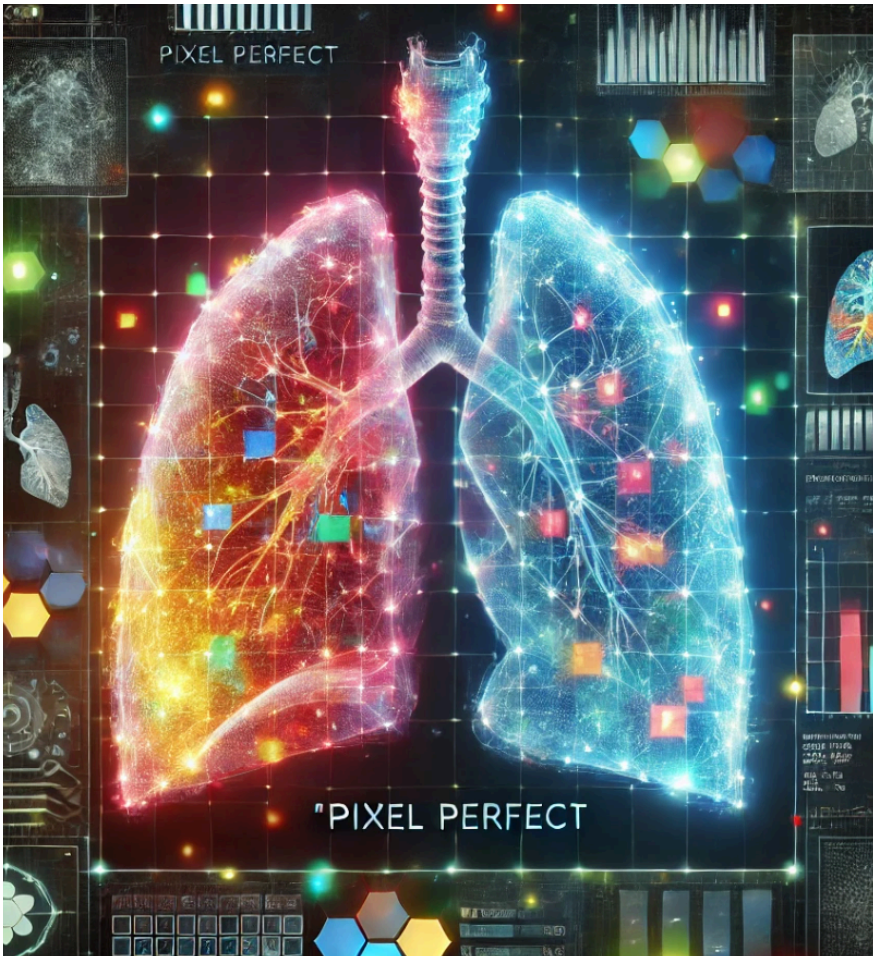


Pixel Perfect

Research Participants:
Isi Emordi
Arnav Jain
Rami Kamel
Kashir Khan
Andy Vayalali

Research Lead:
Ermina Ashraf

Faculty Advisor:
Ovidiu Daescu

Overview

Segmentation models are known for their effectiveness with clear and distinct images. However, the Segment Anything Model (SAM) claims to maintain high accuracy even with ambiguous objects or overlapping features. Our project, Pixel Perfect, aims to put this claim to the test by focusing on the segmentation of pneumonia-affected regions in lung X-ray images. We will train and evaluate a model based on SAM, applying advanced image preprocessing techniques to enhance its performance in segmenting pneumonia. By refining images before analysis, we aim to improve the delineation of pneumonia-affected areas, advancing the capabilities of current segmentation models and pushing the boundaries of segmentation in the medical realm.

Our Purpose: Our main objective is to ***utilize image preprocessing techniques like edge detection to accurately segment pneumonia affected regions in lung x-rays.*** We hope to achieve this in order to answer the following question:

"How effectively can current segmentation models transfer their learning to datasets not seen during training, and can we further increase the efficiency of SAM with various image preprocessing techniques?"

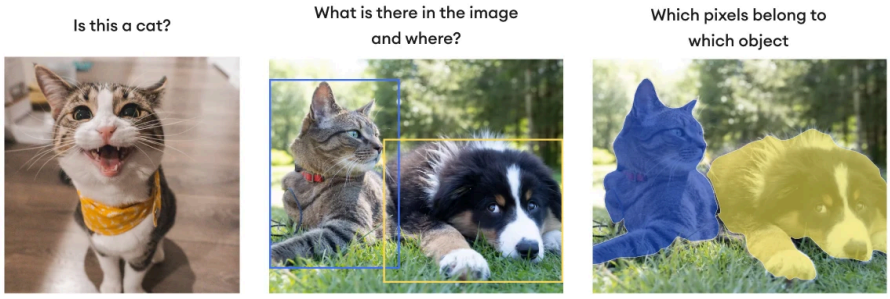
Overall, Pixel Perfect aims to *Push the boundaries of medical image segmentation technology*: By utilizing various image preprocessing techniques and developing advanced algorithms and techniques for precise delineation, we aim to advance the state-of-the-art in medical image analysis

Why Pneumonia?: Pneumonia was chosen to be the focus of this project not because it lacks existing solutions or is groundbreaking in itself, but rather for its unique characteristics in medical imaging that makes it an ideal candidate for advancing segmentation techniques.

Pneumonia's ambiguity in chest x-rays provide an excellent opportunity to push the boundaries of image segmentation algorithms and test SAM's claim of high accuracy with challenging, less-defined features. Not only that but our focus also offers a rich testing ground for developing and refining advanced segmentation techniques

Image Segmentation

A computer vision technique that refers to the process of masking or highlighting regions of interest within an image.



In the image above, the region of interest is located within the bounded boxes

How Does it Work?

Image segmentation consists of 3 primary steps:

1. Feature Extraction: The first step involves extracting meaningful features from the image. These features can include color, texture, edges, or shape information.
2. Grouping: Once features are extracted, they are grouped together based on similarity. This can be done using techniques like clustering or thresholding.
3. Boundary Refinement: The boundaries of the segmented regions are often refined to ensure they accurately represent the objects in the image. This might involve smoothing edges or removing small, noisy regions.

Types of Segmentation?

There are various ways to perform image segmentation. Here are a few common techniques:

1. Thresholding: Method that divides pixels into 2 groups based on their intensity values.
2. Edge Detection: Identifies edges in an image to delineate object from background
3. Region-Based Segmentation: Groups pixels based on similarity (color, texture, etc.)
4. Active Contours: deformable models that evolve to fit boundaries of an object in image.
5. *Deep Learning-Based Segmentation*: Modern techniques leverage deep neural networks, such as U-Net and Mask R-CNN, to achieve state-of-the-art performance in segmentation tasks.

Resources: Segmentation

Intro to Image Segmentation:
https://www.analyticsvidhya.com/blog/2022/05/introduction-to-image-segmentation/

Segment Anything Model (SAM)

The Segment Anything Model (SAM) is a groundbreaking pre-trained model designed for versatile segmentation tasks. Developed and introduced by Meta AI in April 2023, SAM represents a significant leap forward in the realm of computer vision.

Benefits of a pre-trained model:

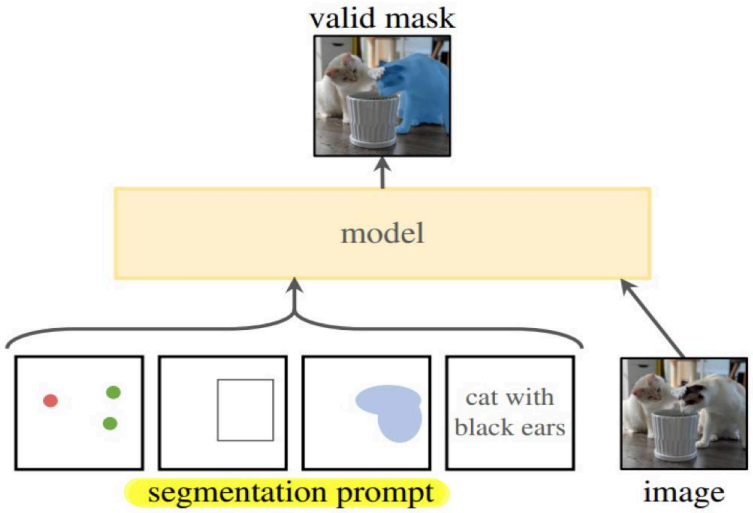
1. Faster training: Pre-trained models can be fine-tuned on smaller datasets much more quickly than training a model from scratch.
2. Improved performance: Pre-trained models often achieve better performance than models trained from scratch, especially when dealing with limited data.
3. Reduced computational resources: Using a pre-trained model can significantly reduce the computational resources required for training.

Key features and capabilities

SAM was mainly developed to create a flexible and powerful segmentation tool that could adapt to a wide range of tasks without the need for task-specific fine-tuning.

It's key features include:

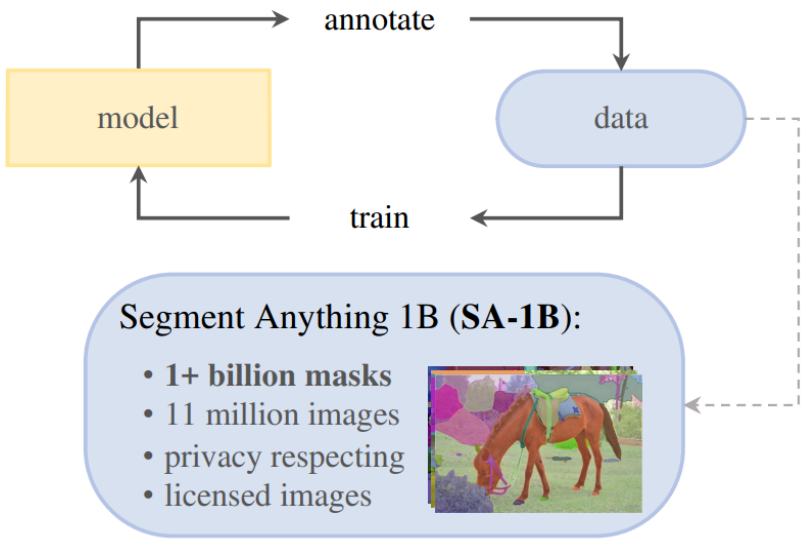
1. Prompt-based segmentation: The SAM model is designed to support a wide range of inputs from spatial to text.



2. Zero-shot learning capabilities: This refers to the ability of a model to accurately classify and recognize images that it has not been trained on. This is possible through SAM's generalization ability which means that the model is expected to recognize features and concepts learned from seen classes and apply this understanding to identify or classify unseen classes.

3. Ability to segment any object in an image: This is one of SAM's most defining features. Its ability to segment any object in an image makes it an extremely flexible and adaptable tool for image segmentation tasks.

4. Diverse dataset of 11 million images and 1.1 billion masks: SAM's extensive training allows the model to learn a wide range of visual patterns, shapes, textures, and object boundaries.

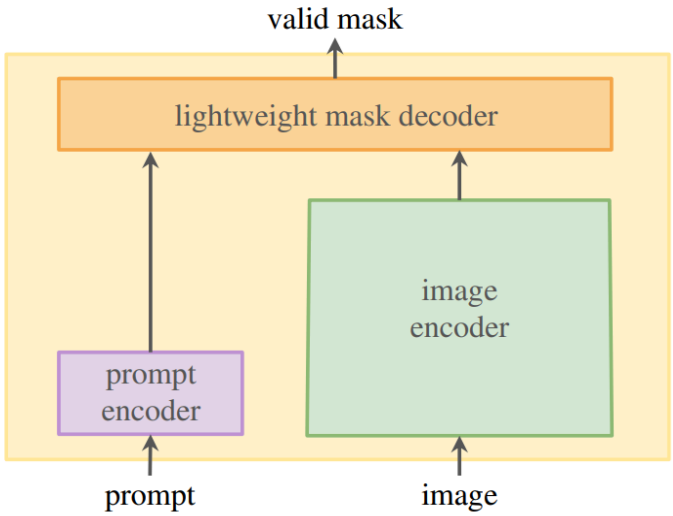


Include description of what's going on in image

Architecture overview

The SAM model consists of a Transformer-based architecture. A transformer model is a neural network that learns context by tracking relationships in sequential data like the words in this sentence.

SAM consists of 3 components: A Vision Transformer (ViT or image encoder), prompt encoder, and mask decoder.



1. Image Encoder: A component of a model that processes an input image to extract meaningful features. In the Segment Anything Model (SAM), the image encoder employs a Vision Transformer (ViT) pre-trained with Masked Autoencoders (MAE). This configuration enables the model to:
    a. Process high-resolution images: Handle images of larger sizes effectively.
    b. Run once per image: Create an image embedding that can be used for multiple prompts, reducing computational overhead.
    c. Be applied prior to prompting: Allow for efficient segmentation by pre-processing the image before user interaction.

2. Prompt Encoder: component of a model that processes and translates different types of input prompts to align with the input image in order to perform tasks like object segmentation. The prompt encoder in SAM can handle multiple types of prompts (text, points, boxes), provided positional encodings for points and boxes, encodes text input using a CLIP text encoder, and processes mask inputs by applying filters that capture the relationship and patterns within the mask

3. Mask Decoder: component of a model that takes image embeddings, prompt embedding, and output token to produce a mask that isolates the region of interest within an image. The SAM model uses a modified transformer decoder block which is used to process combined information from image embedding and prompt embeddings, utilizes prompt self-attention and cross-attention to determine how the prompts relate to the image and how the image relates to the prompts, increases the resolution to match the resolution required for output mask, uses a multilayer perceptron to map the output token to a dynamic linear classifier and computes mask foreground probability which generates the probability for each pixel in the image being part of the area of interest or not,

Additional Components of SAM

Listed below are more key components of SAM:

1. Pre-training algorithm: The algorithm used to train the model on a large dataset before fine-tuning for specific tasks. SAM's ViT is pre-trained on an image classification dataset to help the model learn general features that are useful for segmentation

2. Loss functions: SAM uses a combination of focal loss and dice loss to balance the importance of classification accuracy and segmentation quality.
    a. *Focal Loss*: A loss function that focuses on correcting hard examples (misclassified samples) more heavily than easy examples.
    b. *Dice Loss*: A loss function that measures the overlap between predicted and ground truth segmentation masks

3. Handling vague prompts: Sam utilizes techniques like prompt flexibility, contrastive learning, attention mechanisms, and transfer learning to handle ambiguous prompts

4. Model scaling: For improved performance, SAM can be scaled using 3 distinct model variants each with different level of capacity
    a. Base Variant (ViT-B): The foundational model configuration that provides a balance between performance and computational requirements. Ideal for standard tasks and datasets.
    b. Large Variant (ViT-L): A more advanced configuration with increased depth and width, allowing the model to handle more complex features and higher-resolution inputs. Suitable for tasks requiring enhanced detail and accuracy.

c. Huge Variant (ViT-H): The most powerful configuration with the highest capacity, designed for the most demanding tasks and large-scale datasets. Provides the greatest performance but requires substantial computational resources.

5. Efficiency considerations: SAM utilizes efficiency considerations to optimize training and inference. For example, SAM benefits from mixed precision training which helps to speed up training process and reduce memory usage

SAM's limitations and considerations

1. Fine Structure: SAM may miss fine-grained details in images.
2. Hallucinations: SAM can generate small, disconnected components that aren't present in the original image.
3. Boundary Crispness: SAM's boundaries might not be as sharp as those produced by more computationally intensive methods.
4. Interactive Segmentation: SAM's performance might be outperformed by dedicated interactive methods when many points are provided.
5. Real-time Performance: SAM's overall performance can be slow when using a heavy image encoder.
6. Domain-Specific Tools: SAM might be outperformed by domain-specific tools in certain applications.

Resources: SAM

SAM by Meta AI (paper): https://arxiv.org/pdf/2304.02643

Transformer Models (article): https://blogs.nvidia.com/blog/what-is-a-transformer-model/

About SAM Architecture, Dataset, Training: https://youtu.be/OhxJkqD1vuE?si=RZpSlkSeVB4iit27

Implement SAM (walkthrough video): https://youtu.be/D-D6ZmadzPE?si=ZOWOV9k8ik-0vVnd

Dataset

SA-1B

The SA-1B (Segment Anything 1 Billion) dataset, created by Meta AI, is a massive collection designed for training SAM. It contains over 1 billion masks from 11 million licensed and privacy-preserving images, making it one of the largest segmentation datasets ever released. The dataset's strength lies in its scale and diversity, covering a wide range of object categories and scenes. SA-1B was created using a novel semi-automated process involving an auto-annotator model and human verification, resulting in high-quality segmentation masks rather than just bounding boxes.

1. Assisted-manual: Annotators used a SAM-powered browser tool to label masks.
2. Semi-automatic: Annotators labeled additional objects in images pre-filled with automatically detected masks.
3. Fully automatic: Fully automatic mask generation using the ambiguity-aware model.

		SA-1B		% images		
	# countries	#imgs	#masks	SA-1B	COCO	O.I.
Africa	54	300k	28M	2.8%	3.0%	1.7%
Asia & Oceania	70	3.9M	423M	36.2%	11.4%	14.3%
Europe	47	5.4M	540M	49.8%	34.2%	36.2%
Latin America & Carib.	42	380k	36M	3.5%	3.1%	5.0%
North America	4	830k	80M	7.7%	48.3%	42.8%
high income countries	81	5.8M	598M	54.0%	89.1%	87.5%
middle income countries	108	4.9M	499M	45.0%	10.5%	12.0%
low income countries	28	100k	9.4M	0.9%	0.4%	0.5%

The data is categorized by geographic regions and country income levels, revealing the dataset's global scope but also highlighting significant disparities in representation.

NIH Chest X-ray Dataset

The NIH dataset is known to be the most publicly available dataset in the domain of chest x-rays. It contains 112,120 frontal-view chest x-ray images from 30,805 patients where each image consists of data including age, gender, and additional information. The images are labeled with common thoracic diseases including: Atelectasis, cardiomegaly, effusion, infiltration, mass, nodule, pneumonia, pneumothorax, consolidation, edema, emphysema, fibrosis, pleural thickening, and hernia. Each image is in PNG format and comes with a corresponding CSV file that provides labels and associated patient data.

RSNA Pneumonia Detection Challenge Dataset

The RSNA dataset is a subset of the NIH Chest X-ray dataset designed specifically for detecting pneumonia. The RSNA dataset contains 30,000 chest x-ray images selected from the NIH Dataset and consists of bounding box annotations which the NIH dataset lacks. These bounding boxes indicate the locations of lung opacities that are indicative of pneumonia. The bounding boxes are provided in the form of x and y coordinates and some images even come with multiple bounding boxes to illustrate multiple regions of opacities. The RSNA dataset includes labels for x-rays with pneumonia affected regions, labels for healthy lungs, and labels for x-rays that consist of a different abnormality than pneumonia (no bounding box). The images come in DICOM format, and the provided annotations come in a CSV file format with the bounding box coordinates and associated labels

Our Dataset for Fine-Tuning

For segmentation purposes, we will use the RSNA dataset as the bounding boxes as provided in the dataset serve as valuable starting point segmentation tasks. While bounding boxes may not provide the most accurate results, they will however, provide a general localization of pneumonia-affected regions that can then be refined with various image preprocessing techniques

Model Training

Install Libraries & Tools

Set up environment
Set up GitHub
Load Pre-Trained Model
Install Python Libraries

Data Collection

We already have our dataset!

Data Preparation

Data Augmentation: Artificially increases the size and diversity of a dataset to improve model robustness and prevent overfitting.
https://www.ibm.com/topics/data-augmentation

Normalization: Scales numerical data to a specific range (e.g., 0-1) to improve model performance.
https://www.geeksforgeeks.org/what-is-data-normalization/

Data Splitting: divide test, train, and validation set

Develop Baseline

Data Preprocessing: Prepares data by cleaning, transforming, and formatting it.

Model Selection

Choose SAM variant (ViT-B, ViT-L, ViT-H)

Prepare Data Loader

Create Data Loaders
Define Batch Sizes
Implement Data Augmentation

Define Hyperparameters

Set Learning Rate
Determine Epochs

Select Optimization Algorithm
Configure Loss Function

Fine-Tune Model

Train!
Implement Checkpoints

Monitor Training Progress

Track Metrics
Visualize Training
Adjust Training

Evaluate Model

Test on Validation Set
Measure Performance
Analyze Errors
Continue Fine-Tuning if Time Permits

Image Preprocessing Techniques

Listed are a few (not limited to these) Image Preprocessing techniques we will review

Sobel Operator

Edge Detection: The Sobel Operator is used to detect edges in images by calculating the gradient of image intensity at each pixel. It highlights areas of high spatial frequency where edges are located.

Feature Extraction: Helps in enhancing edge features that can be critical for segmentation models like SAM to accurately delineate boundaries of objects or regions.

OpenCV: Provides functions like cv2.Sobel to apply the Sobel operator.

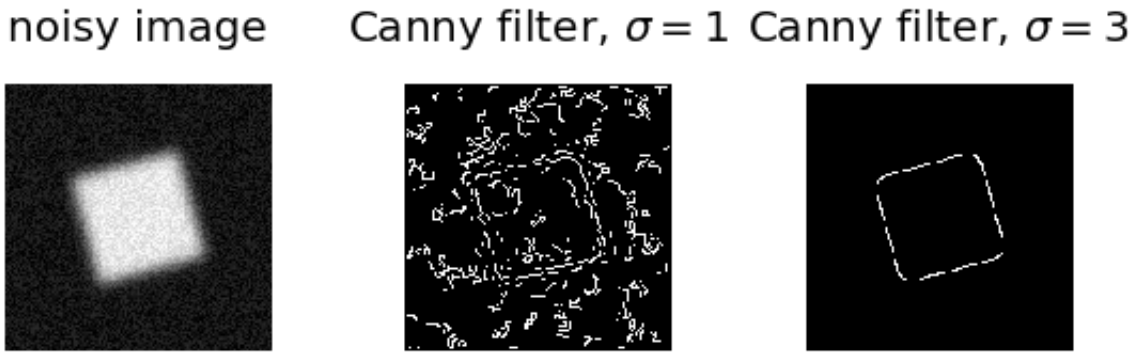
Canny Edge Detection

Edge Detection: The Canny Edge Detector is a multi-stage algorithm that detects a wide range of edges in images. It performs Gaussian smoothing, gradient calculation, non-maximum suppression, and edge tracking by hysteresis.

Precise Edges: Provides more accurate and refined edges compared to simpler methods, which helps SAM in identifying and segmenting detailed structures in images.

OpenCV: Provides a straightforward implementation through cv2.Canny.

Scikit-Image: Also includes edge detection methods.



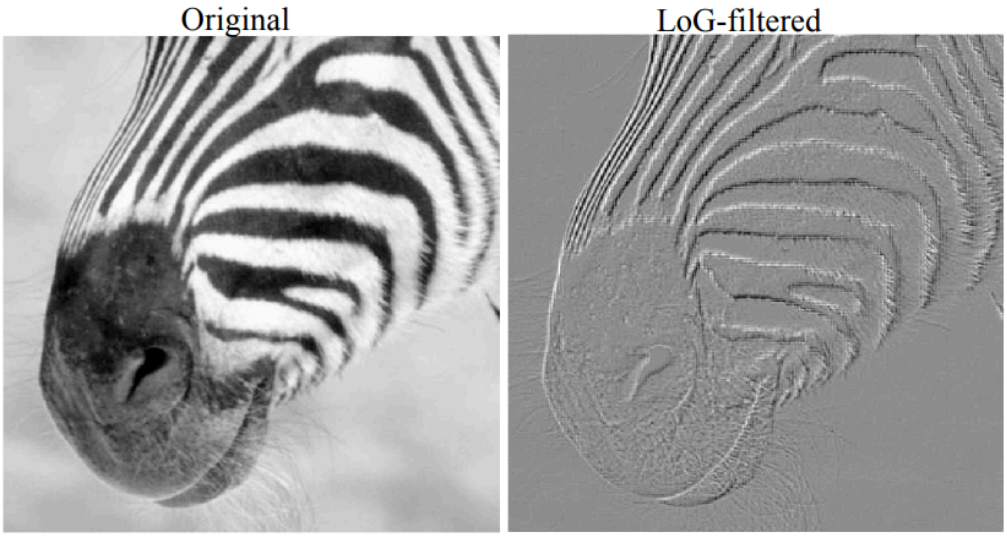
Laplacian of Gaussian (LoG)

Edge Detection and Blurring: The Laplacian of Gaussian combines Gaussian smoothing with Laplacian edge detection to identify edges by calculating the second derivative of the image. It enhances edge detection by reducing noise.

Enhanced Segmentation: Helps in smoothing the image and highlighting edges, which can improve the segmentation accuracy of SAM by providing clearer boundaries.

OpenCV: Can be used for applying Gaussian blur and Laplacian separately.

SciPy: Provides scipy.ndimage.gaussian_laplace for LoG implementation.



Python Libraries

Listed below are the main libraries we'll be using for our research. This list is subject to change

PyTorch

Purpose: PyTorch is a deep learning framework that provides tools to build, train, and fine-tune neural networks. It is highly flexible and allows for dynamic computational graphs, making it ideal for research and experimentation.

Use for Fine-Tuning SAM: It serves as the backbone for implementing and fine-tuning SAM, allowing you to define custom neural network layers, optimization algorithms, and training loops tailored to segment pneumonia-affected regions in lung X-rays.

NumPy

Purpose: NumPy is a library for numerical computing in Python, offering support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

Use for Fine-Tuning SAM: It is used for efficient data manipulation, handling image arrays, and performing mathematical operations needed for data preprocessing and augmentation in the SAM model pipeline.

Matplotlib

Purpose: Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. It is widely used for data visualization in scientific computing.

Use for Fine-Tuning SAM: It helps visualize the results of segmentation, plot model training metrics, and compare model outputs with ground truth masks, facilitating a better understanding of model performance.

Scikit-Learn

Purpose: A machine learning library that provides simple and efficient tools for predictive data analysis, including classification, regression, clustering, and evaluation metrics.

Use for Fine-Tuning SAM: It can be used for evaluating model performance with metrics like precision, recall, and F1-score, as well as for splitting datasets and performing cross-validation.

MONAI (Medical Open Network for AI)

Purpose: A specialized deep learning framework built on PyTorch, tailored for healthcare imaging tasks. It provides domain-specific tools for medical imaging research, including data handling, augmentation, and evaluation.

Use for Fine-Tuning SAM: MONAI provides specialized pre-processing techniques, medical image augmentations, and loss functions that help in enhancing SAM's ability to handle medical images, such as X-rays, more effectively.

Pandas

Purpose: Pandas is a data manipulation library that provides data structures like DataFrames for efficiently managing and analyzing structured data.

Use for Fine-Tuning SAM: It is used to manage and preprocess metadata, such as patient information and image properties, which is crucial for organizing datasets and preparing them for model training and evaluation.

OpenCV (Open Source Computer Vision Library)

Purpose: Library focused on computer vision, providing a wide range of tools for image processing, computer vision algorithms, and machine learning.

Use for Fine-Tuning SAM: It is useful for advanced image processing tasks such as filtering, edge detection, custom augmentation, and ROI extraction, helping to enhance input images before segmentation by SAM.

pyDICOM

Purpose: Library used for reading, modifying, and writing DICOM (Digital Imaging and Communications in Medicine) files, which are the standard format for medical imaging data, such as X-rays, MRIs, and CT scans.

Use for Fine-Tuning SAM: It is essential for handling DICOM files in the RSNA dataset, allowing you to read, preprocess, and extract relevant metadata from X-ray images before feeding them into the SAM model. This ensures that the model is trained on properly formatted and contextually accurate medical data, which is crucial for precise segmentation.

Development tools

The following are some additional tools we will use:

Conda

Purpose: Environment Management

Role: Conda is used to create and manage isolated environments for your project. It ensures that you have the right versions of libraries and dependencies required for your project, without conflicts with other projects.

Tasks:

1. **Create and manage environments:** Use Conda to create a dedicated environment for your project.
2. **Install dependencies:** Install the necessary libraries (e.g., PyTorch, NumPy, OpenCV) into the environment.
3. **Switch between environments:** Easily switch between different project environments.

Kaggle Notebooks

Purpose: Interactive Coding and Experimentation

Role: Kaggle Notebooks offer a cloud-based interactive environment where you can write, run, and test your code. They are excellent for developing models, performing data analysis, and visualizing results without needing local resources.

Tasks:

1. **Code Development:** Write and execute code, perform data preprocessing, and train models directly in the notebook.
2. **Experimentation:** Test and refine different models and preprocessing techniques interactively.
3. **Documentation:** Document your experiments, results, and code with markdown cells for clarity and reproducibility.

GitHub

Purpose: Collaboration and Version Control

Role: GitHub is used for managing your project's code, collaborating with team members, and tracking changes over time. It helps with version control and integrates with tools like Kaggle Notebooks for collaborative coding.

Tasks:

1. Version control: Track changes to your codebase and manage different versions.
2. Collaboration: Share your code with team members, review each other's work, and merge changes.
3. Documentation: Use the repository to store project documentation, like README files, and experiment results.

DeepNote

Purpose: Collaborative Data Science Notebooks

Role: DeepNote provides an online platform for collaborative data science work. It allows you to write, run, and share code in an interactive notebook format, making it ideal for teamwork and real-time collaboration on data analysis and modeling tasks.

Tasks:

1. Code Development: Write and execute code in an interactive environment. DeepNote supports Python, SQL, and other languages commonly used in data science.
2. Collaboration: Work simultaneously with team members on the same notebook, with real-time updates and communication tools to enhance teamwork.
3. Data Analysis and Visualization**:** Perform data analysis, create visualizations, and document findings within the notebook, integrating seamlessly with various data sources and tools.

Schedule

Weekly Objectives

Week 1 (9/9): Finish reading doc, play around with SAM and bring something to next build night to show, do some research on your own time over data preparation

Week 2 (9/16): Install Libraries, get GitHub and env set up. Implement SAM using RSNA (no training, just implementation), self-study on data preparation

Week 3 (9/23):

Week 4 (9/30):

Week 5 (10/7):

Week 6 (10/14):

Week 7 (10/21):

Week 8 (10/28):

Week 9 (11/4):

Week 10 (11/11):

Build Night Objectives

Build Night #1 (9/9): Intros, Icebreakers, brief overview, google form

Build Night #2 (9/16): Round table discussion, review concepts, define week 2 objectives

Build Night #3 (9/23):

Build Night #4 (9/30):

Build Night #5 (10/7):

Build Night #6 (10/14):

Build Night #7 (10/21):

Build Night #8 (10/28):

Build Night #9 (11/4):

Build Night #10 (11/11):

Important Deadlines

Sunday, October 13th: Mid Semester Submission Due

Wednesday, November 13th: Overleaf Posters Due

Sunday, November 17th: GitHub Repository Due

Monday, November 18th: Mock Presentations

Thursday, November 21st: Research Symposium

Challenges

1. Initially, we will evaluate the pre-trained SAM model without fine-tuning by directly applying it to our dataset. We will use the provided bounding box annotations as prompts to generate segmentation masks. Due to the lack of precise segmentation masks, we may not be able to quantify the model's accuracy with numerical metrics.

2. Due to the absence of a dataset containing precise mask annotations for regions of interest, our evaluation may be limited in accuracy. While bounding box annotations provide a general approximation, they may not capture the fine-grained details necessary for a comprehensive assessment of the model's performance.

Important Links

RSNA Dataset
https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data

Github to SAM
https://github.com/facebookresearch/segment-anything

Download Conda
https://www.youtube.com/watch?v=d_jBX7OrptI&t=34s

Dictionary

All Resources

Intro to Image Segmentation:
https://www.analyticsvidhya.com/blog/2022/05/introduction-to-image-segmentation/

SAM by Meta AI (paper): https://arxiv.org/pdf/2304.02643

Transformer Models (article): https://blogs.nvidia.com/blog/what-is-a-transformer-model/

About SAM Architecture, Dataset, Training: https://youtu.be/OhxJkqD1vuE?si=RZpSlkSeVB4iit27

Implement SAM (walkthrough video): https://youtu.be/D-D6ZmadzPE?si=ZOWOV9k8ik-0vVnd

Data Augmentation: https://www.ibm.com/topics/data-augmentation

Data Normalization: https://www.geeksforgeeks.org/what-is-data-normalization/

!Required Reads!

Intro to Image Segmentation:
https://www.analyticsvidhya.com/blog/2022/05/introduction-to-image-segmentation/

SAM by Meta AI (paper): https://arxiv.org/pdf/2304.02643