



İŞIK UNIVERSITY
COMPUTER
SCIENCE AND
ENGINEERING



FACE DETECTION SYSTEM

Bachelor's Thesis

Ümmügülsüm Çamoğlu

217SE2323

Supervised by

Emine Ekin

January 2023

ABSTRACT

In this project, it is aimed at designing a custom face detection machine learning model. An attendance system design is also aimed with the method called face recognition, which compares the detected face with the data that has been taught and in our database. The subject of face detection, which is used in many areas today, has created various difficulties in the detection of the face because it contains multiple parameters. Especially detecting a moving object during real time video can be included in these difficulties. Automatic face detection systems are needed in areas such as face recognition, facial expression recognition. It is roughly based on two different bases for the perception of the face on the picture, one is feature and the other is image based. As a feature, the shape or color of the face can be said. Perfect match with facial features is aimed by separating the image features. There is the Haar Cascade Classifier method, which is a common method to use. In this study, a machine learning model is divided into three stages in the training phase. The first is data collection and data processing. Opencv framework was used for data collection via webcam. Label me tool was used to annotate the faces in the data. Augmentation was done to increase the data obtained in the data and to increase the scale of the data to be trained. Albumentation library was used as an augmentation tool. In the creation of the model, it was created by adding new layers to the VGG16 pre-trained model from keras functional applications. Tensorflow transfer learning technique was used. The methods listed so far were made to perform the face detection part, and secondly, the LBPHFaceRecognizer technique of the open cv library was used for face recognition. In the face recognition phase of the project, the techniques currently used were tried and efforts were made to label a trained model as unknown when it encounters a face that has not been introduced to the model. The deep learning model prepared and the face recognition attendance system created with opencv can be used with the website created using the django library and with SQLite database system.

TABLE OF CONTENTS

1. INTRODUCTION	1
1.1. Problem Definition	1
1.2. Definitions, Acronyms, and Abbreviations	1
1.3. Organization of the Thesis	1
2. LITERATURE REVIEW	2
3. PROPOSED SYSTEM	3
3.1. Introduction	3
3.1.1 Interface Layer	3
3.1.2 Application Layer	3
3.1.3 Storage Layer	3
4. IMPLEMENTATION, TESTS, EXPERIMENTS	4
4.1. Implementation	4
4.1.1. Creating Dataset	4
4.1.2. Library and Tool Used	4
4.1.3. Building The Face Detection Model	4
4.1.4. Transfer Learning	4
4.1.5. VGG16	4
4.1.6. LBPH Face Recognizer	4
4.2. Experiments	4
4.2.1 Changing Learning Rate	4
4.2.2 Changing Dense Layer Unit	4
4.2.3 Changing Activation Function	4
5. CONCLUSION AND FUTURE WORK	5
6. REFERENCES	6

LIST OF FIGURES

Figure 2.1 Edges in a Haar cascade that detects various features in an image	2
Figure 2.2 Detecting some features from faces	2
Figure 3.1 Diagram of Add Person Flow	2
Figure 3.2 Diagram of Detect Person Flow	3
Figure 4.1 Annotation Sample	4
Figure 4.2 Unaugmented Train dataset sample	4
Figure 4.3 Augmented train dataset sample	4
Figure 4.4 Albumentation Code	
Figure 4.5 Structured Layers of Model	4
Figure 4.6 Relation Between the Layers	4
Figure 4.7 Neurons in a Layer	4
Figure 4.8 Sigmoid Activation Function	4
Figure 4.9 A Representation of Transfer Learning	4
Figure 4.10 The Structured Layers of VGG16 Model	4
Figure 4.11 Normal and Bright Light LBPH Algorithm	4
Figure 4.12 Local Binary Pattern Representation	4
Figure 4.13 Euclidean Distance	4
Figure 4.14 Learning Rate Effects on Loss	4

LIST OF TABLES

Table 4.1 Results of Learning Rate Changes	4
Table 4.2 Results of Unit Number in Dense Layer Changes	4
Table 4.3 Behavior comparison of softmax and sigmoid function in the last layer	4
Table 4.4 Behavior comparison of sigmoid and relu function in the last layer	4
Table 4.5 Behavior comparison of tanh and relu function in the hidden layer	4

1. INTRODUCTION

1.1. Problem Definition

At the beginning of the 1970s, there were early attempts at face recognition. These studies were limited to plain backgrounds and simple passport stances on the front, and it was necessary to protect these conditions. Any change in these conditions did not lead to a rework, it is called fine tuning. Until the 1990s, the coding of facial recognition systems and the processing of videos were implemented in a serial manner. During the development process, motion, color and general information were used, and faces at different distances to the camera were detected using statistics and neural networks. When face detection studies are generalized under two different approaches, the first approach is an open approach that directly takes advantage of the features of the face. These features can be skin color or the angles of the face. Techniques are made by manipulating these features. Since it is based on the analysis of the features, the second approach is the image-based approach besides the feature-based approach. This approach is based on classification rather than analysis of any features.

The main purpose of this project is to detect human faces from pictures and to establish an attendance system by recognizing these detected faces. With the use of detected faces by detecting a face, it is aimed to create an attendance system by checking the learned faces. First of all, detecting faces is the first step. Meanwhile, we have an image as input. We need to find out if there is a face in this picture, and if there is, where it is. The dataset we are creating is of great importance in order to train a model that determines the location of a face on the picture. Using the Python language's OpenCv library, a dataset was created with faces drawn from different angles and sizes. The Label Me tool was used to annotate the data we collected and add labels. Since we will detect faces using a bounding box detector, a bunch of images were collected using open cv and a label me tool was used to identify these bounding boxes. Another step that needs to be done to the data set is augmentation. Albumentation is an image augmentation tool. Because manually creating enough datasets to train a model with label me would be cumbersome.

With the completion of the data set, the building of the model is based on the transfer learning approach. If transfer learning is trained for a specific purpose, for example, to detect whether a certain element is present in an image, it is the process of using this model to detect another element. This can be defined as transferring the weights learned for the first task to the

next task. VGG16 is an object detection and classification algorithm. This pre-trained model, which is popular in image classification, was used in this study because it is suitable for use with transfer learning. Faces detected with the trained model, faces detected using the pre-trained LBPH face recognizer are intended to be used.

1.2. Definitions, Acronyms, and Abbreviations

1. Face Detection : Finding the face on the image.
2. Face Recognition : Match the person with given image.
3. Machine Learning : A sub-discipline of Artificial Intelligence that can learn and make better performance as it consumes data.
4. Haar Cascade Classifier : An algorithm for detecting an object in an image.
5. Augmentation : Making wider dataset
6. CNN : Convolutional Neural Network

1.3. Organization of the Thesis

In the following sections of this document, similar structures used in the "Literature Review" section are mentioned first. In the "Proposed System" section below, the approaches to the problem we want to solve are discussed in detail. In another chapter, the application phases of this approach, the tests performed, the tools and experiments used are discussed. Finally, the general evaluation and results of the study are mentioned.

2. LITERATURE REVIEW

Face detection studies are classified as feature-based approach and image-based approach. At various system levels, features of the face including skin tone and face shape are taken advantage of. Additional categories for feature-based approaches include low-level analysis, feature analysis, and active shape models. The first step in low-level analysis is the segmentation of visual features utilizing pixel characteristics like color, edges, and gray-scale. In the Feature Analysis method, a general face concept is organized based on the face geometry. It is aimed to remove the ambiguities between facial features. The positions of the face and facial features are extracted. Feature extractions such as active shape models, pupil and lip tracking have been developed. The technique that is classified as a face group through training without features or analysis is image based. Window scanning is one of them, focused on searching for possible locations of the face. Statistical, neural network, and linear subspace methods are other categories for image-based representations.

One of the known and most famous algorithms in the field of face detection is Viola-Jones Face Detection technique, also known as Haar cascades. Haar cascades predates deep learning studies and is a widely used technique for face detection. You then extract the features from the images. Features extracted from positive images containing faces are separated from these images. Some of the features that are taken from photographs of faces are displayed in Figure 2.1 and 2.2.

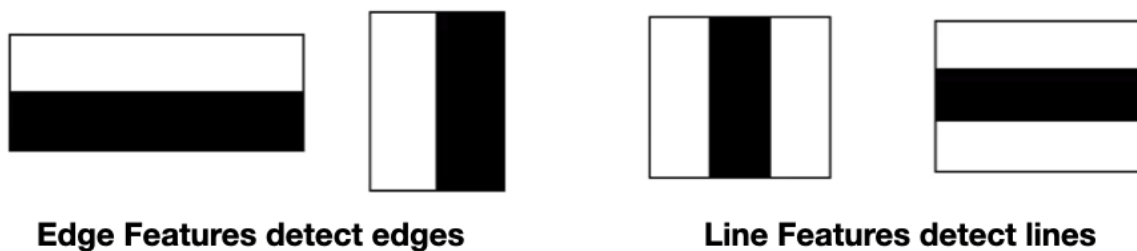


Figure 2.1 Edges in a Haar cascade that detects various features in an image



Figure 2.2 Detecting some features from faces

3. PROPOSED SYSTEM

3.1. Introduction

Face Detection System is a website where you can save people by giving names and ids to the system, by identifying faces. In this way, it aims to make face recognition and recognition quickly. The users can add themselves to the system, detect and recognize people, and view the detections made so far with a day, week and month filter. 3 Tier architecture is used in the system's construction. The user interface is the first layer, while the area where the components are found and specific tasks are carried out is the second layer. The storage layer, which communicates with the second layer, is the last layer.

3.1.1 Interface Layer

This is the stage where users communicate with the system. Here, the user sends requests to the system with forms and actions and receives feedback from the system.

3.1.2 Application Layer

This stage is to work in line with the wishes of the user by performing certain operations and algorithms of the request from the users. Meanwhile, it is in communication with the storage layer as the third layer. When we consider the interfaces of the system, we can examine this stage in three parts. The first of these main stages is to add a person to the system. Here, firstly, the id and name that the user wants to assign to the person with the form are taken. Here, if the entered id already exists in the system, the person is not allowed to continue. After the id and name entered, a window opens to take pictures of the person. This window takes pictures of the person and they disappear. During the photo taking of the person, the face should be in

front of the camera and should be turned slightly left and right. When the video camera is turned on, firstly the face is detected from the video and the face is photographed directly. If no face is detected, the counter that took the photos will stop and the user can see if the photo was taken. We train opencv LBPH Face Recognizer with the data we get from the user. As a result, this person's photos are trained by this model, histograms are created and matched with the id we received from the user. Figure 3.1 contains the diagram of this flow.

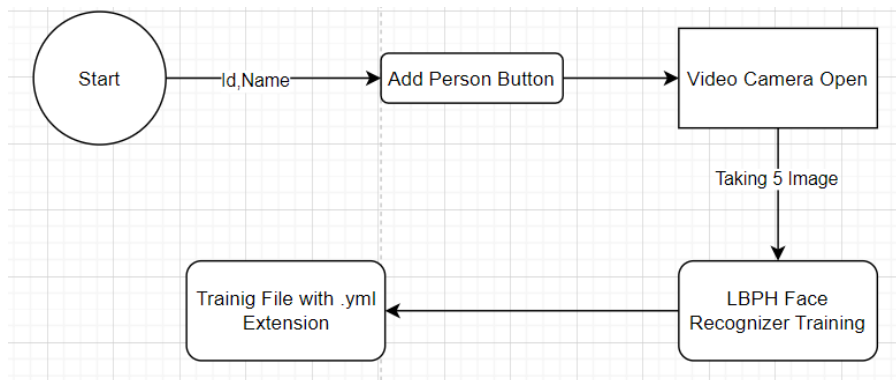


Figure 3.1 Diagram of Add Person Flow

The second main feature, the people detection page, contains the logic from the previous step as main. Here, a window opens with the take attendance command and the video camera is activated. The person is detected and a frame is drawn around the face, and the recognition result is displayed on the screen. Detection is taken with the Scape button and the window is closed. The window closes with the Escape key. In the face detection part, a model called vgg16 is used, which we obtained by rebuilding a model using transfer learning. In addition to this model, our recognition model with .yaml extension, which we obtained by training people at the stage of adding them to the system, determines who the person is with the prediction made. In this flow, the reporting steps are also built as a part of the flow as a result of the detection. The reporting phase is recorded in the database on a daily, weekly and monthly basis.

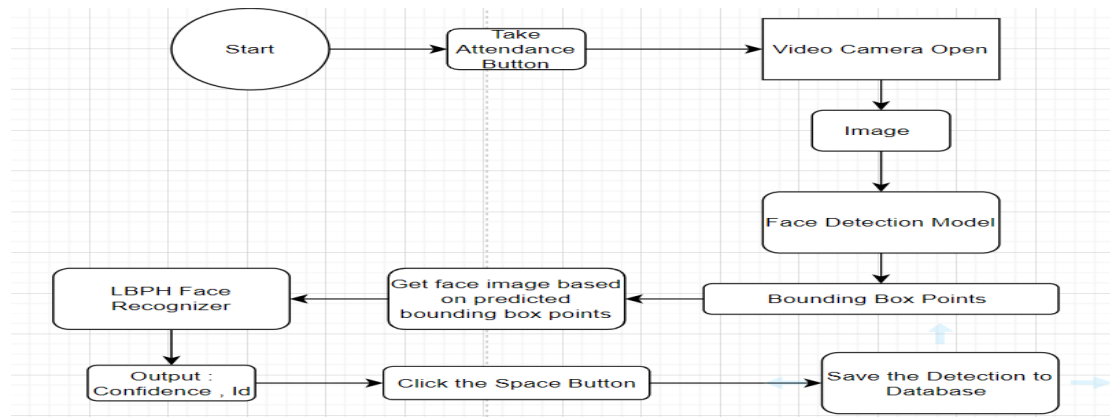


Figure 3.2 Diagram of Detect Person Flow

3.1.3 Storage Layer

Storage layer is the layer where the received detections are stored and daily, weekly and monthly reports are stored.

4. IMPLEMENTATION, TESTS, EXPERIMENTS

4.1. Implementation

The project aims to design an attendance system. For this reason, it has been implemented by dividing it into three important main parts such as adding people, detecting people and keeping records. These main jobs are basically based on detecting a face. In order to detect the face, we need to build a deep learning model that we will train to detect the faces. Before starting the built operations, there is the dataset creation phase, which is a large and important part of the implementation. The creation of the dataset is covered in detail under the “*Creating dataset*” title. Second, we used the “*vgg16*” model, a pre-trained object detection model for object detection, in a model building phase. We retrained this model with the dataset we created by adding new layers in addition to the previously trained layers. It is intended to be used to train a pre-trained model in face detection with a method called “*Transfer Learning*”. The purpose of the new layers added under the “*Building The Face Detection Model*” heading and the compile and train stages of the model are detailed. After this stage of implementation, with the existence of a model that can detect faces in real time, the face recognition stage necessary for the use of this model was started. Adding the people you want to recognize under the “*LBPH Face Recognizer*” title and then using it are explained in detail.

4.1.1. Creating Dataset

In order to create the dataset, 180 photos of 90 photos were taken per person with different facial expressions, taken from different angles. These photos were taken using the Opencv library in python. A tool called Label Me was used to label the captured images and add annotations. With this tool, we can frame the objects we want to detect and annotate them. When we apply this process to the pictures, we have json files that describe each picture as shown in Figure 4.1, together with the pictures.

```
{
  "version": "5.1.1",
  "flags": {},
  "shapes": [
    {
      "label": "face",
      "points": [
        [
          28.123824451410655,
          30.945141065830715
        ],
        [
          159.47178683385582,
          157.27742946708463
        ]
      ],
      "group_id": null,
      "shape_type": "rectangle",
      "flags": {}
    }
  ],
  "imagePath": "example4\\dataset\\User.2.49.jpg",
  "imageData": "/9j/4AAQSkZJRgABAQAAQABAAQ/2wBDAAg",
  "imageHeight": 297,
  "imageWidth": 197
}
```

Figure 4.1 Annotation Sample

We need to divide this data we created into 3 parts at certain rates. These parts are called train, validation and test. We may want to see how the model will behave during the implementation phase. The test set is created for this phase. The set, which is created with a random part of the original dataset that will not be used for train, allows us to see how the model behaves on data that it does not see. In addition, we need to create another data set called the validation set, which allows us to try different configurations related to my model. These configurations can be counted as optimizations and loss functions. To compare the performance of these experiments, another random set is generated from the original data set, not for the test or train, but to validate the model by different configurations. At this point, both test and validation sets say that they contain data independent of the set we train the model, however, the model is

fitted to this validation set to have the best validation metrics. The combination that performs best in this set is chosen. This distribution depends on the length of the master dataset and in general this distribution can vary as 75/15/10, 70/15/15, and 70/20/10. In this study, 70% of the dataset and 15% of the dataset were determined as validation and test sets. The distribution of these data was randomly separated from the instant data set manually.

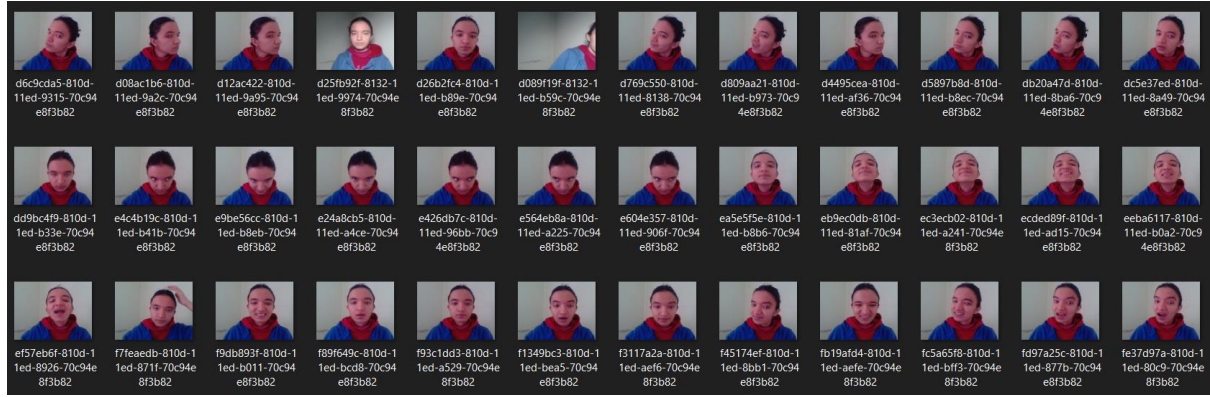


Figure 4.2 Unaugmented train dataset sample

The most important element to train a model in the field of deep learning is the collection of sufficient data. This problem has been solved by the data augmentation method, on the grounds that more data will perform better. This is a technique for generating more training samples from available data.

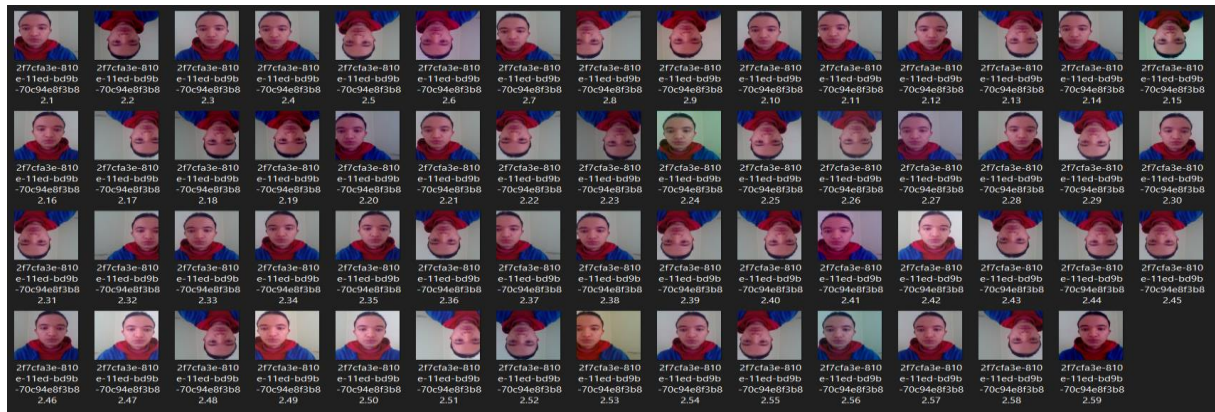


Figure 4.3 Augmented train dataset sample

4.1.2. Library and tools used

4.1.2.1. Open CV

This library has been developed for use in computer vision, machine learning, and image processing. In this project, real-time image acquisition was used to obtain real-time images during dataset creation from scratch. In addition, the face recognition part of the project was carried out using the opencv face namespace.

4.1.2.2. Label Me

LabelMe's purpose is to provide an online annotation tool for building image databases for computer vision research. LabelMe offers polygons, rectangles, circles, lines and points for image annotation. Videos can also be annotated using polygons. (Bandyopadhyay, 2023) In this project, the label me tool was used in the bounding box drawing in order to determine the 4 corner points for the face images created.

4.1.2.3. Albumentation

We mentioned that Augmentation is a technique used to expand a data set. This is made possible by the open source albumentation library. With the albumentation library, various transformations are applied to the original data to create new data from existing data. These transformations can be flipping, shearing, blurring or cropping. This method will not only increase the performance of our model, but also prevent overfitting.

```
augmentor = alb.Compose([alb.RandomCrop(width=450, height=450),
                        alb.HorizontalFlip(p=0.5),
                        alb.RandomBrightnessContrast(p=0.2),
                        alb.RandomGamma(p=0.2),
                        alb.RGBShift(p=0.2),
                        alb.VerticalFlip(p=0.5)],
                        bbox_params=alb.BboxParams(format='albumentations',
                                                    label_fields=['class_labels']))
```

Figure 4.4 Albumentation Code

Figure 4.4 contains the code for the transformations we want to use. The Compose method takes as arguments a list of transformations we want to apply to the images. Examples of transformations are shown in [Figure 4.3](#). The bounding boxes rotations we draw here are also important, and the bounding box rotation is made with the “BboxParams” parameter. The "albumentations" format divides the coordinates in pixels by image width and height to normalize [x_min, y_min, x_max, y_max] values.

4.1.3. Building The Face Detection Model

Considering the construction of Object detection, classifying the object to be detected is the first situation encountered. At the point of classification of an object, considering only the face class, it was considered as a class. In this case, it is aimed to classify faces. BinaryCrossentropy loss is used in this classification process. It bases each of the predicted probabilities as 0 or 1. Calculates the probability score based on the distance from the expected value. It represents how far or close it is to the true value. Another thing we need to guess at this stage is $[x1,y1],[x2,y2]$ to create a bounding box. As always, there are a number of methods you can use to calculate the differences between the predicted values and the actual values in regression situations. The MeanSquaredError class is used to calculate the mean squared error between the estimates and the actual values. After defining these losses, the keras functional api was used to build the model. The Vgg 16 model, which is a classification model for images, is taken as the base architecture. The final version of the model was created by adding new layers to this pre-trained model for the purpose.

Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
input_4 (InputLayer)	[(None, 120, 120, 3)]	0	[]
vgg16 (Functional)	(None, None, None, 512)	14714688	['input_4[0][0]']
global_max_pooling2d_2 (Global MaxPooling2D)	(None, 512)	0	['vgg16[0][0]']
global_max_pooling2d_3 (Global MaxPooling2D)	(None, 512)	0	['vgg16[0][0]']
dense_4 (Dense)	(None, 2048)	1050624	['global_max_pooling2d_2[0][0]']
dense_6 (Dense)	(None, 2048)	1050624	['global_max_pooling2d_3[0][0]']
dense_5 (Dense)	(None, 1)	2049	['dense_4[0][0]']
dense_7 (Dense)	(None, 4)	8196	['dense_6[0][0]']

Figure 4.5 Structured Layers of Model

Basically, any Neural Network used for image processing consists of Input layer, Convolutional Layer, Pooling Layer, Dense Layer. Convolution layers are the layers used to extract features like edges from the image. (Dumane, 2020) In Figure 1.8, the shape of the input we want to put into the model is determined as (120,120,3). When we send this input as an input to the pre-trained vgg16 multi-layer architecture, it can be considered as the Convolutional Layers part. GlobalMaxPooling2D() was used as Pooling Layers. The spatial average of the properties that the GlobalMaxPooling2D layer receives from the last convolution layer is output as an output. Fully connected layers tend to overfit and the GlobalMaxPooling layer can prevent overfitting as it is itself an editor. Pooling is used to reduce the size of the image. When creating a Neural Network, the first convolutional layer

requests the shape of the transmitted image as input. The image will switch to the output dense layer after the convolutional and pooling layers mentioned. Classification is done by sending the output from the last layer to one or more Dense layers to complete the model. The image is categorized using a dense layer based on the convolutional layer output. The Neural Network contains neurons that calculate the weighted average of each layer input, this weighted average is passed through a function called the activation function. The result of this activation function is considered the output of that neuron. The number of dense layers will vary depending on your requirement and the number of output classes.

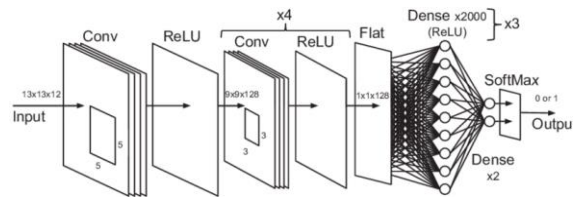


Figure 4.6 Relation Between the Layers

We said that every output that is criticized as planar with the pooling layer is passed through an activation function in the layer and classification is aimed as a result of convergence. Figure 4.6 represents an example of this.

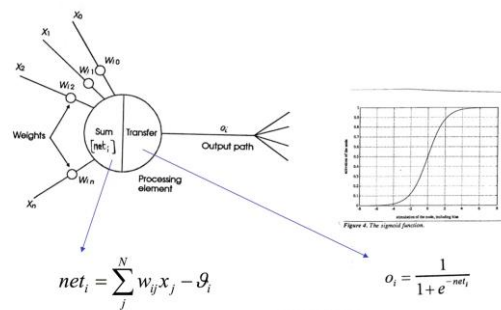


Figure 4.7 Neurons in a Layer

Figure 4.7 shows an example of the neurons in the layers. Output is applied by applying a process according to the selected activation function. Since the last dense layer is expected to have one output and four outputs, a dense layer is used that will give one and four outputs. Each of the units of the last dense layer represents a neuron, the first predicts the accuracy of the 'face' class, while the second four output sigmoid dense layer predicts the presence of each bounding box.

The purpose of an activation function is to add nonlinearity to the neural network. Each layer added to the neural networks, if it is linear, will behave the same no matter how many

layers are added. As a result, the model will be a linear regression model. (Baheti, 2023) Before mentioning which function to choose in a CNN model, it was mentioned that a neural network generally consists of three stages. These are Input Layer, Hidden Layer(s) and Output Layer. The input layer holds the input data, no calculation and no activation function. Inside the hidden layers, we need to use the activation function because complex patterns need to be made non-linear into a network to learn. For this reason, the issue of activation function, which significantly changes the performance of neural networks, is important. The relu function is an alternative to both the sigmoid and tanh functions. Convergence is faster than these two functions. Returns the output as it is if the input value is zero or greater, and equals zero if it is less than zero. Relu is not used in the output layer. (Pramoditha, 2022)

We need to define as many neuron units as there are how many classes we make the last dense layer classification. Softmax activation function is used when we have 2 or more than 2 classes. For this reason, sigmoid was preferred because softmax will not be used in this project. In the following stages, it is possible to examine the result obtained by trying softmax instead of sigmoid under the title of “Changing Activation Functions”. Sigmoid returns its input to a probability value between 0 and 1. Because it returns 0 to 0.5, it is considered the threshold value that determines what kind of two classes the input belongs to. Sigmoid is used in the output layer rather than the hidden layers in CNNs. The reason for this is that it can return output values that can make binary classification depending on the probability value of the input.

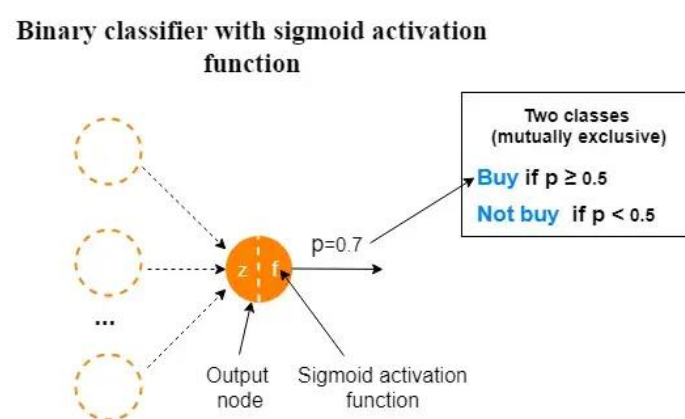


Figure 4.8 Sigmoid Activation Function

Sigmoid is not used in hidden layers because its outputs are not zero-centered. This complicates the optimization process. In a regression problem we use a linear identity function and it is

sigmoid. Binary classifier is also a good choice. Also softmax is used if there is more than one class. Relu is the default activation function for hidden layers. In the Experiment section, you can see a review of the outputs by trying the activation functions.

4.1.4. *Transfer Learning*

Transfer learning is based on the use of features learned in a problem in a new, similar problem. It is generally used when data is not available to train a model completely from scratch. The first step is to take the layers of the pre-trained model. In order not to lose these layers during retraining, mark them as training=False. On top of these layers are added new, purposeful, instructable layers. The old features will be used to make predictions on the new dataset. The model is trained by adding new layers.

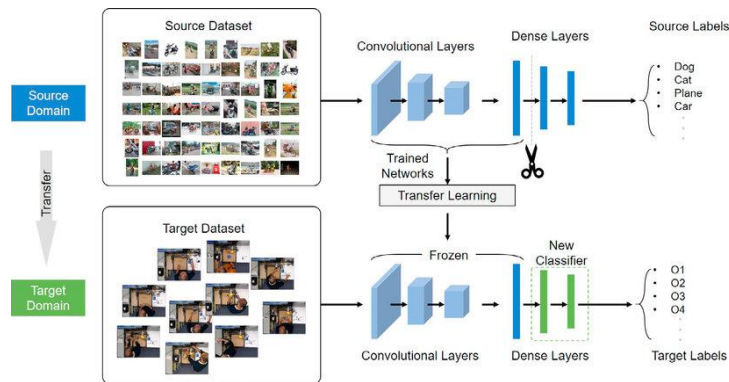


Figure 4.9 A Representation of Transfer Learning

4.1.5. *VGG16*

VGG is a deep convolutional neural network. VGG16 consists of 13 convolutional layers which have 5 max pooling layers, and the last 3 layers are fully connected layers. As a result, it is designed as 13 convolutional layers and 3 fully connected layers. The last 3 layers are fully connected layers, and the first two have 4096 neurons, and this number is 1000 in the last layer. The reason for this is determined by the number of classes of the Imagenet dataset. (Le, 2021) Specifically for this project, it was first adapted as 1 class and secondly as 4 classes.

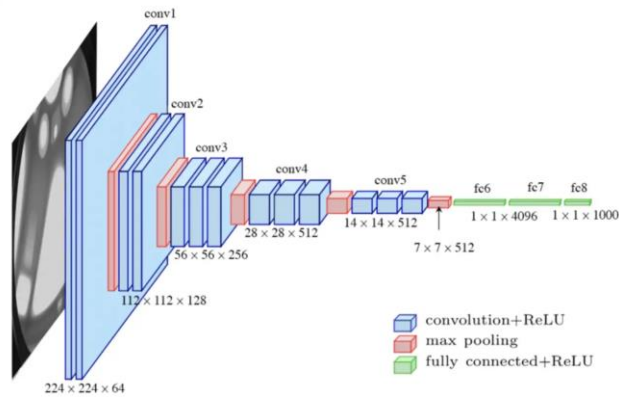


Figure 4.10 The Structured Layers of VGG16 Model

4.1.6. LBPH Face Recognizer

LBPH (Local Binary Pattern Histogram) is a face recognition algorithm. It has the ability to recognize a person's face from both the front and summer faces. In order to examine the basic logic of this algorithm, first of all, the basis of the images should be looked at. It is a matrix that represents part of the images. When this matrix is considered as (3,3) dimensional and 9 elements, values lower than the central element are marked as 0 equal to the central element and higher values are marked as 1. A binary number is obtained with the binary numbers around this central element. If this binary number corresponds to the decimal value, the pixels around the central element have this decimal value. The importance of this algorithm is that it gives powerful results even under very bright or very dark light. All these elements will increase as the brightness increases, but the same result will be obtained based on the central element. In Figure 4.11 one can think of the first as normal light and the second as bright light.(Singh, 2021)

12	15	18
5	8	3
8	1	2

1	1	1
0	8	0
1	0	0

42	55	48
35	38	33
38	30	32

Figure 4.11 Normal and Bright Light LBPH Algorithm

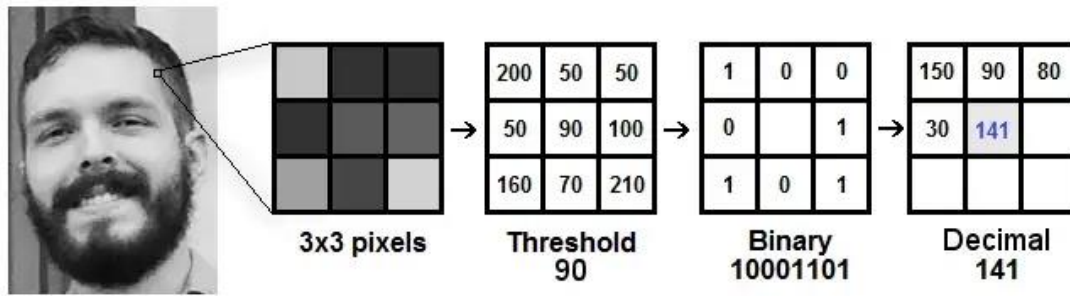


Figure 4.12 Local Binary Pattern Representation

We said that the center value is taken as threshold and this value is used to define new values from 8 neighbors. In the image above, it was revealed by Ahonen, who created the algorithm that each decimal value will correspond to a gray scale. With the `LBPHFaceRecognizer_create()` function, the `grid_x` and `grid_y` values are set to 8 by default. In this way, each image we send to this algorithm is divided into 8x8 and 16 squares. The histogram of each frame is calculated and these 16 histograms are combined to form a large histogram. This final histogram becomes a characteristic representation of the original image. When we give a new image to this model after it has been trained, a histogram of this new image is created with the same operations. This new histogram compares the previously trained histograms. Euclidean distance (Figure 4.13) is used to compare histograms, that is, to calculate the distance between two histograms. As a result of this operation, a confidence value is returned. If this value has the lowest value with which histogram is calculated, the image given as input is returned as a prediction result with the id of this histogram.(Prado, 2018)

$$D = \sqrt{\sum_{i=1}^n (hist1_i - hist2_i)^2}$$

Figure 4.13 Euclidean Distance

4.2. Experiments

4.2.1. Changing Learning Rate

A low learning rate provides a more reliable result, but a high one may cause the model to not converge or not diverge. (Nabi, 2019) As seen in Figure 4.14, the results of giving the learning rate too low or too high a value are not healthy for model training.

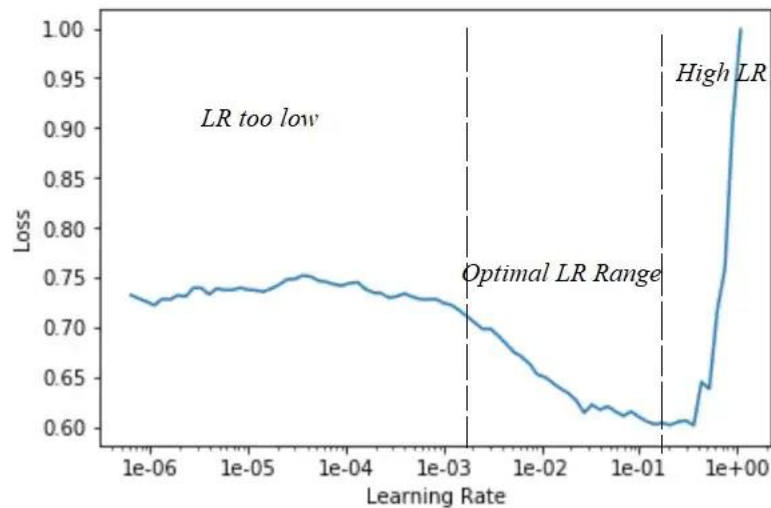

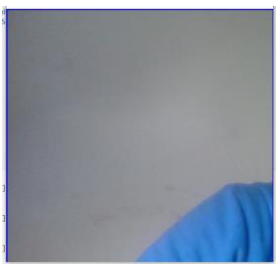
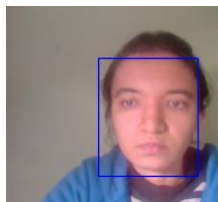
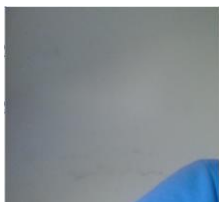


Figure 4.14 Learning Rate Effects on Loss

Although this value is generally used as 0.0001, the outputs of the model trained with this value in order to observe the output when this value is 0.1 in order to display the change are as follows.

Table 4.1 Results of Learning Rate Changes




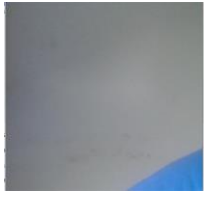

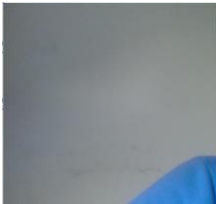
Learning Rate 0.1		Learning Rate 0.0001	
Face on Screen	No Face on Screen	Face on Screen	No Face on Screen
[1.]	[1.]	[0.99987614]	[6.363032e-06]
[0., 0., 1., 1.]	[0., 0., 1., 1.]	[0.6384512, 0.2837015, 0.99872214, 0.82862395]	[1.2123410e-05, 1.2933975e-05, 1.6488408e-05, 2.0819551e-05]
			

As mentioned above, this change in learning rate led to a significant change in the results.

4.2.2. Changing Dense Layer Unit

It was mentioned when explaining the layer that we should specify as many units as the number of outputs we want to get as a result of classification in the last layer. However, at the point where the unit number of the dense layers in the hidden layers is determined according to what, the following results were obtained by giving 3 different values to the unit number of the dense layer in this project.

Table 4.2 Results of Unit Number in Dense Layer Changes

Unit Number 512		Unit Number 1024		Unit Number 2048	
Face on Screen	No Face on Screen	Face on Screen	No Face on Screen	Face on Screen	No Face on Screen
[0.99929285]	[7.992755e-06]	[0.99994326]	[1.945332e-06]	[0.99987614]	[6.363032e-06]
[0.35400575, 0.21977036, 0.7932767 , 0.8028413]	[2.3707908e-05, 3.0666495e-05, 5.2695374e-05, 4.6149085e-05]	[0.43301308, 0.22209606, 0.9086346 , 0.8252162]	[8.0880964e-06, 9.1652655e-06, 8.9053192e-06, 1.3188908e-05]	[0.6384512, 0.2837015, 0.99872214, 0.82862395]	[1.2123410e-05, 1.2933975e-05, 1.6488408e-05, 2.0819551e-05]
					


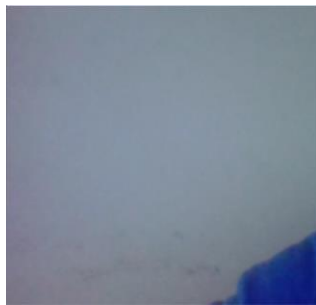
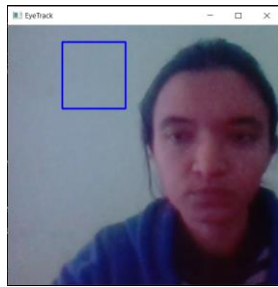
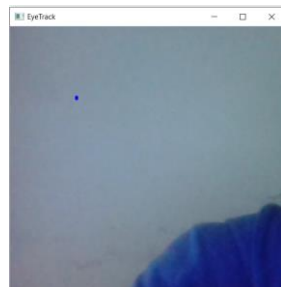
As a result of this experiment, the unit change in the hidden layer did not cause a big change in the results.

4.2.3. Changing Activation Functions

a. Using softmax instead of sigmoid in the output layer

Generally, sigmoid or softmax functions are used in the classification layer in CNN models. However, if there is a binary estimation, sigmoid is used, and softmax is used when classification is made between 2 or more classes. When we use softmax in our project, the behavior of the model is examined.

Table 4.3 Behavior comparison of softmax and sigmoid function in the last layer

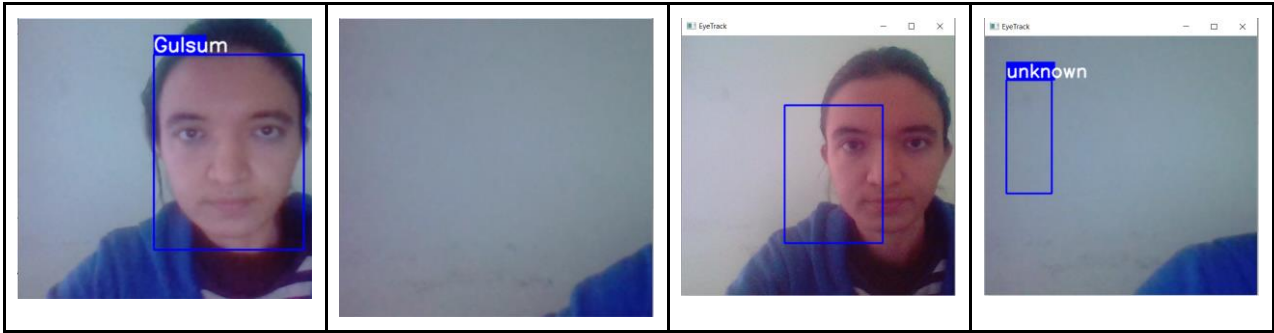
Model with Sigmoid in Last Layer		Model with Softmax in Last Layer	
Face on Screen	No Face on Screen	Face on Screen	No Face on Screen
array([[0.9995716]])	array([[6.674347e-06]])	array([[1.]])	array([[1.]])
[0.5444842, 0.2546344 , 0.99676114, 0.8275146]	[1.3257989e-05, 1.3121627e-05, 1.8851977e-05, 2.1685752e-05]	[0.18823092, 0.09021983, 0.40331295, 0.31823638]	[0.23704197, 0.27112883, 0.23797984, 0.2538494]
			

b. Using sigmoid instead of relu in the hidden layer

The reason I did this experiment is that relu can be an alternative to sigmoid and tanh functions and I aim to examine its behavior in this model.

Table 4.4 Behavior comparison of sigmoid and relu function in the last layer

Model with Relu in Hidden Layer		Model with Sigmoid in Hidden Layer	
Face on Screen	No Face on Screen	Face on Screen	No Face on Screen
array([[0.9995716]])	array([[6.674347e-06]])	array([[0.9629868]])	array([[0.8201492]])
[0.5444842, 0.2546344 , 0.99676114, 0.8275146]	[1.3257989e-05, 1.3121627e-05, 1.8851977e-05, 2.1685752e-05]	[0.35625082, 0.25090438, 0.70277816, 0.715686]	[0.07810568, 0.16419062, 0.2454506 , 0.5820816]



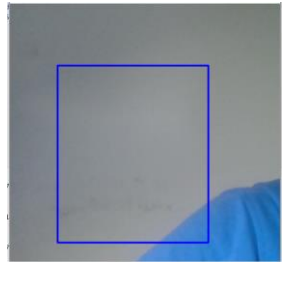
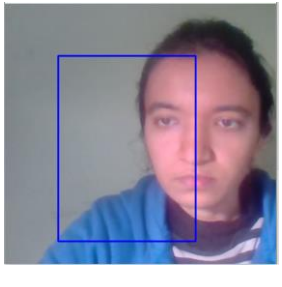


Except for relu, the closest hidden layer activation function is sigmoid to the result we want to get about the single point value when there is a face on the screen. This also applies to the bounding box estimation point. But when there is no face on the screen, the difference in the output results is huge. This is the reason why the desired result cannot be obtained from the model as a result of using sigmoid in the hidden layer.

c. Using tanh instead of relu in the hidden layer

It has been mentioned that the Relu function is a good alternative to the tanh function. For this reason, the results of the experiment carried out in order to observe how the tanh function will give a result in this project are as follows.

Table 4.5 Behavior comparison of tanh and relu function in the hidden layer

Model with Relu in Hidden Layer		Model with Tanh in Hidden Layer	
Face on Screen	No Face on Screen	Face on Screen	No Face on Screen
array([[0.9995716]])	array([[6.674347e-06]])	array([[0.9820064]])	array([[0.97891504]])
[0.5444842, 0.2546344 , 0.99676114, 0.8275146]	[1.3257989e-05, 1.3121627e-05, 1.8851977e-05, 2.1685752e-05]	[0.19771038, 0.19237092, 0.7032593, 0.8724018]	[0.21073799, 0.18839508, 0.7305203, 0.8846346]
			

In this experiment, in the model made with the tanh function, it was seen that the face was on the screen and there was not enough change in the bounding box estimations even though it was in different positions.

5. CONCLUSION AND FUTURE WORK

In this project, the criterion of detecting a targeted face by a trained model has been achieved. Trials have been made to improve this detection function. In order for a model to give better results, it is important to choose the layers that make up the model and the activation functions of these layers. In addition, the loss function and learning rate, which are necessary for the optimization of the model, play an important role in the performance and construction of the model.

In addition to face detection, the LBPH Face Recognition tool of the opencv library was used in the project for face recognition, and it was possible to recognize the introduced people.

In the later stages of this project, improvements can be made on the face recognizer. The face detection model vgg16 model can be fully trained, not with the transfer learning technique.

6. REFERENCES

- [1] E. Hjelmås ve B. K. Low, *Face detection: A survey. Computer Vision and Image Understanding*, 2001.
- [2] D. Landup, *Don't use flatten() - global pooling for cnns with tensorflow and keras*, 2022.
- [3] G. Dumane, *Introduction to convolutional neural network (CNN) using tensorflow*, ” *Medium*, 2020.
- [4]
- [5] P. Baheti, *Activation functions in neural networks [12 types & use cases]*.
- [6] K. Le, *An overview of VGG16 and nin models*, 2021.
- [7] “*Transfer learning and fine-tuning : Tensorflow Core*, ” *TensorFlow.*, 2022.
- [8] H. Bandyopadhyay, “*Labeling with labelme: Step-by-step guide [alternatives + datasets]*, ”, 2023.
- [9] R. Pramoditha, “*How to choose the right activation function for neural networks*, ”, *Medium*, 2022.
- [10] J. Nabi, *Hyper-parameter tuning techniques in deep learning*,, *Medium*, 2019.
- [11] P. Singh, “*Understanding face recognition using LBPH algorithm*, ” *Analytics Vidhya.*, 2021.
- [12] K. S. d. Prado, “*Face recognition: Understanding LBPH algorithm*, ”, *Medium*.
- [13] F. Zhuang, Z. Q, i. Duan, D. Xi, Y. Zhu, H. Zhu ve S. Member, *A Comprehensive Survey on Transfer Learning*, 2019.
- [14] P. Viola ve M. Jones, *Rapid Object Detection using a Boosted Cascade of Simple Features.*, 2001.
- [15] G. Pai, *Recent Trends in Face Detection Algorithm*, 2019.
- [16] X.-L. Xia, C. Xu ve B. Nan, *Facial Expression Recognition Based on TensorFlow Platform.*

- [17] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin ve A. A. Kalinin, *Albumentations: Fast and flexible image augmentations*, Switzerland, 2020.