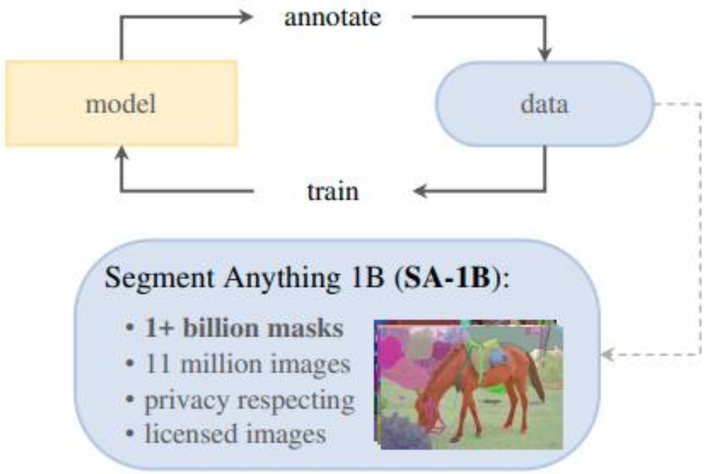
SAM1



(a) **Task**: promptable segmentation



(b) **Model**: Segment Anything Model (**SAM**)

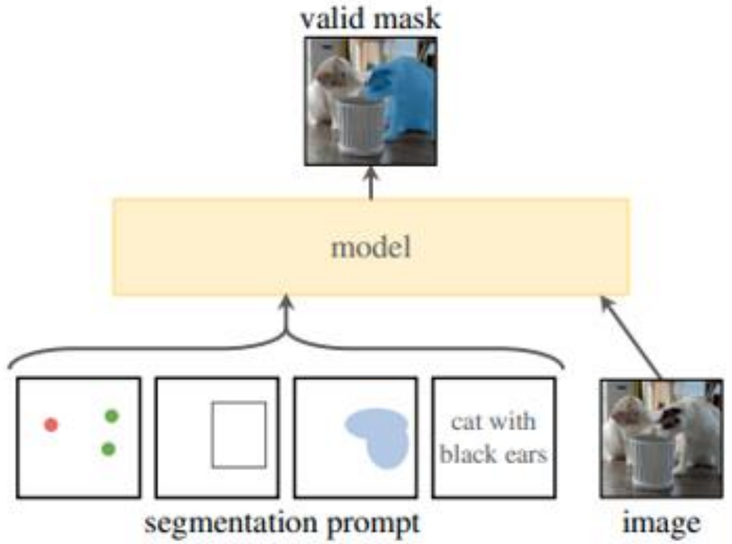


(c) **Data**: data engine (top) & dataset (bottom)

컴퓨터 비전 분야에서 다양한 태스크에 적용 가능한 범용 Foundation Model을 설계

Prompt를 활용하여 특정 또는 일반적 세그멘테이션 작업을 수행할 수 있는 모델을 설계하고, 광범위한 데이터셋으로 Pre-training함

Task



(a) **Task**: promptable segmentation

green point : 사용자가 관심 있는 영역을 지정하는 프롬프트

특정 위치(green point)에 대한 다양한 해석을 제공할 수 있음

이미지와 함께 해당 prompt를 네트워크의 입력으로 주면 그에 맞는 segmentation mask를 출력으로 내보냄

Model

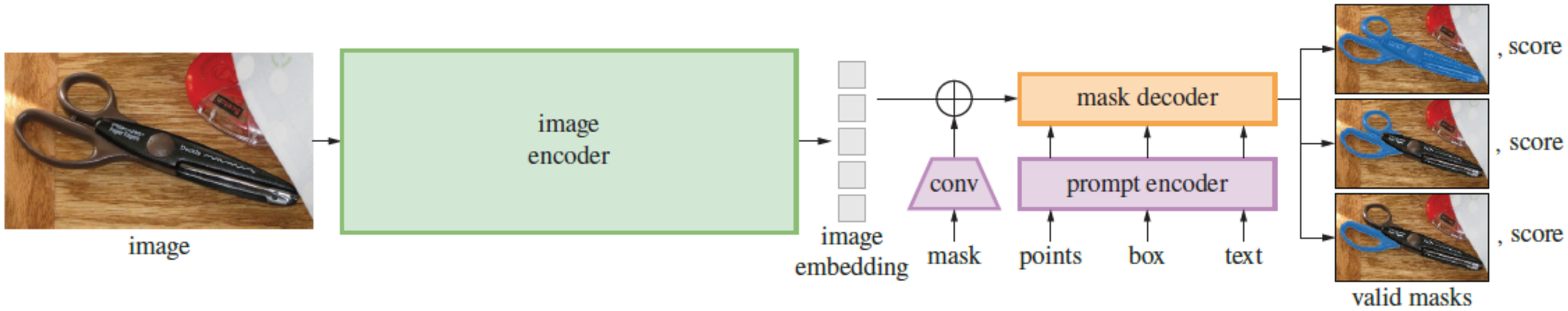
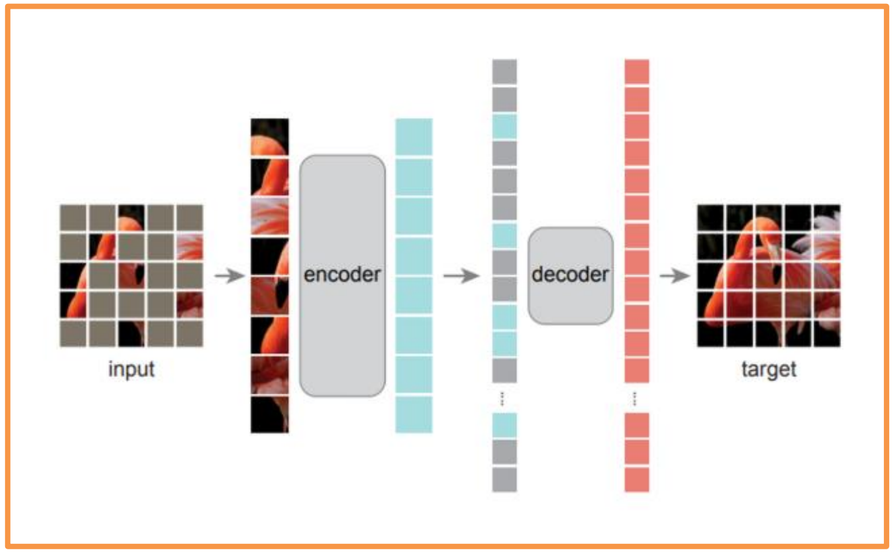
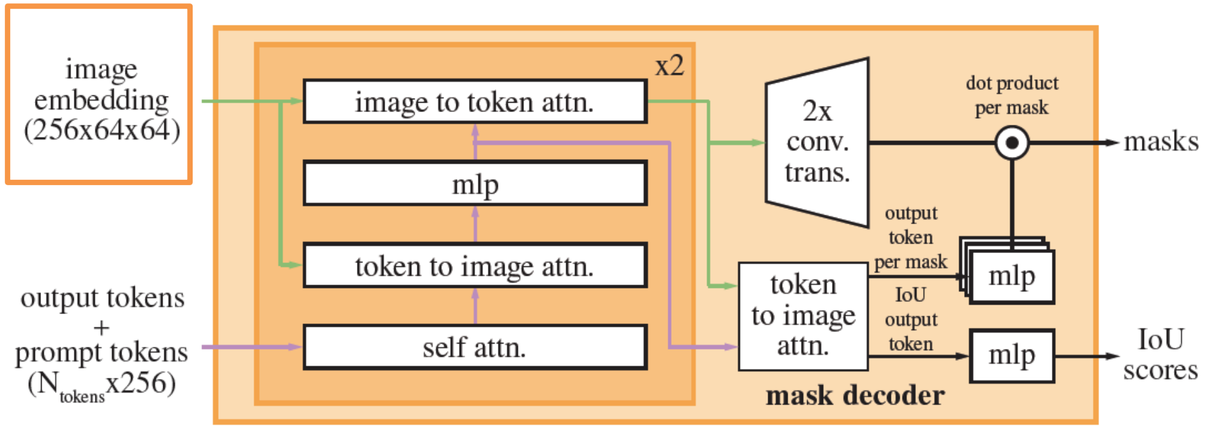


image encoder + prompt encoder + mask decoder

Model

Image encoder



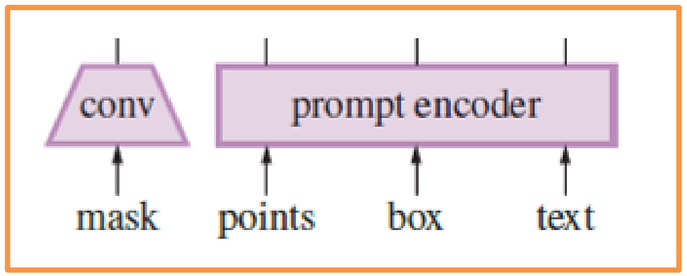


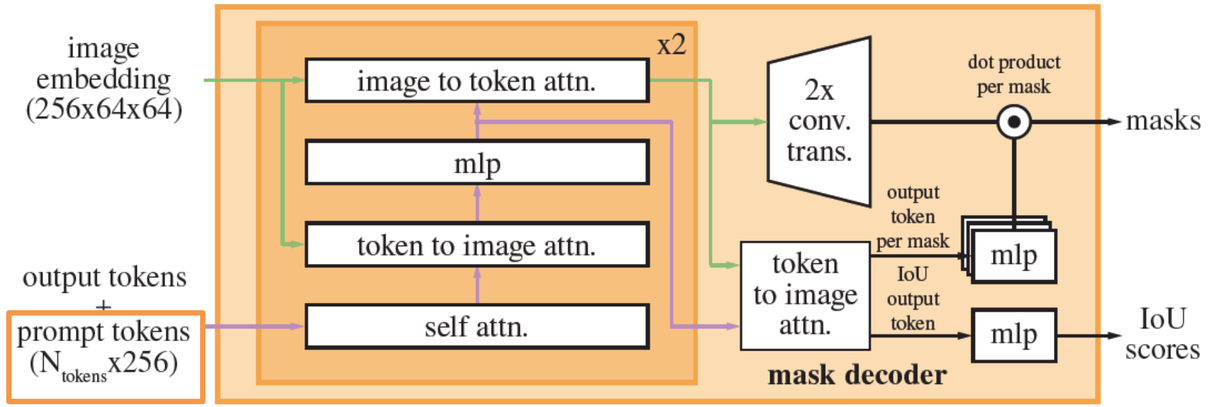
MAE(Masked auto-encoder) 방식으로 학습시킨 ViT 모델을 사용함

MAE는 이미지를 일정한 크기의 그리드로 나누고 랜덤하게 가린뒤, 복원하도록 모델을 학습시키는 기법

Model

Prompt encoder





- ***sparse***: points, boxes, text가 있음.

  point 혹은 box의 경우 positional encoding을 합쳐서 임베딩을 만들었고, text의 경우 CLIP의 text encoder를 활용함.

- ***dense***: masks. 일반적인 convolution 연산을 통해서 임베딩을 생성한 뒤, 이미지와 element-wise로 합침

Model

Mask decoder

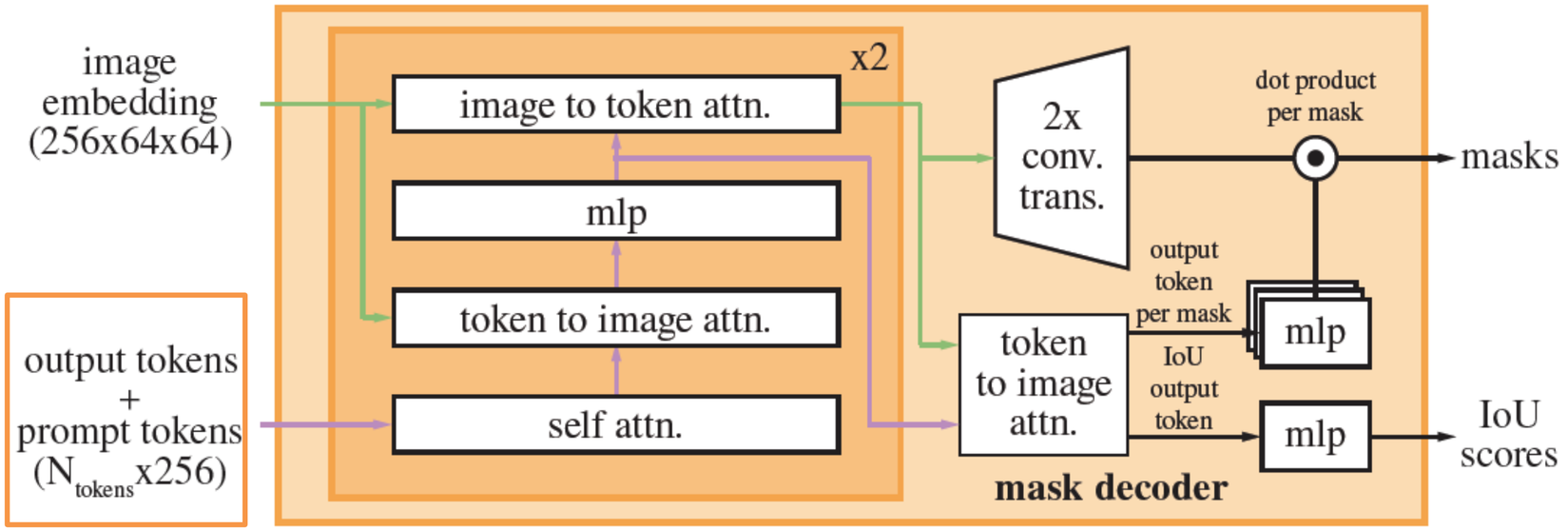
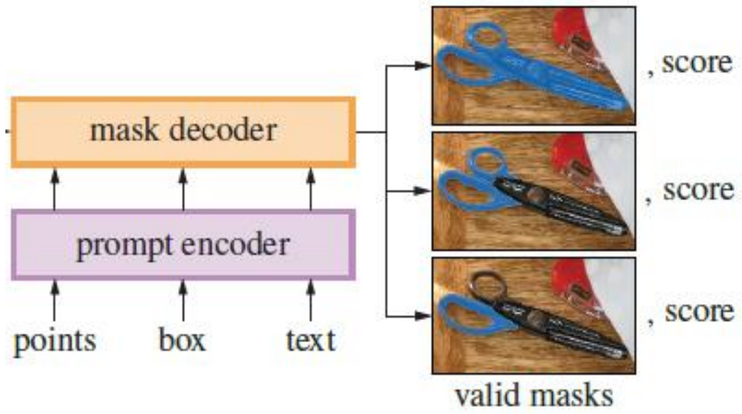


image embedding과 prompt embedding을 받아서 mask를 생성

전체 임베딩을 업데이트하기 위해 prompt-to-image와 image-to-prompt로 양방향으로 self-attention과 cross-attention을 수행함

masks와 IOU scores라는 두 출력 값을 내보냄

Model

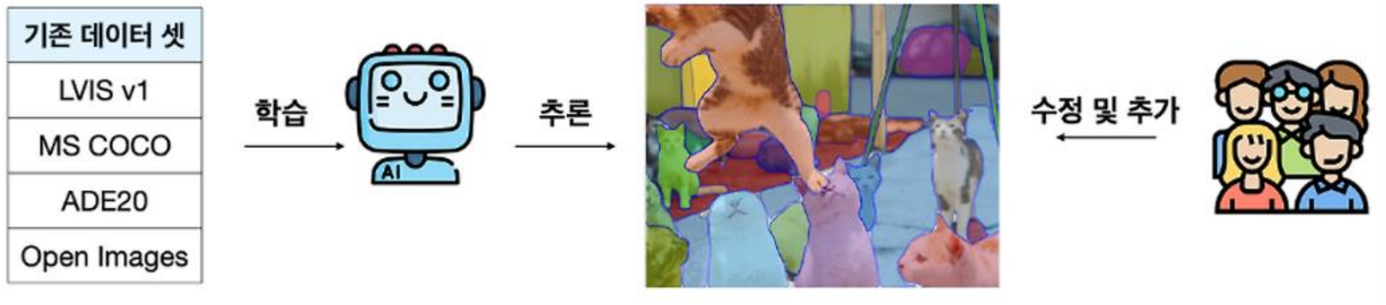


하나의 프롬프트에 3개의 마스크를 생성, loss 계산, 3개의 마스크 중 가장 작은 loss만 역전파 수행
-> ambiguity라는 segmentation task의 기본 특성을 반영하기 위함

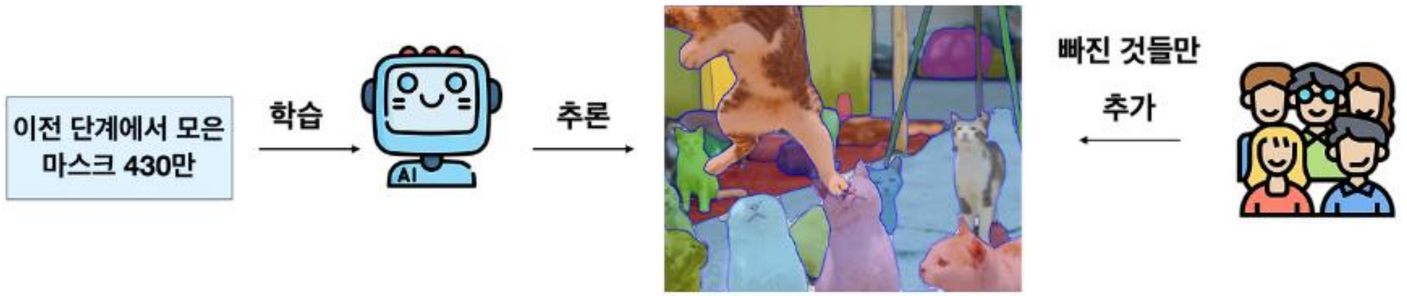
하나의 프롬프트라도 유저의 의도에 따라서 여러 마스크가 정답일 수 있음.
ex) 가방이자 사람인 픽셀

Data

1. Assisted Manual



2. Semi-automatic

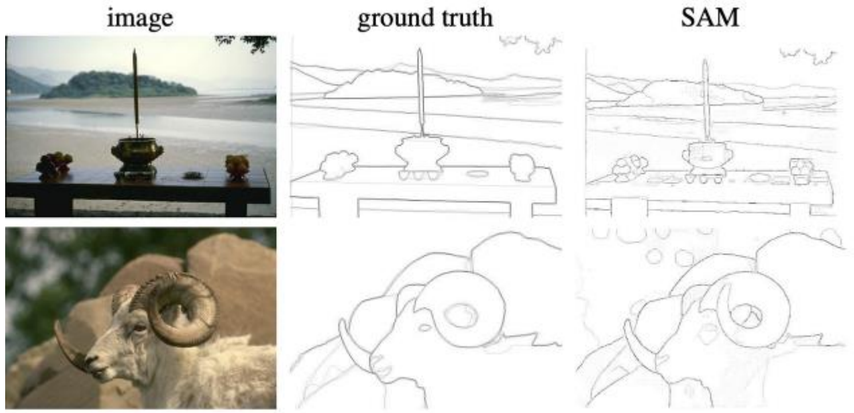


3. Fully automatic

Experiments

1. Singlepoint Segmentation

2. Edge Detection

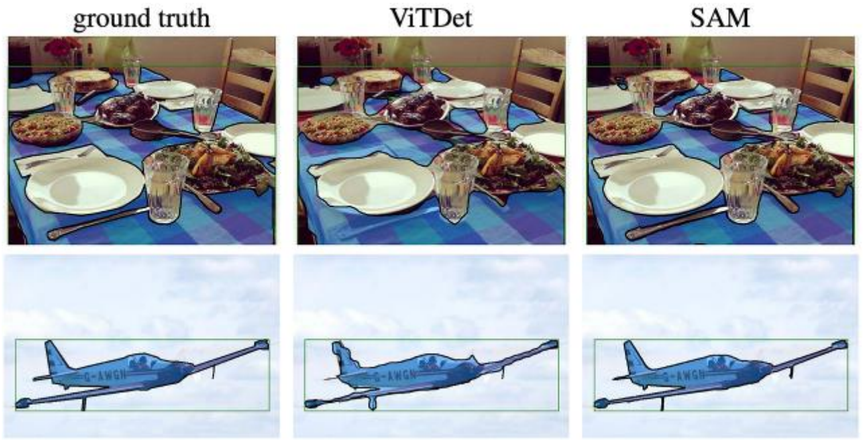


3. Object Proposal

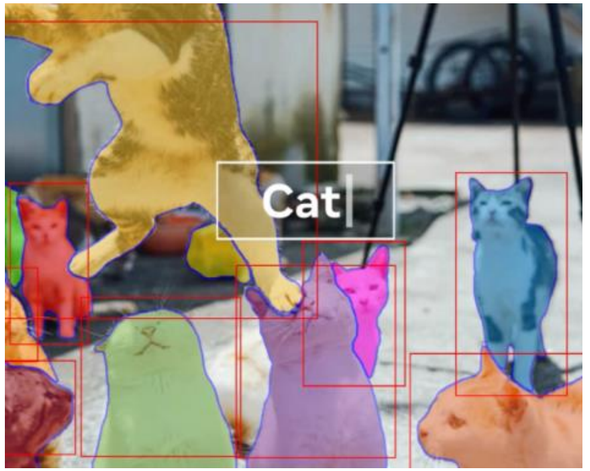
Object proposal

Object Detection

4. Instance Segmentation



5. Text to mask

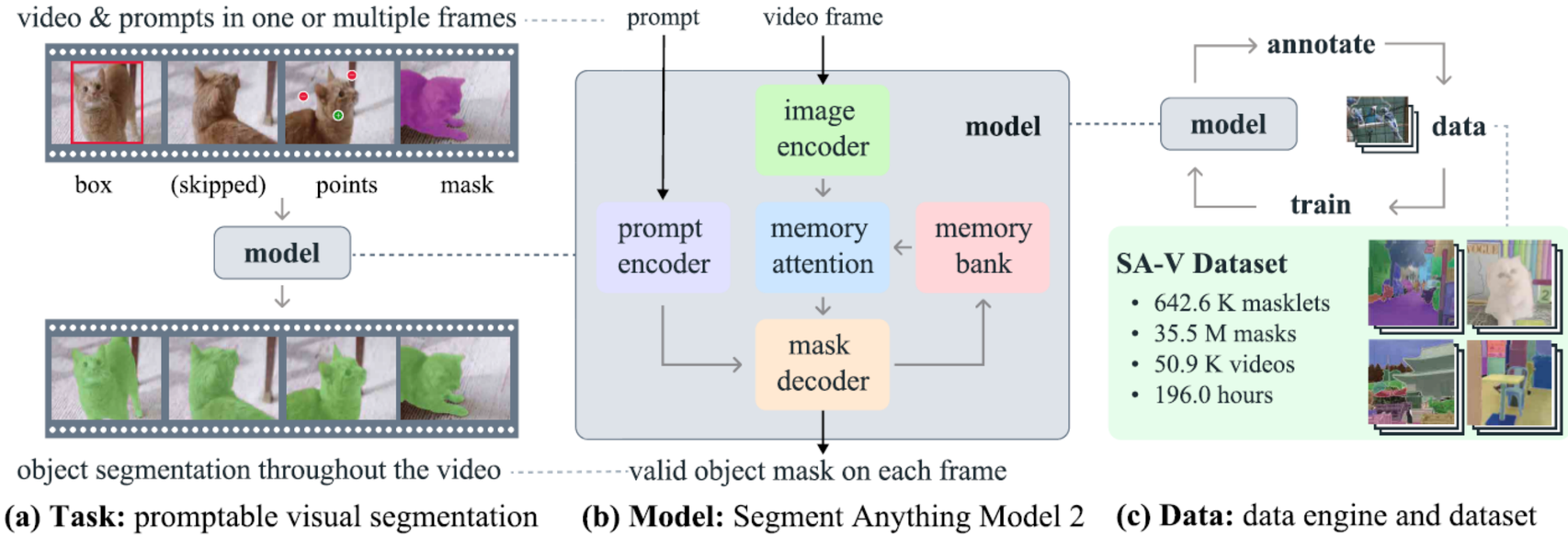


Part.03

SAM2

SAM2

이미지뿐만 아니라 비디오에서도 잘 작동할 수 있는 segmentation model을 구축하는 것을 목표

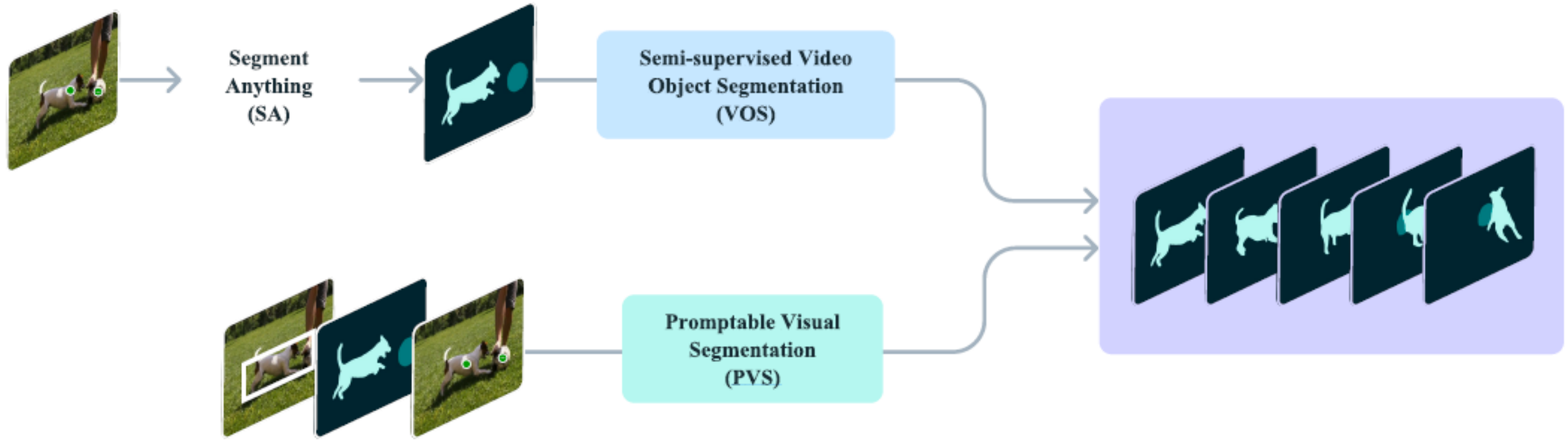


(a) Task: promptable visual segmentation **(b) Model:** Segment Anything Model 2 **(c) Data:** data engine and dataset

Video Segmentation의 과제

- Entity는 움직임, 변형, 가려짐, 조명 변화, 기타 요인들로 인해 모양이 크게 바뀔 수 있음
- 동영상은 카메라 동작, blur, 낮은 해상도로 인해 이미지보다 품질이 낮음
- 많은 수의 프레임을 효율적으로 처리하는 것이 중요함 (이미지는 프레임 하나)

Task : Promptable Visual Segmentation



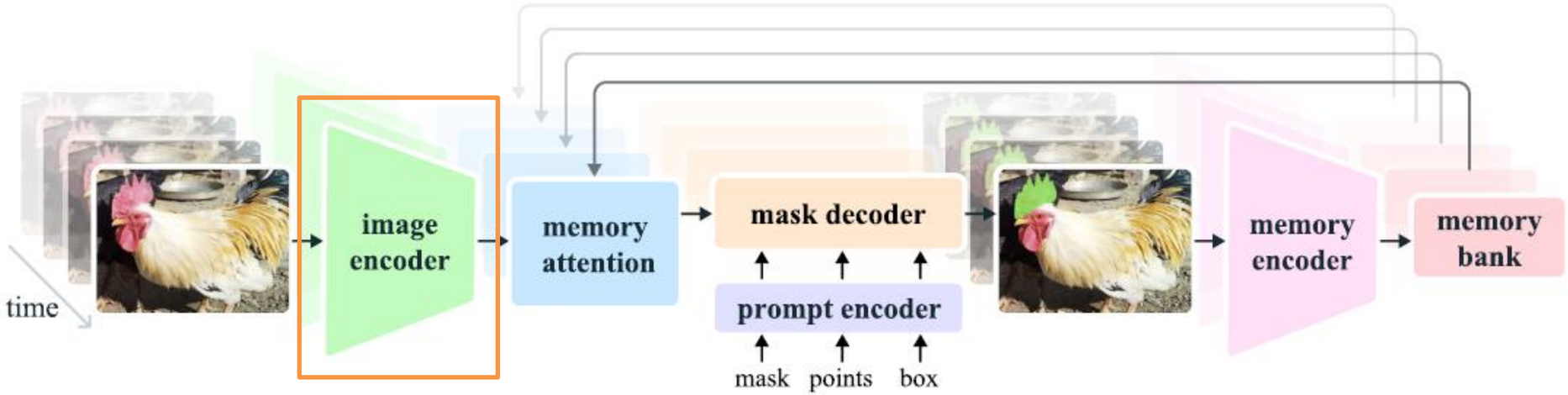
PVS task는 동영상의 모든 프레임에서 모델에 프롬프트를 제공할 수 있음.

프롬프트는 positive/negative click, bounding box, 또는 mask일 수 있으며, object를 정의하거나 모델에서 예측한 object를 정제하는 데 사용

초기 (1 or more) 프롬프트를 수신한 후 모델은 모든 동영상 프레임에서 object의 masklet을 얻음

Model

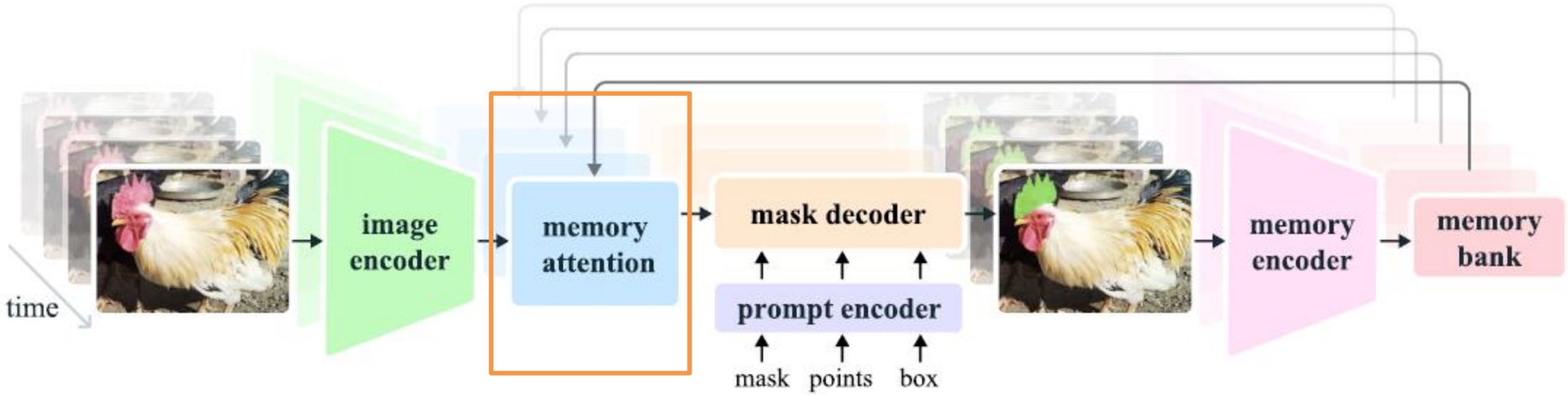
Image encoder



- 긴 동영상을 실시간 처리를 위해 ‘**streaming**’ 방식을 채택하여 처리
- Image Encoder는 전체 상호작용에서 한 번만 실행되며, 각 프레임을 나타내는 feature embedding vector을 제공하는 역할
- MAE로 사전학습된 Hiera Image Encoder를 사용하며, 계층적이기 때문에 디코딩 중에 multi-scale feature를 사용 가능

Model

Memory Attention



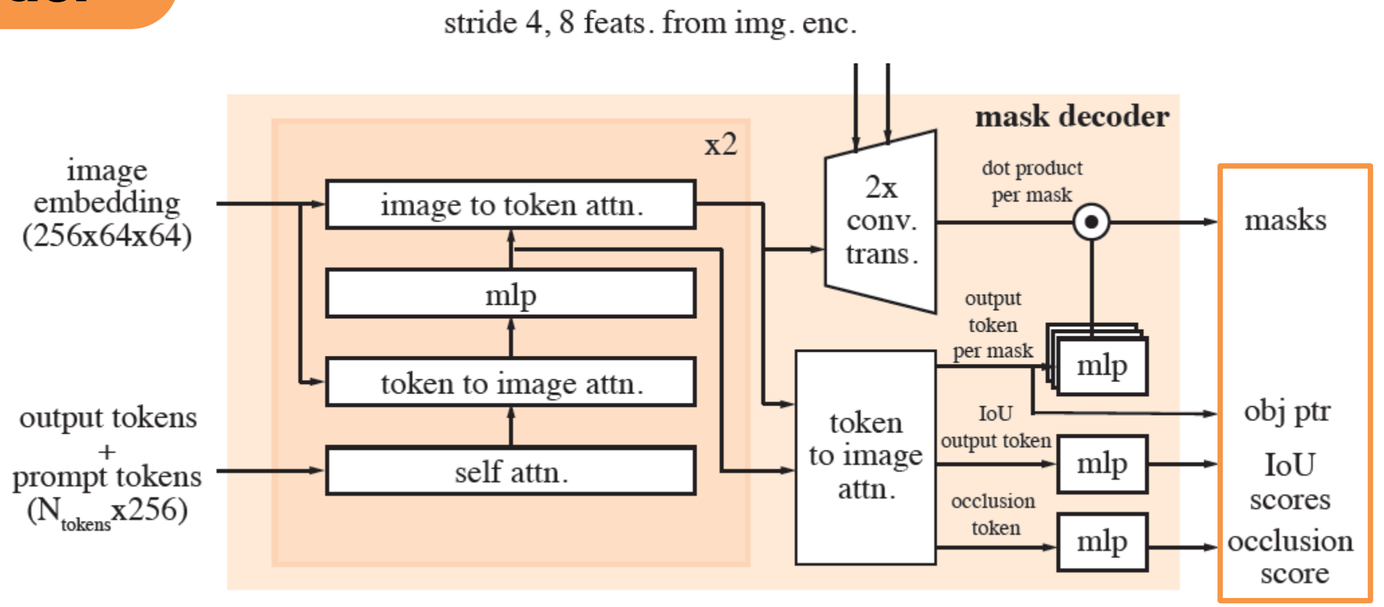
- 역할 : 과거 프레임의 feature와 예측 결과, 그리고 새로운 프롬프트에 따라 현재 프레임의 feature를 조정하여 처리함
- L개의 transformer block을 쌓아 구성

- 첫 번째 블록은 현재 프레임의 이미지 인코딩을 입력으로 받음

- 각 block은 self-attention을 수행한 후, 메모리 뱅크에 저장된 프레임의 메모리와 object pointer에 대한 cross-attention을 수행하고 MLP(multilayer perception)를 거침 (Self-attention과 cross-attention에 바닐라 attention 연산 사용)

Model

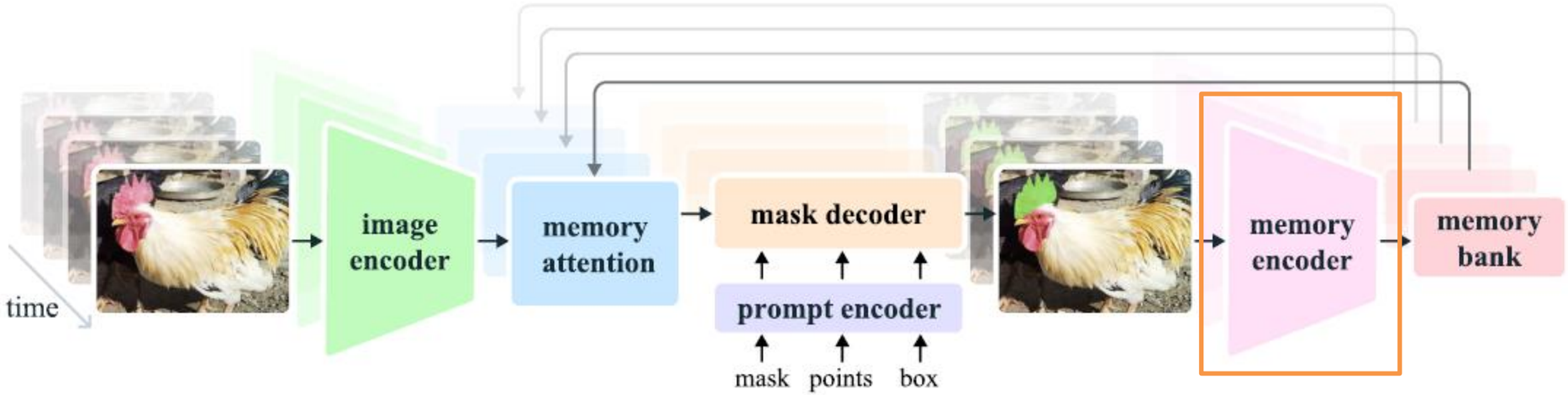
Mask decoder



- Occlusion(가려짐), 객체의 변형, 조명 변화 등을 처리하도록 설계됨
- **Occlusion Token**을 추가하여 객체가 가려지는 상황을 고려하며 이를 통해 모델은 **객체의 가려진 정도를 학습**하고 예측에 반영함
- IoU Scores를 계산하여, 생성된 마스크와 Ground Truth 간의 일치 정도를 평가하며 이 정보를 학습에 반영하여 마스크의 정확성을 향상시킴

Model

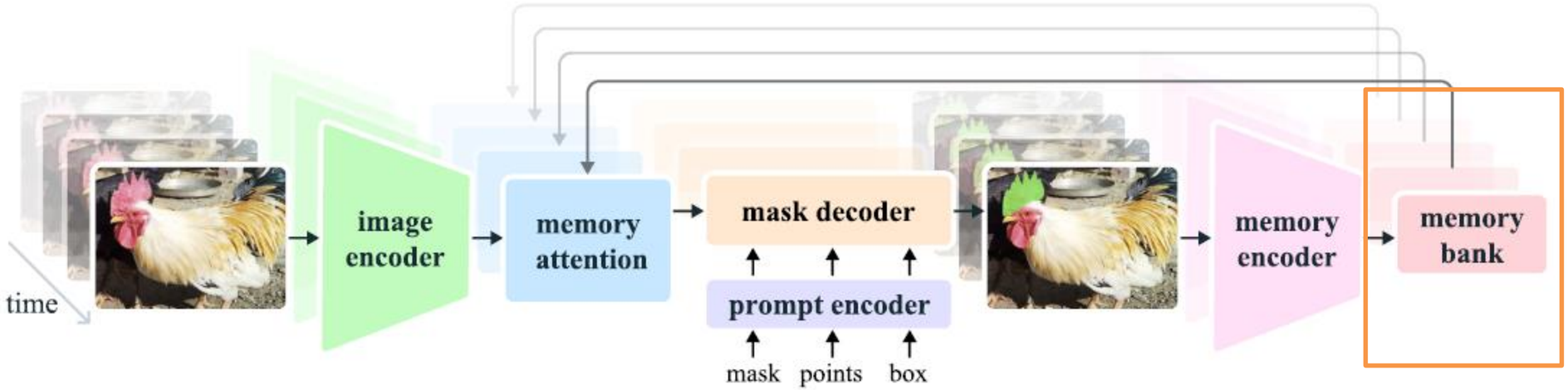
Memory encoder



- 출력 masklet을 Convolutional module을 이용해 downsampling
  - image encoder의 컨디셔닝 되지 않은 frame embedding과 합산
  - 가벼운 convolutional layer들을 통해 정보를 융합하여 메모리 생성

Model

Memory bank



- 비디오에서 target object에 대한 과거 예측 정보 유지
- N개의 최근 프레임의 메모리를 FIFO queue에 유지하고, M개의 프롬프팅된 프레임에 대한 정보를 FIFO queue에 유지 (Spatial feature map)
- Spatial memory외에도, 각 프레임의 mask decoder output token을 기반으로 segment할 object의 high-level semantic 정보에 대한 가벼운 벡터인 object pointer 목록 저장

Results

(a) ML only (b) With Auto

Figure 10 Annotations overlaid on the first frame: (a) manual labels (ML) only, (b) with automatic labels (Auto). Automatic labels increase diversity and coverage.

마스크릿이 오버레이된 SA-V 데이터세트의 예시 비디오(수동 및 자동)

각 마스크릿은 고유한 색상을 가지고 있으며, 각 행은 한 비디오의 프레임을 나타내며 1초 간격임

Part.04

논문 구현

Image

```python
from sam2.sam2_image_predictor import SAM2ImagePredictor  # SAM2 라이브러리

# 모델 로드
sam2_model_id = "facebook/sam2-hiera-large"
device = "cuda" if torch.cuda.is_available() else "cpu"
sam2 = SAM2ImagePredictor.from_pretrained(sam2_model_id)

# 이미지 경로 설정
image_path = r"C:\ai5\본프로젝트\김호정.jpg"
image = Image.open(image_path).convert("RGB")
image_np = np.array(image)


# 이미지 로드 및 전처리
image = Image.open(image_path).convert("RGB")
image_np = np.array(image)

input_points = np.array([[977, 1633], [865, 2697]])  # 사용자가 선택한 점
input_labels = np.array([1, 1])  # 1: 포그라운드, 0: 백그라운드
```

Image

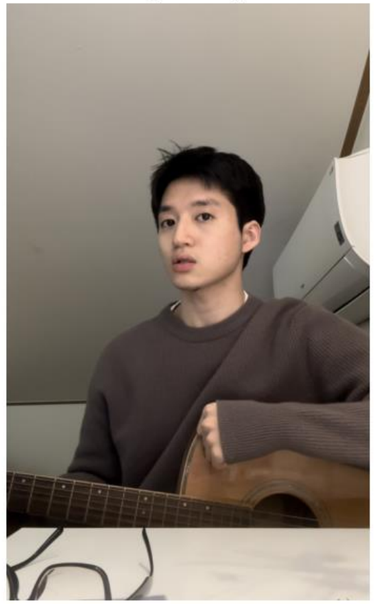
```python
# 세그멘테이션 수행
with torch.inference_mode(), torch.autocast("cuda", dtype=torch.bfloat16):
    sam2.set_image(image_np)
    masks, _, _ = sam2.predict(point_coords=input_points, point_labels=input_labels)

# 마스크 병합
if len(masks) > 0:
    all_masks = np.any(masks > 0, axis=0).astype(np.uint8)  # 마스크 병합
else:
    raise ValueError("No masks were generated.")

# 시각화
plt.figure(figsize=(10, 10))
plt.subplot(1, 2, 1)
plt.imshow(image_np)
plt.title("Original Image")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(image_np)
plt.imshow(all_masks, alpha=0.5, cmap="cool")  # 색상 맵 개선
plt.title("Segmented Image with SAM2")
plt.axis("off")

plt.show()
```

Image

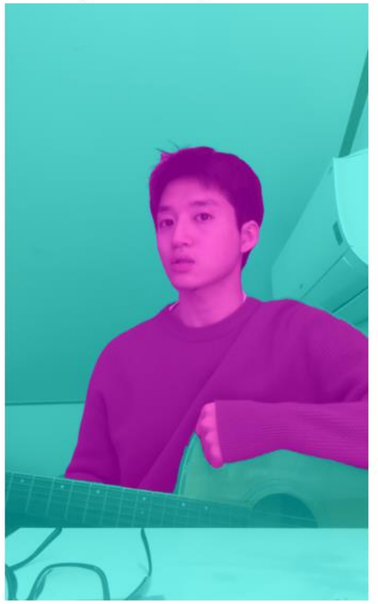
Original Image

Segmented Image with SAM2

Original Image



Segmented Image with SAM2



Image

Original Image

Segmented Image with SAM2

Original Image

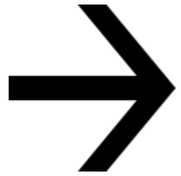
Segmented Image with SAM2

Video

```python
import supervision as sv
import numpy as np
import cv2
from sam2.build_sam import build_sam2_video_predictor

# 경로 설정
PATH = 'C:/ai5/본프로젝트/논문/SAM/'
VIDEO_PATH = 'C:/ai5/본프로젝트/논문/SAM/토리.mp4'
SEGMENTED_VIDEO_PATH = PATH + 'video/토리.mp4'

CHECKPOINT = "C:/Users/hyeonah/sam2/checkpoints/sam2.1_hiera_large.pt"
CONFIG = "C:/Users/hyeonah/sam2/sam2/configs/sam2.1/sam2.1_hiera_l.yaml"

# SAM2 모델 로드
sam2_model = build_sam2_video_predictor(CONFIG, CHECKPOINT)

# 비디오 세그멘테이션
def process_video_with_sam2(video_path, output_video_path, sam2_model):
    cap = cv2.VideoCapture(video_path)
    video_info = sv.VideoInfo.from_video_path(video_path)

    # SAM2 모델 초기화
    inference_state = sam2_model.init_state(video_path)
    sam2_model.reset_state(inference_state)
```

Video

```
colors = ['#FF1493', '#00BFFF', '#FF6347', '#FFD700']
mask_annotator = sv.MaskAnnotator(
    color=sv.ColorPalette.from_hex(colors),
    color_lookup=sv.ColorLookup.TRACK
)

# 초기 포인트 설정 (첫 번째 프레임)
ret, initial_frame = cap.read()
if not ret:
    print("비디오를 읽을 수 없습니다.")
    return

# 초기 객체 포인트와 레이블 설정
# 예제: 두 개의 포인트를 지정
initial_points = np.array([[357, 682], [475, 500]], dtype=np.float32)
labels = np.array([1, 1])  # 객체의 클래스 라벨 (1은 포그라운드)

frame_idx = 0
tracker_id = 1

_, object_ids, mask_logits = sam2_model.add_new_points(
    inference_state=inference_state,
    frame_idx=frame_idx,
    obj_id=tracker_id,
    points=initial_points,
    labels=labels,
)
```



Video

```
    # 세그멘테이션 처리
    with sv.VideoSink(output_video_path, video_info=video_info) as sink:
        for frame_idx, object_ids, mask_logits in sam2_model.propagate_in_video(inference_state):
            ret, frame = cap.read()
            if not ret:
                break

            # 마스크 처리
            masks = (mask_logits > 0.0).cpu().numpy()
            N, X, H, W = masks.shape
            masks = masks.reshape(N * X, H, W)

            detections = sv.Detections(
                xyxy=sv.mask_to_xyxy(masks=masks),
                mask=masks,
                tracker_id=np.array(object_ids)
            )

            # 마스크를 프레임에 적용
            frame = mask_annotator.annotate(frame, detections)
            sink.write_frame(frame)

    cap.release()
    print(f"세그멘테이션된 비디오가 {output_video_path}에 저장되었습니다.")

# 실행
process_video_with_sam2(VIDEO_PATH, SEGMENTED_VIDEO_PATH, sam2_model)
```

Video

Video

THANK YOU
Q&A