HAND GESTURE RECOGNITION

Zachogeorgos Zisimos (P2821904)

Logari Zoi (P2821910)

Stavrou Androniki (P2821926)

Sofroniou Dimitra (P2821929)

2021

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS

MSc in Business Analytics

1. CONTENTS

1 Introduction 2
1.1 Description 2
1.2 Mission 3
2. Data 4
2.1 Data Collection 4
2.2 Dataset Overview 4
2.3 Data Preprocessing 5
3. Algorithms & Experiments 6
3.1 CNN Model 1 7
3.2 CNN Model 2 9
3.3 SVM Model 11
4. General Information 15
4.1 Hand Gesture Recognition App 15
4.2 Members & Roles 15
4.3 Timeplan 16
5. References 17

1 INTRODUCTION

1.1 DESCRIPTION

Over the years, the interaction between humans and computers has been increased. Computer scientists have developed algorithms and methods in order to solve problems and facilitate people's lives.

The use of hand gestures in different software applications has contributed towards improving the referred interaction. It is considered that not only the recognition of faces but also of hand gestures is important to prevent critical situations in time. Precisely, facial recognition is the process of identifying or verifying the identity of a person using their face. Similarly, gesture recognition analyzes human motions to detect for example aggressive behavior in a crowd of people.

According to "marketsandmarkets" the global post-COVID-19 facial recognition market size is expected to grow from USD 3.8 billion in 2020 to USD 8.5 billion by 2025. All in all, it is expected to witness substantial growth for technology vendors in the next five years.

Occasioned by the numerous urgent incidents that have been observed the last years hand gesture recognition could be a solution to this problem. However, surveillance cameras, cannot prevent or detect such incidents except for recording them. How many times have you been in a public area, where Closed circuit television (CCTV) exist, but you do not feel safe? Of course, the human operators in a CCTV system are subject of fatigue and subtraction, which make it difficult for them to detect criminal and terrorist behavior or even a health problem of the human under surveillance. Additionally, the cues leading up to these behaviors may be subtle, easily.

Other fields where hand gesture recognition could be applied is for disabled or deaf-mute people. In other words, there could exist algorithms which would convert hand gestures to words or phrases so as to communicate with other people who are not familiar with sign language.

Finally, last years hand gesture recognition is applied in video games, augmented reality and robot control as well as in smartphones.

1.2 MISSION

Our objective is to build a model that will detect not only the hand but also the gestures. The model will take as input an image of a hand and will give as result the type of the gesture between ten different types. This will have a lot of applications according to the above referred ways of using an algorithm like this.

Having in mind the trigger of creating a hand gesture recognition model is the plenty of urgent incidents that have happened the last months in Greece, we think that it would be a perfect fit for close circuit television (CCTV) systems in order to prevent situation like this. However, one should keep in mind that this could be used after the necessary education behalf of the citizens who should know each type of gesture what exactly mean in order to be called automatically either the police, the fire department, the security or the hospital.

In this point, our task will be divided in two categories:

- Hand recognition
- Recognition of ten different labels

In other words, in the first step, it will be detection of a hand in an image and in the next step the algorithm will classify the gesture of the image between the ten different types that we have decided to study.

For the second step, there will be presented different type of models, such as Convolutional Neural Network (CNN) and Support Vector Machine (SVM) with different configurations.

2. DATA

2.1 DATA COLLECTION

The project is developed using the Google Colab environment. The dataset is available on Kaggle. We decided not to download the dataset locally, so we used the Kaggle API in Google Colaboratory in order to fetch the Kaggle dataset into Google Colab. This method is a way to download the Kaggle dataset into Colab with minimal effort. The first step is to go to our Kaggle account and create a new API token, so a file named kaggle.json is downloaded containing our username and token key. The next step is to create a folder named Kaggle where we will be storing our Kaggle dataset and upload our kaggle.json file into Kaggle folder.

We get an authorization code using the URL prompted and provide it in the empty box that appears. Then, we provide the config path to kaggle.json.

After, we change the working directory and go to Kaggle and copy the API command to download the dataset. Finally, we use the unzip and rm command in order to unzip the dataset and remove the zip file. In this way, we can use our extracted files with ease directly from our Google Drive.

2.2 DATASET OVERVIEW

The most important is to find the appropriate data. Thankfully, there are a lot of sources to find the data needed. We are going to use images for training. The images used are taken from the Hand Gesture Recognition Database available on Kaggle. It contains 20000 images with various hands and hand gestures. The dataset contains 10 hand gestures of 10 different people. There are images of 5 females and 5 males. The 10 hand gestures are the following: palm, l, fist, fist moved, thumb, index, ok, palm moved, c and down. Our aim is to classify different images of hand gestures, this means that we have

to train a machine learning algorithm to learn the hand gestures and be capable of classifying them correctly.

2.3 DATA PREPROCESSING

Prior to training the algorithms, we have to prepare the images. The first step is to load all the images into an array and all the labels into another one. We use the cv2.cvtColor() method to convert an image from one color space to another, specifically to gray color. Also, we use cv2.normalize() method that helps us by changing the pixel intensity and increasing the overall contrast and the cv2.resize method in order to reduce image size so training can be faster.

What is more, data augmentation is used. Data augmentation takes the approach of generating additional training data from your existing examples by augmenting them using random transformations that yield believable-looking images. This helps expose the model to more aspects of the data and generalize better.

After, we split data into training dataset, which will be used during training phase, and on validation dataset, which will be used during training phase in order to validate the results from training. The data were divided on a percentage of 70% for training and 30% for validation.

3. ALGORITHMS & EXPERIMENTS

In our project, we decided to use CNN models as well as SVM since they seem to be the most effective models for prediction. Following, we will describe the two different algorithms.

What is a Convolutional Neural Network (CNN)?

Firstly, we constructed a CNN model. To be more precise, a CNN is a type of artificial neural network used in image recognition and processing that is specifically to process pixel data. In other words, a CNN takes the image's raw pixel data as input and "learns" how to extract these features, and ultimately infer what object or what gesture in our case, they constitute.

The CNN receives an input feature map: a three-dimensional matrix where the size of the first two dimensions corresponds to the length and width of the images in pixels. The size of the third dimension is 3 (corresponding to the 3 channels of a color image: red, green, and blue)

A convolution extracts tiles of the input feature map and applies filters to them to compute new features, producing an output feature map, or convolved feature (which may have a different size and depth than the input feature map).

Convolutions are defined by two parameters:

· Size of the tiles that are extracted

· The depth of the output feature map which corresponds to the number of filters that are applied.

What is a Support-vector Machine (SVM)?

An SVM is a supervised learning model with associated learning algorithms that analyze data for regression or classification analysis, as our problem. SVM generates optimal hyperplanes in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane (MMH) that best divides the dataset into classes. In its most simple type, SVM doesn't support multiclass classification natively. It supports binary classification and separating data points into two classes. For multiclass classification, the same principle is utilized after breaking down the multiclassification problem into multiple binary classification problems.

3.1 CNN Model 1

First of all, our model type is sequential, that is to say that it allows us to build the model layer by layer. Then we add a series of filters to the raw pixel data to extract and learn higher-level features.

In Figure 1 one can see the steps that are followed for a CNN model.

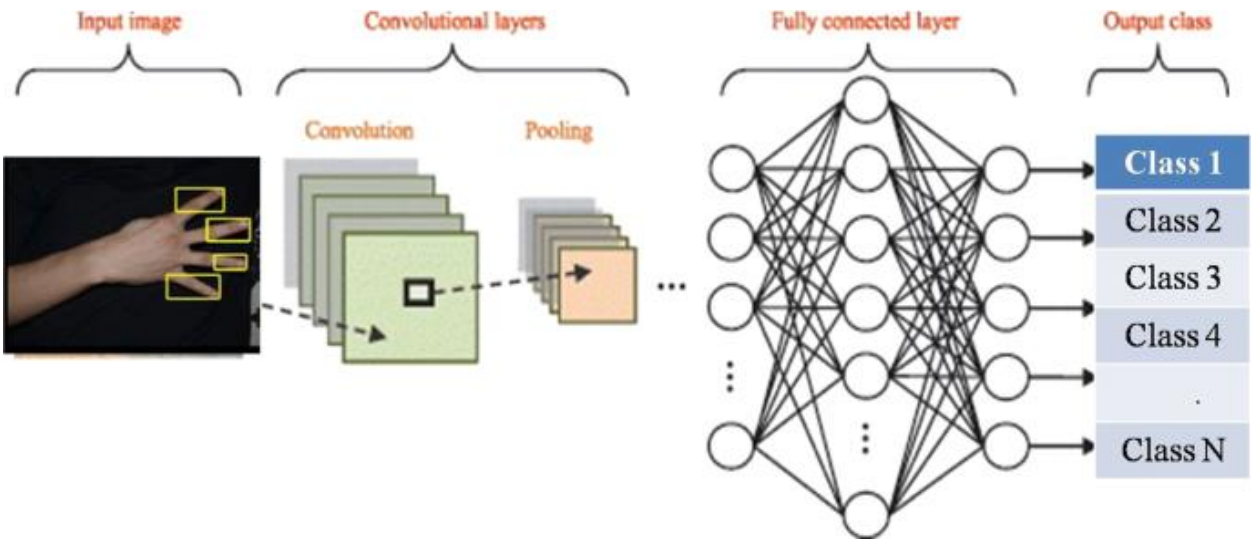


Figure 1: CNN algorithm.

CNNs contain three components:

· *Convolution Layers*, which apply a specified number of convolution filters to the image.

The first two layers are Con2D layers. 16 in the first layer and 8 in the second layer are the number of nodes in each layer and it depends on the size of the dataset. Our first layer also takes an input shape. This is the shape of each input image, 120,320,1 with the 1 signifying that the images are grayscale.

· *Pooling Layers* which downsample the image data extracted by the convolutional layers to reduce the dimensionality of the feature map in order to decrease the processing time.

We used the max pooling algorithm which applies a max filter to non-overlapping subregions of the initial representation. In this way we reduce dimensionality and help over-fitting by abstracting an abstracted form of representation.

· Dense or fully connected layers which perform classification on the features extracted by the convolutional layers and downsampled by the pooling layers.

Fully connected layers connect every neuron in one layer to every neuron in the next layer. Flatten serves as a connection between the convolution and dense layers and flattens the pooled images into one long vector. Dense is the layer type we will use in our output layer. Dense is a standard layer type that is used in many cases for neural networks

```
# Construction of model
model2 = Sequential()
model2.add(Conv2D(16, (3, 3), activation='relu', input_shape=(120, 320, 1)))
model2.add(MaxPooling2D((16, 16)))
model2.add(Conv2D(8, (3, 3), activation='relu'))
model2.add(MaxPooling2D((2, 2)))
model2.add(Flatten())
model2.add(Dense(10, activation='softmax'))
```

Figure 2: CNN Model 1.

Finally, as the last-layer activation we used the "softmax" since it is the most suitable for multiclass, single-label classification.

3.2 CNN MODEL 2

Then we tried for another CNN model in order to ameliorate our results. After a lot of trials we found that the one with the 32 and 64 nodes in each of the Conv2D layers attributes better.

Then we used the "relu"activation and as last-layer activation the "softmax" for the same reason as in model 1.

In this one, we also used the MaxPooling Algorithm since it reduces the representation size and speeds the computation and makes some of the features more prevalent and robust.

```
model = Sequential()
model.add(Conv2D(32, (5, 5), activation='relu', input_shape=(120, 320, 1)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

Figure 3: CNN Model 2.

This model seems to be the most accurate according to the qualitative and error analysis which follows.

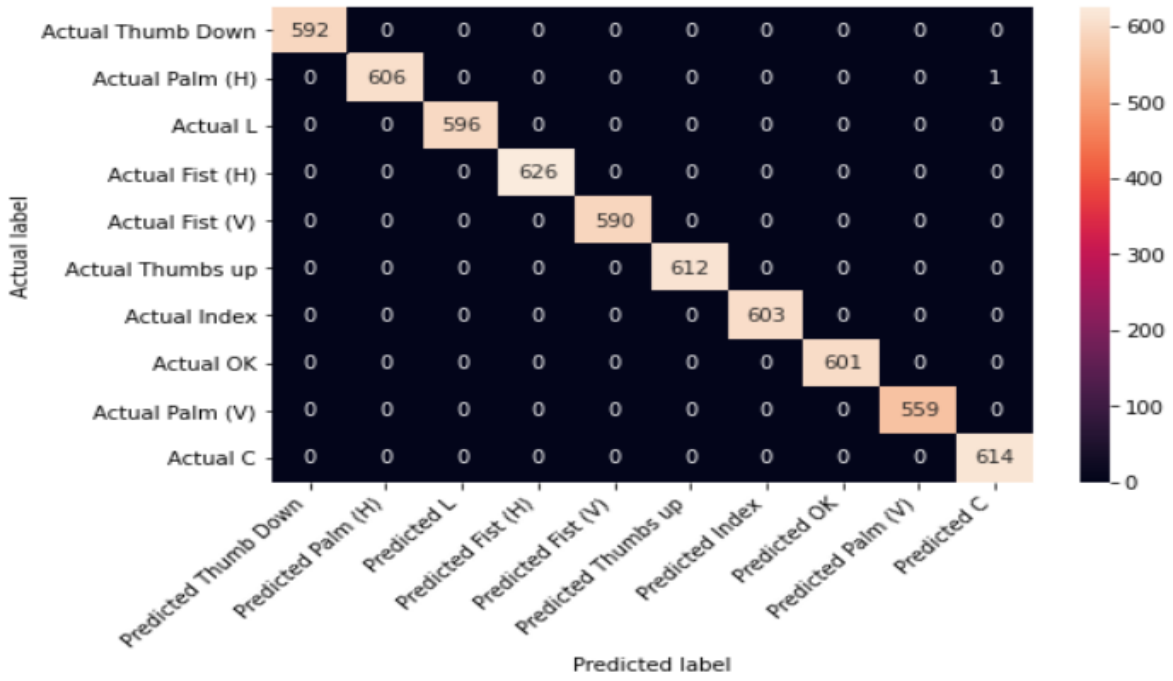


Figure 4: Confusion Matrix of CNN Model 2.

Figure 5: Accuracy Loss in Training & Validation Sets.

In Figure 5 one can see the training loss and the validation loss in the ten different epochs. The validation loss in the training set is zero instead of the validation set where it fluctuates between 0.16 and 0.19.

On the other hand, as you can see from the Figure 6, training accuracy and validation accuracy do not differ a lot. Specifically, the accuracy in the validation set is over 99%.

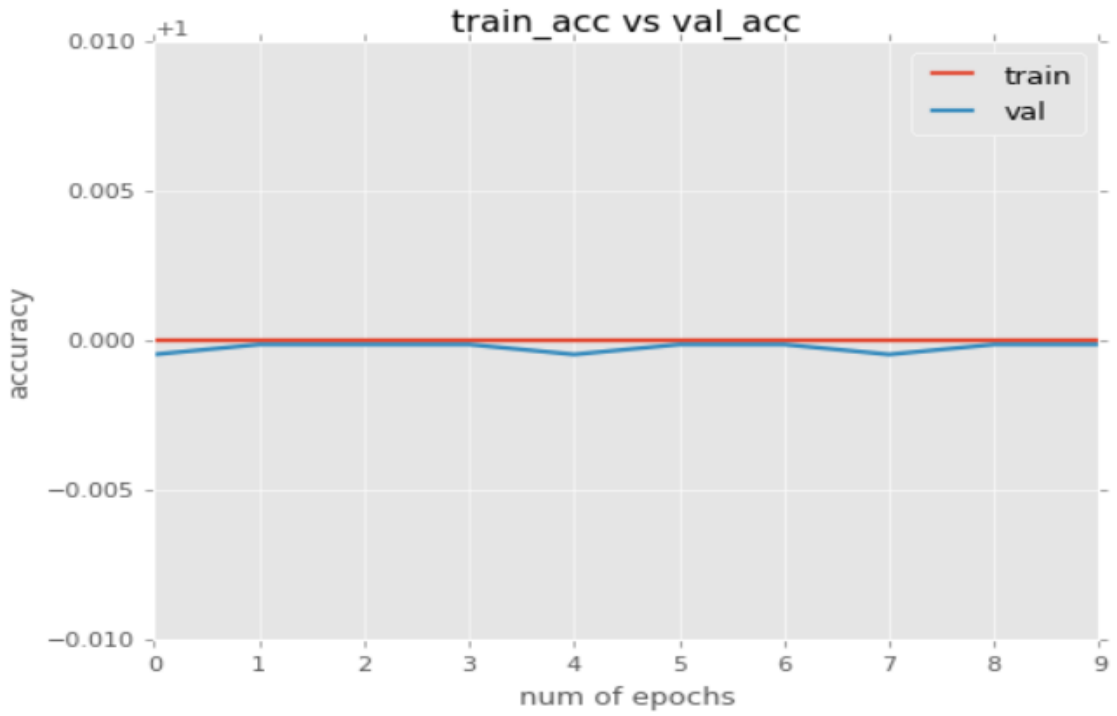


Figure 6: Accuracy in the Training and the Validation sets.

3.3 SVM MODEL

Another model that we applied on our data is the SVM model. That kind of model is effective in high dimensional spaces, and uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

The kernel that we choose for our SVM model is the RBF (Radial Basis Function). When training an SVM with the Radial Basis Function (RBF) kernel, two parameters must be

considered: C and gamma. The parameter C, common to all SVM kernels, trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly. gamma defines how much influence a single training example has. The larger gamma is, the closer other examples must be to be affected.

In our case we use the default parameters for C=1 and for gamma= 0.001

Type of kernels: - we chose the default RBF

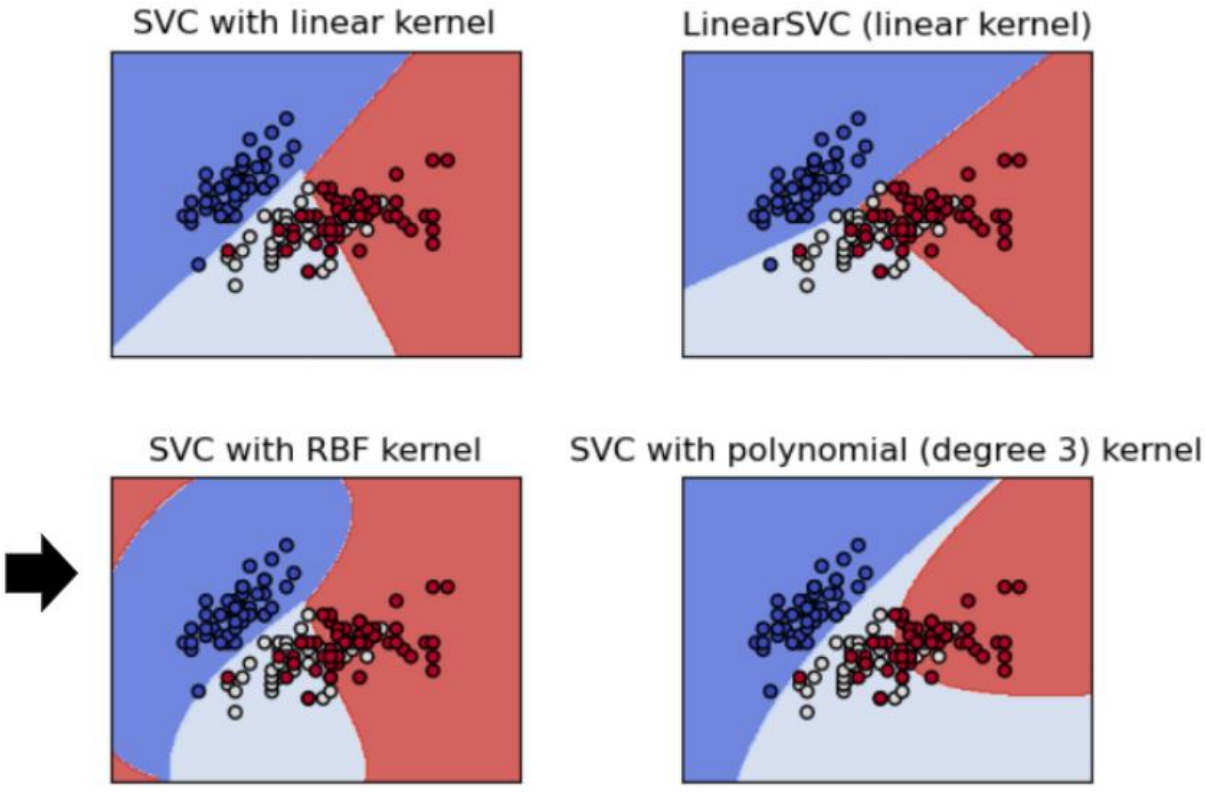


Figure 7: SVM Algorithm.

Data preparation

Apart from the cutting of the images that we use for both SVM and CNN models , for the specific SVM model we had to resize our images . So we changed them to 10x10 in order our model to be able to process them as SVM has a greater limitation in the size.

Of course, our images are in grayscale.

In order to proceed with split of data to training and test we have imported the following

From our dataset 70% will be used for training and 30% for testing.

In order to better present the results of our model , as classification results and confusion matrix we import the following

The confusion matrix

We can observe that there are ten different buckets and the right predictions are depicted in absolute numbers in the digonial of the table.

Our model, when it was unable to correctly classify an image , places it in the first bucket so that is why we have the following view in our results.

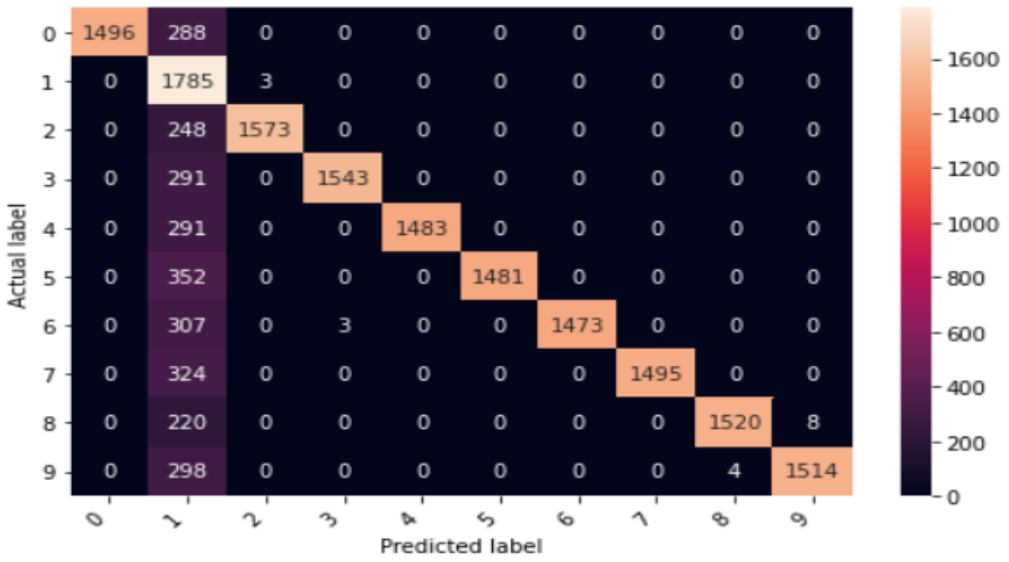


Figure 8: Confusion Matrix of SVM Model

Below we present the results from each run of the 10-fold CV.

	precision	recall	f1-score	support
0	1.00	0.84	0.91	1784
1	0.41	1.00	0.58	1788
2	1.00	0.86	0.93	1821
3	1.00	0.84	0.91	1834
4	1.00	0.84	0.91	1774
5	1.00	0.81	0.89	1833
6	1.00	0.83	0.90	1783
7	1.00	0.82	0.90	1819
8	1.00	0.87	0.93	1748
9	0.99	0.83	0.91	1816
accuracy			0.85	18000
macro avg	0.94	0.85	0.88	18000
weighted avg	0.94	0.85	0.88	18000

Figure 9: Metrics of SVM Model.

Precision means the percentage of your results which are relevant. On the other hand, recall refers to the percentage of total relevant results correctly classified by your algorithm.

Precision, which describes how many of the predicted positives are actually positive, is on 94% which means that the predictive ability is very satisfying

while recall, which describes how many of the actual positives have been predicted as such, is on 85%. Both metrics are acceptable but especially recall is not very impressive , F1 score is also ok.

F-1 score, which is a harmonic mean of precision and recall. For problems where both precision and recall are important, one can select a model which maximizes this F-1 score. For other problems, a trade-off is needed, and a decision has to be made whether to maximize precision, or recall.

Of course, we expect to challenge the model on live camera results in order to feel more confident in our comments.

4. GENERAL INFORMATION

4.1 HAND GESTURE RECOGNITION APP

After having described and explained how we created our model, everything comes down to the question how these can be useful in real world cases. For this reason, we tried to create a simple application which illustrates how these models behave in real time problems. As said before, our goal is to create an application on which we will automatically identify specific hand gestures. We will try to simulate CCTV functionality.

To accomplish that, we have created an HTML application using Flask web frame-work. Application's functionality is simple. There are two buttons, "Start Recording" and "Stop Recording", which start and stop live video correspondingly. When camera opens, we perform hand gesture recognition in real time, returning labels based on classification results. There is no need to describe how the Flask application is set up as it is out of the scope of the project. This work is still in progress and the final results will be provided in the presentation of this project.

4.2 MEMBERS & ROLES

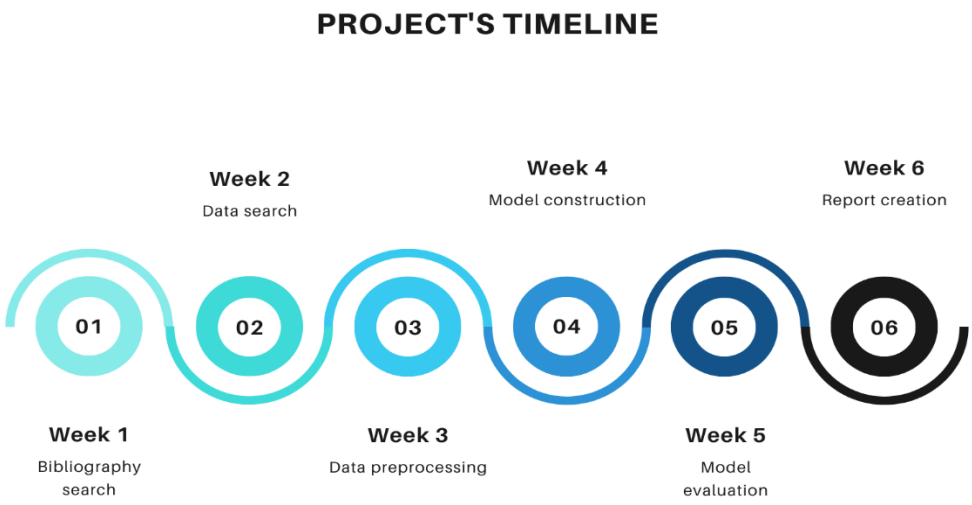
Our team consists of four members. Zisimos Zachogeorgos and Zoi Logari have a business background, Dimitra Sofroniou has a mathematical background and Androniki Stavrou has a statistical background.

We splitted in teams of two members and each team developed one of the models mentioned in previous sections. All four members contributed to each step of this assignment by providing code and ideas on how to make the models.

Zachogeorgos Zisimos and Dimitra Sofroniou were responsible for the SVM model. Zoi Logari and Androniki Stavrou were responsible for the CNN models.

All members contributed to the data preprocessing and the video application.

4.3 TIMEPLAN


PROJECT'S TIMELINE
Week 2
Data search
Week 4
Model construction
Week 6
Report creation
01
02
03
04
05
06
Week 1
Bibliography search
Week 3
Data preprocessing
Week 5
Model evaluation

5. REFERENCES

- Data: https://www.kaggle.com/gti-upm/leapgestrecog/version/1

- https://medium.com/analytics-vidhya/how-to-fetch-kaggle-datasets-into-google-colab-ea682569851a
- https://searchenterpriseai.techtarget.com/definition/convolutional-neural-network
- https://tensorflow.google.cn/tutorials/images/classification?hl=zh-cn
- https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
- https://medium.com/analytics-vidhya/image-classification-using-machine-learning-support-vector-machine-svm-dc7a0ec92e01
- https://scikit-learn.org/stable/modules/svm.html#shrinking-svm
- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, Aurelien Geron, September 2019, O'Reilly Media Inc.
- Notes of lectures on Machine Learning & Content Analytics, MSc in Business Analytics at Athens University of Economics and Business, Haris Papageorgiou
- https://github.com/ageron/handson-ml2

6. LIST OF FIGURES

Figure 1: CNN algorithm. 7
Figure 2: CNN Model 1. 8
Figure 3: CNN Model 2. 9
Figure 4: Confusion Matrix of CNN Model 2. 10
Figure 5: Accuracy Loss in Training & Validation Sets. 10
Figure 6: Accuracy in the Training and the Validation sets. 11
Figure 7: SVM Algorithm. 12
Figure 8: Confusion Matrix of SVM Model. 13
Figure 9: Metrics of SVM Model. 14