Sparse Matching Project Report

Firdavs Nasriddinov

11 December 2023

Abstract

In this report, we explore the applications of Meta's two computer vision models, Segment Anything Model (SAM) and DINOv2, in the realm of semantic segmentation and sparse matching. Using pre-trained checkpoints for both models, we show how two images can be segmented with SAM to create a filter mask of a prominent subject in each image, features in the filtered images can be extracted with DINOv2, and similar features across images can then be mapped. We test the runtime of different pre-trained model sizes and determine which is best for the task of determining unknown space objects in optical images taken onboard a satellite.

Introduction

In the rapidly evolving field of computer vision, semantic segmentation and sparse matching have emerged as pivotal techniques for interpreting and understanding images. This report focuses on the implementation of sparse matching of prominent subjects in two images. We do this through the use of Meta's innovative computer vision models: Segment Anything Model (SAM), and DINOv2. Both models were released to the public by Meta in early April of 2023.

SAM is a promptable segmentation system with zero-shot generalization to unfamaliar objects and images, without need for additional training. In other words, it is an AI model that can "cut out" any object, in any image, automatically.

DINOv2 is a self-supervised vision transformer model that produces universal features suitable for image-level visual tasks (image classification, instance retrieval, video understanding) as well as pixel-level visual tasks (depth estimation, sementic segmentation, sparse matching, dense matching).

In this report, we go over steps to sparse match between images using SAM and DINOv2.

Methods and Results

One way to sparse match between certain subjects in different images is to first filter the images so that only the subjects are present and then proceed with matching of features between respective subjects.

A way to filter the images is to apply semantic segmentation and then determine which segments the subjects are in. Then, make a logical matrix representation of the filter where the pixels that lie in the determined segment(s) are 1 and 0 otherwise.

In this project, Meta's Segment Anything Model (SAM) was used for semantic segmentation. Pre-trained checkpoints, each with different model sizes (in terms of number of parameters), for SAM were downloaded. After loading a model and initializing a mask generator, the masks can be generated with a simple function call. After they are generated, the masks get prepared by going through the generated mask objects and forming an image array to represent the masks.

Figure 1: Semantic segmentation applied on an image of a horse with a background. Generated with largest model size. Left shows input image. Right shows all masks generated through semantic segmentation. Middle shows masks overlayed on input image.

As we can see from Figure 1, SAM was able to successfully segment the image. One thing to note is that the generators parameters can be tuned to change the output of the generator to either have more detailed segmentation or less detailed depending on use case.

After the mask overlay is generated we can manually determine which segment corresponds to the subject (the horse) and only display that.

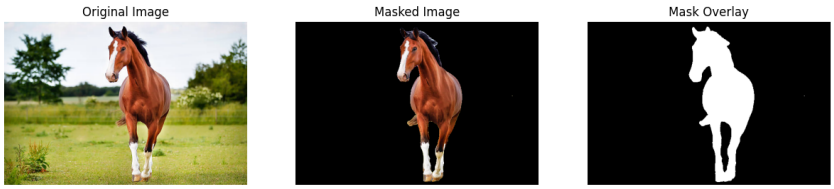

Figure 2: Subject mask filter. The right image is our output mask filter.

After the subject mask of the images are generated, we can use Meta's DINOv2 to extract the features of the masked image. Similar to SAM, pretrained models were downloaded of varying size (number of parameters): "dinov2_vitg14": 1.1 billion, "dinov2_vitl14": 300 million, "dinov2_vitb14": 86 million, "dinov2_vits14: 21 million. After loading a model from PyTorch hub, an image transformer (used ImageNet default parameters) was initialized. To use the model to extract features from an image, the image must first be prepared by applying the image transformer to generate an image tensor and cropping the image tensor to correct dimensions (ensure height and width are multiples of model patch size). Then, the features are extracted as tokens with a function call to the model. The subject mask is then cropped and rescaled to match transformed image. The mask is then applied to the extracted features to generate a dense matching of the subjects.

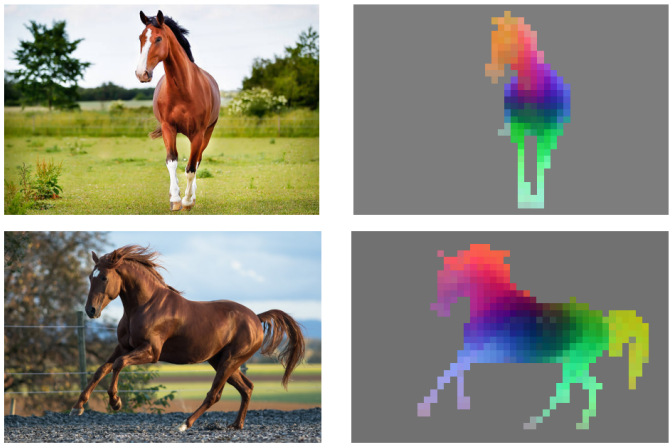

Figure 3: Dense matching (extracted features) of both horses.

In Figure 3, the colors between the two images hold no significance in terms of comparing between images. The gradient of colors signifies the change in features of the subject.

After the features of the two images are extracted, K-Nearest Neighbors (KNN), where $k = 1$, is applied to calculate "distances" between features of one image to the other. These distances are then used to determine a match pair between features of the two images.

Figure 4: Sparse matching between two horses in different images. Only 10% of the matchings are being displayed so that we can have a clear visual of a slight portion of the matchings.

Discussion

1. Testing

Semantic segmentation with SAM was tested using all three model sizes.

Model Size	Avg. Time (s)
Small	5.28
Medium	6.64
Large	7.97

Table 1: Average time over 50 runs to generate masks of first horse image with different SAM model sizes. The smaller the size, the quicker the mask generation.

Note that the model sizes are given qualitatively and not quantitatively (in terms of number of parameters) because the actual model sizes is not made public by Meta, whereas for DINOv2 they are made public.

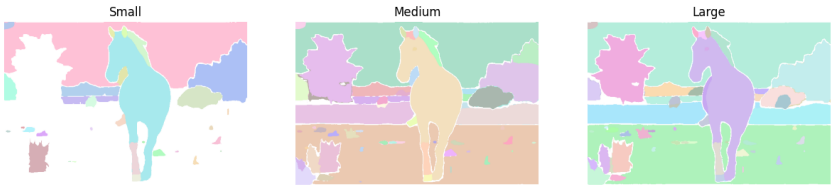

Figure 5: Semantic segmentation through different SAM model sizes. All ran with default SAM parameters.

From Figure 5, we can see that all models achieve very similar segmentations for the horse subject. The larger the size, the more detail in the segmentation. The difference between small model and medium model is fairly noticable while the difference between medium and large is minor.

From the results in Table 1 and Figure 5, using the smallest model is most efficient since there would little to no noticeable difference in the final subject filter mask.

Feature extraction with DINOv2 was also tested using all model sizes.

# Parameters (million)	Avg. Time (s)
21	0.07
86	0.17
300	0.52
1100	1.77

Table 2: Average time over 100 runs to prepare and extract features of first horse image with different DINOv2 model sizes. The smaller the size, the quicker.

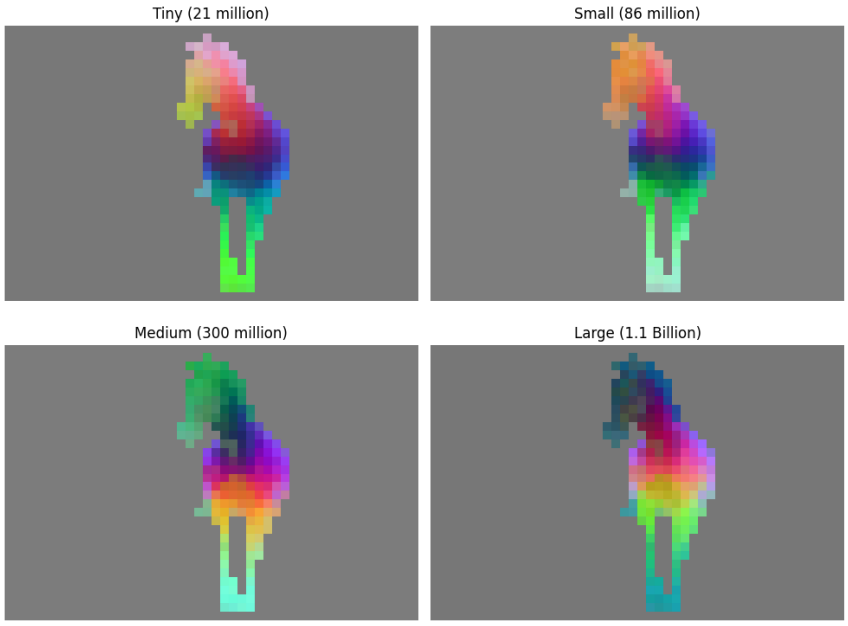

Figure 6: Feature extraction through different DINOv2 model sizes. Images were transformed with ImageNet transformer settings.

In Figure 6, the larger model sizes are better able to pick up the minor differences in features. For example, in the largest model size, it picks up the eyes as a stark contrast to the rest of the body. In addition, with the larger models, more distinct features (i.e. legs vs head) are more distinct in terms of the colors.

In addition, we can see that the time cost of running the larger model is minor and hence using the largest model size would be best.

2. MSEPS Application

When satellites in orbit gather video of surroundings with an optical camera, each frame can be analyzed through semantic segmentation to segment the frame and then apply sparse matching between frames to match similar elements between frames. So, if an unknown space object (i.e. another satellite) crosses by the camera's field of view, the satellite would get segmented in each frame and could be matched between frames to track it.

The process for two continuous frames could be as follows:

1. Apply semantic segmentation to frame A.
2. Apply semantic segmentation to frame B.
3. Match similar corresponding segments between frames.
4. Determine segment velocities (the satellite would move significantly faster than surroundings) by calculating change in relative position over time.
5. Filter out noise and surroundings by ensuring remaining segments move faster than a threshold velocity.

This would make it so that the unknown space object is determined in each frame. This information can be used for various other tasks. For example, the object can then be tracked by manually changing user satellite orientation.

One thing to note in the listed process is that we are now matching segments between frames and not features. This would require tweaking the proposed methods to fit this need.

There are two constraints to consider for using DINOv2 for MSEPS. First, unknown space objects will have a very high velocity and hence will not be in frame for a long time. This means that the models must run quickly so that the space objects are determined and can be tracked before it is too late. To determine which model sizes and parameters to use, the models have to be tested in the use case and then the most efficient one that can do the task must then be selected. Second, there is a limit to a file upload to the satellite. This means that the model checkpoints must fit in a certain sized file to be uploaded in one go. However, an approach to consider is breaking up the files to small parts, uploading each one separately and then reconstructing the large file from the parts on the satellite itself. This approach would have to be further looked into to ensure that it is viable.

Due to these constraints and the results from testing, we can hypothesize that the small sized SAM and medium sized DINOv2 can effectively determine moving space objects.

3. Future Research

After more simulations of satellites in orbit are generated, the models can be tested with simulated images through the process defined in Section 2.

Segmentation and future extraction can be tested through varying the parameters of the model to fit the needs of the model.

```
class SamAutomaticMaskGenerator:
    def __init__(
        self,
        model: Sam,
        points_per_side: Optional[int] = 32,
        points_per_batch: int = 64,
        pred_iou_thresh: float = 0.88,
        stability_score_thresh: float = 0.95,
        stability_score_offset: float = 1.0,
        box_nms_thresh: float = 0.7,
        crop_n_layers: int = 0,
        crop_nms_thresh: float = 0.7,
        crop_overlap_ratio: float = 512 / 1500,
        crop_n_points_downscale_factor: int = 1,
        point_grids: Optional[List[np.ndarray]] = None,
        min_mask_region_area: int = 0,
        output_mode: str = "binary_mask",
    )
```

Listing 1: Initialization parameters for SAM mask generator.

The transformer settings for the image transformer for feature extraction can also be fine tuned.

Another approach to further explore in semantic segmentation is to use DINOv2 itself. For this project, there were difficulties getting that to work in Google Colab and hence SAM was used. This might result in better segmentation times and thus might be the better option. In addition, it would void the reason to have the pre-trained SAM models downloaded as only the DINOv2 models would be required.

Conclusion

In closing, this report has effectively shown how Meta's state-of-the-art computer vision models, SAM and DINOv2, can be applied to sparse matching in images. Leveraging semantic segmentation alongside feature extraction, these models exhibit considerable promise in handling visually complex data. The testing done on various model sizes provides useful insights into their efficiency and performance. Notably, there is potential to use these models to track space objects from satellite photos, opening fresh research directions in this domain. Future work should concentrate on further improving model accuracy and conducting more simulations to fully tap into their capabilities for satellite image analysis.

References

[1] Alexander Kirillov, et al. *Segment anything.* arXiv preprint arXiv:2304.02643, 2023.

[2] Maxime Oquab, et al. *Dinov2: Learning robust visual features without supervision.* arXiv preprint arXiv:2304.07193, 2023.