MobileSam (Re-implementation)

Anonymous submission

Abstract

The number of parameters used in Deep Learning models continues to grow as researchers search for state of the art results. However, this presents a problem, as the number of model parameters increases so do the computational resources needed to evaluate the model. The purpose of this investigation was to determine the effectiveness of using a large foundation model to train another smaller model designed to run on mobile/embedded devices.

This study is a re-implementation of a previous paper focused on training a small parameter model for the deep learning task known as image segmentation. Specifically, the Segment Anything Model recently released by Meta was used as the large foundation model. A technique known as knowledge distillation was used to transfer the knowledge in the larger foundation model to a smaller model. After using knowledge distillation to train the smaller model, the model was then compared using a few different metrics: total parameter count, time required to evaluate the model (latency) and finally, the Mean Intersection over Union (mIoU) of the segmentation masks produced by the model.

The resulting model has 6 M parameters compared to the teacher model's 641 M parameters. Unsurprisingly, the smaller model also has a much lower latency as well, 0.33 seconds compared to 2.15 seconds for the larger foundation model. Finally, the model shows impressive results for it's much smaller size with a mIoU of 63%.

Code — https://anonymous.4open.science/r/ECE570-FINAL-D9EA/README.md
Video — https://anonymous.4open.science/r/ECE570-FINAL-D9EA/ProjectOverview.mp4
Datasets — https://ai.meta.com/datasets/segment-anything/

Introduction

Artificial Intelligence (AI) and Deep Learning are currently transforming many different industries. As AI models and hardware improve, the range of applications and use cases continues to expand. One straightforward yet powerful way to enhance model performance is by increasing the model's parameter size. However, as models grow more complex, deploying them efficiently becomes a significant challenge, particularly on resource-constrained or embedded devices. The need for optimized models and the ability to run them locally is critical, as this can enhance user privacy and reduce interaction latency. To meet these demands, various techniques like quantization, parameter pruning, and knowledge distillation are being explored to streamline AI models for efficient deployment.

An exciting recent development in AI is the push toward foundational models, such as Meta's open-source Segment Anything Model (SAM) (Kirillov et al. 2023) for image segmentation. SAM's release represents a major contribution to the open-source AI community, as high-quality foundational models and large scale datasets in the computer vision space are not very abundant. The combination of vast amounts of image data and this open-source model unlocks new possibilities for innovation and application development. This paper focused on utilizing knowledge distillation to train a SAM model designed for use on mobile and embedded devices, addressing the challenge of bringing sophisticated AI capabilities to resource-limited platforms. The key objective is to reduce the models runtime latency primarily by reducing the number of parameters in the model while retaining much of the foundation model's performance.

Related Work

Segment Anything

The SAM project's main contribution was an image segmentation foundation model and a very large segmentation dataset of over 1 billion segmentation masks and 11 million source images (Kirillov et al. 2023). Provided an input image and a prompt (i.e. free text, points on the image, bounding box and/or another mask) the task of the model is to produce a *valid* segmentation mask. Validity here is emphasized because the prompt provided could lead to an ambiguous optimal result. For example, if the input image is a spotted dairy cow and the prompt is a point on one of the cows spots, a valid segmentation mask could be the entire cow or simply that single spot.

The model architecture consists of three main components, the image encoder, prompt encoder and mask decoder. The input image is passed into the image encoder resulting in an image embedding. Then the image embedding along with the prompt encoder outputs are passed into the mask decoder. To resolve ambiguous prompts, the mask decoder is designed to output multiple segmentation masks. In practice, for a simple prompt such as a single point, the number of segmentation masks output is relatively small. One ben-

efit of designing the the model this way is that any given image embedding (produced by the image encoder) can be reused multiple times by the mask decoder which amortizes the computational costs of generating the image embedding. The image encoder presented in the original SAM paper is a modified vision transformer (ViT) and three different versions are presented, ViT-B (91M parameters), ViT-L (308M parameters) and ViT-H (636M parameters). The mask decoder is much less computationally complex with only 5M parameters.

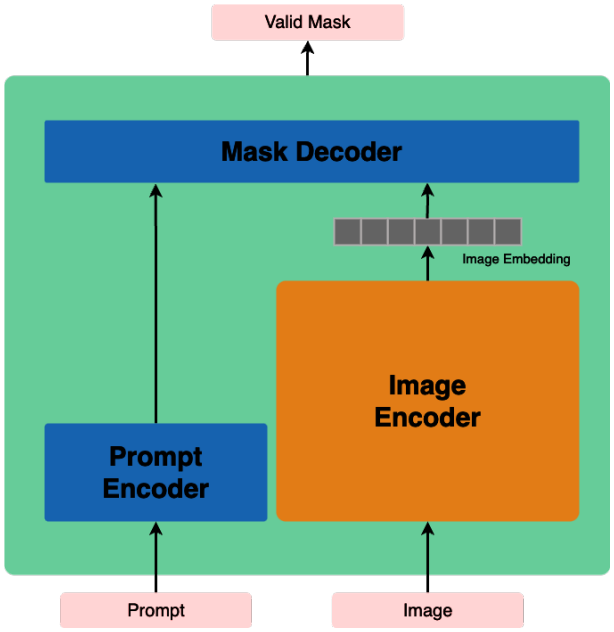

Figure 1: Segment Anything Model (SAM) Overall Architecture

(Kirillov et al. 2023) provides an extensive review of results. The model's zero-shot mask prediction was evaluated on 23 segmentation datasets (not seen during training) using a single prompt point. The SAM model performed very well and showed better accuracy than another common segmentation model (RITM (Sofiiuk, Petrov, and Konushin 2022)) on 16 of the 23 datasets. The model was also evaluated on numerous other task that it was not specifically trained for included edge detection, object proposal generation, instance segmentation and free form text to mask tasks. The model did not perform better than other state of the art models in these additional tasks; however, it did show promising results that indicate the model could easily be composed into other advanced vision applications.

This paper will treat the original SAM model as the gold standard (truth data) and will build upon the work by reusing the prompt encoder and mask decoder architectures and their respective pre-trained weights. The goal is to replace and train the image encoder (the component with the largest number of parameters) with a much smaller vision transformer. The scale of the SA-1B dataset provided in (Kirillov et al. 2023) is incredible with 11 million source images and an average of 100 segmentation masks per image. This dataset was reused during the training phase of the smaller vision transformer.

Just to provide reference to what an input and output of an image segmentation model might look like, an example of the original SAM model input and outputs masks are shown in figures 2 and 3 respectively. This specific example represents evaluating the model over a grid of points in the image.

Figure 2: Segment Anything Model (SAM) example input

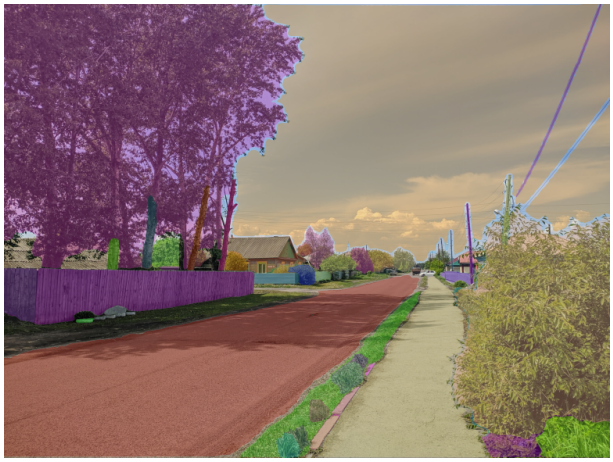

Figure 3: Segment Anything Model (SAM) example output masks

Tiny-ViT

(Wu et al. 2022) provides two main contributions, a fast pretraining distillation framework and the model architecture of tiny vision transformers. The distillation framework is designed to transfer the knowledge of a pretrained larger/-foundational model to a smaller model. This allows smaller models to benefit from the large scale data typically required by foundational models while avoiding the computational complexity of training the large model. The fast pretraining distillation framework begins by evaluating the the teacher model over the input dataset and storing the output logits to

disk. Simply doing this would eliminate the need for forward passes through the teacher model during training of the student model which significantly reduces GPU memory consumption and allows for larger batch sizes during training of the smaller model. However, saving the full logits from the teacher model can consume significant disk space and as a result, the paper describes a method of converting the full/-dense logits to sparse logits before saving. Another method used to reduce the amount of required disk space is using an encoder/decoder to reduce the size of the input image data augmentation parameters. The paper proposes a simple pseudo random number generator as an encoder implementation so the framework could save off a single value (the PRNG seed) for each set of data augmentation parameters. The model architecture of the TinyViT is described in detail and a full Pytorch implementation is provided at the corresponding GitHub repository. The hierarchical vision transformer (Pan et al. 2021) is used as the basic architecture and is progressively scaled down using contraction factors (Feichtenhofer 2020).

Utilizing the distillation framework described above, the TinyViT model achieves 84.8% top-1 accuracy on ImageNet-1k with only 21 million parameters. The model achieves 83.1% top-1 accuracy when training from scratch on the ImageNet-1k dataset. The paper specifically mentions how the distillation framework provides this model accuracy improvement by allowing the forward passes through the teacher model to filter out the hard samples (i.e. samples that have multiple equally prominent objects present). This paper isn't the first to discuss knowledge distillation with respect to ViT models, there are others such as (Shen and Xing 2022); however, (Wu et al. 2022) focuses on the pretraining stage (evaluation and saving of the teacher model logits) and presents the novel approach of using an encoder/decoder pair to save the data augmentation parameters. The work here will build upon this paper by reusing the TinyViT model architecture to replace the original SAM model's image encoder and also making use of he pretraining distillation framework to train the TinyViT on the SA-1B dataset.

MobileSAM

(Zhang et al. 2023) brings the idea of knowledge distillation from (Wu et al. 2022) and the impressive zero-shot performance of SAM model (Kirillov et al. 2023) to resource-constrained devices. The goal of the paper is to replace the ViT in SAM with a smaller model that is able to run on mobile devices. There are three different approaches discussed that each revolve around how the smaller SAM model is trained. The first method is termed coupled distillation in which the input image is passed through the original (i.e. teacher) SAM model and the original mask decoder outputting a set of segmentation masks. Then pass the same input image through a smaller ViT and a similar mask decoder to get another set of masks. Calculating the loss by comparing the two masks (i.e. focal loss (Ross and Dollár 2017) and dice loss (Milletari, Navab, and Ahmadi 2016)) provides a way to train the smaller ViT and mask decoder models. However, this leads to difficulty trying to train both components simultaneously. Thus the author present semi-coupled distillation procedure in which the mask decoder from the original SAM model is used for the smaller SAM model. This makes it easier to train since the masks parameters are frozen and the mask decoder is only being evaluated in the forward pass. There are additional problems with this setup because the choice of mask decoder prompt is still random. As a third alternative, the author presents a decoupled distillation procedure in which the original SAM model ViT-H is evaluated and the image embedding output is directly used to train the smaller models ViT. This setup reduces training complexity in multiple ways because it completely removes the use of the mask decoder model in training and the image embeddings can be compared using simple mean-squared-error loss function.

MobileSAM's results are quite surprising since the smaller model was only trained on knowledge distilled image embeddings from 1% of the original SAM dataset. When comparing the mobile SAM model masks to the original SAM model by measuring the mIoU (mean intersection over union) it achieves $> 70\%$ for zero-shot single-point prompts. The mobile SAM model is also 5x faster to evaluate than the original SAM model when running on a single GPU and was trained in a single day.

This paper will build upon MobileSAM's work by attempting to re-implement the results. Neither (Kirillov et al. 2023) or (Zhang et al. 2023) provide the training/evaluation code in their respective GitHub repositories, therefore, the re-implementation will have a heavy focus on the training and evaluation aspect of MobileSAM.

Problem Definition

The related work collectively addresses the technical challenges involved in adapting large parameter models like the Segment Anything Model (SAM) for deployment on mobile or embedded devices. Specifically, how to maintain high model performance/accuracy while reducing computational complexity during inference and training, allowing models to run in a resource-constrained environment. The TinyViT paper focuses on a family of small and fast vision transformers (Wu et al. 2022). It introduces a knowledge distillation framework where a large "teacher" model transfers its knowledge to smaller (less parameters) models. The paper emphasizes preserving model accuracy while reducing the model's size. Additionally, the knowledge distillation framework presented addresses the issue of data storage when using very large training datasets which are usually employed when training large parameter foundation models. The SAM paper, released by Meta, outlines the data collection methods, training strategies, and architectural design of the SAM model (Kirillov et al. 2023). One of the main challenges solved by the SAM project is that the high-quality mask data required to train the segmentation model didn't exist at the time, thus, the SAM paper covers the design of a data engine that was used to produce the 1 billion masks for training. Finally, MobileSAM brings together ideas from both TinyViT and SAM, to create a new model capable of performing image segmentation on resource constrained devices (Zhang et al. 2023). The MobileSAM model leverages the knowledge distillation techniques from TinyViT to create a more

lightweight version of SAM, capable of running efficiently on mobile devices or even in a web browser.

None of the above mentioned related work provides the data loading code for SA-1B or the training code for TinyViT using the knowledge distillation framework. Therefore, this paper will focus on the details of loading the large-scale SA-1B dataset with a custom developed Pytorch DataLoader, the training code used to distill the knowledge from the foundation SAM model to a much smaller TinyViT and finally, the code and details for evaluating the performance of the image segmentation task using mIoU.

Methodology

Data Loading

The first part of getting starting with any deep learning project usually involves loading the dataset. There are many implementations of data loaders that allow deep learning practitioners to load them using very few lines of code (e.g. Pytorch Image Models - *https://timm.fast.ai/*). However, this doesn't exist for the SA-1B dataset currently. This is likely a result of the sheer size of the dataset. The full dataset size of SA-1B is 11 million source images and 1 billion masks and is packaged as a list of URLs, where each URL points to a tar file containing on average 11,000 images and their corresponding segmentation masks. To train the student model, the masks are not actually needed, only the source images and the image embedding produced by the image encoder. Thus, the code accompanying this paper includes a Pytorch DataSet (*https://pytorch.org/tutorials/beginner/basics/data_tutorial.html*) implementation for both the pretraining dataset which loads the SA-1B images as well as a DataSet implementation for the training of the student model which loads the same images and the image embeddings (produced during the pretraining stage). The student model DataSet implementation also includes the image pre-processing that is required prior to feeding the image into the image encoder component. Ultimately, this implementation allows users to use the SA-1B dataset like they are accustomed to using other popular models inside the Pytorch ecosystem. For this specific re-implementation, only a 2% (220,000 images) of the full SA-1B dataset was used for pretraining and training. Even with such a small percentage, approximately 1.5 TB of disk storage was required for storing the images, segmentation masks and image embeddings. To future-proof the DataSet implementation, a configuration parameter was included so that the user can specify the minimum number of samples to download from the full SA-1B dataset.

Pretraining

After the dataset has been downloaded, the pretraining stage can begin. The knowledge distillation framework described in (Wu et al. 2022) describes saving the teacher model logits to disk so that they can be easily reused during each epoch of training of the student model. This insight becomes very important for large-scale foundation models like SAM that are computationally expensive to run inference on. Because disk space was not an issue (4 TB available), this paper did not implement sparsification of the teacher logits to save space (just saved the dense logits instead). Although this step is simple in terms of the code implementation, this will consume a signification portion of time and hardware resources because the teacher model is so much larger than the student model. To implement this stage, for each input image in the dataset, simply perform a forward pass through the SAM image encoder and save the resulting image embedding to disk for later use.

Training

Following the downloading of the dataset and the pretraining stage where the image embeddings are calculated and stored, is training of the student model. This portion of the project is where the most development time was spent because there weren't any code examples provided by the related work. The first step is to develop the model architecture for the TinyViT which in general is quite complex. Thankfully, this implementation and the implementation from (Zhang et al. 2023) both reuse the architecture components developed in (Wu et al. 2022). Some of the high level configuration for the model architecture include an input image size of 1024x1024 with 3 channels, 4 multi-head attention layers where each layer has a different number of attention heads. For the purpose of this project, the model architecture is not as much the focus as the knowledge distillation from teacher to student model and as a result, this project reuses the model components from TinyViT and configuration parameters from MobileSAM. Because the TinyViT model architecture configuration parameters were reused from MobileSAM, the image embedding tensor size was the same size expected at the input to the SAM mask decoder component. This meant, the trained TinyViT could be used as a drop in replacement for the image encoder in the SAM project. Another benefit of implementing the Pytorch Dataset interface for SA-1B is that the training and testing loops can be the standard Pytorch implementations for model training. A batch size of 8 was used to improve training efficiency and match the MobileSAM training process. After developing the model architecture, training and testing loop, next was establishing the choice of optimizer and loss function. The loss function was chosen to be mean-squared error since the training was comparing the TinyViT image embedding output to the SAM ViT-H image embedding for the same input image. Choosing the optimizer was not so easy, trying with standard SGD didn't work at all and the loss quickly exploded to infinity. However, the Adam optimizer performed pretty well with a learning rate of 0.001 with the addition of an exponential LR scheduler (gamma = 0.9) to reduce Adam's learning rate after each training epoch.

All three steps, downloading the dataset, calculating the teacher model image embeddings (pretraining) and training of the student model were very time consuming and required professional grade hardware. Since there was approximately 1.5 TB of data to download (2% SA-1B), a fast and stable internet connection was necessary. Both the pretraining and training stages are best run using a CUDA capable GPU with at least 16 GB of video ram. For all of these reasons, a cloud GPU service was used during the development of

this project. Any decent server grade GPU will work but an Nvidia RTX 4000 Ada was specifically chosen for it's computational power to cost ratio. The dataset download took nearly a day because the DataSet implementation is currently single threaded. Creating the image embeddings using the teacher model took nearly 36 hours and after determining which optimizer and parameters to use, training for two epochs took approximately 10 hours.

Evaluation

After training, but before the model's performance could be evaluated, the trained weights for TinyViT needed to be combined with the pretrained weights from the original SAM model for the mask decoder and prompt encoder components. Since Pytorch saves/loads the weights as simple Python dictionaries, this was achieved by replacing the SAM model image encoder dictionary keys with the ones from training. Also, the Python based model had to have the image encoder element replaced by the TinyViT implementation. Evaluating the model involved calculating the mean Intersection over Union (mIoU) over a subset of the full dataset. mIoU is a common metric used to evaluate segmentation model performance. The general approach used for this project was to consider the masks generated for a given input image by the original SAM model as truth and then calculate the IoU compared to the mask produced by the trained TinyViT based model. Averaging over all the masks results in the overall mIoU. However, doing this for a single point and single mask per input image is not enough. Instead the input image was divided up into a grid of points and a mask was calculated for all points using both the teacher SAM model and the trained TinyViT based model. Then averaging the IoU over all of these masks provided the final mIoU result.

A visual example of the evaluation results are show in figures 4-7. Figures 4 and 5 show the output of the TinyViT based model and the original SAM model evaluated for a single input point (prompt) in the image and figures 6 and 7 show the same models evaluated over a grid of prompt points in the image. Note that the TinyViT based model doesn't perfrom as well in the regions around the ground and sky.

Figure 4: TinyViT based model evaluated with single point input prompt.

Figure 5: Original SAM model evaluated with single point input prompt.

Experimental Results

All experimental results are summarized by the following table. Note how the model has a decent mIoU while being much smaller in number of parameters. Also note that this is only slightly less than the results obtained by the MobileSAM model. The latency results recored here are based on the MacOS Pytorch MPS backend.

Conclusion and Future Directions

Although this project didn't introduce any novel topics itself, it makes use of prior ideas and re-implements the training of MobileSAM. This project has demonstrated that the knowledge distillation framework is an effective method for using large-scale foundational models as teacher models to train a much smaller model suitable for deployment on edge and resource constrained devices. The scale of data and inference of the teacher model may be computationally expensive but this only needs to be performed once and can be reused in many training rounds of the student model. Although the model doesn't perform as well as the teacher it still has decent performance considering the number of parameters were reduced by nearly 100x.

There are many things that could be done in future work to extend this re-implementation. One simple thing would be to add some data augmentation to the input images, for example, training on both RGB and BGR representations as well as image rotations, scaling, and adding noise. This was likely done by the MobileSAM authors to achieve a higher mIou. The model probably could've been trained for more than 2 epochs as the MobileSAM model was trained for up to two epochs. However, this implementation used 220,000 images instead of the 110,000 used in (Zhang et al. 2023). One performance improvement that would speed up the dataset loading is to parallelize the downloading and decompression of each SA-1B tar file. Currently, the this is single threaded and the download provider is band-limiting the rate at which

Figure 6: TinyViT based model evaluated with grid of points.

Figure 7: Original SAM model evaluated with grid of points.

each tar file can be downloaded. Training of the model could potentially have been improved by attempting to sparsity the teacher logits as this is somewhat similar to dimensionality reduction as it prioritizes the most signification logits in the image embeddings. Additionally, the current implementation only compares mask outputs of the original SAM model and the custom trained TinyViT based model; however, the custom model masks could also be compared to the MobileSAM masks if desired. Finally, as the goal of this project is to train a model suitable for resource constrained devices, it would be an interesting demo to see this running live in the browser. The original SAM model does have a browser based demo but the image embeddings are simple downloaded for a small set of sample images and MobileSAM's web based demo is using a Hugging Face backed for model processing. It should be possible to run directly in the browser using such frameworks as ONNX.

	Original SAM	Mobile SAM	Custom SAM
Image Encoder Parameters	637 M	6 M	6 M
Total Model Parameters	641 M	10.1 M	10.1 M
Model Latency (single point)	2.15 s	N/A	0.33 s
mIoU	1.0 (truth)	0.7447	0.6284

Table 1: TinyViT based model compared to Original SAM and MobileSAM

References

Feichtenhofer, C. 2020. X3D: Expanding Architectures for Efficient Video Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; Dollar, P.; and Girshick, R. 2023. Segment Anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 4015–4026.

Milletari, F.; Navab, N.; and Ahmadi, S.-A. 2016. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, 565–571. Ieee.

Pan, Z.; Zhuang, B.; Liu, J.; He, H.; and Cai, J. 2021. Scalable Vision Transformers With Hierarchical Pooling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 377–386.

Ross, T.-Y.; and Dollár, G. 2017. Focal loss for dense object detection. In *proceedings of the IEEE conference on computer vision and pattern recognition*, 2980–2988.

Shen, Z.; and Xing, E. 2022. A fast knowledge distillation framework for visual recognition. In *European conference on computer vision*, 673–690. Springer.

Sofiiuk, K.; Petrov, I. A.; and Konushin, A. 2022. Reviving Iterative Training with Mask Guidance for Interactive Segmentation. In *2022 IEEE International Conference on Image Processing (ICIP)*, 3141–3145.

Wu, K.; Zhang, J.; Peng, H.; Liu, M.; Xiao, B.; Fu, J.; and Yuan, L. 2022. Tinyvit: Fast pretraining distillation for small vision transformers. In *European conference on computer vision*, 68–85. Springer.

Zhang, C.; Han, D.; Qiao, Y.; Kim, J. U.; Bae, S.-H.; Lee, S.; and Hong, C. S. 2023. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*.