Automatic Labeling/Annotation of Image and Video Data for Feature Extraction

Design Document

Team number: sdmay21-32
Advisers: Chandan Kumar, Ali Jannesari

Team Members: Michael Boyle, Dylan Hodge, Dylan Smith, Jeff Kinard, Mark Endeshaw, Sam Hassebroek

Team Email: sdmay21-32@iastate.edu

Team Website: https://sdmay21-32.sd.ece.iastate.edu

Executive summary

Development Standards & Practices Used

List all standard circuit, hardware, software practices used in this project. List all the Engineering standards that apply to this project that were considered.

- Agile Development Practices
- Test Driven Development
- IEEE 829 – IEEE Standard for Software and System Test Documentation
    - This standard discusses how different testing processes should be done to ensure different levels of software integrity. It does not detail specific testing methods, but rather a general guideline of what should be tested and how to determine what should be tested.
- IEEE 1633 – IEEE Recommended Practice on Software Reliability
    - This standard lays out an appropriate life-cycle for ensuring reliable software. It lays out recommendations for reliability testing and what to look for in one's product to ensure proper reliability.
- IEEE 1074 – IEEE Standard for Developing a Software Project Life Cycle Process
    - This standard lays out the process for creating a life-cycle for a software project. It lays out example models to use as starting points for creating a custom lifecycle plan.

Summary of Requirements

- There shall be…
    - An algorithm to automatically annotate all objects from any image.
    - An extension of the algorithm to annotate objects from video.
    - A further extension to include masking.
    - A backup algorithm to annotate objects missed by the main algorithm.
    - A method of validating the results for each algorithm.
    - Plenty of technical documentation and well-commented code.
    - A focus on enhancing annotation accuracy.
    - A focus on reducing the processing time in large scale video data by utilizing the algorithm.
- Our team shall meet all deadlines set by our instructor and our project contact.

Applicable Courses from Iowa State University Curriculum

- COM S 474/472

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

- Machine Learning methods

- Python coding
- Cocosynth
- Manual masking/annotation

Table of Contents

1 Introduction 1

1.1 Acknowledgment 1

1.2 Problem and Project Statement 1

1.3 Operational Environment 1

1.4 Requirements 1

1.5 Intended Users and Uses 2

1.6 Assumptions and Limitations 2

1.7 Expected End Product and Deliverables 3

2 Project Plan 3

2.1 Task Decomposition 3

2.2 Risks And Risk Management/Mitigation 4

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria 4

2.4 Project Timeline/Schedule 6

2.5 Project Tracking Procedures 7

2.6 Personnel Effort Requirements 8

2.7 Other Resource Requirements 8

2.8 Financial Requirements 8

3 Design 9

3.1 Previous Work And Literature 9

3.2 Design Thinking 9

3.3 Proposed Design 9

3.4 Technology Considerations 11

3.5 Design Analysis 12

3.6 Development Process 12

3.7 Design Plan 12

4 Testing 13

4.1 Testing Description 13

4.2 Unit Testing 13

4.3 Interface Testing 13

4.4 Acceptance Testing 13

5 Implementation 14
6 Closing Material 14
6.1 Conclusion 14
6.2 References 15

List of figures/tables/symbols/definitions

Table 1: Risk Management Table 4
Table 2: Personnel Effort Table 8

Figure 1: Use Case Diagram 2
Figure 2: Task Decomposition Graph 3
Figure 3: Fall Semester Gantt chart 6
Figure 4: Spring Semester Gantt chart 7
Figure 5: Design Thinking Diagram 9
Figure 6: Proposed Data Pipeline for Automated Annotation 12

1 Introduction

1.1 Acknowledgment

We would like to express our gratitude to our graduate assistant Chandan Kumar for his assistance on our project, as well as to Dr. Ali Jannesari for providing us the opportunity to work on this project under his supervision.

1.2 Problem and Project Statement

Our goal is to create an automatic annotation program that has the capability to label, group, and create mask annotations of objects in an image or video. Currently, there exists object detection software that operates on trained data sets to detect and label images containing specific, pre-trained objects. In our case, we hope to create a solution that has the capability to perform similar functionality on any object in an image or video without requiring targeted machine learning beforehand. In our solution, these operations would be performed completely autonomously and identify any/all objects in an image or video without the need for user input.

1.3 Operational Environment

Our project is completely software-based and thus will not need to be constrained by a physical environment. It will be completely defined by the computers and operating systems that it can run on. We hope to develop something that is independent of its operating system and will be able to run as an independent python program.

1.4 Requirements

- There shall be…
    - An algorithm to automatically annotate all objects from any image.
    - An extension of the algorithm to annotate objects from video.
    - A further extension to include masking.
    - A backup algorithm to annotate objects missed by the main algorithm.
    - A method of validating the results for each algorithm.
    - Plenty of technical documentation and well-commented code.
    - A focus on enhancing annotation accuracy.
    - A focus on reducing the processing time in large scale video data by utilizing the algorithm.
- Our team shall meet all deadlines set by our instructor and our project contact.

1.5 Intended Users and Uses

The intended users of our project are our contact (Chandan Kumar) and our client (Ali Jannesari). Other Users could be considered the future students and researchers that will use and update our project. All of our users are using the project for their research and hopefully, people will later use it to label data without the lengthy process of manually training data sets.

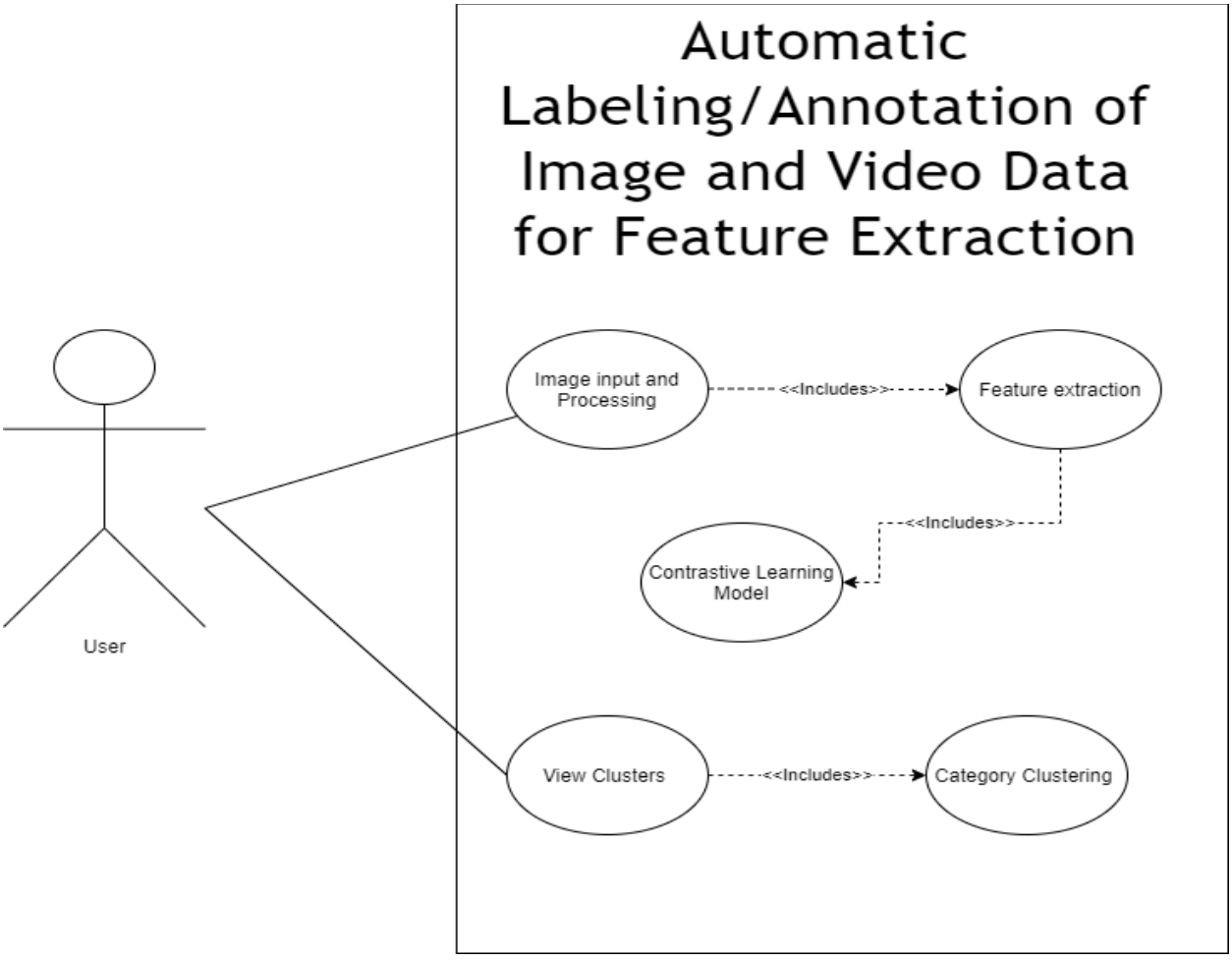

Figure 1: Use Case Diagram

1.6 Assumptions and Limitations

- We assume that:
    - there will be difficulties with extracting accurate images from low-level features images if we come across pixel prediction or semantic gap problems
- Our limitations are:
    - knowing for certain which regions of the image correspond to keyword in the trained data

1.7 Expected End Product and Deliverables

The main end project deliverable will be a written document and a python script. The deliverable will be given to the grad student overseeing the project at the end of the class. The documentation will be written not only on how to use the software but also how to continue developing the software after we have left and someone with no knowledge of the project.

2 Project Plan

2.1 Task Decomposition

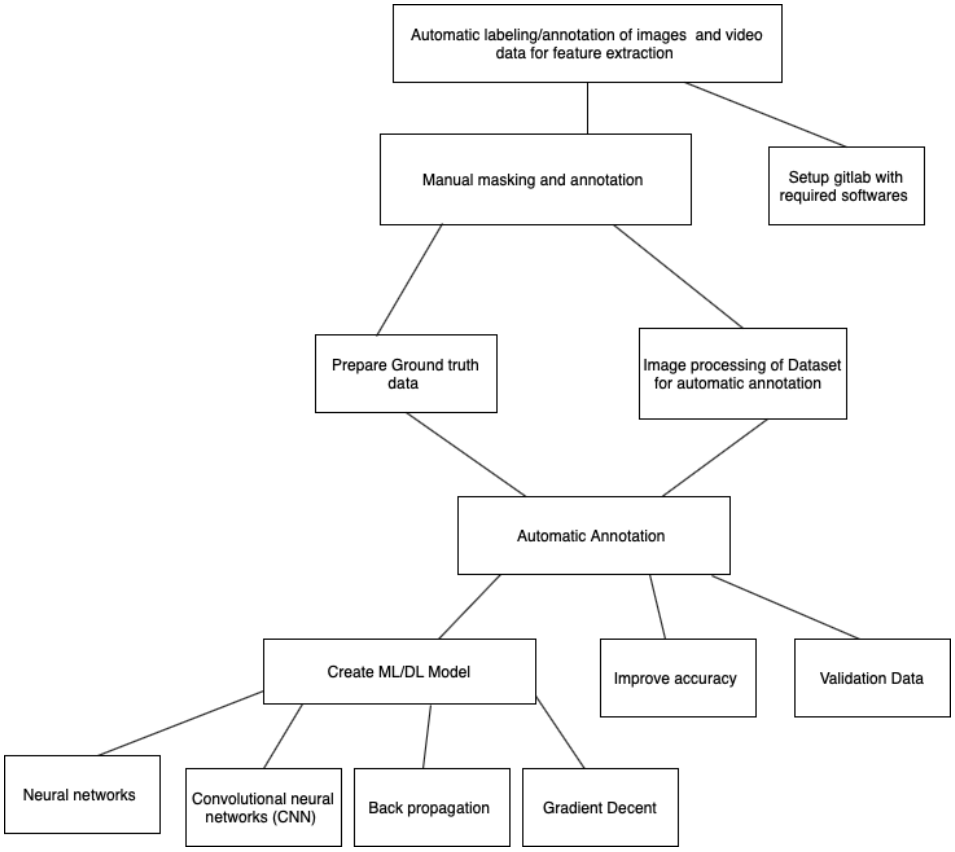

Figure 2: Task Decomposition Graph

1. Setting up GitLab with CI/CD pipeline, shell script to download required software and installation of required software including GIMP, python/python3, Anaconda Navigator, Cocosynth Library, Shapley, and MATLAB.
2. Manual masking and annotation, each team member annotates approximately 18 images.
3. Backup algorithm to annotate objects that are not annotated by the main algorithm
   3.1. Phase 1 and phase 2

I. preparing ground truth data
II. Image processing of the dataset

3.2. Phase 3

I. Create ML/DL model
II. Improve accuracy
III. Validation data

3.3. Phase 4

I. Documentation
II. Presentation

2.2 Risks And Risk Management/Mitigation

Risk description	Probability	Risk mitigation plan	Alternative tool, technology
Low visual recognition between ground dataset and test dataset	0.3	Increase the size of the training set or ground truth data	Using ML/DL to improve the quality of visual recognition
Main algorithm fails to annotate an image	0.4	Develop a backup algorithm	use multiple algorithms
Algorithm fails to detect object in video	0.5	Extend the main algorithm	Apply multiple object track finding algorithms

Table 1: Risk Management Table

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Setting and Installation of the required software

This project will be designed to run and compile in all major operating systems. All the required software and libraries will set up to run in any operating system so that the end-users can compile the project easily.

Manual annotation

Each member of the group did manual masking to have a better understanding of the algorithm and to use the annotated images for the next phase of the project. The manual masking accuracy will be measured by applying the concept of Neural Networks such as Convolutional Neural Networks (CNN) and Mask R-CNN.

Main and backup algorithms for automated annotation

For this project, we will have two algorithms, the main algorithm, and the backup algorithm. The main algorithm should be able to automatically annotate/label the objects inside an image or video. The automatic image annotation uses few trained data to annotate images in the dataset with a shorter time and with greater efficiency. In this algorithm the metadata is assigned to a digital image using appropriate keywords, it explores the correlation between features of the visual image and semantic meaning to draw a function using machine learning. The backup algorithm is to annotate images that are missed by the main algorithm.

This task has three phases, the first and the second phases are preparing the ground truth data and image processing of the dataset. This includes the following major subtasks,

- Model design: model design is the feature detection to describe local features and composition of the objects.
- Trained datasets: the set of trained images and their features to work with the model and to generate the features that match the given image.
- Test set: the set of images for testing against the trained datasets, this will help us to predict the accuracy of the model to get the correct matches.
- Classifier: classifies the given dataset into different classes for search and match optimization.
- Training and testing: use a set of images to check against the ground truth.

The third task is creating machine learning /deep learning models and validation of data. In this phase, we will apply the concept of Neural networks including Convolutional Neural Networks (CNN), Feed-Forward Networks, back Propagation, and Gradient Descent. Each phase of the project will be evaluated using the images that are manually masked and annotated.

2.4 Project Timeline/Schedule

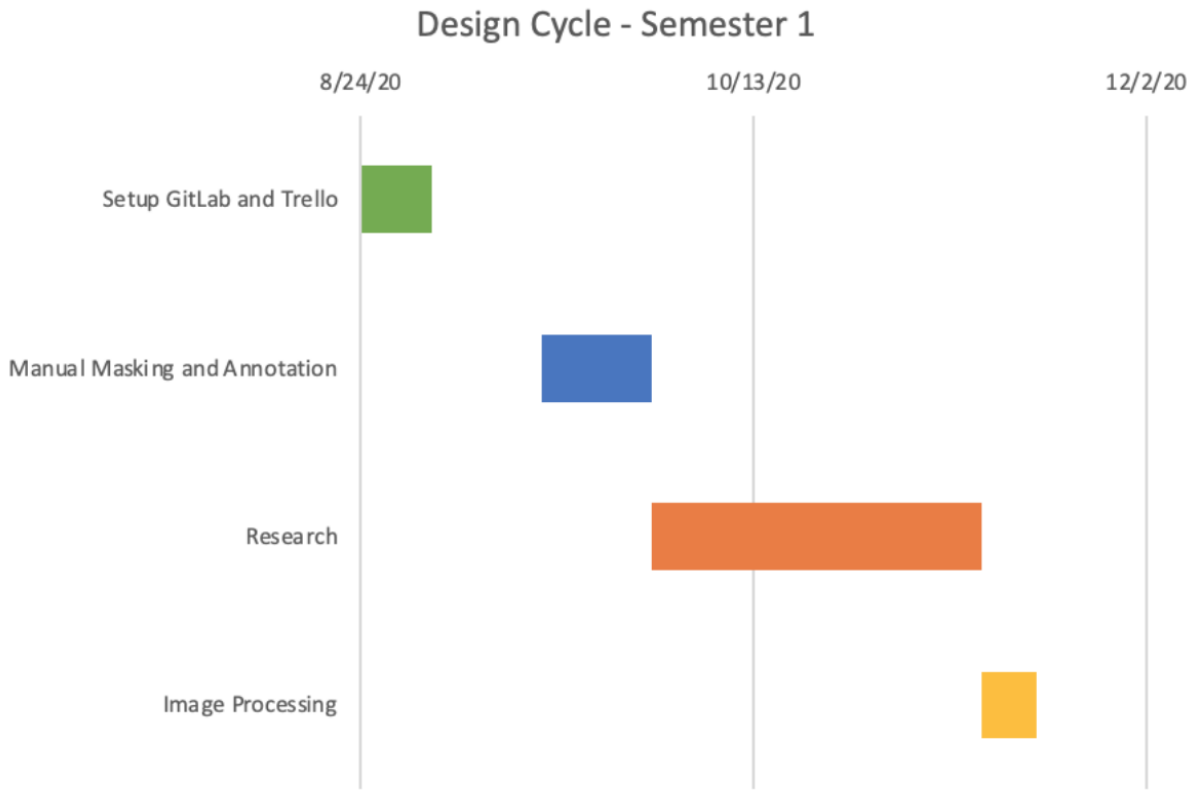

Figure 3: Fall Semester Gantt chart

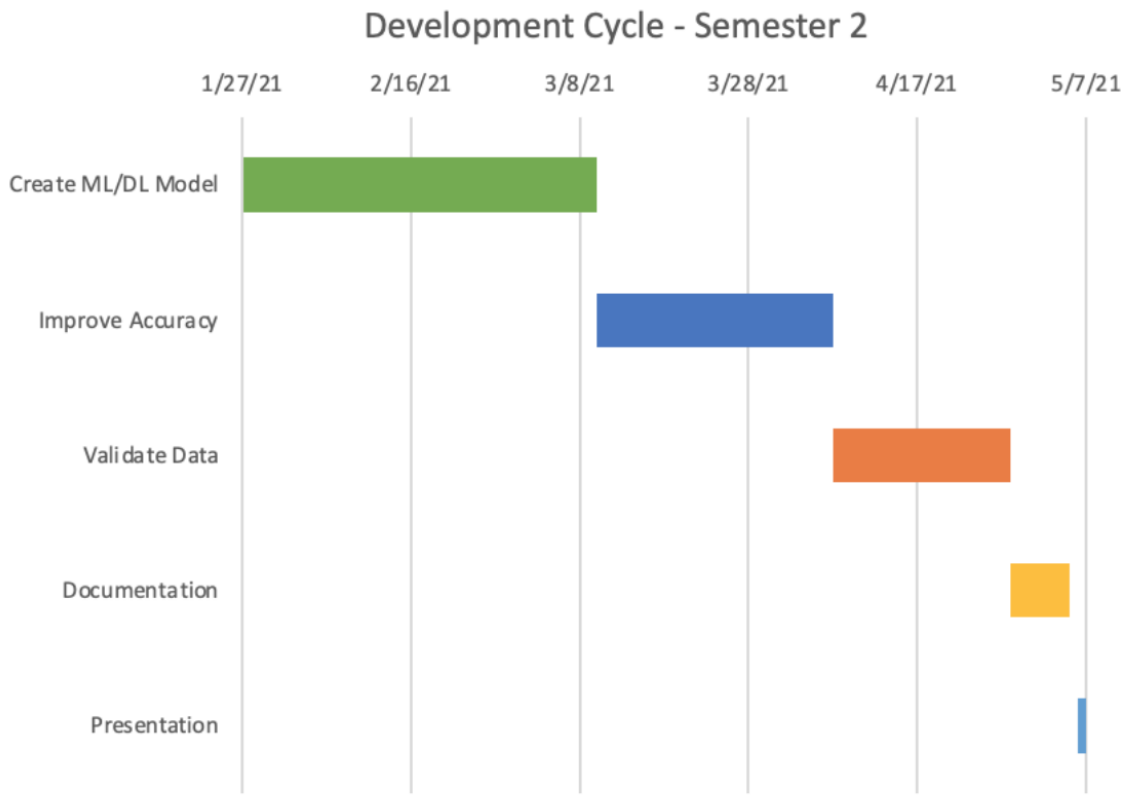

Figure 4: Spring Semester Gantt chart

2.5 Project Tracking Procedures

We are tracking the project progress using Trello to organize and to prioritize project tasks. Each series of tasks are listed on the Trello board, this helps us to break projects down into smaller tasks or create even more detailed to-do lists. The other task managing tool is GitLab which is more than collaboration and a change tracking tool. GitLab enables development teams to work in asynchronous environments, tracking the progress of the project, and working together on a different branch. The other feature of git is GitLab issues, this helps us to post questions on relevant issue threads, post updates, and notify any team member to get them involved in the project. We are also using GroupMe for daily communications and conversation that needs to happen quickly.

2.6 Personnel Effort Requirements

Task	Estimated Person-Hours
Manual masking and annotation (~100 images)	~100 person-hours
Prepare Ground Truth Data and Image Processing (Preparing for Automatic Annotation)	~50 person-hours
Automatic Annotation: - Create ML/DL Model - Improve Accuracy - Validation Data	~120 person-hours total
Documentation and Presentation	~80 person-hours

Table 2: Personnel Effort Table

Under ideal circumstances, each member of the team will take on work that requires similar amounts of effort to accomplish. Given our estimations of our project task requirements, each member would be required to give roughly 60 hours worth of work to help complete the project on time, for a total of about 350 person-hours to complete the project.

2.7 Other Resource Requirements

As far as other resources go, all we require is a VM with these capabilities:

- Ubuntu1804 OS
- 1 Core CPU
- 4GB Memory
- 50 GB Disk Memory

This will be provided by Iowa State University, and it will serve as a host for our GitLab runner.

2.8 Financial Requirements

As our project is entirely web-based, and requires no paid licenses or materials to conduct, including financial requirements is not necessary.

3 Design

3.1 Previous Work And Literature

As our project is research-based, many of the previous work is based on similar papers and topics. We have also spent time discussing with other people doing similar research and helping us to build on their ideas and modify them to fit closer to our project. We referenced an Oxford University academic study to gather insight on how we should cluster the images once detecting the objects inside of them [1]. We also referenced other scholarly articles to gather more information about automatic image annotation techniques and multi-label image annotations [2], [3]. We also discussed with other researchers and Ph.D. students, who were directed to us by our advisor, which allowed us to further refine the rough pipelines that we had from then papers and build out features specific to our research.

3.2 Design Thinking

A lot of our "define" aspects came directly from our advisor and how our project would fit into his research. Some of the other "define" aspects also came from who the project would be handed off to as they would need to be able to easily continue the work where we left off. Our "ideate" has been a large portion of our project as it is research and has not fully been done before. So, in our "ideate" phase, we have been planning many different pipelines and approaches. Many of our ideas and approaches are based on ideas. Many of these ideas are then presented to our advisor where he can give us feedback and help further define them into possible approaches that might actually work.

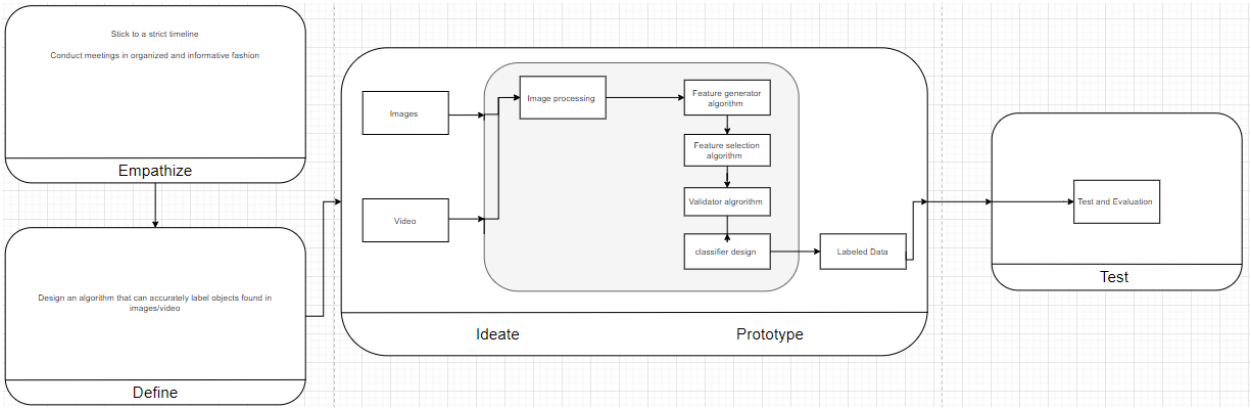

Figure 5: Design Thinking Diagram

3.3 Proposed Design

Our algorithm will take a contrastive approach to cluster high-dimensional feature vectors into similar instance groups. There are six main steps in the pipeline, and they are as follows.

i. Image processing

The algorithm will first take in an image and perform edge detection and blob extraction. Next, it will bound the blobs with a rectangle and crop the image accordingly. All cropped instances will then be passed into our contrastive learning model

ii. Contrastive learning model.

This section of the pipeline will be responsible for training an ML model capable of creating feature vectors that accurately represent different instance labels. In this manner, images that contain the same object instance will output similar feature vectors when pushed through the ML algorithm.

iii. Dimensionality reduction:

Once the feature vectors have been extracted from the ML model, they will need to undergo dimensionality reduction. These feature vectors will be in the rank of the hundreds, meaning they will contain hundreds of dimensions. Mapping this to a hyperspace and clustering is inherently very difficult. To combat this, the algorithm will use the tsne dimensionality reduction algorithm to map the feature vectors to a 2-dimensional plane.

iv. Clustering:

Once the feature vectors have been transformed, they will be clustered for classification. While the algorithm will not be able to say what a group of objects is exactly, it will be able to group all similar objects together in a common cluster.

v. Classification:

At this point, all the feature vectors have been clustered into groups of similar instances. Now, the algorithm will extract a representative from each cluster, grab the associated images, and prompt the user to specify the label of each object. This allows the algorithm to then label the clusters that each representative belongs to.

vi. Label

The algorithm now has labeled clusters that contain all the feature vectors. Since each of the vectors corresponds to a mask stored in memory from the image processing step, it is now possible to assign each of the masks the appropriate label. Now, all that is left is to write the labels in Coco format. This format is a JSON that contains image paths, label id’s, masks, bounding boxes, and other useful information. This JSON is then returned to the user so that they can undergo any ML experimentation they may need to perform with this labeled dataset.

3.4 Technology Considerations

There are three main technologies that will be sure to pose an issue to this project. Below is a list of said technologies with the current countermeasures the team is prepared to enforce should any technical issues arise.

i. Contrastive Learning

This algorithm relies heavily on a new Computer Vision architecture known as contrastive learning. Since this is a new practice, it is prone to unknown discrepancies. It will be imperative to the prosperity of the project to ensure that extensive research is performed to gather as much information about this technology as possible. It appears to be the only viable option given today's research and technology, so it is important that it is optimized to its full potential.

ii. Edge Detection

Since the blob extraction is done through bounding objects that stick out in an edge detection algorithm, it is easy for the blobs to come out inconsistent. Each edge detection algorithm has drawbacks, whether it be light in the image, or similar colors. It is important to check all the options and ensure the most consistent method is chosen. To combat inconsistencies, we have concluded that it may be necessary to use multiple edge detection algorithms and aggregate the results into high-confidence outputs. We have also thought about using a pre-trained Mask RCNN model to extract masks automatically, but this can be influenced by the model's lack of diversified instance options.

iii. Dimensionality Reduction

A major problem with clustering feature vectors for image classification is that the vectors tend to have well over a hundred dimensions. This is very difficult to cluster because there are so many hyperplanes to choose from to draw the cluster. It is conceptually hard to put order to an object that is so abstract, and the algorithm struggles with this feat as well. Luckily, to combat this, we will use the tsne algorithm to transform the feature vector into two dimensions. However, the problem with this is there is obvious information loss. It has to aggregate over one hundred features into just two. We are looking into alternative solutions should this pose a problem, but right now this solution is still better than the alternative of clustering the original feature vectors.

These are the main areas of worry at the moment. However, as with any machine learning project, there are issues to consider regarding the accuracy of the computer vision model itself. While this technology is very sound, with thousands of publications on the subject, it is still important to take measures to ensure consistency of results. This includes proper data processing and data augmentation, as well as optimized hyperparameter tuning.

3.5 Design Analysis

We are currently building our solution in 3.3 and we will later be able to analyze it to see how well it performed.

3.6 Development Process

Our development process follows that of an agile/scrum approach. In this way, we meet weekly with our advisor to assign tasks to each person and define what we want to accomplish each week. These weekly meetings function very similarly to sprint planning and spring retros with sprints length being that of a week. We also use Trello as a Kanban board to plan out the tasks each week and assign them to each person - similar to what is typically done in an agile environment.

3.7 Design Plan

Our design plan is an iterative process in which we are constantly modifying and refining our design based on what works and what does not. Each week, we develop ideas based on our readings and research from the previous week as well as refine our project based on the feedback from our advisor and his knowledge in machine learning and computer vision. The main constraint in our design is creating a pipeline that will function such that we can evaluate our approach and compare it against other solutions for the same problem.

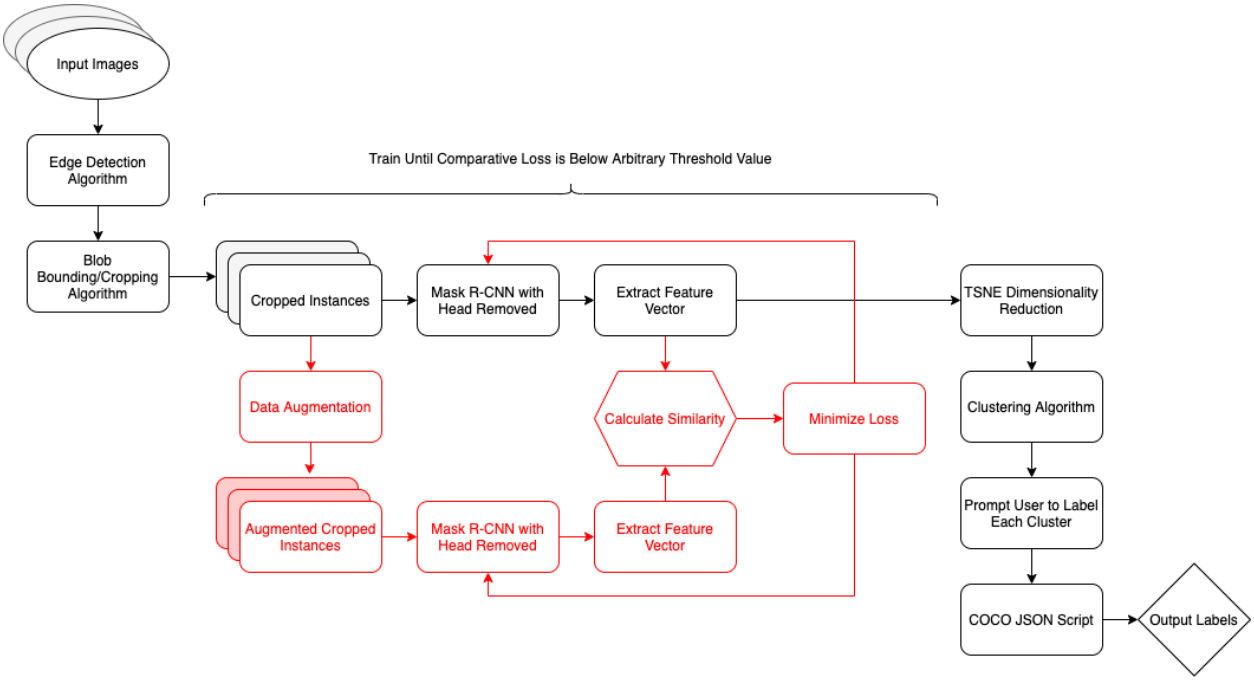

Figure 6: Proposed Data Pipeline for Automated Annotation

4 Testing

4.1 Testing Description

For our project testing, machine learning is a very difficult task and is currently under research. A large amount of our testing will just be making sure that each part of our pipeline runs and that the output is in the correct format that we intend. In the end, when we have a completed running pipeline, we will then be able to test the whole thing against standard accuracy metrics to determine the validity of our results and how well our pipeline compares with other approaches to the same problem.

4.2 Unit Testing

For this project, we are unable to test the detection algorithm itself. While running through the individual steps laid out in the pipeline, we will manually test to ensure that each step is performing as intended. For image processing, we will observe the output to make sure that the images are processed and cropped correctly. When testing the contrastive learning model, we will manually compare the feature vectors to see if they're similar. For clustering, we will manually check that all objects in a cluster are similar. For classification testing, we will manually inspect the classifications, and ensure that the samples are classified correctly. On the final step, we will make sure that the images have been labeled correctly according to how it was trained.

4.3 Interface Testing

Our units will be tested as the pipeline is slowly built out starting at the beginning of the pipeline with the object and feature detection and ending with the grouping of all detected objects. We will test this manually by making sure all the parts run as they should and by checking the output to see if it makes sense after the addition of new parts on to the pipeline.

4.4 Acceptance Testing

If we can fully run a pipeline that we have designed and implemented, that will be step one of the acceptance tests. After we know that we have a working and running code base, we can then compare the results to the pre-trained algorithms that we have developed to test against, as well as our ground-truth dataset. We can use the mAP (Mean Average Precision) metric to score the accuracy of our newly developed pipeline against the other two methods. This will not only test if the correct things are being labeled but also how close they are to the ground truth bounding box.

5 Implementation

As the Spring 2021 semester begins, we anticipate our implementation strategy to follow these core strategies:

- **Hybrid Waterfall/Agile Development Strategy:** breaking work down into “sprints” and meeting with our graduate assistant, Chandan Kumar, frequently to provide deliverables at regular intervals. Go through “stages” of development, such as design, implementation, and testing
- **Task Delegation Strategy:** During a sprint, work will be divided into tasks, and tasks will be placed in a backlog, where team members will then be able to self-assign work to complete.
- **Backlog Maintenance:** Both a Trello task board as well as a GitLab issues list will be used to create and backlog the work tasks described above
- **Project Management:** We will utilize a GitLab repository for version control, task delegation, and project management
- **Bi-Weekly SCRUM Meetings:** Independent from our advisors, team members will meet twice a week to discuss task progress, and any pending issues
- **PyTorch Framework:** Useful for training our ML pipeline once the project has been fully built out. Also allows us to save the model once it has been fully trained.
- **Mask R-CNN architecture as a backbone:** Allows us to have a R-CNN implemented so we do not have to custom build it just for this project.

6 Closing Material

6.1 Conclusion

This semester has been a strong baseline setting up our project and giving us something to compare against. A large portion of the work and time was spent doing research and learning more about machine learning and the various methods involved. We then started with manually labeling and masking the data set given to us, using that dataset we then trained two R-CNN (Mask R-CNN and Faster R-CNN). Next semester our main goal is to build-out the ML pipeline and test it against our pre-trained models. We will know when we have achieved our goals the performance of our newly built untrained pipeline meets or exceeds the metrics set by our work this semester.

6.2 References

[1] Xu Ji, J. Henriques, A. Vedaldi. “Invariant Information Clustering for Unsupervised Image Classification and Segmentation.” University of Oxford, 2019. [Online]. Arxiv, https://arxiv.org/pdf/1807.06653.pdf [Accessed Oct. 1, 2020]

[2] D. Zhang, M. Islam, G. Lu. “A Review on Automatic Image Annotation Techniques.” *Pattern Recognition,* Volume 45, Issue 1, pp. 346-362, Jan. 2012. [Online] ScienceDirect, https://www.sciencedirect.com/science/article/abs/pii/S0031320311002391 [Accessed Oct. 1, 2020]

[3] H. Le, T. Nguyen, D. Ngo-Tien. “Fully Automated Multi-label Image Annotation by Convolutional Neural Network and Adaptive Thresholding” 2016. [Online] ResearchGate, https://www.researchgate.net/publication/311621205_Fully_automated_multi-label_image_annotation_by_convolutional_neural_network_and_adaptive_thresholding [Accessed Oct. 1, 2020]