Face Recognition Uusing Deep Learning

-- CPE 800A Final Report

Raveena Mehta
Dept. of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, USA
rmehta19@stevens.edu

Kishan Teli
Dept. of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, USA
kteli@stevens.edu

Prof. Kevin Lu
Dept. of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, USA
klu2@stevens.edu

***Abstract*—In today's world, face recognition has become an integral part of our lives. We use face recognition feature for multi-purposes in our latest gadgets such as FaceID, etc. In this project, we will implement face recognition technique using deep learning concepts. Our aim is to recognize people from the given dataset such as photos. The name of that person will be displayed on the photo being displayed. We will accomplish this through four important phases: face detection, posing and projecting faces, detecting faces with or without masks and lastly, identifying the person's face through our database and displaying it on the image file given. We implement these phases by using two pre-trained models that are RetinaNet and Xception models for detection and classification respectively, due to their high performance. We will manually train these models through many datasets, to achieve our aim. We are able to implement our system, to detect faces of people with or without masks and also recognize these people from our small database. Our model's precision and recall values are 0.99 and have high accuracy even when dealing with faces having masks. Our model was designed especially for current times of COVID-19 pandemic, where in we can still do facial recognition with consistent accuracy. We can use this feature in many gadgets and applications for authentication and medical purposes.**

Keywords— face detection, posing, classification

I. Introduction

Our project addresses the problem of the increasing digital security threats and issues faced. The idea of facial recognition is better than the other existing solutions for authentication, because it doesn't require user's active co-operation. All the other features such as fingerprints, passcodes, etc. used for authentication purposes, need an active co-operation from users. Thus, the problem of ease needs to be solved. In this scenario, face recognition plays an important role. Another problem ~~which~~ that our project addresses is something that has risen in recent times, is that of COVID-19 pandemic. Due to this, people's need to wear masks on their faces have increased, so as to reduce its spread and be safe themselves as well. Hence, our project recognizes faces even when they have worn masks.

There were many challenges in implementing this project. We tried to achieve our goal by using histogram of oriented gradients (HOG) for face detection and for training our system, we used deep convolutional neural network (CNN). We used linear support vector machine (SVM) model classifier as well for implementing our project. ~~But,~~But we failed to achieve our main goal, i.e., to detect faces while wearing a mask. We could not do so by facial landmark estimation algorithm, as we were unable to map the remaining 34 points of the other half of the face. Within the time constraints, we also had to create our own database in order to recognize the faces. Hence, we researched more on alternatives through which our goal could be achieved. After reading papers and accessing the internet, we finally landed on a great alternative ~~which~~ that gave us results. In order to detect faces with masks, we scrapped our previous solution of predicting faces and instead decided to train our system by RetinaNet pre-trained model. This model along with Xception model used for classification, trained on ImageNet database were implemented to reach our goal of the project. The other challenge ~~which~~ that we faced while implementing the new approach, was that we needed a lot of collection of data, in this case, images of masked people, in order to train our system.

There are many existing works related to facial recognition. Facial recognition could be achieved by using multi-pose representations [2]. This was possible by using 3-D rendering to generate multi-face poses from any input image given. In the ~~above mentioned~~above-mentioned article, they discussed about the implementation of CNN models to achieve the same and how by using this method, they increased their performance more than that of the supervised learning such as gallery fine-tuning and metric learning. There is mention of recognizing faces when live video streaming is being done [5]. FaNC is a novel face normalized method ~~which~~ that is used to reduce pose-induced image variance [6]. There was

Formatted: Font: Not Italic
Formatted: Font: Not Italic
Formatted: Font: Not Italic
Formatted: Font: Not Italic
Formatted: Font: Not Italic
Formatted: Font: Not Italic
Field Code Changed
Formatted: Font: Not Italic
Formatted: Font: Not Italic
Formatted: Font: Not Italic
Field Code Changed
Formatted: Font: Not Italic
Field Code Changed
Formatted: Font: Not Italic

an interesting article that mentioned about the kinship synthesis framework used to predict a smile of a child from the faces of the smile of his/her parents [8]. We studied this article in more detail in order to learn more about predicting faces. The paper discussed about implementing the dynamics and appearance model together, to generate synthesized smile video of the estimated child from the real smile videos of their parents. Their studies show that these synthesized videos that depicted the child's smile had more accuracy than the real videos of their children's smile. They concluded that by synthesizing a dataset, we could achieve high accuracy in facial recognition and prediction. Target face association (TFA) is used to retrieve a set of representative face images from a video that is likely to have a same identity as the target face [10]. We understood the reasons due to which RetinaNet matches the speed of one-stage detectors and also surpasses the accuracy of 2-stage detectors and got introduced to the concept of focal loss for reducing class imbalance [12]. The comparison between Xception and Inception were discussed in an article, on the basis of their accuracy and performance [13].

Our paper will discuss about the phases of our project: Face detection, Posing and projecting faces, Identification of a face using training by RetinaNet model and Xception model instead of using facial landmarks and lastly, Identifying the face from our dataset and displaying the name of the person. We will also detect whether a face is with mask or without mask and display the same on screen. Accuracy will be determined. Recognition should be accurate even if only half of the face is given as the data to our system.

II. Problem Statement

Face recognition involves many complex tasks that needs to be divided into stages. Thus, there are four main stages by which we can achieve face recognition using deep learning. All the four stages are being implemented by two models. Face Mask Detection is done on the basis of RetinaNet model and a separate model used for classification called as Xception.

In order to achieve our objective, we need to first train these models. To do so, we need many datasets, which consists of images in which people wear masks. As our aim is to make a model that detects faces even when a mask is worn, we need to train our model with such images. This way, we make our model more precise. Hence, first step is, to collect as many datasets as possible. We also need to look for datasets that contain images wherein, there are a large number of people, i.e., dense. Once, we have the right datasets, we will start training our models. We will also read images and put data into them through data generator. It also allows parallel loading of the input bytes through shared memory.

As RetinaNet and Xception model, both are pre-trained models, by training them for our specific output, we increase their precision and recall performance metric further. After training our models sufficiently, we then check for output of our project. We will check for our desired output but also, for analysis of our model's performances. We will measure their performance in terms of accuracy, precision, recall, loss and F1-score. F1-score is the harmonic mean of precision and recall. It is chosen as the criterion for evaluation for the classification model. Being bound between 0 and 1, F1-score reaches its best value at 1 and worst at 0.

III. Solution

We had initially implemented face detection using face landmark estimation algorithm. And were successful in it. We had got facial point scores and were able to detect the face of a person from the given image file. These facial point scores are unique to everyone, except identical twins. Each face has 68 facial points. If half of the face was covered by mask or was simply cut off from the image input file, we would have only 34 points of the face. We planned to map the remaining 34 points by using CNN model and SVM classifier. Thus, detecting the face even when mask is worn as well as identify the person from our database and display his/her name on the file processed. But, due to many errors generated in our code and the given time constraint, we opted for another alternative.

Thus, in our project, we have tried to come up with a solution this problem by using RetinaNet model as well as Xception model for classification purpose. Thus, our final mask detection model will run as follows:

1) Apply the RetinaNet Face model for face detection to generate detected face crops from the input image.

2) Xception model for classification into mask and no-mask categories for the detected face, is then applied upon the detections generated by RetinaNet model.

3) The final output of these two would be the faces detected by RetinaNet along with the predicted category for each face, that is whether the subject is wearing a mask or not.

A. The Xception model:

It is called an "extreme version of its predecessor Inception V3" for the reason that an efficient use of model parameters by using depth-wise separable convolutions gives better performance on the ImageNet database. While keeping the number of parameters same in the Inception V3 and Xception model, we ensure that the capacity of the model remains the same but yielding better performance on an almost unparalleled scale, than was achieved using Inception V3 model. Some properties of the Xception model chosen in our case are discussed below:

1) Shape invariance: similar results irrespective of the dimensions of input images (by using Global Avg pooling instead of flattening).

2) Removal of final fully connected layer: Prior layers to the final fully connected layer were trained using the weights of Xception trained on the ImageNet database.

```
base_model = tf.keras.applications.Xception(include_top=False,
                                            input_shape=(None, None, 3),
                                            weights='imagenet')
base_model.trainable = False
layer = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
layer = tf.keras.layers.Dense(1024, activation='relu')(layer)
layer = tf.keras.layers.Dropout(0.5)(layer)
output = tf.keras.layers.Dense(num_classes, activation='softmax')(layer)
model = tf.keras.models.Model(base_model.inputs,output)
model.summary()
```

Fig. 1. Model implementation.

Implementation details for classification architecture:

- ***Batch Size***: 32
- ***Epochs***: 2
- ***Learning rate***: 1e-4 (with decay of 1e-4 / epoch)
- ***Gradient Descent Optimizer***: Adam
- ***Loss function***: Sparse categorical cross entropy
- ***Criterion for evaluation*** (metric): F1-score

```
#We can use Learning rate scheduler here like Cyclical Learning Rate(available in Tf Addons)
model.compile(optimizer=tf.keras.optimizers.Adam(1e-4, decay=1e-4 / epoch),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

callbacks = [
    tf.keras.callbacks.ModelCheckpoint('mask_classification_model.h5',
                                       save_best_only=True,
                                       save_weights_only=True,
                                       monitor='loss')]

model.fit(train_dataset,
          batch_size=batch_size,
          epochs=epoch,steps_per_epoch=step_per_epoch,
          callbacks=callbacks)
```

Fig. 2. Mask classification model.

For the purpose of classification, we use transfer learning with an Xception model trained on the ImageNet dataset with a modified final fully connected layer. A classification problem is, using available training data with defined features and class labels, build a function that can, with high levels of certainty categorize a new unseen set of data into one of the classes. Having only a limited amount of data available for training the classifier, we used transfer learning for the purpose of our task of classifying an input image into the categories of whether the subject is wearing a mask or not. The transfer learning technique can be thought of reusing a pre-trained model, which is trained on another set of input images, which in our case would be the use of Xception model trained on the ImageNet database. Transfer learning was employed to use the model weights of the Xception model.

***B. The RetinaNet model*:**

In order to understand the significance of RetinaNet and focal loss, we need to understand them. RetinaNet is a composite network composed of:

1) Backbone Network (i.e., Bottom-up pathway + Top down a pathway with lateral connections, e.g., ResNet + FPN)
2) Subnetwork for object Classification
3) Subnetwork for object Regression

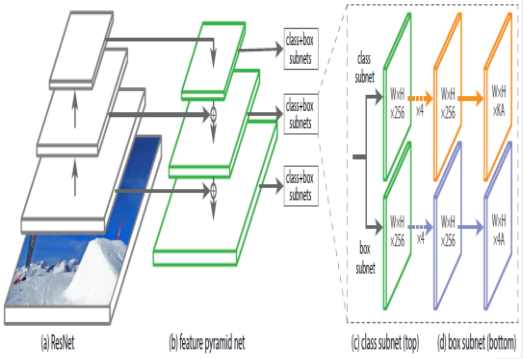

Fig. 3. Architecture of RetinaNet.

We will discuss each component of the architecture in detail separately:

a) The backbone Network:

1) Bottom-up pathway: Bottom-up pathway (e.g., ResNet) is used for feature extraction. So, it calculates the feature maps at different scales, irrespective of the input image size.

2) Top-down pathway with lateral connections: The top-down pathway upsamples the spatially coarser feature maps from higher pyramid levels, and the lateral connections merge the top-down layers and the bottom-up layers with the same spatial size. Higher-level feature maps tend to have small resolution though semantically stronger. Therefore, more suitable for detecting larger objects; on the contrary, grid cells from lower level feature maps have high resolution and hence are better at detecting smaller objects (see Fig. 3). So, with a combination of the top-down pathway and its lateral connections with bottom up the pathway, which do not require much extra computation, every level of the resulting feature maps can be both semantically and spatially strong. Hence this architecture is scale-invariant and can provide better performance both in terms of speed and accuracy.

b) Subnetwork for object Classification:

A fully convolutional network (FCN) is attached to each FPN level for object classification. As it's shown in the diagram above, this subnetwork incorporates 3*3 convolutional layers with 256 filters followed by another 3*3 convolutional layer with K*A filters. Hence output feature map would be of size W*H*KA, where W & H are proportional to the width and height of the input feature map, and K & A are the numbers of object class and anchor boxes respectively. At last Sigmoid layer (not softmax) is used for object classification. And the reason for the last convolution layer to have KA filters is because, if there're "A" number of anchor box proposals for each position in feature map obtained from the last convolution layer then each anchor box has the possibility to be classified in K number of classes. Thus, the output feature map would be of size KA channels or filters.

c) Subnetwork for object Regression:

The regression subnetwork is attached to each feature map of the FPN in parallel to the classification subnetwork. The design of the regression subnetwork is identical to that of the classification subnet, except that the last convolutional layer is of size 3*3 with 4 filters resulting in an output feature map with the size of W*H*4A. Reason for last convolution layer to have 4 filters is because, in order to localize the class objects, regression subnetwork produces 4 numbers for each anchor box that predict the relative offset (in terms of center coordinates, width, and height) between the anchor box and the ground truth box. Therefore, the output feature map of the regression subnet has 4A filters or channels.

C. Focal Loss:

Focal Loss (FL) is an improved version of Cross-Entropy Loss (CE) that tries to handle the class imbalance problem by assigning more weights to hard or easily misclassified examples (i.e., background with noisy texture or partial object or the object of our interest) and to down-weight easy examples (i.e., Background objects). So Focal Loss reduces the loss contribution from easy examples and increases the importance of correcting misclassified examples. Focal loss is used to focus training on hard negatives. To achieve these, researchers have proposed

$(1-p_t)^{\gamma}$ *to the cross entropy loss, with a tunable focusing parameter* $\gamma \geq 0$.

(1)

RetinaNet object detection method uses an α-balanced RetinaNet object detection method uses an α-balanced variant of the focal loss, where α=0.25, γ=2 works the best.

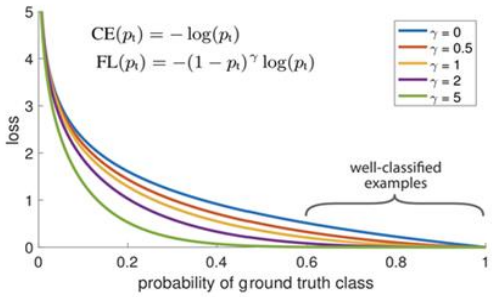

Fig. 4. Plotting loss vs. probability of ground truth class.

So focal loss can be defined as

$$FL(p_t) = -(1-p_t)^{\gamma}\log(p_t). \quad (2)$$

The focal loss is visualized for several values of $\gamma \in [0,5]$ as shown in Fig. 4. We shall note the following properties of the focal loss:

1) When an example is misclassified and p_t is small, the modulating factor is near 1 and does not affect the loss.

2) As $p_t \rightarrow 1$, the factor goes to 0 and the loss for well-classified examples is down weighed.

3) The focusing parameter γ smoothly adjusts the rate at which easy examples are down-weighted. As γ is increased, the effect of modulating factor increases likewise. (After a lot of experiments and trials, researchers have found $\gamma = 2$ to work best). When $\gamma = 0$, FL is equivalent to CE as shown in the blue curve in Fig. 4. Intuitively, the modulating factor reduces the loss contribution from easy examples and extends the range in which an example receives the low loss.

The reasons why RetinaNet is implemented for this program are as follows.

```
class_map = {
    0: 'With mask',
    1: 'Without mask'
    }

color_map_image = {
    0: [0,1,0],
    1:[1,0,0]
    }
```

Fig. 5. Class map and color map.

Other than implementing the models, we needed a proper dataset, in order to train our Mask Detection Model. The steps in which we trained our model were:

D. Data Collection for Training examples:

We used a data that was classified into two categories of images: images of people wearing masks and the ones without mask. The people wearing masks on their face in an incorrect manner would be classified into the "without mask" category. Data collection was done using a variety of sources from interfaces such as Kaggle API and Bing Search. As an additional source, the Real World Masked Face Detection (RMFD) dataset was used to facilitate training of our mask detection algorithm using the classification based approach of detecting masks followed by predicting whether the detected face contains a mask or not. The total number of images in the training set were divided into the two categories as follows:

1) With mask: **8072 images**

2) Without mask: **8086 images**

E. Data Collection for Inference:

The images collected for the purpose of testing our model had to come from a crowded setting to provide a realistic test scenario for my proposed model for mask detection. This task was accomplished using Google search for images and available YouTube videos for the purpose of scraping the web for realistic examples suited to inference in a real world surveillance setting. Test Dataset (Images):

1) Source (Google image search):
https://images.google.com/ Count: 117
2) Source (Associated Press images):
http://www.apimages.com/ Count: 13

F. Reading images:

We need to apply transformations on the input image before defining inputs for the model architecture. Transformations are done so as to ensure that the inputs to the network are of the right size and respective values sit in a similar range. Input image was resized to (128,128) for (height, width) and values in the image tensor were scaled to the range of (-1,1). Entire dataset was split into training and test set for cross validation. Split ratio was chose such that 90 percent of the dataset is part of the training set and 10 percent for test set.

G. Data generator:

Using a data generator, we batched individual samples into a unified set. Data generator spawns processes for loading data to input while training the model. This function also provides parallel loading for batches in the training data using shared memory and separate process.

```
def show_img(dataset):
  plt.figure(figsize=(15,15))
  for i in range(8):
    for val in dataset.take(1):
      img  = (val[0][i]+1)*127.5
      plt.subplot(4,2,i+1)
      plt.imshow(tf.cast(img,tf.uint8))
      plt.title(val[1][i].numpy())
      plt.subplots_adjust(hspace=1)
  plt.show()

train_dataset = data_generator(train_images,train_labels)

show_img(train_dataset)
```

Fig. 6. Data generator.

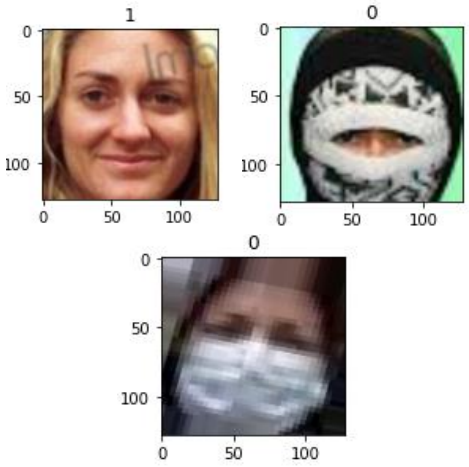

Fig. 7. Datasets for training.

We made our own little database for recognition purposes, so as to see whether the model matches the face from its known database and then displays its name. Thus, in this way, we applied our mask detection model by combining two models of RetinaNet and Xception to detect faces with masks and without masks. We trained ~~this models~~these models using different datasets for more accuracy and precision.

IV. Numerical Results and Analysis

We tried to test the classification model, i.e., the Xception model. We tried to check the ground truth and prediction values generated for each image.

```
plt.figure(figsize=(15,15))
for i in range(8):
  for val in train_dataset.take(1):
    plt.subplot(4,2,i+1)
    img  = (val[0][i]+1)*127.5
    plt.imshow(tf.cast(img,tf.uint8))
    y_pred = model.predict(np.expand_dims(val[0][i],axis=0))
    y_pred = np.argmax(y_pred,axis=1)
    plt.title('Ground Truth{}, Predicted{}'.format(val[1][i],y_pred))
    plt.subplots_adjust(wspace=1, hspace=1)
plt.show()
```

Fig. 8.: Code generating ground truth and predicted values.

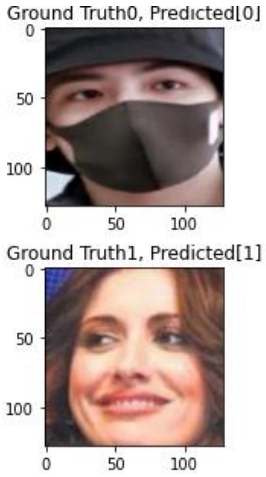

Fig. 9.: Ground truth and prediction values.

We also evaluated classification model's performance metric in terms of accuracy, precision, recall, and F~~f~~1-~~score~~ as shown in Fig. 10.

```
pred_labels = []
for image_path in (test_images):
  image = read_img(image_path)
  y_pred = model.predict(np.expand_dims(image,axis=0))
  y_pred = np.argmax(y_pred,axis=1)
  pred_labels.append(y_pred)

print(classification_report(test_labels, pred_labels))
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	807
1	0.99	0.99	0.99	809
accuracy			0.99	1616
macro avg	0.99	0.99	0.99	1616
weighted avg	0.99	0.99	0.99	1616

Fig. 10.: Performance metrics of classification model.

We first tried, to check whether faces are detected in an image having ~~more~~ a large number of people, i.e., to check whether our RetinaNet model works properly or not. The results are shown in Fig. 11.

Fig. 11.: RetinaNet model results.

Then, at the end, we implemented our database to our model to see the desired output as shown in Fig. 12.

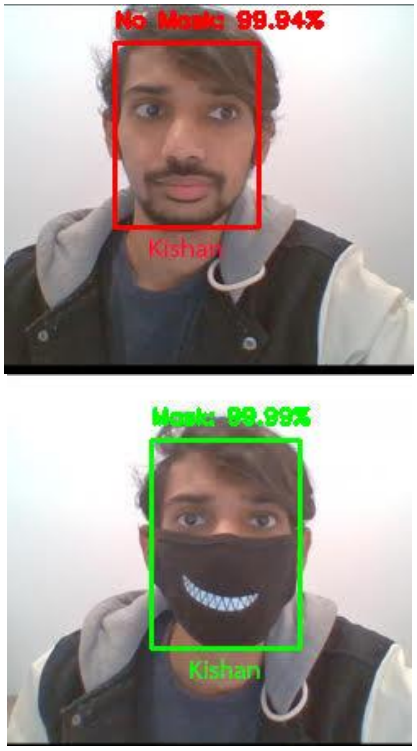

Fig. 12. Mask or no mask.

Thus, the loss and F1-score of our model were major factors that we looked at for numerical analysis. Our mask detection model works with higher precision and recall of 0.99. Our model could detect a person's face with or without mask with more than 90% accuracy. Our model achieves a high F1-score that shows that it can perform well in real world scenarios to classify with certainty, the mask and without mask categories on face crops. It also recognizes whose face it detects, i.e., identifies a person, in any scenario, if it is in its database.

TABLE I
LOSS AND F1-SCORE

Results obtained on the classification model:

- ***Loss***: 0.0182
- ***F1 score*** 0.99

We successfully implemented Mask Detection Model by using RetinaNet model because of its high precision and recall performance and as it uses focus loss. Xception model was implemented because it uses depthwise separable convolutions, giving better performance on ImageNet database. By combining the two, we were able to recognize faces from an image input file, in which people wore masks. It was able to detect faces with masks as well. Our model delivers better performance than CNN models or models that uses facial landmark algorithms for facial recognition. We can use this model during the pandemic crisis. We would like to use this model to predict a person's face, as well, in the near future. We would like to create a solid general database that can identify famous personalities, celebrities, etc. in the future.

REFERENCES

[1] Andrew Jason Shepley, "Deep Learning for Face Recognition: A critical analysis," July 2019

[2] Wael Abd Almageed et al., "Face Recognition Using Deep Multi-Pose Representation," March 2016

[3] Manik Sharma, J.Anuradha, H K Manne and G S CKashyap, "Facial Detection Using Deep Learning," 14th ICSET 2017

[4] Shikang Yu, Hu Han, Shiguang Shan, Antitza Dantcheva and Xilin Chen, "Improving Face Sketch Recognition via Adversarial Sketch-Photo Transformation," 14th International Conference on Automatic Face and Gesture Recognition 2019

[5] Yiming Lin, Shiyang Cheng, Jie Shen and Maja Pantic, " MobiFace: A Novel Dataset for Mobile Face Tracking in the Wild," 14th International Conferenece on Automatic Face and Gesture Recognition 2019

[6] Philipp Werner, Frerk Saxen, Ayoub Al-Hamadi and Hui Yu, "Generalizing to Unseen Head Poses in Facial Expression Recognition and Action Unit Intensity Estimation," 14th International Conference on Automatic Face and Gesture Recognition 2019

[7] Nate Crosswhite, Jeffrey Byrne, Chris Stauffer, Omkar Parkhi, Qiong Cao and Andrew Zisserman, "Template Adaptation for Face Verification and Identification," IEEE 12th International Conference on Automatic Face and Gesture Recognition 2017

[8] Itir Onal Ertugrul and Hamdi Dibeklioglu, "What Will Your Future Child Look Like? Modeling and Synthesis of Hereditary Patterns of Facial Dynamics," IEEE 12th International Conference on Automatic Face and Gesture Recognition 2017

[9] Weixuan Chen and Rosalind W. Picard, "Eliminating Physiological Information From

Facial Videos," IEEE 12th International Conference on Automatic Face and Gesture Recognition 2017

[10] Ching-Hui Chen, Jun-Cheng Chen, Carlos D. Castillo and Rama Chellappa, "Video-based Face Association and Identification," IEEE 12th International Conference on Automatic Face and Gesture Recognition 2017

[11] Zhenliang He, Meina Kan, Jie Zhang, Xilin Chen and Shiguang Shan, "A Fully End-to-End Cascaded CNN for Facial Landmark Detection," IEEE 12th International Conference on Automatic Face Recognition 2017

[12] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He and Piotr Dollar, "Focal Loss for Dense Object Detection," Facebook AI Research (FAIR), Feb. 2018

[13] Zaeem Ahmad Varaich and Sidra Khalid, "Recognizing Actions of Distracted Drivers Using Inception v3 and Xception Convolutional Neural Networks," IEEE 2nd International Conference on Advancements in Computational Sciences (ICACS) 2019

[14] Gregory Castanon and Jeffrey Byrne, "Visualizing and Quantifying Discriminative Features for Face Recognition," IEEE 13th International Conference on Automatic Face and Gesture Recognition 2018

[15] Zhihao Peng, Wei Zhang, Na Han, Xiaozhao Fang, Peipei Kang and Luyao Teng, "Active Transfer Learning," IEEE Transactions on Circuits and Systems for Video Technology, April 2020

[16] Francois Chollet, "Xception: Deep Learning With Depthwise Seperable Convolutions," arXiv, April 2017