Segment Anything Model

Alexander Kirillov / 5 Apr 2023

논문 구현

CloseAI팀 이상헌 김유철 이정훈 박준혁

목차

- overview
- model
- 구현결과
- QnA

Overview

Overview

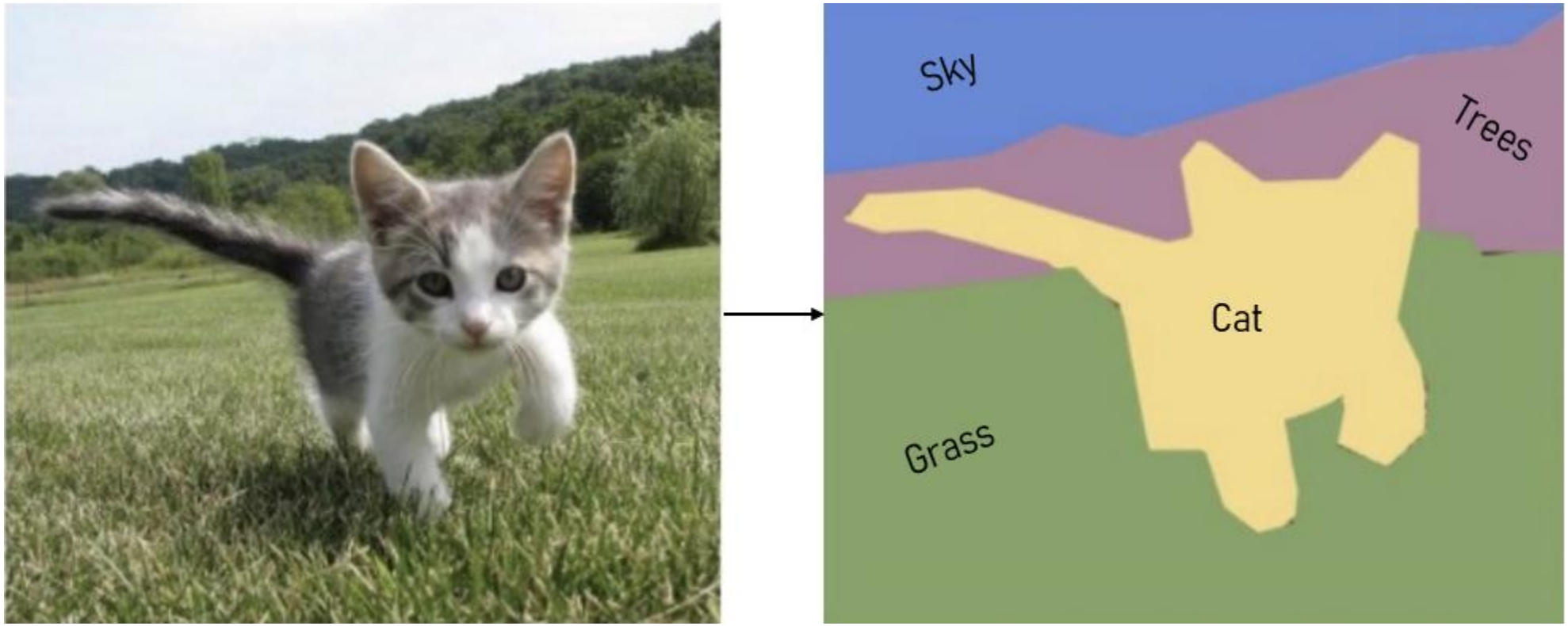
- GPT가 NLP부문에서 뛰어난 성능을 보임

- Computer Vision에서도 뛰어난 모델을 만들고 싶어
  새로운 **task**, **model**, **data**를 개발

- Segmentaion은 물론 다른 태스크들에서도 성능을 높이고 싶었다

Image Segmentation

이미지 안에서 물체의 영역을 마스킹하는 컴퓨터 비전 테스크
픽셀 단위로 어느 클래스에 속하는지를 분류한다



SAM의 3가지 핵심 요소

- **Task**: 어떤 task로 학습시켜야 GPT처럼 general한 모델을 만들 수 있을까?

-> **promptable segmentation**

- **Data**: 이 모델을 학습시키려면 어떤 데이터들이 필요할까?

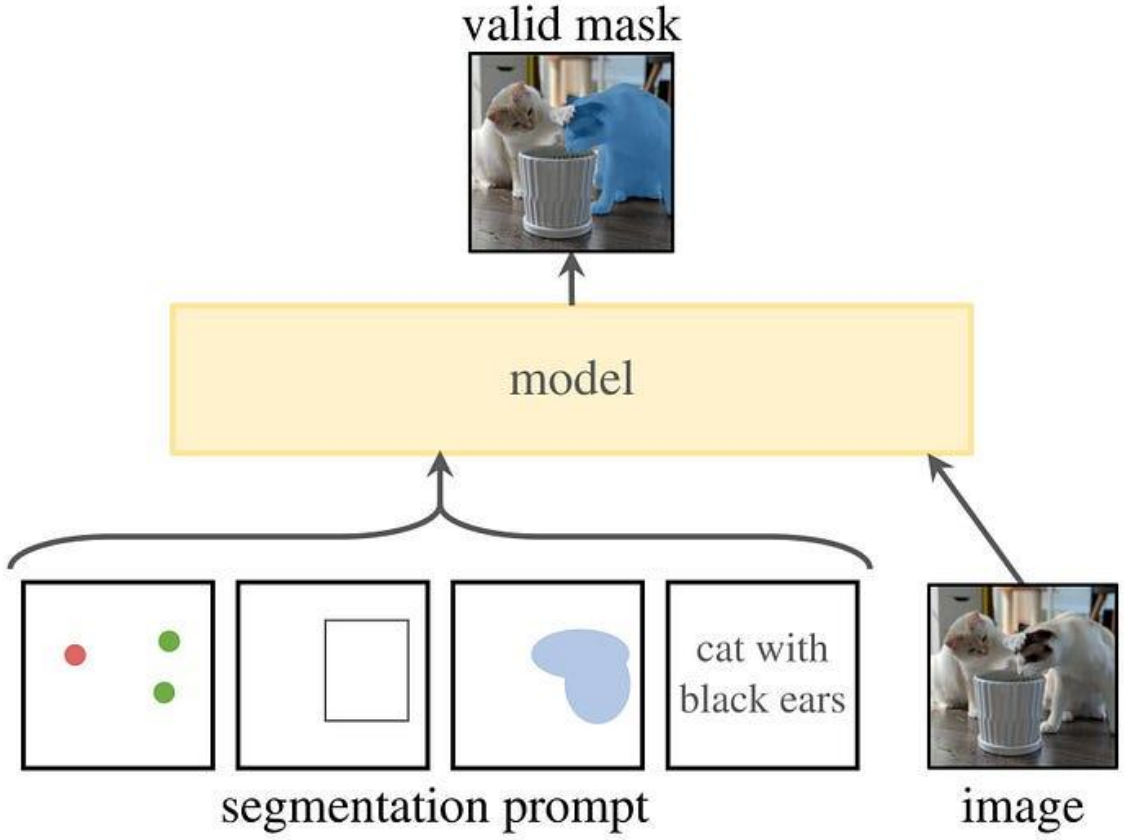
-> **SA-1B**(데이터셋): 1100만장 이미지에 대한 10억개의 마스크

- **Model**: 이 task를 잘 수행하면서도 general하려면 어떤 구조여야 할까?

-> **Image encoder, prompt encoder, mask decoder** 활용 구조

Task

- **promptable segmentation**
  마스크를 생성하고자 하는 대상을 유연하게 prompt로 지정할 수 있는 task



- Zero-shot transfer
- Related tasks
- Pre-training
- Ambigous 해결

Data

- SA-1B(데이터셋)
- Data engine 이용

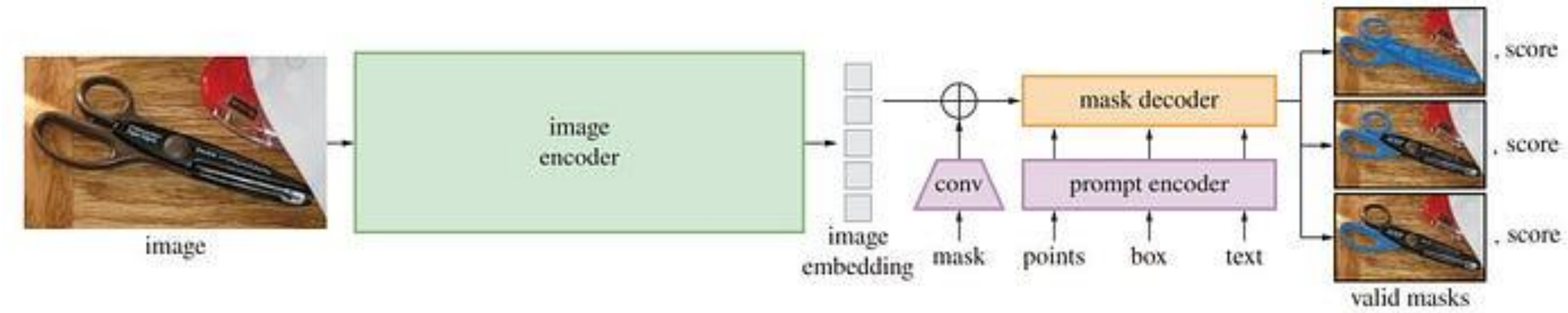
1. Assisted-manual stage

2. Semi-automatic stage

3.Fully automatic stage

-> 데이터셋의 모든 1,100만 개 이미지에 완전 자동 마스크 생성을 적용하여 총 11억 개의 고품질 마스크를 생성

Model



- Image Encoder
- Prompt Encoder
- Mask Decoder

Model(Image Encoder)

Image Encoder: MAE(Masked Auto-Encoder) 방식의 pre-trained ViT를 사용
MAE : 일정한 크기의 그리드로 나누고 랜덤하게 가린 뒤, 복원하도록 학습

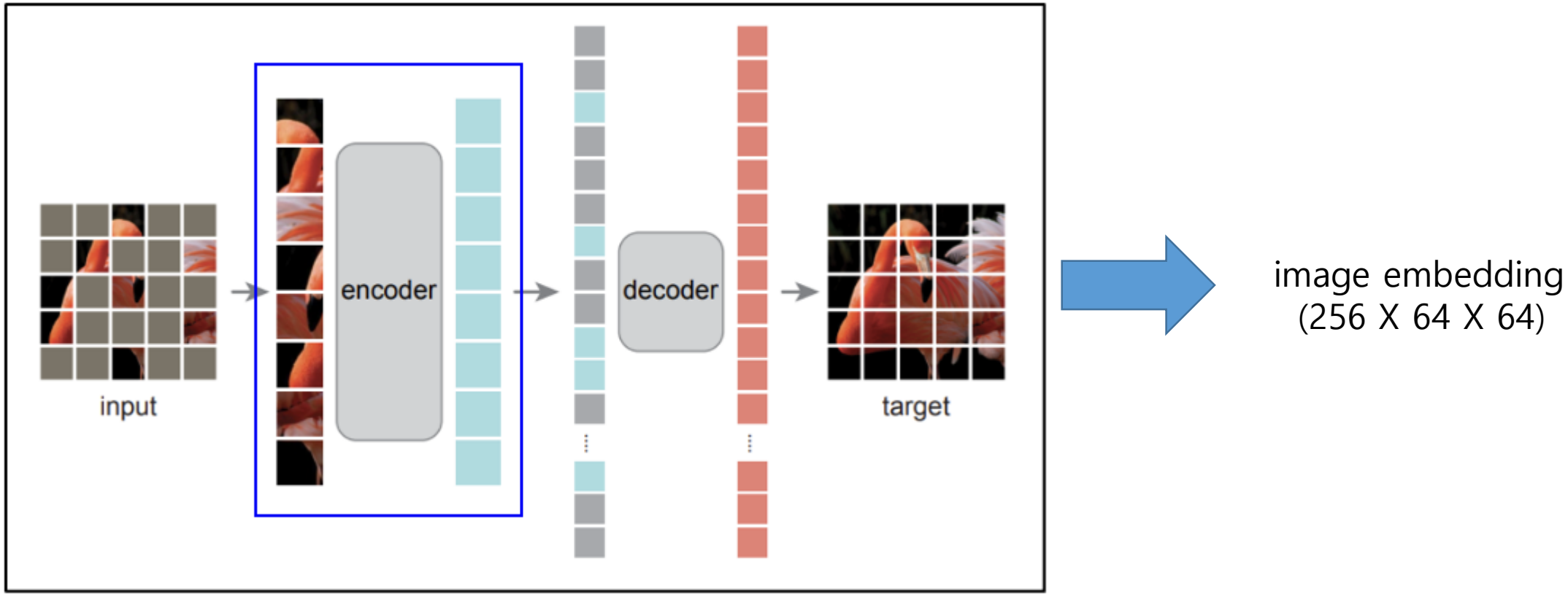
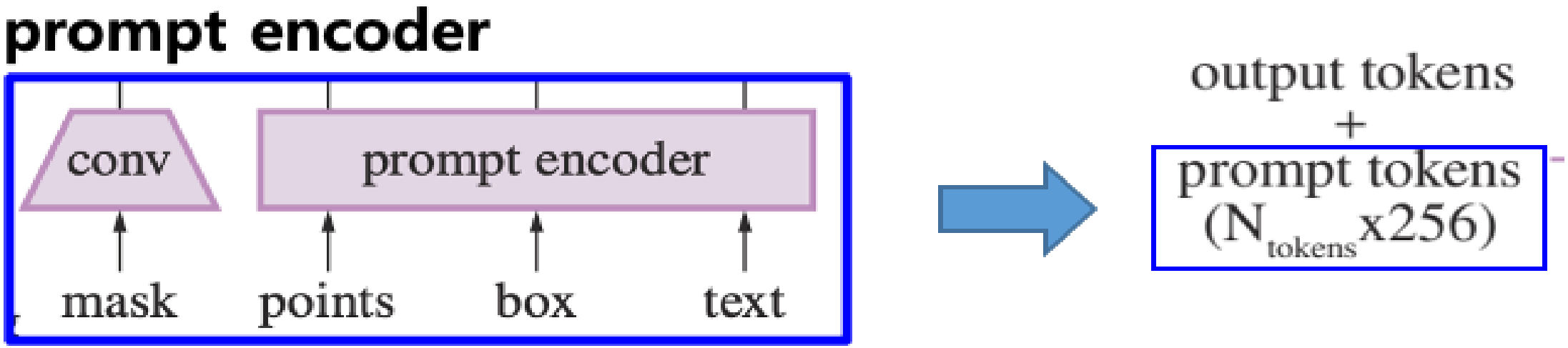


image embedding
(256 X 64 X 64)

Model(Prompt Encoder)

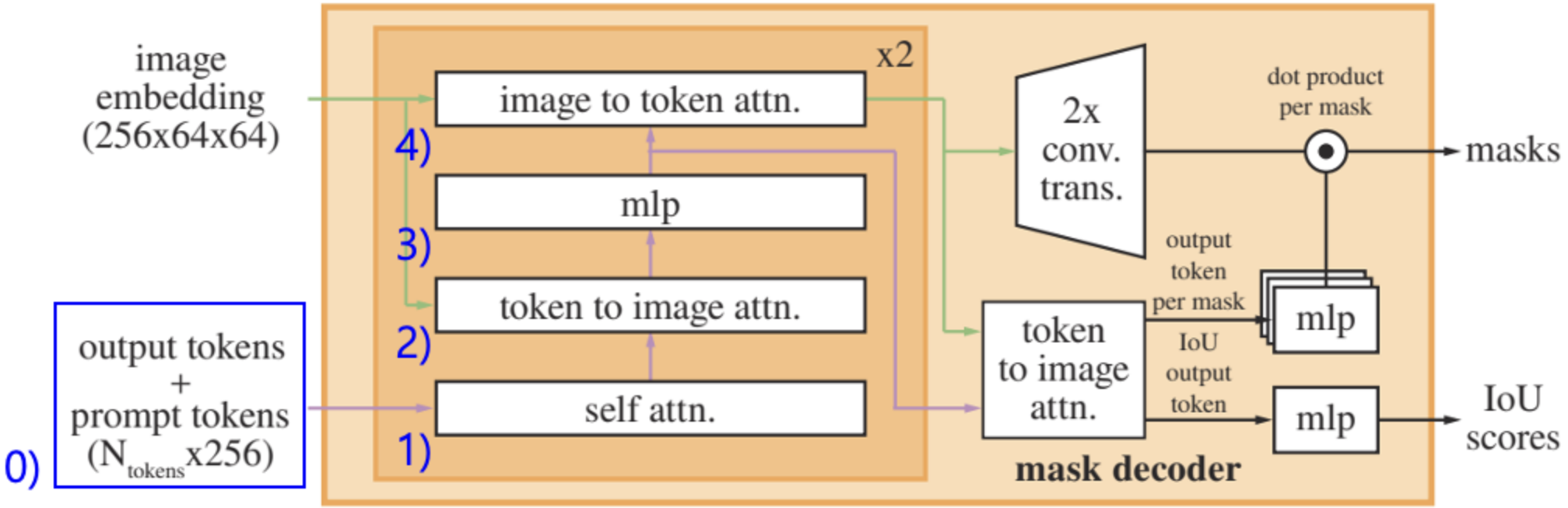
Prompt Encoder: 두 가지의 prompt를 고려한다
Sparse는 (점을 찍어서 명령하는 points, 박스를 그려 명령하는 boxes, 단어로 명령하는 text)
Dense는 (직접 마스크를 제공하는 masks)

- points, boxes는 positional encoding(해당 점의 위치 + 피사체와 배경을 구별하게 학습된 임베딩)를 추출
- text는 off-the-shelf text encoder로 추출
- masks는convolution 레이어를 통과시켜 16배 작게 만들고 이미지 임베딩과 element-wise로 더함



Model(mask decoder)

Mask Decoder : 이미지 임베딩, 프롬프트 임베딩, 출력 토큰을 마스크에 효율적으로 매핑한다.

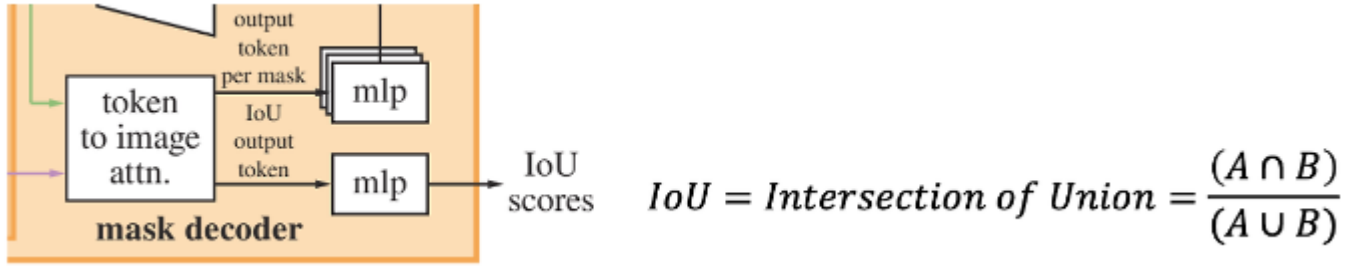


Dynamic mask prediction head를 따르는 Transformer decoder block을 변형

decoder block은 모든 embeddings을 업데이트하기 위해 prompt self-attention과 cross-attention을 두 방향(prompt-to-image embedding과 image-to-prompt embedding)으로 사용

Ambiguity-aware

입력 prompt가 모호한 경우에도 여러 답을 고려하여 최적의 결과를 제공



$$IoU = Intersection\ of\ Union = \frac{(A \cap B)}{(A \cup B)}$$

Whole Mask Loss
- 전체 객체를 포함하는 마스크에 대해 계산된 손실

Part Mask Loss
- 객체의 일부에 대한 마스크에 대해 계산된 손실

Subpart mask Loss
- 객체의 더 작은 부분이나 세부 구조에 대한 마스크에 대해 계산된 손실

Loss

Focal Loss
- 불균형 클래스 문제를 해결하고, 어려운 예제에 더 큰 가중치를 부여

$$\text{Focal Loss} = -\alpha_t(1 - p_t)^\gamma \log(p_t)$$

Dice Loss
- 분할된 객체와 실제 객체 간의 유사성을 최대화하며, 주로 작은 객체에 대해 더 나은 성능을 발휘

$$\text{Dice Loss} = 1 - \frac{2|X \cap Y|}{|X| + |Y|}$$

핵심 코드

```python
class MaskDecoder(nn.Module):
    def __init__(self, embed_dim, num_heads, mlp_dim, num_layers, img_size, num_classes):
        super(MaskDecoder, self).__init__()
        self.embed_dim = embed_dim
        self.img_size = img_size

        # Transformer 디코더 레이어를 쌓음
        self.layers = nn.ModuleList([
            nn.TransformerDecoderLayer(d_model=embed_dim, nhead=num_heads, dim_feedforward=mlp_dim)
            for _ in range(num_layers)
        ])
        self.decoder_norm = nn.LayerNorm(embed_dim)

        # 마스크를 예측하는 최종 컨볼루션 레이어
        self.mask_conv = nn.Conv2d(embed_dim, num_classes, kernel_size=1)

    def forward(self, x, memory):
        # x: Transformer 인코더의 출력
        # memory: 이전 디코더 출력 또는 인코더의 출력 (skip connection 용도)

        for layer in self.layers:
            x = layer(x, memory)

        x = self.decoder_norm(x)

        # Transformer 출력의 형태를 이미지 형태로 변환
        x = x.permute(1, 2, 0).view(-1, self.embed_dim, int(self.img_size**0.5), int(self.img_size**0.5))

        # 마스크 예측
        x = self.mask_conv(x)

        # 최종 출력 크기 조정
        x = F.interpolate(x, size=(self.img_size, self.img_size), mode='bilinear', align_corners=False)

        return x
```

구현 결과

Q&A