Developing a facemask detection web application using TensorFlow and Flask

Overview:

The global coronavirus pandemic has necessitated new public policies in the interest of safety. Based on overwhelming scientific evidence of the effectiveness of facial masks in limiting the spread of Covid-19, many cities and states within the United States have implemented policies mandating facial coverings in public spaces such as retail businesses and restaurants. (Peeples, 2020) Computer vision technology provides a solution for enforcing and measuring compliance with these public mandates. The purpose of this project was to develop a facemask detection application using a real-time camera feed as the input.

Data Acquisition:

Developing the application first involved training a model that could correctly classify facial images. For this task images were selected from the 'Face Mask Classification' dataset published on Kaggle. (Makwana, 2020) The dataset consisted of 220 images of people wearing masks and 220 not wearing face masks for a total of 440 images. Approximately 20 percent of the images were used in a validation set for model training with the remaining images used for the training dataset.

Model Architecture:

Model training utilized transfer learning in Tensorflow 2.0 using MobileNetV2 as a convolutional base. The bottom layers of the convolutional base were frozen with the remaining layers free to adjust based on model training. Relatively high validation accuracy exceeding 96% was achievable within 5 epochs. All training was performed within a Google Collab notebook with the model saved to Google Drive.

Approach and implementation:

The decision was made to develop the classifier using the Flask web application Python library. Using flask ensured the camera feed, overlaid with classification results, could be displayed to users without requiring physical access to the machine. The application works by performing a continuous analysis of frames captured by the device camera. Performing classification on a camera feed that clearly did not contain a face did not seem helpful. The application first performs face detection on a camera image using a face detection model from the open-cv Python library. This model allows for detecting multiple faces within the same image. However, making a prediction for more than one image was prone to errors. For example, an object in the image incorrectly identified as a face would receive a face-mask prediction. To partially address this problem, only identified faces with the largest returned dimensions receive a prediction.

If no face is identified in a camera frame, a response is overlaid on the web video feed indicating no faces were identified in the image. In the event of one or more faces, the largest face is overlaid with a rectangle reflecting the dimensions returned from the face detection function. Before a prediction is generated, a distinct image is created consisting of only the portion of the image containing the dimensions of the identified face. This is done to ensure that only relevant image information is used in rendering a prediction. Once the application is loaded with the Tensorflow model, predictions can be returned through a call to the run_prediction function. The function returns an array of two items consisting of the probability that the image contains a face with a facemask and the probability that the image contains a face without a face-mask. If the probability is higher for the image containing a person

wearing a mask, a message is overlaid on the web camera feed informing the user of the prediction with confidence. A similar message is overlaid if the prediction returned indicates an image has a higher probability of containing a face not wearing a mask.

Recommendations for future enhancements:

There are several identified issues and enhancements that could be made to improve the application utility. While the model could have used binary classification, the initial objective was to perform a multiclass classification of images containing faces not wearing facemasks, wearing facemasks improperly, and wearing facemasks properly. Due to limitations in the availability of quality datasets, the 3-class prediction was descoped. However, the application could easily be adapted for a model performing 3-class prediction with minor adjustments. The face-detection components of the application are also areas of improvement. Differences in lighting have a clear impact on the ability to correctly detect faces in a camera feed. While algorithms from OpenCV were a somewhat reliable way of performing face detection, there are also other alternatives available that may improve accuracy such as using AWS's Recognition service through an API call.

Adding additional features to the Flask application was also considered. In a practical mask-detection application, classification results would likely be stored as a timestamped image within a permanent or semi-permanent storage location. Implementation of this feature could simply involve generating an API call to a service such as Amazon S3 to store time-stamped images. Depending on budget and requirements every image or only a portion of the images could be sent to S3. The flask web application could also be used for additional functionality. For example, letting the users directly snapshot or download an image from the feed could be enabled through custom buttons or other controls within the browser.

The application was initially developed on macOS and later tested on a machine running Windows. Future work would benefit from containerizing the application using Docker or a similar containerization service for simplified deployment. However, details on specific libraries used for model development have been provided in the Github README.txt instructions to assist developers in ensuring compatibility between environments.

References

Makwana, D. (2019, August). Face Mask Classification, Version 1. October 27, 2020 from https://www.kaggle.com/dhruvmak/face-mask-detection

Peeples, L. (2020). Face masks: What the data say. Nature, 586(7828), 186-189. doi:10.1038/d41586-020-02801-8