Abstract:

The task of face recognition has been actively researched in recent years. This paper provides an up-to-date review of major human face recognition research. We first present an overview of face recognition and its applications. Then, a literature review of the most recent face recognition techniques is presented. Description and limitations of face databases which are used to test the performance of these face recognition algorithms are given. Human emotions are mental states of feelings that come off spontaneously rather than through conscious effort and are accompanied by physiological changes in facial muscles which imply expressions on the face. Non-verbal communication methods such as facial expressions, eye movement, and gestures are used in many applications of human-computer interaction, which among them facial emotion is widely used because it conveys the emotional states and feelings of persons In the machine learning algorithm some important extracted features used for modelling the face, so, it will not achieve a high accuracy rate for recognition of emotion because the features are hand-engineered and depend on prior knowledge. Convolutional neural networks (CNN) have developed in this work for recognition facial emotion expression. Facial expressions play a vital role in nonverbal communication which appears due to internal feelings of a person that reflects on the faces. This paper detected emotion from those features from the positioning of the mouth and eyes. This paper will be proposed as an effective way to detect anger, contempt, disgust, fear, happiness, sadness, and surprise. These are the seven emotions from the frontal facial image of human Beings.

Contents :

**1.Introduction :**

1.1 Origin.

1.2 languages Used.

1.2.1 Python Packages.

1.3 Software and Hardware Used.

1.3.1 Recommended Hardware.

**2.** **REVIEW OF LITERATURE:**

**3. Implementation :**

3.1 Deep Learning .

3.1.1 Predictive Model.

3,1.2,CNN Model (Convolution Neural Network Model).

3.1.3.VGG Model.

3.2. Training and Validation of datasets.

3.3 Creation of the CNN(Convolution Network Model).

3.3.1 Implementing CNN model blocks 1 ,2,3,4,5,6,7.

3.3.3 Printing The summary of the model.

3.4 adding model checkpoint.

3.5 adding early stop.

3.6 adding reduce learning rate.

3.7.Compilation and generation of epochs.

**4 . Results & Conclusion :**

Introduction:

Origin:

Face recognition is an important research problem spanning numerous fields and disciplines. This because face recognition, in additional to having numerous practical applications such as bankcard identification, access control, Mug shots searching, security monitoring, and surveillance system, is a fundamental human behaviour that is essential for effective communications and interactions among people. Face recognition starts with the detection of face patterns in sometimes cluttered scenes, proceeds by normalizing the face images to account for geometrical and illumination changes, possibly using information about the location and appearance of facial landmarks, identifies the faces using appropriate classification algorithms, and post processes the results using model-based schemes and logistic feedback.

Face expressions play a key role in understanding and detecting emotion. Even the term "interface" suggests how important face plays in communication between two entities. For facial emotion recognition, the traditional approaches usually consider a face image that is distinguished from an information picture, and facial segments or milestones are recognized from the face districts. After that different spatial and worldly highlights are separated from these facial segments. At last dependent on the separated highlights a classifier, for example, Keras library, random forest, is trained to produce recognitions results. This work is an applied, deep learning model. Deep learning is a well-set model in the pattern recognition domain. It uses a Convolutional Neural Network (CNN) algorithm. CNN is a specific sort of artificial neural network that uses a machine-learning unit.

**Languages Used :** Python 3.7

**Python Packages Used :** TensorFlow & Keras.

**Software and Hardware Used :**

1. Visual StudioCode.

2. **Processor:** Intel(R) Core(TM) i5-9400F CPU .

3. **System Type** : x64-based PC, 16 gb RAM.

4.**GPU used :** NVDIA 1650 .

**Recommended Hardware :**

Intel I5 Processor , x64 Bit system , 8gb ram , any GPU .

REVIEW OF LITERATURE

This section gives an overview on the major human face recognition techniques that apply mostly to frontal faces, advantages and disadvantages of each method are also given.

Eigenfaces

The task of facial recognition is discriminating input signals (image data) into several classes (persons). The input signals are highly noisy (e.g. the noise is caused by differing lighting conditions, pose etc.), yet the input images are not completely random and in spite of their differences there are patterns which occur in any input signal. Such patterns, which can be observed in all signals could be - in the domain of facial recognition - the presence of some objects (eyes, nose, mouth) in any face as well as relative distances between these objects. These characteristic features are called *eigenfaces*in the facial recognition domain (or *principal components*generally). They can be extracted out of original image data by means of a mathematical tool called *Principal Component Analysis* (PCA).  
By means of PCA one can transform each original image of the training set into a corresponding eigenface. An important feature of PCA is that one can reconstruct any original image from the training set by combining the eigenfaces. Remember that eigenfaces are nothing less than characteristic features of the faces. Therefore one could say that the original face image can be reconstructed from eigenfaces if one adds up all the eigenfaces (features) in the right proportion. Each eigenface represents only certain features of the face, which may or may not be present in the original image. If the feature is present in the original image to a higher degree, the share of the corresponding eigenface in the ”sum” of the eigenfaces should be greater. If, contrary, the particular feature is not (or almost not) present in the original image, then the corresponding eigenface should contribute a smaller (or not at all) part to the sum of eigenfaces. So, in order to reconstruct the original image from the eigenfaces, one has to build a kind of weighted sum of all eigenfaces. That is, the reconstructed original image is equal to a sum of all eigenfaces, with each eigenface having a certain weight. This weight specifies, to what degree the specific feature (eigenface) is present in the original image.

Neural Networks

Graphs

Once we have the means to generate and compare image graphs, recognition of faces in identical pose is relatively straight-forward. Matters become more complicated when trying to recognize faces across different poses.

Recognition in Identical Pose

Assume we are given 1000 facial images of identical pose, e.g. all looking straight into the camera, and correctly labeled with the name of the person they show. This constitutes the model gallery. For face recognition we would proceed as follows:

* **Step 1: Building a face graph.** The first step to bootstrap the system is to define the graph structure for the given pose. Thus, we take the first image and manually define node locations on the face that are easy to localize, such as the corners of the eyes or mouth, the center of the eyes, the tip of the noise, some points on the outline etc. We also define edges between the nodes. This constitutes our first face graph.
* **Step 2: Building a face bunch graph.** The single face graph defined above can be viewed as a bunch graph with just one instance in it. It can be matched onto the second face image, but if the first two face images are not very similar, the match is of poor quality. For instance, the tip-of-the-nose node might by placed at the cheek, so we need to move the node onto the tip of the nose by hand. After some such manual correction the graph is acceptable and constitutes the second instance in the bunch graph. The bunch graph with two instances is then matched onto the third image, and after some manual correction we have a third instance for the bunch graph. By repeating this process, the bunch graph grows, and as it grows the match onto new images gets more and more reliable. If we are satisfied with the quality of the matches and only little manual correction is needed, we are done with building the bunch graph. Let say this happens after having processed the first 100 images.
* **Step 3: Building the model gallery of graphs.** Since we now have a bunch graph that provides sufficient quality in finding the node locations in a new face, we can process the remaining 900 images fully automatically. To avoid the distinction between manually corrected and fully automatically generated model graphs, one can create the bunch graph on an extra set of 100 images distinct from the 1000 model images. Then all 1000 model graphs can be created automatically. We are now in the position to perform face recognition on a new probe image.
* **Step 4: Building the probe graph.** Assume we are given a new image and shall find the depicted person in the gallery. First we need to create a graph for the probe image. This process works exactly as for the model images, just that we use the probe image.
* **Step 5: Comparison with all model graphs.** The image graph is compared with all model graphs, resulting in 1000 similarity values. These form the basis of the recognition decision. Notice that this does not require EBGM anymore, only the graphs are compared according to the similarity function ([3](http://www.scholarpedia.org/article/Elastic_Bunch_Graph_Matching#Eq-3)).
* **Step 6: Recognition.** For recognition it is obvious that the model graph with the highest similarity with the image graph is the candidate to be recognized. However, if the best similarity value is relatively low, the system might decide that the person on the probe image is not in the model gallery at all; and if there are more than one very high similarity values, the system might decide that the person is most likely in the model gallery but that there are several possible candidates. Only if the highest similarity is high and the next one is low can the system recognize the face on the probe image with high reliability.

Template matching

Here, we present an original template based on edge direction. It has been noticed that the contour of a human head can be approximated by an ellipse. This accords with our visual perception and has been verified by numerous experiments. The existing methods have not sufficiently used the global information of face images in which edge direction is a crucial part, so we present a deformable template based on the edge information to match the face contour. The face contour is of course not a perfect ellipse. To achieve good performance, the template must tolerate some deviations. We follow a few steps for achieving our goal. First of all we take an input image which contains a single face in it. On the given input image we apply the sobel operator for detecting the edge in the image. Then we apply a threshold value to the image to binarize the image. The edge which we get after applying the sobel operator is thick. After thinning We try to eliminate the noise present in the image. We apply edge linking to the nearest points.

**Support Vector Machine**

**Support Vector Machine** (**SVM**) is a supervised machine learning model used for two-group classification problems. After giving an **SVM** model set of labeled training data for each category, they’re able to categorize new test data. SVM classifies data based on the plane that maximizes the margin. The SVM decision boundary is straight. SVM is a really good algorithm for image classification. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true for image segmentation systems, including those using a modified version SVM that uses the privileged approach. Faces are high dimensionality data consisting of a number of pixels. Data in high dimensionality is difficult to process and cannot be visualized using simple techniques like scatterplots for 2-dimensional data.What we will do is to use PCA to reduce the high dimensionality of data and then feed it into the SVM classifier to classify the pictures.

**Implementation :**

**Deep learning :**

**Predictive Model :** Predictive modelling is the process of using known results to create, process, and validate a model that can be used to forecast future outcomes. It is a tool used in [predictive analytics](https://www.investopedia.com/terms/p/predictive-analytics.asp), a data mining technique that attempts to answer the question "what might possibly happen in the future?"

### **KEY TAKEAWAYS**

* Predictive modelling is the process of using known results to create, process, and validate a model that can be used to make future predictions.
* Two of the most widely used predictive modelling techniques are regression and neural networks.
* Companies can use predictive modelling to forecast events, customer behaviour, as well as financial, economic, and market risks.

**CNN Model (Convolution Neural Network Model) :**

**Convolutional neural network** (**CNN**, or **ConvNet**) is a class of [deep neural networks](https://en.wikipedia.org/wiki/Deep_neural_network), most commonly applied to analysing visual imagery.

CNNs are [regularized](https://en.wikipedia.org/wiki/Regularization_(mathematics)) versions of [multilayer perceptron’s](https://en.wikipedia.org/wiki/Multilayer_perceptron). Multilayer perceptron usually mean fully connected networks, that is, each neuron in one [layer](https://en.wikipedia.org/wiki/Layer_(deep_learning)) is connected to all neurons in the next [layer](https://en.wikipedia.org/wiki/Layer_(deep_learning)). The "fully-connectedness" of these networks makes them prone to [overfitting](https://en.wikipedia.org/wiki/Overfitting) data. Typical ways of regularization include varying the weights as the loss function gets minimized while randomly trimming connectivity. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in the filters. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

Convolutional neural networks are comprised of two very simple elements, namely [convolutional layers](https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/) and [pooling layers](https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/).

Although simple, there are near-infinite ways to arrange these layers for a given computer vision problem.

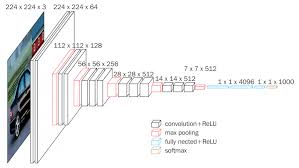
Fortunately, there are both common patterns for configuring these layers and architectural innovations that you can use in order to develop very deep convolutional neural networks. Studying these architectural design decisions developed for state-of-the-art image classification tasks can provide both a rationale and intuition for how to use these designs when designing your own deep convolutional neural network models.

## **VGG**

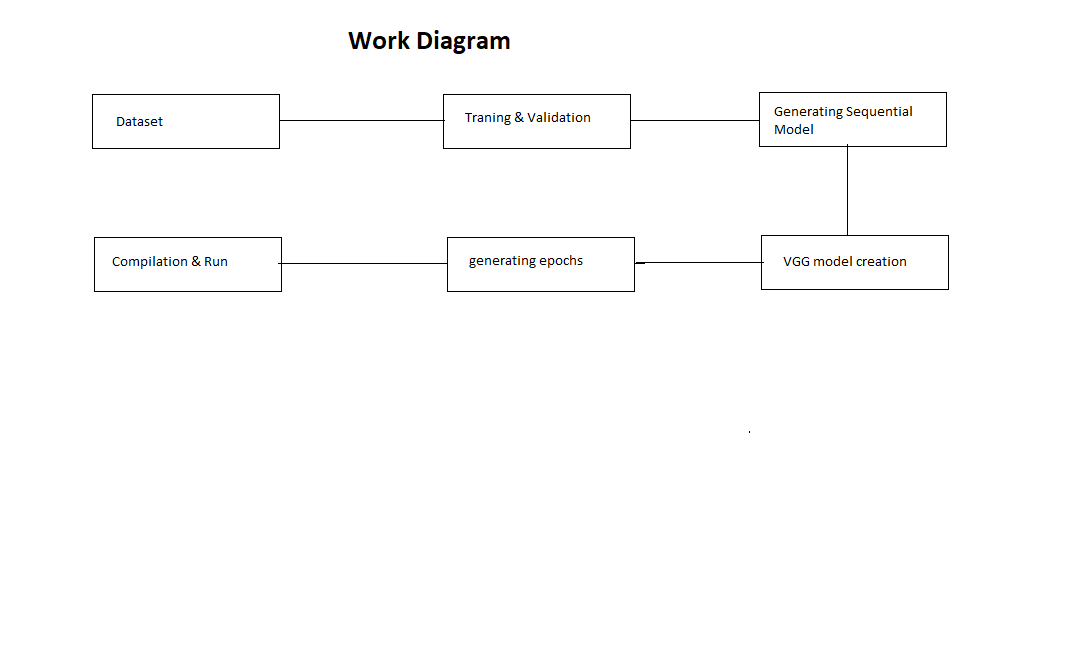
The development of deep convolutional neural networks, An important work that sought to standardize architecture design for deep convolutional networks and developed much deeper and better performing models

Max pooling layers are used after most, but not all, convolutional layers, learning from the example in Alex Net, yet all pooling is performed with the size 2×2 and the same stride, that too has become a de facto standard. Specifically, the VGG networks use examples of two, three, and even four convolutional layers stacked together before a max pooling layer is used. The rationale was that stacked convolutional layers with smaller filters approximate the effect of one convolutional layer with a larger sized filter, e.g. three stacked convolutional layers with 3×3 filters approximates one convolutional layer with a 7×7 filter.

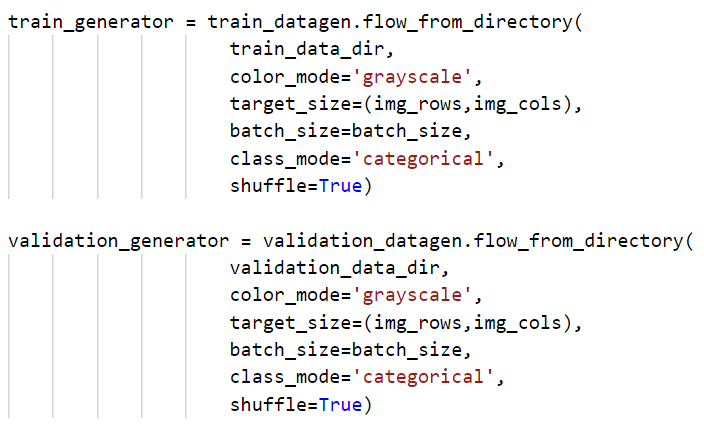
**VGG model**



**Work Diagram :**



**Training and Validation of datasets :**



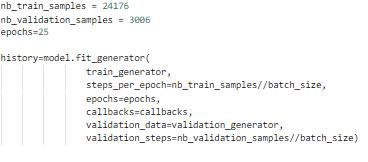
**Implementing CNN model blocks 1 ,2,3,4,5,6,7. :**



**Adding model checkpoint, early stop ,learning rate :**



**Compilation and generation of epochs**.



**Result and Analysis:**

From The upcoming results we can conclude that the accuracy for this program was

We found the nuber of Total Programs : 1328167

Trainable Programs : 1325991

Non Trainable programs : 2176

