HexaFace: A Deep Learning-based Web Portal for Facial Applications

http://hexaface.ddns.net/

Project Report for CMPT 733
Spring 2020, SFU
Big Data Squad
Mohammad Javad Eslamibidgoli, Olawunmi Oluwatoba Kolawole, Shahram Akbarinasaji, Ruihan Zhang, Ke Zhou

1. Background and motivation

Just a few years ago, we could only see facial recognition used in sci-fi movies. Nowadays, facial-based applications are everywhere due to the development of deep learning in image recognition. We detect and identify the driving environment in auto-pilot. We unlock our phones by facial verification. We monitor and track real-time data to make a safer city. In this project, we would like to explore the magic in a deep learning-based portal for facial applications. We are aware of the top face recognition and analysis APIs: Amazon Rekognition, Kairos, Trueface.ai, Face++, IBM Watson's Visual Recognition and Microsoft Face API.

Advancements in deep learning and in particular, convolutional neural networks have significantly reduced preprocessing steps in image recognition tasks. As an example, in classical approaches like bag of visual words pipeline for image classification, one needs to extract features from the images and generate a visual dictionary similar to approaches adopted in natural language processing, then learn the visual dictionary using clustering methods like k-means, next, generate histogram of visual words, and finally train and test the data to classify the images using a classifier such as support vector machines. Instead of this rather expensive approach which also requires significant storage, in a typical CNN approach, we can train the model for image classification in an end-to-end process. Some classification tasks related to facial applications are gender prediction, age prediction or facial emotion recognition that we focused on in this project.

Another highly important task is face detection. Here it aims to find out the position of all faces in animage, usually framed by a rectangular bounding box. The input is an image and the output is several rectangular frame positions containing the face. Face detection is a simple task for humans. For the needs of social life, we have a special face detection module in our brain, which is very sensitive to human faces. Even for simple pictures, the brain can easily detect human faces and their respective expressions. Automatic face detection is the basis of all applications surrounding automatic face image analysis, including but not limited to: face recognition and verification, face tracking in monitoring scenarios, facial expression analysis, facial attribute recognition (gender/age recognition, face scores), image and video search, digital album organization and presentation, etc. Commercial digital cameras have built-in face detection to assist auto-focus. Many social networks such as Facebook use the face detection mechanism to implement person labeling.

Face recognition is a method of identifying or verifying the identity of an individual using their face. Face recognition systems are used to identify people in photos, videos, or in real-time. A facial recognition system uses biometrics to map facial features from a photograph or video. It then compares the information with a database of known faces to find a match. Face recognition is a general topic that includes both face identification and face verification. Validating a claimed identity and either accepting or rejecting the identity claim is the responsibility of face verification (one-to-one matching). While the goal of face identification is to identify a person based on the input image of a face. This face image has to be compared with all the faces in the dataset (one-to-many matching).

And last but not least, Generative Adversarial Networks (GANs) are one of the most interesting deep learning methods we have today. We can generate imaginary fashion models without the need for photographers or makeup artists, upscale low-resolution imagery to high-resolution bitmaps and we can even improve astronomical images and model distributions of dark matter. The applications are endless. Prominent Computer Scientist Yann LeCun described GANs as "the coolest idea in machine learning in the last twenty years”.

2. Problem statement

The focus of this project revolves around 6 facial applications: face detection, face verification, face synthesis, gender and age prediction, and facial emotion recognition. In other words, we aim to develop models that address the following 6 questions:

1. Can our model detect multiple human faces in an image?
2. Does it recognize two different faces?
3. How about gender prediction based on a face?
4. What about estimating age?
5. Or facial emotions?
6. Can it generate new faces from a given set of face images? How does noisy training data (imbalanced data) affect our model?

These tasks are important for user identification and authentication, secure access control systems for ATMs and phones, criminal identification, helping blind people, face anonymization in privacy protection, smart advertising, automated monitoring systems for seniors, and patients, and so on. We created a full-stack computer vision model using Django for the above six applicable face tasks (hence we named the app *HexaFace*). In short, we trained several models on appropriate datasets with deep learning methods and evaluated them separately for each task, then we developed a web application in the Django framework for deploying these models.

There are several challenges in building a deep learning model for facial applications. First, we performed an extensive literature review to find appropriate datasets and methodologies for each task. For some tasks, we tried various datasets and methods to achieve a desirable performance like for age estimation and face verification tasks. The other inherent challenges are changes in position, pose, lighting, or expressions in faces in the dataset. Appropriate preprocessing was required to extract relevant facial images as we did for the gender prediction task. Datasets are usually imbalanced and noisy which leads to low performance or overfitting problems. We overcome this to some extent by combining different datasets in facial emotion recognition tasks as well by using data augmentation techniques. The training process of deep neural networks is computationally expensive, therefore, for all tasks, we used cloud computing to train the models. Hyperparameter tuning was another challenge that required weeks of experience with CNNs. Overall we had the opportunity to learn from all these challenges.

3. Data science pipeline

The general data science pipeline is shown in Figure 1. For data collection, we download the appropriate dataset for each task and evaluate their quality in terms of resolution, variations of age, ethnicity, and background against similar datasets to choose the best one. Next, the overall steps in data preparation and model development include: 1. Preprocessing and splitting the data, 2. Choosing network architecture, 3. Initializing the weights, 4. Finding a learning rate and regularization strength, 5. Minimizing the loss function and monitoring the progress, and 6. Evaluating the models on the test set. Finally, for model deployment, we develop a web application in the Django framework that uses HTML, CSS, and JavaScript on the client-side, and on the server-side, it employs Python and TensorFlow to deploy the deep learning models.

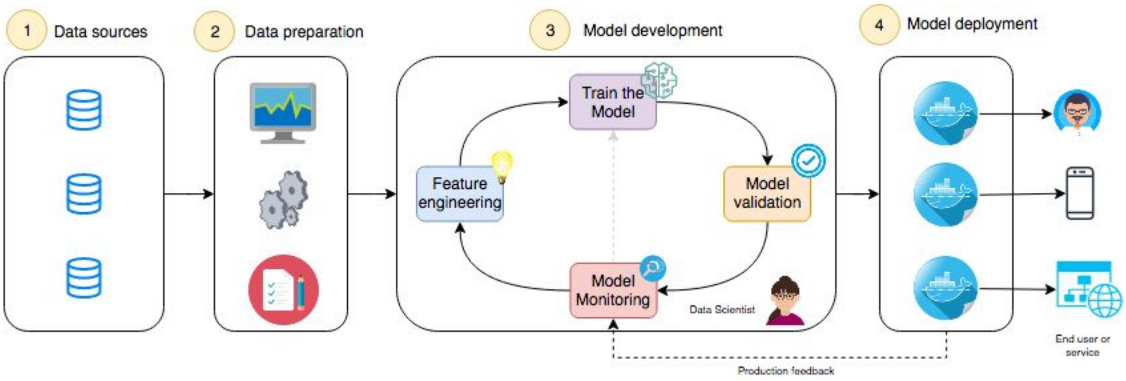


Figure 1) Data science pipeline

3.1 Face generation

3.1.1 Data collection and preprocessing

Figure 2 shows a typical pipeline for the face generation task. We acquired a dataset of 60,000 high-quality celebrity faces from CelebA dataset: https://www.kaggle.com/jessicali9530/celeba-dataset. All images were globbed and pushed into our preprocessing pipeline. Each image is decoded into a 3 channel RGB jpeg and resized to be 64 pixels wide and 76 pixels tall. All values in the pixel were converted into floats to prepare for the normalization process. Pixel values were normalized from 0 to 255 to numbers between -1 and 1. Each image was shuffled and placed into batches of 256 elements. TensorFlow caching methods were implemented as well to guarantee faster repeated processing.

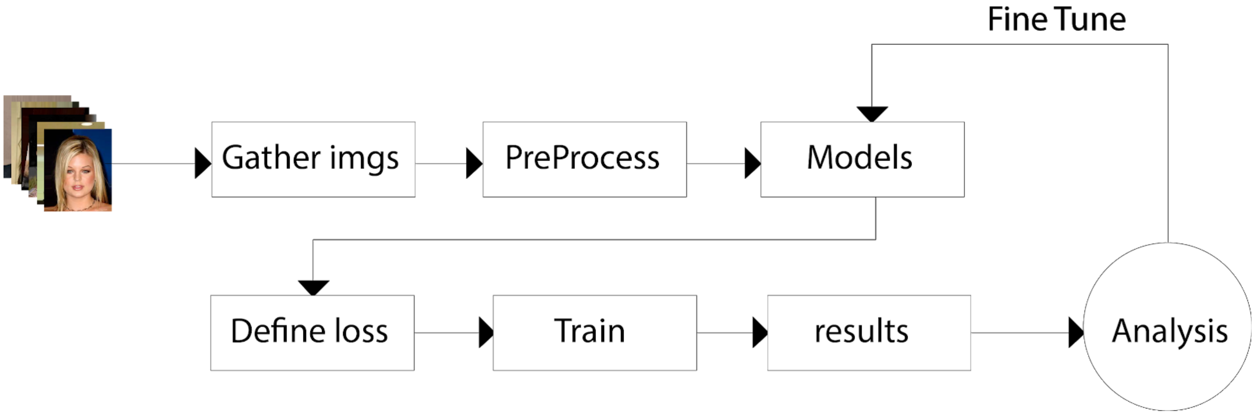


Figure 2) Face Generation Pipeline

3.1.2 Model building and training process

We utilized the DCGAN for facial synthesis (see section Methodology for details). Adversarial Network training is very difficult. Both models have to be trained to complete a zero-sum game. This means that an improvement to one model comes at the cost of the other model. The generator will not improve if the discriminator is too perfect and vice versa. This results in an unstable training process that can easily lead to failure states. For example, Generating the same faces all the time or generating noise. To mitigate this, training was done with two Adam optimizers one for each model in 190 epochs. Every training step, both optimizers are applied to their respective models until convergence is reached. It took 2 days and an NVIDIA 6GB GPU to complete the epochs. Next, outputs were analyzed during this stage. With human inspection, influential instances were identified and used to inform the next training cycle and parameter tuning. The skeleton code for our pipeline was obtained from https://www.tensorflow.org/tutorials/generative/dcgan.

3.1.3 Model results and product

Initially, our problem was only related to finding a way to generate synthetic faces. After principal research and trials, we had another problem: computational resources. There was a necessity to generate synthetic faces with limited time and resources at our disposal. We also needed to choose a model architecture and set hyperparameters that would facilitate relatively faster learning. In this domain, all experiments would be carried out on clusters of GPUs for multiple days or months but in our situation, room for experimentation was limited. Any hyperparameter tune or model change would take at least 6 hours to see any results so we could not explore different settings at leisure. With these constraints, our tools and analysis methods had to be deliberate, results-driven and incremental.

Empirical study and research. There are many advanced GAN models out there, styleGAN, GANILLA, SRGAN, and HDGAN to name a few. They presented amazing results, but due to the GPU times required, they were not possible for us to implement. We needed to find a model that could present decent quality outputs with lesser resources, as a result, we settled on the Deep Convolutional Generative Adversarial Network or DCGAN. It was created for scalability and stability during training while also leaving a low computational footprint.

Explainable AI. As stated previously, any model change or parameter setting takes multiple hours to observe any results. Our actions have to be informed and deliberate when taking steps to explain our model's behaviour. We employed a combination of different approaches to achieve this.

Post-hoc Explanation. We could not carry out this study ourselves but instead, we examined the results of researchers in the field. A deep explanation carried out in the paper Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks provided us with detailed explanations of model behaviour and visualized internals. In addition, it also provided architectural guidelines and best practices for implementation.

Naïve Approach. Combined with our internal knowledge, we compiled other DCGAN best practices from different sources and conducted multiple cycles of naïve experiments to find influential instances. Implement a setting or parameter, obtain a generalized output, perform human inspection, improve the model either by applying or deleting best practice implementations.

3.2 Age Prediction

3.2.1 Data collection and preprocessing

For this task data was downloaded from IMDB & WIKI https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/, which contains 500+K face images with age labels. Pre-processing includes mainly two parts: data transformation and data cleaning. For data transformation, the date of birth is stored as Matlab, converted into python DateTime. Age was calculated by provided info in the data, photo, and date of birth. Cleaning steps include the deletion of pictures that do not contain faces, deletion of pictures that include more than one face, and deletion of paintings by removing ages that are greater than 100.

3.2.2 Model building and training process

Transfer learning is applied for building the model (see section Methodology for details). Loaded the model as the pre-model and froze all layers except for the last seven layers of the model, which contains three convolutional layers. We freeze the previous layers because it contains enough information for the image itself and the last 3 convolutional layers can perceive the info for the age. In the training process, the model was trained with 1000 epochs. In each epoch, 2000 images were selected randomly from the dataset and validation was applied in each epoch to check the performance of the model. We used a VGG-face model and freeze all the layers except for the last 7 layers that contain 3 convolutional layers. In each epoch, a random collection of 1000 images was going to the training process. The model is tested in a validation set for each epoch.

3.2.3 Model results and product

Naively, age estimation is a regression problem since we would like to get an output age based on the features we extracted from the images. However, we approach this problem as a classification problem for the following reasons. The first reason is that age estimation is a very challenging task since people can look different even if they are the same age. Also, many features would affect the apparent age such as race, facial emotion, make-up, etc. Tackled this problem as a classification problem would let us be more flexible since we can group the data by merging different units into the groups such as age 0, 1 and 2 into group 1 and 3, 4 and 5 can go to group 2. In this project, we define labels to 101 classes, from age 0 to age 100. We did not group the units into groups since we would like to explore how good our model could be. Another reason to treat this as a classification problem is that the prediction can reflect more info than a regression problem. There are two ways for the output prediction. The first one is to output the class with the highest probability. Also, because optimization is softmax, the prediction is a probability distribution of all the classes. We could multiply each class with their probability to get the final apparent age estimation. In this way, the model is reflecting more information compared with the regression model. The model was tested by the test dataset and showing an average of 6 for mean absolute error.

3.3 Gender Prediction

3.3.1 Data collection and preprocessing

For gender prediction, we used the Wikipedia dataset that includes 62,328 facial images **https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/**. This dataset has a metadata of face score, age and gender labels for each image. Higher face scores corresponds to a single frontal face in the image, and lower scores corresponds to those having multiple faces. For our study, we selected the images with scores above 5 in the wikipedia dataset that resulted in 3210 images in our dataset. We then converted the pixels to images in a folder structure and reshape them to 100 * 100 for training purposes.

3.3.2 Model building and training process

For gender prediction, we employed transfer learning using VGG16 (see section Methodology for details). In this approach the architecture and weights of a pretrained model is adopted to extract features from the image. The pretrained model (in this case VGG16) is already trained on a large benchmark dataset to solve a similar problem (in this case ImageNet dataset). In order to preprocess the images in an input format of VGG16, the target size needs to be set as 224×224 (size of images in ImageNet) and then be converted to an input array. The predicted output of

VGG16 is an array of probability distributions over 1000 categories (the number of classes in ImageNet). For age prediction, we exclude the top two dense layers of VGG16 architecture, thus, our model contains 13 convolutional layers plus 1 fully connected layer of VGG16 architecture. We apply this model on all the images in the training set of imdb-wiki dataset to extract the features and along with their encoded labels we create a dataset. Finally, we use this dataset to train a classifier like logistic regression. To train the classifier, we split the generated dataset to a training set and test set with the size of 0.2. Next, we instantiated several models such as logistic regression, random forest or a multi-layer perceptron, and fitted the classifiers on the training set and predicted on the test set. scikit-learn was used for this task.

3.3.3 Model results and product

Following tables show performance of the MLP classifier on the test set of extracted features from VGG16. More details on model evaluation are provided in the evaluation section of this report.

	Precision	Recall	F1-Score	Support
Male	0.92	0.90	0.91	359
Female	0.88	0.89	0.88	283

Table 1

Accuracy			0.90	642
Macro Avg	0.90	0.90	0.90	642
Weighted Avg	0.90	0.90	0.90	642

Table 2) MLP classifier performance on VGG extracted features for gender prediction

3.4 Face detection

3.4.1 Data collection

The training dataset used for face detection is the Face Wilder dataset. The dataset has 32,203 images and is available at http://shuoyang1213.me/WIDERFACE/. There is a text file in the dataset, named wider_face_train.txt, representing the bounding boxes position. Each line of this file has an image file name and its bounding boxes coordinates of the upper left corner and lower right corner.

Another used dataset for this task is the Face Landmark dataset which contains 5,590 LFW images and 7,876 other images downloaded from the web. The training and validation set are defined in trainImageList.txt and testImageList.txt respectively. Each line of these text files starts with the image name followed by the boundary positions of the face bounding box and positions of five facial points. Dataset is available at http://mmlab.ie.cuhk.edu.hk/archive/CNN_FacePoint.htm.

3.5 Facial emotion recognition

3.5.1 Data collection and preprocessing

For this task we combined the FER-2013 dataset https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data with the Facial Expression Research Group Database (FERG-DB). FERG-DB database https://grail.cs.washington.edu/projects/deepexpr/ferg-db.html. The former consists of 28,709 grayscale images of faces in 7 classes namely, 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral. The image size is 48x48 pixels. However, there are several limitations in this dataset that makes it challenging for classification tasks. For example, the distribution of classes is not balanced as shown in Figure 3. Some images are corrupted and noisy and some are incorrectly labeled. We had an attempt on this problem in CMPT 726, Machine Learning course but could not obtain accuracy better than around 60%. This time we improved our model by incorporating more images in the training set from FERG-DB dataset. FERG-DB consists of stylized characters with annotated facial expressions. The database contains 55,767 annotated face images of six stylized characters. The characters were modeled using the MAYA software and rendered out in 2D to create the images. This would resolve the possible overfitting issue as well as the problem of an imbalanced dataset. For this purpose, we resized the FERG-DB from 256x256 to 48x48 in grayscale, and prepared a dataset similar to FER-2013 then appended the two dataframes together for training purposes (Figure 4).

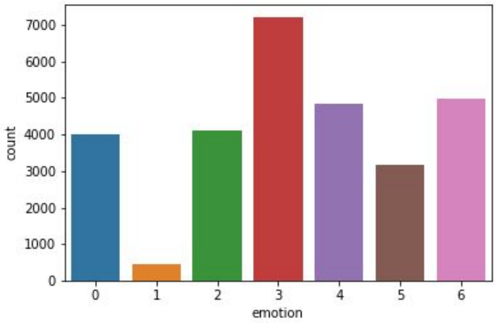


Figure 3) Class distribution in the training set of the FER-2013 dataset

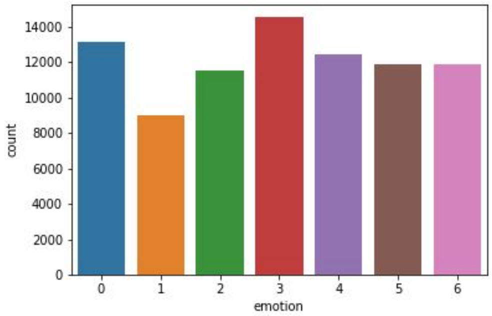


Figure 4) That for the combined FER-2013 and FERG-DB. (0:Angry, 1:Disgust, 2:Fear, 3:Happy, 4:Sad, 5:Surprise, 6:Neutral)

3.5.2 Model building and training process

In facial emotion recognition task, we developed a VGG-like CNN architecture from scratch (see section Methodology for details) which involves 6 convolutional layers (64, 64, MaxPool, 128, 128, MaxPool, 256, 256, MaxPool) and two fully connected layers (1024,1024) followed by Softmax for multi-label classification. For this aim, we employed TensorFlow and Keras to create the sequential CNN architecture. Stochastic Gradient Descent (SGD) with momentum was employed for optimization. Categorical cross-entropy with accuracy as the metric was used for training. Data augmentation was employed to randomly generate new data by shifting, zooming, stretching, etc. of the available images to reduce the overfitting problem. Figures 4 and 5 show the learning curves over 1000 epochs for FER-2013 as well as FER+FERG datasets. The accuracy is significantly improved in the latter case from 60% to 90%, which is promising. However, as can be seen, the model still suffers from overfitting problems, even though data augmentation and drop out techniques were used. The overfitting issue is less paramount for FER+FERG dataset as compared to FER-2013 dataset. To resolve the issue, more facial emotion data with more variety in features like age, gender, or ethnicity needs to be added to the training set.

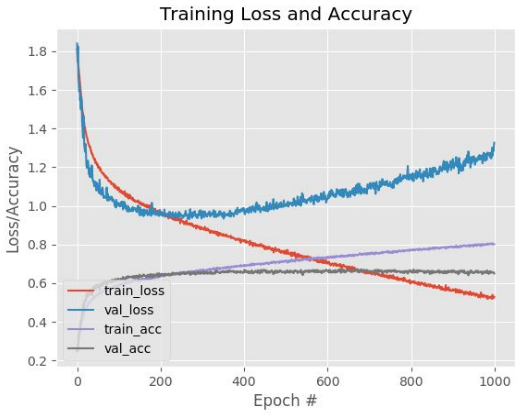
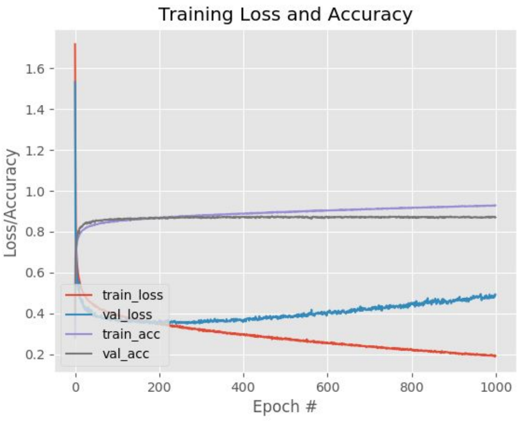


Figure 5) Training loss and accuracy in FER-2013 dataset



Training loss and accuracy in FER+FERG dataset

3.5.3 Model results and product

The following tables show the evaluation report for the final model used in the web application. More details on performance evaluations are provided in the Evaluation section of this report.

	Precision	Recall	F1-Score	Support
Anger	0.87	0.85	0.86	2680
Disgust	0.99	0.98	0.99	1821
Fear	0.83	0.80	0.82	2332
Joy	0.92	0.91	0.92	2870
Sadness	0.79	0.80	0.79	2411
Surprise	0.92	0.94	0.93	2366
Neutral	0.80	0.83	0.81	2416

Table 3

Accuracy			0.87	16896
Macro Avg	0.87	0.87	0.87	16896
Weighted Avg	0.87	0.87	0.87	16896

Table 4 CNN model performance on FER+FER-G dataset for facial emotion recognition

3.6 Face Recognition

Face recognition includes two main categories: face identification and face verification. Face identification aims to compare an input image with all faces in a database to discover if there is a match (one-to-many) while face verification is the task of comparing an input face with another face to answer whether they are a match or not (one-to-one).

3.6.1 Data collection and preprocessing

AT&T Database of faces is the target dataset to evaluate the performance of the system. There are 40 folders representing distinct persons in the dataset, and each folder contains 10 images of a person in different times, lighting, expression (open or closed eyes, smiling, etc.) and details (wearing glasses or not). The size of each image is 94x112, with 256 grey levels and all the images are in PGM file format. We generated a test set based on this dataset. The generated test set holds 240 pairs in which half of the pairs belong to the same person and the rest belongs to different persons.

The VGG-Face model accepts an input image of size 224x224 to classify. To realize this, a four-step preprocess is being performed. Initially, there is no guarantee that the input image is confined to the person's face and it may be a full body image. So, the MTCNN library is used to crop the face in the input image. The implemented model works with RGB images in JPG file format. As mentioned before, the dataset images are in PGM format which led us to converting the images of the dataset to JPG file format in RGB as the second step of preprocessing. Next, the detected face is converted to an array of 224x224. Finally, the input array is converted to float data type and normalized to be in range -1 and 1.

3.6.2 Model building and training process

Model architecture. Training a convolutional neural network from scratch for face verification has two limitations. It needs a large amount of labeled data and a prerequisite to retrain the model for every new person. In addition, training takes a lot of time and computational resources. So, transfer learning is being used for this purpose. The VGG-Face model, trained on 2622 identities with over 2.6M pictures, is being used as the base model. The architecture of this model is illustrated in Figure 6.

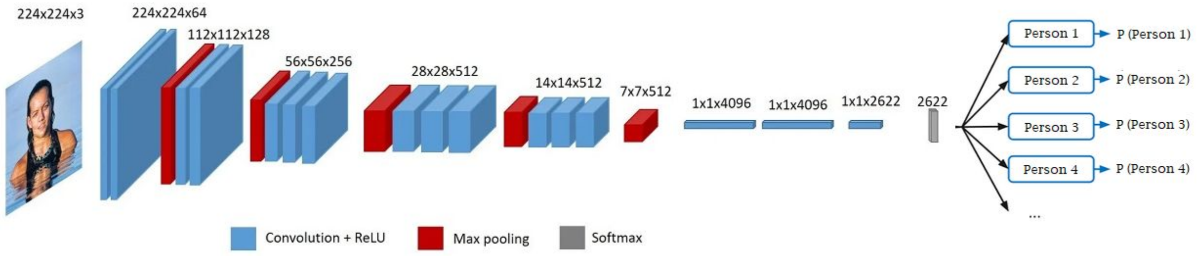


Figure 6) VGG-Face model architecture

Facial Features. The last layer, Softmax function, computes the probability of the input image belonging to one of 2622 identities. We remove the Softmax layer to obtain the output of the last layer *f(2622)* which represents a vector of 2622 facial features (Figure 7).

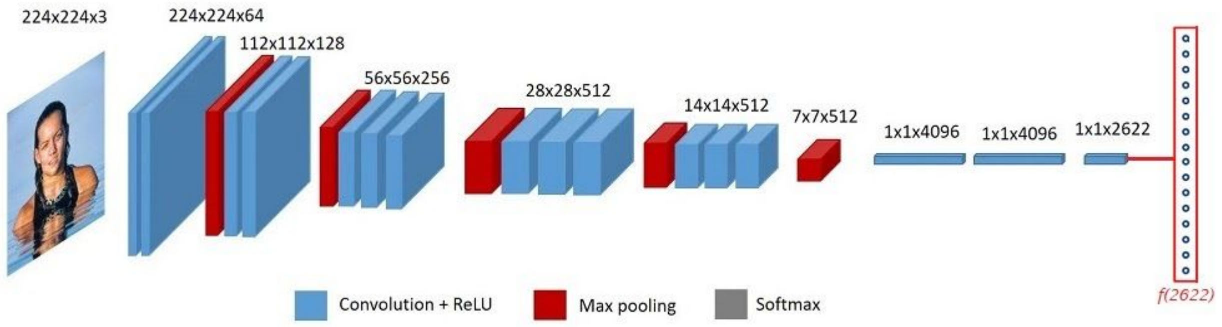


Figure 7) VGG-Face model architecture without softmax

Distance Function. In order to compare if two faces belong to the same person or not, there is a requirement to compare input feature vectors against each other to find a similarity score. Cosine and Euclidean distance are the popular similarity scores used for face verification denoting the chance that input images belong to the same person or not. Following equations show how each of these similarity scores can be computed:

$$euclidean(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2}$$

$$cosine(p,q) = \frac{P.Q^T}{\|P\|.\|Q\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}.\sqrt{\sum_{i=1}^{n} B_i^2}}$$

These values will be small for the faces of one person, but large for a pair of different people's faces. For the cosine similarity score, values are near 0.

4. Methodology

4.1 Convolutional Neural Networks for image classification

Among the six applications, we focused on, gender, age, and facial emotion predictions are classification tasks. For this aim we developed CNN models from scratch, as well, we used transfer learning as explained in the following.

The figure shows a CNN architecture of VGG16 (a.k.a OxfordNet) proposed in 2014 by Visual Geometry Group at the University of Oxford and won the ImageNet competition for image classification in the same year, which involves 13 convolutional layers and 3 fully connected layers. In a typical CNN architecture (Figure 8), convolutional layers are applied to an input image to extract features from the image by applying various kernels to generate so-called activation maps. The activation maps have larger depth but smaller spatial dimensions in comparison to the input image. Each activation map in a convolutional layer basically interacts with a certain part of the image to detect features such as corners, edges, color distributions or various shapes. The deeper the network the more complicated features can be extracted from the image. After convolution, fully connected layers are applied to generate a 1D feature vector from the activation maps of the last convolution. Finally for image classification, a classifier is applied to output the probability distribution over the categories. Other operations in a CNN architecture involve applying activation functions like Rectified Linear Unit (ReLU) to increase non-linearity, Max pooling for down-sampling, flattening to convert feature maps to a single column representation for fully connected layers, as well as dropout layers to reduce the overfitting problem by randomly killing some neurons.

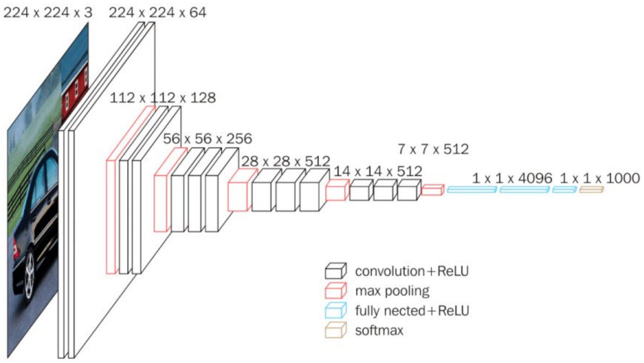


Figure 8) VGG16 Convolutional Neural Network architecture

4.2 Transfer Learning

Due to the relatively high amount of time and computational resources required to train a ConvNet model from scratch, transfer learning is usually adopted which instead of training a new CNN model, the architecture and weights of a pretrained model can be adapted to extract features. A pretrained model is already trained on a large benchmark dataset to solve a similar problem. The most popular pre-trained models for image classification is VGG16 (also called OxfordNet). These models can achieve face recognition or object classification. In this project, for gender prediction, we employed VGG16, for age prediction and face verification we employed the VGG-Face model.

4.3 Multi-task Cascaded Convolutional Network (MTCNN)

The Multi-task Cascaded Convolutional Network (MTCNN) model is an excellent model for face detection. This model has three convolutional networks (P-Net, R-Net, and O-Net) and is able to outperform many face-detection benchmarks while retaining real-time performance.

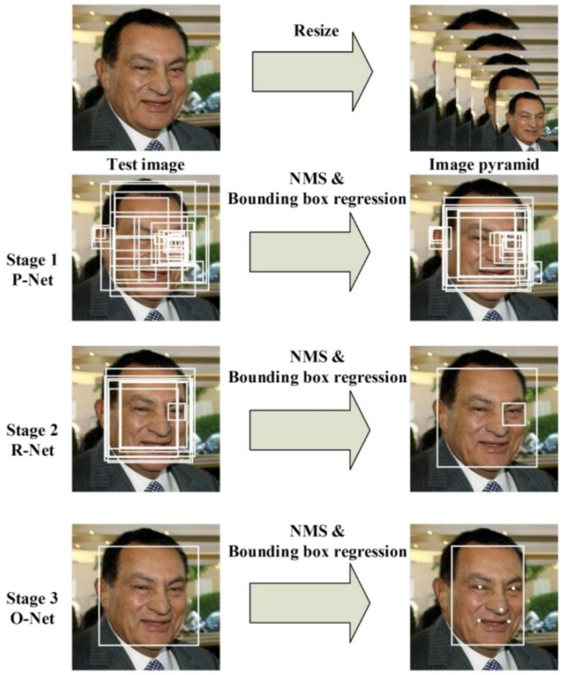


Figure 8) Overall structure of MTCNN

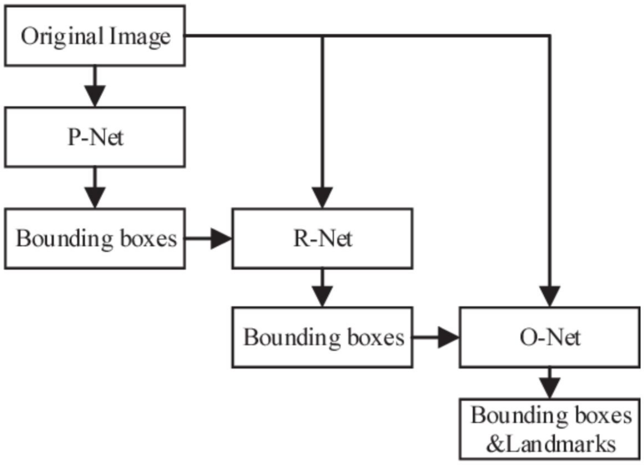


Figure 9) Workflow of MTCNN

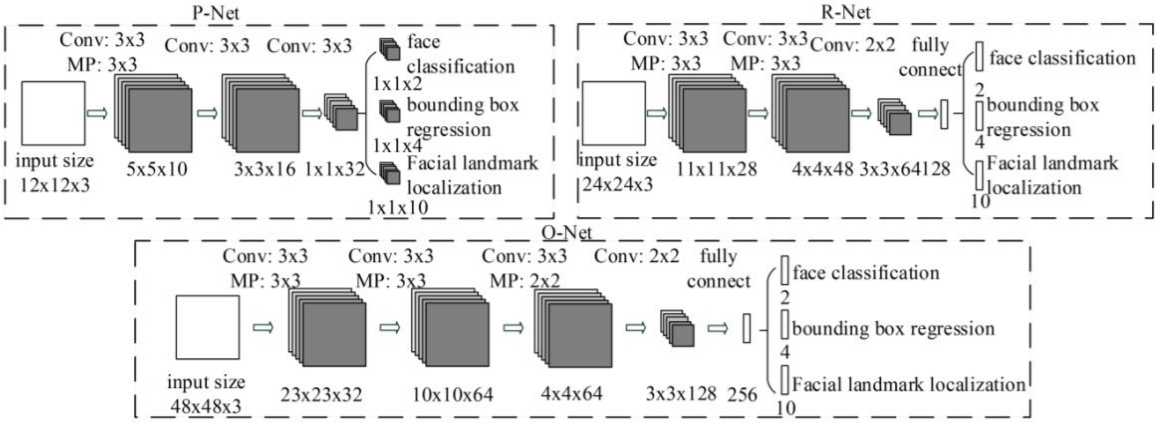


Figure 10) MTCNN model architecture

Multi-task training.

- main tasks: face classification, face bounding boxes regression,
- For each layer, there are three types of errors, but for different training data, the three types of errors should have different weights, so the following formula is proposed(Alpha is the weight of each type of error, and each layer has a different weight).

$$\min \sum_{i=1}^{N} \sum_{j\in\{det,box,landmark\}} \alpha_j \beta_i^j L_i^j$$

Build training data.

- Data types:
    1. positives: The IOU with ground truth is above the threshold.
    2. negatives: The IOU with ground truth is below the threshold.
    3. part faces: The IOU with ground truth is between the above two thresholds.
    4. landmark faces: Tags with landmarks.
- Usage:
    1. Face classification tasks use positives & negatives.
    2. Bounding box prediction uses positives & part faces.
    3. Landmark localization prediction uses landmark faces.
- Online Hard Sample mining
    1. In each mini-batch, calculate all losses and sort them in descending order.
    2. Get the first 70% of the samples for gradient descent training

Loss function.

- Face classification error, whether the model predicts a face:

$$L_i^{det} = -(y_i^{det} \log(p_i) + (1 - y_i^{det})(1 - \log(p_i)))$$

- Bounding box regression error, the model predicts the offset, not the bbox itself:

$$L_i^{box} = \left\| \hat{y}_i^{box} - y_i^{box} \right\|_2^2$$

- The landmark localization error predicts the following key points of the human face: eyes, nose, and two corners of the mouth.

$$L_i^{landmark} = \left\| \hat{y}_i^{landmark} - y_i^{landmark} \right\|_2^2$$

4.4 Siamese Neural Network

We construct a Siamese Neural Network for face recognition which is basically two identical networks with the same parameters and weights. The pretrained VGG-Face model forms our Siamese Neural Networks. To make it clear, the following figure shows how the constructed network operates. But in reality, there is only one network available. Initially, the first input image is passed to the network to obtain the feature vector x(1) and then the second input image will be passed to the same network to obtain feature vector x(2). Finally, both feature vectors are sent to the implemented distance function and the result is compared to the determined threshold.

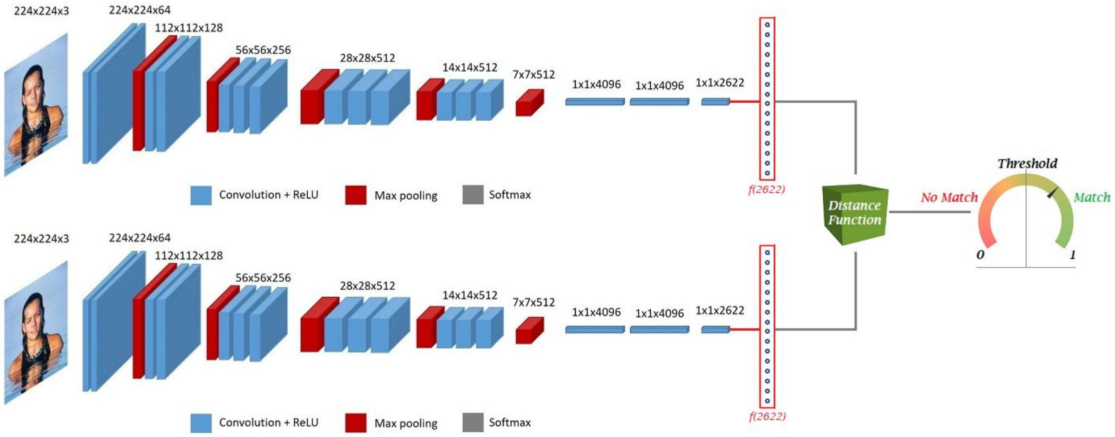


Figure 11) Siamese Neural Network

4.5 Deep Convolutional Generative Adversarial Network

We utilized the DCGAN (Deep Convolutional Generative Adversarial Network) which consists of two models. A generator model trained for generating synthetic imagery and a discriminator model trained for telling if generated samples are real, from the training data or fake.

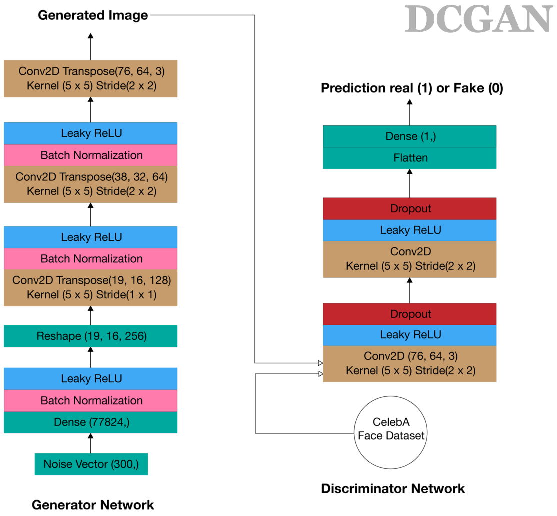


Figure 12) DCGAN Model Architecture

The generator consists of strided convolution-transpose layers, batch normalization layers, and LeakyReLU activations. The discriminator is composed of strided convolution layers, batch normalization layers, LeakyReLU activations, and dropout layers. The generator takes a noise description vector as input and outputs a generated face while the discriminator takes both the generated face and the faces from our dataset as inputs. It learns to identify the face dataset as real (Label 1) and the generated faces as fake (Label 0).

Two loss functions were defined. One for the generator and another for the discriminator. The generator loss is a cross-entropy between a vector of 1s and the output of the discriminator's prediction on the generated face. i.e the generator is trying to make its outputs closer to the real label of 1. The discriminator loss is an addition of two cross entropies. The cross-entropy between a vector of 1s and the discriminator's prediction on real faces and the cross-entropy between a vector of 0s and the discriminator's prediction on fake faces. The discriminator learns to classify real faces as 1 and fake faces as 0.

$$\min_G \max_D V(D,G) = {}_{x\sim P_{data}}(x)[logD(x)] + {}_{z\sim p_z}(z)[log(1-D(G(z)))]$$

D(x) is the scalar probability that the output of the generator is taken from the real training data. D(G(z)) is the scalar probability that the output of the generator is from pz. The discriminator is maximizing the probability that it is correctly classifying the output as real or fake. The generator is maximizing the probability that the discriminator will incorrectly classify fake output as real. This is expressed as minimizing 1−D(G(z)). The generator consists of strided convolution-transpose layers, batch normalization layers, and LeakyReLU activations. The discriminator is composed of strided convolution layers, batch normalization layers, LeakyReLU activations, and dropout layers. The generator takes a noise description vector as input and outputs a generated face while the discriminator takes both the generated face and the faces from our dataset as inputs. It learns to identify the face dataset as real (Label 1) and the generated faces as fake (Label 0).

5. Model Evaluations

5.1 Facial Emotion Recognition

Evaluating the network on the FER-2013 dataset

	Precision	Recall	F1-Score	Support
Anger	0.54	0.57	0.56	777
Disgust	0.83	0.56	0.67	94
Fear	0.54	0.45	0.49	816
Joy	0.87	0.83	0.85	1436
Sadness	0.50	0.59	0.54	910
Surprise	0.77	0.76	0.77	625
Neutral	0.60	0.62	0.61	1084

Table 5)

Accuracy			0.65	5742
Macro Avg	0.66	0.63	0.64	5742
Weighted Avg	0.66	0.65	0.65	5742

Table 6) CNN model evaluation on FER-2013 dataset for facial emotion recognition

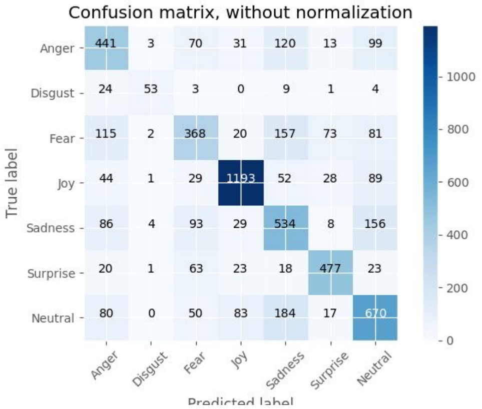


Figure 12) Confusion matrix without normalization on FER-2013 dataset

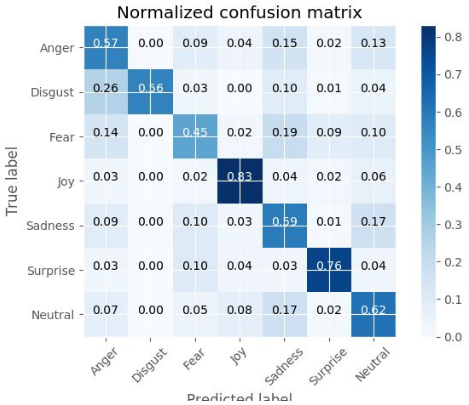


Figure 13) Normalized confusion matrix on FER-2013 dataset

Evaluating the network on the combined FER-2013 and FERG-DB dataset

	Precision	Recall	F1-Score	Support
Anger	0.87	0.85	0.86	2680
Disgust	0.99	0.98	0.99	1821
Fear	0.83	0.80	0.82	2332
Joy	0.92	0.91	0.92	2870
Sadness	0.79	0.80	0.79	2411
Surprise	0.92	0.94	0.93	2366
Neutral	0.80	0.83	0.81	2416

Table 5)

Accuracy			0.87	16896
Macro Avg	0.87	0.87	0.87	16896
Weighted Avg	0.87	0.87	0.87	16896

Table 6) CNN model performance evaluation on FER+FERG dataset

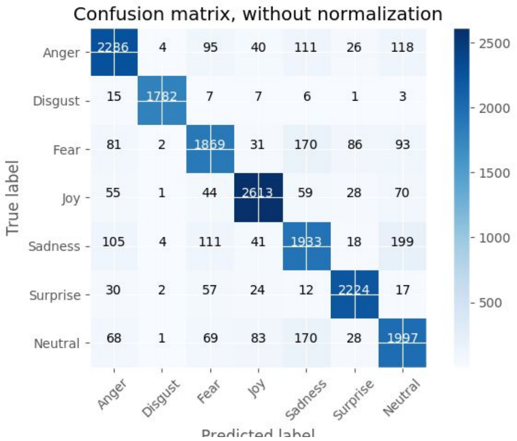


Figure 14) Confusion matrix without normalization on FER+FERG dataset

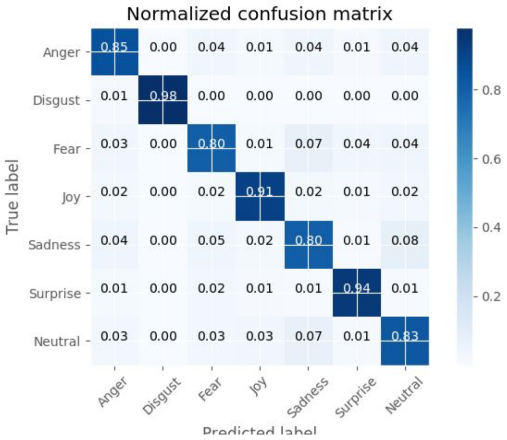


Figure 15) Normalized confusion matrix on FER+FERG dataset

5.2 Gender prediction

Evaluating logistic regression on the test set of extracted features from VGG16:

	Precision	Recall	F1-Score	Support
Male	0.91	0.91	0.91	359
Female	0.88	0.89	0.88	283

Table 7)

Accuracy			0.90	642
Macro Avg	0.90	0.90	0.90	642
Weighted Avg	0.88	0.89	0.88	283

Table 8) Logistic regression evaluation on the test set of extracted features from VGG16

Evaluating MLP on the test set of extracted features from VGG16

	Precision	Recall	F1-Score	Support
Male	0.92	0.90	0.91	359
Female	0.88	0.89	0.88	283

Table 9)

Accuracy			0.90	642
Marco Avg	0.90	0.90	0.90	642
Weighted Avg	0.90	0.90	0.90	642

Table 10) MLP evaluation on the test set of extracted features from VGG16

Classifier	Accuracy	Weighted Average precision	Weighted Average recall	Weighted Average f1-score
Logistic regression	0.90	0.90	0.90	0.90
MLP	0.90	0.90	0.90	0.90
Decision Tree	0.78	0.78	0.78	0.78
Random Forest	0.88	0.88	0.88	0.87

Table 11) Comparison of the different classifier performances on the test set of extracted features from VGG16

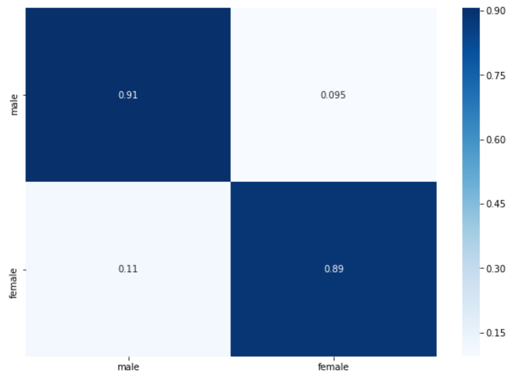


Figure 16) CM for LR

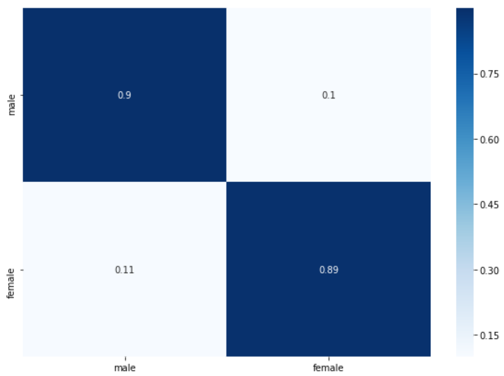


Figure 17) CM for MLP

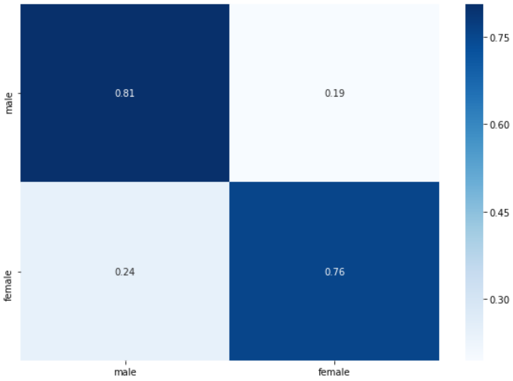


Figure 18) CM for RF

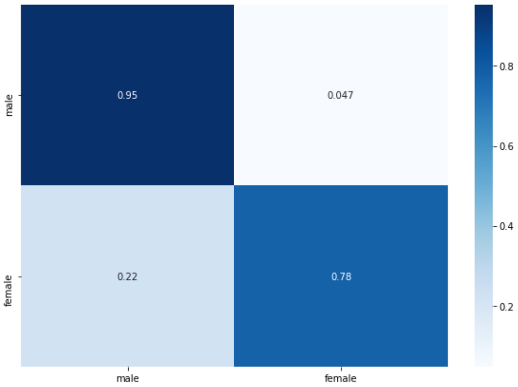


Figure 19) CM for DT

Confusion matrices of logistic regression, MLP, random forest and decision tree classifiers.

5.3 Face Recognition

The purpose of this task is to figure out if we can apply face verification to different domains. We used the AT&T Database of Faces to test the performance of the constructed Siamese Neural Network. The test set is organized in a way such that the half of the test set contains pairs of the same person, while the other half holds pairs of different persons. The cosine distance and Euclidean distance of the input images' feature vectors are compared against a threshold to realize if they are a match or not. The following table shows the results of testing the model with different threshold values of 0.20, 0.30, 0.40, and 0.50.

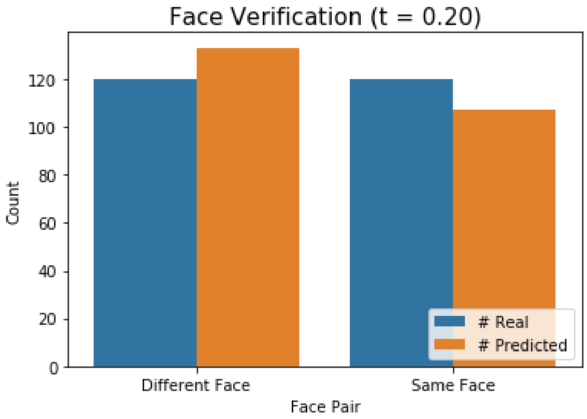
<table>
<tr><td rowspan="5">

Figure 19) Verification with t=0.20</td><td>Metric</td><td>Value</td></tr>
<tr><td>Accuracy</td><td>0.94</td></tr>
<tr><td>Precision</td><td>1.00</td></tr>
<tr><td>Recall</td><td>0.89</td></tr>
<tr><td>F1-Score</td><td>0.94</td></tr>
</table>

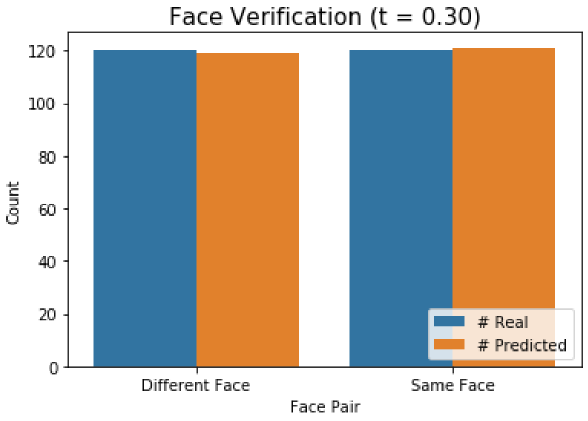


Figure 20) Verification with t=0.30

Metric	Value
Accuracy	0.98
Precision	0.98
Recall	0.99
F1-Score	0.98

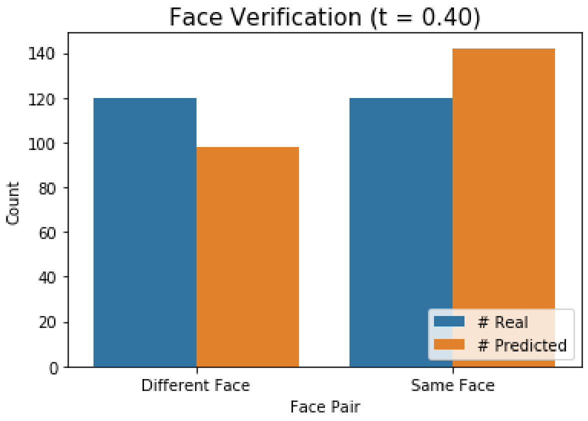


Figure 21) Verification with t=0.40

Metric	Value
Accuracy	0.90
Precision	0.84
Recall	1.00
F1-Score	0.91

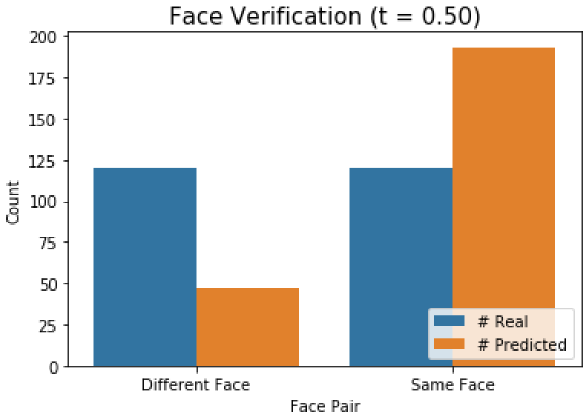


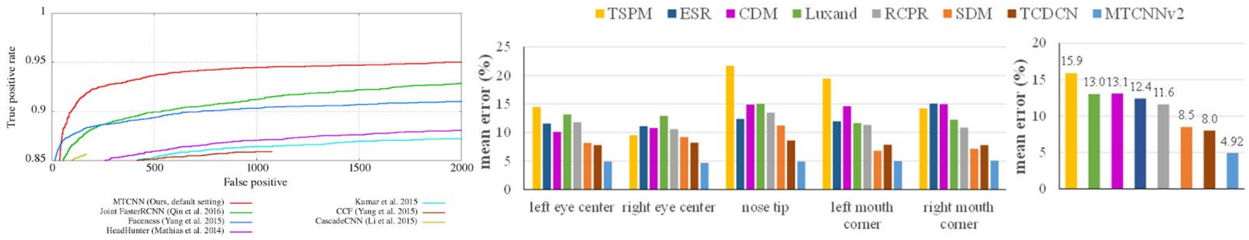
Figure 22) Verification with t=0.50

Metric	Value
Accuracy	0.69
Precision	0.62
Recall	1.00
F1-Score	0.76

The performance of the system heavily depends on an appropriate selection of the threshold i.e. this is the operating domain which determines the optimal value of the threshold. In criminology, in order to verify if an arrested criminal is the one we are looking for, a higher threshold is preferred, however for user authentication, we need to consider a stringent one.

5.4 Face Detection

For the face detection part, we mainly reproduce the MTCNN model by Tensorflow from the popular paper “Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks”. The MTCNN model has high real-time performance and the accuracy achieves at 95%. To evaluate the performance of face detection, the paper compares MTCNN against the state-of-the-art methods in FDDB, and the state-of-the-art methods in WIDER FACE. The figure shows that the method consistently outperforms all the previous approaches by a large margin in both the benchmarks. In the evaluation of face alignment, the paper compares the face alignment performance of MTCNN against the following methods: RCPR, TSPM, Luxand face SDK, ESR, CDM, SDM, and TCDCN. The figure shows that MTCNN outperforms all the state-of-the-art methods with a margin.



(c) Result on FDDB for face detection

(d) Result on AFLW for face alignment

Figure 23) Face detection results adopted from Ref. 8

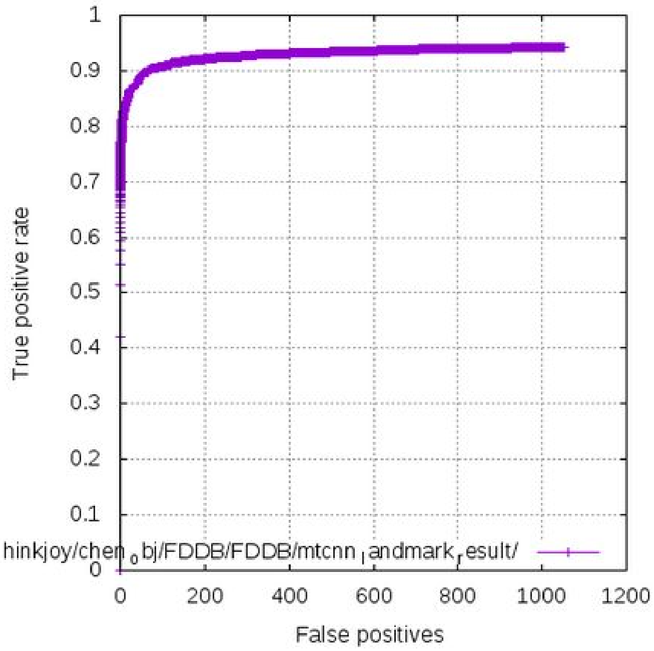


Figure 24) MTCNN performance evaluation adopted from Ref. []

5.5 Face Synthesis

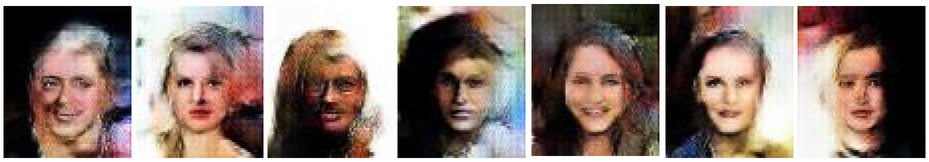


Figure 25) Random samples of generated faces

Our goal was to explore, explain model behaviour, and generate decent quality samples with the limited computational resources at our disposal. Our usage of TensorFlow, DCGAN, our exploratory methodologies and problem-solving lead us here. Results are of decent quality, in addition, we were able to observe how noisy training data (imbalanced data) affected our model? Our model was trained with a higher ratio of female samples, to prove this, we generated 60 faces 10 times with our web app. We used our gender detection model to extract gender information before plotting a graph using JavaScript.

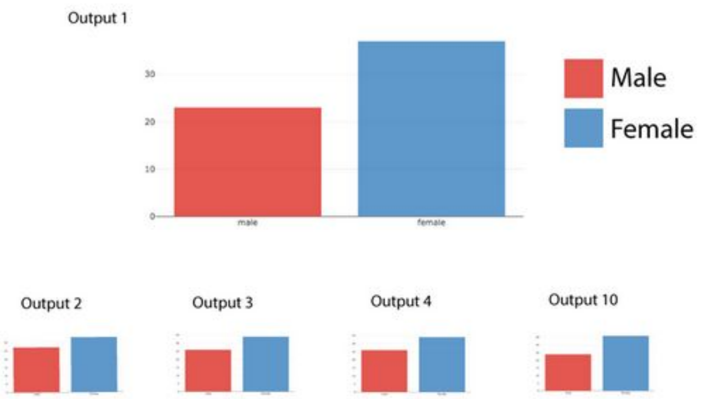


Figure 26) Synthesis on training data with more women

Our gender detection model had an accuracy of 90% and it is natural to expect some noise. Even with this expectation, more female samples were recorded each time.

5.6 Age Prediction

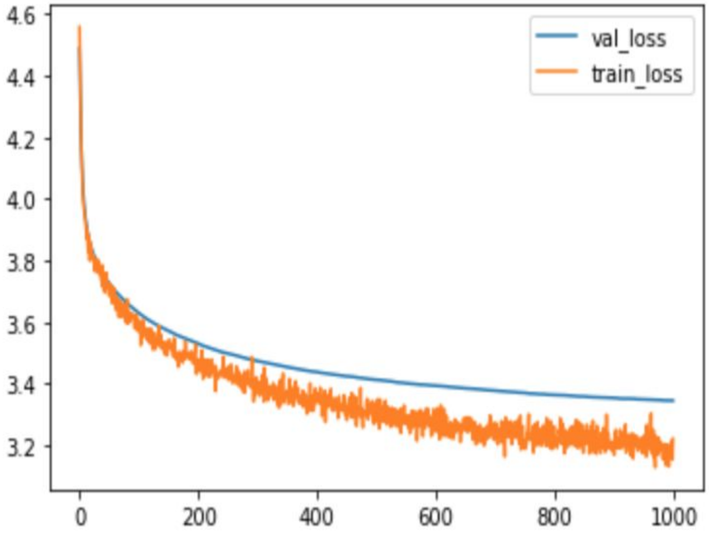


Figure 27) Age prediction training and validation loss

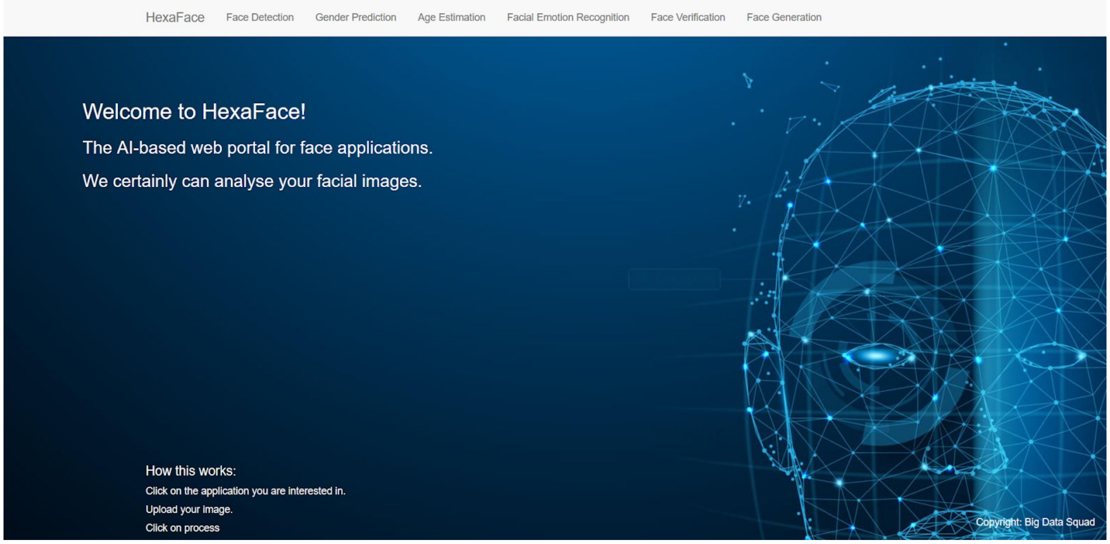
The training loss and validation loss are shown above for 1000 epochs. The validation error is decreasing as the epoch increases. There is no overfitting problem in this training process because for each epoch, only small and random portions of the images are selected and fed into the model. The MAE is around 6 for age estimation. Overall results are acceptable since age estimation itself is a challenging task based on only one image of the person and also the perceived age depends on a lot of factors. Also, people try very hard to hide their real age!

6. Data Product

We have developed a deep learning-based web portal for a facial application named HexaFace; **http://hexaface.ddns.net/**. Django framework is being used to build the web application. We used Python to develop the backend of the portal. Convolutional Neural Networks is the basis for different applications in this project which is being used widely in this project.

As a whole, a combination of these facial applications can be used for different purposes such as user identification and authentication in electronic transactions, finding missing persons, smart advertising, automated driving safety systems, etc. One of the promising domains to apply this application is criminology. Consider the time that police officers are looking for a criminal with some specific information. What if they arrest a criminal and suspects if they have arrested the right person or not. Using this application, they can take a picture of the arrested person and examine the data obtained in this application against the real data they have. These data can be gender, age (is that person the same as the age range of the person we are looking for?), emotion (is the arrested person relaxed or stressful?), and finally use face verification to find similarity between the arrested person's face and the wanted person.

Running the server, you can navigate to the homepage on **http://hexaface.ddns.net/** address. The top bar shows the links to different apps. By clicking each link, the system navigates to that app. The following figure shows the main page of the application.



Welcome to HexaFace!

The AI-based web portal for face applications.

We certainly can analyse your facial images.

How this works:

Click on the application you are interested in.

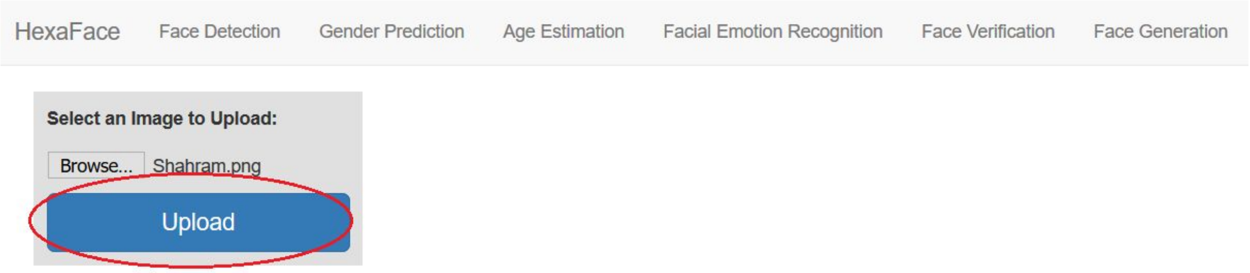
Upload your image.

Click on process

Copyright: Big Data Squad

Figure 28) Product main page

After loading each application you will notice a “Browse” button on the page except for face verification that has two “Browse” buttons. Using this button you have the chance to upload your images to the server.



Select an Image to Upload:

Shahram.png

Figure 29) How to upload an image

When you click the “Upload” button, a thumbnail of the uploaded image will be shown on the screen as well as an additional button to start processing the image.

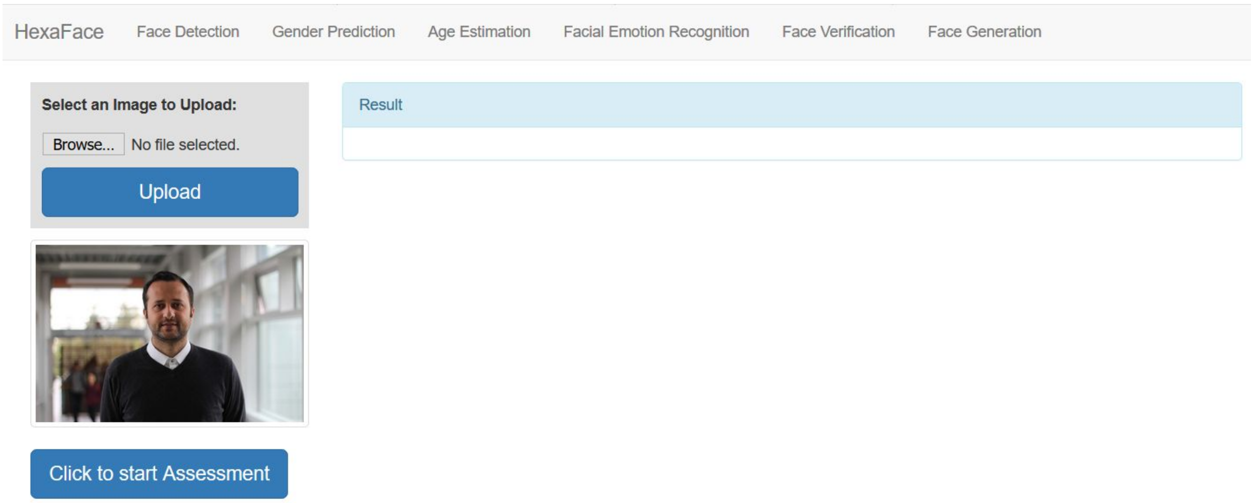
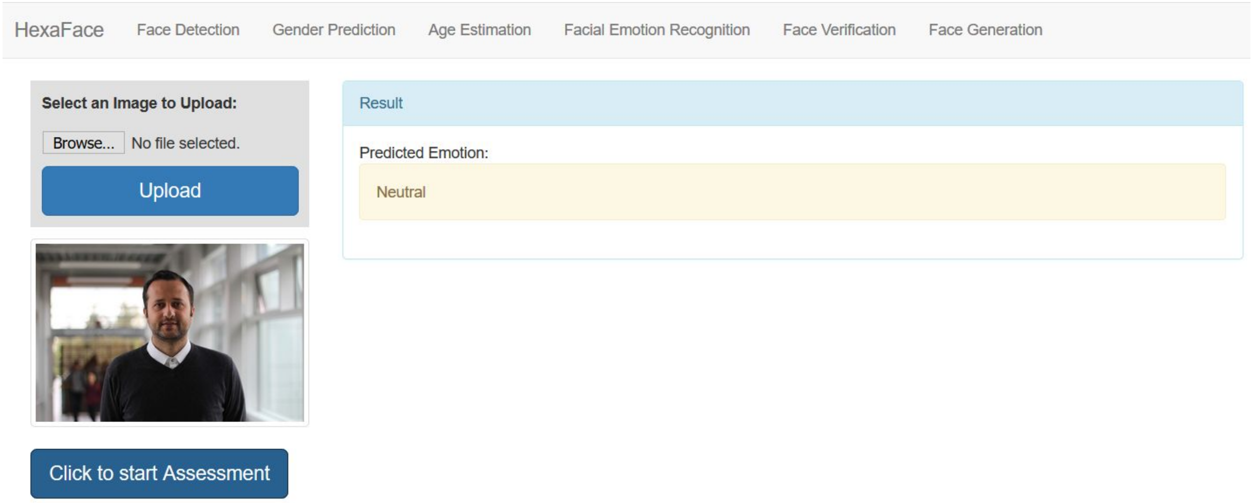


Figure 30) How to start an assessment

To start processing the image, click on the considered button in the left down corner of the page. After that, the final result of the process will be shown in the Result panel on the right of the page.



Predicted Emotion:

Neutral

Figure 31) Results of image processing

The way that every other app in the project works is the same as the described process above and there is no necessity to describe all apps here in this part.

7. Lessons learnt

We learned a lot about teamwork. Every member of our team is responsible for one facial app and fulfills the full process of data gathering, preprocessing, model training, tuning, testing and evaluation. Since it's a Computer Vision related project, we studied a lot of papers about VC and VC machine learning knowledge. We also used trained models weights to do transfer learning. We learned how to design a website with Django and set it up to launch our web application integrated with model files. Finally, we ran our instance on google cloud. The most challenging part of our project is that Computational resources and time are a huge bottleneck with model training and difficulties in finding high-quality datasets.

8. Summary and outlook

In this project, we developed a full-stack deep learning web portal for several facial applications, namely, face detection and recognition, gender prediction and age estimation, facial emotion recognition as well as facial synthesis **http://hexaface.ddns.net/**. Face detection and recognition are significantly important in applications like user identification and authentication, developing secure access systems for ATMs, phones, criminal identification, helping blind. Facial synthesis can be very helpful to generate anime characters, or for face anonymization, or even in the art area. Gender prediction and age estimation can be directly used for smart advertising, developing access control systems, or visual surveillance. Facial emotion recognition, on the other hand, can be employed in automated healthcare systems, monitoring babies or seniors, as well as in safety driving systems. These are only a few possible applications that our models can be applied to.

For full-stack development of deep learning models in face applications, we first trained the models on the appropriate datasets and evaluate them separately as reported here. Then we developed a web application in the Django framework that on the client-side uses HTML, CSS, and JavaScript, and on the server-side, it employs python and TensorFlow to deploy the deep learning models.

As for methodology in classification tasks like emotion recognition, we trained VGG-like Convolutional Neural Networks from scratch for extracting features from facial images and then classifications. As well, we employed transfer learning with VGG16 for gender prediction and age estimation. For facial synthesis and face recognition, we employed generative adversarial networks and one-shot learning.

For gender and age prediction, we used the IMDb-wiki dataset, and transfer learning with VGG16 to extract the features, then we trained a classifier on top of that for gender and age predictions. We got an accuracy above 90% for gender prediction and a +/-6 year confidence for age estimation. For facial emotion recognition, we trained a VGG-like CNN architecture from scratch on the FER-2013 dataset but we did not obtain any accuracies better than around 60%, We combined this dataset with FERG-DB and trained the model again, this time we obtained an accuracy of around 90%. For face detection, we trained an MTCNN architecture on the WIDER

FACE dataset. This method first classifies the face in the image before placing a bounding box and a confidence score around it. An accuracy of 95% was obtained for this model. We used a Siamese Neural Network using a pre-trained DeepFace model for face verification. After Retrieving facial features from the model, we compare their distance against a threshold to verify whether they are the same faces or not. We achieved an accuracy of 98% by testing the model on the AT&T Database of Faces. For face synthesis, we trained on the CelebAHQ dataset of faces and these are some of the generated samples. Imbalanced data affected our GAN. It made it generate more faces of females. For every output cycle, more female faces were generated.

For our future work, we intend to further improve our model performances. Moreover, we can integrate a combination of models for multi-task output for specific applications. Incorporating usage of GPUs will increase the processing speed of the app, especially for facial synthesis and face verification tasks.

9. References

[1] Yang, Yi, and Shawn Newsam. "Bag-of-visual-words and spatial extensions for land-use classification." Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems. 2010

[2] LeCun, Yann. “RI Seminar: Yann LeCun : The Next Frontier in AI: Unsupervised Learning.” Youtube, 18 Nov. 2016, www.youtube.com/watch?v=IbjF5VjniVE.

[3] Brownlee, Jason. “How to Implement GAN Hacks in Keras to Train Stable Models.” Machine Learning Mastery, 12 July 2019, machinelearningmastery.com/how-to-code-generative-adversarial-network-hacks/.

[4] Radford, Metz, & Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ArXiv, 19 Nov. 2015, arxiv.org/abs/1511.06434.

[5] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[6] Karazeev, Anton. “Generative Adversarial Networks (GANs): Engine and Applications.” Medium, 17 Aug. 2017, blog.statsbot.co/generative-adversarial-networks-gans-engine-and-applications-f96291965b47.

[7] Goodfellow, Ian J.. "NIPS 2016 Tutorial: Generative Adversarial Networks." ArXiv abs/1701.00160 (2016): n. pag.

[8] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, Yu Qiao , " Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks," IEEE Signal Processing Letter

[9] Tensorflow. "Deep Convolutional Generative Adversarial Network." Tensorflow Core, 2020, www.tensorflow.org/tutorials/generative/dcgan.

[10] https://github.com/AITTSMD/MTCNN-Tensorflow

[11] Li, Jessica. "CelebFaces Attributes (CelebA) Dataset." Kaggle, 1 June 2018, www.kaggle.com/jessicali9530/celeba-dataset.

[12] https://www.eff.org/pages/face-recognition

[13] https://us.norton.com/internetsecurity-iot-how-facial-recognition-software-works.html

[14] https://www.idiap.ch/~marcel/labs/faceverif.php

[15] http://www.robots.ox.ac.uk/~vgg/publications/2015/Parkhi15/poster.pdf

[16] https://sefiks.com/2018/08/06/deep-face-recognition-with-keras/
[17] https://hackernoon.com/one-shot-learning-with-siamese-networks-in-pytorch-8ddaab10340e
[18] https://medium.com/machine-learning-bites/deeplearning-series-face-recognition-383b1558a99e
[19] https://towardsdatascience.com/one-shot-learning-face-recognition-using-siamese-neural-network-a13dcf739e
[20] https://www.kaggle.com/kasikrit/att-database-of-faces

[21] https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/

[22] http://shuoyang1213.me/WIDERFACE/

[23] http://mmlab.ie.cuhk.edu.hk/archive/CNN_FacePoint.htm

[24] https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data

[25] https://grail.cs.washington.edu/projects/deepexpr/ferg-db.html