Project Plan
Group: Humanity's Last Hope
Austin Williams
Brandon Mommsen
Ty Bergstrom
CSCE A401
September 10, 2020

Our Project: A web app that allows users to connect a video stream and get real-time statistics on mask usage.

1. Overview

- **Description:** An API that takes an input video stream and uses a machine learning model to determine people wearing and not wearing mask in the video in order to track statistics for face mask usage of people in the feed over a period of time. The results would be served on a webpage. The page would show metadata (including timestamp, camera location, etc.), processed images (the video stream with bounding boxes around people), and statistics of the percentage of people wearing masks over the time period. Our API would live on a dedicated server hosted in Google Cloud.
- **Use cases:**

1. A business or restaurant owner wants to keep track of how many customers wear masks, possibly to ensure compliance with laws. They place a web cam at their front door and stream the video feed connected to our app, and our app provides the mask usage statistics.
2. A user logs in and forwards a stream from a public webcam, such as a live stream from a bus stop or town square. Our app can let them know the mask usage statistics for that location, possibly so they can make a choice whether they feel safe before going out to that location.
3. A user loads a stream from their phone camera to get a quick summary of mask usage of people nearby.

2. Technical description

Languages, Tools, & Technology

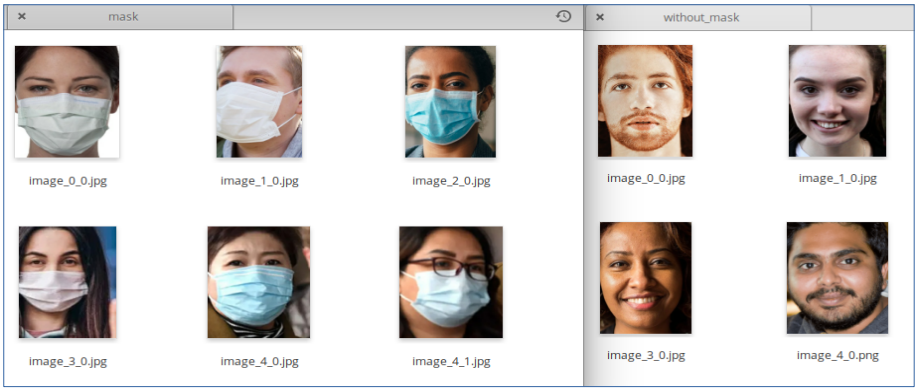
- Python (Tensorflow, Keras, OpenCV) for Machine Learning
- Bash scripting to help pre-processing, and tools such as Photoshop to help building the datasets.
- Python with a web framework (Flask or Django) for building the backend.
- HTML/CSS/Javascript for the web page.
- Google Cloud for server instance (probably Linux).

Machine Learning

- **To detect wearing masks**
- We noticed that we need at least two different models and datasets for different use cases.
- The first model concerns using low quality video streams. Faces will be small, blurry, and pixelated. We would not use face detection or make any classification. We would instead build an object detection model. The dataset would have to be faces cropped from video streams in order to capture faces as they appear in video. We would train a neural network to detect these two “objects,” the faces with masks, and the normal faces without masks. Thus, every frame of an input video would be scanned for these two “objects” and return any detections above a probability threshold. Another implementation might first use a pre-trained human detection model, and then scan only the human Region Of Interest (ROI) for a face with or without a mask.

An example of the kind of dataset we would have to build. Source: https://tryolabs.com/blog/2020/07/09/face-mask-detection-in-street-camera-video-streams-using-ai-behind-the-curtain/

- The other model concerns video streams with people’s faces up close to the camera, where the model would have a chance to get a clear capture of each face. For this implementation, we would first use a pre-trained face detection model, which works fairly well with faces wearing masks. The extracted faces would be shown to our model, and our model makes a binary classification whether a face wearing a mask or not. The dataset would include normal images of people’s faces wearing and not wearing masks, in addition to higher quality crops from video streams. However, we would probably attempt to include object detection for masks, instead of relying only on face detection.



An example of a dataset that leverages face detection. Source: personal screenshot.

- In conclusion, we need two models with their own datasets. One satisfies the requirements for pointing a camera at a crowd in a low quality stream and trying to detect what we are looking for. The other takes advantage of different requirements for scanning more clearly visible faces at closer range. The different datasets reflect what we expect the models will see.

- **Some ML challenges**
- Data collection: we need to find good data, collect enough data, and find enough time for data collection and processing. It will be more challenging and time consuming to get the faces-cropped-from-video-stream dataset, and to make sure it is the right dataset for the kind of streams our model will encounter.
- We need to set up a GPU on our server and make sure that we meet performance requirements for working with input video.
- Security: we need to foresee the hacks that people might use to trick our model, such as just covering your mouth with your hand, or covering your face with a large photo of a face wearing a mask. This requires training the models to detect and reject fakes.
- Neural network selection and hypertuning: we will need to take care to get the most accuracy out of our models, especially the object detection model.

- **Other possible ML features, as time allows:**
- We could build an age prediction model to exclude children from the mask wearing statistics, because some businesses do not require children to wear masks. It would be another ML project that requires a dataset of images of children's faces and everyone else, and training with a net to make the binary classification on input faces. It would only apply when our models detect faces without masks.
- We could include another classification of what kind of mask a person is wearing. It would be another ML project with a categorical classification. For a face wearing a

mask, we would predict whether it is a medical/surgical mask, fabric mask, homemade mask, bandanna, etc. It would require a dataset with enough samples of people wearing each class of mask. This might be combined with helping determine whether a person is wearing a mask or just covering their mouth with their hand.

App Front End

The frontend client provides an interface for three sources of video streams:

1. Video file upload
2. Internal device camera or usb camera
3. A stream from a web page

In each case there is an associated view to facilitate setup of the stream source. When setup is complete, streams can be sent to the backend. The results (metadata, processed images, and statistics) are displayed on the page in the client browser.

Back End

The backend receives the images in the connected stream and passes them through our data pipeline. It would automatically determine and adjust which model to use, based on factors such as image quality or polling for face detection. The pipeline uses the stream as input for our machine learning models, via a classification script.

The backend will also handle object tracking. In order to keep statistics on mask usage, we need to implement object tracking to track people across frames in a stream. We are still in the process of researching this, and it is likely more difficult than we can describe at this time.

The backend processes and returns image metadata, modified images, and calculated statistics to the frontend.

References

https://tryolabs.com/blog/2020/07/09/face-mask-detection-in-street-camera-video-streams-using-ai-behind-the-curtain/

This source mostly provides an overview of how to implement the kind of project we would like to do. It contains no links to source code or datasets.

3. Schedule for next 3 months

Week	ML	Front End	Back End	Misc.
9/8/2020	Research		Research	Project plan due
9/15/2020	Research, data collection / processing	Learn about front end dev	Learn about back end dev, choose framework	
9/22/2020	Data collection / processing	Begin web app development	Begin web app development	Set up Google cloud instance
9/29/2020	Data collection / processing, begin training	Web app development	Web app development	Begin building api that works with model
10/6/2020	Training, hypertuning	Web app development	Web app development, object tracking research & human detection	Set up GPU
10/13/2020	Training, hypertuning, testing, data collection / processing as needed	Frontend route / view for uploading video / process thru api	Object tracking development	Integrate object tracking with ML, set up classification with input video
10/20/2020	Training, hypertuning, testing, data collection / processing as needed	Frontend route / view for connecting internal camera / process thru api	Object tracking development & improvements	Integrate api with ML model, testing
10/27/2020	Testing & tuning as needed	Frontend route / view for getting video from web page / process thru api	Calculate statistics, process output	Integrate frontend with backend, testing
11/3/2020	Add features (age prediction, mask type classification)	Display statistics, display output	Calculate statistics, process output	Set up continuous integration if desired
11/10/2020	Testing & tuning as needed	Display statistics, display output	Testing	Testing

11/17/2020	Testing	Testing	Testing	Testing
11/24/2020		Testing	Testing	No additional features, refactoring, testing
12/1/2020		Testing	Testing	Testing
12/8/2020				Finals week, project complete

A relative timeline for some of the ML and backend:

This is a blend of tasks that are mostly separate but related and also dependent on each other to begin progress on the next step, which is not reflected as well in the above schedule. We will probably use smaller timelines schedules for shorter time frames like this as we go. Additionally, some items will be ready to integrate with others before they are polished.

ML	**Video / Detection / Classification**	**Object Tracking**	**Statistics**
Data collection			
Data processing			
Training	*Prep: Need GPU*		
Hypertuning & Testing	*Prep: Use a pre-trained model to detect people in input video*	*Prep: Track people across frames*	
Model ready ~10/13/2020	*Prep: Use our model to detect and / or classify faces with & without masks in video*	Track faces with & without masks across frames	*Prep: count how many people have been detected in video*
(Continue testing & tuning)		**Object tracking ready ~10/20/2020**	Count how many faces with & without masks have been detected in video
(Continue development as necessary)		(Continue development as necessary)	**All ready for the front end to use ~10/27/2020**
			(Continue development as necessary)