

1. Le modèle SAM

SAM 1, ou Segment Anything Model, est un modèle d'intelligence artificielle développé par Meta (Facebook) en 2023. Son but est de segmenter n'importe quel objet dans une image, sans avoir besoin de l'entraîner à l'avance sur une catégorie spécifique (comme "chat", "voiture", etc.).

- Il fonctionne en mode "zero-shot", c'est-à-dire qu'il peut segmenter des objets jamais vus durant son apprentissage.
- Il est promptable, ce qui signifie que l'utilisateur peut lui indiquer où segmenter (en cliquant ou en dessinant une boîte), et il génère automatiquement un masque précis de la zone.
- Il a été entraîné sur un gigantesque ensemble de données d'images, appelé SA-1B, avec plus d'un milliard de masques.

En pratique, on clique sur une zone d'une image (ex. : un motif d'une fresque), et SAM 1 en déduit la forme complète à segmenter. Il a été entraîné pour être généraliste, c'est-à-dire qu'il n'est pas limité à une classe d'objet (comme "chien" ou "voiture") mais peut segmenter tout ce qui est visuellement cohérent.

La segmentation

La segmentation est le processus de division d'une image en de plus petites régions, où chaque région correspond à un objet spécifique ou un arrière-plan dans l'image. Le but principal de SAM est de devenir un "foundation model" pour la segmentation d'image c'est-à-dire un modèle qui a été entraîné sur une très grande quantité de données générales et qui peut être adapté à un large éventail de tâches spécifiques. Les "foundation modes" actuels sont par exemple GPT 4, Bard, CLIP ou encore BERT.

Compréhension du fonctionnement du modèle

Pour créer cela META a créé un "Data Engine" appelé "model-in-the-loop annotation strategy" et il collecta des données d'entraînement en 3 phases :

- Phase manuellement assisté : META a engagé des "annotateur", des gens qui vont étiqueté des images avec des masques de segmentations, dans cette phase les annotateur on étiqueté tous les masques qui pouvaient voir par ordre d'importance, ils ont collecté approximativement 4.3M de masque provenant de 120k images

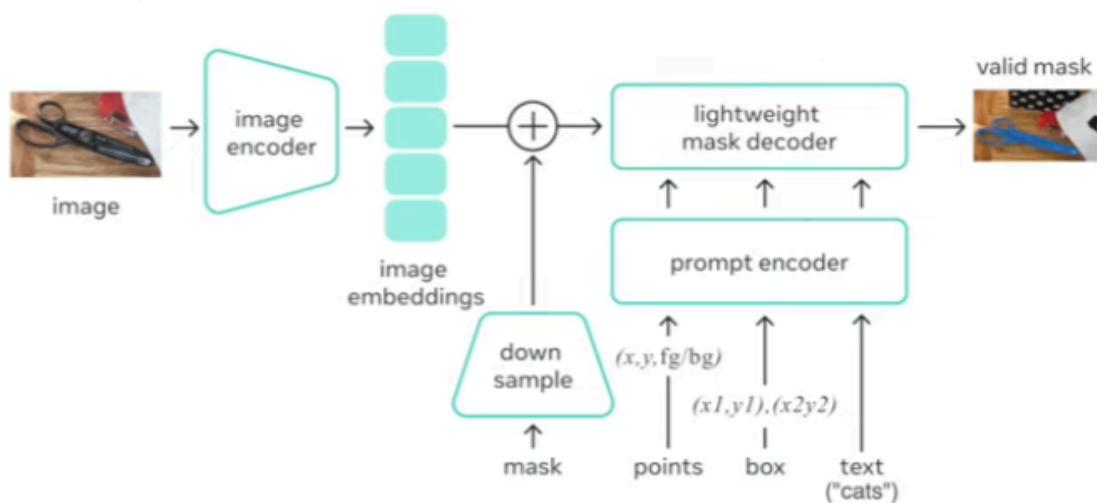
- Phase semi-automatique : le but était d'améliorer la diversité du dataset, cette fois ci, les annotateurs devaient ajouter des masques aux images en plus des masques générés automatiquement, cela a grandement augmenté le nombre de masques moyen par image passant de 44 à 72.
- Phase automatique : cette fois on a simplement appliqué au modèle SAM une grille de 32x32 points sur l'image complète et lui ont demandé d'ajouter par lui seul les masques, ils l'ont appliqué à 11M d'images haute résolution et ont produit plus de 1.1B de masque haute qualité.

Le résultat de cette phase est un dataset impressionnant appelé "SA-1B" de plus de 1B d'image, c'est 6x que l'ancien plus gros dataset d'image "OpenImage V5" et 400x plus de masques.

La première étape du modèle est l'encodage de l'image puis l'"embedding" c'est-à-dire la vectorisation de celle-ci. C'est la partie qui prend le plus de temps au modèle, cependant après celle-ci, la segmentation devient quasi instantanée. La deuxième étape est le "prompt encoder" ou encodeur d'instruction, celui-ci peut prendre des points, des boîtes (ensemble de points), ou encore du texte ou un masque et en ressort un "prompt embedding" c'est à dire la mise en forme sous de vecteur de notre instruction.

La dernière étape est la fusion de ces parties, c'est- à-dire la fusion entre nos deux vecteurs puis ces partie sont insérée dans le "lightweight mask decoder" qui s'occupe de cette fusion pour produire notre masque de segmentation : image embedding + prompt embedding -> lightweight mask decoder = output masked image. Ceci est la traduction du diagramme ci-dessous

Segment Anything Model



Pour éviter les ambiguïtés sur les masques produits, SAM peut produire plusieurs masques et ajoutés à chacun d'entre eux un score de confiance, si l'utilisateur clique de nouveau sur la même partie.

Comment SAM délimite-t-il techniquement les zones ?

Une fois que l'image est encodée par un backbone de type Vision Transformer (ViT), chaque région de l'image est représentée par un vecteur dans un espace de grande dimension (souvent 256 à 1024 dimensions). De l'autre côté, le prompt (click, boîte ou masque) est lui aussi transformé en vecteur (ou prompt embedding). Le lightweight mask decoder prend alors ces deux représentations, visuelles et interactives, et les fusionne en utilisant un "mécanisme d'attention". Ce mécanisme permet au modèle de pondérer l'importance de chaque région de l'image en fonction de sa similarité vectorielle avec le prompt. Autrement dit, SAM compare mathématiquement les vecteurs (par exemple via des produits scalaires ou distances cosinus) pour identifier quelles zones de l'image sont les plus proches, dans l'espace des caractéristiques, du prompt fourni. Cela revient à chercher, dans un espace de représentation appris, les zones les plus "similaires" au point ou à la forme donnée. Grâce à son entraînement massif, SAM apprend à associer ces similarités à des frontières visuellement plausibles, sans avoir besoin de connaître la nature exacte de l'objet.

2. Présentation du modèle SAM 2

Dans le cadre de mon projet de restauration de fresques par traitement d'image, j'ai découvert le modèle Segment Anything Model 2 (SAM 2), développé par Meta. Ce modèle représente une évolution du modèle SAM initial, en élargissant ses capacités de segmentation au-delà des images fixes pour inclure les vidéos. SAM 2 repose sur une architecture de type transformer, enrichie d'un mécanisme de mémoire permettant au modèle de conserver en contexte les objets segmentés dans les images précédentes.

Cela s'avère particulièrement utile dans mon cas, où une même fresque peut être photographiée sous différents angles, ou partiellement abîmée, rendant la segmentation moins triviale d'une image à l'autre.

L'interaction avec SAM se fait à travers des "prompts", un terme technique qui désigne simplement des actions de l'utilisateur pour indiquer la zone à segmenter. Dans notre cas d'usage, cela se traduit par un simple clic sur la zone d'intérêt dans l'image. Le modèle utilise ce clic comme point de départ pour générer un masque de segmentation, qu'il peut ensuite propager automatiquement aux autres images ou aux images suivantes dans une séquence, en tenant compte des précédentes corrections.

Ainsi, il est possible d'affiner la segmentation si nécessaire en ajoutant un second clic, sans devoir recommencer tout le processus. Ce système interactif permet de gagner du temps, tout en garantissant une précision de découpe fine, très pratique pour restaurer des détails complexes comme les motifs d'une fresque ou les contours d'un personnage. Ce système fonctionne aussi bien avec SAM (déjà implémenté) que avec SAM2, il n'y a pas de changement à ce niveau.

3. Avantages pour la restauration d'images patrimoniales

Tout d'abord, le modèle est très rapide : il est capable de traiter une image avec une vitesse environ six fois supérieure à celle du modèle précédent SAM (1), ce qui en fait déjà un atout majeur. Ensuite, sa capacité à proposer des résultats de qualité avec peu d'interactions (souvent un seul clic) simplifie l'annotation manuelle.

SAM 2 a été entraîné sur un jeu de données très vaste et varié, incluant non seulement des objets classiques mais aussi des parties d'objets, comme un doigt, un bijou, un morceau de tissu. Cette capacité est très intéressante dans le domaine de la restauration car le travail se fait souvent sur des zones fragmentaires fortement dégradées.

Le modèle ne dépend pas de catégories prédéfinies : il peut segmenter n'importe quelle forme ayant une frontière visible, ce qui le rend particulièrement adapté à la diversité iconographique des fresques historiques.

Ce gain de temps et de qualité est lié à l'introduction d'une mémoire temporelle, absente dans SAM 1, qui permet à SAM 2 de conserver en contexte les objets déjà identifiés et d'adapter les segments sur les images suivantes. Dans le cas de fresques partiellement dégradées, cette mémoire pourrait permettre une meilleure continuité et stabilité dans la segmentation.

Le papier note que SAM est de manière assez surprenante performant dans la reconnaissance des bordures dans l'image. Bien que pas aussi performant que des modèles entraînés spécialement pour cela, il est quand même capable d'approcher leurs scores d'assez proche.

4. Utilisation du modèle

Cette section va expliquer comment j'ai pu tester et utiliser SAM 2.

Basé sur ce projet : <https://github.com/facebookresearch/sam2>

Premièrement suivre l'installation du github (attention utiliser sudo sinon ça ne marche pas) et ensuite se diriger vers le notebook qui nous intéresse (image_predictor_example.ipynb pour notre cas). Placer les images qui nous

intéressent dans le dossier image, déplacer le fichier notebook à la racine et remplacer le path des images dans le notebook. Il n'y a plus qu'à run toutes les cellules et cela devrait fonctionner.

On peut ensuite simplement suivre les exemples d'utilisations et changer les coordonnées pour les adapter à notre image.

Deux choses sont importantes :

Le multimask_output est activé par défaut. Cette option permet de générer 3 masques et le modèle attribue un score à chacun d'entre eux.

Le score est l'estimation de la qualité du masque selon le modèle. Cela arrive que le modèle estime un score surévalué dans les images ci-dessous.

Mask 1, Score: 0.924



Mask 2, Score: 0.847



En sélectionnant le pied, on voit bien que le 2ème masque est mieux segmenté que le 1er alors qu'il y a 8 points de score de moins que le premier. Donc, activer le *multimask_output* peut rester intéressant, par défaut, plus loins dans le notebook, un seul masque sera choisi avec : *mask_input = logits[np.argmax(scores), :, :] .* Cette ligne va permettre de ne récupérer que le masque avec le score le plus élevé, il peut être intéressant de l'enlever.

En sélectionnant la même zone avec SAM 1 actuellement sur l'application graphique on observe ce léger défaut au niveau du pied aussi :

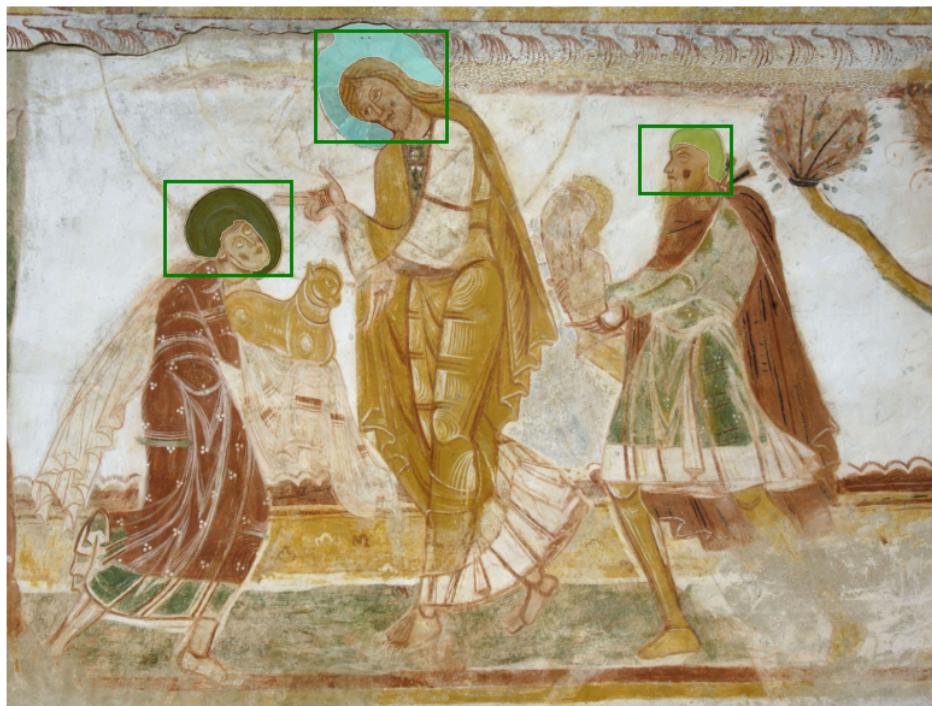


Quelques résultats

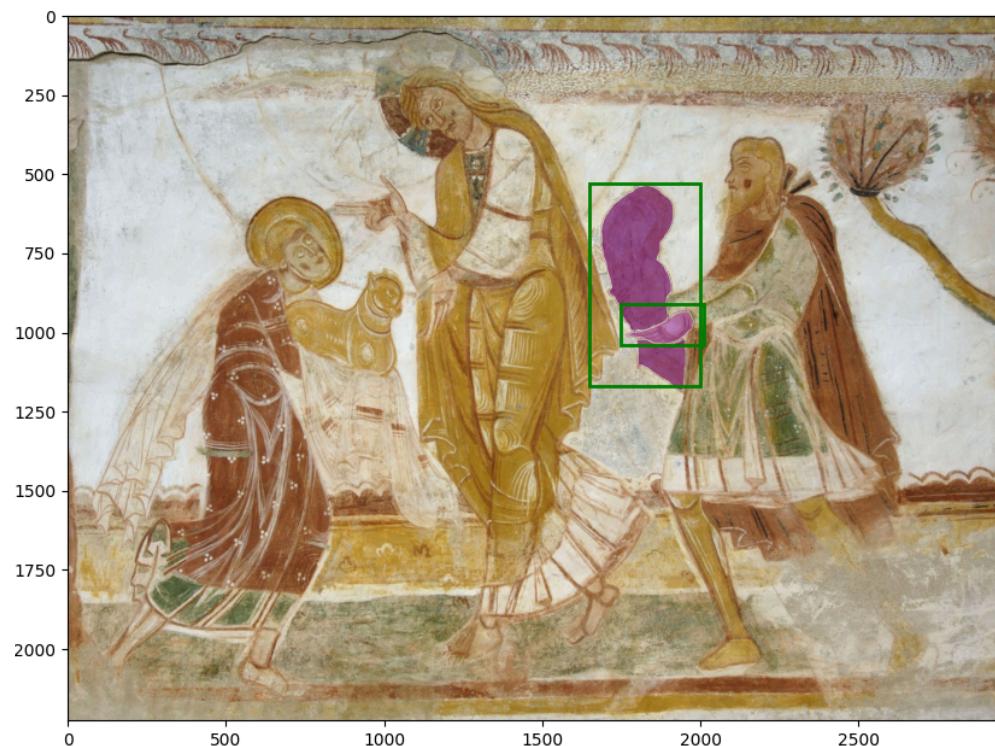
En plus des résultats au-dessus où l'on sélectionne des points sur l'image, on peut aussi sélectionner via des box, les résultats sont aussi vraiment satisfaisant.



On peut aussi en faire plusieurs sur l'image.

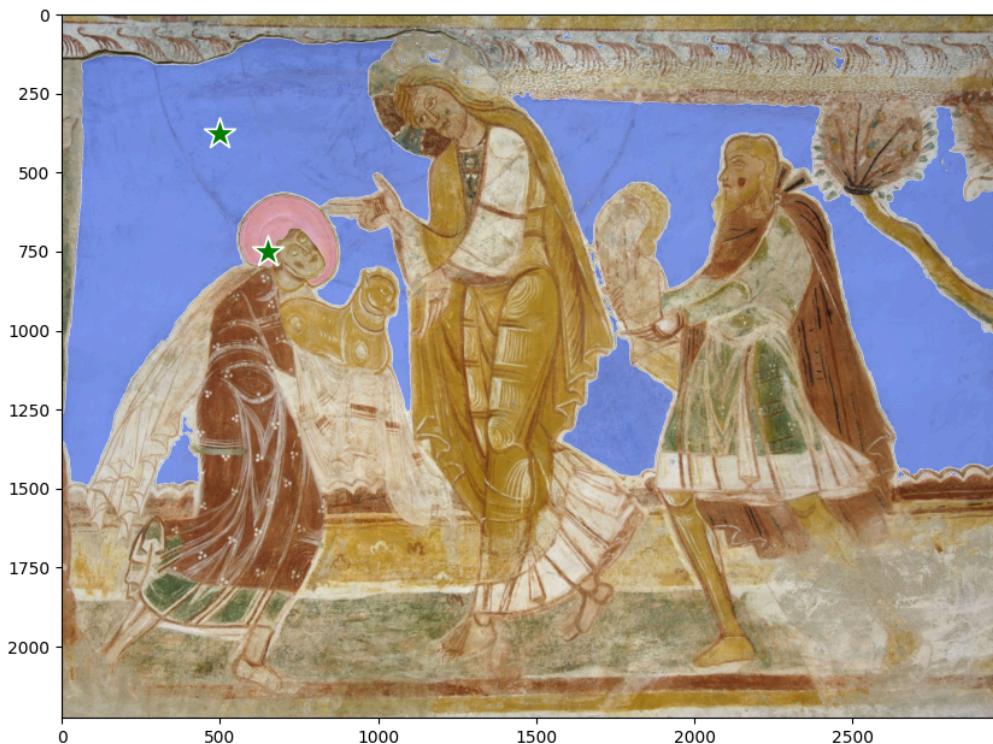


On peut également les superposer, ce qui fonctionne aussi surprenamment bien.



La main est clairement segmentée par-dessus l'autre élément.

De manière similaire est possible de le faire avec des points sur l'image.



Il est aussi possible de faire du batching, c'est-à-dire de le faire pour plusieurs images différentes, il suffit de préparer les prompts à l'avance. On peut donc remarquer que même si l'accent de SAM 2 est apporté sur la vidéo, ses gains performance sur de l'image par rapport à SAM 1 restent assez impressionnante.

TERMES TECHNIQUES

Backbone : c'est la “colonne vertébrale” du modèle, c'est-à-dire la partie principale qui extrait les informations utiles de l'image. Dans les modèles de vision par ordinateur, le backbone est un réseau de neurones profond qui transforme l'image brute en un ensemble de vecteurs représentant ses caractéristiques visuelles (couleur, texture, forme, etc.). C'est l'étape d'extraction de caractéristiques.

Vision Transformer (ViT) : c'est un type de backbone qui repose sur une architecture de transformer, à l'origine conçue pour le traitement du langage naturel (comme GPT). Dans le cas de ViT, l'image est découpée en petits blocs (appelés patches, par exemple de 16x16 pixels), puis chaque patch est converti en vecteur. Le transformer va ensuite traiter tous ces vecteurs en parallèle, en utilisant un mécanisme d'attention pour détecter quelles parties de l'image sont importantes ou liées entre elles. Ce traitement permet d'encoder l'image entière tout en capturant des relations complexes entre les différentes régions de l'image.

Embedding (ou vectorisation) : c'est le processus qui transforme une donnée (comme un pixel, une région, ou un prompt) en un vecteur de nombres (par exemple : [0.52, -1.31, 0.09, ...]). Ces vecteurs permettent au modèle de faire des calculs, comme mesurer des similarités, effectuer des regroupements, ou produire des décisions. On parle d'image embedding pour la version vectorisée de l'image, et de prompt embedding pour celle du point cliqué ou de la boîte dessinée.*