Fine-tuning Segment Anything Model (SAM) for Satellite Detection in Aerial Images

Firdavs Nasriddinov

19 March 2024

Abstract

In this report, we explore fine-tuning Meta's Segment Anything Model (SAM) for satellite detection in aerial images. SAM is a promptable segmentation system with zero-shot generalization to unfamiliar objects and images. To fine-tune SAM, we first load in a dataset of 1108 train images, 200 validation images and 200 test images where each image is a 512 x 641 grayscale image with a corresponding label that specifies the bounding box surrounding the satellite in view. We then segment each image with all SAM models of different sizes (base, large, huge) as well as MobileSAM (an optimized version of SAM developed for smaller-scale applications). We then iterate through each segmentation for each image and determine if any is within the bounding box for the satellite. With this, we can make a proper dataset for fine-tuning where we have a corresponding segmentation mask for the satellite in the image. This resulted in a final dataset of 411 training images, 102 validation images and 101 test images. In addition, we include another 25 test images without any satellites. Finally, for each image in the train and validation dataset, we resize the image to three difference sizes (512 x 512, 1024 x 1024, 2048 x 2048) and then get 256 x 256 patches of the resized images. We then discard all patches that don't contain any part of the satellite. We then train the SAM model for each resize size and test the model on the test set. We find that the 512 size model gets a test accuracy of 61.9% with an average runtime of _ seconds for each image. The 1024 size model gets a test accuracy of 54.0% with an average runtime of _ seconds for each image. The 2048 size model gets a test accuracy of 39.7% with an average runtime of _ seconds for each image. These runtimes are from running on a _. Based on these results we conclude that the 512 size model is the best model for satellite detection in aerial images.

Introduction

Methods and Results

Each prepared image was gathered from United States Geological Survey (USGS) and satellites were artifically added into frame. The images were then segmented with different SAM checkpoints (base: 93.7 M parameters, large: 312 M parameters, huge: 641 M parameters) as well as MobileSAM (5 M parameters). Then, for each image, we iterate through each segmentation and determine if any is within the bounding box for the satellite. If so, we keep the segmentation mask correspoding to the satellite and discard the rest.

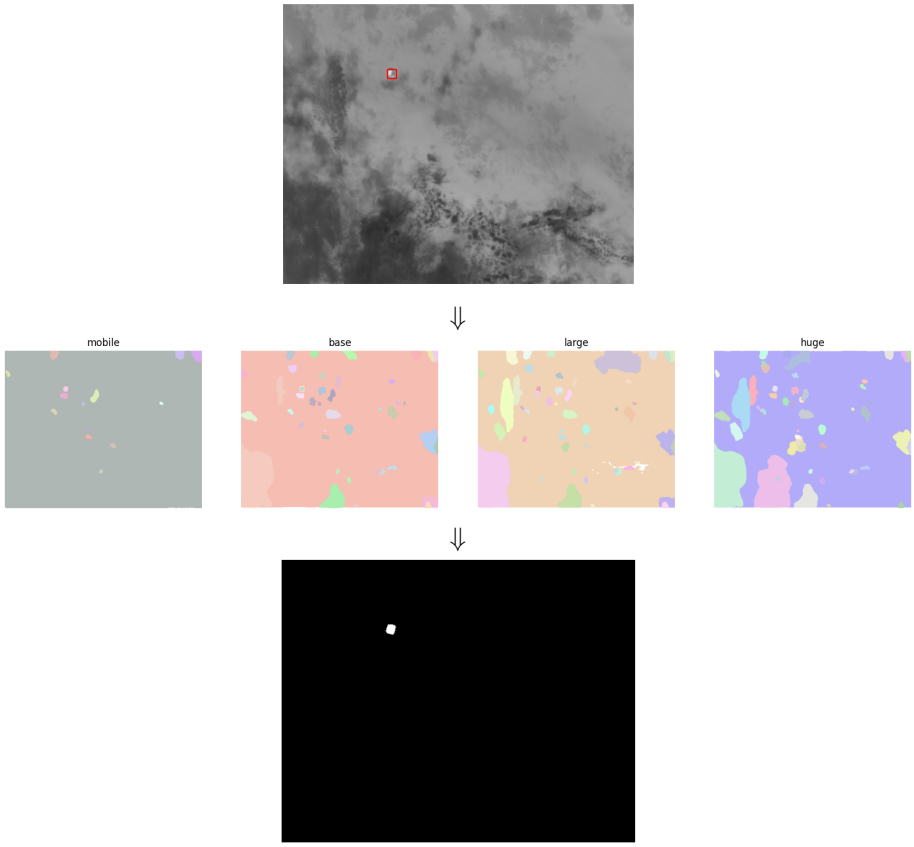

Figure 1: Example of generating satellite mask from input image. We first segment image through each SAM model and then determine segmentation mask within bounding box.

The final dataset for fine-tuning consisted of 411 training images, 102 validation images and 126 test images (25 without any satellites in view). Note that for training all images had satellites in view due to nature of fine-tuning SAM where there must be a valid non-empty segmentation mask for each image.

For fine-tuning we load in the base SAM model from the transformers library and and only train the mask decoder layer. For each model size, we first train on the training set for 25 epochs with a learning rate of 0.00001 with Dice Cross-Enropy Loss (DiceCELoss) from the monai library. DiceCELoss returns the weighted average of the Dice loss [2] and cross-entropy loss between two input tensors.

We make a prediction by first resizing the image to the corresponding image size that matches the model. We then split the image into 256 x 256 patches. We run the model on each patch. We then stitch the patches back together to get the final mask. Foe each patch, the model outputs logits for each pixel in the image. We apply a sigmoid to the logits to get probility matrix that has values between 0 and 1 for each pixel that corresponds to the likelihood of the pixel being a part of the satellite.

$$\text{prob} = \frac{1}{1 + e^{-\text{logits}}} \tag{1}$$

We then threshold the probability matrix at a set value to get the final binary mask. We use the validation set to determine the best threshold value.

We found that a threshold of 0.65 resulted in the highest validation accuracy.

$$\text{pred} = \text{prob}[\text{prob} > 0.65] \tag{2}$$

After, we resize the probabilty and prediction to the original image dimensions. Below is an example of a prediction.

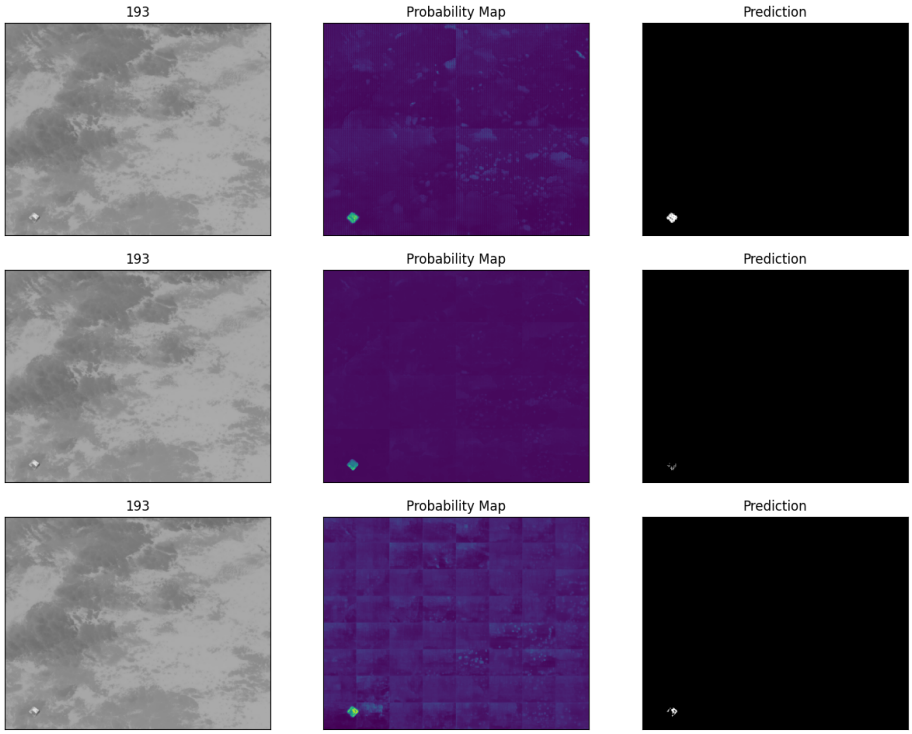

Figure 2: Example of prediction for test image 193 with different image/model sizes. Top to bottom: 512, 1024, 2048. From the probability map, we can see the different patches from the slight variation in color at the borders.

For determining the accuracy of the model, we use the joint intersection over union (IoU) metric. The joint IoU is the intersection over union of the predicted mask and the ground truth mask. We test that the joint intersection is over 50% to mark it as a valid prediction.

Discussion

1. *Testing*

Image/Model Size	Accuracy (%)	Time on RTX 4080 Super (s)	Time on Jetson TX2 (s)
512	61.9	0.55	
1024	54.0	2.10	
2047	39.7	8.26	

Table 1: Test accuracy and average runtime for each model size.

We see that the 512 size model has the highest accuracy and lowest runtime. The high accuracy is likely due to the fact that the model has to process less patches and thus has less chance of predicting a satellite when there isn't one. The low runtime is due to the fact that the model has to process less patches.

2. Future Research To improve model accuracies, there are several things that can be further explored:

- Get more data. The current dataset is relatively small and more data can help the model generalize better.
- Train on more epochs. The current model was only trained for 25 epochs. Training for more epochs can help the model learn more features. However, this can also lead to overfitting and thus a balance must be struck.
- Fine-tune a larger sized SAM model. The current model was only fine-tuned on the base model.
- Use a different loss function. The current model was trained with DiceCELoss. There are other loss functions that can be explored such as the Generalized Dice Loss [2].
-

Conclusion

References

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

[2] Carole Helene Sudre, et al. *Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations*, Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, held in conjunction with MICCAI 2017 Quebec City, QC,..., 2017, Vol. 2017, pp. 240–248. `https://api.semanticscholar.org/CorpusID:21957663`