Term Project Report: SAM

Xiaohui Zhang
School of Mathematical Sciences, Peking University
2200010821
2200010821@stu.pku.edu.cn

Abstract

We apply Segment Anything Model (SAM) to BCTV dataset. We first examine its performance, finding that its lacks enough generalization ability and performs bad. Then we finetune the model on BCTV dataset, and find that it performs better than the original model. Meanwhile, we seek to classify the strategies of the organs besides segmenting them, so we add a classification head to the model and train it on BCTV dataset. We find that the model performs well on the classification task. Finally, we talk about the problems we met and the future work we can do.

1 Introduction

Image segmentation is an essential component in many computer vision systems. And many researchers have been interested in this field and proposed many methods to solve this problem.
Last year, a new model called Segment Anything Model (SAM)[1] has been proposed by Meta AI Research, FAIR. It is a promptable model and is pre-trained on a broad dataset SA-1B dataset. However, studies[2,3] have shown that the model lacks enough generalization ability and performs bad on some datasets.
Medical image segmentation, as an important branch of image segmentation, surely can not be ignored. Many medical image analysis tasks rely on image segmentation, such as organ segmentation, lesion detection, and tumor delineation. So there is a need to develop a model that can segment medical images well.
In this project, we first apply SAM to segment medical image of BTCV dataset[4]. We find that the model performs bad on this dataset. So we finetune the model, making it perform better on BTCV. Meanwhile, we seek to classify the strategies of the organs besides segmenting them, so we add a classification head to the model. Finally, we talk about the problems we met and the future work we can do.

2 Preliminaries

2.1 Segment Anything Model

Built on the largest segmentation dataset to date, SAM is a foundation model for image segmentation. It is a promptable model and is pre-trained on a broad dataset SA-1B dataset. And it has three main components: an image encoder, a prompt encoder and a mask decoder.

2.1.1 Content of SA-1B dataset

SA-1B dataset consists of many diverse, high-resolution, licensed, and privacy-protecting images and high-quality segmentation masks. The images contain only daily scenes such as natural landscape, animals and furniture.

Preprint. Under review.

2.1.2 Image Encoder

The Image Encoder is an MAE pretrained Vision Transformer (ViT)[5] minimally adapted to process high resolution inputs. It runs once per image and can be applied prior to prompting the SAM model.

2.1.3 Prompt Encoder

The Prompt Encoder supports two set of prompts: sparse (points, boxes, text) and dense (masks). Points and boxes are represented by positional encodings summed with learned embeddings for each prompt type. Dense prompts are embedded using convolutions and summed element-wise with the image embedding.

2.1.4 Mask Decoder

The Mask Decoder maps the image embedding, prompt embedding, and an output token to a mask. It employs a modification of a Transformer decoder block followed by a dynamic mask prediction head. Here the decoder block uses a *two-way-attention* block, i.e. prompt self-attention and cross-attention in two directions (prompt-to-image embedding and vice-versa), to get all the embeddings. After running two blocks, the image embedding is upsampled and fed into an MLP to get the output token, which is finally fed into a dynamic linear classifier to compute foreground probability at each pixel.

2.2 BTCV Dataset

BTCV dataset[4] is a challenging medical segmentation dataset, which contains 3-D CT volumes of 13 organs (some of which are not complete), including Spleen, Right Kidney, Left Kidney, Gallbladder, Esophagus, Liver, Stomach, Aorta, IVC, Portal and splenic veins, Pancreas, Right adrenal gland, and Left adrenal gland. The dataset is split into 24 training volumes and 6 validation volumes with images and their labels.

3 Data Preprocess and Augmentation

Data Loading

The BTCV raw dataset is stored in **.nii.gz** format, which is a common format for medical images. We find that the **monai**[6] package has provided a convenient and efficient framework for loading and preprocessing this format of data. So we take advantage of this package to load BTCV dataset.

Data Separation

BCTV dataset contains 3-D CT volumes, each of which contains 13 organs (some of which are not complete). Due to the characteristics of scanning, the volumes have some pixels of background, which is useless for organ segmentation. So we first scan the volumes and separate the slices with no organ (i.e. totally background), and then save each meaningful slice as a 2-D image.
For classification, we also save the 2-D images of each organ separately by spliting them according to the ground truth masks with their class label, making it easier to get the true class.

Data Augmentation

In order to eliminate overfitting and increase the generalization ability of the model, we use the **monai.transforms** package to augment the data. Considering the characteristics of the data, we mainly use the following transforms:

- **Orientation**: change the orientation of the data
- **ScaleIntensity**: scale the intensity of the data to the range [0, 1]
- **RandFlip**: randomly flip the data in the spatial dimension
- **RandRotate**: randomly rotate the data in the spatial dimension

- **RandShiftIntensity**: randomly shift the intensity of the data
- **ToTensor**: convert the data to tensor

4 Apply SAM to BTCV dataset

We first apply SAM to segment medical images of BTCV dataset. We use the original model without finetuning to deal with the 2-D slices of the 3-D CT volumes. Several types of prompts are considered in this experiment, including points and boxes. For points, we use the following prompts: *one central point*, *one random point* and *three random points*. Specifically, the central point is calculated with **cv2.distanceTransform** function, which can be used to calculate the distance of each pixel to the nearest background pixel. Then we choose the pixel with the maximum distance as the central point. The random points are generated with **random.randint** function with seed 0. For boxes, we use the following prompts: *bound box* and *larger box*, where *larger box* extends its corresponding *bound box* in width and height both for 5 pixels respectively.
We take mDice over different organs as the evaluation metric, which is defined as follows:

$$Dice = 2 \times \frac{intersection}{union} \tag{1}$$

where

$$union = A \cup B, \quad intersection = A \cap B \tag{2}$$

where A and B are the slice of the ground truth and the prediction mask in binary type respectively.

The mDice of different prompts are shown in Table 1.

Table 1: mDice of different prompts

Prompt	
Type	mDice
one central point	0.6124
one random point	0.5929
three random points	0.6437
bound box	0.8649
larger box	0.8267

From the performance of different prompts with SAM without finetuning, we can see that the model performs better on the box prompts than the point prompts in general. We propose that a box can just bound the organ, which leads to a more precise location of the segmentation region. For point prompts, SAM performs better on the *three random points* prompt than the other two prompts. This is probably because the *three random points* prompt can provide more information, thus making the segmentation region more accurate. For box prompts, the *bound box* prompt performs better than the *larger box* prompt. This is mainly because the *larger box* prompt may contain more background pixels, which slightly confuses the model.

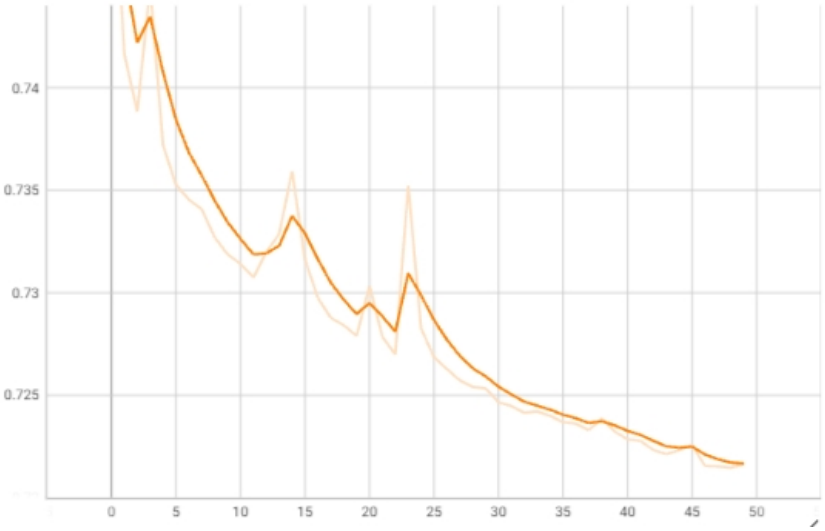
5 Finetune SAM on BTCV dataset

5.1 Preparation

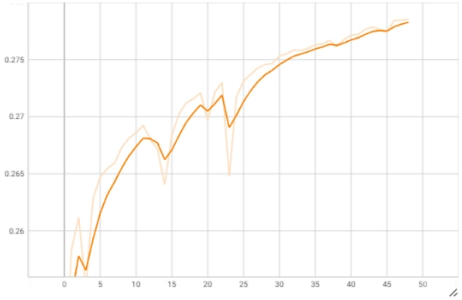
As mentioned above, in order to save cuda memory usage and increase training speed, we save the preprocessed 2-D slices of the 3-D CT volumes and load the SAM checkpoint ViT-H model without its image encoder.

5.2 Training Strategy

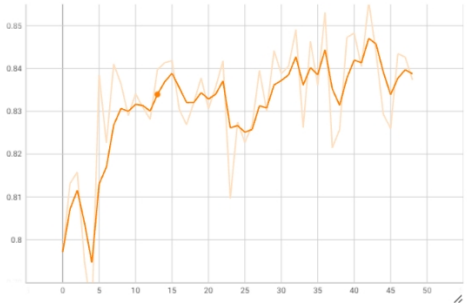
As the Dice loss of different slices may have a large variance, we take a relatively large batchsize of input data to get a stable gradient. This is one of the most important settings in our training strategy.



(a) training loss



(b) training dice



(c) validation dice

Figure 1: one central point

We use the SGD optimizer with momentum 0.9 and weight decay 0.0001. The learning rate is set to 0.1, which is relatively high because the value of Dice loss and gradient are small. The total training epoch is 50. We finetune the model with different prompts, including points and boxes mentioned in Section 4.

5.3 Curve and Final Performance

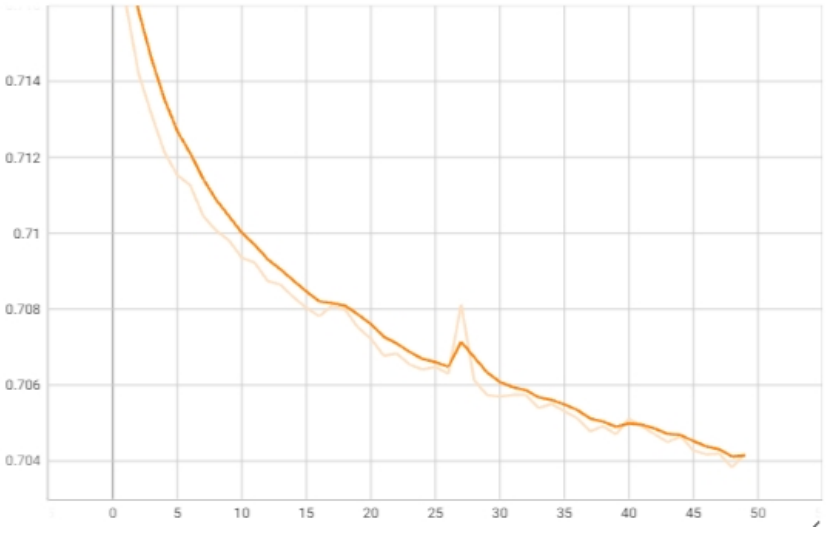
Here we take the example of both prompt *one central point* and *bound box*. Curve of the training process is shown in Figure 1 and Figure 2. We can see that the model converges quickly and the training loss is stable. Here the training loss and validation loss are both Dice loss. The reason why they are not in the same order of magnitude is that we normalize the predicted mask in the training process to ensure the smooth gradient backpropagation and stable parameters updating.

Table 2 shows the mDice of different prompts after finetuning.

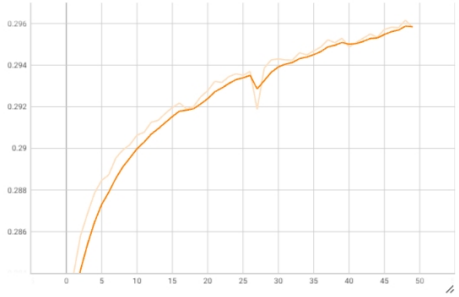
Tabel 3 shows the comparison between our finetuned model and other baseline.

6 Classify the organs based on SAM

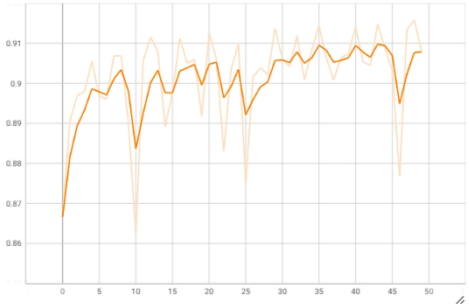
After finetuning and segmenting on the BTCV dataset with SAM, we can consider classifying each segmented organ. So we try to modify the mask decoder based on the original SAM **Mask Decoder**, adding a **class_prediction_head** parallel with **Mask Output Token** and **IoU Output Token**. Similar to the prediction of the masks, here we make use of image embedding and prompt



(a) training loss



(b) training dice



(c) validation dice

Figure 2: bound box

Table 2: mDice of different prompts after finetuning

Prompt Type	mDice(original)	mDice(finetuned)	improved(%)
one central point	0.6124	0.8556	39.71
one random point	0.5929	0.8535	43.95
three random points	0.6437	0.8532	32.55
bound box	0.8649	0.9158	5.89
larger box	0.8267	0.9161	10.81

Table 3: performance comparison

Model	U-Net[7]	ResU-Net[8]	DeepLabV3[9]	PIPO-FAN[10]	Finetuned SAM(bound box)
Avg mDice(%)	92.1	92.4	89.2	93.6	91.6

tokens to generate final tokens for classification. The **class_prediction_head** outputs a class logits for prediction.

6.1 Training Strategy

We fix the finetuned parameters of **Prompt Decoder** and original **Mask Decoder**, only training the ***two-way-Transformer*** and **class_prediction_head** in the modified classifier model. Here we use the Adam optimizer with learning rate 1e-5. The total training epoch is 50.

6.2 Final Performance

Table 4 shows the classification accuracy of different organs with point prompts.

Table 4: Classification accuracy of different organs with point prompts

organ	
Type	Accuracy
Spleen	0.8477
Right Kidney	0.7917
Left Kidney	0.8631
Gallbladder	0.7647
Esophagus	0.9139
Liver	0.9544
Stomach	0.8729
Aorta	0.9134
Inferior Vena Cava	0.7818
Portal Vein and Splenic Vein	0.5802
Pancreas	0.7698
Right Adrenal Gland	0.3167
Left Adrenal Gland	0.7639
Weighted Average	0.8331

6.3 Visualization and Analysis

We can find that the classification accuracy varies greatly among different organs. We think there are three main reasons. First, the distribution of the data of different organs is quite different, thus leading to different training difficulty and variance of the classification accuracy. According to statistic, the proprotion of Liver, Aorta, and Inferior Vena Cava take 12.93%, 17.28% and 14.97% respectively, while Right Adrenal Gland takes only 2.72%. Second, the segmentation dice of different organs is also different, which may affect the classification accuracy. Third, the region of some organs is too small, which makes the attention mechanism of the model hard to focus on the region of interest, thus leading to a low classification accuracy.
Figure 3 shows one example of the classification results. The left is the raw image and the right is the region of *Right Adrenal Gland* , which is very small. We can see that the model fails to classify it correctly.

7 Problems and Future Work

Although we have finetuned the SAM model and add a classification head after the original model, the performance of the model is still far from satisfactory and has much space to be improved. More efficient and effective methods to get a better segmentation and higher accuracy are waited to be explored.
Segmentation is a challenging task, especially for medical images. However it can not be the terminal in this field. We can also consider other tasks, such as lesion detection, tumor delineation and so on. We can also try to design a model that can distinguish abnormal organs and even classify

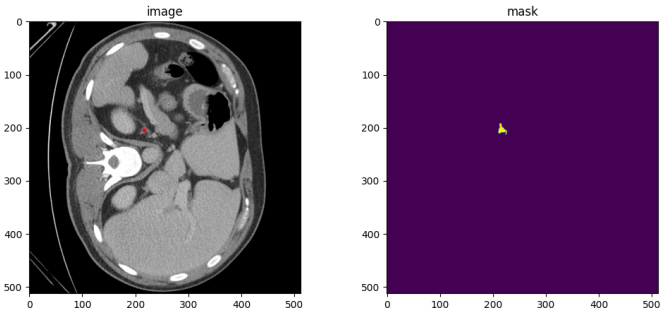


Figure 3: Classification Results

the kind of pathological tissues or organs, thus helping doctors to diagnose diseases more accurately and quickly.
In terms of SAM, the lack of direct 3-D segmentation ability is an open question to be explored. We can thing about how to extend the model to 3-D segmentation and how to make it more efficient and effective in more diverse fields.

References

[1] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg,Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. arXiv:2304.02643, 2023.

[2] Sheng He, Rina Bao, Jingpeng Li, Jeffrey Stout, Atle Bjørnerud, P. Ellen Grant, Yangming Ou. Computer-Vision Benchmark Segment-Anything Model (SAM) in Medical Images: Accuracy in 12 Datasets. arXiv:2304.09324, 2023.

[3] Saikat Roy, Tassilo Wald, Gregor Koehler, Maximilian R Rokuss, Nico Disch, Julius Holzschuh, David Zimmerer, and Klaus H Maier-Hein. SAM.MD: Zero-shot medical image segmentation capabilities of the Segment Anything Model. arXiv:2304.05396, 2023.

[4] Landman, B., Xu, Z., Igelsias, J., Styner, M., Langerak, T., Klein, A.. Multi-atlas labeling beyond the cranial vault-workshop and challenge. 2017.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. ICLR, 2021.

[6] M. Jorge Cardoso, Wenqi Li, Richard Brown, Nic Ma, Eric Kerfoot, Yiheng Wang, Benjamin Murrey, Andriy Myronenko, Can Zhao, Dong Yang, Vishwesh Nath, Yufan He, Ziyue Xu, Ali Hatamizadeh, Andriy Myronenko, Wentao Zhu, Yun Liu, Mingxin Zheng, Yucheng Tang, Isaac Yang, Michael Zephyr, Behrooz Hashemian, Sachidanand Alle, Mohammad Zalbagi Darestani, Charlie Budd, Marc Modat, Tom Vercauteren, Guotai Wang, Yiwen Li, Yipeng Hu, Yunguan Fu, Benjamin Gorman, Hans Johnson, Brad Genereaux, Barbaros S. Erdal, Vikash Gupta, Andres Diaz-Pinto, Andre Dourson, Lena Maier-Hein, Paul F. Jaeger, Michael Baumgartner, Jayashree Kalpathy-Cramer, Mona Flores, Justin Kirby, Lee A.D. Cooper, Holger R. Roth, Daguang Xu, David Bericat, Ralf Floca, S. Kevin Zhou, Haris Shuaib, Keyvan Farahani, Klaus H. Maier-Hein, Stephen Aylward, Prerna Dogra, Sebastien Ourselin, Andrew Feng. MONAI: An open-source framework for deep learning in healthcare. arXiv:2211.02701, 2022.

[7] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. MICCAI, 2015.

[8] X. Han. Automatic Liver Lesion Segmentation Using A Deep Convolutional Neural Network Method. arXiv: 1704.07239, 2017.

[9] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. arXiv:1706.05587, 2017.

[10] Xi Fang, Pingkun Yan, Senior Member, IEEE. Multi-organ Segmentation over Partially Labeled Datasets with Multi-scale Feature Abstraction. arXiv:2001.00208, 2020.

Code Discription

1. Apply SAM to BTCV dataset and use mDice to evaluate the model's performance. Code is in file "segmentation_task1.py".

2. Finetune SAM with BTCV dataset and make comparison with the original model on the segmentation performance. Code is in file "finetune_task2.py".

3. Modifiy the original mask decoder to enable the model to classify the organs besides simply segmenting. Code for modified classifier is in file "modified_classifier.py", and code for its training and validation is in file "classify_train_task3.py" and "classify_val_task3.py" perspectively.