Federal University of Juiz de Fora
Institute of Exact Sciences
Bachelor's Degree in Computer Science

Development of an Artificial Intelligence-Aided Software for Annotating Image Datasets

Paulo Victor de Magalhães Rozatto

JUIZ DE FORA
SEPTEMBER, 2024

Development of an Artificial Intelligence-Aided Software for Annotating Image Datasets

Paulo Victor de Magalhães Rozatto

Federal University of Juiz de Fora

Institute of Exact Sciences

Department of Computer Science

Bachelor's Degree in Computer Science

Advisor: Luiz Maurílio da Silva Maciel

JUIZ DE FORA

SEPTEMBER, 2024

Development of an Artificial Intelligence-Aided Software for Annotating Image Datasets

Paulo Victor de Magalhães Rozatto

MONOGRAPH SUBMITTED TO THE FACULTY OF THE INSTITUTE OF EXACT SCIENCES OF THE FEDERAL UNIVERSITY OF JUIZ DE FORA, AS AN INTEGRAL PART OF THE NECESSARY REQUIREMENTS TO OBTAIN A BACHELOR'S DEGREE IN COMPUTER SCIENCE.

Accepted by:

Luiz Maurílio da Silva Maciel
D.Sc. in Systems Engineering and Computer Science

Marcelo Bernardes Vieira
D.Sc. in Image and Signal Processing Sciences

Saulo Moraes Villela
D.Sc. in Systems Engineering and Computer Science

JUIZ DE FORA
SEPTEMBER 10, 2024

Abstract

Deep learning is a highly successful class of methods in the artificial intelligence (AI) field that has a variety of applications. To perform well, deep learning models require a large amount of high-quality annotated data. Data annotation is a time-consuming and laborious task that requires a significant amount of human labor, which makes it expensive. This work aims to reduce the time required to annotate image datasets by building an easy-to-use software tool that has semi-automated annotation powered by an artificial intelligence model. We developed a web-based tool and employed HQ-SAM, a deep neural network for image segmentation based on Vision Transformers, to generate polygon annotations based on the user's prompts. Although HQ-SAM has a good zero-shot generalizability, we fine-tuned it on the Bean Leaf Dataset to evaluate how well the network adapts to specific tasks. We observed an increase in accuracy of the fine-tuned model in comparison with the pre-trained one. We tested our tool with 20 participants, all of whom are from the computer vision and graphics fields. We asked them to annotate the same two images both manually and AI-aided, and we recorded the annotation times. Lastly, we asked the participants to fill out a usability form about their user experience. In our evaluation, we registered a median speedup of 1.5× regarding the AI-aided annotation compared to manual annotation and overly positive answers regarding our tool's ease of use and usefulness.

Keywords: Deep learning, web application, semi-automated annotation, image datasets, vision transformers.

Resumo

A aprendizagem profunda é uma classe de métodos muito bem-sucedida no campo da inteligência artificial (IA), possuindo uma ampla gama de aplicações. Para ter um bom desempenho, os modelos de aprendizagem profunda exigem uma grande quantidade de dados anotados de alta qualidade. A anotação de dados é uma tarefa demorada, trabalhosa e que requer uma quantidade significativa de esforço humano, o que a torna cara. Este trabalho tem como objetivo reduzir o tempo necessário para anotar conjuntos de dados de imagens através da criação de uma ferramenta de software que seja fácil de usar e na qual se tenha um sistema de anotação semi automatizado através do uso de um modelo de inteligência artificial. Essa ferramenta foi desenvolvida como um sistema Web e empregou-se a HQ-SAM, uma rede neural profunda para segmentação de imagens baseada no Vision Transformers, para gerar anotações de polígonos com base em entradas do usuário. Embora o HQ-SAM tenha uma boa generalização sem especialização, foi feito um ajuste fino com o conjunto de dados Bean Leaf para avaliar a capacidade de adaptação da rede a tarefas específicas. Observou-se um aumento na precisão do modelo ajustado em comparação com o modelo pré-treinado. Testou-se a ferramenta com 20 participantes, todos eles das áreas de visão computacional e computação gráfica. Pediu-se aos participantes que anotassem as mesmas duas imagens manualmente e com auxílio de IA, e registrou-se os tempos de anotação. Por fim, pediu-se aos participantes que preenchessem um formulário de usabilidade com perguntas sobre sua experiência de usuário. Nesssa avaliação, registrou-se um *speedup* mediano de 1,5× em relação à anotação auxiliada por IA em comparação com a anotação manual e as respostas às em relação à facilidade de uso e à utilidade da ferramenta proposta foram predominantemente positivas.

Palavras-chave: Aprendizagem profunda, aplicação Web, anotação semi automatizada, conjuntos de dados de imagens, transformadores visuais.

Acknowledgements

The completion of this monograph is the result of a journey filled with challenges and learning experiences. I would like to express my deep gratitude to everyone who, in some way, contributed to the completion of this work.

First, I thank my advisor, Professor Luiz Maurílio, for his guidance, understanding, and support during the writing of this monograph and other scientific initiation projects. His knowledge and commitment were crucial to finishing this project and bringing me to where I am in my academic journey.

To my parents, who have always supported me unconditionally at every step of my life. You are my foundation and my greatest examples of human beings. To all my family, for all the love, support, and teachings you have offered me throughout my life. A special thank you to Marly, my grandmother and my sponsor, a strong woman who is always there for her grandchildren.

Finally, I thank all my friends and colleagues who stood by me during this journey. Every conversation, every collaboration, and every shared moment contributed to the realization of this dream.

"'I checked it very thoroughly,' said the computer, 'and that quite definitely is the answer. I think the problem, to be quite honest with you, is that you've never actually known what the question is.'"

Douglas Adams (The Hitchhiker's Guide to the Galaxy)

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
CEP	Comitê de Ética em Pesquisa com Seres Humanos (Ethics Committee on Human Research)
HQ-SAM	Segment Anything in High Quality
IoU	Intersectionover Union
MLP	Multi-Layer Perceptron
NIfTI	Neuroimaging Informatics Technology Initiative
SAM	Segment Anything Model
UI	User Interface
ViT	Vision Transformer

1 Introduction

Deep learning is a class of methods that can learn from raw data the representations required to complete a certain task. They are made up of several simple modules called neurons, each of which slightly transforms the input in a nonlinear way. Those neurons are arranged into layers. By stacking up multiple layers and interconnecting them, it is possible to learn a wide range of complicated functions that model the data fed in (LECUN; BENGIO; HINTON, 2015).

Although deep learning has a lengthy history, only recently computer power and data availability have increased to the point where this technique could stand out. The year of 2012 can be considered a watershed moment because it was when the deep convolutional neural network architecture known as AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) outperformed all traditional techniques in the ImageNet Large Scale Visual Recognition Challenge, which has been held annually since 2010 (RUSSAKOVSKY et al., 2015).

For over a decade now, deep learning has been the state of the art of many artificial intelligence applications, including speech recognition, scientific data processing (such as those from particle accelerators), predicting the behavior of potential therapeutic drug molecules, and many others (LECUN; BENGIO; HINTON, 2015).

According to LeCun, Bengio and Hinton (2015), the most common method for fitting a deep learning model is via a technique known as supervised learning. It works by first collecting a vast amount of data, annotating it, and then executing the so-called training algorithm. For example, given a set of pictures of cats and dogs, each with a label that correctly identifies the image class, a deep learning model can iteratively pass through the dataset, attempt to guess which class of animal is present in each picture, compare its predictions to the correct labels, and then self-correct itself from its mistakes.

Having high-quality annotations is essential for a supervised learning. Aside from labels that indicate which class an image belongs to, there are different forms of annotation for different purposes. As an instance, object detection in a computer vision context refers

to locating objects in an image such as faces, cars, pedestrians, and so on (ZHAO et al., 2019). Annotations for that type of problem, in addition to the object class, require the object's coordinates in the image, such as its bounding box.

Another classic problem is the image segmentation. There are different kinds of segmentation, but overall, the main goal is to create multiple meaningful partitions of an image (MINAEE et al., 2021). Segmentation annotations are frequently more complex than the ones for other tasks since the problem requires pixel-wise class assignment.

Despite of its effectiveness, deep learning presents numerous challenges, including it being intrinsically a blackbox — i.e., how the model makes decisions is hardly explainable in human language — and data-related issues, such as its scarcity for specific tasks and overfitting (SMITH; SMITH; HANSEN, 2021). Overfitting occurs when a model performs well on training data but not well on unseen data, implying that it does not generalize its learning. Among other factors, overfitting is induced by using a dataset with an insufficient amount of elements, so noise and outliers cause significant influence on the training results (YING, 2019).

One of the primary causes of data scarcity is the cost of annotating a dataset. Consider the case of MS COCO, one of the most popular datasets in computer vision for object detection and segmentation. Over 70,000 hours of human labor were required to provide approximately 160,000 annotated images, which represented roughly fifty percent of the initial plan (LIN et al., 2014).

Annotating datasets is expensive mostly because it requires many hours of human labor. Thus, introducing automation, such as an artificial intelligence agent, which can carry out at least some part of the work for itself, could significantly reduce dataset creation costs.

This work proposes to develop an annotation software tool that implements such an artificial intelligence agent. As deep learning algorithms dominate the state of the art in artificial intelligence, it is natural to employ them for the development. The tool should make use of a model trained on previously existing data, so it is able to make reasonable predictions about how new data should be annotated. The human annotator's work is then reduced to reviewing the agent's forecast and correcting any errors that may arise.

1.1 Objectives

The main objective of this project is to create a software application that uses an artificial intelligence model to partially automate image annotation. As a result, we intend that the tool can improve the efficiency in new datasets creation processes by reducing the number of hours necessary of human work. Also, we have the following specific objectives:

- To investigate which artificial intelligence models can be employed to aid annotation;
- To develop user-friendly software with a low learning curve;
- To be able to export annotation data in appropriate forms for use;
- To evaluate the proposed tool with users.

2 Theoretic Foundations

In this chapter, we will give a brief introduction to some concepts of artificial intelligence (AI), focused on the techniques used by this work. In Section 2.1, we discuss the origins of the AI field and the basic functioning of artificial neural networks. Following this, in Section 2.2, we examine the vision transformer, which is the base architecture of the deep learning model used by us. Lastly, we discuss in Section 2.3 the HQ-SAM (KE et al., 2023a) and its ability to segment images based on human prompts.

2.1 Artificial Intelligence and Neural Networks

Artificial Intelligence is a term first used in 1955 in "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence" (MCCARTHY et al., 2006). The proposal aimed to raise funds for the realization of a conference with the goal to advance the understanding of "(...) how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves."

Although artificial neural networks (ANNs) were not initially considered the most promising area, the AI community has been studying them at least since the Dartmouth Conference. The ANNs are based on a simplified model of biological neurons. An artificial neuron consists of weights $w_1, w_2, ..., w_n$, a scalar called bias b, an activation function f, and inputs $x_1, x_2, ..., x_n$. Equation (2.1) gives the neuron's output y, and Figure 2.1 provides a graphic representation (HAGAN; DEMUTH; BEALE, 2014).

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.1}$$

Artificial neural networks are built in layers. There is always an input layer consisting of the input vector fed into the network, as well as an output layer containing the results. The intermediate layers are known as hidden layers, and they consist of one or more artificial neurons. Traditionally, all neurons in a layer share the same input, and the output of one layer serves as the input for the next. This type of architecture is known

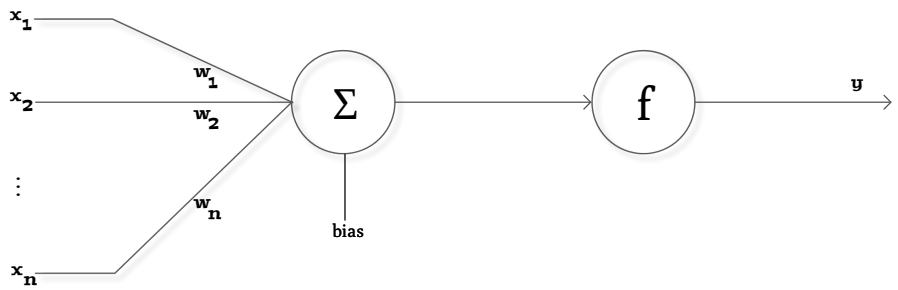


Figure 2.1: A graphical representation of an artificial neuron. Adapted from Hagan, Demuth and Beale (2014).

as feed-forward neural networks and it is illustrated in the Figure 2.2.

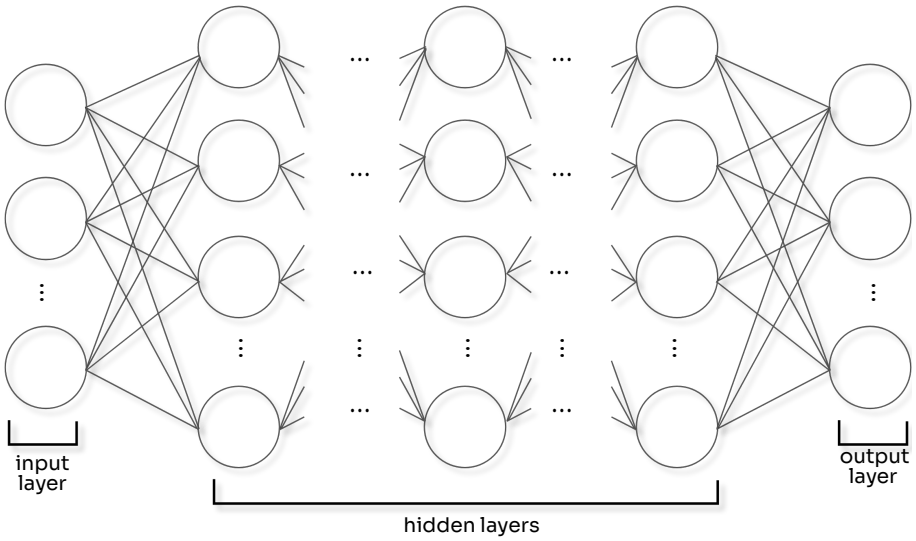


Figure 2.2: A graphical representation of an artificial neural network (Adapted from Hagan, Demuth and Beale (2014)).

A neural network requires good weights to function properly. The most frequent method for finding them is to employ an algorithm known as backpropagation. Initially, the weights are set randomly, and the model is given some data for making predictions. The forecasts are then compared to the correct labels or annotations using an objective function, which quantifies how far the forecast deviates from the desired output. Next, the gradient of the objective function is calculated, because its opposite indicates the direction where the prediction error decreases the most. Finally, the chain rule is employed to update each node in the network along the negative gradient direction. This process is repeated until the mean of the objective function stops decreasing (LECUN; BENGIO;

HINTON, 2015).

2.2 Vision Transformers

From a theoretical standpoint, multi-layer feed-forward networks with non-linear activation functions are already universal approximators. They are capable of approximating any measurable function with any degree of precision (HORNIK; STINCHCOMBE; WHITE, 1989). However, over time, several different ANN architectures have been proposed with the goal of improving efficacy in certain domains.

For instance, Vaswani et al. (2017) proposed the Transformer architecture for machine translation. The novelty was that this model only used the so-called attention mechanism, whereas most previous state of the art approaches used techniques such as recurrency and convolution. The Transformers launched a new era of high-quality models in natural language processing. As a result, new efforts began to utilize them in other fields as well.

Dosovitskiy et al. (2021) introduced the Vision Transformer (ViT) for image classification. The authors demonstrated that the ViT outperformed the state of the art at the time on large datasets. The ViT has since become part of a variety of computer vision models, including the one employed in this study. Thus, we present ViT here with further details.

As Figure 2.3 illustrates, how the Vision Transformer works can be split into six main steps. First the image input is split in a sequence of fixed-size 2D patches, because Transformers work with sequential data. Then, the patches are flattened and mapped to D dimensions through a learnable linear projection. In addition, an extra learnable embedding, which later is going to be used as a classification token, is prepend to the sequence of patches embeddings. In order to retain positional information, a learnable positional encoding is applied. Next, the Transformer encoder takes the embeddings as input, computes the attention between them, updating each embedding based on all the others. Finally, the classification is performed by a multi-layer perceptron (MLP) fed with the final state of the extra embedding, which is expected to serve as the image representation.

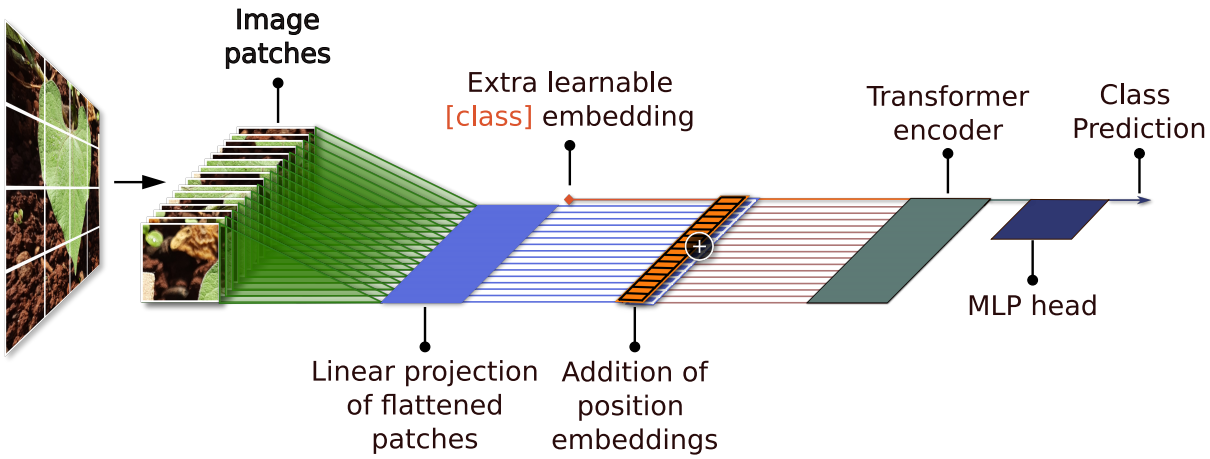


Figure 2.3: Vision Transformer (Adapted from Dosovitskiy et al. (2021)).

Figure 2.4 depicts the Transformer encoder's internal structure. Essentially, it is made up of L blocks of normalization, multi-head attention, residual connection, a second normalization, MLP, and a final residual connection. A residual connection is a connection that bypasses one or more layers (HE et al., 2016).

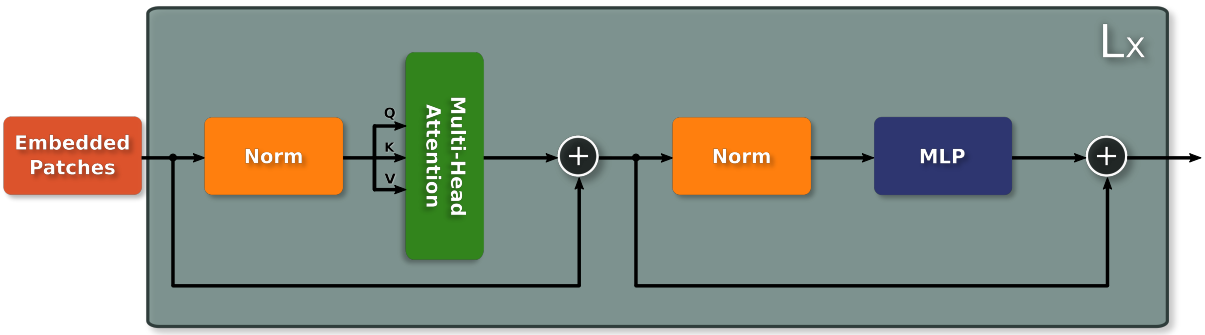


Figure 2.4: Transformer encoder (Adapted from Dosovitskiy et al. (2021)).

The attention layer has three inputs, which are denoted as queries, keys, and values — in ViT, all the three are distinct linear projections of the same data, but queries source could be different than the values and keys sources in some other cases. The model compares queries and keys for a given set of inputs to determine how much each value should contribute to the final output. Let Q, K, and V be matrices containing, respectively, queries, keys, and values. The matrices Q and K share the same dimensionality d_k, while V has dimensionality d_v. The similarity of Q and K is computed through the dot product and, for numerical stability reasons, it is divided by $\sqrt{d_k}$. Then, a softmax function is applied to obtain a probability distribution. Finally, the result is multiplied

by V to produce the attention. Equation (2.2) summarizes these processes.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

Also, the queries, keys and values can be linearly projected onto h different heads. According to Vaswani et al. (2017), "Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions." In the end, the outputs from the heads are concatenated and linearly projected in order to get the final values.

2.3 Segment Anything in High Quality (HQ-SAM)

The HQ-SAM (KE et al., 2023a) is a highly accurate segmentation model with zero-shot generalizability, which is used in this present work to semi-automate annotation. It is based on the Segment Anything Model (SAM) (KIRILLOV et al., 2023), a ViT-based architecture trained on the largest segmentation dataset ever created at publishing time, consisting of around 1.1 billion masks and 11 million images. As most of SAM's training dataset was generated automatically, it struggles with some fine-grained segmentation tasks. Furthermore, due to the size of the model and dataset, direct fine-tuning is difficult to accomplish without compromising generalization performance.

HQ-SAM tackles the issue by reusing the SAM's weights wherever feasible, adding a few new structures, and curating some existing datasets to create a new one with fine-grained masks. HQ-SAM keeps the encoder, the heavier component of SAM, untouched. The encoder is made of ViTs, and its main role is to take an image input, extract useful features, and represent it in a lower dimension. The encoder output is called image embedding.

The other component of the HQ-SAM is the decoder. It uses the image embedding and a user prompt to generate the segmentation mask. The user prompt indicates the region in the image to be segmented, which might be a set of points, a bounding box, or a coarse segmentation mask. Those prompts are tokenized and a learnable output token is prepended.

Figure 2.5 shows the original SAM's decoder. First, the attention is applied to the tokens only (self-attention). Then, the tokens attend to the image embedding (i.e, the queries come from the tokens while the keys and values come from the image embedding) and they pass through a MLP. Next, the image embedding obtains tokens' information by attending to them. This process is repeated twice. Afterward, the resulting image embedding is up-scaled through two convolution layers, and the tokens attend the image embedding again. The updated output token is sent through an MLP to match the image embedding dimensions, and the mask prediction is obtained using the dot product of the image embedding and the output token. The diagram omits it, however at each attention layer, positional encoding is applied to the image embedding, and the original prompt tokens are added to the updated ones.

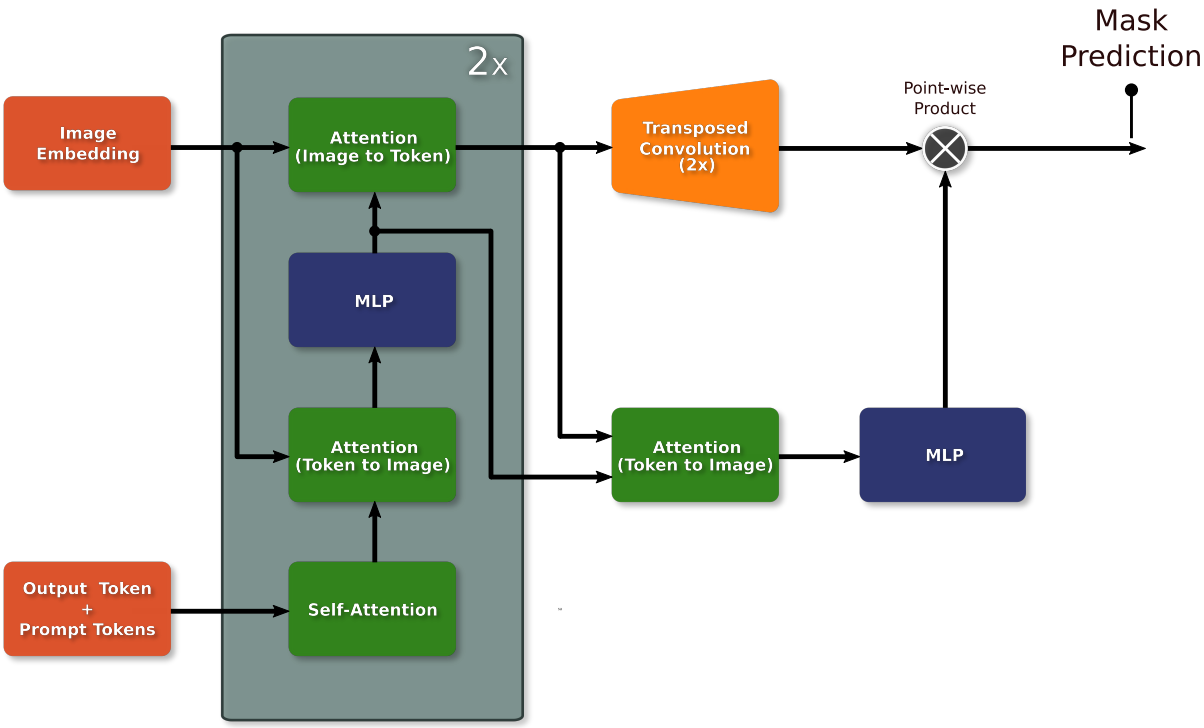


Figure 2.5: SAM decoder (Adapted from Kirillov et al. (2023)).

The structures added by HQ-SAM are shown in Figure 2.6. The proposed approach is to include one extra learnable token, which the authors call as the HQ-Output token. Also, HQ-SAM sums the mask features, i.e. the image embedding in the original decoder after the convolutions, with the output of the encoder's first and last attention blocks, a process referred to as Global-local Fusion. The premise is that the encoder's

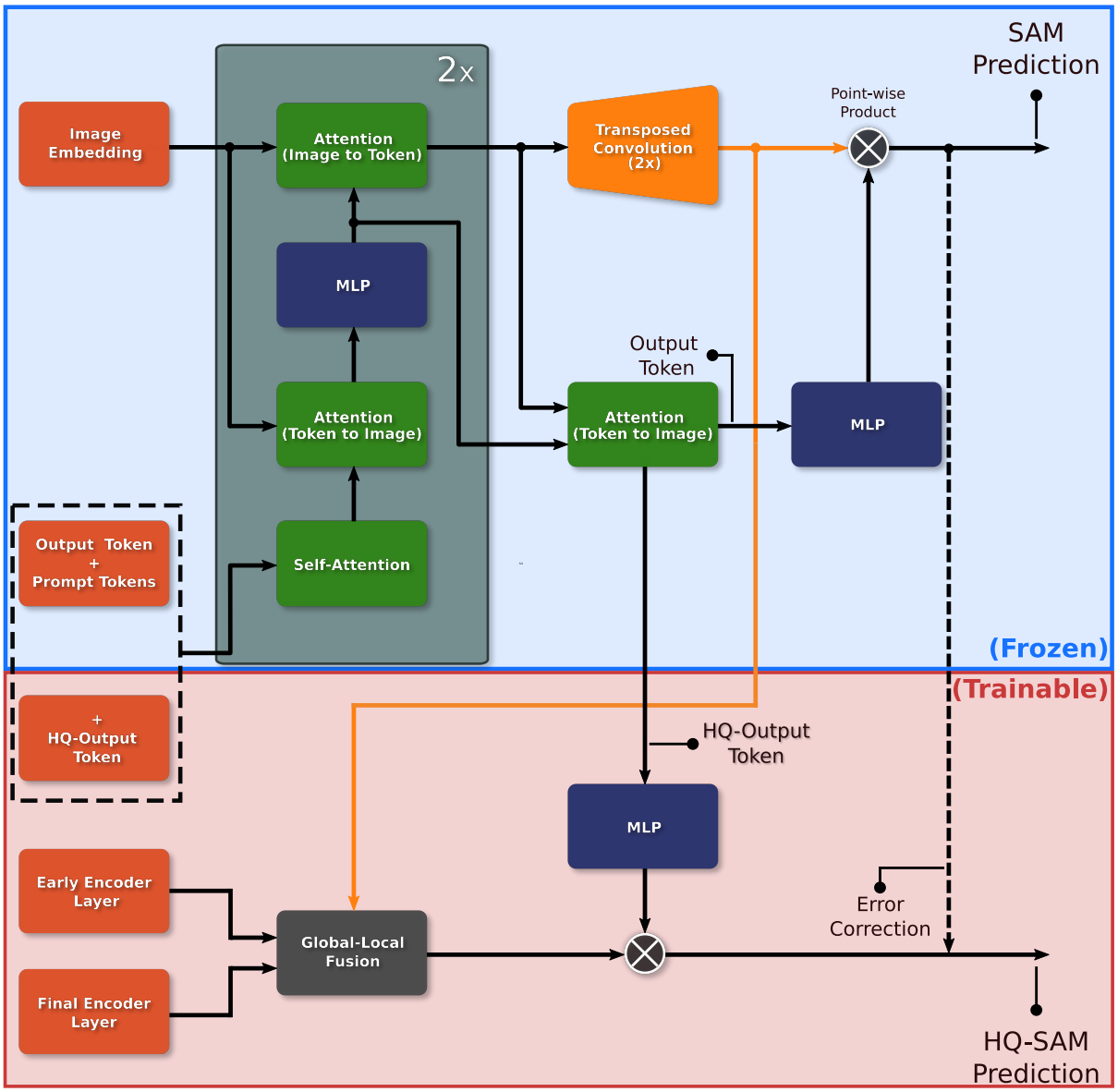


Figure 2.6: HQ-SAM Decoder (Adapted from Ke et al. (2023a)).

early layers should contain lower-level local features, while the final layers should have more global context information. The segmentation mask is then predicted by taking the dot product of the Global-local Fusion output and the up-scaled HQ-Output token. At training time, only the new structures are updated via back-propagation, and at inference time, the original SAM output and the HQ-SAM output can be combined for error correction.

3 Related Work

Over the years, various dataset annotation softwares have been developed. Some were designed for specific cases, while others had a broader reach. For example, the MS Coco project created its own annotation tool, where users painted over objects to create segmentation masks (LIN et al., 2014). On the other hand, LabelMe, previously one of the most influential annotation softwares, had a wider purpose. It was an open-source project and an online platform where individuals collaborated to construct an open dataset using polygon annotations (TORRALBA; RUSSELL; YUEN, 2010).

However, a limitation that both LabelMe and MS Coco share is that they rely entirely on human labor. Subsequent works on dataset annotation employed a variety of computer vision techniques to add automation into this task.

Boonsri and Limpiyakorn (2023) applied traditional computer vision techniques to detect and annotate the coordinates of cannabis seeds in an image. The authors intended to help Thai growers distinguish between female and male cannabis plants before planting them because only female cannabis is commercially viable. The authors first needed to build a dataset, so they proposed a software that takes as input an image containing a handful of seeds distributed on a white background, converts it to gray-scale, then smooths it with Gaussian blur. The image is then passed through a threshold, and contours are calculated to find the bounding boxes for the seeds. Their tool writes out the results in JSON files, and they still need to perform the male/female classification.

Classic algorithms can also be used in scenarios with less control over the image's background and noise than in the cannabis seeds case. Qin et al. (2018) used edge detection to create a general-purpose annotation software called ByLabel. Their tool uses an algorithm called Edge Drawing (TOPAL; AKINLAR, 2012) to obtain edges segments all across the image. These segments are partitioned into smaller fragments either manually or automatically — automatic partition is based on turning angle heuristics. The user then chooses the edge fragments which will compound the final annotation. The expectation is that edge selection will be less costly than traditional annotation methods. The

authors led an experiment with 10 volunteers with no experience in annotation. According to their evaluation, using ByLabel reduces the annotatiom time by a factor of 56% compared to using the traditional LabelMe.

Entering in the realm of semi-automation powered by artificial intelligence, Yu et al. (2023) proposed the use of unsupervised learning algorithms to simplify annotation. The authors work with a specialized dataset containing images of rock thin sections utilized in geological and mineralogical studies. Their tool uses a clustering algorithm to split the image into several non-overlapping groups of similar pixels called superpixels. The human annotation is then reduced to labeling the superpixels of interest. The authors report a speedup of 5× to 8× when comparing experts using their tool versus generating segmentation masks with QGis[1] or Photoshop.

Supervised learning also plays an important role in dataset annotation, and is applied in various approaches. Fîciu et al. (2018) aimed to automate the annotation process completely. The authors employed an architecture called Mask R-CNN (HE et al., 2017) to predict bounding boxes and segmentation masks. The bounding box is utilized as input, alongside the image, to help Polygon-RNN (CASTREJON et al., 2017) predict an annotation polygon. The results are exported in JSON format. One limitation of this method is that it cannot generalize to previously unseen classes. In fact, the authors report results for only four categories: people, cars, trucks, and bicycles. The addition of new classes would require retraining Mask R-CNN.

Most annotation tools allow human users to fix the neural network's predictions. For example, Philbrick et al. (2019) developed a software to annotate medical images. Their system supports plugins that enable the deployment and use of different neural networks to predict segmentation masks, as long as the neural network is implemented in Keras[2] running on TensorFlow.[3] The user can change the predictions using drawing tools, including painting, erasing, and filling. Also, annotated images can be used to fine-tune models within the software. According to the authors, their software design is limited to the annotation of images stored in the Neuroimaging Informatics Technology Initiative

[1] https://www.qgis.org/
[2] https://keras.io/
[3] https://www.tensorflow.org/

(NIfTI) file format.

Other strategies attempt to use annotators' knowledge to improve the model's predictions. For example, in the case of object detection, Pugdeethosapol et al. (2020) proposes asking users to click on an object and then use the click location to select among bounding boxes generated by a pretrained backbone. Moreover, the user can correct the predictions, and the corrected data is used to train a second network incrementally during runtime. Over time, the second model is expected to make better predictions and require fewer user adjustments during annotation.

There are interactive approaches for image segmentation as well. Sambaturu et al. (2023) developed a framework for annotating urban city scenes in which a pretrained network gives mask predictions, and the user can make adjustments by scribbling where the network made mistakes. These corrections are utilized to determine local loss at scribbling pixel coordinates and to backpropagate during inference. Backpropagation is limited to only a few final layers for efficiency factors. Based on an evaluation performed with two experts, the authors claim a time savings of up to 14.7× compared to complete human annotation.

Differently from previously cited works, we propose using segmentation prompts to reduce annotation time. Prompts are now widely used in AI text generators like ChatGPT.[4] However, unlike ChatGPT, our prompts are the ones supported by the SAM/HQ-SAM model rather than text prompts. Given that HQ-SAM has zero-shot capabilities, our tool can be generalized for a wide range of classes, while also being possible to fine-tune the model to perform better in specific classes. In addition, we propose conducting a more comprehensive quantitative and qualitative evaluation of the proposed tool in comparison to other studies. We proposed evaluating our tool with both experienced and inexperienced participants, aiming for a significant sample size based on existing literature.

[4]https://chatgpt.com/

4 System Development

This work's proposed system is a single-page web application with a desktop-like workflow. Our application is backed up by a server that serves the content, manages the data, and handles the interaction AI model. Thus, we use a client-server architecture, as shown in Figure 4.1.

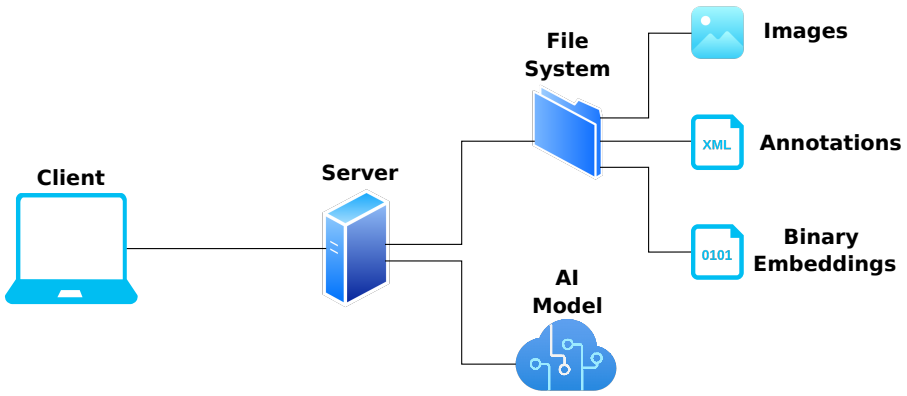


Figure 4.1: Overall system Architecture.

In the following subsections we will discuss the software with further details. In Section 4.1 we present the main languages, libraries and tools used in the application. In Section 4.2 we describe the fine-tuning process we performed in the HQ-SAM network. In Section 4.3 we discuss the server-side implementation. Lastly in Section 4.4 we present the client-side and the main features available for the users through the user interface (UI).

4.1 Languages, Libraries, and Tools

The web client was developed with HTML, SASS[5] (a CSS extension language), and plain JavaScript. We also used Parcel,[6] a JavaScript build tool that enabled us to use NPM[7] for dependency management, as well as supplying source-mapping and code minification

[5]https://sass-lang.com/
[6]https://parceljs.org/
[7]https://www.npmjs.com/

out-of-the-box.

The server was implemented in Java 17, and it uses the Spring Boot framework to create API endpoints, handle HTTP requests, and serve static files. We chose Gradle as the build tool and dependency manager for the back-end. We used the OpenCV[8] library for image processing and to perform a few computer vision tasks.

The HQ-SAM was written in Python, and we used existing Python scripts to fine-tune the network. To integrate HQ-SAM into the system, we exported the trained weights to an ONNX[9] file, which can be run directly by the server powered by the ONNX Runtime for Java.

4.2 Fine-tuning the HQ-SAM Network

Although our ultimate goal is to produce a general-purpose annotation tool, the software development was initially focused on meeting the requirements for annotating the Bean Leaf Dataset (SILVA, 2023). This set consists of images containing a prominent bean leaf and an augmented reality marker. We fine-tuned the model to improve HQ-SAM accuracy in that dataset and developed a feature in our system that supports multiple HQ-SAM checkpoints. During the fine-tuning process, we froze the original SAM layers and trained only the structures introduced by HQ-SAM.

There were three HQ-SAM pre-trained versions available to fine-tune: ViT-B, ViT-L, and ViT-H. The difference between them is the size of their backbones. A smaller backbone is lighter but less accurate, whereas a larger backbone is heavier but more accurate. Due to hardware constraints, we chose the ViT-B, which is the lighter model.

Only leaves were used in the fine-tuning because the marker annotations were already finished. The available dataset contained a total of 3756 images organized into 300 folders, each folder containing different pictures from the same leaf. We randomly selected 20% of the folders (60 in total) to provide the images of our testing set, while the remaining 80% provided the training set. The original images had a resolution of 3468×4624. They were cropped to 1024×1024, the HQ-SAM input resolution, keeping

[8] https://opencv.org/
[9] https://onnx.ai/

the leaf in the center.

We kept almost the same parameters used by HQ-SAM :

- Optimizer: Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, $\lambda = 0$);
- Initial learning rate: 10^{-3};
- Learning rate decay factor: 10^{-1} every 10 epochs;
- Number of epochs: 12;
- Seed: 42.

The only difference is that we reduced the training batch size from 4 to 2 due to VRAM limitations. We performed the fine-tuning on a Quadro M5000 GPU with 8GB VRAM, which took a total of 11.7 hours to complete. The results will be presented in Section 5.1.

The HQ-SAM repository (KE et al., 2023b) contains a script for exporting the model's decoder to ONNX but not one for exporting the encoder. So, in order to obtain all of the fine-tuned weights in the same format, we adapted SAM Exporter (NGUYEN, 2023), a tool licensed under MIT that can export the complete original SAM to ONNX, to export the HQ-SAM.

4.3 Server

The server code is self-contained; it compiles into a JAR that includes all required libraries, ONNX files, and WEB files. The application creates a directory for itself in the user's home folder, which is a common place to store application data. The application's directory contains three automatically created sub-directories:

1. *datasets*: the directory where the server will look for the images and annotations data. The content in this directory is statically served, and its subdirectory structure is exposed through an endpoint so the client can implement directory navigation;

2. *checkpoints*: the directory where the server will look for user's fine-tuned checkpoints of the HQ-SAM decoder. They are recognized as alternatives to the default checkpoint included in the JAR file.

3. *embeddings*: the directory where the server will look for image embeddings before running HQ-encoder. Whereas the HQ-SAM decoder is lightweight, the encoder is heavy. As the encoder's output for an image does not change no matter the prompt, we save it to the disk for speeding up later usages of the same image.

When handling prediction requests, the server responds with a list of coordinates, which represents the segmentation polygon of an object. However, the output of HQ-SAM is a binary segmentation mask. The conversion from a binary image to a sequence of points is performed through OpenCV, first by using the Suzuki et al. (1985) algorithm to find the mask contour and then by using the Douglas and Peucker (1973) algorithm to get the corners of an approximated polygon. This process is illustrated in Figure 4.2.

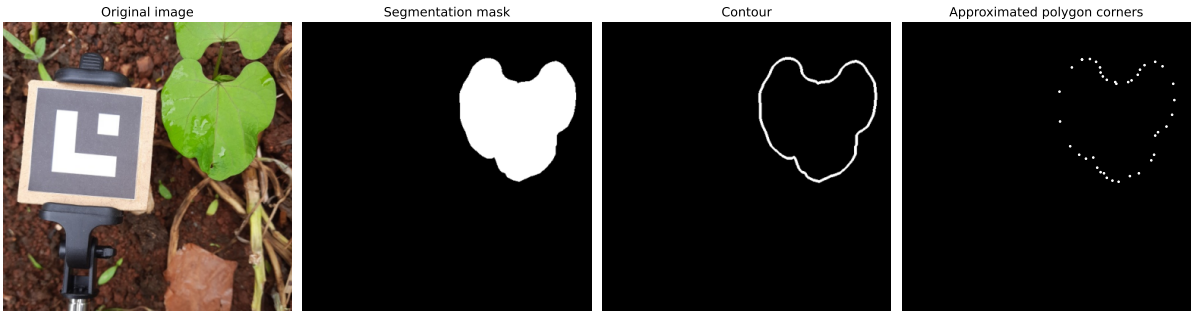


Figure 4.2: Image to polygon steps.

The server only stores annotations on disk at the client's request. In the save request, the client can specify where will be the location of the annotation within the datasets directory, as well as the file name and format, as long as it is in text format. This allows support for multiple dataset structures and annotation formats as long as the client knows to read them.

In addition, the server serves the default client on the user's localhost port 8080. As a result, a single JAR file can be provided as a complete software package, and it does not require installation.

4.4 Client

The UI displays the client's main features, as shown in Figure 4.3. The interface is mostly in Portuguese, as it is initially intended for Brazilian users. The screen is divided into four sections: a bar with the main features at the top, a bar with secondary information at the bottom, a file list on the right side, and a canvas that displays the content in the center.

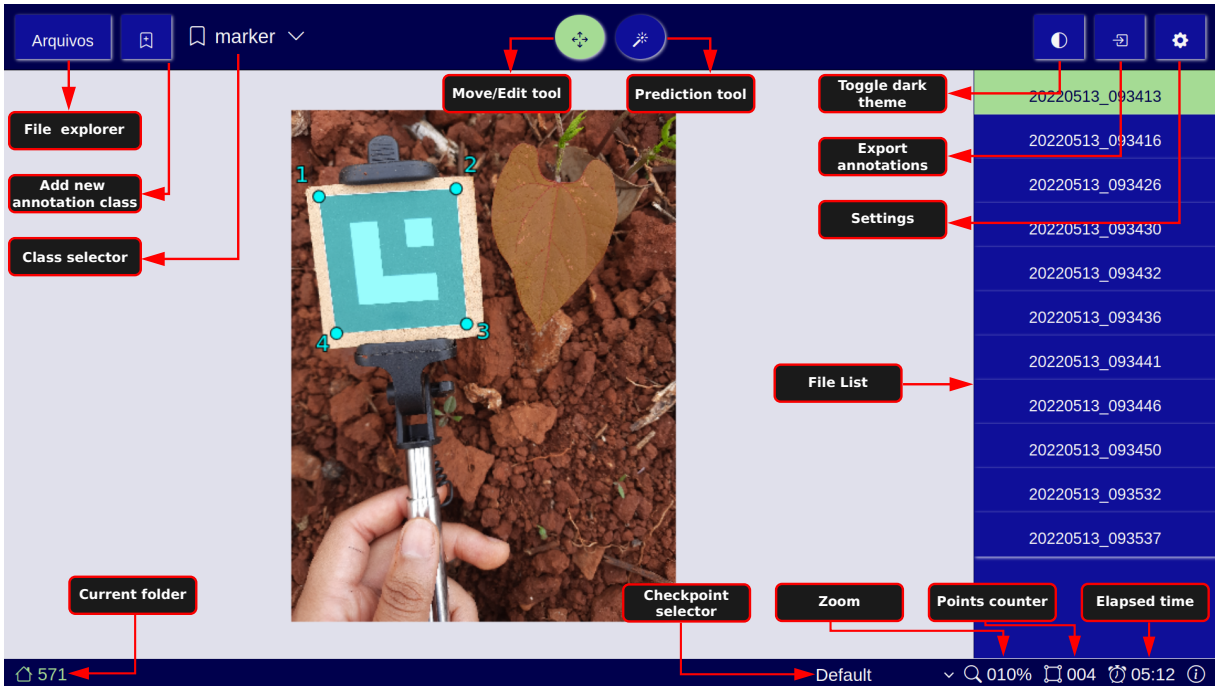


Figure 4.3: User interface main elements.

In the center of the top bar, we placed the buttons that should be used more frequently, which are the tools to edit annotations manually and to use the AI model. We assume that the file explorer, class selector, and class creation are the second most commonly used buttons, thus we positioned them on the left. The right-side buttons are likely to be used less frequently. Respectively, they toggle the dark theme, export annotations, and open the settings modal.

We employed modals to keep the client as a single-page application while having a few additional and straightforward interfaces, as shown in Figure 4.4. The file explorer modal displays the browseable list of folders in the server's *datasets* folder. The class creation modal contains only some fields to choose the class name, set an optional limit

of points, pick a color, and choose whether the points should be enumerated in the UI. The settings modal only include three sliders for controlling the opacity of the polygons, the maximum magnification allowed, and the scrolling speeding.

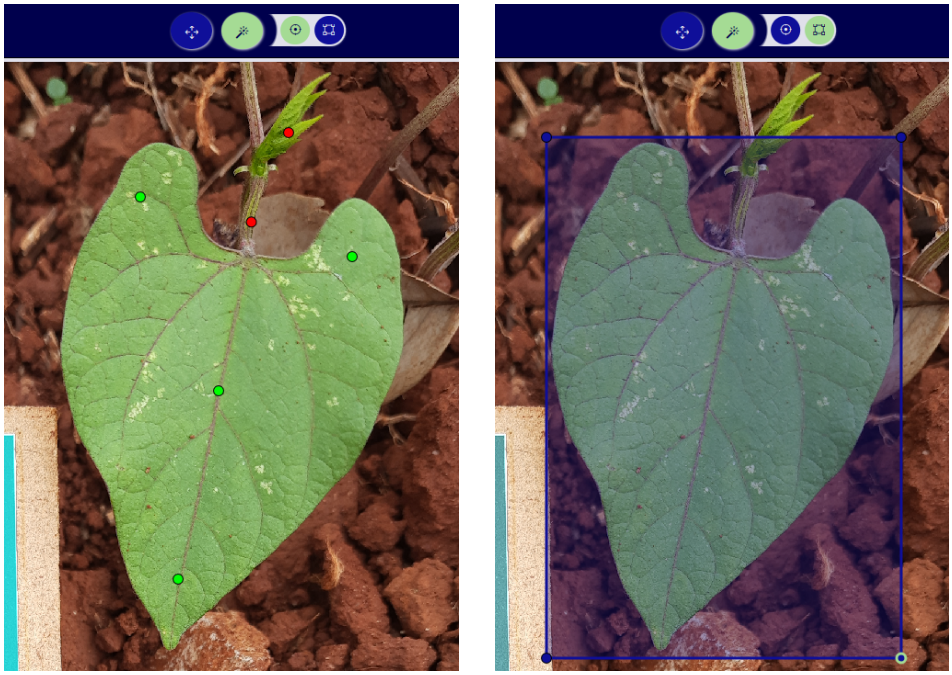
Figure 4.4: User interface modals.

When a folder is selected, the client retrieves a list of image files and displays their names in the right bar, in alphabetical order. The first image of the list and its corresponding annotations (if they exist) are rendered on canvas. The user can switch to the other images by clicking on their names on the list.

Once an image has been rendered on the canvas, the user can manually add annotations by clicking on the screen with the left mouse button while using the editing tool. Each click generates a point, resulting in a polygon. Users can also edit annotations by clicking and dragging points to different locations, or delete them by clicking on them and typing DELETE or BACKSPACE on the keyboard.

The AI tool provides users with two prompt alternatives, depicted in Figure 4.5. The first option is to use point prompts, which can indicate foreground or background. The foreground prompts can be added with a left click and are represented by green dots, while the background prompts need CTRL + left click and are represented by red dots.

The second option lets users draw a box around the object they want to segment. In either case, users should press the ENTER key after completing the prompts. These prompts serve as input alongside the image to the HQ-SAM, which then computes a segmentation mask based on them.



(a) Point prompts to segment a leaf. (b) Box prompt to segment a leaf.

Figure 4.5: A demonstration of the two different prompt modes available.

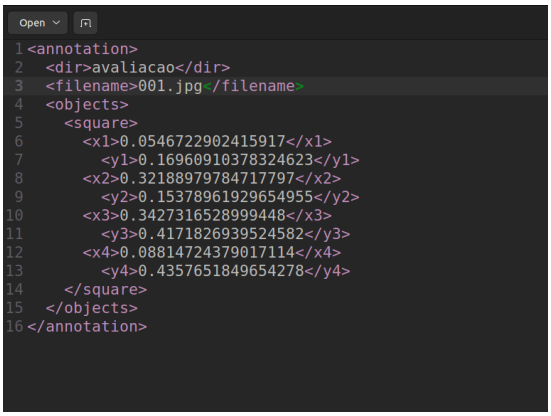
Regardless of the tool used, users can move the image on the canvas by clicking and dragging with the mouse's right button, and zoom in or out using the mouse scroll. Furthermore, when the mouse cursor hovers over a selectable object (polygons or points), the object is highlighted with an outline to provide visual feedback.

The bottom bar displays secondary but useful information such as the name of the currently open directory, the number of points the selected annotation has, the amount of zoom currently applied to the rendered image, and a timer that begins when the user opens a folder. Additionally, users can see which checkpoint is currently being used by HQ-SAM in the back-end, and they can click on it to select another one if there are other options stored in the checkpoints folder in the server.

Lastly, users can save their work to the server using the conventional CTRL + S shortcut. The annotations are normalized by the image's height and saved in an XML file, as in the Figure 4.6. Users can also export and save locally the annotation files by

clicking the export option in the top bar. The download is a zip file including one folder with the XMLs and one folder with the annotations in a JSON file following the MS COCO format.

(a) An augmented reality marker annotated with four points and assigned to the class "Square."



(b) The resulting XML file containing the marker's annotations points normalized, the image file name, and root directory.

Figure 4.6: A depiction of an annotation and the corresponding XML file.

5 Experiments and Results

This chapter presents the methodology and findings of our research and is divided in two sections. Section 5.1 discusses the process of fine-tuning the HQ-SAM model and how it affects segmentation performance. Section 5.2 discusses the performance study to determine how our AI-aided annotation compares to manual annotation, as well as the usability study to assess the user-friendliness of the annotation tool developed as part of this research.

5.1 Fine-tuning HQ-SAM Network

To evaluate the effect of the fine-tuning for the Bean Leaf Dataset, we used the intersection over union (IoU) and the Boundary IoU (CHENG et al., 2021). These metrics are the same as those used by the HQ-SAM authors. In fact, we calculated them using their implementation.

The IoU is obtained using Equation (5.1). We divide the area of the intersection between the ground truth mask (G) and the prediction mask (P) by the area of their union. The Boundary IoU in Equation (5.2) is close to the IoU, but it focuses on the edges. G_d and P_d are the sets of all pixels within d pixels distance from the ground truth and prediction contours respectively" (CHENG et al., 2021). The parameter d used by HQ-SAM is 2% of the image diagonal and we kept it unchanged.

$$\text{IoU} = \frac{|G \cap P|}{|G \cup P|} \tag{5.1}$$

$$\text{Boundary IoU} = \frac{|(G_d \cap G) \cap (P_d \cap P)|}{|(G_d \cap G) \cup (P_d \cap P)|} \tag{5.2}$$

We evaluated both the predictions using box prompts and point prompts on the test set, keeping the same patterns. For the box prompts, we used the leaves' bounding boxes. For the point prompts, we used only the center of the bounding box. We chose

using a single point prompt both for simplicity and because it represents the minimal effort annotation method in our tool.

The graphs in Figure 5.1 summarize the results for the box prompts, whereas Figure 5.2 shows the results for the point prompts. In both figures, when comparing pre-trained weights to fine-tuned ones, we can observe that the mean increased while the spread decreased for both metrics. For the box prompts the mean IoU went up from $98.6\% \pm 0.7\%$ to $99.1\% \pm 0.3\%$ and the mean Boundary IoU went up from $(90.9 \pm 3.7)\%$ to $93.3\% \pm 1.8\%$. Meanwhile, for the point prompts, the mean IoU increased from $86.6\% \pm 19.0\%$ to $97.7\% \pm 2.4\%$ and the mean Boundary IoU increased from $66.3\% \pm 25.5\%$ to $80.3\% \pm 15.4\%$.

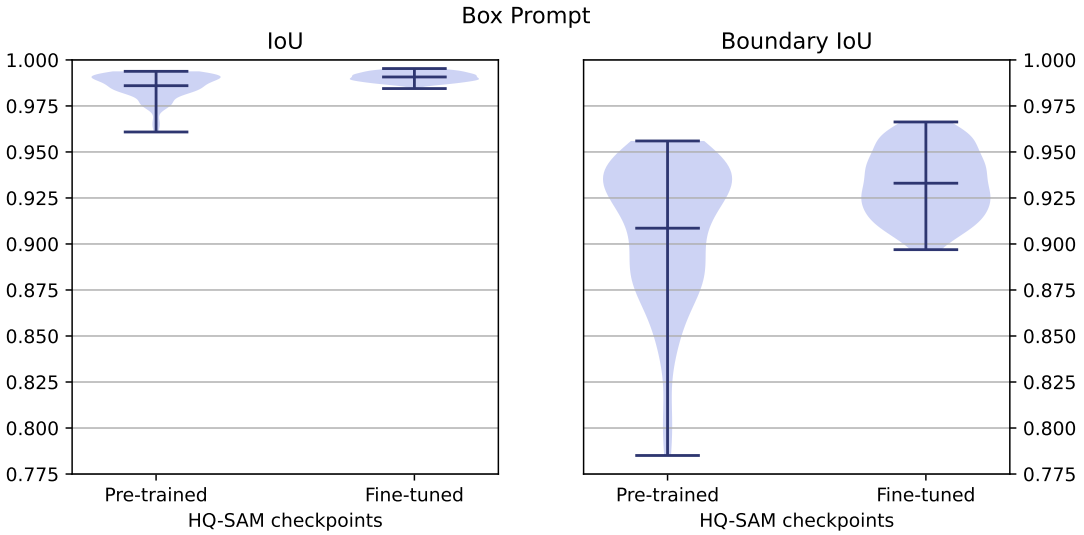


Figure 5.1: Comparison of IoU and Boundary IoU values for the pre-trained and the fine-tuned checkpoints using box prompts.

Figure 5.3 shows a qualitative example of the fine-tuning effect. As shown in Figure 5.3a and Figure 5.3c, the model without fine-tuning incorrectly assigns some green background elements to the leaf. The Figure 5.3c contains a specially large background area that was mistakenly included in the segmentation, suggesting that single-point prompts tend to be more ambiguous than box prompts. The fine-tuned model on the same images for both prompt types predicts the segmentation masks more accurately, following the actual leaf contour, as shown in Figure 5.3b and Figure 5.3d.

In summary, we conclude that the fine-tuning was successful. Box prompts carry more information than single-point prompts, but the fine-tuning method considerably

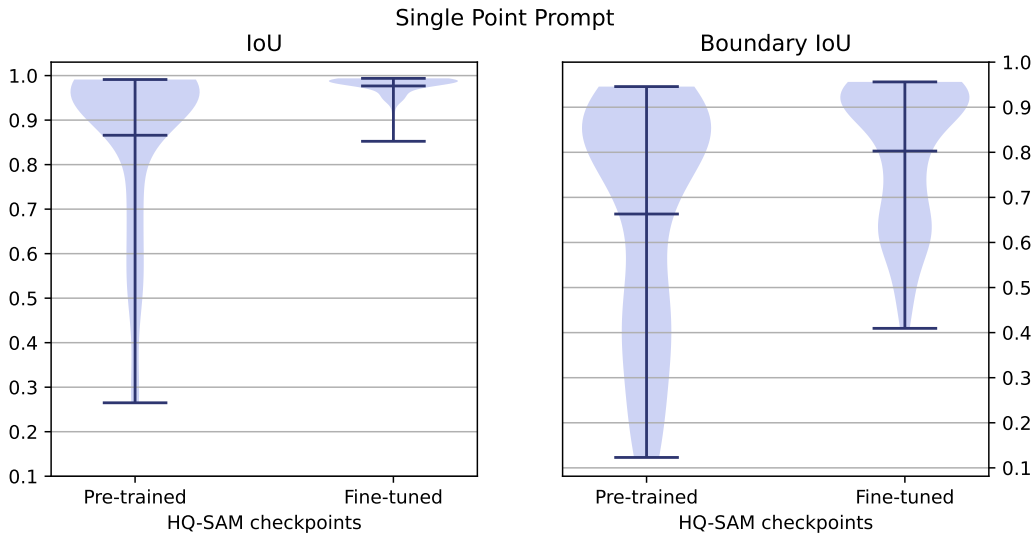


Figure 5.2: Comparison of IoU and Boundary IoU values for the pre-trained and the fine-tuned checkpoints using point prompts.

reduced the difference between them, making both of them viable options. Furthermore, the predictions became more consistent, making bad performance outliers less likely to occur.

5.2 Software Evaluation

In order to assess whether our proposed software meets its objectives (ease-of-use and annotation time reduction), we conducted an evaluation with 20 undergraduate and graduate students working on computer graphics or computer vision projects. The sample size is consistent with the recommendations of Alroobaea and Mayhew (2014) for usability studies intended for statistical significance and performance metric analysis.

We instructed the participants on how to use the software and asked them to annotate leaves in two distinct images. For the first image, we chose a leaf that is more regular in shape and hence easier to annotate. For the second, we chose a leaf with a wavy surface and a cut on the left side, making it more difficult to annotate. These images are displayed on Figure 5.4. Later in this work, we reference the first image as leaf 1 and the second as leaf 2.

The participants annotated the same two images from scratch twice: once completely manually and once only fixing the AI tool predictions. To diminish bias coming

(a) Default HQ-SAM prediction with box prompt.

(b) Fine-tuned HQ-SAM prediction with box prompt.

(c) Default HQ-SAM prediction with point prompt.

(d) Fine-tuned HQ-SAM prediction with point prompt.

Figure 5.3: Qualitative demonstration of the fine-tuning effect.

from variations on the AI prompts, we asked participants to utilize a single point prompt around the leaf's center for the first image and a box prompt for the second. We measured completion times and saved the annotations to compare later to the dataset's ground truth.

After finishing the annotations, the participants responded to a questionnaire about their user experience. Most of the questions used the 5-point Likert scale (LIKERT, 1932) to ask about how hard it was using the features in the application. Also, there was

(a) First image (leaf 1). (b) Second image (leaf 2).

Figure 5.4: Images used in the software evaluation.

a question about whether the participant had already used other annotation tools, and an optional open field to give feedback about the system. More specifically, the questions were:

1. Have you ever used other annotation software? (yes or no);
2. How easy is to understand the user interface? (very easy, easy, neither easy nor hard, hard, or very hard);
3. How easy is to use the polygon creation tool? (very easy, easy, neither easy nor hard, hard, or very hard);
4. How easy is to add a class for the annotation polygon? (very easy, easy, neither easy nor hard, hard, or very hard);
5. How easy is to use the artificial intelligence tool with point prompts? (very easy, easy, neither easy nor hard, hard, or very hard);
6. How easy is to use the artificial intelligence tool with box prompts? (very easy, easy, neither easy nor hard, hard, or very hard);
7. How easy is to fix the annotations generated by artificial intelligence? (very easy, easy, neither easy nor hard, hard, or very hard);

8. How easy is to export the files to the desired format after finishing the annotations? (very easy, easy, neither easy nor hard, hard, or very hard);

9. In general, how useful is the artificial intelligence tool for annotation? (very useful, useful, neutral, useless, very useless)

10. Open field for you to leave your comments, critics, and suggestions for the application.

It is worth noting that all evaluations were done online. We passed the instructions to each participant individually via Google Meet sessions and then sent them the URL to the software program. To ensure consistency, we asked participants to use the Google Chrome browser. For the questionnaire, we used the Google Forms. Also, we only started our research after receiving the approval of the CEP (*Comitê de Ética em Pesquisa com Seres Humanos* – Ethics Committee On Human Research) under the CAAE 78839724.3.0000.5147 *(Certificado de Apresentação para Apreciação Ética* – Certificate of Presentation for Ethical Consideration). The ethics committee's opinion is available in Portuguese on Annex A.

Regarding the questionnaire results, in the first question, only half of the participants reported prior experience with other annotation tools. In the last question, all participants said the AI tool was "very useful" for annotation. Figure 5.5 summarizes the results from the second to the eighth question, which are all about the system's ease of use. We highlight that in those questions, we kept a range of 70–90% of the participants answering "very easy", with the remainder responding "easy." The only exceptions were questions 4 and 7, which had one "neither easy nor difficult" answer each.

Furthermore, we received 13 answers to the open question asking for feedback. A few of them only contained compliments such as "I liked the interface, very intuitive, easy to use and with a very good design" and "The precision of AI makes correction much simpler." There were also a handful of suggestions, which we list below:

- To change the default class color to increase contrast;

- To show files on the files explorer as currently it only shows folders;

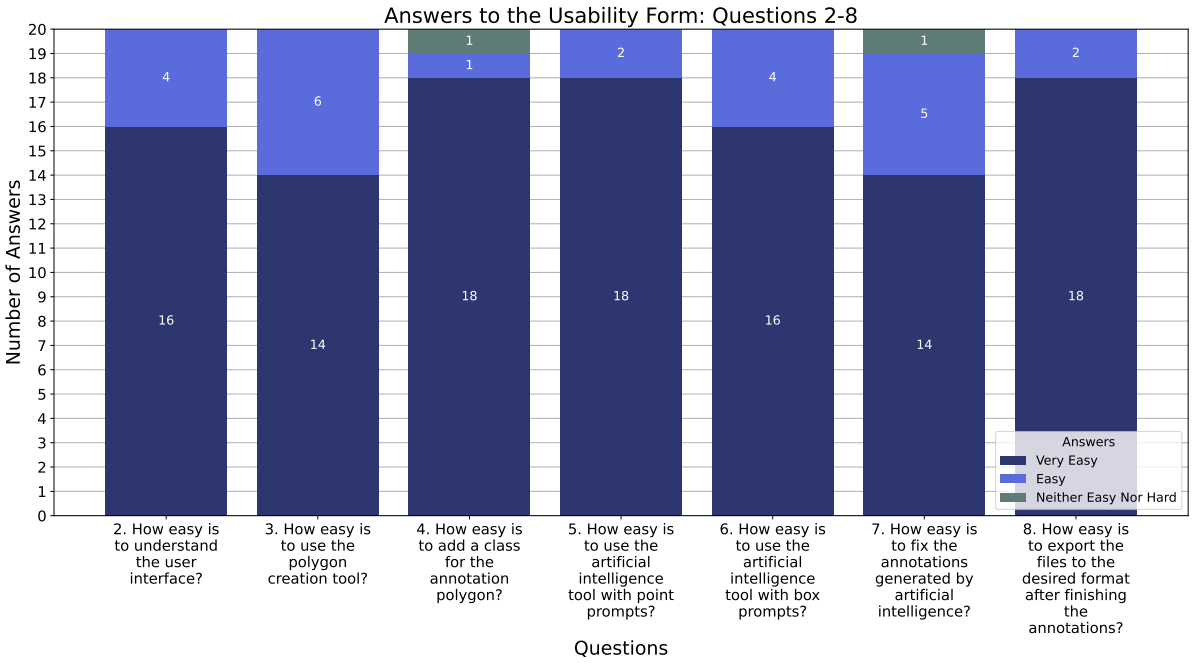


Figure 5.5: Answers to the questions 2-8.

- To add a loading spinner while changing from one image to other;
- To add an instructions pop-up, which may be opened by default when the user accesses the tool for the first time;
- To change the icons of the export button and the add class button so they are more intuitive.

Compared to the ground truth, the annotation accuracies using manual and AI tools were similar to each other. Using the mean IoU, the manual annotation accuracy was 98.8%±0.4%, whereas the AI-aided annotation accuracy was 98.8%±0.3%. Similarly, the mean Boundary IoU was $(96.1 \pm 1.2)\%$ for manual annotation and $96.3\% \pm 0.8\%$ for AI-aided annotation.

Investigating the annotation times, we observe in Figure 5.6 an overall reduction to finish the task while using AI. The mean time for manual annotation was (755 ± 399) seconds, while it was (470 ± 318) seconds for AI-aided annotation. However, we note that our data contains outliers pushing the standard deviation up, and that the time reduction was unequal for the leaf 1 and leaf 2.

To summarize the results, we calculated the speedup of AI annotation compared to manual annotation. The mean speedup was $1.9\times \pm 1.01\times$, but due to outliers and a

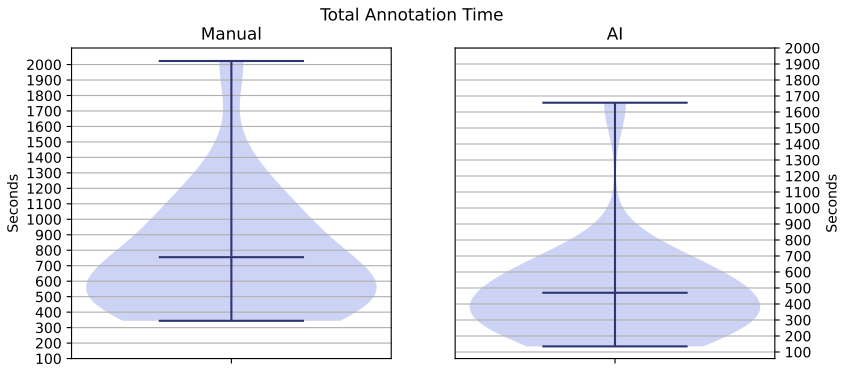


Figure 5.6: Summed up annotation times for the two leaves.

high standard deviation, we believe the median is a less biased metric. The boxplots in Figure 5.7 show us that while the overall median speedup was 1.5×, the median speedup for the leaf 1 was 1.3× and for the leaf 2 it was 1.7×. One possible explanation for such a difference may lie on the fact that the leaf 2 is harder to annotate, so the AI tool is more helpful. Furthermore, the prompts may have had an impact on this difference, as the box prompts used in leaf 2 carry more information and provide better mask predictions than the point prompts used in leaf 1.

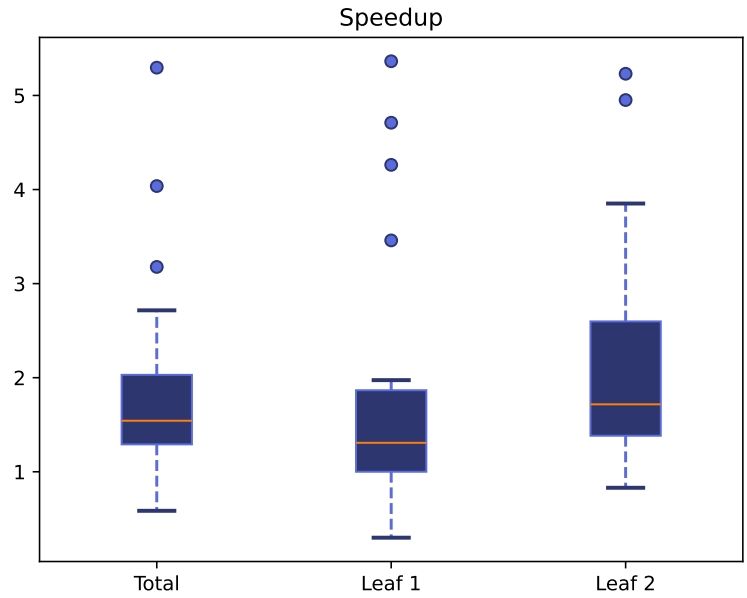


Figure 5.7: AI-aided time speedup compared to manual annotation.

Given the high accuracy of HQ-SAM predictions and the model's inference time

being significantly faster than human annotation, speedup values may appear to be lower than expected. However, the possible reason for those results is how participants corrected the AI's annotations. Some participants passed over most of the points in the annotation polygon, making minor changes that increased their overall annotation time.

Another interesting finding is that, on average, participants who reported not having used other annotation software completed the task faster. Also, they maintained comparable accuracy to those who had prior experience with other tools. Figure 5.8 shows that the speedup for the non-experienced was 1.8, while for the experienced it was 1.4. The mean IoU and and the mean Boundary IoU for both groups annotating the first leaf are in Table 5.1 and the metrics for the second leaf are in Table 5.2.

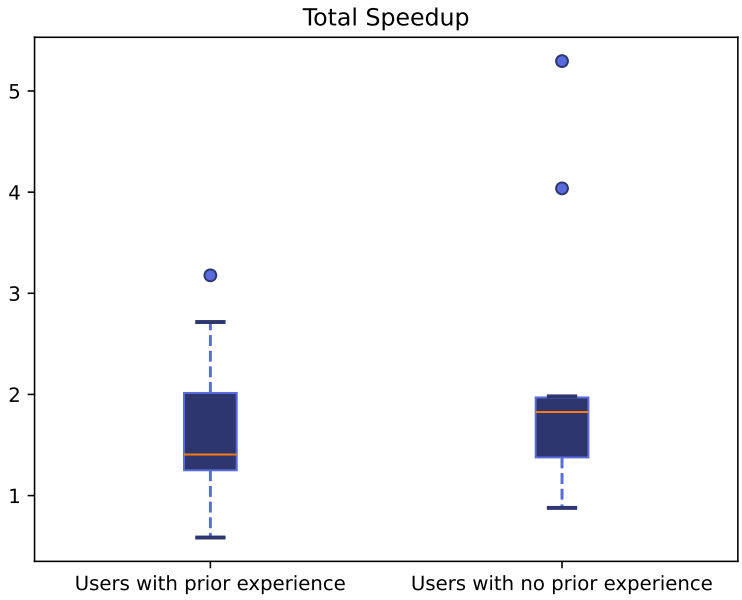


Figure 5.8: Comparing total speedup between participants with prior annotation experience with other tools and participants with no prior experience.

Table 5.1: Accuracy metrics obtained for leaf 1.

		Experience with other tools	
		No prior experience	Prior experience
Mean IoU	Manual	$98.6\% \pm 0.5\%$	$99.0\% \pm 0.2\%$
	AI	$98.4\% \pm 0.4\%$	$99.0\% \pm 0.3\%$
Mean Boundary IoU	Manual	$95.5\% \pm 1.3\%$	$96.7\% \pm 0.6\%$
	AI	$96.0\% \pm 0.9\%$	$96.4\% \pm 0.8\%$

We interpret the participants' similar accuracy regardless of prior expertise as an

Table 5.2: Accuracy metrics obtained for leaf 2.

		Prior experience with other tools	
		No prior experience	Prior experience
Mean IoU	Manual	$98.6\% \pm 0.5\%$	$98.9\% \pm 0.2\%$
	AI	$98.8\% \pm 0.4\%$	$98.9\% \pm 0.1\%$
Mean Boundary IoU	Manual	$95.5\% \pm 1.5\%$	$96.5\% \pm 0.6\%$
	AI	$96.3\% \pm 0.8\%$	$96.5\% \pm 0.3\%$

indication of the proposed software's ease of use and learning, which is consistent with the usability form's results. Aside from that, we consider that we have met the time reduction goal, as we achieved a median speedup of 1.5× in the general case. As a result, we believe this current study to have completed its primary objectives satisfactorily.

6 Conclusion

Deep learning models have contributed to significant advances in some artificial intelligence tasks over the last decade. To perform supervised learning, these models are fitted with large amounts of data that must be carefully selected and annotated by humans. As a result, long hours of human labor are required to create a high-quality dataset, which can be a bottleneck for further deep learning advancements.

In this context, this work looked into how to diminish annotation time when creating image datasets. Our proposal was to semi-automate the annotation task. We build a software that makes use of a previously trained promptable segmentation model and converts the binary mask predictions to editable polygons. By doing so, we reduced annotation to prompting and correcting prediction errors, with the expectation that humans would take less time to do it than they take in manual annotation.

We evaluated our software tool with 20 participants who belong to the computer graphics or computer vision fields. We verified a mean speedup of $1.9\times \pm 1.1\times$ and a median speedup of $1.5\times$ using our proposed AI tools over manual annotation. Therefore we conclude that this work was successful in its main objective of reducing the amount of labor (measured in time) required for annotating images.

The other objectives for our tool included user-friendliness, a low learning curve, and the inclusion of an exportation feature. Considering the experiences reported by participants through our user experience questionnaire, we conclude that these objectives were also met. Furthermore, we observed that participants with no experience in annotation annotated faster than participants with experience in other tools, while maintaining comparable accuracy. This observation provides additional evidence of our proposed tool's ease of use and low learning curve.

During the development phase, our proposed software assisted in the annotation of a dataset consisting of images of bean leaves. However, the software can still be improved and applied to other datasets. For further work, we have to attend the feedback provided by the evaluation participants and keep improving the software. We intend to

extend some currently available features, such as making available other deep learning models beyond HQ-SAM, and adding support to other annotation formats. Besides that, we judge it would be beneficial to add support to multiple simultaneous users annotating the same image, i.e, make the software a collaborative tool.

Bibliography

ALROOBAEA, R.; MAYHEW, P. J. How many participants are really enough for usability studies? In: IEEE. *2014 Science and Information Conference*. [S.l.], 2014. p. 48–56.

BOONSRI, P.; LIMPIYAKORN, Y. Semi-automated image annotation for cannabis seed gender detection model. In: IEEE. *2023 IEEE 3rd International Conference on Software Engineering and Artificial Intelligence (SEAI)*. [S.l.], 2023. p. 189–193.

CASTREJON, L. et al. Annotating object instances with a polygon-rnn. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 5230–5238.

CHENG, B. et al. Boundary iou: Improving object-centric image segmentation evaluation. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. [S.l.: s.n.], 2021. p. 15334–15342.

DOSOVITSKIY, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. In: *International Conference on Learning Representations*. [S.l.: s.n.], 2021.

DOUGLAS, D. H.; PEUCKER, T. K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, University of Toronto Press, v. 10, n. 2, p. 112–122, 1973.

FÎCIU, I. et al. Automatic annotation of object instances by region-based recurrent neural networks. In: IEEE. *2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP)*. [S.l.], 2018. p. 287–291.

HAGAN, M. T.; DEMUTH, H. B.; BEALE, M. H. *Neural Network Design (2nd Edition)*. 2. ed. [S.l.]: Martin Hagan, 2014.

HE, K. et al. Mask r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2961–2969.

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural networks*, Elsevier, v. 2, n. 5, p. 359–366, 1989.

KE, L. et al. Segment anything in high quality. In: *NeurIPS*. [S.l.: s.n.], 2023.

KE, L. et al. *Segment Anything in High Quality*. 2023. Accessed 09 Jul 2024. Available from Internet: ⟨https://github.com/SysCV/sam-hq⟩.

KIRILLOV, A. et al. Segment anything. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2023. p. 4015–4026.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, v. 25, 2012.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015.

LIKERT, R. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: SPRINGER. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. [S.l.], 2014. p. 740–755.

MCCARTHY, J. et al. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, v. 27, n. 4, p. 12–12, 2006.

MINAEE, S. et al. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 44, n. 7, p. 3523–3542, 2021.

NGUYEN, V.-A. *SAM Exporter*. 2023. Accessed 09 Jul 2024. Available from Internet: ⟨https://github.com/vietanhdev/samexporter⟩.

PHILBRICK, K. A. et al. Ril-contour: a medical imaging dataset annotation tool for and with deep learning. *Journal of digital imaging*, Springer, v. 32, p. 571–581, 2019.

PUGDEETHOSAPOL, K. et al. Automatic image labeling with click supervision on aerial images. In: IEEE. *2020 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2020. p. 1–8.

QIN, X. et al. Bylabel: A boundary based semi-automatic image annotation tool. In: IEEE. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. [S.l.], 2018. p. 1804–1813.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015.

SAMBATURU, B. et al. Scribblenet: Efficient interactive annotation of urban city scenes for semantic segmentation. *Pattern Recognition*, Elsevier, v. 133, p. 109011, 2023.

SILVA, K. G. F. da. *Uma Base de Imagens de Folhas de Feijão e uma Rede Neural Profunda para Estimativa Não-Destrutiva de Área Foliar*. Dissertação (Mestrado) — UNIVERSIDADE FEDERAL DE JUIZ DE FORA, 2023.

SMITH, M. L.; SMITH, L. N.; HANSEN, M. F. The quiet revolution in machine vision-a state-of-the-art survey paper, including historical review, perspectives, and future directions. *Computers in Industry*, Elsevier, v. 130, p. 103472, 2021.

SUZUKI, S. et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, Elsevier, v. 30, n. 1, p. 32–46, 1985.

TOPAL, C.; AKINLAR, C. Edge drawing: a combined real-time edge and segment detector. *Journal of Visual Communication and Image Representation*, Elsevier, v. 23, n. 6, p. 862–872, 2012.

TORRALBA, A.; RUSSELL, B. C.; YUEN, J. Labelme: Online image annotation and applications. *Proceedings of the IEEE*, IEEE, v. 98, n. 8, p. 1467–1484, 2010.

VASWANI, A. et al. Attention is all you need. *Advances in neural information processing systems*, v. 30, 2017.

YING, X. An overview of overfitting and its solutions. In: IOP PUBLISHING. *Journal of physics: Conference series*. [S.l.], 2019. v. 1168, p. 022022.

YU, J. et al. Superpixel segmentations for thin sections: Evaluation of methods to enable the generation of machine learning training data sets. *Computers & Geosciences*, Elsevier, v. 170, p. 105232, 2023.

ZHAO, Z.-Q. et al. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, IEEE, v. 30, n. 11, p. 3212–3232, 2019.

Annex A – Ethics Committee's Opinion

UNIVERSIDADE FEDERAL DE
JUIZ DE FORA - UFJF

PARECER CONSUBSTANCIADO DO CEP

DADOS DO PROJETO DE PESQUISA

Título da Pesquisa: Desenvolvimento e Avaliação de um Software de Anotação de Conjuntos de Dados de Imagens Auxiliado por Inteligência Artificial

Pesquisador: LUIZ MAURILIO DA SILVA MACIEL

Área Temática:

Versão: 2

CAAE: 78839724.3.0000.5147

Instituição Proponente: Departamento de Ciência da Computação

Patrocinador Principal: Financiamento Próprio

DADOS DO PARECER

Número do Parecer: 6.846.107

Apresentação do Projeto:

Trata-se de segunda versão do projeto submetida ao CEO. A partir de informações do arquivo Informações Básicas, "este projeto pretende ajudar a reduzir a quantidade de horas e de trabalho humano necessários para a anotação de um conjunto de dados de imagens. Para isso, se propõe o desenvolvimento de um software de anotação semi-automatizada, no qual um modelo de inteligência artificial auxiliará o anotador humano no cumprimento da tarefa. Após o desenvolvimento, a aplicação será avaliada com discentes de graduação ou pósgraduação da UFJF. A avaliação se dará por medir e comparar o tempo de conclusão da tarefa de anotação com e sem o auxílio da inteligência artificial, além de um questionário sobre a percepção individual dos participantes sobre a aplicação".

Objetivo da Pesquisa:

Conforme o pesquisador, o objetivo primário é "reduzir a quantidade de horas de trabalho humano necessárias para a anotação de um conjunto de dados de imagens. Para isso, pretende-se desenvolver e avaliar uma aplicação capaz de semi-automatizar o processo de anotação através do uso de inteligência artificial".

Por sua vez, os objetivo secundários são: "desenvolver um software de anotação de fácil utilização e baixa curva de aprendizado; conseguir exportar os dados de anotação em diversos formatos adequados para a utilização; avaliar o impacto da utilização da aplicação pelos anotadores".

Endereço: JOSE LOURENCO KELMER S/N
Bairro: SAO PEDRO **CEP:** 36.036-900
UF: MG **Município:** JUIZ DE FORA
Telefone: (32)2102-3788 **E-mail:** cep.propp@ufjf.br

Avaliação dos Riscos e Benefícios:

O discorrido sobre riscos pelo pesquisador foi são: "os riscos aos participantes dessa pesquisa são mínimos. Não serão coletados dados sensíveis ou pessoais e não haverá uma identificação nominal nos resultados das anotações ou nas respostas do questionário. Contudo, existe uma pequena chance de identificação. Caso alguém saiba a ordem em que os participantes participaram da pesquisa, pode-se usar os metadados automaticamente gerados de hora de criação dos arquivos de anotações ou respostas do questionário para identificar quais resultados pertencem a cada participante. Considerando que a pesquisa contará com etapa em ambiente virtual, também devem ser considerados os riscos inerentes a esse ambiente, como a possibilidade de vazamento dos dados preenchidos no formulário e o uso dos dados pela empresa responsável pela plataforma utilizada para coleta. Para minimizar esses riscos, os dados coletados serão baixados da nuvem e mantidos apenas no computador pessoal dos responsáveis pela pesquisa".

Quanto aos benefícios:

Os participantes serão beneficiados por aprenderem a usar e passarem a dispor da ferramenta desenvolvida neste trabalho, a qual tem potencial para acelerar ou viabilizar a anotação de novos dados a serem utilizados pelos participantes em suas próprias pesquisas. Espera-se que com os resultados o tempo necessário para a anotação de conjunto de dados seja reduzido, requerendo menos horas de esforço de anotadores humanos".

Comentários e Considerações sobre a Pesquisa:

O projeto está bem estruturado, apresenta o tipo de estudo, número de participantes, critério de inclusão e exclusão. As referencias bibliográficas são atuais, sustentam os objetivos do estudo e seguem uma normatização. O orçamento lista a relação detalhada dos custos da pesquisa que serão financiados com recursos próprios.

Considerações sobre os Termos de apresentação obrigatória:

Estão anexados o TCLE, o currículo Lattes da pesquisador e equipe, e declaração de infraestrutura e folha de rosto assinada pelo diretor do ICE.

Recomendações:

Não há.

Conclusões ou Pendências e Lista de Inadequações:

Diante do exposto, o projeto está aprovado, pois está de acordo com os princípios éticos norteadores da ética em pesquisa estabelecido na Res. 466/12 CNS e com a Norma Operacional Nº 001/2013 CNS. Data prevista para o término da pesquisa: 10/07/2024.

Considerações Finais a critério do CEP:

Diante do exposto, o Comitê de Ética em Pesquisa CEP/UFJF, de acordo com as atribuições definidas na Res. CNS 466/12 e com a Norma Operacional Nº001/2013 CNS, manifesta-se pela APROVAÇÃO do protocolo de pesquisa proposto. Vale lembrar ao pesquisador responsável pelo projeto, o compromisso de envio ao CEP de relatórios parciais e/ou total de sua pesquisa informando o andamento da mesma, comunicando também eventos adversos e eventuais modificações no protocolo.

Este parecer foi elaborado baseado nos documentos abaixo relacionados:

Tipo Documento	Arquivo	Postagem	Autor	Situação
Informações Básicas do Projeto	PB_INFORMAÇÕES_BÁSICAS_DO_PROJETO_2294945.pdf	09/05/2024 15:11:33		Aceito
Projeto Detalhado / Brochura Investigador	PROJETO_DETALHADO.docx	09/05/2024 11:59:58	PAULO VICTOR DE MAGALHAES ROZATTO	Aceito
TCLE / Termos de Assentimento / Justificativa de Ausência	TCLE.docx	09/05/2024 11:59:04	PAULO VICTOR DE MAGALHAES ROZATTO	Aceito
Outros	Lattes_Paulo.pdf	04/04/2024 20:04:17	LUIZ MAURILIO DA SILVA MACIEL	Aceito
Outros	Lattes_LuizMaurilio.pdf	04/04/2024 20:03:35	LUIZ MAURILIO DA SILVA MACIEL	Aceito
Declaração de Instituição e Infraestrutura	DeclaracaoInfraestrutura.pdf	04/04/2024 19:56:58	LUIZ MAURILIO DA SILVA MACIEL	Aceito
Folha de Rosto	folhaDeRosto_assinado.pdf	18/03/2024 13:27:22	LUIZ MAURILIO DA SILVA MACIEL	Aceito
Outros	formulario_avaliacao.docx	15/03/2024 11:32:47	PAULO VICTOR DE MAGALHAES ROZATTO	Aceito

Situação do Parecer:

Aprovado

Necessita Apreciação da CONEP:
Não

JUIZ DE FORA, 23 de Maio de 2024

Assinado por:
Patrícia Aparecida Baumgratz de Paula
(Coordenador(a))

Endereço: JOSE LOURENCO KELMER S/N
Bairro: SAO PEDRO **CEP:** 36.036-900
UF: MG **Município:** JUIZ DE FORA
Telefone: (32)2102-3788 **E-mail:** cep.propp@ufjf.br