Face-Mask Compliance Recognition and Tracking

Guangwei Zhou, Yixuan Huang, Prasad Vagdargi, Luis Madera

Goals

Our goal for this project was to detect and track people wearing masks correctly, incorrectly, or not wearing a mask at all as a way to track the spread of an infectious disease. A person not properly wearing a mask might be a potential vector for spread. Our 3 objectives within the project were therefore:

1. To detect people and faces in a video of a crowd walking around.
2. Crop and classify detected faces into unmasked, correctly masked and incorrectly masked.
3. Use this information to track an unmasked or improperly masked person in video, over time to identify possible spread.

We achieved all of the above goals, although not in order. We split the entire project into these 3 tasks and aimed to complete Step 2 first, since it was the most crucial step. This was followed by Step 1 and then to integrate everything together, Step 3.

Methods

We approached each problem separately first, to test and debug our methods.

For Step 1: We first tested classical computer vision methods within OpenCV such as Haar Cascade for face detection [1]. These did not quite work well and needed extra parameters to be tuned for face detection.

Next, we switched to using a face recognition library [2] created by Adam Geitgey, which uses deep learning algorithms to detect all the faces that appear in an image, and then create rectangular sub-images each containing a face detected. If trained with labeled faces, this library is also able to recognize who the person is. This library provides a better performance than the previous one, and hence we applied it to every frame of the input video, so that all the faces that appeared in the video will be detected and tracked. Although the tracking in this specific case does not consider optical flow and motion, the result achieved will be equivalent. It is also important to mention that, unlike the previously mentioned Haar Cascade detection method, the Adam Geitgey detection method is able to detect faces wearing masks well. Otherwise, we could not have proceeded to classification in the next step.

Once we obtain the result of the face detection algorithm, we can crop out all the face sub-images and proceed to the classification models in order to determine whether the faces are wearing masks correctly, wearing masks incorrectly, or not wearing masks at all.

For Step 2: We tested different methods using deep learning for this task. Particularly, ResNets have been shown to be promising in Face Detection. We used a synthetic masked dataset for this task, developed using Flickr-Faces [4] dataset which consists of a mask overlaid by deforming it onto faces based on landmarks. The dataset is limited because of its artificial mask placement, along with the fact that only one type of mask (blue surgical mask) was used and overlaid on all the different faces. The total number of faces was ~137,000, and they were resized to 128x128 resolution. From the entire dataset, we chose a subset of approximately 40,000 images for initial development and testing. The dataset was split into 70% training - 15% validation - 15% testing for this project. The faces were available as correctly masked and incorrectly masked, so we augmented the dataset further with unmasked faces from the Flickr-Faces dataset, resized to the same size. Based on the difficulty of this task, ResNet-18 is chosen to minimize runtime and computational resource requirements. This network was initially developed to classify into 2 classes: masked vs

unmasked, and then later extended to classify into 3 classes: Masked, unmasked and incorrectly masked.

To ensure the network was generalizable, we augmented the data heavily using different augmentation methods such as random affine and perspective transforms, along with vertical and horizontal flips. The training hyperparameters are as shown. The classes were weighted by frequency to address class imbalance. The network converged in 10 epochs.

Parameter	Value
Optimizer	SGD
Learning Rate	$1x10^{-4}$
Momentum factor	0.9
Loss metric	Cross entropy loss
Batch size	1024
Number of epochs	10

Fig. 1: Training hyperparameters

We tested our method on synthetic dataset and achieved 98.21% accuracy on testing dataset. The model was able to predict most of the cases correctly, as shown in the confusion matrix.

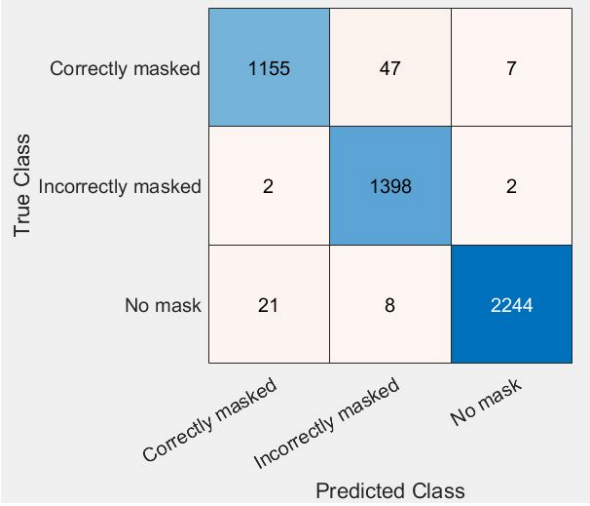

Fig. 2: Test results: confusion matrix

Our next task was to transfer the network and fine tune it on a limited real-world dataset. We used the dataset [5] consisting of 5000 faces. The results are as shown below. The network was able to fine tune and predict if the person had a mask on in the real-world dataset too.

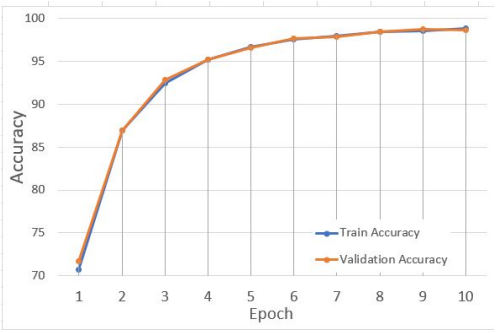

Fig. 3: Loss curve for trained mask detection network

To analyze our networks and verify that it was detecting the masks only, we used GradCAM to observe which regions of the face was the network focused on. As expected, the network focused on the mouth region the most, seen below.

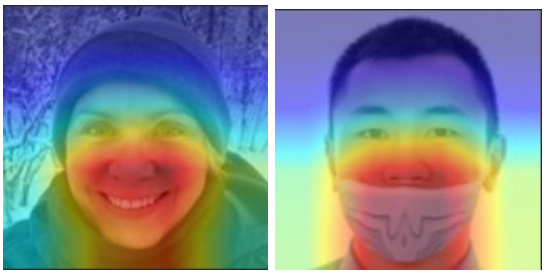

Fig 4: GradCAM heatmaps focusing on mouth region.

For Step 3: We looked at previous literature for person tracking in video, and with experience about the limitations of classical computer vision from Step 1, we decided to use pretrained deep neural networks to solve this problem. For this task, we used a combination of YOLO v5[6] network for prediction of objects in a video scene, along with DeepSort[7] for tracking a person consistently in the duration of a video.

YOLO (You Only Look Once) is a network that uses deep learning algorithms to achieve object detection. YOLO is faster than many other methods such as R-CNN because R-CNN needs to classify many small segments in an image, which is very time consuming. In contrast, YOLO uses only a neural network to complete the task. Specifically, the YOLO algorithm takes an image as input and splits the image into grids. Then, it uses the grids to predict whether a grid contains the target object. Lastly, it uses this information to predict a class for the object.

DeepSort uses 2 convolutional layers along with 6 residual layers and a dense layer to estimate the tracks and state of every object in the region of interest. The crucial part of the network is the use of deep appearance descriptors for each track, following which a cosine distance metric is minimized in the appearance space when training the network. The resulting network consistently identifies a person through a video, even if the person is occluded temporarily or turns around, which is important for our task. The network has low false detections and a fast runtime.

The 3 components (people tracker, face detector and classifier) are then applied on videos to observe the masking condition of individuals appearing in the video. To match with the training data of the classifier, the bounding boxes from the face detector are enlarged by 50% to ensure that the whole face is contained in the box. The masking condition of the face subimage is then determined by the trained classifier. The inter-frame correspondence of face subimages between frames is established by Deepsort. In this way, the people appearing in the video can be individually tracked throughout the video with their masks identified.

To test our approach in the most realistic scenario possible, we downloaded videos and news clips from YouTube from the beginning of the pandemic, where a large mix of both people with and without masks is available. We tested our networks on this and generated output videos which can be viewed here. We verified these results qualitatively since there were no quantitative labels available for such videos.

Eg. 1: The news reporter being identified and tracked as a person, and the face is detected and tagged as unmasked.

Eg. 2: A person being detected and tagged as wearing a mask incorrectly. Note that this was done within a crowd.

Eg. 3: A group of people being detected and marked as masked and unmasked simultaneously along with the bounding boxes.

Current Limitations

The network is currently limited in application due to inconsistent face detections from Step 1. Existing face detection methods in a video are extremely well tuned to detect using all available facial features, especially the mouth region. The lack of these features when a mask is worn means faces are only intermittently detected even if the person as a whole is detected - it was more intermittent, as observed in the video. The detection becomes even more challenging when the individuals are not directly facing the camera. However, when the face was detected, we observed a good accuracy in classifying if the face was masked or not.

One solution which we looked into for this problem was to train a custom face detector using YOLO for masked faces only. However, there is no large scale dataset available for this task which includes bounding boxes of faces.

Another limitation of the current method is the lack of real and diverse data for incorrectly

worn masks. Our current real dataset has 2 classes of masked and unmasked faces only, and given the large number of ways in which people can wear a mask incorrectly, it was a difficult task to achieve without labeled data.

Future extensions

Future work in this project can include fine tuning the face detection if a masked dataset with bounding boxes is available. Another improvement can be to include more real world data of people wearing masks incorrectly and fine tune the networks.

A more interesting direction would be to explore unsupervised learning for face detection instead of relying on labeled data, since a lot of data of people wearing masks would be available publicly using YouTube without labels. Therefore detecting and clustering faces would be a useful improvement.

Things learned

The project required a mix of different networks and techniques to complete the final task. We learned about implementing the different networks for different tasks, along with the advantages and disadvantages of each for a different task. We learned about the different loss functions for each task and the reasoning behind why it is a valid loss function for the task.

Another thing learned was the division of the large task into smaller tasks, followed by assembling the different parts together to create a usable, final outcome. Developing such a system requires coordination and compatibility between the data format and prediction output format for each network.

Advice for students and instructors

Under the special circumstances during the global pandemic, the remote classes can be challenging. We believe the instructors have done a great job in terms of delivering the class content. The lecture recordings and the asynchronous format of the course was extremely helpful. One advice we have for students is to get started on the class material early, before the synchronized lecture.

References:

[1] Haar Cascade for face detection and classification, OpenCV

[2] Adam Geitgey, Face Recognition, 2017, GitHub repository

[3] MaskedFace-Net -- A Dataset of Correctly/Incorrectly Masked Face Images in the Context of COVID-19

[4] Flickr-Faces Dataset

[5] Real-World Masked Face Dataset, RMFD

[6] You Only Look Once v5, Ultralytics

[7] DeepSort, Simple Online and Realtime Tracking with a Deep Association Metric