Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Media Informatics

Dénes Csizmadia

VIDEO BASED MOBILE APPLICATION OF EMOTION RECOGNITION BY DEEP LEARNING

SUPERVISOR

Dr. Gábor Szűcs

BUDAPEST, 2017

HALLGATÓI NYILATKOZAT

Alulírott **Csizmadia Dénes**, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2017. 12. 15.

...…….

Csizmadia Dénes

Table of Contents

Abstract

The Artificial Intelligence also emerged as a new field of research in the middle of the 20th century. Recently, it has been very popular again, thanks to the development of technology. Whereas the large computing capacity enabled large amounts of data to be managed.

In the beginning of my thesis work I will present the machine learning, including the deep neural networks in more details, which are the subareas of the Artificial Intelligence. My work is not only about building the model, but also about introducing a field of application. My task was to create a system that can detect human emotions, where the user receives a feedback about their emotions via a mobile app.

The rest of my thesis described implementation of two more applications in details. First of all, there will be represented such a version, which is based on a server connection. The web server is capable of loading in pre-trained models and of recognizing emotions via videos, coming from mobile applications. I utilized Convolutional and Recurrent Neural Networks as models, in which a video database labelled with emotions meant the training data.

The other application of my thesis work is built on a new framework, which was released by Apple in September, 2017. With the help of it, the trained models can already be integrated into the application and are accessible directly from there.

Összefoglaló

A mesterséges intelligencia már a XX. század közepén felmerült, mint egy új kutatási terület. Mostanában azonban újra nagy népszerűségnek örvend, köszönhetően a technológia fejlődésének. A nagy számítókapacitás ugyanis nagy mennyiségű adatok kezelését tette lehetővé.

A dolgozatom elején ennek egy részterületét, a gépi tanulást mutatom be részletesebben, azon belül is a mély neurális hálókkal történő tanítást. A munkám nem csak a modell felépítésével és tanításával foglalkozik, hanem egy felhasználási területet is bemutat. A feladatom az volt, hogy egy olyan rendszert alkossak, amely emberi érzelmeket képes felismerni. A felhasználó egy mobil alkalmazáson keresztül kap visszajelzést az érzelmeiről.

A dolgozatom további részében két ilyen alkalmazás megvalósítását mutatom be részletesen. Elsőként egy szerver kapcsolatot igénylő verziót ismertetek. A web szerver képes előre tanított modellek betöltésére, és a mobilalkalmazástól kapott videó alapján az érzelmek felismerésére. Modelleknek Konvolúciós és Rekurrens Neurális Hálózatokat használtam, amelyekhez egy érzelmekkel címkézett videó adatbázis szolgált tanító adatként.

Diplomamunkám egy másik alkalmazása, az Apple által 2017 szeptemberében kiadott új keretrendszeren alapszik. Ennek segítségével a tanított modellek már közvetlenül az alkalmazásba építhetőek és szerver kapcsolat nélkül onnan használhatóak.

1 Introduction

1.1 Motivations

I personally have been interested in this topic for a while. During my electric engineering bachelor studies, I was specialized in info-communication technologies. That's why I have met the variability of the media types and got a solid base in the topic of the image, sound and video processing. As the title of my thesis work suggests I had to handle video files and media streams, so I could utilize these former studies.

Besides the fact, that there are two fields which I am really interested in, I would like to involve them into my thesis topic. One of them is the mobile software engineering, so I have developed mobile applications to the iOS platform. Developing for mobile devices was not a completely new field for me, even though I met several new cases, which had to be solved. The other one is the data mining, which is connected to my master studies where I am specialized in business intelligence technologies. It can be said that the data mining is one of the biggest trends nowadays, and I personally believe that it has a lot of fields which could be research topics for a long time in the future, for example the emotion recognition with the deep learning technologies.

My consultant Dr. Gábor Szűcs also motivated me in the topic selection. I really liked his lectures and his researching field, that's why I decided to ask him to be my consultant. I would like to say thanks as I could always turn to him for help with the technical and administrative questions. At last, but not least I would like to say thank you to Dóra Kléri for helping me with the grammar correction and clarification.

1.2 Actuality of the topic

Human emotion analysis (also known as emotion recognition) has attracted significant attention in the computer vision community during the past decade, since it lies in the intersection of many important applications, such as human computer interaction, crowd analytics etc. Besides that, the sentiment analysis of the social media contents is also significant because several economic indicators can be measured and user sentiments towards events or topics can be extracted. There is no doubt, that the images and videos represent an increasing proportion among the media contents and more and more effort is needed to handle them [1][2].

Having already dealt with the emotion recognition of the visual and multimedia contents, but formerly human observation base patterns and relationships were searched in a face image or in a speech sound file. However, significant results cannot be achieved due to the diversity of the cases. Nowadays, it has become a central trend again thanks to the boom of the Artificial Intelligence (AI) technologies, for instance machine learning. As a computer can do much more observation, so it can set up a more precise system of rules for the detection of each emotions.

The AI is not just a topic nowadays, the goal of artificial intelligence was arbitrarily set by Alan Turing, an English mathematician who was a pioneer in this field during the 1950s. He proposed the imitation game test where a computer needs to be indistinguishable from a machine. The technological progress has not been there for researchers to create an intelligence that passes the Turing test. Because of the disappointment, the founding of the AI researches and the interest decreased. Thus, until the last few years, it was a period known as AI winter [3].

Two relatively recent trends have sparked: widespread use of AI and machine learning again. Firstly, the availability of massive amounts of training data, which are manageable with our large capacity storage drives. Remarkable that 90% of the data in the world today has been created in the last two years alone. Secondly, the powerful and efficient parallel computing provided by GPU (Graphics processing unit) computing. In the following Figure 1, the stock prices in the last five years of the NVIDIA company, which is the most significant graphic card producer, can be seen. This is how the significance and the spread of AI researches can be perceived [4].

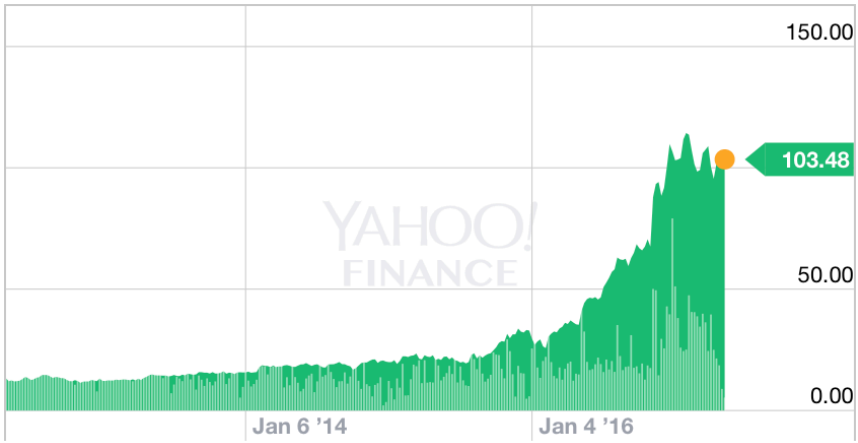

Figure 1 : NVIDIA stock prices [4]

The goal is the same now, which is that the computers must behave like a human being, but with the difference that there are less technical limitations. Just to mention some of the greatest achievements:

In 2015, the Google company released the InceptionV3, which was the third edition of the inception convolutional neural network. This network can classify images more accurately than the humans. In 2016, the AlphaGo (developed by Alphabet Inc.'s Google DeepMind) AI computer program won against the world champion in the last game, in which man was better than the computer, called GO. According to many AI researchers, the greatest innovation in 2017 will be the chat bot, which can pass through the Turing test, so we will not be able to notice that we are talking to a machine. In this, emotional recognition is also really important.

1.3 The description of the task

As the title of my thesis work suggests the goal is to develop such an application, which allows recognition of emotions in real time. I have developed the applications for iOS operation system, which is released by Apple. The recognition is based on machine learning models, which have been taught with an emotionally labelled video database. Initially I have designed a system based on a server-client architecture, because at the time when I started to write my thesis I did not have the opportunity to make prediction utilizing the model directly on the iOS system. The Apple came up with a solution for this problem by releasing the 11th version of the iOS in this September. The CoreML, the framework for machine learning, was the biggest attraction in this version of the operation system. In my thesis work I am going to introduce the initial architecture and an enhanced one without web server, based on this framework. In both cases the goal is to identify the six basic emotions.

1.3.1 The six basic facial emotions

The science of the facial emotions begins with Charles Darwin, who stated in his book of 1872, in The Expression of the Emotions in a Man and Animals: “the same state of mind is expressed throughout the world remarkable uniformity”. Since then many studies have attempted to classify human emotions, and prove how your face can give away your emotional state [44][45].

The most significant research came from Paul Ekman, who was a pioneer psychologist in the field of emotion recognition. Ekman and his research team investigated isolated culture of people from the Fori tribe in Papua New Guinea in 1972. The tribe members were able to identify correctly the emotion of anger, disgust, fear, happiness, sadness and surprise from the pictures. Moreover the pictures, having been made here, could be recognized all over the world. Since then it has been clarified by several scientists that the expression of these six emotions is regardless of the culture and it is not a result of a learnt behaviour. That is why we call them basic or primary emotions [44].

In my thesis work I concentrated on the recognition of these emotions with help of deep learning. Secondary emotions like love, shame or pride can only be seen in micro expressions, which are very quick facial expressions involuntarily made by people within particular circumstances. These ones occur often in a fraction of a second, therefore the capacity of smartphone camera is not enough to recognize them. They can only be captured by special high speed cameras [44].

1.3.2 **Emotion recognition from the human voice**

The emotions in the human speech can be examined in two ways. First, when we would like to extract the exact meanings of the words. Recently, this is what can be called as central research area, however non-verbal information also plays an important role in the human communication. Let's just think about how many different meanings the 'really' word has, depending on how it is said [67].

In my thesis work I have investigated the speech from this non-verbal aspect, which has already been analysed by researchers, as well. For example, they found that joy, anger and fear have higher mean frequency and amplitude than the neutral speech has. Sadness has an opposite effect on frequency and amplitude with low variations regarding this values [68].

So that, I could also create a model, I had to measure not just the dynamic or the intensity of the sound in the time domain, but also the pitch in frequency domain. Then I could examine emotional effects of these values.

1.3.3 **The basic architecture of the task**

Figure 2 : The basic architecture of the task

As the Figure 2 above shows the first approach in my thesis work task consists of three main development projects. To begin with, the user interface which is a runnable iOS mobile application developed mainly with the help of swift programing language. Right at that moment, when the application is launched, a video stream turns on via the built-in camera of the device. This live stream appears on the screen of the device, and the application keeps getting samples of it. These samples that is pictures are pre-processed in the application, which means the faces are detected and transformed the pictures to an appropriate size. These pre-processed image appears in the lower part of the application meanwhile they are transmitted to the web application server, which is developed by Django technology.

After that comes the next step, which is represented as the second process in Figure 2. During this process, the server loads the convolutional neural network (CNN) model, which is taught by the database, having been labelled with emotions. After that the weights, having been created during the pre-learning, got loaded. I predict the probability of the emotions appearing by accident in the picture. The network is capable of recognising six basic emotions, these are the following ones: surprise, anger, disgust, fear, happiness, sadness.

In the last step the web application sends these probabilistic values to the mobile application. Then these values will appear with animation on the screen of the application. The strength of the incoming sound is also visualized in a real time, but it does not have any effect on the state of the emotion levels. It will be the feature of the following, enhanced application.

1.3.4 Enhanced architecture without web server

Since this September, with the new version of iOS operation system there is a chance to integrate trained machine learning models into the application. CoreML is the name of this framework which enables the classification with a neural network model directly in the application.

Apple provides some popular models, which are already in an iOS application compatible i.e. mlmodel format. In addition, there is an open source python library, called core ML Tools, so that the self-developed and trained models could be converted to this format.

In the following Figure it can be seen that in this new architecture the task only consists of two main components. Firstly, the model, where I have used not only the same Convolutional Neural Network (CNN) model as in the previous case, but I have implemented a new Recurrent Neural Network (RNN) model, as well. This new model is going to be responsible for the prediction of the emotional levels from sounds.

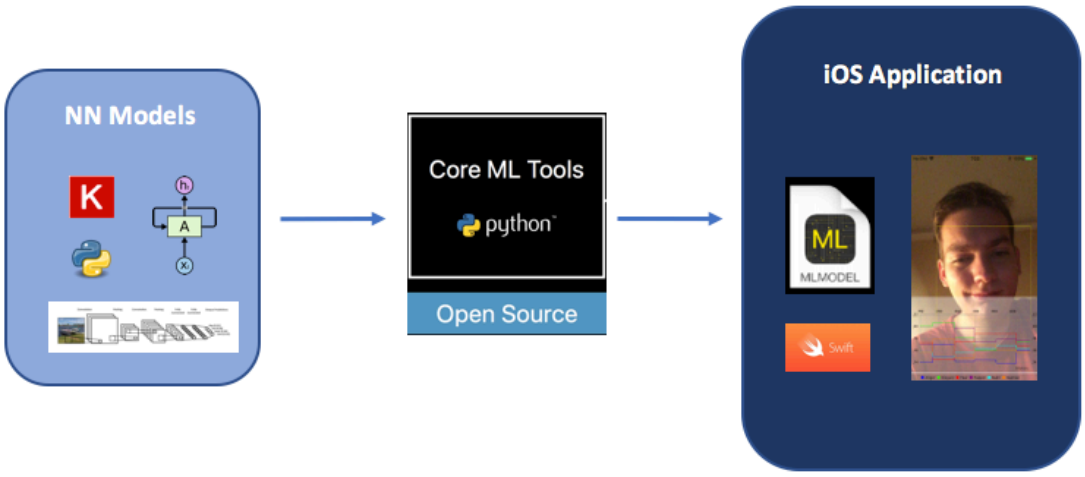

Figure 3 : Enhanced architecture without web server

Between the two components there is the Core ML Tools python library, which is responsible for the conversations from the Keras models to mlmodel formats. These

outputs can be dragged easily and dropped to the iOS project. Having processed the input video stream these models are called like a function by the application [46].

2 Deep learning with neural networks

2.1 Machine Learning

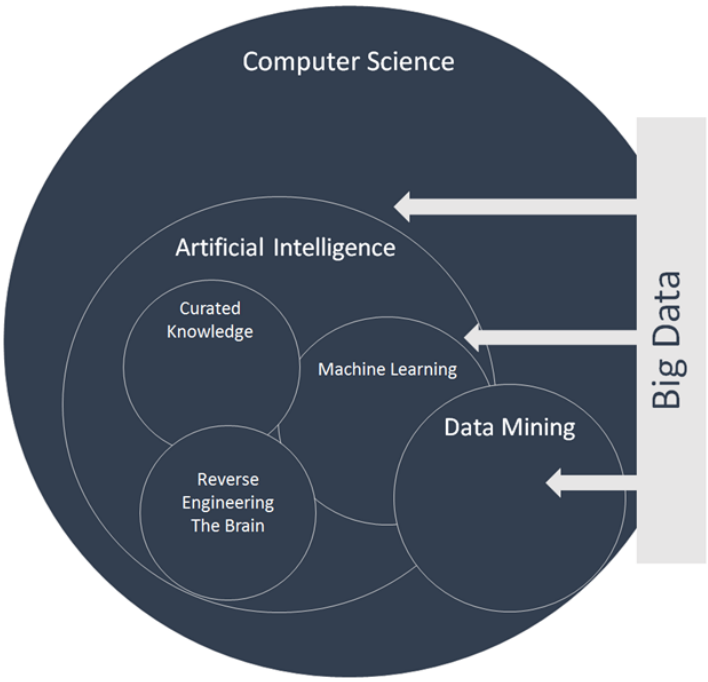

Figure 4 : The relationship of the machine learning [6]

As the Figure 4 shows machine learning is a core subarea of artificial intelligence. It is defined by Arthur Samuel in 1959 as the "field of study that gives computers the ability to learn without being explicitly programmed".

It has common features with the area of the data mining, in both cases the purpose is to find patterns, correlations and deeper relationships within large data sets to predict outcomes [7].

As I have already mentioned it above, machine learning seems to be a way of "programming by example." Machine learning has got a great advantage, compared to direct programming, which is the fact, that we can get a lot more precise and accurate results with it. It is obvious, as the algorithms in machine learning are data driven, thus they deal with large amounts of data. Moreover, human beings tend to be influenced by impressions or they just take into consideration only a few examples [11].

Three types of machine learning can be distinguished, they are the following ones: unsupervised learning, reinforcement learning, supervised learning.

When the model is trained by a set of inputs, so labelled examples are not available, we talk about unsupervised learning. Just to mention an example, the self-organizing map algorithm, in which we would like to make groups, in other words clusters, with the help of neural network.

In case of reinforcement learning the algorithm reacts to the observation of the environment. The actions have an influence on the surroundings, and this process generates a feedback, which is essential for the operation of the learning algorithm.

In my case the classification of the videos is based on emotions, which is a typical supervised learning problem. It is represented in the Figure 5 below.

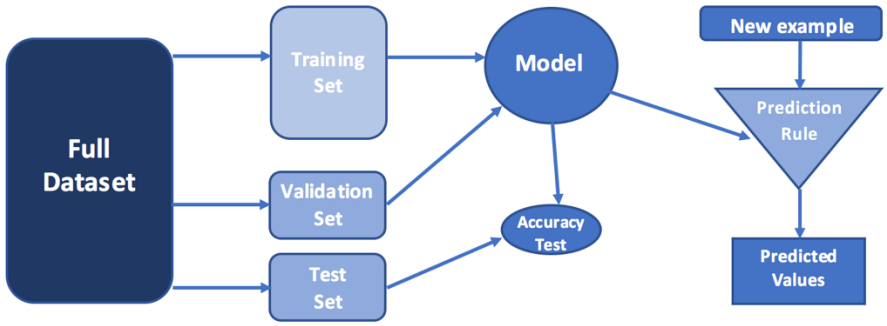

Figure 5 : Typical learning problem

The full dataset includes labelled data, which means that a target value is added to each examples. The dataset is divided into three parts: Training set, test set, validation set.

The teaching of the model starts with the training set. In my case, the model will be a Convolutional Neural Network or a Recurrent Neural Network. The validation set has an important role in the process of training. With the help of it, we can control and change the behavior of our model in the given cases, in this way we can avoid the overfitting.

After having accomplished our model, we can examine its accuracy with the help of test set. If the model is accepted, the new cases can be predicted by a system of rules, based on the model [11].

2.2 Affective Computing

Affective computing is a machine learning inspirited researching area in the intersection of the computer science, psychology, and cognitive science, which has been determined by R.W. Picard as the following way:

"Affective Computing is computing that relates to, arises from, or deliberately influences emotion or other affective phenomena." (Picard, MIT Press 1997)

The aim is the developing of systems and devices that can recognize, interpret, process, and simulate human affects and get the ability to have emotions.

The topic of my thesis work also deals with the fundaments of such a system, more precisely it is about an application, which is able to identify the emotions of the user. It is only about the fundaments, as the Affecting computing involves some reaction to the appropriate feelings. It can be achieved in use cases later on, which I will mention at the end of my thesis work [5].

2.3 Neural networks

"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs."

("Neural Network Primer: Part I" by Maureen Caudill, AI Expert, Feb. 1989)

2.3.1 **Biological motivations**

The fundaments of the neural networks were lied down about a hundred years ago and they can be found in neurobiological studies. It is such complicated field of study that the scientists spent several decades examining the operation of the neural system [8].

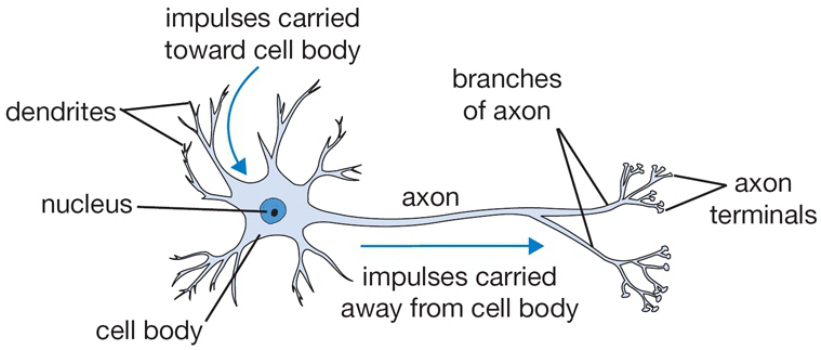

Figure 6 : Neuron

About 86 billion neurons can be found in the human nervous system. They are the basic computational unit of the brain. They are attached to several synapses. We can see a cartoon drawing of a biological neuron in the Figure 6. The input signals come from the dendrites and the neuron transfers them to output signals via its axon. The axon has got another important role: it connects via synapses to dendrites of other neurons. The dendrites carry the signal to the cell body where they all get summed. If the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon [15].

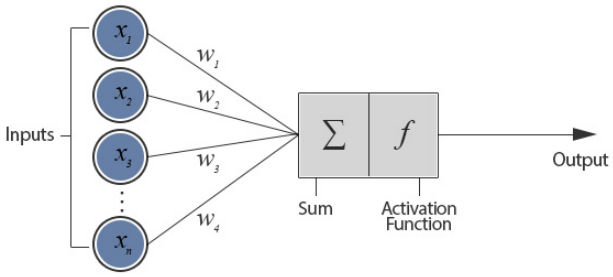

Figure 7 : Perceptron [13]

In 1943 McCulloch and Pitts invented the first artificial model for biological neurons using simple binary threshold functions. After that it took more than a decade for Rosenblatt to invent the perceptron, which is a mathematical model of a biological neuron. It can be seen in the Figure 7. In case of an actual neuron the dendrite receives electrical signals via the axons of other neurons. It is a bit different in a perceptron, whereas the electrical signals are represented as numerical values. The synapse is a process, in which the amount of the electrical signals is modulated. The perceptron emphasizes it by multiplying each input value by a value called the weight.

An actual neuron fires an output signal only when the total strength of the input signals exceeds a certain threshold. The firing of the neuron is modelled in a perceptron by calculating the weighted sum of the inputs to represent the total strength of the input signals, and applying an activation function on the sum to determine its output. As in biological neural networks, this output is fed to other perceptrons [10].

2.3.2 Theoretical background

2.3.2.1 Layered organisation

The network is built upon perceptrons, which can be seen in the Figure 7. As the following Figure 8 shows, these perceptrons are connected to an acyclic graph, so the outputs of each ones can become inputs of others.

The examples arrive to the network via the input layer, which communicates with one or more hidden layers. Here happen the weighted summing operations and the applying of the activation functions. Then the hidden layers are linked to an output layer where the predicted value is the output [12].

As it can be seen in the Figure 8, we don't have any feedbacks, because it would cause an infinite loop of the network. In most cases we use fully-connected layers in which neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connections [9].

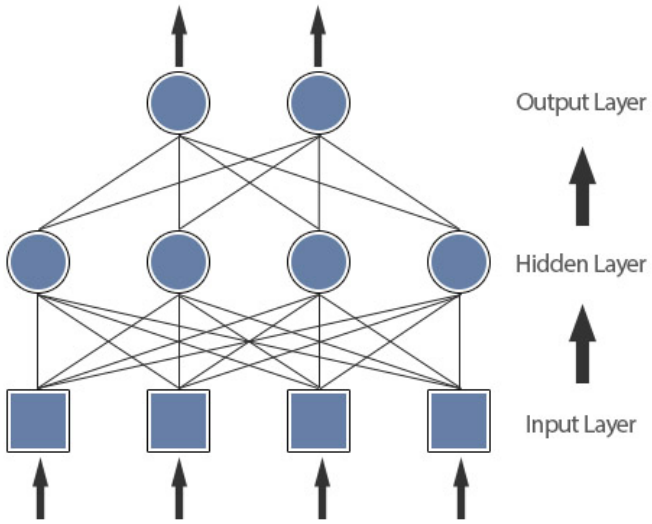

Figure 8 : Fully-connected neural network [13]

2.3.2.2 Linear separability

If a function can be separated with a straight line on the graph, that is linearly separable and can be modelled without the implementation of a hidden layer in the neural network. In the following Figure 9 a chart of the OR function can be seen, which is a great example for the linear separability. The right side of the Figure 9 proves that this function can be implemented with just one input and one output layer [13].

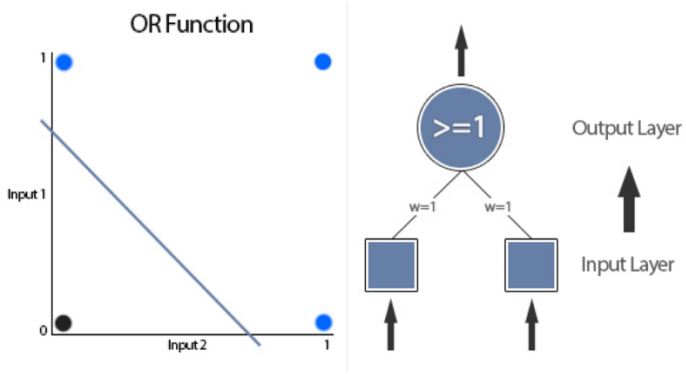

Figure 9 : Neural Network implementation of the OR function [13]

The following Figure 10 depicts that in the case of XOR function, which cannot be separated with a single line, a hidden layer is necessary. Without this layer, this function cannot be feasible with a neural network model [13].

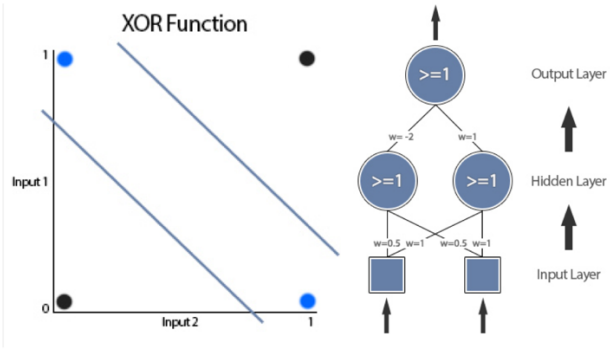

Figure 10 : Neural Network implementation of the XOR function [13]

Given the fact, that in case of even such a simple function as the XOR, extra layers are needed, it must be admitted that in most cases of the everyday life deep networks are used.

2.3.2.3 Activation functions

If we want the network to be trainable, we need differentiable i.e. continuous activation function, because we use gradient descend (can be seen later on) method to train. The simple threshold, which could have been seen above, is not such a function, therefore the sigmoid function is used in most cases, which can be expressed in the following way:

$$sigm(x) = \frac{1}{1 + e^{-x}}$$

This function has several favourable features. On the left side of the following image it can be seen clearly, how the sigmoid function's value ($sigm(x)$) approaches 1 as the input value (x) becomes highly positive, and how it approaches 0 as x becomes highly negative [18].

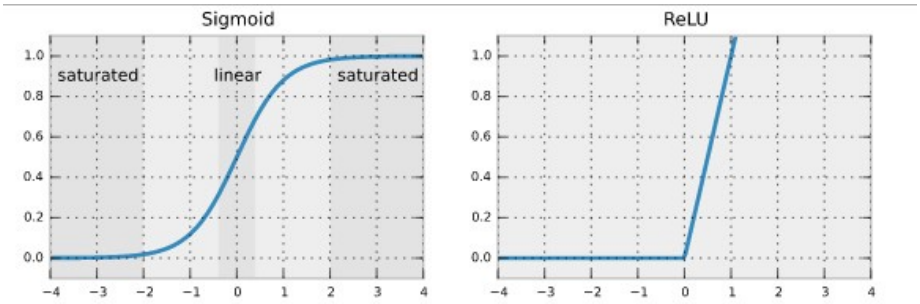

Figure 11 : Activation functions [14]

Another frequently used activation function, mainly in case of the convolutional networks, is the RELU (Rectified Linear Unit). This function can be computed in the following way:

$$RELU(x) = \max(0, x)$$

As it can be seen in the right side of the Figure 11, the output of this activation function is simply cuts off at zero. I have used the RELU function in my thesis work, because it accelerates the convergence of stochastic gradient descent (later on), because of its linear and non-saturating form [9].

I would like to mention here the tangent hyperbolic activation function as well, which plays an important role especially in case of recurrent networks. This can be expressed with the following formula:

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1 = 2 * sigm(2x) - 1$$

As the following Figure shows that the $\tanh(x)$ has non-linear, sigmoid alike characteristics. The difference here can be found in the output set, because it transposes a real-valued number to range between -1 and 1 [9].

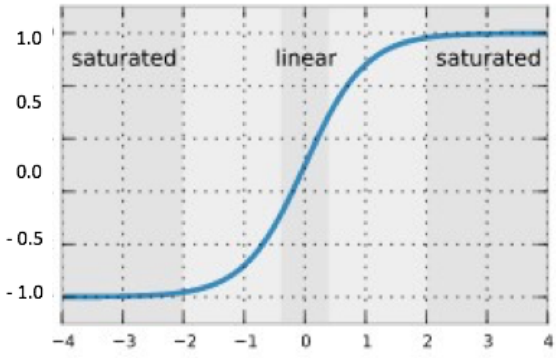

Figure 12 : Tangent hyperbolic activation [14]

2.3.2.4 Cost function

The cost function shows the error of the model, i.e. how accurate the predictions of the model for a given set of parameters are. The slope of this function curve helps us to make the model more accurate, because the gradients of this function tell us how to change our weight parameters. Several types of functions can be used, the most commonly used are the following ones:

$$C = (O_{desired} - O_{observed})^2$$

Where the $O_{desired}$ is the expected output value of the set of input value and the $O_{observed}$ is the output which we get in the reality [18].

2.3.2.5 Gradient descend and backpropagation

Widrow and Hoff were the ones, who introduced the first network trained by a gradient descent rule in 1960. This can be used to minimize a cost function with the help of the negative gradient of the function, which points into the direction in which the levels are decreasing the fastest. In machine learning we calculated gradients from the weight in the neural network, because we wanted to update them. This can be imagined as the following Figure 13 shows; we start at the higher point of the surface of the cost function. We calculate the negative gradient, which shows the direction where we could step down. We continue this process iteratively and finally we reach the bottom of this surface, i.e. the local minimum [16][8].

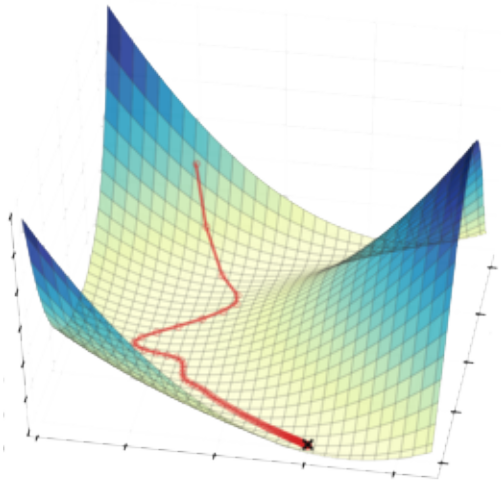

Figure 13 : Gradient descend [19]

The updating of the weights can be described as the following way:

$$w_{new} = w + \alpha * \frac{dC}{dw}$$

Where w_{new} is the the actual w weight after the updated process, α is the learning rate, which influenced the size of the weight refresh. As I mentioned above the derivated cost function shows the direction, in which we would like to change the parameter. This can be calculated with the help of the backpropagation process, which is illustrated in the following Figure 14. In this case the estimated error (cost function) is propagated back to each weights with the help of the chain value. This schema was proposed by Rosenblatt in 1961 [8][20].

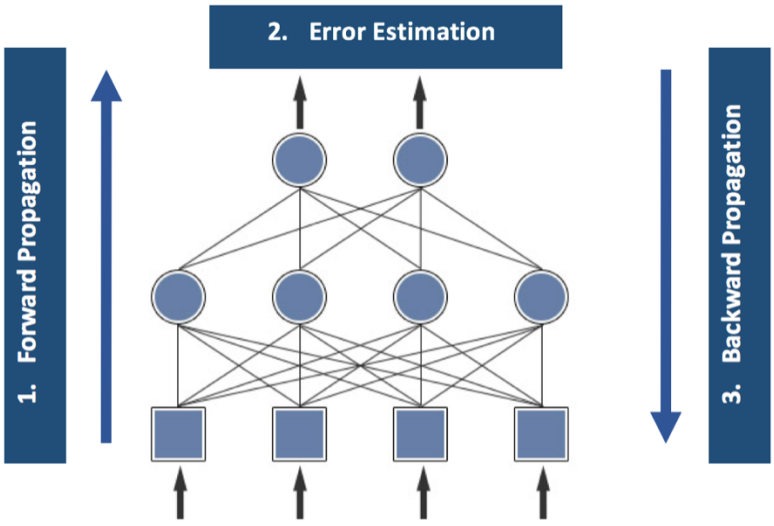

Figure 14 : Forward and backward propagation [13]

Depending on how much sample we use to compute gradient, three variants of the gradient descend can be distinguished. Firstly, the batch gradient descend, where we perform just one update in every epoch, which are calculated from the whole dataset. Epoch is the process when all the training examples pass forward and backward once. Secondly, the stochastic gradient descend unlike the previous one performs update for each training sample. Finally, the mini-batch gradient descend, which is the most recently used when training neural network. In this case the predefined number of training samples are used in the update process. Typically, the mini-batch sizes range between 50 and 100 [23].

2.4 Deep learning

Today, the most researched Artificial Neural Network based area is the Deep Learning. The Deep Learning expression refers to a deep neural network model, which can be determined as the composition of multiple non-linear processing layers. This model can be trainable and it is able to learn representations of data with multiple levels of abstraction. Therefore, the main strength of these networks is that it can extract meaningful features automatically from multimedia contents. That is why the deep learning nets have achieved a great success in processing images, video, speech and audio [24].

2.4.1 Avoiding of the overfitting

Deep neural networks usually consist of more than a hundred layers, which means there are extremely complex models. Thus, overfitting is expected to come up at times, when the model overreacts the variation of the training data and the prediction performance for the new examples will be poor. Fortunately, there are some solutions to avoid the overfitting, such as: dropout and batch normalization.

2.4.1.1 Dropout

Dropout is a technique for reducing the overfitting. The key concept is to drop perceptrons and their connections randomly out of the neural network. This prevents units from fitting too much to the training data [22].

2.4.1.2 Batch Normalization

As the data flows through a deep network, the weights and parameters adjust to those values. This means we can't use big learning rate because of the variance of the values. In the case of batch normalization, we normalize the data after every layer. It acts not only a regularizer role, but also we can accelerate the learning time with a bigger learning rate constant [21].

2.4.2 Convolutional Neural Networks

The Convolutional Neural Network is kind of deep network, which has a particular feature to contain not fully connected layers too.

A typical network, which is used for image classification can be seen in the following Figure 15:

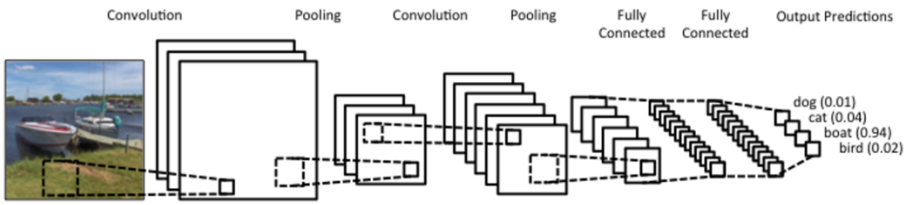

Figure 15 : Convolutional Neural Network [25]

In case of image classification, in the first layer the Convolutional Neural Network learn to recognize simple forms such as arcs and edges. After that the model uses these simple shapes to determine higher-level features, for example emotional facial shapes and any other more complex objects [25].

Three main types of layers are used to build up this network, these are the Convolutional Layer, Pooling Layer, and Fully-Connected Layer.

2.4.2.1 Convolutional Layer

The first layer in a Convolutional Neural Network, after which it was named, is the not fully connected Convolutional Layer. As it can be seen in the left side of the Figure 15, the key object is a square-shaped filter, which slides across all the areas of the input image. As the image represents, this filter is also an array of numbers.

As the filter is sliding, or convolving, around the input image, we get a single number from the summed multiplication of the values in the filter and the original pixel

representation numbers of the image. Thus, we get as many computed numbers as many unique location is available for the filter. Finally an array, which is the feature map, and which we get at the end of this process. Obviously, the filter size is always smaller than the size of the input image and the filter depth is equal. We almost always represented a pixel of the image with the red blue and green value, that is why the depth is three [9].

2.4.2.2 Pooling Layer

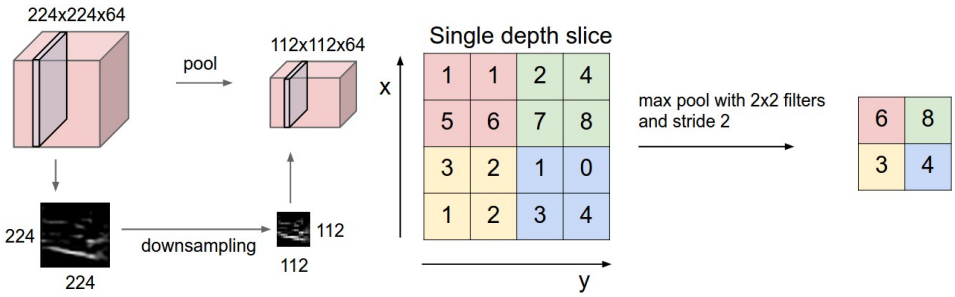

Figure 16 : Max Pooling [9]

In general, the Convolutional Layer is followed by a Pooling Layer in a Convolutional Network architecture. As it can be seen in the left side of the Figure 16, the Pooling has a kind of downsampler function, thus the amount of parameters and computation are reduced in the network. Hence, this layer also controls and reduces the probability of the overfitting.

The most common used max-pooling algorithm is depicted in the right side of the previous Figure 16. A filter with 2x2 size was applied on a single depth slice and was slided twice in all directions of the representation. As the name of the algorithm suggests, the maximum value was selected from the filter frame. In this case 75% of the data was eliminated and the depth dimension remained unchanged [9].

2.4.2.3 Fully-Connected layers

Fully-Connected layers are generally located at the end of the Convolutional Neural Network. After the feature extractions, this layer is responsible for the classification, so the prediction rules are constructed here. There are several possible layouts of the hidden layers, but the output layer consists of as many units as many classes we would like to predict.

2.4.3 **Famous Convolution Neural Networks**

The first successfully used Convolutional Neural Network was the LeNet architecture, which was developed by Yann LeCun in 1990's. This network was able to classify handwritten characters from zip codes or bank checks with minimal pre-processing [9] [26].

Nowadays, the most famous architectures turn up on a prestigious competition, called ImageNet Large Scale Visual Recognition Challenge (ILSVCR). The participants have to detect hundreds of object categories and classify millions of images. The challenge has been organized annually since 2010 [27].

The first popular Convolutional Network on this challenge was the AlexNet in 2012, developed by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton. It has many common features with the LaNet, but AlexNet was deeper and more complex with its 60 million parameters and 650,000 neurons [28].

In 2014, the ILSVRC winner network was the GoogLeNet from Google. It was a relatively computational and memory efficient Convolutional Network, based on a great achievement: the "Inception" block, which is depicted in the following Figure 17. This can reduce the number of the parameters in the network and the parallel computation [29].

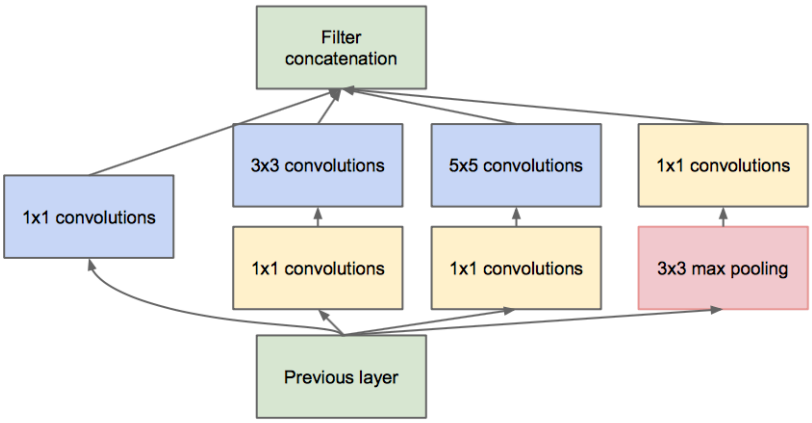

Figure 17 : The "Inception" module [29]

I have used the Inception-V3 network in my thesis, which is one of the enhanced versions of GoogLeNet. This network can be seen in the following Figure 18, which is built from nine Inception module.

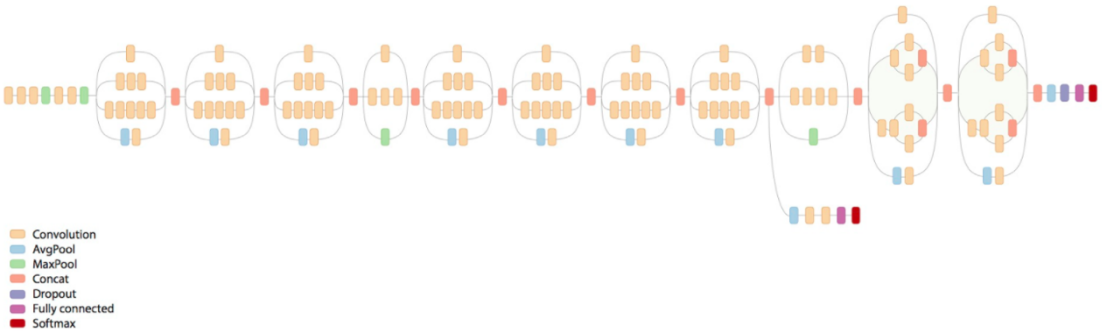

Figure 18 : Schematic diagram of Inception-V3 [30]

Finally, I would like to mention two more Convolutional Neural networks, which I have not investigated in detail, but they are considered to be quite important. The first one is the ResNet50, which was the winner of ILSVRC 2015. This network was published by Microsoft research and its speciality is that it doesn't contain fully connected layers. The other one is the VGG16, which was developed by researchers at University of Oxford, where the main goal was the simplification [9].

2.4.4 **Recurrent Neural Networks**

In case of human way of thinking, our prior knowledge makes everything more understandable. For example, let's see a word, which could have special meanings in different contexts. This function couldn't be realised by a simple neural network, because they are not able to store the information from a previous state. To do this, as shown in the following figure, we have to use feedback-loops-contained Recurrent Neural Networks. This sequence model achieves great performance on important tasks such as language modelling, speech recognition and machine translation.

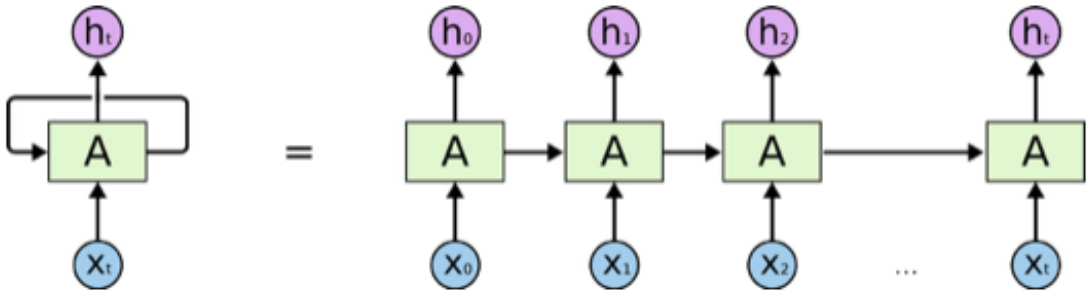

Figure 19 : Representation of the feedback loops in RNN [48]

As you can see the decision at a given time in the recurrent network affects the decision at the next moment. Thus, this network has two sources of input, the present and

the recent past, which together determine the output. So we can say the Recurrent Networks act like they have memories [47].

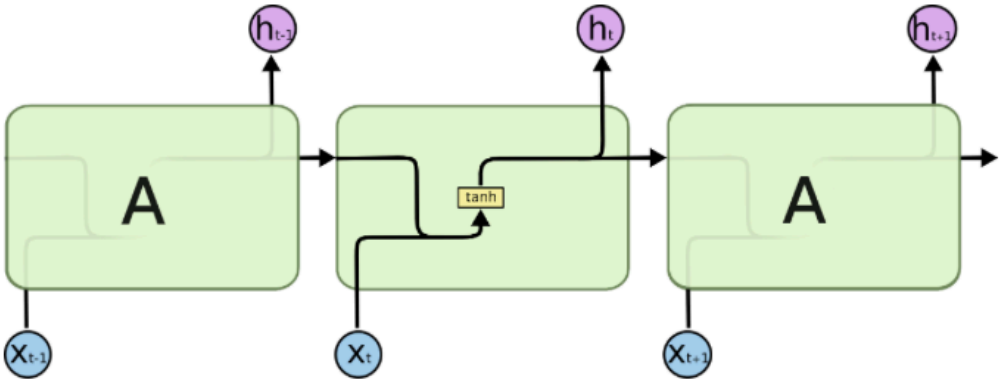

Figure 20 : Classical RNN [48]

In the Figure above, a classical Recurrent Neural Network is depicted, which can be described with the following mathematical formula:

$$h_t^l = \tanh\left(T_{n,n}X_t + T_{n,n}h_{t-1}^l\right), where\ X_t = h_t^{l-1}$$

Here $T_{n,n}$ is an affine transform ($R^n \rightarrow R^n$) and $h_t^l \in R^n$ is a hidden state in layer l in timestamp t [49].

However, the classical Recurrent Neural Networks may look efficient, but in practice they are not exactly used in this way. The reason for this is the long-term dependency problem. Long-term dependency means, when the desired output at a given time depends on inputs presented at a lot earlier time. The problem with the simple Recurrent Networks, especially because of the nature of the gradient descending process, is that they are able to learn short-term structures, but they can't learn long-term system behaviours [48] [50].

2.4.4.1 Long Short Term Memory networks (LSTMs)

The first significant solution to the long-term dependencies problem is the LSTM networks, which can solve complex, artificial long time lag tasks. This network architecture was introduced by Hochreiter and Schmidhuber in 1997 [51].

LSTMs also have chain like structure, but in this case instead of neuron layers, here we can talk about memory blocks. As the following Figure 21 depicts, in a block

there are four layers, with sigmoid or hyperbolic tangent activation function, interacting in a very special way [48].

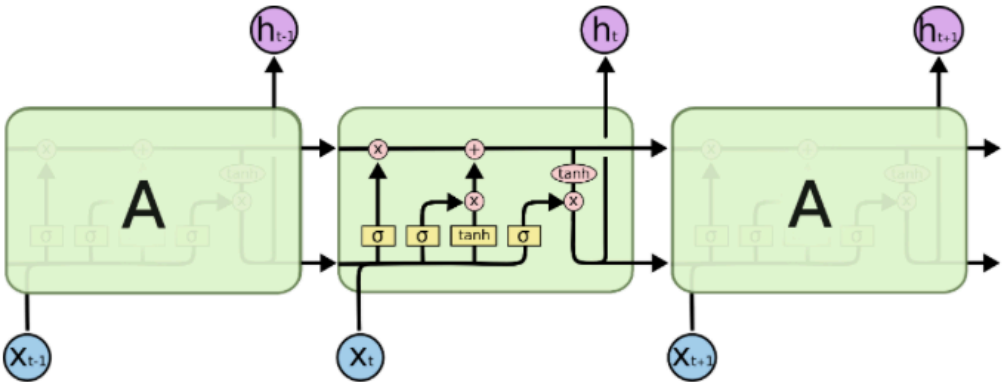

Figure 21 : LSTM as a chain like structure [48]

The long-term memory can be called as cell state in other words, which is a vector of memory cells ($C_t^l \in R^n$). It is symbolised with a horizontal line running through the top of the following Figure 22. This cell state is controlled by structures called gates. Gates conditionally decide which information can be let through, continuously changing the cell state [52].

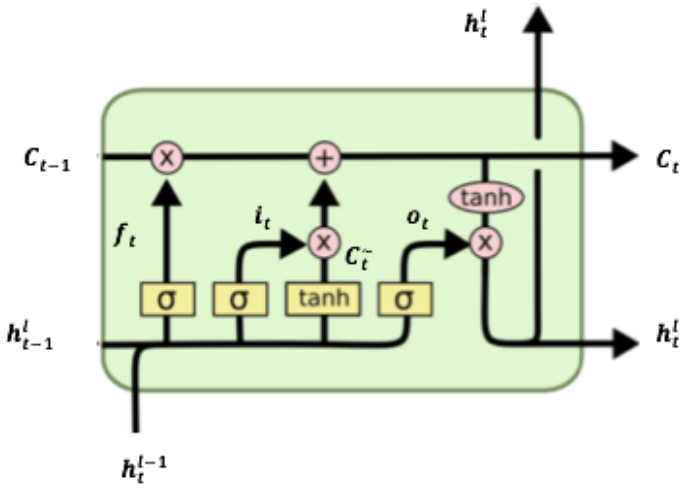

Figure 22 : The LSTM block [48]

First comes the forget gate (f_t) in a block, which conditionally decides what information we are going to preserve from the previous cell state. The decision can be expressed with the following formula using a layer with sigmoid activation function:

$$f_t = sigm\left(T_{n,n}^f h_t^{l-1} + T_{n,n}^f h_{t-1}^l\right)$$

The next decision regards to the new information we are going to store in the cell state. The input gate (i_t) is responsible for this process.

$$i_t = sigm\left(T^i_{n,n} h^{l-1}_t + T^i_{n,n} h^l_{t-1}\right)$$

The hyperbolic tangent layer will be the unit which can create the vector of the new potential values, with which it is possible to change the cell state. This vector is signed as $C^{\sim}_t$ and given with the following formula:

$$C^{\sim}_t = sigm\left(T^C_{n,n} h^{l-1}_t + T^C_{n,n} h^l_{t-1}\right)$$

Having arrived at this point we have calculated and decided everything, which is needed to update the old cell state (C_{t-1}). We only need to forget about the previous state by multiplying the old state by f_t and we update the state as you can see in the following expression.

$$C_t = f_t * C_{t-1} + i_t * C^{\sim}_t$$

At last but not least, the output value of the block has to be decided. The output gate (o_t) is the key in this process, which is similar to other gates, decides what parts of the cell memory vector we would like to output.

$$o_t = sigm\left(T^o_{n,n} h^{l-1}_t + T^o_{n,n} h^l_{t-1}\right)$$

Then we set the cell state values between -1 and 1 by utilizing the nature of the hyperbolic tangent activation. Finally, the output of the LSTM block is given by a simple multiplication [53][48]:

$$h^l_t = o_t * \tanh(C_t)$$

2.4.4.2 Other Recurrent Neural Networks

Besides of the LSTMs, there are more Recurrent Neural Networks because this is an area of active research. The literature mentions the LSTM variants in general. Depending on the application area, each of them has its pros and cons.

Such a popular variant of them was introduced by Gers and Schmidhuber. It is a simple LSTM with peephole connections, which allow all gates to inspect directly the current cell state [54].

Another variant of the LSTM, which I have tried in my thesis work as well, is the Gated Recurrent Unit (GRU). It was introduced by Cho et al. in 2014. The following Figure shows an architecture about this network:

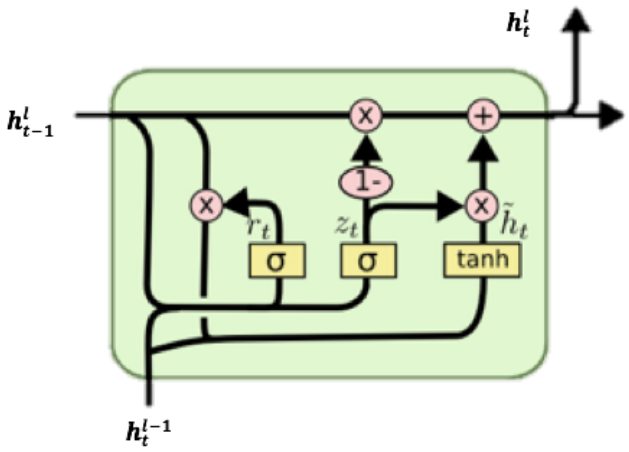

Figure 23 : The GRU architecture [48]

As you can see it in the Figure 23, similarly to the LSTM unit, the GRU has gating units that modulate the flow of information inside the block. However, here separate memory cell is not used. The newly computed state is similar to the LSTM unit, but GRU does not have any mechanism to control the degree, it just exposes the full hidden content without any control. Thus, the process can be expressed as the following way:

$$z_t = sigm\left(T_{n,n}^z h_t^{l-1} + T_{n,n}^z h_{t-1}^l\right)$$

$$r_t = sigm\left(T_{n,n}^r h_t^{l-1} + T_{n,n}^r h_{t-1}^l\right)$$

$$h_t^{\sim} = \tanh\left(T_{n,n}^h \left(r_t * h_{t-1}^l\right) + T_{n,n}^h h_t^{l-1}\right)$$

$$h_t^l = (1 - z_t) * h_{t-1}^l + z_t * h_t^{\sim}$$

Thanks to this simpler and computationally more efficient architecture, the training process became faster while the model performance remained nearly similar. That is why this relatively new variant is becoming more and more popular [48][55].

2.5 Technologies of implementation

There are several implementation technologies with different pros and cons, which can be used for developing neural network and deep neural network models. Most of them are open source. Just to mention some of them, the Microsoft Cognitive Toolkit (CNTK) or the Torch framework, which is based on LUA programming language and

used by Facebook. It was an important aspect for me that the deep learning library can be used with python programming language, because I could utilize my former studies and I considered it to be a modern, efficient and fast solution for machine learning problems. These technologies, for example the Theano, TensorFlow and Keras, which I will present in more detail in the following sections.

2.5.1 **TensorFlow**

This library was developed by the Google Brain Team originally for internal use, but in 2015 it was released as open source. It uses data flow graphs for numerical computing, where the nodes in the graph represent mathematical operations and the graph edges are the data arrays, i.e. tensors. The biggest advantage is the ability to deploy computation on multiple CPUs and GPUs parallel [31].

2.5.2 **Theano**

Theano is an also open source python based library, specialised in optimizing and evaluating of high-level mathematical expressions with multi-dimensional arrays. The developers, University of Montreal and Yoshua Bengio's team, wanted to achieve tight integration with other python packages, so that it would be easier to use with former python knowledge. Additionally, this is also important that Theano would perform computations fast enough due to the GPU support [32].

2.5.3 **Keras**

Keras is a high-level also open source library, which can run on top of either TensorFlow or Theano. Its primary author is François Chollet, a Google engineer. With its several great features, it enables fast experimentation and makes prototypes quickly.

In Keras, the pre-trained models can be imported and extended easily with new modules. A module is a standalone block, which can be neural layers, cost functions, optimizers, initialization schemes, activation functions and regularization schemes. These ones and the GPU support are also able to train complex Convolutional and Neural Neural Networks with this framework. As I have also utilized this library to create my model, I will represent the operation of it in the next chapter in more details [33].

3 Implementation of the learning model

3.1 Initial database

The Ryerson Multimedia Research Lab from Toronto made an emotion video database available for my research purpose. This contains 720 video samples from eight human subjects, speaking six different languages. In the 3-6 seconds-length video clips six basic human emotions are expressed: anger, disgust, fear, happiness, sadness, and surprise. All the video files are in AVI format, with a total size of 4.2GB [17].

3.2 Image extract from video

After the data collection comes the pre-processing step, where I transformed the raw multimedia data into the appropriate format, required by the models. First, I extracted the image frames from the video. For this purpose, I developed the following three functions: getVideoSamples, cropFaceFromImage and resizeImage.

3.2.1 **Video sampling**

In order to use the video for unsupervised learning, I had to sample it to get frames i.e. still images. The getVideoSamples is such a function, in which the number of the frames per second can be added as an input parameter.

3.2.2 **Face detection**

I wanted to focus on emotion detection from faces, because they show the feelings most accurately. The cropFaceFromImage function performs not only the face detection but also the cropping from image.

I used Haar feature-based cascade classifiers for the face detection, which was presented in 2001 by Paul Viola and Michael Jones. This method can process images extremely rapidly and achieve high detection rates. A typical cascade architecture can be seen in the following Figure 24, which allows background regions of the image to be quickly discarded while spending more computation on promising face-like regions [34].

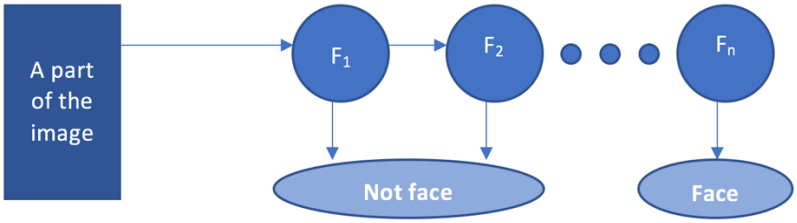

Figure 24 : Cascade architecture

The classifier F functions are trained with lots of positive images, where faces are contained, and negative ones, without any faces. In the following Figure 25 the extraction of the Haar features can be seen. These features are calculated with an easy subtraction of the sum of pixels under white rectangle and the sum of pixels under black rectangle [35].

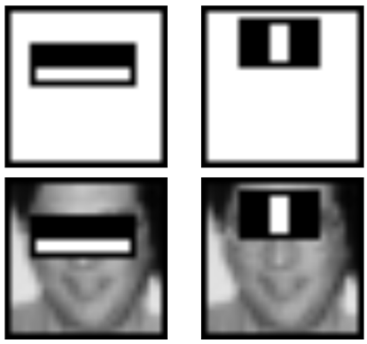

Figure 25 : Haar features [34]

The following code snippet presents a face detection process with the openCV python package. In the first step I loaded the appropriate classifier, which is stored in xml format. Then the detectMultiScale function finds faces in the image with the help of this classifier. In the last steps the cropping operation can be seen.

```
face_cascade = cv2.CascadeClassifier("haarfrontalFeatures.xml")
faces = face_cascade.detectMultiScale(image, 1.3, 5)
for (p,q,r,s) in faces:
    faceCropped = image[q:q+s, p:p+r]
    …
```

3.2.3 **Image resizing**

Perhaps, the resizeImage is my simplest pre-processing function because it has only one input parameter, which is the size of the required image. I used 299 pixels for

width and the same quantity for height of the image size to train my Convolutional Neural Network models.

3.3 Sound extraction

After the image extraction I converted the sound of the video to such a format, which is mathematically processable. My first aim was not to understand the meaning of the words, but to investigate the effect of the amplitude and the frequency changes on the emotions. To do that I had to extract these two pieces of information from the raw media file.

3.3.1 Converting video to audio

I have used the Ffmpeg multimedia framework to extract the sound track from the video. This framework has a command line tool for the conversations, so I called this similar to the following bash command for each video files:

```
ffmpeg -i videofile -ab 160k -ac 2 -ar 44100 -vn outputname.wav
```

From this command can be seen I have used an uncompressed two-channel vaw audio format sampled 44,100 times per second with 16 bits per sample [57].

3.3.2 Extraction of the amplitude and frequency values

To get amplitude values from the wav file I had to use the wavfile library from the scipy software package. The read function returns the sample rate (in samples/sec) and 32-bit data from a WAV file [56]. After that I made the length of the series unified.

Besides the amplitudes I have also investigated the frequency values. Thus, the sound samples in the time domain were converted to discrete frequency values. This process is the Discrete Fourier Transform (DFT), which can be defined as follows:

$$F(k) = \sum_{n=1}^{N} f_n \, e^{-2\pi jkn/N}$$

Where $F(k)$ are the Fourier Coefficients i.e. the frequency values, f_n is the given time series with N samples. The whole transformation ($1 \leq k \leq N$) requires $O(N^2)$, but exists a fast Fourier transformation (FFT) algorithm that can be completed in $O(N \, log \, N)$ operations.

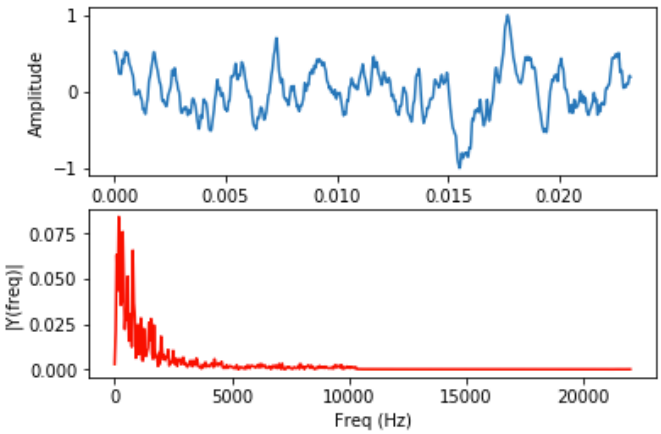

Figure 26 : Amplitude and frequency values

The Figure 26 shows that I have calculated frequency values after each 1024 samples. The sample frequency is 44100 Hz, thus I have investigated the frequency changes per 25 millisecond [58].

3.4 Deep network programming

As I mentioned it above, I used the Keras python based machine learning package. In the following sections I will present two kinds of approaches. One of them is when I constructed a totally new model and the other one is, when I applied a pre-trained Convolutional Neural Network.

3.4.1 Creating a new CNN model

The following code snippet shows how a basic Convolutional Neural Network in Keras can be built up:

```
model = Sequential()
model.add(Convolution2D(32,3,3,activation='relu',input_shape=(299,299,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(6, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='SGD',
              metrics=['accuracy'])

model.fit(X_train, Y_train, batch_size=32, nb_epoch=10, verbose=1)
```

In the first row the Sequential() class, the simplest type of the core data structure is instantiated. Then a convolution layer is stacked on the empty model. I had to determine

the shape of the input, which is the size of the image representation array. As you can see I have used 299_X299 pixel images. Number 3 means the depth of the encoding, which is the three colour values at each pixel coordinates. In both cases I applied 32 filters with 3_X3 size and RELU activation functions for each perceptrons in the network. Now the output array shape is 297_X297_X32, which is called feature map. The reason for that is there are 88209 different locations that 3_X3 filter can fit on 299_X299 input image. This number of locations can be mapped to a 297_X297 array and every filter creates a new dimension that is why the depth is 32 [9].

After that I have added a Max-Pooling layer with 2_X2 pool size. This operation transformed the array shape to 148_X148_X32. This downscale layer almost halved the input in both spatial dimension, while the depth dimension remained at the original value.

As the Pooling layer is, the Dropout is also able to prevent the overfitting. It is not exactly a layer, rather than an operation on the previous layer. I used that twice in the previous code snippet with a parameter which determines the fraction of the units to drop.

Before I could stack the fully connected layers on the model, I applied a Flatten layer where the model dimensions were reduced, so the representation values became flat and the matrix shape changed from 148_X148_X32 to an array with 700928 values. At the end of the network two Dense layers, i.e. fully connected layers can be seen. The first one has 128 neurons with RELU activation function and the second one has 6 ones, because I wanted to classify six types of emotions. The softmax activation means that in this case I would like to get probability values as an output of the network.

This sequence model is depicted in the following Figure:

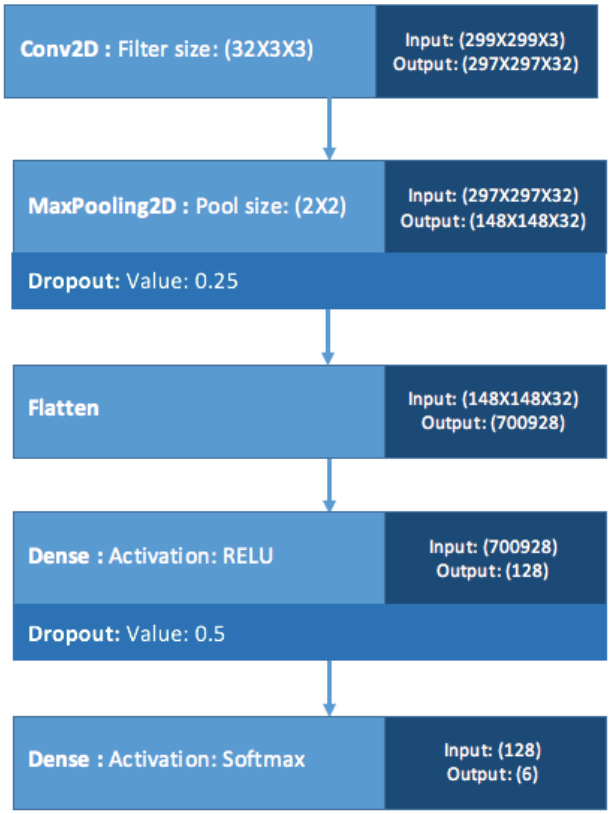

Figure 27 : The new CNN model

In the Compile method, the parameters of the learning can be determined. The first parameter defines the type of the loss function. I had to use one, which is good for the categorical classification. The next parameter refers to the optimization process, which I set to Stochastic Gradient Descend. The last one is the type of the metrics, which judges the performance of my model.

Finally, comes the most time-consuming process, where I fitted the model to my training image set. I have chosen 32 for the size of the batches and trained until 10 epochs. The verbose parameter sets only the logging on the console, so the format of the feedbacks [33].

3.4.2 **Using pre-trained CNN model**

In the following code snippet, I will present a model which is built on the pre-trained Inception-V3 model by Google. A schematic diagram of this model can be seen in the Figure 18.

```
base_model = InceptionV3(weights='imagenet', include_top=False)
```

```
for layer in base_model.layers[0:177]:
    layer.trainable = False
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(6, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)
```

In the first step, I loaded the base model with weights, which was trained on the ImageNet database. In my case I didn't want to use this model for object detection, thus I didn't include the top fully-connected layers. As you can see I stacked my custom Dense layers on the top of the network. In the for loop I configured the first 177 layers out of 311 to be untrainable, so the connection weights of these perceptrons in these layers cannot be trained when I fit my rebuilt model to the emotions [33].

3.4.3 **Creating an LSTM based RNN model**

First, I have created the first Long Short Term Memory based Recurrent Neural Network for time series. The following code snippet shows that I have started with a sequential class. Then I stacked three LSTM layers to make it capable of learning higher-level connection. In the first case, I have given the length of the series as an input shape parameter. The return_sequences parameter decides whether to return the last predicted output in the last timestamp or to return the full sequence in every timestamp.

```
model = Sequential()
model.add(LSTM(10, return_sequences=True, input_shape = (20000,1)))
model.add(LSTM(6, return_sequences=True))
model.add(LSTM(8))
model.add(Dense(6,activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='SGD',
              metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, batch_size=1, verbose=1)
```

The following Figure shows the layers in this model and the data dimensions in each layer.

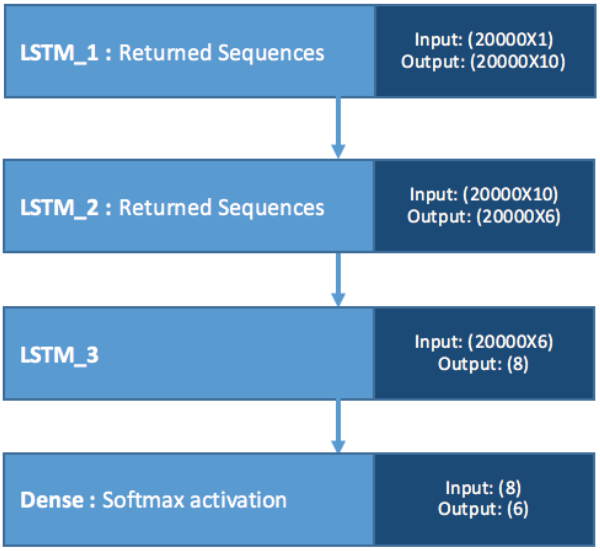

Figure 28 : RNN with LSTM layers

3.4.4 **Creating an RNN with GRU**

In this second approach instead of LSTM I have built up the model with Gated Recurrent Unit layers and trained on the extracted frequency values.

In this case the input shape at the first GRU layer is (20,1024), because I represent the amplitude values with 1024 values in the frequency range. The time domain signal was divided into 20 parts, that is why 20 frequency calculations are included in the series of 20 timestamps.

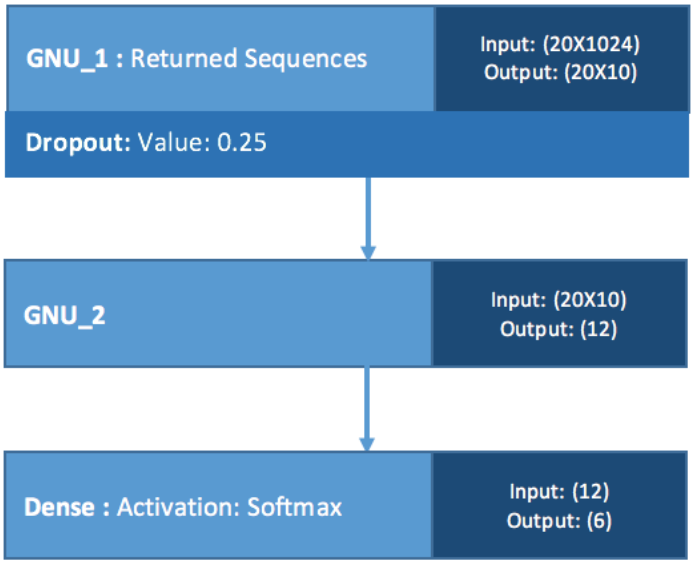

Figure 29 : RNN with GRU layers

3.4.5 **Save models**

After the learning process, I had to store my model and its weights, because the web server would load them to make predictions. The following code snippet shows the loading process.

```
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
model.save_weights("model.h5")
```

I have chosen JSON (JavaScript Object Notation) format to store my models. This is a human-readable text format which consists of key-value pairs. Besides that, I use this format in the communication between my mobile application and the web server. Then I had to store the trained weights, which can be seen in the last line.

3.4.6 **Converting to Core ML model**

So that I could use the trained model in my iOS application I had to convert it. I managed to do this with the following code line [59]:

```
coreml_model = coremltools.converters.keras.convert(keras_model)
```

3.4.7 **Training with validation dataset**

The Keras framework allows to use validation dataset during the training process. It is possible to specify as a function parameter, when the fit function is called. Or there is another option as I have used in the following codeline:

```
history = model.fit(x, y, validation_split=0.2, epochs=8, batch_size=64)
```

Here I specified the validation_split argument to 0.2, which means the validation data used will be the last 20% of the training data. The fit method always returns a History object. One of its attribute stores the training loss values and metrics values at successive epochs, as well as validation loss values and validation metrics values [60].

Three possible cases can be distinguished. First, when the validation and training error are also high, it is likely that we run into the trouble of underfitting. In case of overfitting the training error is relatively low, but the validation error is high. Finally, the ideal case is when the validation error is low, it is only slightly higher than the training error.

The following Figure 30 shows the loss values of the first Convolutional Neural Network model during 7 epochs.

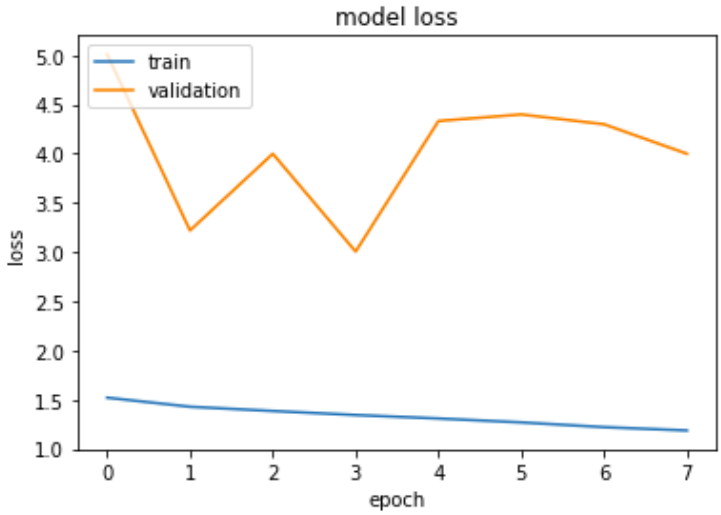

Figure 30 : Loss values during the base CNN model training

According to the Figure it seems my model suffers from the overfitting phenomena. The reason behind this can be the small number of training data or the simple model without any dropout layers. It also can be seen after the third epoch, the validation error is not decreasing, therefore it is not worth training the model after that.

The following Figure shows the same training with validation split, but now I have used the pre-trained, extended CNN.

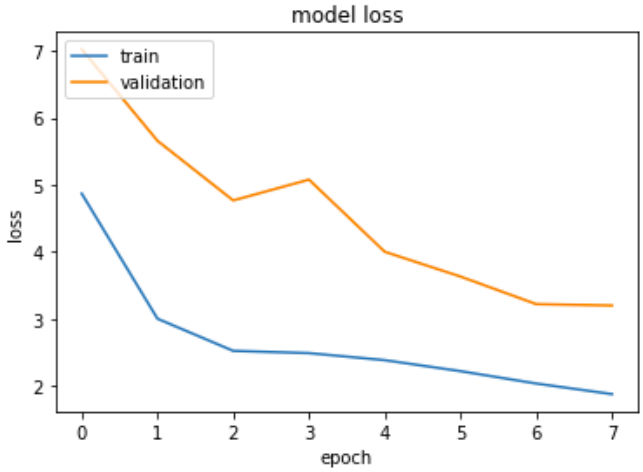

Figure 31 : : Loss values during the pre-trained CNN model retraining

In this case the curves of the validation and the training loss are closer to each other, which means I could reduce the overfitting. Thanks to the several Dropout and Pooling layers from the 314 layers.

3.5 Evaluation of the developed solutions

I recorded my own videos to evaluate my models. I pre-processed these labeled videos and I predicted probabilities to each of the emotions. Then I calculated logarithmic loss or cross-entropy value from these predictions and the desired outputs. This function can be determined in the following way:

$$L_{log}(Y,P) = -\frac{1}{N}\sum_{i=0}^{N-1}\sum_{k=0}^{K-1} y_{i,k} log p_{i,k}$$

Where Y means the encoded true labels for set of samples as a 1-of-K binary indicator matrix. And P is a matrix of probability estimates. I used this function in the following way: [36]

```
Y_predict = model.predict_proba(X_test)
score = log_loss(Y_test, Y_predict)
```

Here I called the log_loss function from the sklearn package. Regarding to my first CNN model I have got 3.81 as logarithmic loss value and in the second case I have got 2.98. As I expected the partly pre-trained model performs more accurate predictions. However, neither of these models are so accurate. The main reason for this can be the small number of the training data.

I was a little bit astonished by the performance of the Recurrent Neural Networks, because they produce more accurate predictions than the CNN models. The first model, which is built up LSTM layers and trained on amplitudes has 1.85 as logarithmic loss value. In the second GNU case with frequency values the loss value is 1.79.

Besides of this general solution, where I have applied the dataset of Ryerson Multimedia Research Lab I have also tried a specific solution. In this specific case I have trained the model with not just the collected dataset but also with a part of my captured and tagged videos. As I have expected I could achieve better performance with this specified model.

4 The basic iOS application

As I have already mentioned it above, I prepared two applications. After a brief introduction into the world of iOS development, I introduce the first application in this chapter, which is able to communicate via webserver. The application is capable of showing the live stream coming from the built-in camera of the device. On the one hand this video stream is getting sampled, on the other hand they are pre-processed. This means that the faces are detected and the images are transformed to an appropriate size. Then the pre-processed images are encoded in Base64 format and transmitted to the web application server.

4.1 Developing on iOS platform

4.1.1 Development Environment

I have used Xcode, which is an integrated development environment (IDE) of Apple, which is used to build apps for Apple products including the iPad, iPhone, Apple Watch, and Mac. Xcode provides tools to manage your entire development workflow from creating your app, to testing, optimizing, and submitting it to the App Store.

Xcode includes the iOS SDK (Software Development Kit), the debugger, and the LLVM (Low Level Virtual Machine) compiler for compiling C, C++, Objective-C and Swift programmer languages.

Moreover, Xcode includes a program for developing the UI (User Interface). This is the Interface Builder editor, which makes it simple to design a full user interface without writing any code. Simply you can drag and drop windows, buttons, text fields, and other objects onto the design canvas to create a functioning user interface [37].

4.1.2 Notes on developing for iOS

You can download the Xcode program free of charge from the App Store on OSX system. But, if you would like to upload your application to the Application Store, you need to buy a developer profile. However, Apple reviews all apps submitted to the App Store, in order to determine whether they are reliable, perform as expected, and are free of offensive materials. That’s why we always have to follow the App Review Guidelines as a developer, which provides rules and examples across a range of topics, including user

interface design, functionality, content, and the use of specific technologies. To sum up, I can say the iOS is a relatively closed system, but the users always find qualitative applications in the App Store.

4.1.3 A little bit of the programming language

An iOS application can be programmed either in a web format, which means that the application runs in a browser view, or in a native format. In case of a native format the program is written in Objective-C or Swift (perhaps C or C++), and turns into a binary code. In that case, the functions of the iOS can be reached easier and we also get a faster application.

Until Xcode 6 the Objective-C was the only official programming language of the Apple. This language is based on the characteristics of C language and it is entirely object-oriented, which means that the objects communicate via messages. Many people have doubts about it, as its syntaxes do not take after the modern languages and it requires a lot more codes when it is used. There was a great need for a new language, which could be attractive not just for the young programmers but for the elder Objective-C ones, as well. Thus became Swift, the new programing language, developed by Apple, also supported in June, 2014. I have also developed my application in Swift, but I often had to deal with sample codes in the documentations, which are written in Objective-C.

Swift is a script-like language with a compact syntax. Compared to Objective-C, it is more compact, which enables an easier coding. Due to its stricter rules, most of its errors can already be realized during the compiling time [70].

4.2 The structure of my application

The MVC (Model-View-Controller) architectural planning pattern is offered by Apple to develop applications. It is represented in the following Figure 32.

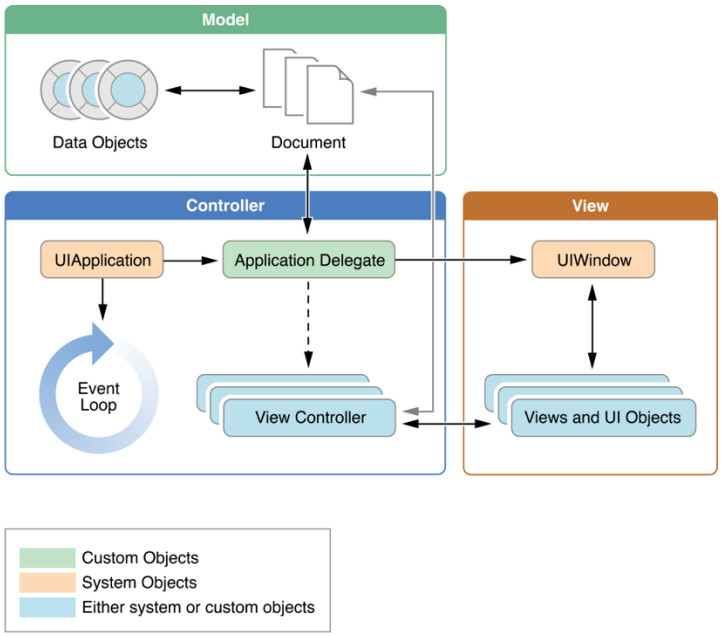

Figure 32 : Model View Controller architecture [42]

The View has a complex role, it draws onto the screen, recognizes the interactions of the users, then transmits them to the Controller. More basic View classes can appear simultaneously on the screen, and they can be embedded into each other. They don't have a direct contact to the model objects.

The Controller creates a bond between the View and the Model objects. Its basic class is the UIViewController, which controls a view hierarchy, creates the views and processes the incoming events. The Controller itself never draws onto the screen, as it is the task of the view.

The Model objects contain the data, with what the application is served. There is no separate Model class, it's up to the programmer's intents, how he would accomplish it [70].

4.2.1 **The Controllers**

In the following Figure 33 a part of my storyboard file can be seen.

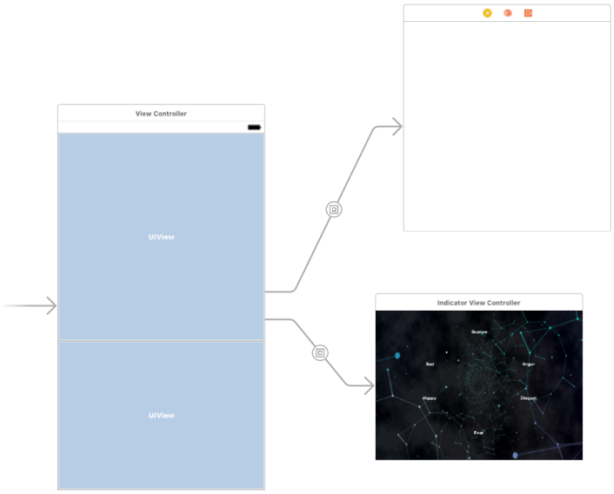

Figure 33 : A part of my storyboard file

The storyboard is a visual representation of the user interface of an iOS application. In this file, we can design the basic architecture of the application and define views and connections between screens. The following screenshot of my application also depicts that I have split the screen into two parts with ContainerView UI elements [41].

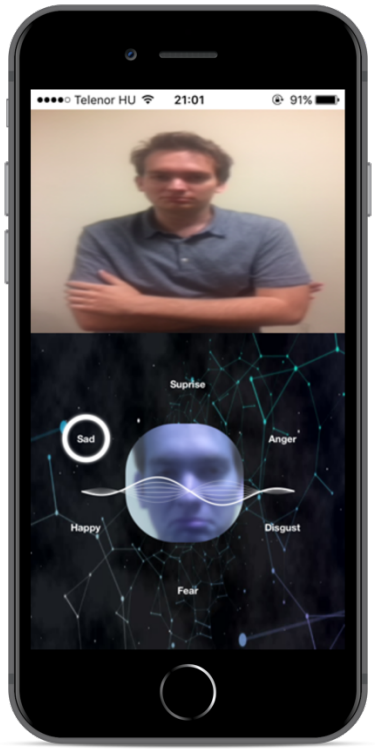

Figure 34 : Screenshot of my first iOS application

On the top of the screen there is the CameraViewController, which handles the video stream. This means that this controller class shows it on the screen and transfers the sampled images to the image pre-processor class. Then this class passes the processed image onto the web server manager.

The bottom part of my application is the IndicatorViewConotroller. It has got the main role after the response is arrived from the web server. This class receives information from the web service manager and it represents them on the screen. Moreover, having got the amplitude values of the incoming sound of the video stream, the class controls the waving animation on the screen.

4.2.2 The Views

The views are important, because they determine the user's impressions about the application. The exact features are not defined yet, thus I only implemented a primitive design. I have used 3rd party libraries for the animations. These are the 'KDCircularProgress', which is a circular indicator of the predicted emotions and the 'SwiftSiriWaveformView', which is intended to feedback the sound [43].

During the whole project, I used the Auto Layout, so that different screen sizes would be supported. The Auto Layout is an expressing method, which defines the sizes and positions of the views. Our expectations and constraints concerning the style of the surface can be defined. According to this, the system counts out the positions and sizes.

4.2.3 The Models

There are two Model classes in my application. These are the ImageProcessor and the Webservice.

4.2.3.1 Image pre-processing

I did the same things in the ImageProcessor as in the pre-processing step before the model training. However, I had to use here swift programming language and iOS libraries to detect and crop faces and resize the result image. It is good that I did these steps in the application, because I can send the relevant part of the picture to the server, reducing the data traffic.

4.2.3.2 Web service manager

In the following code snippet the PostImage class function can be seen from my Webservice class.

```
    class func PostImage(imageData: UIImage){
        let postURI = "http://192.168.43.4:8000/images/"
        let strBase64:String = imageData.base64EncodedString()
        let parameters: Parameters = [
            "image": strBase64
        ]
         Alamofire.request(postURI, method: .post, parameters: parameters,
encoding: JSONEncoding.default, headers: nil)
          .responseJSON { response in
                let str = response.result.value as! String
                let dict = convertToDictionary(text: str)
     DispatchQueue.main.async(){
      NotificationCenter.default.post(name:NSNotification.Name(rawValue:
"ChangeIndicators"), object: nil, userInfo: dict)
}}}
```

Here I have used the Alamofire 3rd party component, which is a HTTP networking library written in swift. This library provided me a more comfortable way to handle HTTP requests [43].

In the third line, you can see the Base64 coding process. Base64 is an encoding scheme, which represents the binary data in an ASCII string format. Then I prepared the parameter of the POST request. The request function of the Alamofire class starts the request on a background thread and the response can be handled in a closure. Closure is a self-contained block of functionality. In this closure, I reclaim the main thread from the operating system and I send a notification to the IndicatorViewController class to change the status of the indicators.

5 The web server

The web server receives Base64 encoded images in JSON format from the mobile application. After the prediction, the server sends the values back to the mobile application. These processes are based on the REST (REpresentational State Transfer) architecture.

5.1 The REST architecture

REST is a design principle, which generally runs over HTTP or HTPPS protocol. This architecture allows interactions between clients and services via a limited number of operations. These are the POST request to create a resource on the server, the GET to retrieve a resource, the PUT to update it and the DELETE to remove it. For this operation, the architecture uses its own unique universal resource indicators (URIs) [38]. When a web service implement the REST architecture, as my web application as well, it can be called as RESTful API (Application Programming Interface)

5.2 The Django technology

Django is a high-level open source Python Web framework. The developers emphasize that it is ridiculously fast, reassuringly secure and exceedingly scalable [39].

5.3 The main part of my web application

The following code snippet shows a POST request implementation with the help of the Django REST framework [40].

```
@api_view(['GET', 'POST'])
def image_list(request, format=None):
    if request.method == 'POST':
        serializer = ImageSerializer(data=request.data)
        if serializer.is_valid():
             loaded_model = KerasManager.loadModel()
             imageData = base64.b64decode(request.data["image"])
             image = load_img(imageData)
             imageRepresentation = img_to_array(image)
             predict = loaded_model.predict_proba(imageRepresentation)
             an = float(predict[0])
             di = float(predict[1])
                ...
            return Response(json.dumps({"an" : an, "di" : di, "fe" : fe,
               "ha": ha, "sa" : sa, "su" : su}), status=HTTP_201_CREATED)
          return Response(serializer.errors, status=HTTP_400_BAD_REQUEST)
```

When the image data is arrived, comes the loading of the stored model. For this purpose, I used the loadModel class function of the KerasManager manager class. Before the prediction, I had to decode the image from Base64 format and make an array representation. If these processes were successful, the server sends back the prediction array in JSON format via a HTTP response. Otherwise an error message is sent from the web server.

6 The enhanced iOS application

As I mentioned it before, beside the web server based application I have developed another one, as well. For this enhanced application no internet connection is required, because the models are involved in the native application.

In this part I will represent, in which this enhanced application is supposed to be more sophisticated, but in other approaches, it is also meant to be easier.

6.1 Usage of the mlmodel

After the model conversion from Keras library I get mlmodel format, which is an open format designed by Apple. In this model file description of the layers, the inputs and outputs, the class labels can be found. And, of course it also contains the learned weights and other parameters [61].

Having dragged the mlmodel files into the iOS project Swift model classes are automatically generated. Then the prediction can be done simply as follows:

```
guard let result = try? CNNmodel.prediction(input: imageData) else {
     return nil
    }
```

The guard statement is used to transfer program control out of a scope if the prediction fails [62]. Not the model call was the more complicated process but the pre-processing.

6.1.1 Image pre-processing

Regarding to the images I have used similar functions as in case of the previous application to detect, crop faces and resize the result image. However, even more operations were required in the pre-processing. In swift MLMultiArray format is used to represent input or output data for a model.

So that I could prepare such an array of the pictures, I needed to process them pixel by pixel. The red, green and blue channels needed to be filtered from each pixels, whereas the Core ML neural network models use a number of channels $*$ the height of the image $*$ width of the image layout for images [63].

6.1.2 **Sound pre-processing**

In my application I used frequency domain-based model, as I have already described it above, it was built up of GRU layers.

So that I could call it, I had to transform the time-domain sign coming via microphone to frequency values, for which I used the AKFFTTap class from the AudioKit framework [64].

6.2 The new User Interface features

As it can be seen in the following picture, the design of the application was recreated, and I designed a totally different UI. It contains three main elements: the yellow square, the wave giving a feedback of the voice, which was already part of the previous application, and finally the chart view.

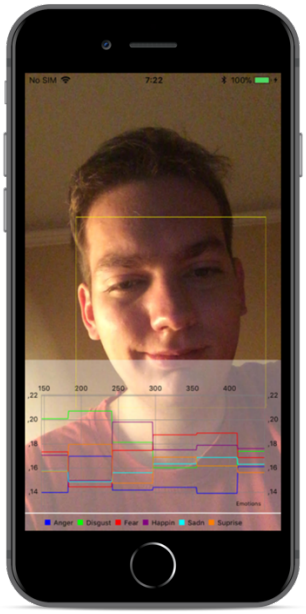

Figure 35 : The enhanced iOS application

6.2.1 **The yellow square**

The square on the screen symbolizes that part, which is the input picture into my CNN model. Of course, it is the face itself, which is drawn around in a square format after detection. For that purpose, I used UIBezierPath class, which is the part of iOS software development kit. After defining the square shape, additional methods of the class can be used to render it in the current drawing context [65].

6.2.2 **Chart**

I have used Charts third party library for the animated chart at the bottom of the screen. This library contains eight different types of charts, which are easily customisable.

The new values appear on the right side of the screen; in this way we can see a diagram moving horizontally [66].

6.3 Testing of the application

To test whether the operation of the application is optimized, I have used an Xcode integrated performance analyser tool, called Instruments. With the help of this tool the operation of the application can be monitored, the memory leaks and the network issues can be easily detected. In this following Figure a test can be seen under the usage of the application [69].

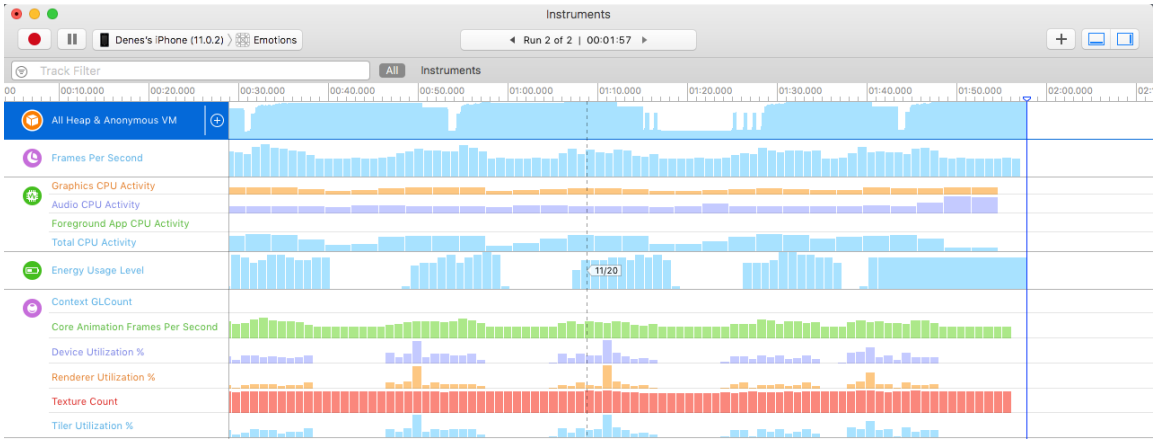

Figure 36 : Application test with Instruments tool

In this Figure a significant CPU and memory usage can be seen during the prediction processes.

Then I tested the User Experience, which officially is defined as "a person's perceptions and responses that result from the use or anticipated use of a product, system or service" (ISO 9241-210).

It could be the most important aspect in case of a mobile application, so I asked several people to test it with permanent emotions. After that I gathered the feedbacks from the users, regarding the operation and usage.

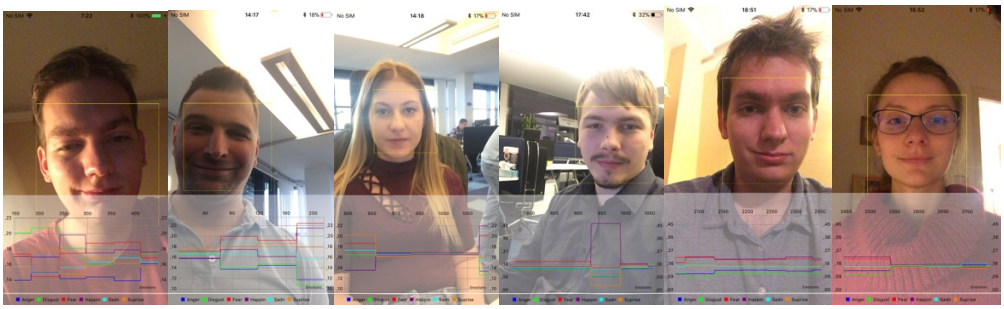

Figure 37 : User Experience testing

6.4 Experiences

This enhanced application seems to be better in several attempts: it does not require internet connection and the data remain on the device in safety. Unfortunately, the prediction process seemed to be a little bit slow. I have to admit, that I used a relatively old iPhone 6 device for testing. The newly released devices have dedicated GPU for machine learning calculations, which means the prediction process is probably faster on them.

7 Summary and future works

During my thesis work I could get familiar with the Machine Learning, Neural Networks, and Deep Learning technologies. I could investigate the structure of the Convolutional Neural Networks and Recurrent Neural Networks as well. Moreover, I built more models and trained them. Unfortunately, I couldn't reach very good performance values, because the video database wasn't big enough. Thus I will have to collect more emotional video databases in the future to make my model more accurate.

Besides that, I will have to train my model with a computer, which has stronger parameters than my laptop and has a strong video card. Personally, I believe that with a bigger amount of the training data and with a stronger training device I could increase significantly the performance of my model.

I have encountered challenges not only in the creation of the model but also in the pre-processing of data. Sampling the videos and finding the faces in the picture was also my task. Of course, the video includes the sound, which had to be processed as well, not just in time domain, but also in frequency domain.

I could also design and implement two iOS mobile applications. One of which was capable of recording videos, communicating with the server and displaying information on the screen. The other enhanced application uses the machine learning library, having been released by Apple in September. The most important fact about it is that the trained models were integrated directly into the application, in this way I could call them directly from the application, without any internet connection.

Comparing the two solutions, both of them have its advantages and disadvantages. The prediction is faster in the server based solution, whereas a server has got a bigger capacity, the uploaded data can be utilised to improve the model and to monitor the usage. Although it generates data traffic because of the uploading, and the problem of data privacy also comes up.

When it comes to the integration of the model into the application, the problem of data traffic disappears. Though the prediction takes quite a long time, I feel that this solution could be outstanding in the future. The newly released iPhone devices have dedicated GPU for machine learning calculations. However, it takes some years until they become widely-spread.

Developing for mobile devices was not a completely new field for me, even though I met several new cases, which had to be solved. For example, the appropriate handling of the main and the background threads of the application. Moreover, getting familiar with new apple and 3rd party libraries. If I could develop more accurate models and I could reduce the prediction time of the models, I believe my applications would be liked in the App Store of Apple.

I am glad that I had the chance to investigate the topic of emotion recognition, because as I have mentioned it previously, it is used in many important cases. Moreover, it is essential for creating a real artificial intelligence.

References

[1] Hong-Wei Ng, Viet Dung Nguyen, Vassilios Vonikakis, Stefan Winkler: Deep Learning for Emotion Recognition on Small Datasets Using Transfer Learning Advanced Digital Sciences Center (ADSC), Singapore
Available from: http://vintage.winklerbros.net/Publications/emotiw2015.pdf 22.05.2017

[2] Quanzeng You, Jiebo Luo, Hailin Jin, Jianchao Yang: obust Image Sentiment Analysis Using Progressively Trained and Domain Transferred Deep Networks
Available from:
https://www.cs.rochester.edu/u/qyou/papers/sentiment_analysis_final.pdf
22.05.2017

[3] Online definitions:
http://www.investopedia.com/terms/a/ai-winter.asp
http://whatis.techtarget.com/definition/Turing-Test 22.05.2017

[4] NVIDIA compony and stock prices:
http://www.nvidia.com/object/machine-learning.html#sthash.wSQ1lEkT.dpuf
https://finance.yahoo.com/quote/nvda?ltr=1 27.05.2017

[5] Picard, R.W.: Affective computing. MIT Press, Cambridge (1997)
Available from:
https://www.pervasive.jku.at/Teaching/_2009SS/SeminarausPervasiveComputing/Begleitmaterial/Related%20Work%20(Readings)/1995_Affective%20computing_Picard.pdf 27.05.2017
MIT Media Laboratory: http://affect.media.mit.edu 27.05.2017

[6] Machine learning image:
http://www.clipartkid.com/images/842/artificial-intelligence-machine-learning-data-mining-big-data-diagram-KQDEsT-clipart.png 27.05.2017
Machine lerning wikipedia:
https://en.wikipedia.org/wiki/Machine_learning 27.05.2017

[7] SAS: Data Mining..
Available from: https://www.sas.com/en_us/insights/analytics/data-mining.html
27.05.2017

[8] Kishan Mehrotra, Chilukuri Mohan and Sanjay Ranka: Elements of Artificial Neural Networks, 1997
Available from:
https://www.researchgate.net/publication/228824281_Elements_of_Artificial_Neural_Nets 27.05.2017

[9] Stanford course: Fei-Fei Li, Justin Johnson, Serena Yeung: CS231n: Convolutional Neural Networks for Visual Recognition, Spring 2017

[10] Caroline Clabaugh, Dave Myszewski, and Jimmy Pang: The Intellectual Excitement of Computer Science, 2000
Available from:
https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Neuron/index.html 27.05.2017

[11] Rob Schapire: Theoretical Machine Learning, February 4, 2008
Available from:
http://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0204.pdf 27.05.2017

[12] A Basic Introduction To Neural Networks
Available from: http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html 27.05.2017

[13] Lee Jacobson: Introduction to Artifical Neural Networks
Available from: http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7 27.05.2017

[14] Gustav Larsson: Initialization of deep networks, 2015
Available from: http://deepdish.io/2015/02/24/network-initialization/ 27.05.2017

[15] M. D. Garris, R. A. Wilkinson, and C. L. Wilson, Analysis of a Biologically Motivated Neural Network for Character Recognition, ACM Press, George Mason University, May 1991
Available from:
http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=906359 27.05.2017

[16] Jeff Knisley, Dept. of Math., East Tennessee State University, Multivariable Calculus Online, Chapter 7 (Properties of the Gradient)
Available from: http://math.etsu.edu/multicalc/prealpha/Chap2/Chap2-7/printversion.pdf 27.05.2017
Gradient descend
Available from: http://wiki.fast.ai/index.php/Gradient_Descent 27.05.2017

[17] Ryerson Multimedia Lab: Emotion Database
Available from: http://www.rml.ryerson.ca/rml-emotion-database.html 27.05.2017

[18] MIT open course, number: 6.034, Prof. Patrick Henry Winston: Aftifical Inteligence, Fall 2010
Available from: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/ 27.05.2017

[19] Gradient descend image
Available from: http://dsdeepdive.blogspot.com/2016/03/optimizations-of-gradient-descent.html 27.05.2017

[20] David E. Rumelhart, Goeffrey E. Hinton and Ronald J. Williams: Learning representations by back-propagating errors, 1986 Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia, USA

[21] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” arXiv:1502.03167 [cs], Feb. 2015.
Available from: https://arxiv.org/pdf/1502.03167v2.pdf 27.05.2017

[22] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov: Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Department of Computer Science University of Toronto, 2014
Available from:
http://www.jmlr.org/papers/volume15/srivastava14a.old/source/srivastava14a.pdf 27.05.2017

[23] Sebastian Ruder (2016): An overview of gradient descend optimization algoritms
Available from: http://sebastianruder.com/optimizing-gradient-descent/index.html 11.12.2017

[24] Yann LeCun, Yoshue Bengio and Geoffrey Hinton (2015) : Deep Learning
Available from:
https://www.cs.toronto.edu/~hinton/absps/NatureDeepReview.pdf 27.05.2017

[25] Denny Britz (2015): Understanding convolutional neural networks for NLP
Available from: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/ 27.05.2017

[26] Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner (1998): Gradient-Based Learning Applied to Document Recognition
Available from: http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf 27.05.2017

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei (2015) : ImageNet Large Scale Visual Recognition Challenge
Available from: https://arxiv.org/pdf/1409.0575.pdf 27.05.2017

[28] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton : ImageNet Classification with Deep Convolutional Neural Networks
Available from: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf 27.05.2017

[29] Christian Szegedy, Pierre Sermanet, Wei Liu, Yangqing Jia, Dumitru Erhan ,Scott Reed, Dragomir Anguelov, Andrew Rabinovich, Vincent Vanhoucke (2014): Going deeper with convolutions
Available from: https://arxiv.org/pdf/1409.4842.pdf 27.05.2017

[30] Jon Shlens (2016): Train your own image classifier with Inception in TensorFlow
Available from: https://research.googleblog.com/2016/03/train-your-own-image-classifier-with.html 27.05.2017

[31] Tensorflow official page: https://www.tensorflow.org/ 27.05.2017

[32] Theano documentations: http://deeplearning.net/software/theano/ 27.05.2017

[33] Keras documentations: https://keras.io/ 27.05.2017

[34] Paul Viola and Michael Jones (2001): Rapid Object Detection using a Boosted Cascade of Simple Features
Available from: http://wearables.cc.gatech.edu/paper_of_week/viola01rapid.pdf 27.05.2017

[35] OpenCV documentations: Face Detection using Haar Cascades
Available from:
http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html 27.05.2017

[36] ScikitLearn documentations: Model evaluation: quantifying the quality of predictions
Available from: http://scikit-learn.org/stable/modules/model_evaluation.html 27.05.2017

[37] XCode overview:
Available from: https://developer.apple.com/xcode/ 27.05.2017

[38] Alex Rodriguez (2008) : RESTful Web services: The basics
Available from: http://www.gregbulla.com/TechStuff/Docs/ws-restful-pdf.pdf 27.05.2017
REST wikipage:
http://wiki.servicenow.com/index.php?title=REST_API#gsc.tab=0 27.05.2017

[39] Django official page: https://www.djangoproject.com 27.05.2017

[40] Django REST framework: http://www.django-rest-framework.org 27.05.2017

[41] Apple documentations: Storyboard
Available from:
https://developer.apple.com/library/content/documentation/General/Conceptual/Devpedia-CocoaApp/Storyboard.html 27.05.2017

[42] MVC Figure:
https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html 27.05.2017

[43] 3rd party libraries:
KDCircularProgress: https://github.com/kaandedeoglu/KDCircularProgress 27.05.2017
SwiftSiriWaveformView:
https://github.com/alankarmisra/SwiftSiriWaveformView 27.05.2017
Alamofire: https://github.com/Alamofire/Alamofire 27.05.2017

[44] Cole Calistra (2015) : The Universally Recognized Facial Expression of Emotions
Available from: https://www.kairos.com/blog/the-universally-recognized-facial-expressions-of-emotion 25.11.2017
Management Mania (2016): Six Basic Emotions
Available from: https://managementmania.com/en/six-basic-emotions 25.11.2017

[45] Paul Ekman (1999) : Handbook of Cognition and Emotion (Sussex, U.K. John Wiley & Sons, Ltd), Facial Epressions (Chapter 16)
Available from: https://www.paulekman.com/wp-content/uploads/2013/07/Facial-Expressions.pdf 25.11.2017

[46] Apple : CoreML documentations
Available from: https://developer.apple.com/documentation/coreml 25.11.2017
https://developer.apple.com/documentation/coreml/getting_a_core_ml_model 25.11.2017

[47] DeepLearning4J : A Beginner's Guide to Recurrent Networks and LSTMs
Available from: https://deeplearning4j.org/lstm.html 25.11.2017

[48] Christopher Olah (2015) : Understanding LSTM Networks
Available from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ 25.11.2017

[49] Wojciech Zaremba, Ilya Sutskever, Oriol Vinyals (2015): Recurrent Neural Network Regularization
Available from: https://arxiv.org/pdf/1409.2329.pdf 25.11.2017

[50] Tsungnan Lin, Bill G. Horne, Peter Tino and C. Lee Giles: Learning long-term dependencies is not as difficult with NARX recurrent neural networks
Available from: https://clgiles.ist.psu.edu/papers/UMD-CS-TR-3500.long-term.dependencies.narx.pdf 25.11.2017

[51] Sepp Hochreiter and Jürgen Schmidhuber(1997): Long short-term memory
Available from: http://www.bioinf.jku.at/publications/older/2604.pdf 25.11.2017

[52] Jason Browniee (2016) : Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras
Available from: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/ 25.11.2017

[53] Klaus Greff, Rupesh K. Srivastava, Jan Koutn´ık, Bas R. Steunebrink, Jürgen Schmidhuber (2017) : LSTM: A Search Space Odyssey
Available from: https://arxiv.org/pdf/1503.04069.pdf 25.11.2017

[54] Felix A. Gers, Nicol N. Schraudolph and Jürgen Schmidhuber (2002) : Learning Precise Timing with LSTM Recurrent Networks
Available from:
http://machinelearning.wustl.edu/mlpapers/paper_files/GersSS02.pdf 25.11.2017

[55] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho and Yoshua Bengio (2014): Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling
Available from: https://arxiv.org/pdf/1412.3555.pdf 25.11.2017

[56] Scipy wavfile library documentation
Available from:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.io.wavfile.read.html#scipy.io.wavfile.read 25.11.2017

[57] Ffmpeg documentation
Available from: https://www.ffmpeg.org/documentation.html 25.11.2017

[58] C. Kankelborg (2014): DFT and FFT
Available from:
http://solar.physics.montana.edu/kankel/ph567/examples/Octave/dft/dft.pdf
25.11.2017

[59] Coremltools documentation by Apple: Convert a Keras model to Core ML format
Available from:
https://apple.github.io/coremltools/generated/coremltools.converters.keras.convert.html 25.11.2017

[60] Keras documentation: The Sequential model API
Available from: https://keras.io/models/sequential/ 25.11.2017

[61] Analytics Vidhya: Intro to Apple's CoreML
Available from: https://www.analyticsvidhya.com/blog/2017/09/build-machine-learning-iphone-apple-coreml/ 25.11.2017

[62] Apple documentation: The Swift Programming Language (Swift 4)
Available from:
https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/index.html 25.11.2017

[63] Apple documentation: MLMultiArray
Available from: https://developer.apple.com/documentation/coreml/mlmultiarray
25.11.2017

[64] AudioKit documentation: AKFFTTap class
Available from: http://audiokit.io/docs/Classes/AKFFTTap.html 25.11.2017

[65] Apple documentation: UIBezierPath class
Available from: https://developer.apple.com/documentation/uikit/uibezierpath
25.11.2017

[66] Daniel Cohen Gindi: Chart library
Available from: https://github.com/danielgindi/Charts 25.11.2017

[67] J. Nicholson, K. Takahashi and R. Nakatsu : Emotion Recognition in Speech Using Neural Networks
Available from: https://link.springer.com/article/10.1007/s005210070006
25.11.2017

[68] Vladimir Hozjan and Zdravko Kačič: Context-Independent Multilingual Emotion Recognition from Speech Signals
Available from: https://link.springer.com/article/10.1023/A:1023426522496
25.11.2017

[69] Apple documentation: About Instruments
Available from:

https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/index.html 25.11.2017

[70] Dr. Imre Kelényi, Dr. Forstner Bertalan and Dr. László Blázovics: iOS-based Software Development course materials
Available from: https://www.aut.bme.hu/Course/ios 01.12.2017