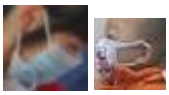
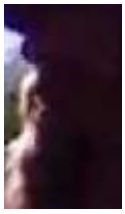
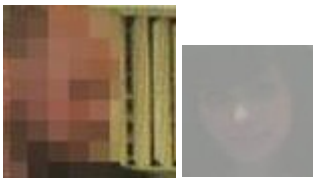
1. Exploratory data analysis:
    a. Visualization: we chose pictures that are in our opinion controversial – meaning they seemed to us to be hard to classify using the CNN model.
    The pictures we chose are:

    Low resolution pictures: 
    For reference: the first low resolution picture was classified as 0, although we can clearly see it is a side view of a person in a mask.

    Unknown picture (not clear if it's even a face) : 

    Blurred (Pixelated and Noise) : 

    Transparent masks: 

    b. Most of the people are from Asian heritage probably because wearing masks is a common habit in eastern countries.
    furthermore, we noticed a lot of the samples were classified by the rules explained to wearing masks, contrary to our opinions of what is considered wearing a mask.
    Another problems that we considered with the photos: some have more than one person, some are of bad quality, the size and number of pixels per unit changes drastically between some pictures and their shapes varies.

    also – too many celebrities. Barak Obama appeared at least three times, which may cause overfitting.

2. Experiments:
    a. Pictures Quality: As we mentioned, the data was of different quality and size for each picture. Furthermore, some pictures were blurred, noisy and filled with other distortions (like greyscale and more).
    we decided that most of them will be a good data augmentation for

the CNN model to learn from but the main problem was the size and un-normalized tensors.
We used transfomations on the images: first to turn them into tensors, then resizing them to

Labels: since the labels and and IDs of the pictures were in it's path, we needed to create a new type of dataset.
Our customized dataset inherits from torch.Dataset and we configured a way to save all the labels and images after transformations into the dataset, and then the dataloader.

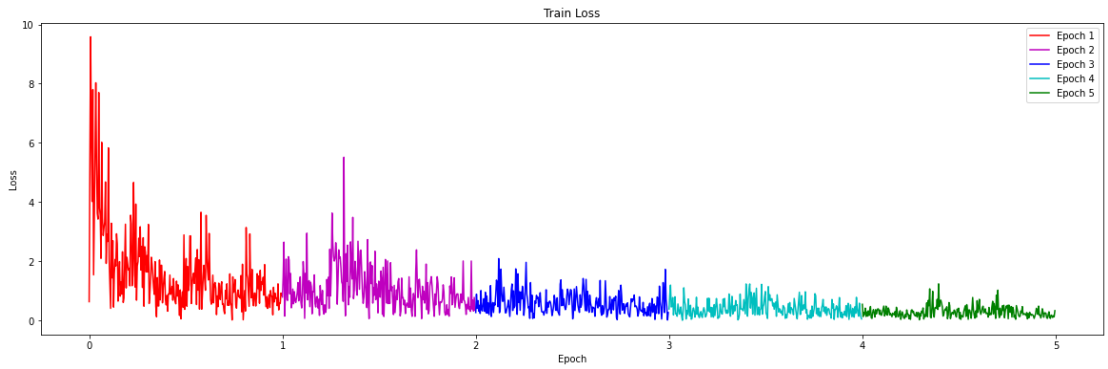
we used two sizes : 256 x 256 and 512 x 512. We then split the training data to 4000 pics for validation, and the rest trained the model.
we saved the datasets using the function torch.save() for each size: so overall 2 validation sets and 2 training sets, each of size 256 or 512.

b. At first we naively ran the model presented to us in the tutorial. We changed the structure to suit our data, including the transformation mentioned (normalizing and resizing).
We noticed that when we resized the photos to 512, the loss rises, and accuracy and F1 measures drop.
We measured the mean measures of our training set and found it to be much closer to 256.
After that, because the running time was pretty low, and we belived we have enough samples to avoid overfitting – we added a third layer, that downsampled the data again to 64 output channels.
We tried it and compared it.
we decided then to go with the model that resized the pictures to 256x256 using 3 hidden layers.

c. The loss function we decided to use is NLL, since we are facing a binary classification problem, and NLL is highly dependable to these kinds of problems. Also, because the rules that made people decide if a person is wearing a mask or not were such that they divided the face into regions, and that at least three were to be covered, it hinted that the CNN model will deem this problem a binary classification for every region in the face – which makes NLL much more fitting to use.
d. Optimization: We used the Adam optimizer like we used in the tutorial.
e. Regularization : We used Dropout as we saw in the tutorial, mainly because we worried adding the third layer might cause it to overfit, so we decided to keep it just as we learned in the tutorial.

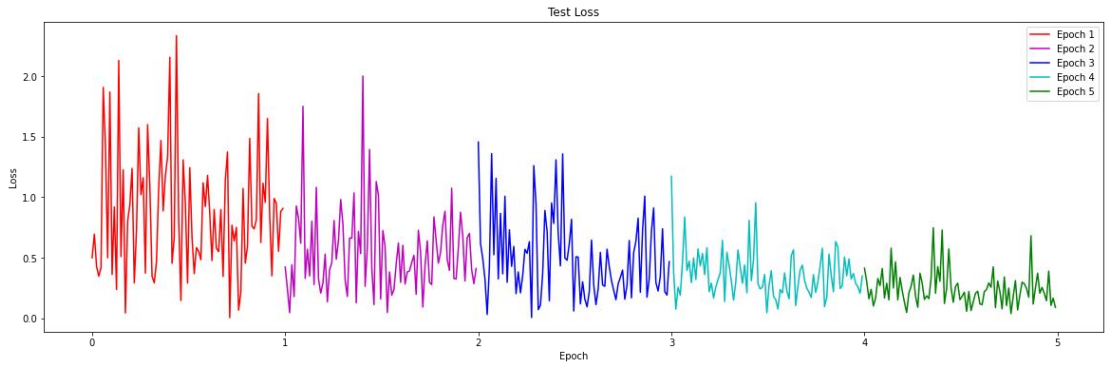
Graphs

1. Graph of decreasing Train Loss of batches for every epoch:



This is our final chosen module. We can notice that the variance of the loss every batch creates in every epoch, decreases drastically over time. We can see that not all batches necessarily decrease with every epoch, but eventually almost all of them do.

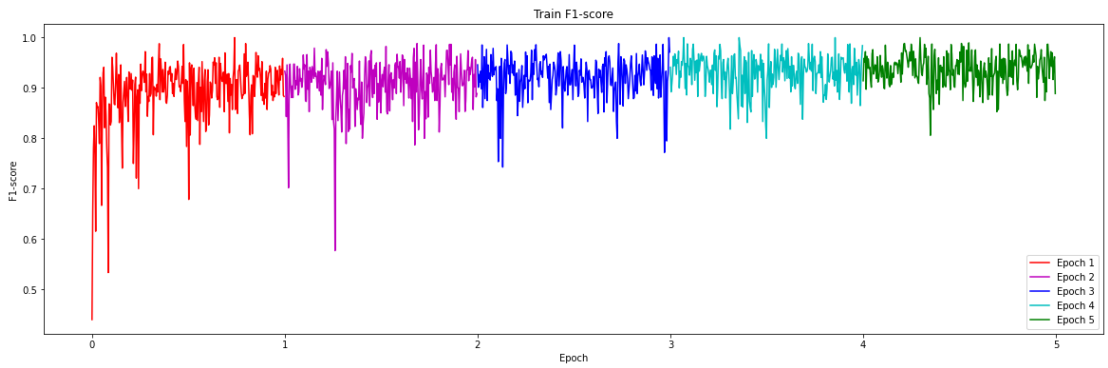
2. Graph of decreasing Test Loss of batches for every epoch:



We can see the improvement of the test loss over time is rapidly decreasing, similarly to the train loss, since we measured it after every epoch was trained.

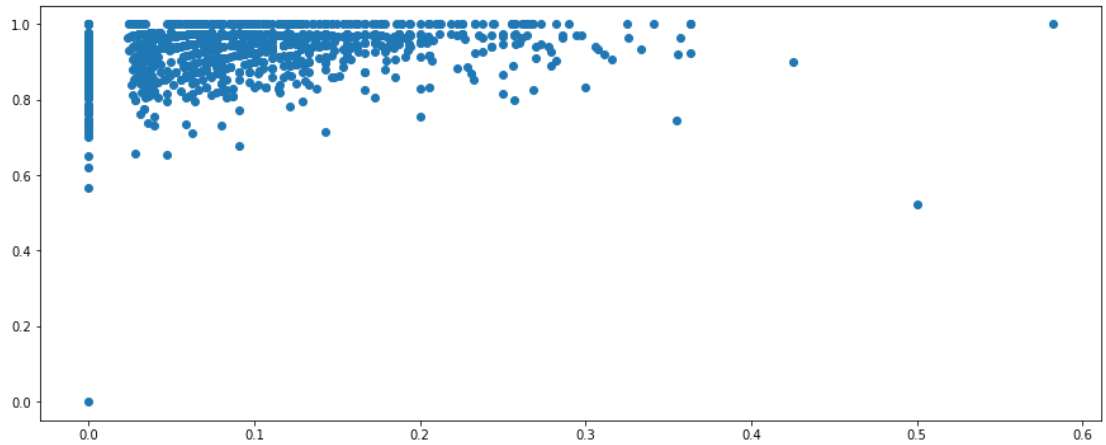
The final train loss can be seen in the green color, after epoch 5 (all batches are between 0-0.6 approx.)

3. F1 Train score:



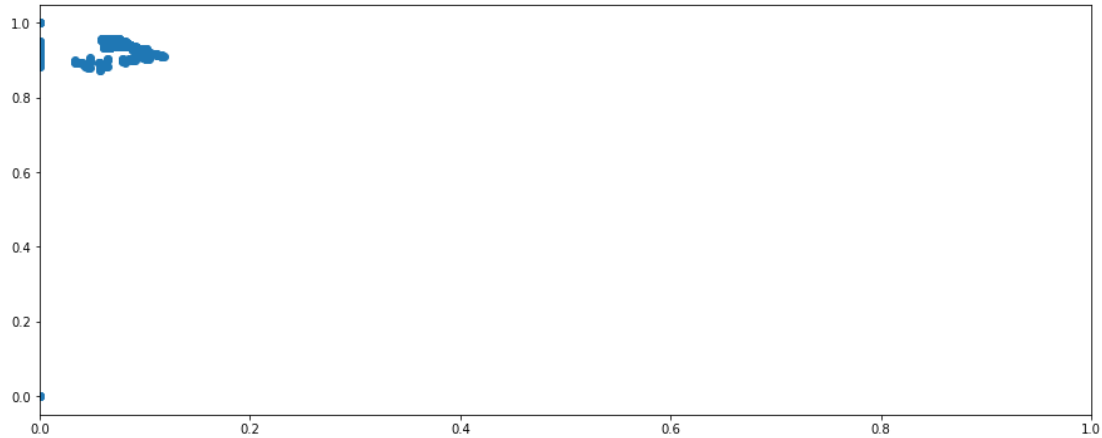
We can see the convergence of the score for every training epoch.

4. Train ROC-AUC Graph (TPR to FPR):



We can see that for every epoch the ROC graph reaches the “elbow” and pretty much converges to the upper left corner.

5.Test ROC-AUC (TPR to FPR):



As we can see, the test set showed much better results in the evaluation proccess.

Conclusions:

a. Our F1 score for our final model was: 0.9390
but this was for the specific splitting we did for out train-validation sets, which was seeded, so it may differ.

b. We observe that a pretty simply convolutional network can reach high end that would be applicable to many fields in visual (and even non-visual) problems.

c. Resizing the pictures to add more mass was proven to lower the success drastically, contrary to our initial belief.

d. We tried a couple of different models, and noticed that when the F1 measure is high in all of them, we have to look at the network more as a black box, because we are not sure which improvement to the F1 score was derived from which change to the model.