

# **Internship Report**

# **Masked Face-Recognition**

**Submitted by**

**Burouj Armgaan**  
**En. No. 2018BCSE082**

**Under the guidance of**

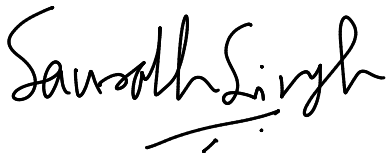
**Dr. Saurabh Singh**  
**Donguk University, South Korea**



**National Institute of Technology Srinagar, Kashmir**

# Certificate

This is to certify that the internship on “Masked Face-Recognition” carried out by Burouj Armgaan bearing enrollment number 2018BCSE082 under my guidance submitted to Computer Science Engineering Department, National Institute of Technology Srinagar in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science Engineering.

A handwritten signature in black ink, reading 'Saurabh Singh'. The signature is fluid and cursive, with a horizontal line underlining the name.

Dr. Saurabh Singh  
Assistant Professor  
Department of Industrial & Systems Engineering  
Donguk University

# Abstract

Due to the current pandemic, everyone is obliged to wear masks in public/workplaces. Although modern-day face recognition has a par-human (if not better) accuracy owing to the huge gains in computer vision (CV) in the previous 5 years, CV models struggle to recognize masked faces. Masked face-recognition has shot up in importance as removal of face masks for recognition of the full face or use of physical biometric methods like fingerprint scanning risk the spread of Covid-19.

In this report, a triplet loss based Siamese network is developed to generate embeddings that can be used to recognize masked faces. This approach combines transfer learning using ResNet50 and occlusion removal to develop the embedding model. A basic Neural Network classifies the generated embeddings. The proposed technique does not require a masked-face dataset for training. Any face-recognition dataset can be utilized.

# Masked Face Recognition

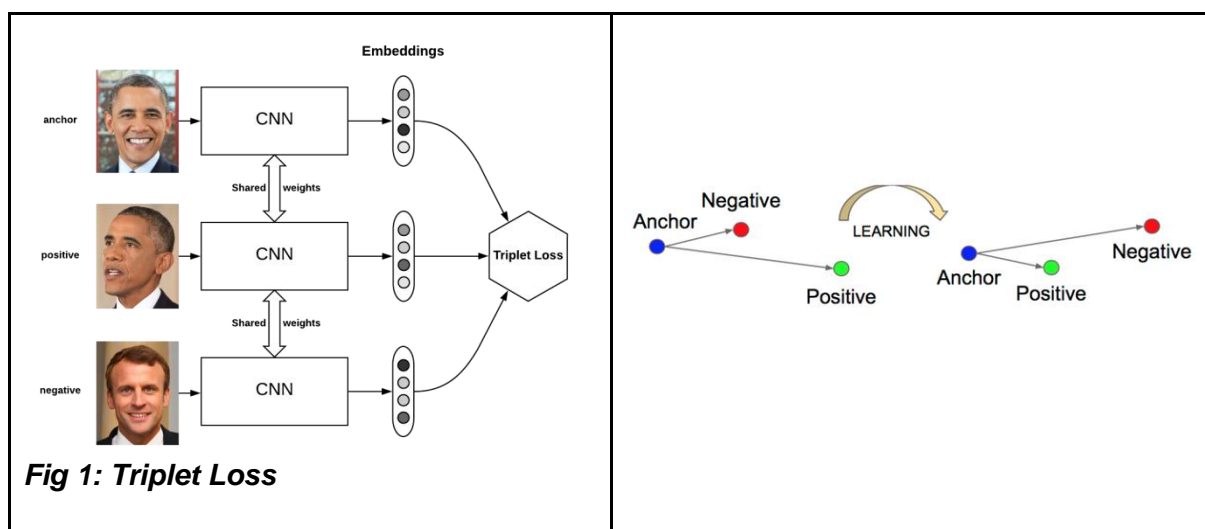
## 1. Introduction

Face recognition is a computer vision task in which the model predicts the identity of the person in the image. The naive way to accomplish this is to train a Convolution Neural Network (CNN) over a dataset requiring lots of images of the same person. This however is not efficient. Usually, only a few images of an individual are available. Add to that, if a new person's data needs to be added to the model's knowledge so that he/she can also be recognized, then the whole model needs to be trained again. This is clearly inefficient.

A better way is to use a technique called one-shot learning. In one-shot learning, a model is trained to generate embeddings of people in a way similar to embeddings of words in Natural Language Processing i.e. similar words get embeddings that are close together in the embedding space and dissimilar words get embeddings that are farther away from each other. As usual, to train a model, an objective function is required. One of the best objective functions used in this approach is the Triplet loss [4]. The triplet loss takes in three inputs, an anchor image embedding, a positive (similar) image embedding and a negative (dissimilar) image embedding (Fig. 1). Mathematically, the triplet loss is calculated as follows:

$$\mathcal{L}(A, P, N) = \max\left(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0\right)$$

where alpha is the desired margin between a  $\|f(a) - f(p)\|^2$  and  $\|f(a) - f(n)\|^2$ . Greater the value of alpha, greater the gap.



A model trained with triplet loss doesn't need to be retrained from scratch if a new individual's data is to be added. Even one image of that individual is sufficient to generate that individual's embeddings. Once the embeddings are generated by the

model, any classification model can be used for making predictions, like a Neural Network or an SVM classifier.

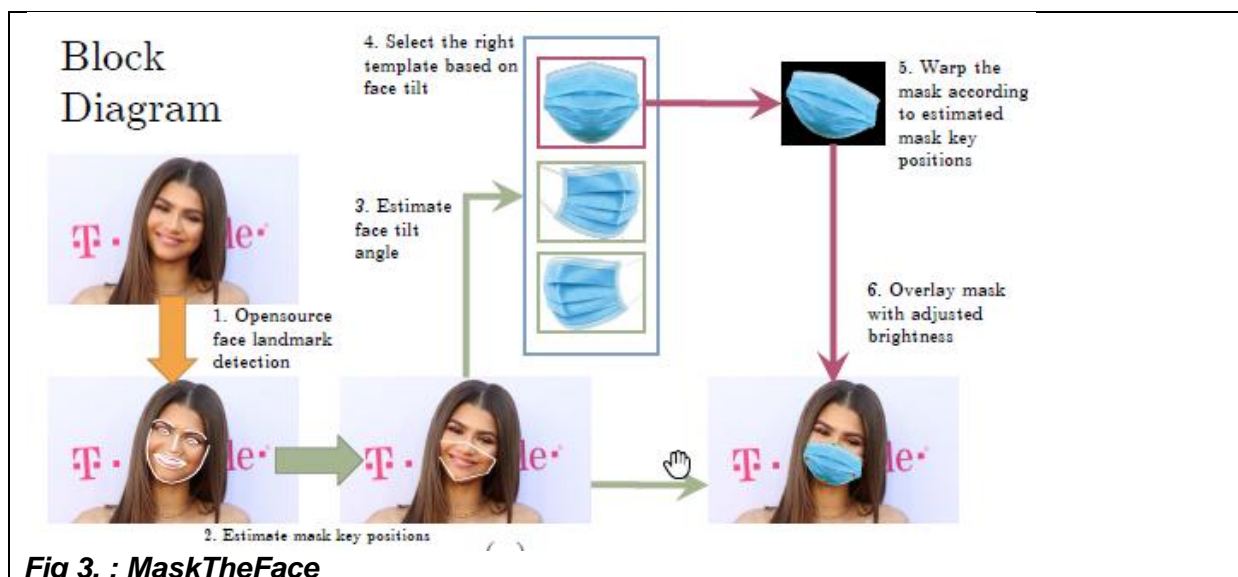
### 3. Related Work

#### 3.1 Simulated Masked Face Datasets

Zhongyuan Wang et.al. [1], Aqeel Anwar and Arijit Raychowdhury [2] use simulated masked face datasets to tackle the problem (Fig. 2). The idea is to simply train a brand new model on masked faces. The lack of such a dataset necessitates a detour in the form of simulating artificial masks on faces. Zhongyuan Wang et.al. [1] uses landmark detection to identify the correct spot for appending the mask on the face. In this manner, a simulated masked dataset is created. Building on top of this paper, Anwar and Arijit Raychowdhury [2] critic the flaw in the previous approach that the dataset generated lacks ethnic diversity and variations in mask types and patterns. In response, they developed a computer-vision script named MaskTheFace (Fig. 3) which can be used to convert any face-dataset into a masked face-dataset by appending masks onto them. MaskTheFace provides various mask types and patterns along with ease of use to make usage of simulated masked faces more robust. This preprocessing step is, however, time-consuming.

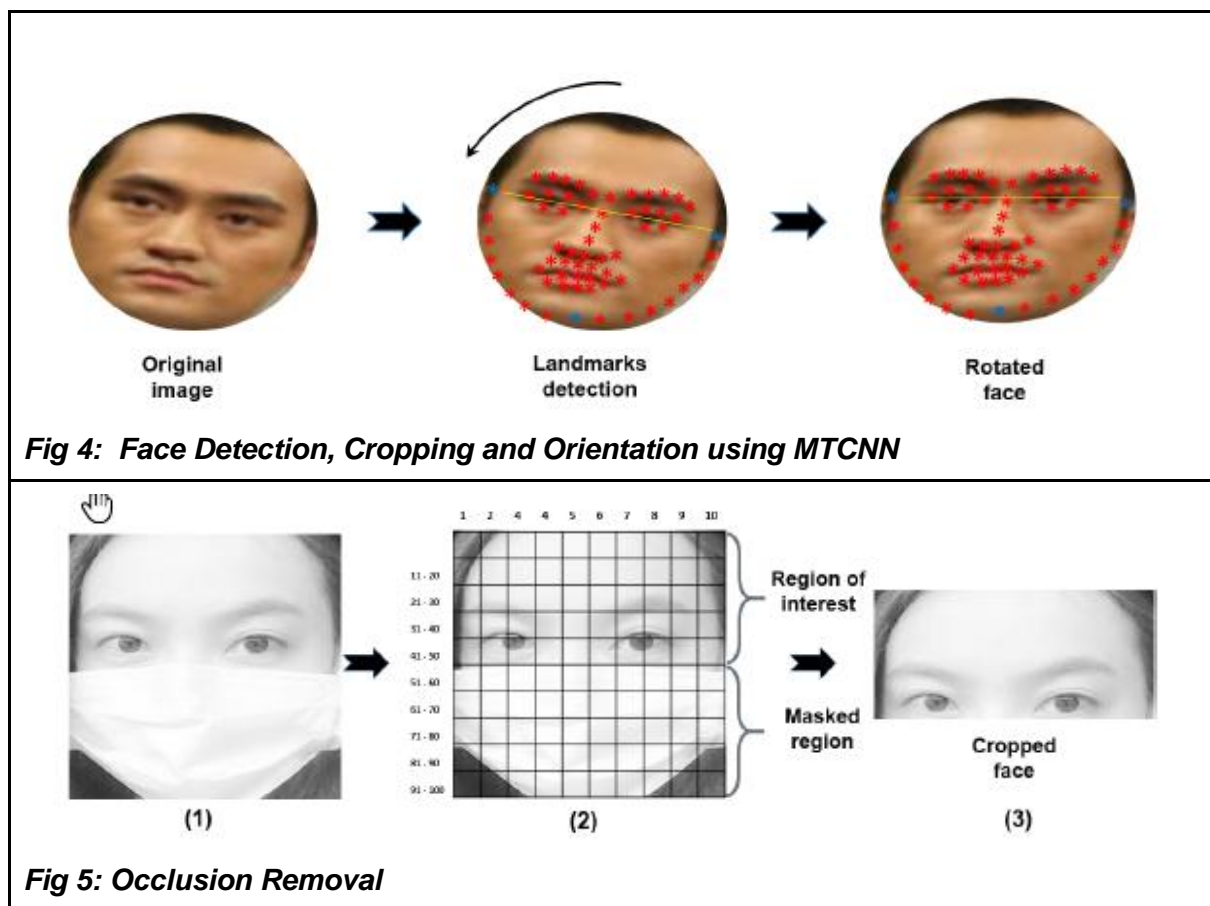


**Fig. 2: Simulated Masked Faces**



### 3.2 Occlusion Removal

A completely different approach is employed by Walid Hariri [3]. Instead of simulating masked faces, he discards the masked portion of the face completely and utilizes only the upper, non-masked half of the face for training. This approach goes as follows: detect, crop and orient the face using facial landmarks using MTCNN (Fig. 4); divide the face into 100 blocks and number them horizontally 1 through 100 from top left; discard the last 50 blocks (which essentially means to take the upper half of the face) (Fig. 5); use a pre-trained model for generating embeddings; supply embeddings as input to an RBF layer (bag-of-words paradigm [5]) for output. His work claims that any deep CNN can be used for generating embeddings and requires no training.



## 4. Proposed work

The approach used in this project is inspired by Walid Hariri [3]. It makes sense to discard the masked portion of the face. When we humans encounter a masked face, we too disregard the face entirely and focus on the upper half of the face for recognition. The very concept of embeddings is to map similar images as close

embeddings and dissimilar images as embeddings that are farther apart. The masked half of the face biases the embeddings to be closer leading to poor performance.

This approach diverges from Walid's [3] in that it does not employ the bag-of-features paradigm and doesn't use off-the-shelf deep CNN. Experimentation with off-the-shelf FaceNet gave very poor results. So, instead, transfer learning is employed, retraining ResNet50 as a triplet loss based Siamese model. The dataset used is RMFRD, which was employed as a benchmark by [1,2,3]. The preprocessing step involves detecting, cropping and orienting the faces using MTCNN. Each face is divided into 100 parts and the first 50 are chosen, in correspondence to Walid's work [3]. The clipped faces are resized to shape (160,160,3). These form the base dataset. Three datasets are generated on top of it: anchor, positive and negative, for training the triplet loss employed in the Siamese network. Each dataset is huge individually (86000+ images). Only 40,000 images were used from each dataset. That means 120,000 images need to be brought into memory. This is impossible to do as one batch or without replacement of previously read batches. In order to overcome this, TensorFlow's Data API is used. After training the Siamese model, another model is created to recognize faces. The model is formed by simply concatenating dense layers at the end of the embedding generating model (while freezing the layers of the embedding model).

The bottlenecks in the training process are the preprocessing step and saving the triplets to disk. Since both involve disk access every time an image is read and CPU computation every time an image is cropped and preprocessed, preprocessing MFRD with a total of 86340 images took more than 8 hours. Training is relatively fast with GPUs as the model is not being trained from scratch.

## 6. Pseudocode

### 6.1 Preprocessing

1. Iterate over the directories and save the names of individuals in a list (assuming that the folder names are the names of the individuals).
2. In each iteration, iterate over the images of the current directory.
  - a. Read the image;
  - b. Pass it through MTCNN
  - c. Crop the upper half of the MTCNN output (given that MTCNN detects a face).
  - d. Subtract mean and divide by standard deviation.
  - e. Write the image to disk.

## 6.2 Generating Triplets

1. Iterate over all the directories.
2. In each directory,
  - a. Read processed images in an array, say, anchor.
  - b. Parallelly create a shuffled copy of the anchor array and append them to another array named positive.
  - c. Save the IDs (iteration number) for both anchor and positive in separate lists.
3. Create the negative array and its IDs by shuffling the whole Positive array and its IDs. (Since the image corpus is so large, a random shuffle will also act as a strong negative)
4. Turn the Anchor, Positive and Negative arrays into TensorFlow datasets.
5. Club the 3 datasets into a single TensorFlow dataset.
6. Save the datasets to disk.
7. Save the anchor dataset and its labels to disk as a TensorFlow dataset as well. This will be used to train the Recognizer model.

## 6.3 Training the Siamese Model

1. Use Tensorflow's subclassing API to build the base model that generates the embeddings. Let's call it the embedding model.
2. Use the subclassing API to create a custom Distance layer that measures the distance between the embeddings generated by the model.
3. Use the subclassing API to create a custom model with a custom loss as the triplet loss which utilizes the embedding model and the Distance layer.
4. Train the TensorFlow dataset generated in the previous step. Note that this model does not have any labels to train against. We are just reducing the triplet loss.

## 6.4 Training the face-recognition model

1. Use Tensorflow's subclassing API to append dense layers to the embedding model. Remember to freeze the layers of the embedding model.
2. Pass the anchor dataset from the Triplet generation part to the model.



## 7. Results

```
cosine_similarity = keras.metrics.CosineSimilarity()

positive_similarity = cosine_similarity(anchor_embedding, positive_embedding)
print("Positive similarity:", positive_similarity.numpy())

negative_similarity = cosine_similarity(anchor_embedding, negative_embedding)
print("Negative similarity", negative_similarity.numpy())
```

```
Positive similarity: 0.922919
Negative similarity 0.8898251
```

Example cosine similarity score. Notice that the similarity score for embeddings of the same person is higher than that of different persons. The difference may seem small but even in triplet loss, the margin is set between 0 and 1.

```
Epoch 96/100
625/625 [=====] - 38s 61ms/step - loss: 0.8389
- accuracy: 0.7476
Epoch 97/100
625/625 [=====] - 38s 61ms/step - loss: 0.8212
- accuracy: 0.7542
Epoch 98/100
625/625 [=====] - 38s 61ms/step - loss: 0.8327
- accuracy: 0.7570
Epoch 99/100
625/625 [=====] - 38s 61ms/step - loss: 0.8133
- accuracy: 0.7602
Epoch 100/100
625/625 [=====] - 38s 61ms/step - loss: 0.8110
- accuracy: 0.7596
```

Model accuracy after 100 epochs.

## 8. Conclusion

The presented approach utilizes transfer learning and occlusion removal to overcome the problem of masked face recognition. A triplet loss based siamese network is trained using ResNet50 architecture and the RMFRD dataset. An accuracy of 75% is achieved.

## 9. Code

Visit my Kaggle Notebook:

<https://www.kaggle.com/armgaan/masked-face-recognition>

Datasets used and saved model weights: <https://www.kaggle.com/armgaan/triplets-for-masked-facerecognition>

Visit my Github repository:

<https://github.com/Cossak/Masked-Face-Recognition>

## 10. References

- [1] Wang, Z., Wang, G., Huang, B., Xiong, Z., Hong, Q., Wu, H., ... Pei, Y. (2020). Masked Face Recognition Dataset and Application. ArXiv Preprint ArXiv:2003.09093.
- [2] Anwar, A., & Raychowdhury, A. (2020). Masked Face Recognition for Secure Authentication. ArXiv Preprint ArXiv:2008.11104.
- [3] Hariri, W. (2020). Efficient Masked Face Recognition Method during the COVID-19 Pandemic. ArXiv Preprint ArXiv:2105.03026.
- [4] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 815–823).
- [5] Passalis, N., & Tefas, A. (2017). Learning Bag-of-Features Pooling for Deep Convolutional Neural Networks. In 2017 IEEE International Conference on Computer Vision (ICCV) (pp. 5766–5774).