

My Re-implemented SAM by reading the paper Segment Anything [ICCV, 2023]

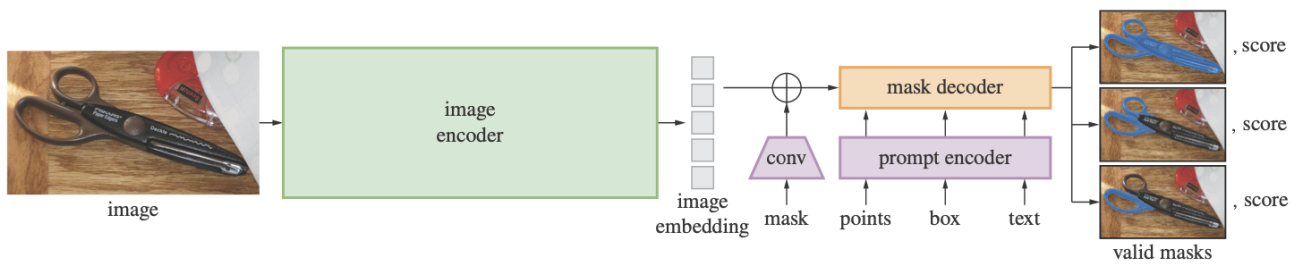


Figure 4: Segment Anything Model (SAM) overview. A heavyweight image encoder outputs an image embedding that can then be efficiently queried by a variety of input prompts to produce object masks at amortized real-time speed. For ambiguous prompts corresponding to more than one object, SAM can output multiple valid masks and associated confidence scores.

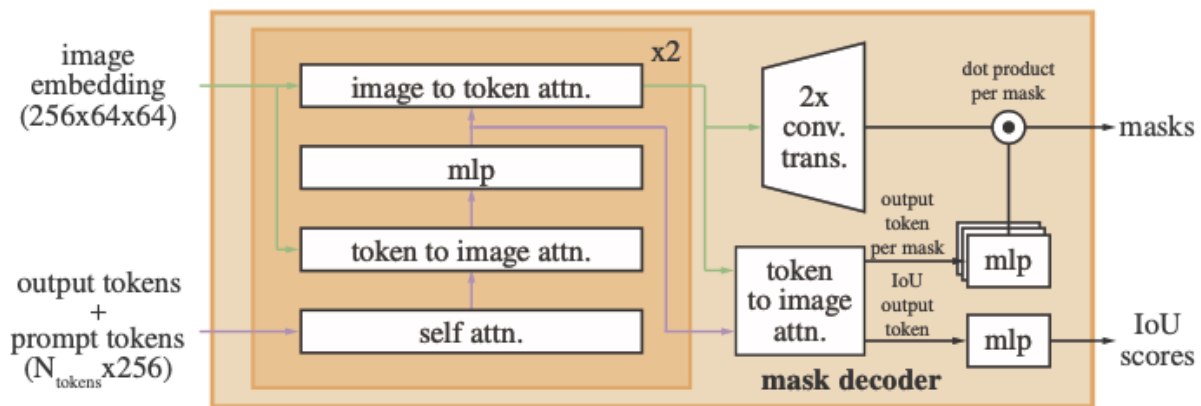


Figure 14: Details of the lightweight mask decoder. A two-layer decoder updates both the image embedding and prompt tokens via cross-attention. Then the image embedding is upsampled, from which the updated output tokens are used to dynamically predict masks. (Not illustrated for figure clarity: At every attention layer, positional encodings are added to the image embedding, and the entire original prompt token (including position encoding) is re-added to the token queries and keys.)

For re-implementation, I referred not only the content but also appendix. The implementation part was mainly the overall architecture shown in figure 4, and the detailed model

This work was done for the assignment below.

Assignments

Only full or long papers
in the main conf.

Choice 1 (CS-related students)

- Read one paper presented in 2020-2024 at TOP journals/conferences
Journal: TPAMI/TMM/TIP/TCSVT/TOG/TOMM
Conf: CVPR/ICCV/ECCV/SIGGRAPH/ACMMM/NeurIPS/ICML/ICLR
- Implement them **by yourself** and submit source code to github (make it open).
Authors' original source code must not be used.
- Add an explanation explaining
 - Why this paper is important (what the technical core is, why the paper is accepted)
 - What you have implemented
- If non-top j/c paper is selected, you will NOT get a credit.

Choice 2 (for non-CS-related students)

- Read two papers (same restriction as the above)
- Make an open blog page explaining
 - Why this paper is important (what the technical core is, why the paper is accepted)

Instructions

- Submit a report or URL to UOTS by **Aug. 1st 11:59pm, JST**
- You can use any libraries or project codes as long as it is properly cited.
- Implementation for your own research project cannot be accepted.
- Your blogs might be listed in my lecture page.

9

ヒント architecture 初期化 (**init** メソッド):

image_encoder, prompt_encoder, mask_decoder の3つの主要なコンポーネントを初期化します。入力画像の正規化に使用する平均 (pixel_mean) と標準偏差 (pixel_std) をバッファに登録します。順伝播 (forward メソッド):

バッチ入力を受け取り、画像を前処理してエンコードします。エンコードされた画像からポイントやボックス、マスクなどのプロンプトをエンコードします。マスクデコーダーを使って低解像度マスクとIoU予測を生成します。マスクを元の画像サイズに後処理し、結果をリストに格納して返します。マスクの後処理 (postprocess_masks メソッド):

低解像度のマスクを元の画像サイズにリサイズし、パディングを除去します。前処理 (preprocess メソッド):

画像を正規化し、必要に応じてパディングを追加して正方形の入力に変換します。