How to segment (m)ANY THING(s)

Tim Hudelmnaier

14.01.2025

1. **Calculate image embeddings on the EMBL cluster**
2. **Test assisted and automatic segmentation using micro-sam**
3. **Learn how to finetune your very own models**

Getting Started

Setting up the environment to run interactive segmentation using micro-sam

Install pixi

Linux / MacOS

```shell
curl -fsSL https://pixi.sh/install.sh | bash
```

Windows

```powershell
iwr -useb https://pixi.sh/install.ps1 | iex
```

Download code to run Micro-Sam

```shell
git clone https://github.com/tim-hudelmaier/sam_finetuning_utils.git
```

Running the Demo

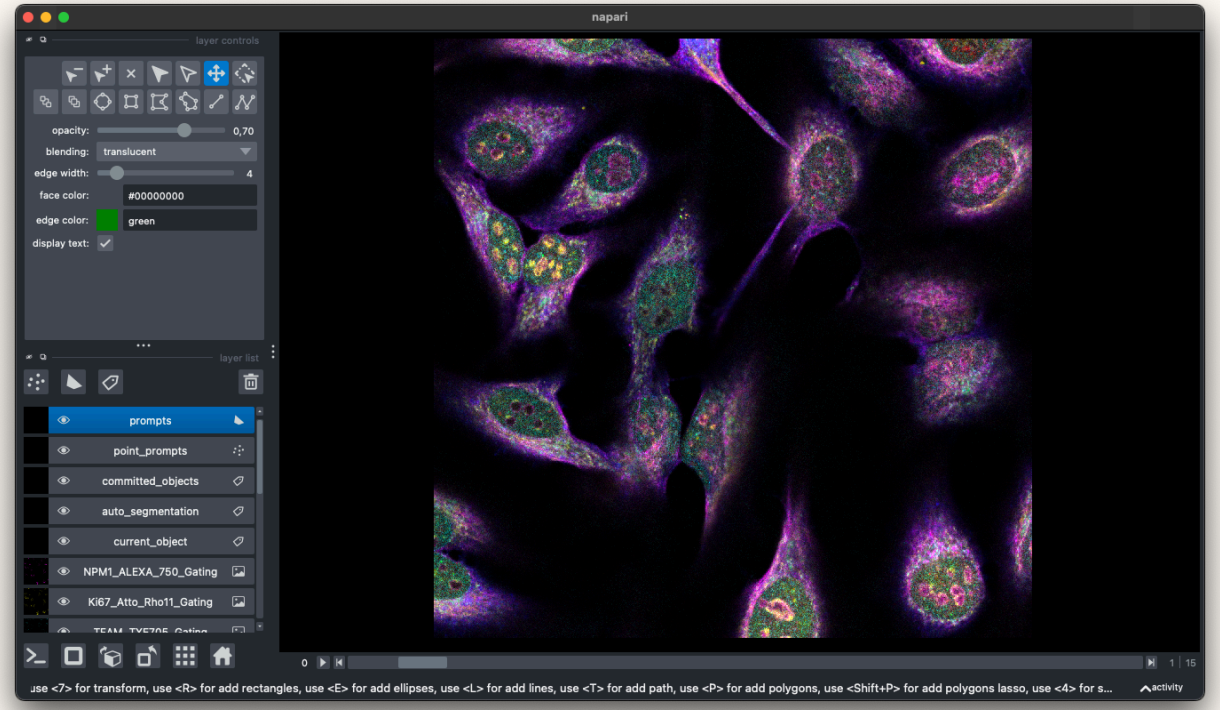
Start Napari and load and image to use for segmentation.

Shortcurt

```
pixi run demo
```
Shell

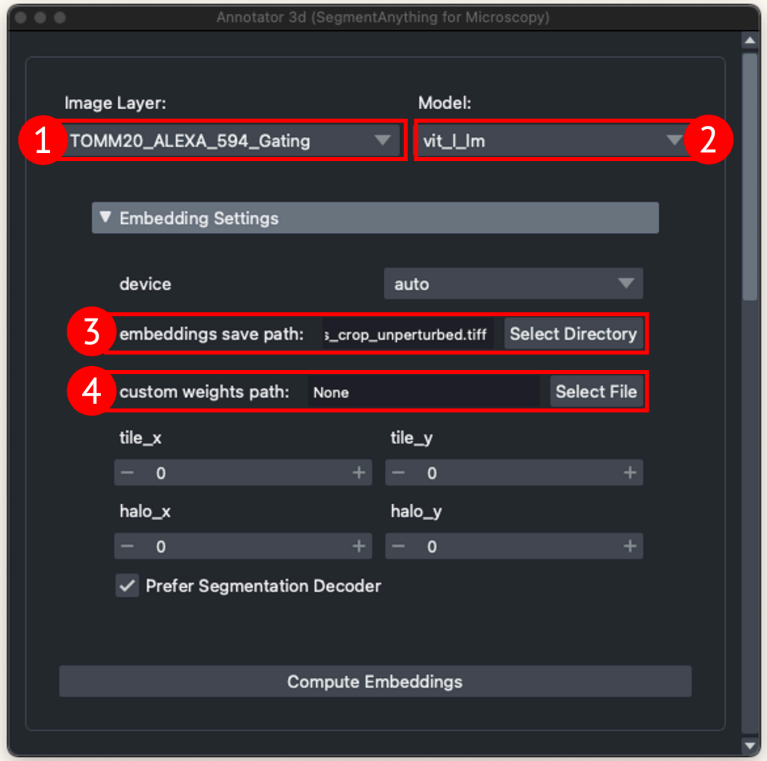
Bring your own image

```
pixi run
python -m
napari
```
Shell



Automatic Segmentation

Open the Micro-Sam plugin using Plugins > SegmentAnything for Microscopy > Annotator 3D



1. Channel to use if you calculate embeddings locally. Not important if you use precomputed embeddings.
2. Model to use to segment the image (*vit_l_lm* works best, *vit_b_lm* is the only one I can run on my machine).
3. Path to precomputed embeddings (for demo: `sample_data/demo_embeddings_unperturbed`)
4. Path to finetuned model.

Prompt Based Segmentation

Shortcuts for faster segmentations

Moving things into view

< / >	Switch Between z-planes
Space	Hold to pan view

Point based prompting

2 / P	Start point prompt tool
t	Toggle prompt (positive / negative)
3	Select points
del	Delete Points
S	Segment using point prompts
Shift + C	Clear segmentations
C	Commit / save segmentations

Precomputing embeddings yourself (1/2)

Configuring your run on the cluster.

```Shell
apptainer exec --nv --bind /scratch:/scratch docker://timjhudelmaier/pixi-micro-sam-th:latest \
  /bin/bash -c "cd /repo && pixi run python -m automatic_3d_segmentation \
  --img_path /path/to/your/image.tif \
  --ndim 3 \
  --model_type vit_l_lm \
  --checkpoint /which/model/to/use/best_model.pt \
  --output_path /where/to/store/segmentation_masks.tiff \
  -channels_path /optional/path/to/subset/of/channels.txt \
  --merge_channels True \
  --merge_method max \
  --embeddings-out-path /where/to/save/your/embeddings"
```

Precomputing embeddings yourself (2/2)

Settings of the job scheduler on the cluster.

```shell
#SBATCH -J your-job-name
#SBATCH -A saka
#SBATCH -t 48:00:00
#SBATCH -N 1
#SBATCH -p gpu-el8 -C gaming -n 16 --gres=gpu:3090:1 --mem-per-gpu 64283
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --mail-user=your.mail@embl.de
#SBATCH -o /where/to/write/logs/slurm.%N.%j.out
#SBATCH -e /where/to/write/logs/slurm.%N.%j.err
```

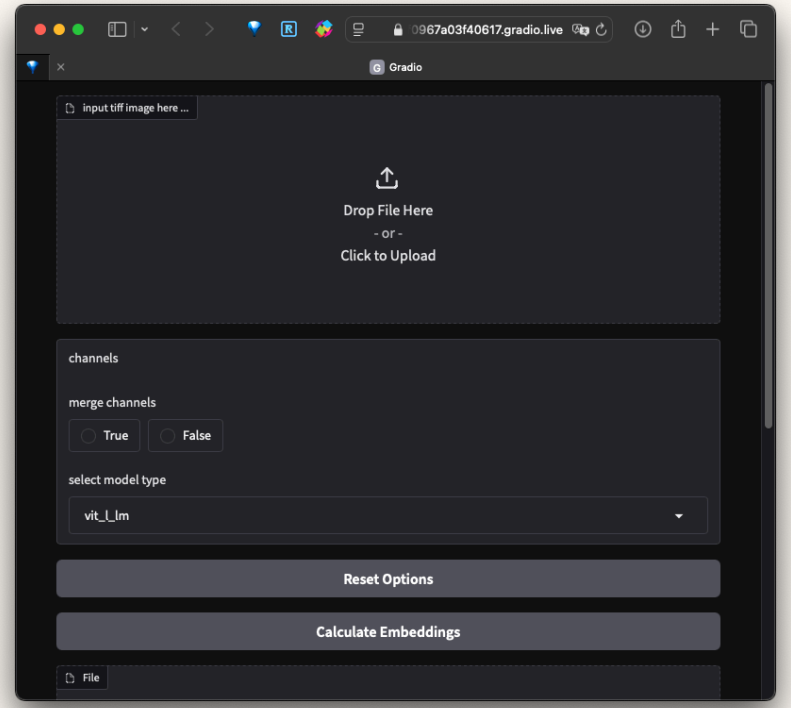
Starting your job on the cluster

```shell
sbatch path/to/run_finetune.sh
```

This sucks!

Precomputing Embeddings the Easy Way

You can now easily access this workflow with a web interface.



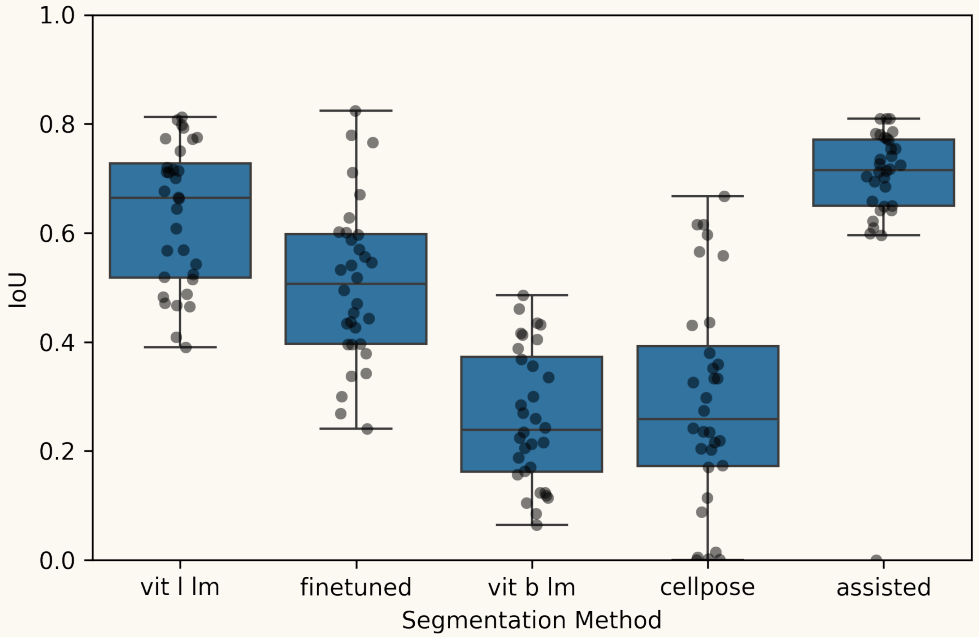
Gradio Demo Site

- upload your image
- select which model and which channels to use
- calculate embeddings
- download everything

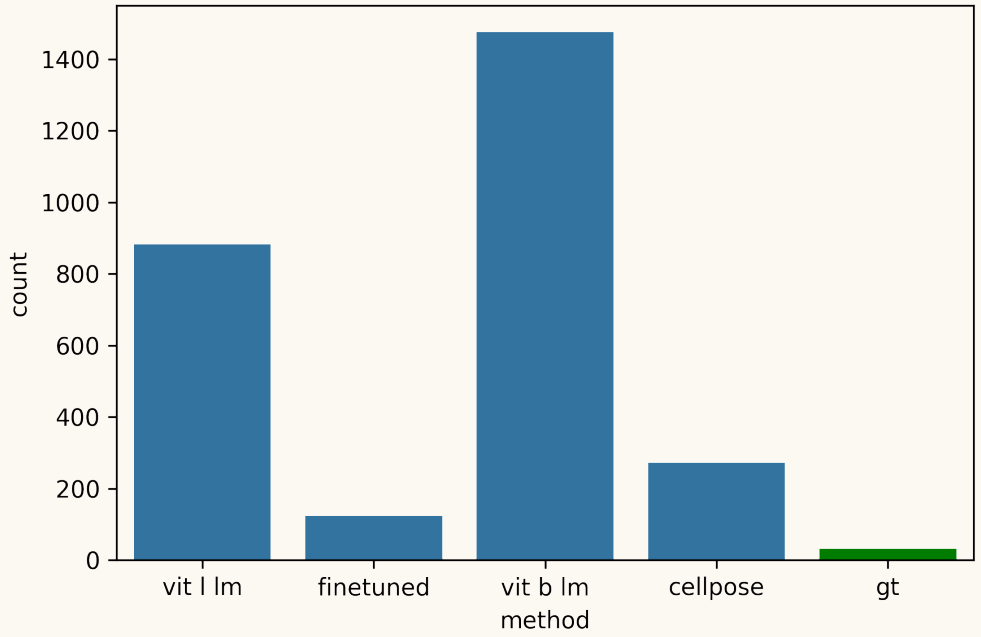
Finetuning

Finetuned models generate less false positive masks, but compromise IoU.

IoU of models vs segmentation by hand



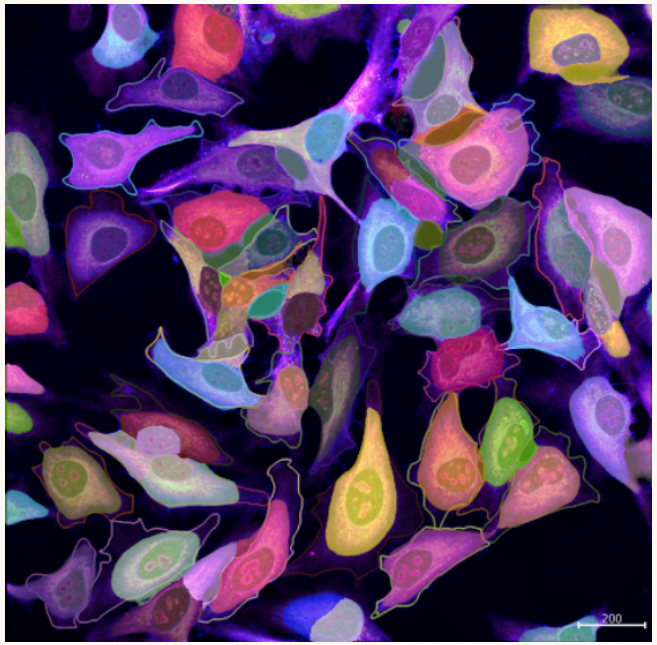
Number of masks generated by each model



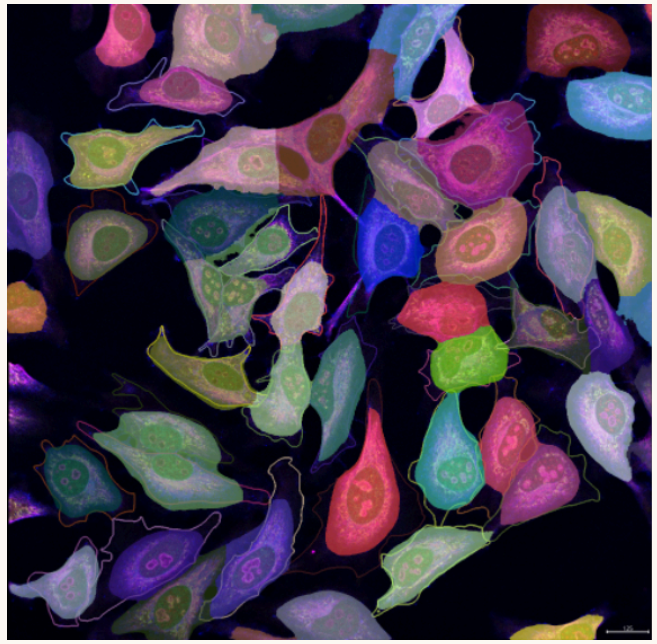
Finetuning

Finetuned models tend to oversegment. Cellpose heavily undersegments.

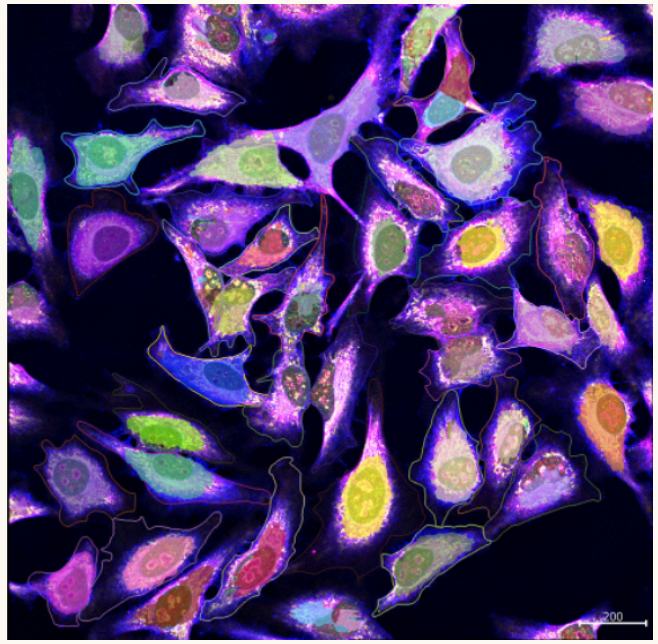
ViT L LM (zero shot)



Finetuned (ViT L LM)



Cellpose (zero shot)



Finetuning workflow step-by-step

Configuring your finetuning run.

1. Segment your training data and save images in one folder, masks in another.
2. Upload your training data to the EMBL cluster.
3. Modify the finetuning config to match your requirements (see following slides.)
4. Run the finetuning script.

```shell
sbatch path/to/run_finetune.sh
```

5. Download the trained model from the cluster.
6. Use the model for new segmentations.

Train your own model

How to finetune micro-sam on your own data.

```Shell
apptainer exec --nv --writable-tmpfs --bind /scratch:/scratch docker://timjhudelmaier/pixi-micro-sam-th:latest \
  /bin/bash -c "cd /repo && pixi run -e cuda python -m finetune \
  --config_path /path/to/the/default_config.json \
  --img_dir /path/to/image/files \
  --label_dir /path/to/labels \
  --output_dir /where/to/save/the/finetuned/model
  "
```

Notes

- your images and labels should have the same starting filename so they are matched correctly
- the config for the job scheduler works just as in the auto segmentation example

Oh no, another config file!

Configuring your finetuning run.

```JSON
"batch_size": 1,
"patch_shape": [300, 300],
"train_instance_segmentation": true,
"n_samples": 100,
"n_objects_per_batch": 5,
"n_epochs": 10,
"model_type": "vit_l_lm",
"checkpoint_name": "model_name",
"channels_of_interest": null,
"clahe": false,
"merge_channels": true,
"merge_method": "max"
```

- all upper config options are relevant to optimize model training and to fit the model on the GPU (no need to customize them)
- `model_type` is the model you want to use for training (recommendation: *vit_l_lm*)
- `channels_of_interest` can be used to use only a subset of all channels for training
- if true, `merge_channels` will combine the values of all channels using the method given in `merge_method` (options: mean (default), max, sum)

Big Thanks to Wouter-Michiel!