Milestone 1 Report

Team Deinonychus – Patrick Creighton, Luka Rogic, Christopher Lee, Juntong Luo

Main Goals

From our proposal, the main things that we wanted to accomplish for milestone 1 are: each member researches what they are in charge of; create repository, Jupyter notebooks, and web application files; write a detailed to-do list on GitHub. Each person is in charge of the project's main aspects as follows: Patrick, face recognition model; Luka, object detection model; Christopher, creating the web application using Flask; Juntong, Python backend for the web application. We also decided to work on a private repository, so that we can have more flexibility in modifying repository settings and setting up integrations. This repository can be found at ML-Image-Processor (github.com), and accessed using the GitHub account with username `colab-user-5546`, and password `cpen2021`. We will clone our private repository to the given team repository before milestone deadlines. As for the task of writing a detailed issues list on GitHub, we decided that it would not be necessary until we begin revising existing code, which will occur some time before milestone 3. However, some issue templates were created to help with this task.

Face Recognition Model

The first section of the face recognition notebook in the ML notebooks folder details how to clone a subdirectory of a private GitHub repository to Colab, useful for automatically importing curated datasets in the future. Then, some initial explorations were made on the Labelled Faces in the Wild dataset, which helped with implementing the function that reduces the number of dataset classes and samples to what we specify. The Training and Results section were mostly copied from a past project cited in the notebook. We were able to reproduce this project's results, but encountered multiple difficulties after processing the dataset using the `reduceClassesAndSamples` function. After consulting the professor during office hours, we concluded that the next course of action would be to form the dataset from scratch, which would be necessary anyways to use our own images. For milestone 2, we will also explore code from other past successful projects, conduct further research on support-vector machines, and aim to get our face recognition model fully functional.

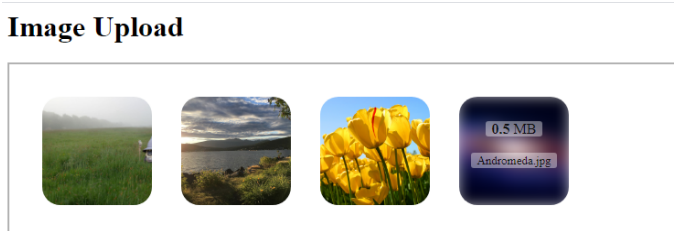
Object Detection Model

In the project proposal the main candidate for our model was Yolov5, but we also wanted to consider other models such as the Faster R-CNN or even Yolov4 (which is generally faster than Yolov5). After some research we decided to use Yolov5 because it is a faster model, simpler to figure out, and still achieves similar accuracy to slower models. It struggles with objects that are small and close together, which are not a big concern for our use case. There are several different versions of the Yolov5 model, from their smallest, Yolov5s to biggest, Yolov5x The bigger the model, the more accurate it should be, and the slower training gets. We plan to start with the Yolov5s model and move on to the bigger versions if our results are not accurate enough. We also ran the Yolov5s model with help from Roboflow's datasets and tutorial to get a sense of how to use the model and how fast it is. The Yolov5s model trained on 603 images for 100 epochs in ~10 minutes and the Yolov5x model trained on the same images and epochs ran in ~45 minutes, which gives us a good training runtime benchmark (both versions had similar

accuracy). Some challenges that we might encounter is that the tutorial model outputs images with bounding boxes drawn on, which is not desired for our final product. In the next milestone, we hope to create our datasets so we can begin selecting and training our model.

Flask Web Application

We chose to use Flask to implement our application, as described in the proposal, due to the vast amount of tutorials and resources available online for it, along with the minimal changes required in Python code to incorporate the machine learning back-end with the application interface. We also considered using a Python desktop application, as it would be more efficient for working with a large number of images (no need for uploading/downloading images). However, building a Python GUI would be much more difficult because we could not find any applications that could serve as our starting point. The goal for this first milestone when it came to the webpage was to add the very basics necessary for our app: A working webpage and a basic image uploader. Code was copied from (1) to create the image uploader but in later milestones, we will have to adapt upon this basic code to achieve our more complex end-product. The files necessary to create the web application are provided in the GitHub repository and consist of main.py, app.yaml, requirements.txt, the directories 'uploads' and 'templates', and main.html (in templates directory). Our webpage currently consists of a box which users can upload images (drag into box, or select from file explorer), showing a preview of the image uploaded.



We were able to host this webpage on Google App Engine (with the guidance of (2)), which we will further look into in future milestones. For the web application part of the project, we are on track with our milestone goals but a minor problem we are currently facing is that only JPG image files are supported. A potential quick solution to this could be to convert all input images to JPG files. For the next milestone, we hope to allow users to download images from our webpage, which will be one step towards our end goal of automatically and manually modifying image tags, and returning these to the user. As well, we hope to begin thinking about the design of the application.

(1) https://blog.miguelgrinberg.com/post/handling-file-uploads-with-flask
(2) https://realpython.com/python-web-applications/

Python Backend

We looked into how tagging works for images and kept our findings in the file "Backend/Metadata for Tags". It includes the compatibility of different metadata related to image tagging. A convenient Python library we found for editing image metadata is pyexiv2, which can edit all types of metadata and runs on both Windows and Linux. In addition, we wrote a demo Python script named "tagall.py" in "Backend/ImageTagger", which adds random tags to images under the same folder. For milestone 2, we hope to figure out how to use the output of the ML models we are working with as the input to the image tagger.