CAHSI Project Report

May 23, 2024

Advisors: Professor Cihang Xie and Professor Kuan Huang

Students: Josh Angel and Oliver Chang

1 Motivation

Image segmentation has been in a variety of fields such as autonomous driving, medical imaging, and object-dection. Segment Anything (SA), by Meta, has further improved the vision-field by introducing the first segmentation foundation model. Segment Anything comprises of multiple promptable segmentation tasks, a transformer-based model, and data engine (providing 1+ billion masks and 11 million images). As such, Segment Anything boasts impressive zero-shot performances on novel datasets.

We seek to apply SA to medical imagines. In particular, we segment COVID-19 chest x-ray scans from Qatar University & Tampere University & Hamad Medical Corporation. However, computationally fine-tuning a foundation model efficiently and tailored models like ViTDet-H still outperform SA's zero-shot capabilities stand in the way from accomplishing this task. Thus we turn to Low Rank Adaptation (LoRA) to efficiently fine-tune SAM on COVID-19 lung scans.

2 Dataset Preparation

2.1 Helper functions

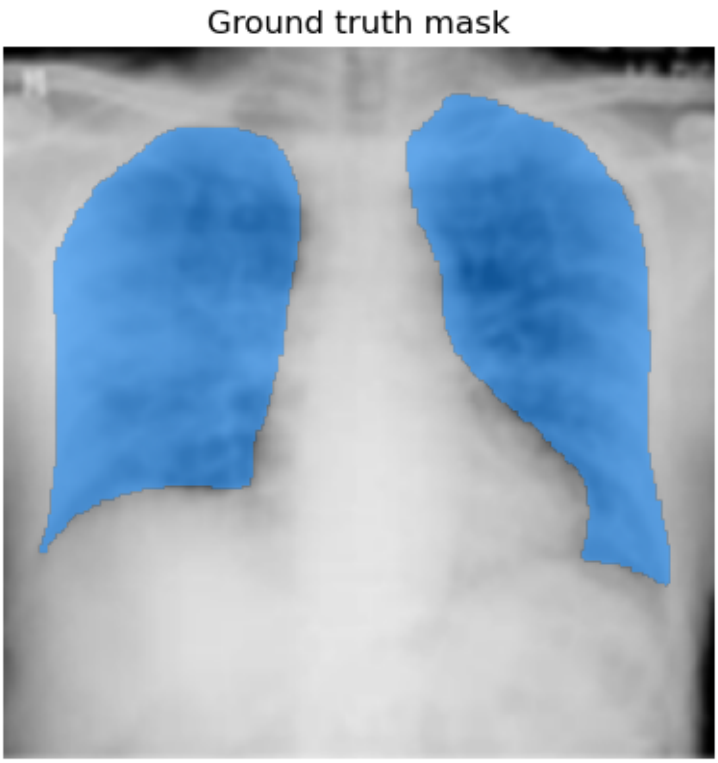
2.2 Load in subset of raw data as numpy arrays

2.3 Make dataset dictionary

```
[4]: {'image': <PIL.PngImagePlugin.PngImageFile image mode=RGB size=256x256>,
      'label': <PIL.PngImagePlugin.PngImageFile image mode=L size=256x256>}
```

2.4 Display sample

```
[5]: (-0.5, 255.5, 255.5, -0.5)
```



2.5 SAM Dataset Prep

3 Fine-tuning

```
pixel_values torch.Size([2, 3, 1024, 1024])
original_sizes torch.Size([2, 2])
reshaped_input_sizes torch.Size([2, 2])
input_boxes torch.Size([2, 1, 4])
ground_truth_mask torch.Size([2, 256, 256])
```

```
SAM total params: 93729252
```

3.1 LoRA integration

3.1.1 Mask Decoder

3.1.2 Vision Encoder

```
LoRA-SAM total params: 13440996
Percentage of params reduced: 0.8565976393367569
```

```
 38%|        | 15/40 [00:10<00:17,  1.46it/s]
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
```

```
Cell In[21], line 20
     17 # Training phase
     18 for batch in tqdm(train_dataloader):
     19   # forward pass
---> 20   outputs = model(pixel_values=batch["pixel_values"].to(device),
     21                  input_boxes=batch["input_boxes"].to(device),
     22                  multimask_output=False)
     23   # compute loss
     24   predicted_masks = outputs.pred_masks.squeeze(1)

File /opt/conda/lib/python3.11/site-packages/torch/nn/modules/module.py:1501, i
 ↪Module._call_impl(self, *args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File /opt/conda/lib/python3.11/site-packages/transformers/models/sam/
 ↪modeling_sam.py:1355, in SamModel.forward(self, pixel_values, input_points,
 ↪input_labels, input_boxes, input_masks, image_embeddings, multimask_output,
 ↪attention_similarity, target_embedding, output_attentions,
 ↪output_hidden_states, return_dict, **kwargs)
   1352 vision_hidden_states = None
   1354 if pixel_values is not None:
-> 1355     vision_outputs = self.vision_encoder(
   1356         pixel_values,
   1357         output_attentions=output_attentions,
   1358         output_hidden_states=output_hidden_states,
   1359         return_dict=return_dict,
   1360     )
   1361     image_embeddings = vision_outputs[0]
   1363     if output_hidden_states:

File /opt/conda/lib/python3.11/site-packages/torch/nn/modules/module.py:1501, i
 ↪Module._call_impl(self, *args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
```

```
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File /opt/conda/lib/python3.11/site-packages/transformers/models/sam/
 ↪modeling_sam.py:1043, in SamVisionEncoder.forward(self, pixel_values,␣
 ↪output_attentions, output_hidden_states, return_dict)
   1038     layer_outputs = self._gradient_checkpointing_func(
   1039         layer_module.__call__,
   1040         hidden_states,
   1041     )
   1042 else:
-> 1043     layer_outputs =␣
 ↪layer_module(hidden_states, output_attentions=output_attentions)
   1045 hidden_states = layer_outputs[0]
   1047 if output_attentions:

File /opt/conda/lib/python3.11/site-packages/torch/nn/modules/module.py:1501, i␣
 ↪Module._call_impl(self, *args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File /opt/conda/lib/python3.11/site-packages/transformers/models/sam/
 ↪modeling_sam.py:936, in SamVisionLayer.forward(self, hidden_states,␣
 ↪output_attentions)
    933     height, width = hidden_states.shape[1], hidden_states.shape[2]
    934     hidden_states, padding_shape = self.window_partition(hidden_states,
 ↪self.window_size)
--> 936 hidden_states, attn_weights = self.attn(
    937     hidden_states=hidden_states,
    938     output_attentions=output_attentions,
    939 )
    940 # Reverse window partition
    941 if self.window_size > 0:

File /opt/conda/lib/python3.11/site-packages/torch/nn/modules/module.py:1501, i␣
 ↪Module._call_impl(self, *args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
```

```
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File /opt/conda/lib/python3.11/site-packages/transformers/models/sam/
 ↪modeling_sam.py:829, in SamVisionAttention.forward(self, hidden_states,␣
 ↪output_attentions)
    826 batch_size, height, width, _ = hidden_states.shape
    827 # qkv with shape (3, batch_size, nHead, height * width, channel)
    828 qkv = (
--> 829     self.qkv(hidden_states)
    830     .reshape(batch_size, height * width, 3, self.num_attention_heads, - )
    831     .permute(2, 0, 3, 1, 4)
    832 )
    833 # q, k, v with shape (batch_size * nHead, height * width, channel)
    834 query, key, value = qkv.reshape(3, batch_size * self.
 ↪num_attention_heads, height * width, -1).unbind(0)

File /opt/conda/lib/python3.11/site-packages/torch/nn/modules/module.py:1501, i␣
 ↪Module._call_impl(self, *args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File /opt/conda/lib/python3.11/site-packages/loralib/layers.py:242, in␣
 ↪MergedLinear.forward(self, x)
    240 result = F.linear(x, T(self.weight), bias=self.bias)
    241 if self.r > 0:
--> 242     result += self.lora_dropout(x) @ T(self.merge_AB().T) * self.scalin
    243 return result

File /opt/conda/lib/python3.11/site-packages/loralib/layers.py:215, in␣
 ↪MergedLinear.merge_AB(self)
    209     return w.transpose(0, 1) if self.fan_in_fan_out else w
    210 delta_w = F.conv1d(
    211     self.lora_A.unsqueeze(0),
    212     self.lora_B.unsqueeze(-1),
    213     groups=sum(self.enable_lora)
    214 ).squeeze(0)
--> 215 return T(self.zero_pad(delta_w))
```

```
File /opt/conda/lib/python3.11/site-packages/loralib/layers.py:208, in␣
 ↪MergedLinear.merge_AB.<locals>.T(w)
    207 def merge_AB(self):
--> 208     def T(w):
    209         return w.transpose(0, 1) if self.fan_in_fan_out else w
    210     delta_w = F.conv1d(
    211         self.lora_A.unsqueeze(0),
    212         self.lora_B.unsqueeze(-1),
    213         groups=sum(self.enable_lora)
    214     ).squeeze(0)

KeyboardInterrupt:
```

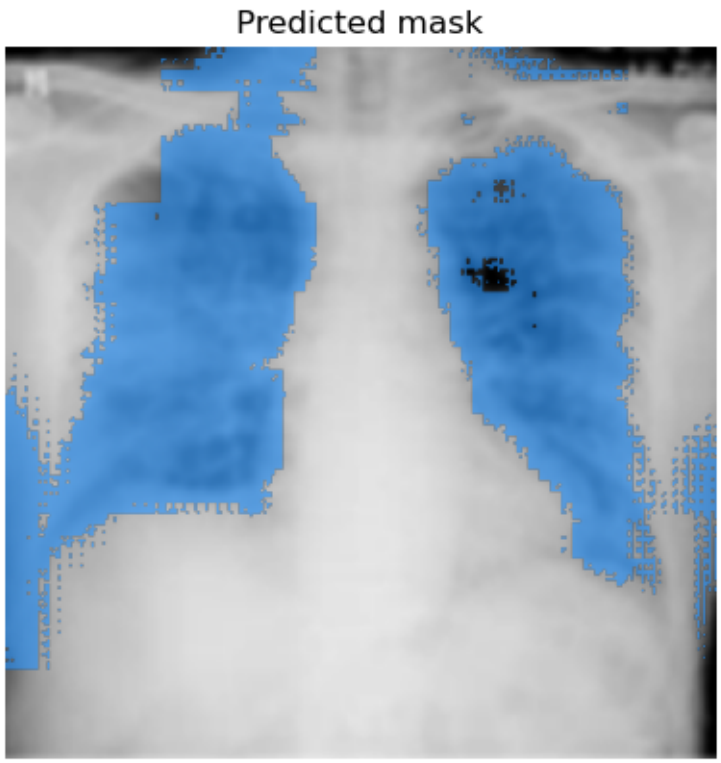
4 Inference

[49]:

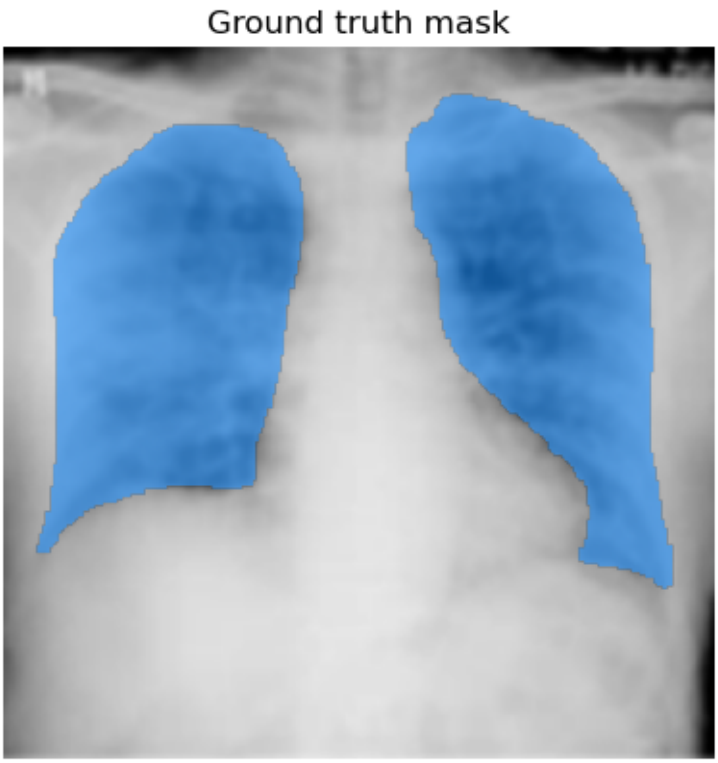
```
pixel_values torch.Size([1, 3, 1024, 1024])
original_sizes torch.Size([1, 2])
reshaped_input_sizes torch.Size([1, 2])
input_boxes torch.Size([1, 1, 4])

(1, 1, 4)
(256, 256, 1)
```

[53]: (-0.5, 255.5, 255.5, -0.5)



[54]: (-0.5, 255.5, 255.5, -0.5)




```
Sample IoU: 0.6381479822453883
```

5 Inference Exam

5.1 Load Test Set

```
Test data shape: (100, 256, 256, 3)
Test labels (masks) data shape: (100, 256, 256)
```

[65]:
```
['datasets/QaTa-COV19/QaTa-COV19-v2/Test
Set/Images/sub-S08047_ses-E16280_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test
Set/Images/sub-S08047_ses-E16628_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test
Set/Images/sub-S08047_ses-E16837_run-1_bp-chest_cr.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test
Set/Images/sub-S08047_ses-E17130_run-1_bp-chest_cr.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test
Set/Images/sub-S08047_ses-E17511_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test
Set/Images/sub-S08047_ses-E17539_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test
Set/Images/sub-S08047_ses-E17682_run-1_bp-chest_vp-ap_dx.png',
```

```
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test
Set/Images/sub-S08047_ses-E17934_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test
Set/Images/sub-S08047_ses-E18100_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test
Set/Images/sub-S08047_ses-E26555_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test
Set/Images/sub-S08074_ses-E18689_run-1_bp-chest_vp-ap_cr.png']
```

[66]:
```
['datasets/QaTa-COV19/QaTa-COV19-v2/Test Set/Ground-
truths/mask_sub-S08047_ses-E16280_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test Set/Ground-
truths/mask_sub-S08047_ses-E16628_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test Set/Ground-
truths/mask_sub-S08047_ses-E16837_run-1_bp-chest_cr.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test Set/Ground-
truths/mask_sub-S08047_ses-E17130_run-1_bp-chest_cr.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test Set/Ground-
truths/mask_sub-S08047_ses-E17511_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test Set/Ground-
truths/mask_sub-S08047_ses-E17539_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test Set/Ground-
truths/mask_sub-S08047_ses-E17682_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test Set/Ground-
truths/mask_sub-S08047_ses-E17934_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test Set/Ground-
truths/mask_sub-S08047_ses-E18100_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test Set/Ground-
truths/mask_sub-S08047_ses-E26555_run-1_bp-chest_vp-ap_dx.png',
 'datasets/QaTa-COV19/QaTa-COV19-v2/Test Set/Ground-
truths/mask_sub-S08074_ses-E18689_run-1_bp-chest_vp-ap_cr.png']

/opt/conda/lib/python3.11/site-packages/datasets/features/image.py:339:
UserWarning: Downcasting array dtype int64 to uint8 to be compatible with
'Pillow'
  warnings.warn(f"Downcasting array dtype {dtype} to {dest_dtype} to be
compatible with 'Pillow'")
/opt/conda/lib/python3.11/site-packages/datasets/features/image.py:348:
UserWarning: Downcasting array dtype int64 to int32 to be compatible with
'Pillow'
  warnings.warn(f"Downcasting array dtype {dtype} to {dest_dtype} to be
compatible with 'Pillow'")
```

[71]:
```
Dataset({
    features: ['image', 'label'],
    num_rows: 100
})
```

```
Average IoUs over 100 test samples: 0.31679712669981375
```