Ray-Sam

Group 3 ScaDaMaLe24

Jingyu Guo
Nils Mechtel
Songtao Cheng
Thanadol Sutantiwanichkul

github.com/thanadol-git/ray-sam

README MIT license

This is the project repository for **WASP Scalable Data Science and Distributed Machine Learning 2024 Group 3**.

In this project, we fine-tuned the **Segment Anything Model (SAM)**, an advanced vision foundation model developed by Meta AI. SAM is designed for promptable segmentation tasks and is a highly versatile tool for image segmentation across diverse domains. For more information, you can explore the SAM GitHub repository and the accompanying paper, "Segment Anything".

We built on the fine-tuning approach demonstrated by the micro SAM repository, which specializes in adapting SAM to fluorescence microscopy datasets. Our project introduces a novel contribution by leveraging the **Ray framework** to enable scalable, distributed training of SAM, making it suitable for handling large-scale microscopy datasets.

To achieve this, we utilized **Ray Train** and its `TorchTrainer` module. The `TorchTrainer` is a tool designed for data-parallel PyTorch training, automating the setup of distributed environments for scalable execution. It launches multiple workers as specified in the scaling configuration, establishes a distributed PyTorch environment for those workers, and seamlessly ingests input datasets. Each worker executes the user-defined `train_loop_per_worker` function, which contains the core training logic. This framework allowed us to scale SAM fine-tuning efficiently across multiple nodes, making it highly adaptable to large microscopy datasets.

This repository includes the code, configuration files, and documentation required to reproduce our results and experiment further with distributed fine-tuning of SAM.

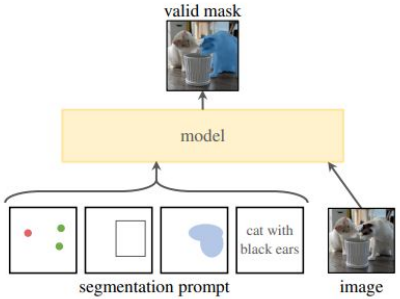
GitHub

Contents

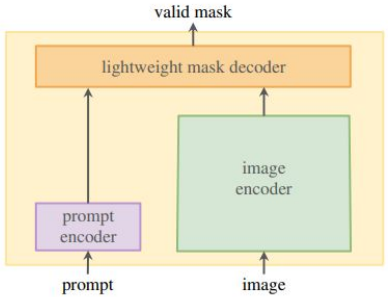
- SAM and micro-SAM
- Human Protein Atlas dataset
- Introduction to Ray
- Integration of Ray in our codebase
- Results
- Outlook

The Segment Anything Model (SAM)

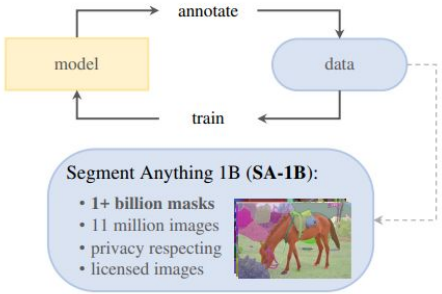
- Started by Meta
- Promptable
- Zero-shot generalization
- Foundation model for segmentation
- 1x10^9 masks on 11x10^6 images
- segment-anything.com



(a) **Task**: promptable segmentation



(b) **Model**: Segment Anything Model (**SAM**)



(c) **Data**: data engine (top) & dataset (bottom)

https://www.biorxiv.org/content/10.1101/2023.08.21.554208v1

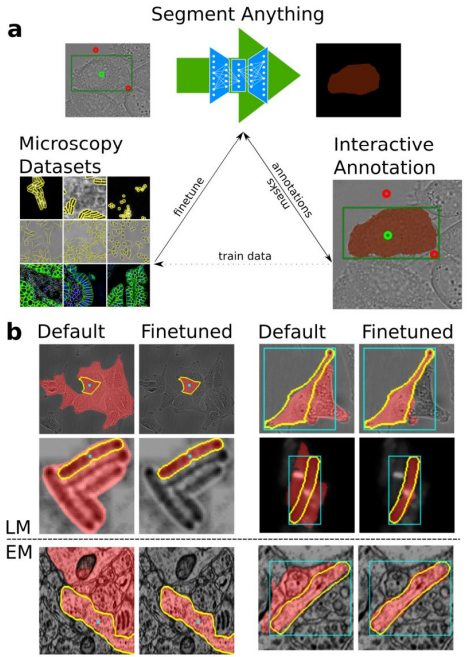
Microscopy images with SAM

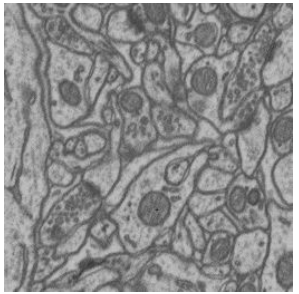
https://github.com/computational-cell-analytics/**micro-sam**

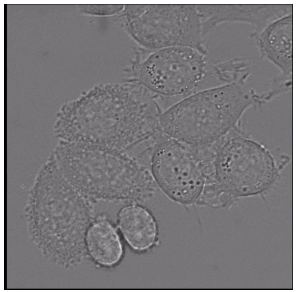
- Constantin Pape and team in Göttingen

Key features

- Performance improve from defaults
- Significantly improve even with new datasets
- EM and LM images
- Speed up annotation
- Versatile with several microscopy applications







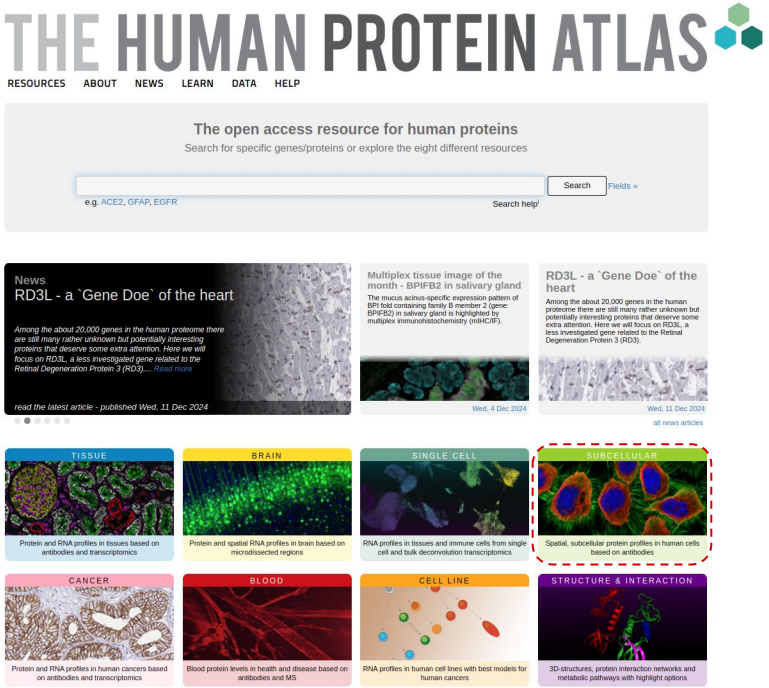
https://www.biorxiv.org/content/10.1101/2023.08.21.554208v1

The Human Protein Atlas (HPA)

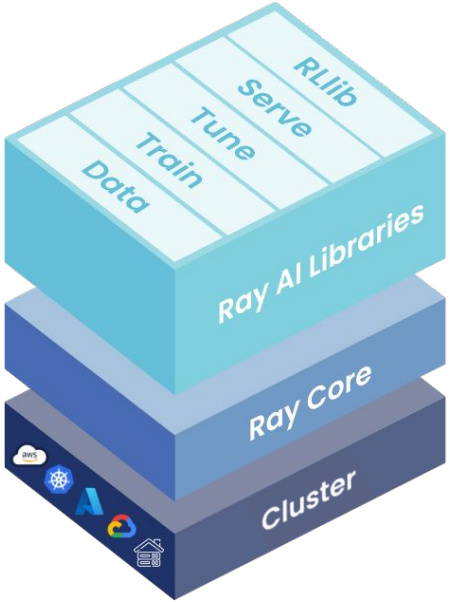
- Current version 24
- 2nd most visited biological database
- Global Core Biodata Resource
- Started in Stockholm and funded by KAW
- Initially based on antibody production technology
- Integrative high-throughput technologies
  - Sequencing: NGS/scRNA-seq
  - Precision medicine: MS/PEA
  - Imaging: IHC/IF

Dataset

- 50 cell-segmented images from human cell lines



RAY - An open-source framework for distributed computing



Overview_of_Ray.ipynb

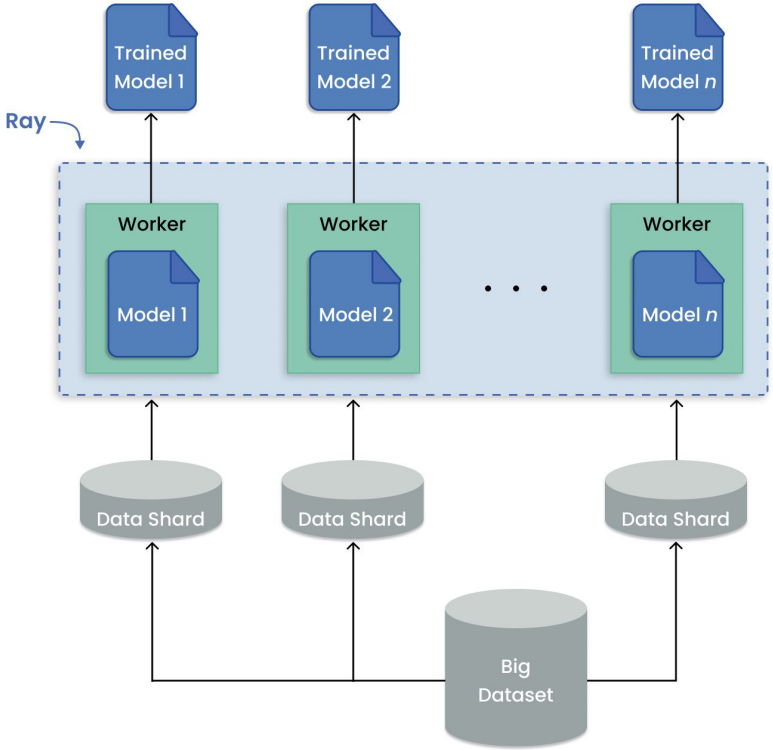
Ray AI Libraries

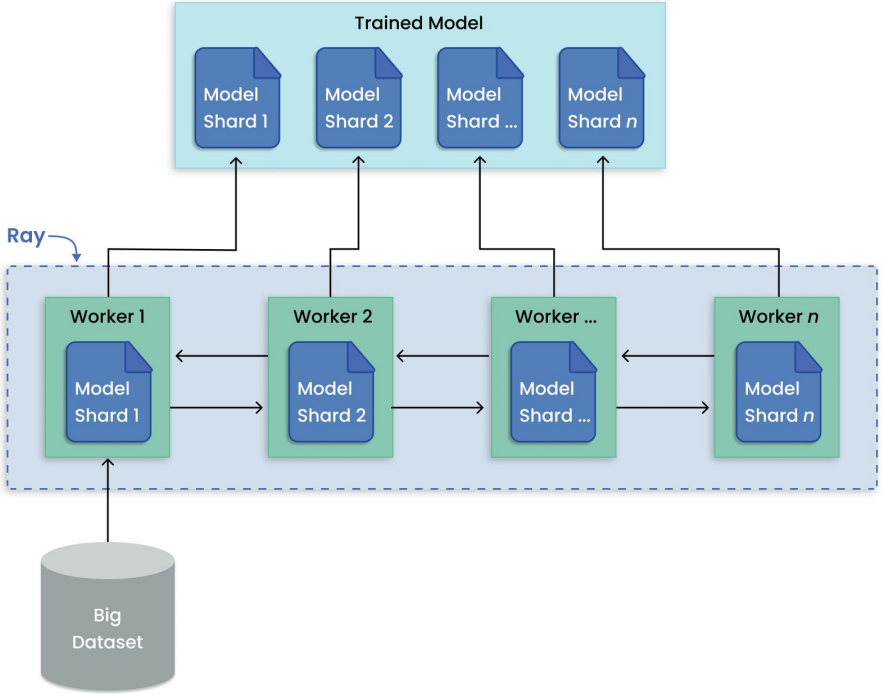
- Easy-to-use APIs for scaling machine learning workflows
- Scales from a single machine to large clusters
- Built-in fault tolerance to handle system failures

Ray Core

- Python-first API with a concise and intuitive design
- Supports low-level distributed task scheduling and execution
- Dynamic resource scaling for efficient utilization

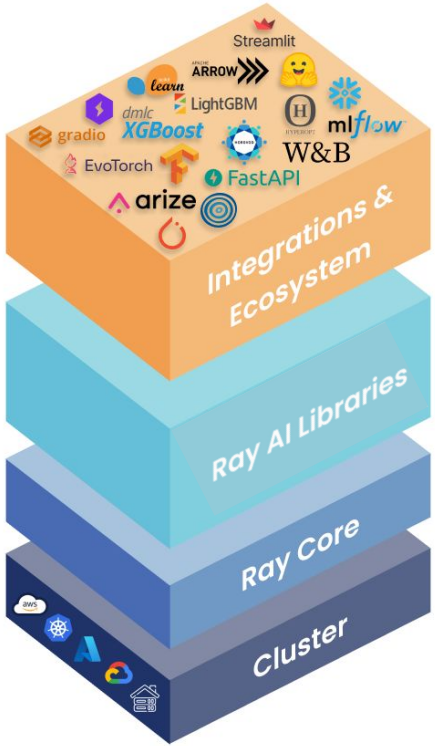
Ray supports both parallel and distributed model training





Overview_of_Ray.ipynb

Ray Ecosystem - A foundation for large-scale AI



a growing ecosystem of integrations and partnerships with popular projects

high-level libraries which enable simple scaling of AI workloads

a low-level distributed computing framework with a concise core, Python-first API

Growing ecosystem of integrations and partnerships:

- OpenAI: Trains its largest models, including ChatGPT, using Ray
- HuggingFace: Utilizes Ray Train for scaling model training efficiently
- Real-world and large-scale AI applications

Overview_of_Ray.ipynb

How to create a Ray cluster

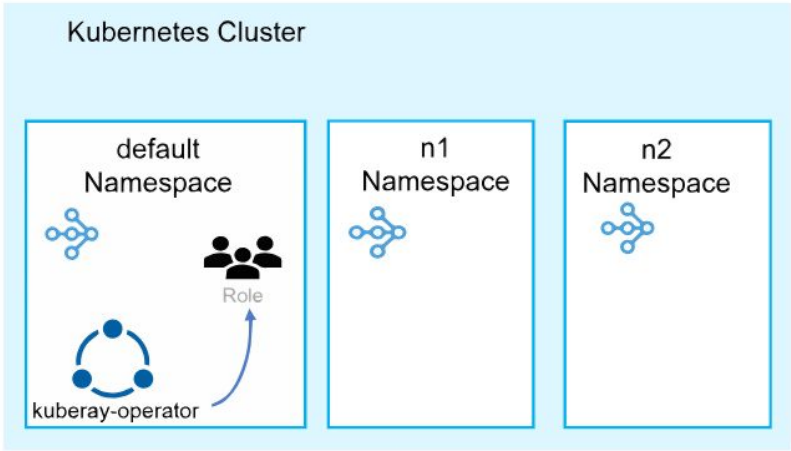
Ray Head Node

```
# Initialize Ray Head Node
import ray

ray.init()
```

Ray Worker Node

```
# Initialize Ray Worker Nodes
import ray

ray.init(address="auto", num_gpus=8, num_cpus=32)
```

Use Kubernetes to deploy Ray at scale



KubeRay: robust management of resources and scaling

docs.ray.io

Codebase Overview

- Core Components
  - Micro-SAM Segmentation Engine
  - Torch-em Integration
  - Scalable Fine-tuning Notebooks
- Key Features
  - Microscopy imaging adaptations
  - Environment-agnostic configuration
  - Ray-powered scalable training

RaySam

```
ray-sam
├── Dockerfile
├── create_conda_env.sh
├── demo-micro-sam
├── micro_sam_ray   # Ray implementation of SAM
│   ├── automatic_segmentation.py
│   ├── evaluation
│   ├── inference.py
│   ├── instance_segmentation.py
│   ├── models
│   ├── training
│   ...
├── notebooks       # Jupyter notebooks for scalable finetuning
│   ├── learn-ray.ipynb
│   └── sam_finetuning_ray.ipynb
├── run_sam_finetuning_hpa.py
└── torch_em        # torch-em dependency
    ├── data
    ├── model
    ├── trainer
    ...
```

Technical Architecture

- Segmentation Pipeline
  - Preprocessing: Includes modules for data loading, normalization, augmentation, and handling of medical image formats.
  - **Segmentation**: Utilizes the modified Micro-SAM for microscopy image segmentation.
  - Post-processing: Gathers segmentation results for visualization.
- Distributed Computing with Ray
  - **Task Distribution**: Ray enables efficient distribution of segmentation tasks across multiple nodes.
  - **Scalability**: Tested on local machines and cloud environments.
- Deployment Configurations
  - Environment Setup: Configurations for setting up distributed environments, including Docker and Kubernetes.
  - Resource Management: Optimized with Ray for resource allocation.

Outcomes

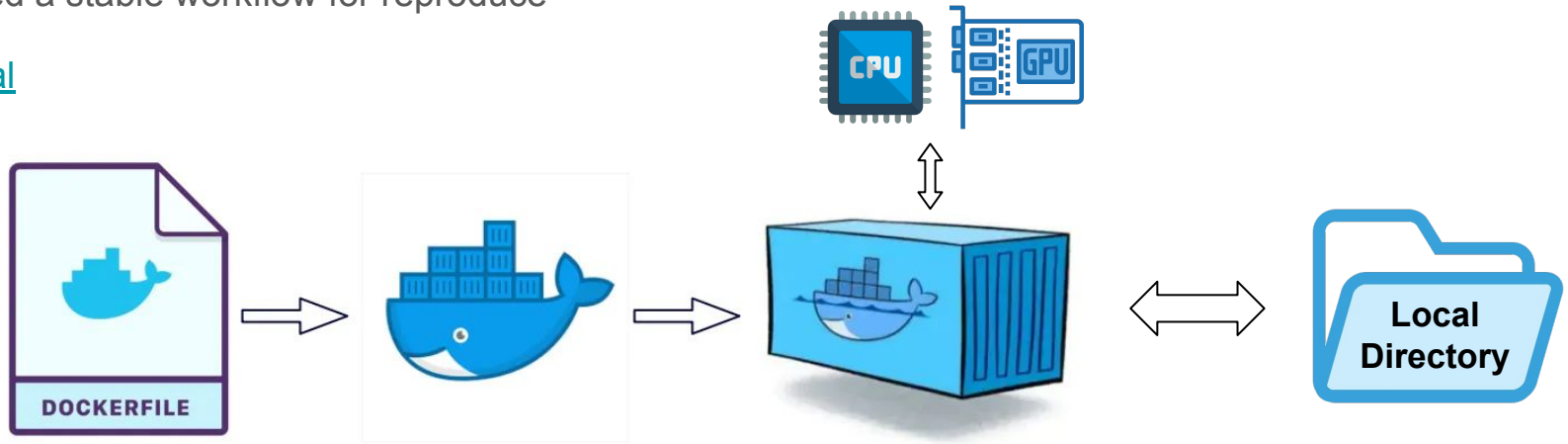
★ Benefits
- **Scalability**: Capable of handling growing data demands in deep learning and image processing.
- **Efficiency**: Reduces processing time for large datasets.
- *Accuracy: Can enhance segmentation performance through large-scale optimization.*

❏ To-dos
- ❏ Performance: Continuous improvement of the segmentation model.
- ❏ User Interface: Development of a user-friendly interface for users.
- ❏ Prettier Code: Refactor code for readability and maintainability.

Results

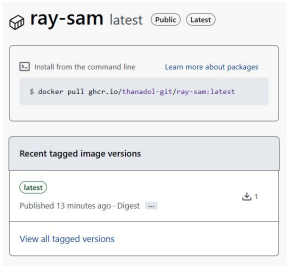
Created a stable workflow for reproduce

Tutorial



Docker File(link)

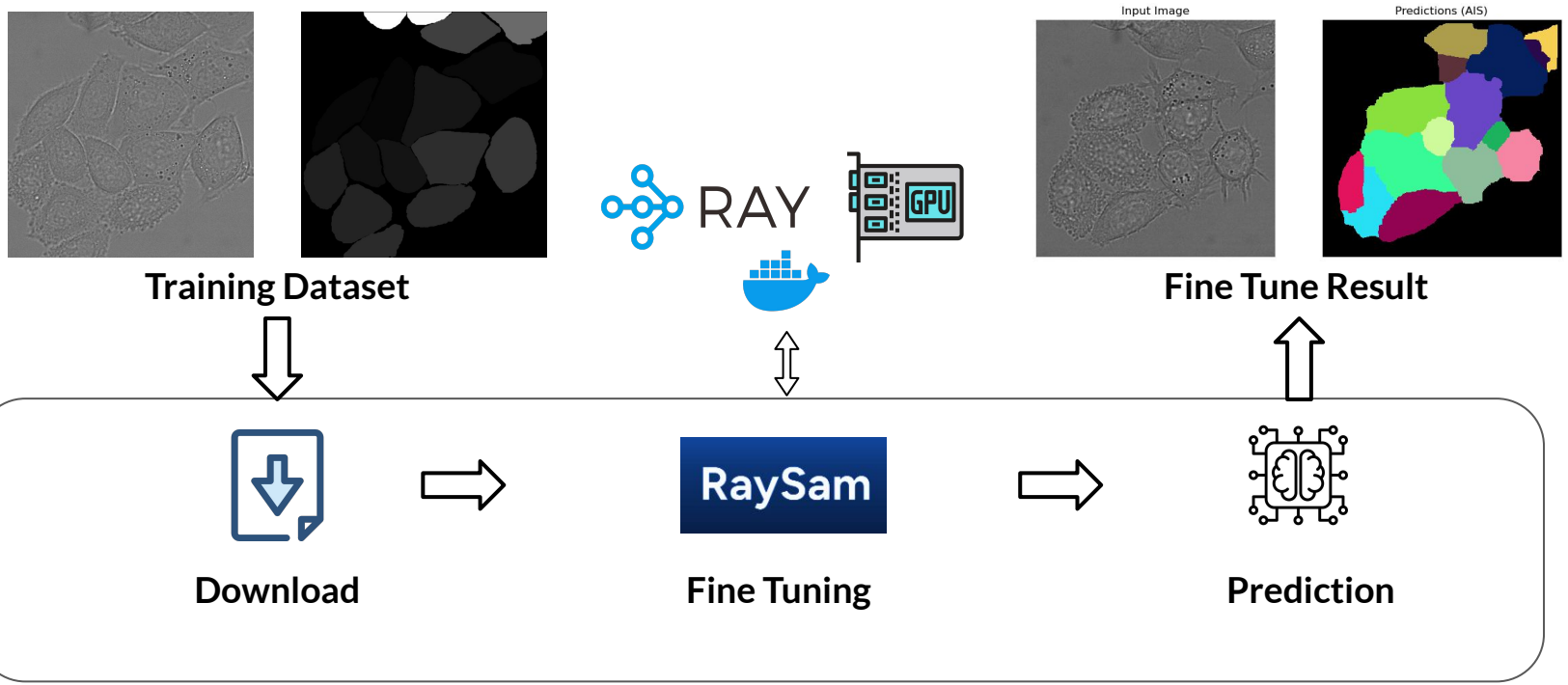
Docker Image(link)



Results

Test fine tuning based on micro_sam's example code

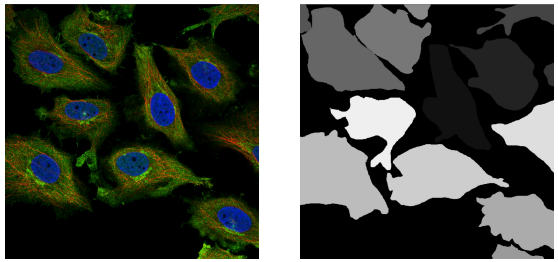
demo_sam_finetuning_ray.py



Results

Run our fine tuning with HPA image dataset

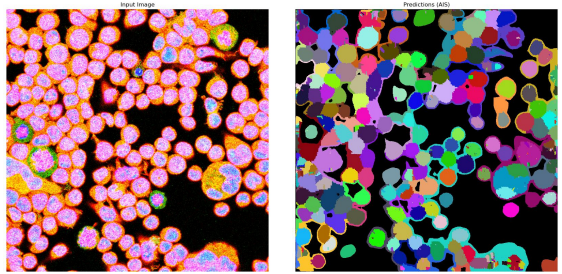
run_sam_finetuning_hpa.py



Training Dataset

RAY

GPU



Fine Tune Result

Download

RaySam

Fine Tuning

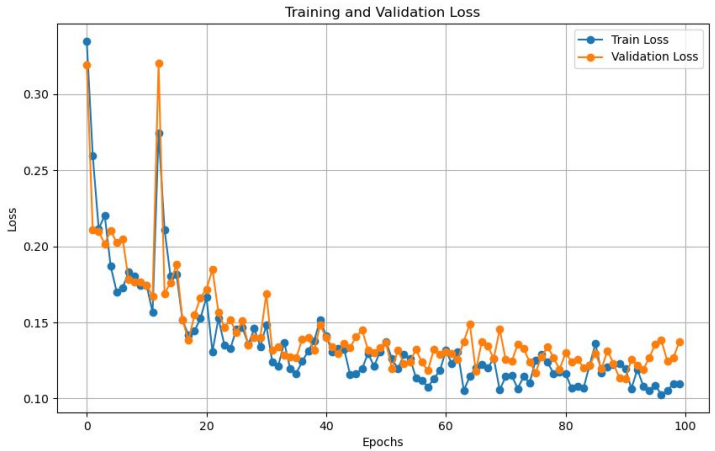
Prediction

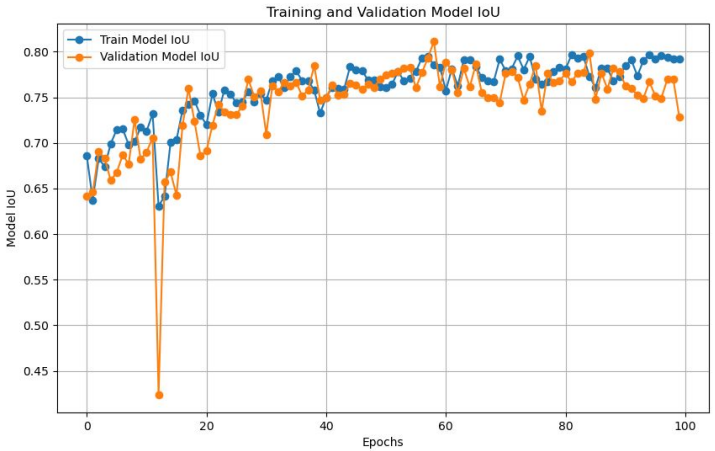
Results

Loss and IoU metrics during 100 epochs

Link to metrics plots, training logs and settings

Loss: mask loss + IoU regression loss





Summary

- Completed
  - Successfully modified the Python packages **‘micro_sam’** and **‘torch_em’** to make them compatible with Ray
  - Developed a model fine-tuning workflow using Docker
  - Conducted SAM model fine-tuning with HPA dataset on a local server
- Next Steps:
  - Execute SAM model fie-tuning on the Kubernetes cluster
  - Performance tuning
  - Iterative expansion of segmented Human Protein Atlas dataset
  - Analyze underlying biology of cell segmentations

Thanks for listening!