**Real time static gesture detection using machine learning**

**By**

**Sandipgiri Goswami**

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science (MSc) in Computational Sciences

The Faculty of Graduate Studies

Laurentian University

Sudbury, Ontario, Canada

© Sandipgiri Goswami, 2019

**Abstract**

Sign gesture recognition is an important problem in human computer interaction with signiﬁcant societal influence. However, it is a very complex task, since sign gestures are naturally deformable objects. Gesture recognition contains unsolved problems since last two decades, such as low accuracy or low speed, and despite many proposed methods, no perfect result has been found to explain these unsolved problems. In this paper, we suggest a machine learning approach to translating sign gesture language into text.

In this study, we have introduced self-generated image data set for American sign language (ASL). This dataset was a collection of 36 characters which contain A to Z alphabets and 0 to 9 number digits. The proposed system can recognize static gestures. This system can learn and classify specific sign gesture of any person. We used a convolutional neural network algorithm for classified image to text. We achieved 99.00% accuracy on the alphabet gestures and 100% accuracy on digits.

Keywords: *Sign gestures, Image processing, Machine learning, Conventional neural network.*

**Table of Contents**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Abstract…………………………………………………………………………………………… | | | | | | |  |
| Acknowledgments………………………………………………………………………………… | | | | | | |  |
| Table of contents………………………………………………………………………………….. | | | | | | |  |
| List of Figures…………………………………………………………………………………….. | | | | | | |  |
| List of Tables……………………………………………………………………………………… | | | | | | |  |
| **1** | **Introduction** | | | | | | 1 |
|  | 1.1 | Sign Language……………………………………………………………………… | | | | |  |
|  | 1.2 | Mythologies and Misunderstandings about Sign Language………………………... | | | | |  |
|  | 1.3 | Objective…………………………………………………………………………… | | | | |  |
|  | 1.4 | Methodology………………………………………………………………………. | | | | |  |
|  | 1.5 | Outline……………………………………………………………………………... | | | | |  |
| **2** | **Related Work** | | | | | |  |
|  | 2.1 | American Sign Language using Machine Learning……………………………….. | | | | |  |
| **3** | **Data set** | | | | | |  |
|  | 3.1 | American Sign Language………………………………………………………….. | | | | |  |
|  | 3.2 | Characteristics of American Sign Language……………………………………….. | | | | |  |
|  | 3.3 | Statistics about sign language use in Canada ……………………………………… | | | | |  |
|  | 3.4 | Dataset and variables ……………………………………………………………… | | | | |  |
|  | 3.5 | Capturing Images for Dataset ……………………………………………………… | | | | |  |
| **4** | **Hand Gesture Detection** | | | | | |  |
|  | 4.1 | Introduction ………………………………………………………………………… | | | | |  |
|  | 4.2 | Hand Detection Approaches ………………………………………………………. | | | | |  |
|  |  | 4.2.1 | Colour ……………………………………………………………………… | | | |  |
|  |  | 4.2.2 | Hand shape…………………………………………………………………. | | | |  |
|  |  | 4.2.3 | Learning detector from pixel values ……………………………………….. | | | |  |
|  |  | 4.2.4 | 3D model-base detection …………………………………………………… | | | |  |
|  |  | 4.2.5 | Motion ……………………………………………………………………… | | | |  |
|  | 4.3 | My Approach for Hand Detection………………………………………………….. | | | | |  |
|  |  | 4.3.1 | Skin Detection………………………………………………………………. | | | |  |
|  |  | 4.3.2 | Contour Comparisons………………………………………………………. | | | |  |
| **5** | **Machine Learning Approach** | | | | | |  |
|  | 5.1 | Artificial Intelligence ………………………………………………………………. | | | | |  |
|  | 5.2 | Image Processing…………………………………………………………………… | | | | |  |
|  | 5.3 | Machine Learning ………………………………………………………………….. | | | | |  |
|  |  | 5.3.1 | | Machine Learning Approach……………………………………………….. | | |  |
|  |  | 5.3.2 | | Structure of Machine Learning Algorithm…………………………………. | | |  |
|  |  | 5.3.3 | | Model Complexity…………………………………………………………. | | |  |
| **6** | **Neural Network** | | | | | |  |
|  | 6.1 | History……………………………………………………………………………… | | | | |  |
|  | 6.2 | Structure of Neural Networks………………………………………………………. | | | | |  |
|  |  | 6.2.1 | | | Model of Neuron…………………………………………………………… | |  |
|  |  | 6.2.2 | | | Topology of the Network………………………………………………….. | |  |
|  |  | 6.2.3 | | | Cost Function……………………………………………………………… | |  |
|  |  | 6.2.4 | | | Optimization Procedure…………………………………………………… | |  |
|  | 6.3 | Convolutional Neural Networks…………………………………………………… | | | | |  |
|  |  | 6.3.1 | | | | Structure of CNN………………………………………………………….  ………………… |  |
|  |  | 6.3.2 | | | | Training of CNN…………………………………………………………… |  |
|  | 6.4 | Regularization of Neural Networks………………………………………………… | | | | |  |
| **7** | **Experiment and Result** | | | | | |  |
|  | 7.1 | Software……………………………………………………………………………… | | | | |  |
|  | 7.2 | Hardware and Software Configuration………………………………………………. | | | | |  |
|  |  | 7.2.1 | | | | Architecture………………………………………………………………… |  |
|  | 7.3 | Model Structure……………………………………………………………………… | | | | |  |
|  | 7.4 | Result ………………………………………………………………………………... | | | | |  |
| **8** | **Conclusion and Future work** | | | | | |  |
|  | 8.1 | Conclusion…………………………………………………………………………… | | | | |  |
|  | 8.2 | Future work………………………………………………………………………….. | | | | |  |
|  | Bibliography.. | | | | | |  |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| Figure 1.1 | Project overview for American sign language……………………………. |  |
| Figure 2.1 | A Data Glove design with Sensor………………………………………… |  |
| Figure 2.2 | A Glove device with Sensor………………………………………………... |  |
| Figure 3.1 | American Sign language Manual Alphabet……………………………….. |  |
| Figure 3.2 | American Sign language numbers…………………………………………. |  |
| Figure 3.3 | Data set images…………………………………………………………….  . |  |
| Figure 4.1 | Hand posture detection steps………………………………………………. |  |
| Figure 4.2 | HSV Colour Space………………………………………………………… |  |
| Figure 4.3 | Images of detecting hand postures…………………………………………. |  |
| Figure 5.1 | Diagram of image processing pipeline……………………………………. |  |
| Figure 5.2 | The figure shows different levels of generalization of the model………………… |  |
| Figure 5.3 | Relationship between the model complexity and its ultimate accuracy is the relationship between training and testing error……………………………. |  |
| Figure 6.1 | Diagram of artificial neuron………………………………………………. |  |
| Figure 6.2 | Activation Functions ………………………………………………………. |  |
| Figure 6.3 | Fully connected Feed Forward Neural Network…………………………… |  |
| Figure 6.4 | Structure of Convolutional Neural Network………………………………. |  |
| Figure 6.5 | A zero padded 4x4 matrix………………………………………………….. |  |
| Figure 6.6 | Principle of Max-pooling…………………………………………………. |  |
| Figure 6.7 | Dropout: (a)Standard fully connected network. (b) Network with some neurons deactivated. (c) Activation of neuron during training phase. (d) Activation of neuron during testing phase………………………………… |  |
| Figure 7.1 | CNN network architecture for Alphabets…………………………………. |  |
| Figure 7.2 | Epochs vs. validation accuracy for digits…………………………………. |  |
| Figure 7.3 | Epochs v's validation accuracy for alphabets ……………………………… |  |
| Figure 7.4 | Confusion matrix for 0 to 9 digits………………………………………… |  |
| Figure 7.5 | Confusion matrix for A to Z digits ……………………………………….. |  |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| Table 1.1 | Sign Language in the Americas…………………………………………….. |  |
| Table 2.1 | Statics about Sign Language as a Mother Tongue………………………….. |  |
| Table 2.2 | Statics about Knowledge of Sign Languages………………………………… |  |
| Table 3.1 | Dataset Description and Image property……………………………………... |  |
| Table 7.1 | Hardware configuration……………………………………………………... |  |
| Table 7.2 | Software configuration………………………………………………………. |  |

**Chapter 1**

**Introduction**

**1.1 Sign Language**

The World Health Organization (WHO) estimated that, 250 million people in the world are deaf as well as dumb. [1] These group people of use symbolic language to communicate with other people. This symbolic language is called sign language. Sign Language is a build for communication used worldwide among hard of hearing, and deaf peoples. Sign language is not a unique language signed consistently in different countries. Sign language is not recent improvement. There is proof that speaking through gestures has been around since the start of human development [20]. Different counties have their own sign language such as American Sign Language, French Sign Language, Indian Sign Language and Puerto Rican Sign Language to name a few. Table 1 gives information about different sign languages used in western continent. Gesture based communication is dependent on region and has significant differences from other languages. It is very important to understand sign language when we communicate with deaf or young children and their families. Lack of understanding results in significant challenges in understanding this community and may result miscommunication.

Table 1.1: Sign Language in the Americas [20]

|  |  |  |
| --- | --- | --- |
| **North America** | **Central America** | **South America** |
| • American Sign Language  •Inuit Sign Language  • Quebec Sign Language  •Puerto Rican Sign Language | •Costa Rican Sign Language  •Guatemalan Sign Language  •Honduras Sign Language  •Mayan Sign Language  •Mexican Sign Language  •Nicaraguan Sign Language  •Panamanian Sign Language  •Salvadorian Sign Language  •Tijuana Sign Language | •Argentine Sign Language  •Bolivian Sign Language  •Brazilian Sign Language  •Chilean Sign Language  •Colombian Sign Language  •Ecaudorian Sign Language  •Paraguayan Sign Language  •Peruvian Sign Language  •Uruaguayan Sign Language  •Venezuelan Sign Language |

Sign Language is a language which is used to convey messages by hand movements, facial expression and body language for communication. It is mainly used by deaf and people who can hear but cannot speak. Sometime family members and relatives must learn sign language to interpret which enables deaf and wider communities to communicate with each other.

**1.2 Mythologies and Misunderstandings about Sign Language**

The Sign language is enclosed by many mythologies and misunderstanding. Most people who are disabled not thinking that Sign language is just simple a manual representation of the spoken language which is not true. In fact, our language and sign language of the deaf have little in common. Sign language has the difficulty of the verbal language, but it is self-determining from the alphabets. The best example is British Sign Language and American Sign Language which are meaningless, although the facts show that disable people from the United States and Britain perfectly understand each other.

Another common misunderstanding about sign language is that it is globally understandable which is of course not true. As explained above, the Sign language that is used by the deaf in Unites States and Britain are not the same. The different sign languages might be similar in some alphabets but a deaf person from one country to country communicate as fluently as hearing people from the two countries.

Since sign language is a language of its distinct language, finger spelling or the use of guidebook alphabet cannot be used as an alternative to sign. It is utilized in marking the words with a non-existing sign or when the sign isn't known. In addition, Deaf person would take hours to convey a few minutes of messages through finger spelling.

**1.3 Objective:**

My main objective of my thesis is to help the deaf community to increase their self-esteem and IQ level and improve their communication skills. Students who are deaf or have a deaf parent or have a close relative with deaf individual will learn by themselves about sign language alphabets and numbers. Deaf community will learn their first step towards the American sign language. Although correct usage of sign gesture plays a very important part in effective communication, deaf students are also encouraged to establish connection to the deaf community and to carry their new knowledge and skill beyond the classroom and into the community at large. Some of important objective is discussed in Chapter 3.

**1.4 Methodology**

In this thesis, Image classification and machine learning have been used for Interpreting American sign language. For image classification, computer vision algorithms were used to capture images and to process data set for filtering as well as reducing noise from images. Finally, the data set is trained Sign machine learning algorithm, conventional neural network for measuring accuracy of training data sets, result of thesis algorithm is explained in chapter 7. The abstract view of the derived approach combining the image classification and machine learning for American sign language is shown in Figure 1.

Feature Vector

Prepare Image

Feature Extraction

Neural Network

Feature Extraction

Convert RGB to GRAY

Classification

Apply Threshold/Edge Detection

Classified Image

*Figure 1.1: Project overview of American sign language*

**1.5 Outline:**

**Chapter 2**

**Related Work**

**2.1 American Sign Language using Machine Learning**

American sign language recognition is not a new machine learning problem. During recent decades, different researches already worked on different classifiers such as linear classifiers, neural networks and Bayesian networks [2-11].

As per research point of view a linear classifier is easy to work with because the linear classifier is relatively simple models, it requires sophisticated feature extraction and preprocessing methods to get good results [2, 3, 4]. Singha and Das [2] achieved an accuracy of 96% on Ten classes for images of gestures of one hand using Karhunen-Loeve Transforms. These translate and rotate the axes to build up a new framework based on the variance of the data. This technique is useful after using a skin color detection, hand cropping and edge recognition on the images. They use a linear classifier to recognize number sign including a thumbs up, first and index finger pointing left and right, and numbers only. Sharma [4] has done research using Support Vector Machines (SVM) and k-Nearest Neighbors (KNN) to illustrate each color channel after background noise deletion and noise subtraction [4]. Their research suggests using contours, which is very useful to represent hand contours. They got an accuracy of 62.3% using a Support Vector Machines on the segmented color channel model.

Machine learning is most commonly used for image recognition. Hidden Markov Model (HMM) and Dynamic Time Warping (DTW), two kinds of machine learning methods, are widely applied to achieve high accuracies [5, 6, 7]. These are mostly good at capturing time-based patterns, but they require clearly characterized models that are defined before learning. Starner and Pentland [5] used a Hidden Markov Model and a 3-Dimesional glove that detects hand movement. Since the glove can attain 3-Dimesional detail from the hand regardless of spatial orientation, they achieved the best accuracy of 99.2% on the test set. Using Hidden Markov Model uses time series data to track hand actions and classify based on the position of the hand in recent frames.

Suk [6] suggested a system for detecting hand gestures in a continuous video stream using a dynamic Bayesian network or DBN model. They try to classify moving hand gestures, such as creating a circle around the body or waving. They attain an accuracy of nearly 99%, but it is worth noting that all hand gestures are different from each other and are not American Sign Language. However, the motion-tracking feature would be applicable for classifying the dynamic letters of ASL: j and z.

Artificial Neural networks (ANN) have been used to capture American Sign language transformation [8, 9, 10, 11]. Possibly, the most important advantage of artificial neural networks is that they represent the most important classification structures.However, ANN require significantly more time and data to train. Up to the present time**,** most have been comparatively low.Mekala [8] classified video of the ASL alphabet into text using unconventional feature abstraction and a three-layer Neural Network.They extracted features using hand situation and movement.In the past, American sign language classification, could recognize the presence and position of 6 “points of interest” in the hand, each finger and the center of the palm. Mekala also used Fourier Transforms of the images to classify what section of the frame the hand is positioned in.Whereas they claim correctly categorize 100% of images with this framework, there is no indication of whether this result was reached in the training, validation or test set.

Admasu and Raimond [9] classified Ethiopian Sign Language and achieved a 88.5 % accuracy result using a feed Forward Neural Network.They use a substantial amount of image preprocessing, including image size standardization, image background deduction, contrast adjustment, and image segmentation**.** Gabor Filter and Principal Component Analysis method were used to extract features.The most related work up to date is by L. Pigou’s [11] research of ANN’s to categorize 20 Italian gestures from the “ChaLearn 2014 Looking at People” gesture recognizing competition.They used a Microsoft Kinect on whole body images of people performing the gestures and reach a cross-validation accuracy of 91.7%.With the 3-Dimensional glove, the Kinect allows detection of depth features, which helps significantly in classifying American sign language.

Non-Vision based technology such as Glove-based hand shape recognition normally contains the person wearing gloves and a certain quantity of wires to connect this glove to a computer. These methods are very hard and non-natural way to communicate with the computer [15]. This device required electricity or electromagnetic interference to get data about the hand, which is sufﬁcient to provide a description of a handshape gesture [16]. Scientists refer to data gloves in different ways, e.g. CyberGlove and Accele Glove.

Figure 2.1 shows the position of the sensors in a data glove proposed by Bedregal [17]. Basically, a timeline of frames can characterize any movement. Thus, a timeline of hand arrangement represents a hand movement using a data glove. An arbitrary generated hand conﬁguration was used to replicate the data transfer [17]. Each expression of the handshape is represented by a tuple of interval angles from each sensor. The detection was applied to Brazilian Sign Language (LIBRAS), using Fuzzy logic.

****

*Figure: 2.1. A Data Glove design with Sensor.*

In this paper they developed a similar hardware device called the Accele Glove. In their research they used a microelectronic mechanical system (MEMS) to extract hand conﬁguration. They have been functional in Vietnamese Sign Language for twenty-three gestures with Fuzzy logic. They achieved the results by handshape, with an overall 98% precision. The relative angles between palm and finger is the data found from the sensing device. The glove covers six accelerometers and a BASIC Stamp microcontroller as in Figure 2.2[18-19].

****

*Figure 2.2 A Glove device with Sensor*

Researchers [] have proposed a new system for a gesture-to-speech/text for the deaf community, applied to Arabic Sign Language. This author includes the design and implementation of a smart glove. The main advantage of this glove is that it does not depend on light conditions, which means it gives better accuracy in dark environments. As per the author, the glove is low price, low power consumption and has full mobility. Another benefit of these gloves is that they attached ﬂex sensors which used a wireless interface to a microcontroller.

**Chapter 3**

**Data set**

**3.1 American Sign Language:**

American sign language [13] is used to communicate between the deaf community and normal community. However, there are only 2.5 million ~ 5.0 million who speak sign language which significantly limits the number of people they can easily communicate with [12].



*Figure 3.1 American Sign language Manual Alphabet [13].*

American Sign Language is implemented from French sign language which was introduced by Thomas Hopins Gallaudet in United States [12]. ASL is like French sign language; individuals who speak American Sign Language can effectively communicate in French Sign Language. A variation of American Sign Language exits there are variations between English spoken in England, United States or Australia, there are differences in their sign languages [12].

****

*Figure 3.2. American Sign language numbers. [13]*

**3.2 Characteristics of American Sign Language:**

* American Sign language is an entire visual-gestural dialect with its very own language structure, vocabulary, and linguistic structure.
* Like other sign languages, it utilizes the hands, the body, and the face looks (counting mouth developments) to express significance and the eyes to see the meaning.
* Hand - to-hand connection is especially critical in ASL since it has no composed frame. There are in any case, documentation frameworks that are utilized for recording signs on paper.
* ASL is separate from English and is unique from other sign languages. An example of the distinctiveness of sign languages from each other and from the surrounding spoken language(s) is that, although English is the shared spoken language of the U.S., Canada, and Britain, signers of ASL do not understand signers of British Sign Language (BSL).

**3.3. Statistics** **about sign language use in Canada:**

In Canada, Statistics Canada reports that, as indicated by the 2006 Census, 8,995 people revealed a gesture-based communication just like their primary language or one of their first languages, as given below.

Table 2.1: Statics about Sign Language as a Mother Tongue [14].

|  |  |
| --- | --- |
| American Sign Language | 2,485 |
| Quebec Sign Language | 730 |
| Sign languages, not included elsewhere | 5,780 |

In addition, Statistics Canada reports that as per the 2006 Census, 43,090 people reported knowledge of a gesture-based communication, as provided below.

Table 2.2: Statics about Knowledge of Sign Languages [14].

|  |  |
| --- | --- |
| American Sign Language | 11,110 |
| Quebec Sign Language | 730 |
| Sign languages, not included elsewhere | 5,780 |

**3.4 Dataset and variables:**

I have created my own data set. This dataset was a collection of 36 characters which contain A to Z alphabets and 0 to 9 number digits. I used right hand to capture 1000 images for specific alphabets and numbers. The code was implemented to convert flip images to right to left hand image. The height and width ratios vary significantly, but average approximately 50X50 pixels. The dataset contains over 72,000 images in gray scale color. Additionally, people can add their images to this dataset. Below figure shows an image of A to Z alphabet.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **A** |  | **N** |
|  | **B** |  | **O** |
|  | **C** |  | **P** |
|  | **D** |  | **Q** |
|  | **E** |  | **R** |
|  | **F** |  | **S** |
|  | **G** |  | **T** |
|  | **H** |  | **U** |
|  | **I** |  | **V** |
|  | **J** |  | **W** |
|  | **K** |  | **X** |
|  | **L** |  | **Y** |
|  | **M** |  | **Z** |

*Figure 3.3: Data set images.*

Table 3.1: Dataset Description and Image property

|  |  |
| --- | --- |
| **Property** | **Description** |
| Alphabets | A to Z |
| Numbers | 0 to 9 |
| Color | Gray Scale |
| Dimensions | 50x50 |
| Height | 50 pixels |
| Width | 50 pixels |
| File type | JPEG |

**3.5 Capturing Images for Dataset**

To detect hand gesture using skin colour, there are different approaches including skin colour-based methods.In this thesis, after detecting and subtracting the face and other background, skin recognition and a contour comparison algorithm were used to search for the hand and discard other background colour objects for every frame captured from a webcam or video file.Hand to extract their contours and saved the further for evaluation with the contours of the skin detected area of every frame.After detecting the skin area for each frame captured, the contours of the detected areas were compared with the previously saved hand histogram template contours to remove other skin like objects existing in the image.If the contour comparison of the spotted skin area complies with any one of the saved hand histogram contours, then it captured hand gesture only.

**Chapter 4**

**Hand Gesture Detection**

**4.1 Introduction**

Object detection plays a very important role in my own dataset. To analyze and extract relevant data about an object of interest from an image, one needs to first get that object in the image. Hand posture detection refers to finding the place and size of hand within a sequence of images. Nowadays it is a very popular problem in computer vision problem and has many numerous applications, such as gesture recognition, sign language recognition, computer graphics games, and human-computer interaction (HCI).

Skin colour [21, 22, 23] is an important property to detect hand and tracking. However, Colour image has the different problem of removing other objects with similar colours such as face and human arm. To solve this problem, we introduce a new method to detect hand postures only using face detection and subtraction, skin detection, and hand postures contour detection and comparison algorithm [24]. The face was detection removed because the skin detection will detect the face and the face’s contours are very similar to the first-hand gesture contours. An interest of the area has been captured and another same skin colour has been removed from an interest of the area. After removing an unwanted area of the face, I detected the skin area using the hue, saturation, value (HSV) colour model since it has real-time performance and it is strong against alternations, scaling and lighting conditions. Then, the interested area of contours was compared with all the existing hand posture template contours to eliminate unwanted interest of area like objects existing in the image.

**4.2 Hand Detection Approaches**

There are different approaches for hand detection have been introduced in the literature that employs different visual features and, in many cases, their combination. These are different approaches such as motion, skin colour, shape, and 3D models of hands. Hand detection methods were discussed in [25] and will be discussed later in this chapter.

**4.2.1 Colour**

Skin colour detection has been used in many hand gesture recognition projects. The main objective of giving a model of skin colour is the choice of the colour space to be utilized. Different colour spaces have been introduced such as RGB, normalized RGB, HSV, YCrCb, and YUV. Colour spaces that effectively divide the chromaticity from the luminance parts of colour are typically regarded as preferable as I did in my approach by removing the Value (V) section in the HSV model. This is because of employing chromaticity-dependent mechanisms of colour only.

Generally, skin colour detection can be disordered by background objects that have a skin colour distribution like human skin colour. Some project has been done on this problem by using background subtraction [26, 27]. On the contrary, it was expected that unwanted background subtraction normally depends on the camera system that does not move with respect to a fixed background. Another solution [28, 29] has applied the dynamic modification of background compensation techniques.

**4.2.2 Shape Hand**

Shape property has been working to discover in frames. More details can be acquired by reducing the contours of objects in the frame. If the contour is perfectly detected, it provides a good presentation of the hand gesture which is indirectly related to a viewpoint, skin colour, and lighting. Typically, contour extraction based on edge recognition uses many edges fitting to the hand image area but also to distinct background objects.

Accordingly, sophisticated post-processing techniques are needed to develop the presentation of this method such as our approach in [24] by combining skin colour detection with contours detection and comparison after face subtraction. A second method that has been used in fingertip finding is pattern matching. Patterns can be images of fingertips [30] or fingers [31] or generic 3D cylindrical models [32]. These pattern matching methods can be upgraded by using extra image features such as contours [26].

**4.2.3 Learning Detectors from Pixel Values**

Currently, this technique that uses a machine learning method named posting has shown remarkably strong results in face recognition and good results in hand recognition [33]. In [34], an object recognition method was proposed in which a weak classifier may be a simple finder that uses basic image block differences efficiently calculated using an integral image. On the other hand, this technique may provide an unnecessary number of weak classifiers. The AdaBoost technique has a drawback because it does not consider the removal of chose weak classifiers that no longer take part in the recognition procedure. Also, there is some problem to distinguish the hand using the Viola-Jones method [34, 35] related to rotation and cluttered background.

**4.2.4 3D Model-Based Detection**

One major advantage of a 3D model-Based method is that it can allow for view-independent detection. The used 3D models must have enough degrees of freedom to adapt to the dimensions of the hand that exist in an image. Different mock-ups use different image features to build feature-model communications. Line and point features are applied in kinematic hand models for recovering angles created at the links of the hand [36, 37, 38]. The hand gesture is then estimated based on the relations between the 3D model and the observed image features.

**4.2.5 Motion**

Motion is a dynamic feature employed by some procedures for hand detection. Motion based recognition requires a highly controlled arrangement, and it adopts that the only movement in the image resulted from hand motion. In the case of fix cameras, the issue of movement assessment is solved with background maintenance and successive removal. This explanation is employed in [50,51] for recognizing the hand from other skin- coloured objects, and for testing with lighting conditions resulting from coloured lights. The difference in pixel strength between two consecutive frames is close to zero for the background pixels. Motion objects are placed by selecting and maintaining a suitable threshold.

**4.3 My Approach for Hand Detection**

 I propose an integrated system for detection, segmentation, and tracking of the hand in a gesture recognition system using a single webcam. Some other methods that use colour gloves [39, 40], my method can detect the plain hand posture by integrating two useful features: skin colour detection and contour matching. My proposed hand posture, finding algorithm has real-time performance and is strong against rotations, scaling, a cluttered background, and lighting conditions. Section 4.4.2 shows the strength of my proposed hand posture detection algorithm based on comparison with other different methods. Detecting the human hand in a plain background will boost the performance of hand gesture recognition systems. In this method, the speed and result of recognition will be the same for any frame size taken from a webcam such as 640×480, 320×240 or 160×120 and the system will be also robust against a plain background because I process the detected hand posture area only. A smaller image size that holds the detected hand posture area only must be like the training image size of a training stage as I will discuss the training and testing recognition systems stages in section.

 To detect the hand gesture in the image, a four-phase system was designed according to my approach and as shown in Figure 4.1. First, we will open camera which has a square box to capture hand gesture. Second Put your hand in those boxes and make sure your hand covers within the square box. Third, the skin colour locus for the image was removed from the user’s skin colour after face deletion. Then the last step, the hand gesture was spotted by removing false positive skin pixels and identifying hand gesture and other real skin colour regions using contours matching with the loaded hand gesture pattern contours. Skin Recognition Area Loading Hand Postures Patterns Contours Face Detection and Subtraction Capturing Images from Webcam or Video file Templates Contours Comparison with Skin Area Figure 4.1: Hand posture detection steps

*Figure 4.1 Hand posture detection steps*

**4.3.1 Skin Detection**

Skin detection is a useful approach for many computer vision applications such as face recognition, tracking and facial expression, abstraction, or hand tracking and gesture recognition.There are recognized procedures for skin colour modelling and recognition that will allow to differentiate between skin and non-skin pixels based on their colour**.**To get suitable distinction between skin and non-skin areas, a colour transformation is needed to separate luminance from chrominance [42].

The input images normally are in Colour format (RBG), which has the drawback of having components dependent on the lighting situations.The misunderstanding between skin and non-skin pixels can be decreased using colour space transformation**.**There are different approaches to detection skin colour components in other colour spaces, such as HSV, YCbCr, TSL or YIQ to provide better results in parameter recovery under changes in lighting condition. Researches have shown that skin colours of individuals cluster closely in the colour space for all people from different societies, for example, colour appearances in human faces and hands vary more in intensity than in chrominance [41, 43].Thus, take away the intensity V of the original colour space and working in the chromatic colour space (H, S) provides invariance against illumination situations. In [42], it had been well-known that removal the Value (V) component and only using the Hue and Saturation components, can still permit for the detection 96.83% of the skin pixels.In my application, I use the hue, saturation, value (HSV) colour model since it has shown to be one of the most adapted to skin-color detection [44].It is also well-matched with the human colour perception. In addition, it has real-time execution and it is more robust in cases of rotations, scaling, cluttered background, and changes in lighting condition.So, my projected hand gesture detection algorithm is real-time and robust against the mentioned previous changes.The other skin like objects existing in the image are removed from contour comparable with the loaded hand postures prototype contours**.**

The HSV colour space is gained by a nonlinear transformation of the essential RGB colour space.The conversion between RGB and HSV was described in [45].Hue (H) is a section that characterizes a pure colour such as pure yellow, orange or red, whereas saturation (S) provides a measure of the degree to which a pure colour diluted by white light [46]. Value (V) attempts to represent brightness along the gray axis such as white to black, but since brightness is subjective, it is thus difficult to measure [46].

According to [47] and Figure 4.2, Hue is estimated in HSV colour space by a position with Red starting at 0, Green at 120 and Blue at 240 degrees.The black mark in the diagram at the lower left on the screen determines the hue angle.

Saturation is a ratio that ranges between 0.0 along the middle line of the cone (the V axis) to 1 on the edge of the cone.Value ranges, string from 0.0 (dark) to 1.0 (bright).



*Figure 4.2 HSV Colour Space [47]*

According to [41],the HSV model can be resulting from non-linear transformation from an RGB model according to the following calculations.



4.1

4.2

4.4

4.3

As per a classification point of view, skin-color detection divided into two class problem: skin-pixel vs non-skin-pixel classification. Currently, there are different known classification approaches exits such as thresholding, Gaussian classifier, and multilayer perceptron [48, 52, 53].

In my research, I used a thresholding technique that allows getting a good result for higher computation speed when compared with other techniques, given our real-time requirements. This thresholding classification is used to find the values between two components H and S in the HSV model as I removed the Value (V) component. Usually, a pixel can be observed as being a skin-pixel when the following threshold values are synchronized satisfied: 0° < H < 20° and 75° < S < 190°.

**4.3.2 Contour Comparisons**

Once the skin colour has been detected, the contours of the detected skin colour are recovered and then compared them with the contours of the hand gesture patterns. Once skin colour contours are recognized as belonging to the hand gesture contour patterns, that area will be identified as a region of interest (ROI) which will then be used for tracking the hand movements and saving the hand posture in JPEG format in small images as shown in Figure 4.3. After that stored images will further be used to extract the features needed to recognize the hand postures in the testing stage.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

*Figure 4.3: Images of detecting hand postures.*

If there is both hand gesture in the image, my system will substitute in detecting one of the two hands for every frame captured because the Open Computer vision function cvBoundingRect will circle one rectangle only around the detected hand, which has the main matching contours with the overloaded hand posture templates contours. The single frame will circle the detected hand posture for one frame and may enclose the other hand posture for the next frame if it has a higher matching contour.

**Chapter 5**

**Machine Learning Approach**

**5.1 Artificial Intelligence**

Research in Artificial Intelligence (AI) is very general and spans across numerous different area such as mathematics, computer science, philosophy, economics and even ethics. This field is very wide and can be attempted from many different viewpoints. Therefore, this explanation will not be very exhaustive. For comprehensive and more complex joint into subject, refer to [54].

One of many possible definitions of AI can be brief as a search to develop an artificial intelligent agent. In other words, it is an effort to create intelligent machines that are either intelligent or can be perceived as intelligent ones. One of the most important skills of intelligent agents is the sense of vision. Sense of vision is usually required for positive degree and not always it is necessary that it rivals the abilities of the human visual apparatus.

First tries to resolve the vision problems were tackled from the so-called bottom-up approach in which the system was instructed with a hard-coded set of protocol describing the vision. It was expected that as the understanding of an instrument allowing humans to abstract information from visual scene, the hard-coded systems can be fed this understanding and thus more skilled systems can be created. The problem with this method was that it highly underestimated the difficulty of reinforcement of these protocols. Therefore, it mainly failed to devise such a system.

This main idea the investigators to postulating that in order to resolve the problem of deploying vision capabilities for the artificial intelligent system, it is compulsory to introduce a procedure that would allow AI systems to extract patterns from providing data. It is an overview of systems that can learn. A process that enables systems to learn is usually called machine learning.

Machine learning is again relatively extensive term that can be used in multiple different frameworks. In this work, it is meant to be understood as a technique that is used to create mathematical representations used for image detection. There are numerous types of machine learning models that are useful for different tasks. The task that is conversed in this work and is also arguably most commonly attempted is called classification, which is the task to classify the occurrence of input into correct discreet and mainly predetermined class. One most common type of machine learning task is called regression, which is based on the input data trying to estimation unknown continues valued quantity.

**5.2 Image Processing**

Computer vision is an important topic in image processing. Investigation on the subject was approaching the problem from bottom-up perception for many years. This was the before-mentioned effort to validate directions guiding the vision of living organisms. This method was actually very successful in certain environments.

General explanation of computer vision in image processing can be brief in following steps:

• Image capture - Image is captured using a camera or similar device and digitized.

• Pre-processing - Captured image is modified to highlight important features such as noise reduction, contrast normalization etc.

• Segmentation detection - Selection of region of interest like edges, similar surfaces.

• Description – Feature extraction of radiometric, photometric descriptors and so on.

• Classification - Some means to categorize the object.



*Figure 5.1: Diagram of image processing pipeline. [55]*

Specific steps are shown in figure 5.1. Even though machine learning been as a field of study since the second half of the 20th century, there was no wider implementation of its techniques in image processing for a very long time. It was first announced in the classification step of the processing pipeline. In other words, complex difficulties were simplified by reducing the visual information contained within the image into a handful of simple features that were nourished into machine learning model.

This carries the problem that these applications are not very useful. Each application is usually only capable of resolving the very narrow problem and any deviation from ideal circumstances can mean failure. An application can have complications with varying contrast, brightness, scaling, rotation etc.

The second problem is often the fact that because the image must be pre-processed numerous times before it is fed into the machine learning model it requires additional time and computational resources. This is less of a problem with existing hardware innovations, but it is still not unimportant factor and it can have a negative effect on the cost of the result. This is where machine learning in general indicates a noteworthy advantage.

Conventional approach to computer vision can find success in applications that are deployed in very limited environments with rigid constraints. In controlled location, it is usually very simple to define the problem in formal guidelines. Even though it can be still available in certain places, it starts to be forced out by application of machine learning simply because the barrier of entry for extensive adoption reductions every day.

**5.3 Machine Learning**

As before described, Machine Learning is a process that is used to create models that can abstract information from data to resolve given problem and consequently repeatedly improves their performance.

Interesting viewpoint that can be used is to view machine learning as a form of information compression. Where the machine learning model is trying to abstract information from input data in such a way that the amount of a data used to save is summary while the information contained within is preserved.

**5.3.1 Machine Learning Approach**

There are mainly two different types of machine learning methods:

• Unsupervised Learning

• Supervised Learning

Both are mainly used for different types of machine learning tasks.

**Unsupervised learning**

In unsupervised learning method, the model is trained by detecting new data and take out patterns in the date without being instructed on what they are. Contrasting to supervised learning defined below, the benefit of this method is that the model can learn from data without supervision. This means that there is no need for input data to be explained, therefore it takes a smaller amount time and resources to deploy these models in practice.

The biggest difficulty of supervised learning method in real world applications is to obtain appropriate data. Appropriate data in this context means, data that were someway classified into different categories, which can be very boring and slow process. In some situation, the task itself avoids the usage of labelled data. Labelled data are impossible to find or don’t exist at all.

Mainstream of unsupervised learning procedures belongs to a group called clustering algorithms. These algorithms are centered on the idea to analyze ordered clustering of data in input space to determine their relationship. This is achieved by the belief that data point clustering in input space is likely to exhibit like properties.

Illustrations of unsupervised learning models are:

• K-means -clustering model

• Self Organizing Maps (SOMs)

• Principal Component Analysis (PCA) - dimensionality reduction

Image classification usually does not depend on the use of unsupervised learning methods; therefore, the following writing describes only supervised learning methods.

**Supervised learning**

Supervised learning method is more commonly used. This method needs training data with a specific format. Each instance must have assigned label. These labels make available supervision for the learning algorithm. Training process of supervised learning is constructed on the following principle. First, the training data are fed into the model to produce estimates of output. This estimate is compared to the assigned label of the training data in order to evaluate model error. Based on this error the learning algorithm alters model’s parameters in order to reduce it.

**5.3.2 Structure of Machine Learning Algorithm**

Although machine learning algorithms are various and are using different methods its structure can be generalized. Structure of nearly all machine learning algorithms can be defined as composed of the following components:

• Dataset description

• Model

• Cost function

• Optimization technique

Almost all supervised learning algorithms use the same Dataset description. The other three components can vary intensely. This level of analysis is suitable for building of intuition for Neural Networks (NNs) and description of its individual components.

**Dataset description**

Supervised learning requires data sets with detailed properties. Each dataset holds set of 𝑛 instances which contains of a pair of input vector 𝑥𝑖 and output scalar 𝑦𝑖. Input vector

(5.1)

Where 𝑖 is an index of instance, 𝑝, is length of input vector

Specific components of input vector must be of unified type. In case of image as input data it is the value of individual pixels (e.g. 0-255). In other cases, they could be real values. Almost commoner in machine learning, it stands that input should be normalized. This belief holds in images automatically since each pixel must have its values in a fixed range. It is very significant in other types of machine learning tasks, where this is not guaranteed.

Output scalar 𝑦𝑖 characterized class of the given instance. The type of this output value, thus must obtain only certain values. To put it differently, it must be a set of cardinalities equal to the number of all possible classes.

**Model**

The model is predicted tackle that takes input 𝑥𝑖 to predict values of its output 𝑦𝑖. For each model has parameters represented by vector 𝜃, which are used to during the training process. The modest example of model type is a linear model, also called linear regression. Parameters 𝜃 of this model year

, (5.2)

Where 𝑝 is the number of parameters equal to size of input vector 𝑥𝑖. Prediction 𝑦^𝑖 of the model on instance 𝑖 is computed as

𝑦𝑖 = (5.3)

Therefore, estimates of the model on the entire dataset in matrix notation are

𝑦^ = 𝑋𝜃. (5.4)

Estimates in expanded notation are equal.

**Cost Function**

To achieve the learning skill of the machine learning algorithm, it is necessary to approximation the error of its predictions. This is assessed with so called cost function and called loss function.

This function must have certain properties. The ability of the machine learning algorithm to learn rests on the approximation of its improvement with the change of its parameters. Therefore, cost function must be at least partially differentiable. In case of linear regression, it is most common to the usage sum of square error. The main aim being that derivative of this function for linear model has only one global minimum.

The cost function is defined as

𝐽(𝜃) =(𝑦𝑖 − ^𝑦𝑖)2 = (︁𝑦𝑖 − 𝑥𝑖𝑇𝜃)︁ (5 . 6)

For the optimization determinations it is usually useful to express the cost function in matrix notation

𝐽(𝜃) = (𝑦 −𝑋𝜃)𝑇 (𝑦 −𝑋𝜃) (5.7)

**Optimization technique**

The last part of learning algorithm is the optimization technique. It consists of an update of model’s parameters 𝜃 in order to progress its prediction. In other words, to find 𝜃 such that the value of cost function 𝐽(𝜃) for given dataset is as small as possible.

To examine the change of cost function on given dataset it is necessary to compute the derivative of 𝐽(𝜃) with respect to 𝜃

𝜕𝐽(𝜃)/𝜕𝜃= 𝜕/𝜕𝜃 [︁(𝑦 −𝑋𝜃)𝑇 (𝑦 −𝑋𝜃)]︁

= 𝜕/𝜕𝜃 [︁𝑦𝑇𝑦 + 𝜃𝑇𝑋𝑇𝑋𝜃 − 2𝑦𝑇𝑋𝜃]︁

= 2𝑋𝑇𝑋𝜃 − 2𝑋𝑇𝑦 (5.8)

For linear model is possible to find optimal result which is a global minimum of the cost function. The optimal result

𝜃 =(︁𝑋𝑇𝑋)︁−1𝑋𝑇𝑦 (5.9)

is found by comparing the partial derivative of 𝐽(𝜃) to 0. The only condition is that 𝑋𝑇𝑋 must be non-singular.

Unfortunately, only very simple problems can be like using the model as simple as linear regression. More complex model usually means more complex cost function. The optimization process of more complex cost functions cannot be definite to find the global minimum. In this case, the optimization technique must be about iterative character. To put it in a dissimilar way, the algorithm has to method the minimum of iterations. Many of the iterative approaches belong to the group called gradient based optimization.

**5.3.3 Model Complexity**

In the first calculation, it could be said that the task of supervised machine learning is to model the relationship between input output data most correctly. The problem with this definition is that in the real-world application, there have never been enough data to capture the true relationship between the two. Therefore, the task of machine learning is the attempt to suppose the true relationship by detecting incomplete picture.

Hence the most significant property of the machine learning model is its generalization capability. That is capability to produce meaningful outcomes from data that were not previously detected.

Generalization ability is reliant on the complexity of the model and its relationship to the complexity of underlying problem. When model does not capture the complexity of the problem appropriately it is defined as under fitting. In case the complexity of model surpasses the complexity of the underlying problem, then this phenomenon is called over fitting.

In both extremes the generalization ability suffers. In the earlier case the model is unable to capture true intricacies of the problem and therefore is unable to predict wanted output reliably. In the last case it tries to capture even the subtlest data perturbation that might be in fact an outcome of the stochastic nature of the problem and not the real underlying relationship. This can also cause the fact that input data is lost some variable necessary to capture the true relationship. This fact is inescapable, and it thus must be careful when designing machine learning model. Representation of this phenomena in the case of two variable inputs is in Figure 5.2.



*Figure 5.2: Figure shows different levels of generalization of the model [56]*

Typically, the machine learning model is trained on as much input data as possible in order to reach the best possible performance. At the same time its error rate must be verified with independent input data to check whether the generalization ability is not deteriorating. This is typically accomplished by splitting available input data into training and testing set. Frequently in 4:1 fraction of training to test data. The model is trained with training data only and the presentation of the model tests on the test data. Connection between test and train error can be found in Fig. 5.3. Even though the true generalization error can never be truly detected, its estimate of the test error rate is enough for most machine learning tasks.

**Regularization**

Regularization is any alteration that is made to the learning algorithm that is intended to decrease its generalization error, but not its training error [57]. As it has already been stated, the most significant aspect of machine learning is striking the stability between over and under fitting of the model. To support with this problematic concept of regularization was devised. It is a method that helps to penalize the model for its complexity. Basic idea consists of adding a term in the cost function that increases with model complexity. When this is applied to cost function from equation 5.7

𝐽(𝜃) = (𝑦 −𝑋𝜃)𝑇 (𝑦 −𝑋𝜃) + 𝜆𝜃𝑇𝜃 (5.10)

Where 𝜆 is a parameter that controls the strong point of the preference [57].



*Figure 5.3: Relationship between the model complexity and its ultimate accuracy is the relationship between training and testing error [58].*

**CHAPTER 6**

**NEURAL NETWORKS**

This chapter is dedicated to description of NN in general and its special type called CNN.

**6.1 History**

History of Neural networks can be arguably dated from 1943, when Warren Mcculloch and Walter Pitts invented mathematical model encouraged by the Biology of central nervous systems of mammals [59].

This encouraged the invention of Perceptron, created in 1958 by Frank Rosenblatt. Perceptron used very modest model mimicking biological neuron that was based on the mathematical model of Pitts and Mcculloch. Definition of the Perceptron model also defined an algorithm for direct learning from data.

In the beginning Perceptron looked very promising, but it was soon discovered that it had severe restrictions. Most projecting voices of criticism were Marvin Minsky. Minsky published book in which he laid out a case that the Perceptron model was unable to resolve complex problems [60]. Amongst others the book contained mathematical proof that Perceptron is incapable to solve simple XOR problem. More generally the Perceptron is only proficient of solving linearly separable problems. However, according to Minsky, this criticism wasn’t malicious, it in effect stifled the interest in NNs for over a period.

Awareness in NNs was rejuvenated in the early 80’s, when it was shown that any previously raised up deficiencies could have been resolved by usage of multiple units. This was later exacerbated by development of back-propagation learning algorithm, which allowed the possibility to gather neurons into groups called layers, which can be weighted into hierarchical structures to form a network. NN of this type were generally called Multilayer Perceptron (MLP). In 80s and 90s the awareness in NNs plateaued again and general research on AI was more focused on other machine learning methods. In the field of classification problems, it was particularly SVM and ensemble model. AI research communities also established several other paradigms of NNs that were likewise inspired by Biology of certain aspect of central nervous system but took different methods. Most significant examples were SOM and Recurrent Neural Network (RNN).

By the year 2000, there were very few research groups that were applying enough attention to the NNs. There was also a certain disdain for NNs in the academic world and AI research community. The success of NNs that was promised almost half a century ago was finally coming across around 2009, when the first networks with huge number of hidden layers were effectively trained. This led to typical adaptation of umbrella term deep learning which by and large refers to Deep Neural Network (DNN). The term deep indicates that networks have many hidden layers

The key theoretic vision was to learn complex functions that could represent high-level abstractions such as vision recognition, language understanding, etc. There is a requirement for deep architecture.

NNs in the times before Deep Neural Networks had only one or two hidden layers. These are currently called shallow networks. Typical Deep Networks can have a number of hidden layers in order of 10’s, but in some cases even hundreds [61]

Still, progress of Neural Network into the direction of structures with high number of hidden layers was obvious, its training was an unresolved technical problem for a very long time. There were fundamentally three reasons why this invention didn’t come sooner

1. There were no procedure allowing the number of hidden layers to measure.

2. There wasn’t enough of labels data required to train the NN.

3. The computer hardware wasn’t powerful enough to train adequately large and complex networks successfully.

The first problem was tackled by the creation of CNNs [62]. The second problem was explained simply when there were more data presented. This was primarily achieved thanks to effort by large companies like YouTube, Google, Facebook, etc. But also, with support of large community of experts and hobbyists in data sciences.

Both inventions in computational hardware and improvement of training methods were needed to resolve the third problem. One of the technical revolutions was use of Graphics Processing Units (GPUs) for the demanding computation involved in the training of a complex network. Thanks to the fact that training process of NNs is typically large number of simple resulting computations, there is a great possibility for parallelization.

**6.2 Structure of Neural Networks**

The term NN is very general and it defines a comprehensive family of models. In this framework NN is distributed and parallel model that is capable of approximating complex nonlinear functions. Network is made from multiple computational components called neurons assembled topology.

Explanation of NN structure will follow the convention laid out in the explanation of learning algorithm. Meaning that an explanation of the learning algorithm is composed of model, cost function and optimization technique. The difference comes into performance with the fact that the model of NN is much more complex than the model linear regression. Therefore, the investigation is divided into model of neuron and topology of the network.

**6.2.1 Model of Neuron**

The neuron is a computational unit carrying out the nonlinear transformation of its inputs

*𝑦 = 𝑔(𝑤𝑇𝑥 + 𝑏) (6.1)*

Argument 𝑤𝑇𝑥 + 𝑏 of function 𝑔 is often observed as 𝑧. Therefore, the equation can be rewritten as

*𝑦 = 𝑔(𝑧). (6.2)*

Typical schema is shown on Figure 6.1, which describes the inputs, weights bias and activation function.



*Figure 6.1: Diagram of artificial neuron [63].*

As it was already stated model of neuron was stimulated by biology. First attempts to make a model of a neuron had multiple elements equivalent with neurons of the human brain. As research proceeded this equality ceased being as important and modern NN models correspond to their biological matching part only superficially.

**Inputs**

Each neuron has multiple inputs 𝑥 that are combined to execute some operation. Each input has elected weight assigned to it.

**Weights**

Inputs of a neuron are weighted by parameters 𝑤 that are changed during the learning process. Each weight gives strength to each individual input into the neuron. The basic awareness is that when the weight is small the input doesn’t affect the output of the neuron very much. Its effect is large in the opposite case.

**Bias**

Another changeable parameter is bias 𝑏 that controls impact of the neuron.

**Activation Function**

For NN to estimate nonlinear function each neuron must perform the nonlinear transformation of its input. This is completed with activation function 𝑔 (𝑧) that performs the nonlinear transformation. There are numerous different normally used activation functions. Its usage depends on the type of network and on the type layer in which they activate.

One of the oldest and historically most frequently used activation functions if sigmoid function. It is defined by

*𝑔(𝑧) = 1/1 + 𝑒−𝑧 (6.3)*

Problem with sigmoid is that its gradient becomes flat on both extremes and as such it reduces the learning process [64].

One more activation function is the hyperbolic tangent. It is defined as

*𝑔(𝑧) = 𝑡𝑎𝑛ℎ(−𝑧). (6.4)*

The hyperbolic tangent function doesn't use that much in feed forward NN, but it is mostly used in RNN. Currently, most commonly used activation function is restricted Linear Unit (ReLU). It is very generally used in both convolutional and fully connected layers. It is defined by

*𝑔(𝑧) = max{0, 𝑧}. (6.5)*

It has a disadvantage because it is not differentiable for 𝑧 = 0, but it is not a problem in software execution and one of its biggest advantages is that it can learn very speedily.

All three activation functions are illustrated in Figure 6.2.

**6.2.2 Topology of the Network**

There are several different generally used topologies. The two most frequently used in deep learning are feed-forward and recurrent. Feed forward networks are categorized by the fact that during activation the information move only in a forward direction from inputs to outputs. A recurrent network has some sort of feedback loop.

Another principle of topology is how are individual neurons in the network linked. Most commonly are NNs ordered in layers. In each layer there can be from one to n neurons. Layers are hierarchically fixed.  The first layer is called input layer, the last layer is called an output layer and the layers intermediate are called hidden.

Description of the network recreations on interconnections between individual layers. The most common structure is called fully connected where to each neuron in hidden layer 𝑙 has input associates from all neurons from previous layer 𝑙 − 1 and its output is associated with input of each neuron in following 𝑙 + 1 layer. The entire structure is illustrated in Figure 6.3.



*Figure 6.2: Activation Functions*

After this point on the term NN will refer to Feed-forward Fully Connected Neural Network.

Types of neurons are dependent on the type of the layer. Currently the core difference is in their activation function, which wasn’t the case for a long time. In history all layers had neurons with sigmoid activation function. It was mostly because the output sigmoid layer can be easily mapped onto probability distribution, since it obtains values between 0 and 1. Only relatively recently it was found that network composed of neurons with ReLU activation function in the hidden layers can be trained very speedily and are more resistant against over-fitting. Activation functions are still subject of ongoing research.

Neurons in output layer necessity output that can produce probability distribution



*Figure 6.3: Fully connected Feed Forward Neural Network [65].*

that can be used for approximation the probability of individual classes. For this reason, most frequently used activation function of output neuron is called SoftMax.

SoftMax is standardized exponential function. It is used to represent probability of an instance existence member of class 𝑗 as

*(6.6)*

where 𝐾 is total number of classes.

**6.2.3 Cost Function**

Cost functions of NNs are a complex subject that exceeds the scope of this thesis. One of the most common cost functions used in NNs for classification in multiple classes is categorical cross entropy. In SoftMax activation function from Equation 6.6 is a cost function defined as

Where 𝑦 (𝑖) if the correct class of the instance and 𝑛 is the total number of instances.

**6.2.4 Optimization Procedure**

Every optimization technique for NN is constructed on gradient descent. In other words, it is an iterative process that goes to lower training error of the network by differentiating of the cost function and adjusting parameters 𝜃 of the model by following the negative gradient.

The problem is that cost function of the whole network is very complex and has many parameters. To find the gradient of the cost function, it is compulsory to go through all the units in the network and estimate their contribution to the overall error. A method that is used to solve this problem is called back-propagation.

Back-propagation if frequently confused to be a complete learning algorithm which is not the case, it is only the method to compute the gradient [57].

**Back-propagation**

To approximation the influence of individual units in a network the back-propagation is used to compute delta 𝛿 𝑙 𝑗, where 𝑙 is layered and 𝑗 is an index of neurons in that layer. The algorithm starts at the output of NN, more exactly its cost function.

*𝛿𝐿 = ∇𝑥𝐶 ⊙ 𝑔'′() (6.8)*

Where 𝐿 is the last layer of the network and ∇𝑥𝐶 is the gradient of the cost function with respect to 𝑥 and ⊙ is the Hadamard product.

In subsequent lower layers the deltas are calculated as

where (𝑤𝑙+1) 𝑇 is from Equation 6.1.

Each neuron has two changeable parameters 𝑏 and 𝑤. To estimate the rate of change in parameter 𝑏 𝑙 𝑗 from Equation 6.1 it needs to be computed

*(6.10)*

Change of weight 𝑤 𝑙 𝑗𝑘 from Equation 6.1 it needs to be computed as

*(6.11)*

**Gradient Descent Optimization**

Back-propagation estimations gradient of all modifiable parameters 𝑏 and 𝑤 in the network. These parameters can be denoted by vector 𝜃. Therefore, the gradient of the function to be minimized can be written as ∇𝜃𝑡−1 𝑓(𝜃𝑡−1).

The modest learning algorithm is called gradient descent. Even though simple, it is a very robust learning algorithm.

*(6.12)*

*(6.13)*

The algorithm has meta-parameter 𝜂, which is often called learning rate. It determines how fast are 𝜃 parameters updated. Modest gradient descent has the shortcoming that update of parameters has been always closely proportional to change of gradient. This might turn out to be a problem when the gradient change slows down. This process is also often called Stochastic Gradient Descent (SGD). The word stochastic indicates that during training the algorithm is using a random choice of instances to train. There are various variations on the gradient descent method. Following explanations are taken from [66].

**Adam**

It is more complex learning algorithm that combines 𝐿2 norm and classical momentum-based optimization. It should converge quicker than classical Gradient Descent.

*)*

*(6.16)*

**6.3 Convolutional Neural Networks**

CNNs are a specialized type of NNs that was initially used in image processing applications. They are arguably most effective models in AI inspired in biology.

Even though they were shown by many different fields, the main design principles were drawn from neuroscience. Since their achievement in image processing, they were also very successfully implemented in natural language and video processing application.

Aforementioned stimulation in biology was based on the scientific work of David Hubel and Torsten Wiesel. Neurophysiologists Hubel and Wisel, studied vision system of mammals from late 1950 for several years. In the research, that might be measured little gruesome for today’s standards, they linked electrodes in brain of an anesthetized cat and measured brain response to visual stimuli [67]. They discovered that feedback of neurons in the visual cortex was triggered by a very narrow line of light shined under a specific angle on a projection screen for the cat to see. They determined that specific neurons from visual cortex are responding only to very specific patterns in the input image. Hubel and Wiesel were given the Nobel Prize in Physiology and Medicine in 1981 for their discovery.

In the following text is assumed that convolutional layer is working with rectangular input data (e.g. images). Even though the Convolutional networks can also be also used to classify one-dimensional or three-dimensional input.

**6.3.1 Structure of CNN**

The structure of Convolutional networks is typically made of three different types of layers. Layer could be either Convolutional, Pooling or fully connected. Each type of layer has different protocols for forward and error backward signal propagation.



*Figure 6.4: Structure of Convolutional Neural Network [68]*

There are no specific protocols on how the structure of individual layers should be organized. However, with exemption of recent development CNNs is typically structured in two parts. First portion, usually called feature extraction, is using groupings of convolutional and pooling layers. Second portion called classification is used fully connected layers. This is shown in Figure 6.4.

**Convolutional layer**

As the name suggests this layer employs convolution procedure. Parameters supply to this layer is simply called input. Convolution process is performed on input with specific filter, which is called the kernel. The output of convolution process is typically called feature map.

Input in Convolutional layer can be imaged or feature map from the previous layer. The kernel is typically of rectangular shape and its width can array from 3 to N pixels. Feature map is created by convolution of kernel over each definite element of the input. Convolution is defined in more detail in the section describing training of CNN.

Depending on the dimensions of kernel and layer’s padding preferences the process of convolution can produce feature map of different size than input. When the size of output should be preserved, it is compulsory to employ zero padding on the edges of the input. Zero padding in this case must add the essential amount of zero elements around the edges of the input. This amount is determined by

*p =((h−1)/2) (6.19)*

where h is width of used kernel. In contrary case the feature map is reduced by the 2p. Decrease of the size of the feature map can be in some cases desirable. Zero padding is shown on Figure 6.5.

Drop of feature map can go even further in the case of use of stride. Application of stride specifies by how many input points are navigated when moving to neighboring position in each step. When the stride is one, kernel is moved by one on each step and the resulting size of feature map is not affected. Each Convolutional layer is typically configuration of several different kernels. In other words, the output of this layer is the tensor holding feature map for each used kernel. Each of these is premeditated to underline different features of input image. In the first layers these features are typically edges. The next layer, the more complex features are captured. Each kernel that is used is applied to all inputs of the image to produce one feature map which basically means that neighboring layers are distributing the same weights. This might not be appropriate in some applications and therefore it is possible to use two other types of networks. Locally linked which basically means that applied kernel is about the same size as the input and smooth convolution which means alternation of more than one set of weights on entire input.

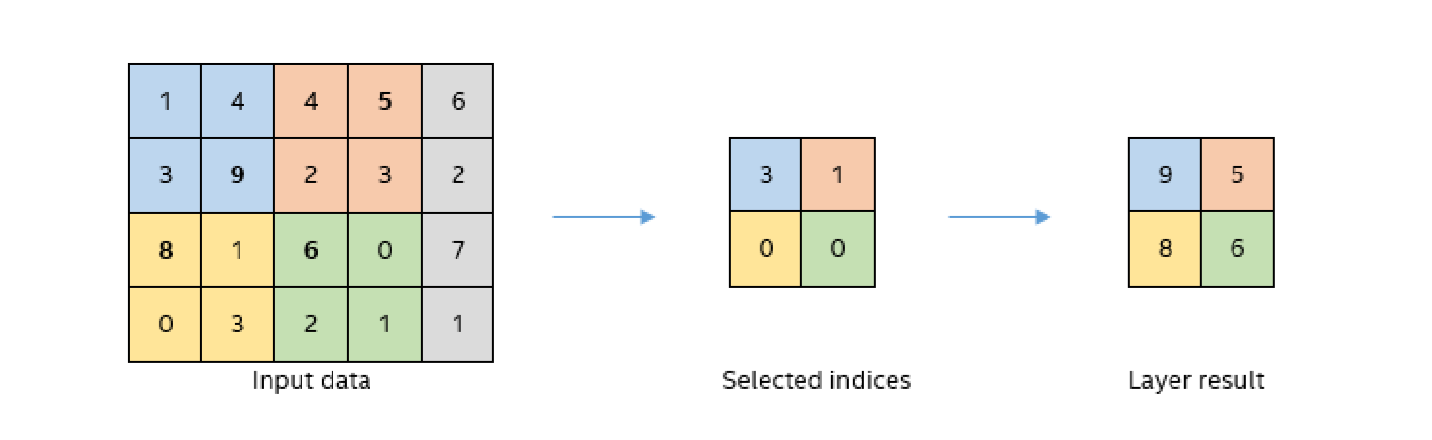


*Figure 6.5: A zero padded 4x4 matrix [69]*

Smooth convolution is interesting because with clever combinations with Max Pooling clarified bellow it allows to train specific feature from multiple positions. Each convolutional layer has non-linearity of its output that is sometimes also called the detector phase. This is equal to the activation function of NNs. The activation function of CNN is normally ReLU.

**Pooling layer**

This layer typically doesn’t find any learning process, but it is used for the dejected sample size of the input. The Principle is that input is divided into multiple not overlapping rectangular elements and units within each element are used to make a single unit of output. This reduces the size of the output layer while preserving the most important information checked in the input layer. In other words, pooling layer wrappings information contained within the input. Type of process that is achieved on each element determines a type of pooling layer. This action can be averaged over units within an element, selecting a maximal value from an element or alternatively learned a linear combination of units within the element. Learned linear combination introduces forms of learning into the pooling layer, but it is not very predominant. Selecting of maximal value is record common type of pooling operation and in that case the layer is called Max-Pooling accordingly. The positive outcome of Max-pulling down-sampling is that take out features that are learned in convolution are invariant to small shift of the input. Principle of Max-Pooling is shown on Figure 6.6.



*Figure 6.6: Principle of Max-pooling [70]*

As already said, another benefit of Max-pooling arises when combined with tiled convolution. To generate simple detector that is invariant to rotate it possible to use four different kernels that are rotated by 90 degrees among each other and when the smooth convolution is used to tile them in groups of four, the Max-pooling makes sure that resulted feature map holds output from the kernel with the strongest signal. The Max - Pooling layer will be used to define the process of training of CNNs.

**Fully-Connected layer**

Fully-Connected layer is equal to layer from Fully Connected Neural Network (FCNN) that was already described. Its training also follows already describes the procedure.

**6.3.2 Training of CNN**

An optimization procedure of CNN is similarities to FCNN. Situation with CNN is more complex because network is made of different types of layers. Forward signal propagation and backward error propagation are following special protocol for each layer. Calculations used in this section were inspired from [73]. The first stage is called forward-propagation, where the signal is broadcast from the inputs of the CNNs to its output. In the last layer the output is associated with the desired value by cost function and error is estimated. In second stage is again using the backpropagation algorithm to estimate the error contribution of individual units. Inconstant parameters of the network are again optimization by gradient descent algorithm.

**Forward Propagation of Convolution Layer**

Each convolutional layer is preforming convolution process on its input. Presuming that the input of a layer is of length N × N units and the kernel is of length m × m. Convolution is calculated over (N−m+1)×(N−m+1) units without zero padding.

Calculation of convolution output xlij is defined as0

where is index of current layer, ωab are weights of the kernel and is output of previous layer. Output of convolutional layer is computed by squashing of output of convolution operation through non-linearity:

where g represents this non-linear function.

**Backward Propagation of Convolution Layer**

Backward propagation for convolutional layer is following the similar principles as described in Section 6.2.4. The difference is in fact that convolution kernel shares weights for entire layer and kernels do not have bias described in Section 6.2.1. Given partial derivative of error from previous layer with respect to output of convolutional layer , influence of kernel weights on the cost function it needs to calculate

From Equation 6.20 it follows that ,Thus

To calculate deltas (equivalent to Equation 6.9) using the chain rule

*=*

Since is already given the deltas was calculated by the derivation of activation function. Last phase comes to propagation of error into previous layers by equation

Again, from Equation 6.29 it follows that =ωab, therefore

The result looks suspiciously like convolution operation and can be taken as the convolution of error with flip kernel.

**Forward Propagation of Pooling layer**

Feed forward operation of pooling layer is straight forward as defined in section 6.3.1. The ratio is typically four to one, which means that the input matrix is divided into not overlapping sub-matrices of size 2×2 and each of these produces 1 output. Another possibility is to have overlapping sub-matrices, where length of sub-matrix is larger the number of pixels between application of pooling.

**Backward Propagation of Pooling Layer**

As said in section for forward-propagation, there is no explicit learning process happening in pooling layer. The error is propagated backwards dependent on how the signal was propagated forward. In Max-Pooling layer the error is propagated only to the unit with maximal output in forward-propagation phase. The error is propagated very sparsely, as an outcome.

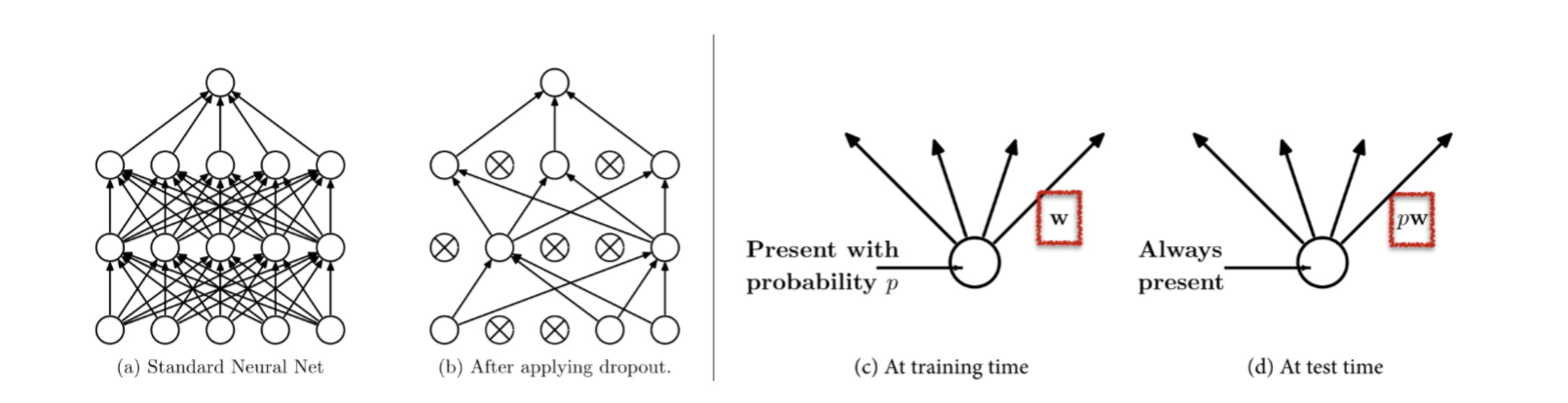
**6.4 Regularization of Neural Networks**

Control of difficulty applies to both NN and CNN. There are several popular regularization techniques that typically consist of adjustment of cost function or optimization algorithm. Somewhat different approach is to modify the structure of the network during training stage.

Dropout By far the greatest regularization technique is to combine the predictions of many different models. This technique greatly improves generalization ability of combined model while preventing over-fitting. Exactly on this idea base ensemble models. The problem with ensemble models is that they require more computational time and very expensive. Because of this, ensembles are usually made of many very simple models [71]. This idea is especially problematic with DNNs, which are model with many parameters that are difficult to train. Moreover, even when trained models are existing in some applications it still isn’t viable to evaluate many different models in a production environment. Another problem is that there might not be enough data to train these different models.

All these problems can be resolved by dropout technique. The basic idea is that each neuron in the network has a certain probability to be deactivated during one iteration. This potential for deactivation is estimated in every iteration, to ensure that network has a different architecture every time. Deactivated means that it will not broadcast any signal through. This forces, individual neurons to pick up features that are less dependent on its surrounding.

Possibility for deactivation is a hyper-parameter that can be tuned, but reasonable default value is 0.5. Dropping out is only happening in the training stage. In testing stage are all weight connection multiplied by the probability of a dropout. This is completed because the activation of the network must stay roughly equivalent in both training ant testing stages. The basic concept is shown in Figure 6.7



*Figure 6.7: Dropout: (a) Standard fully connected network. (b) Network with some neurons deactivated. (c) Activation of neuron during the training phase. (d) Activation of neuron during testing phase [72].*

**Chapter 7**

**Experiment and Result**

This chapter is about description of proposed experiment and results. In the first part is software tools in this study. The next part is description of software and hardware configuration of testing equipment. Next phase preparation of dataset is described. The last portion of this chapter is dedicated to specifics of implementation and results.

**7.1 Software:**

In order to select a suitable tool for implementation of CNN for classification, search of available software tools and libraries was conducted. Now days there are different software tools available for machine learning. Some of these are universal tools for machine learning, but some are exactly designed for deep learning. For the last decade the software tools for machine learning has undergone a renaissance. There is a wide selection of them available and new tools are announced quite frequently. For example, Caffe21 was presented very recently on April 18th. Almost every frequently used programming language has either some software add-on library or at least some available Application Programming Interface (API). The choice of the software tool was influenced by several aspects. Firstly, the implementing language had to be well known and somewhat majority. Enough of available learning documentation materials had to be available, preferably in the form of videos. The most significant factor was good support for learning on the GPU.

**Theano**

It is tested python library. Designed to describe, optimize and evaluate mathematical expressions with multi-dimensional arrays. This makes it suitable for machine learning needs. Theano is made on top of Numpy, which is a python module that enables effective operation with tensors and basic image processing procedure. Mixture of Numpy and Scipy brings a rich set of tools for image processing and data processing. Its abilities can arguably rival MatLab, while being open source and free of cost. Theano’s major rival is currently Tensorflow development. One of the drawbacks of Theano is its low-level nature. Development of machine learning algorithms directly can be very complicated. This is maybe the reason it slowly falls by the wayside. This is also the reason why Theano as a tool is not fit for direct implementation of CNN models.

**Tensor flow**

Tensor flow is similar tool like Theano. As the name suggests this tool is focused on effective work with tensors. It was originally created for internal use in Google for a machine learning project, but it was launched as open source in 2015. Tensor flow calculations are expressed as stateful dataflow graphs, which enables efficient support for GPU supported computation. Tensor flow is currently advertised as one of the fastest frameworks for deep learning needs. Its drawback is like Theano, in the fact that it is very low level and direct usage for operation of Deep learning models is not perfect.

**Caffe**

Caffe is a deep learning tool that goals be modular and faster. It is created by the Berkeley AI Research and by community contributors. Yangqing Jia developed the project during his PhD at UC Berkeley. C++ is programming language is used to implement, but it's also available API for several other languages as for example python. Its main drawback is its lack of quality documentation and material. This fact is partially improved by the existence of Model Zoo, which is a collection of favorite models that are available freely. Caffe was in the last years used by Facebook for example, mainly because its performance capabilities. Caffe is more geared towards the development of large production application than it is for study purposes.

**Keras**

Keras is new software for machine learning, but developed project written in python. It is high lever neural network API. It is built capable of running on top of either Theano or Tensor flow libraries. It is very simple with emphasis on quick model development. It is very simply extensible. At the present time Keras has one of the largest communities among similar tools for deep learning. It has very good documentation and materials which containing many code demonstrations and other resources that help users to get started very rapidly.

**7.2 Hardware and Software Configuration**

Training of Neural Networks notoriously computational expensive and it required a lot of resources. From the bottom level perspective, it translates into many multiplications of matrices. Modern Central Processing Units (CPUs) are not made of such computations and therefore are not very efficient. On the other hand, modern GPUs are designed to preform exactly these operations.

At present on the market, there are two main parallel computing platforms CUDA and OpenCL. They both have their own advantage and disadvantage, but the major difference is that CUDA is proprietary, while OpenCL is available free. This divide translates into hardware productions as well. CUDA is mostly supported by NVIDIA and OpenCL is supported by AMD. NVIDIA with its CUDA platform is presently leader in the domain of deep learning. Therefore, for training of CNN models was selected GPU from NVIDIA. The selected training model was GIGA BYTE GeForce GTX 1080. Details information about hardware configuration is in Table 7.1.

Table 7.1: Hardware Configuration

|  |  |
| --- | --- |
| GPU | GeForce GTX 1080 4GB |
| CPU | Intel(R) Core(TM) i7-8550 CPU @ 2.00GHz |
| Memory | DIMM 1333MHz 8GB |

From the list of considered software tool was selected Keras. The reason being that Keras satisfied all consideration factors and because it was written in python which was most aware to the me. Support of efficient GPU in Keras is relying on either Tensor flow or Theano back-end. From the different user perspective, it doesn’t really matter either way, but Tensor flow was selected because it was observed as faster of the two. GPU-accelerated library package of primitives for deep neural networks. Details information about software configuration is brief in table 7.2

Table 7.2 Software Configuration

|  |  |
| --- | --- |
| Keras | 2.04 |
| Tensorflow | 1.1.0 |
| CUDA | 7.5 |
| Python | 3.53 |
| Operating System | Window 10 |
| Open CV | 2.0 |

**7.2.1. Architecture**

Our architecture was commonly used in CNN architecture. In this architecture consisting of multiple convolution and dense layer. The CNN architecture included three types of three convolution layer and each layer has their own max pooling layer and one group of fully connected layer followed by a dropout layer and the output layer.

The CNN architecture is developed using the Kera’s deep learning framework with Python and Tensor ﬂow backend. The same network is later used for digit classiﬁcation. The proposed CNN contains 3 convolutional layers and 3 max pooling layers. The only fully-connected layer is the ﬁnal layer and the layer after the last convolutional layer. The input layer takes in batches of images of size 50×50×3. ReLU is used between the hidden layers and a sigmoid activation is used in the output layers of both classiﬁcation and detection. The ﬁrst convolutional layer has 16 ﬁlters of size 3×3. It is followed by a pooling layer that uses the max pooling operation with the size of 2×2. The second convolutional layer has 32 ﬁlters with a size of 5×5. Similarly, to the ﬁrst convolutional layer, it is followed by a max-pooling layer with the kernel size of 2×2. The third convolutional layer has 64 ﬁlters with the same kernel size as previous convolutional layer. The 2×2 max pooling is applied yet again. The parameters of the described layers are also illustrated in Figure 7.1.



*Figure 7.1. CNN network architecture for Alphabets.*

**7.3 Model Structure:**

In this study We applied Keras sequential model for CNN, which is a concept that is suitable for modeling of feed forward network. Definition of the network is made of layers. Concept of layer in Keras sequential model doesn’t fully map into already described definition of the layer from topological viewpoint. Keras layers are finer grained and in order to create an equivalent topological layer it is essential to use multiple Keras layers. A model is created simply by calling Sequential constructor

from keras.models import Sequential

model = Sequential()

Layers are added by calling an add method on an object of sequential model

model.add(NumbersOfLayer),

where NumbersOfLayer is definition of the layer.

**7.3.1 About Keras Layers**

All models were created by configuration of follow layers.

**Convolutional layer**

Convolutional layer architecture was about following structure

*Conv2D(filters=num, kernel\_size=(x, x), strides=(s,s), padding=’valid’, input\_shape=shape)*

Where a name is the number of filters that the layer will have, x is size of the kernel, so is the number of pixels in stride and input\_shape description size of the input matrix.

**Activation Function**

To create activation function on the output of the layer user can require parameter activation of the layer itself or create activation as a layer

*Activation(acitvation\_function)*

where activation\_function cab be ’softmax’ or ’relu’. Both specifications are equal because Keras automatically applied linear activation function for each layer.

**Pooling Layer**

Pooling layer can be defined as

*MaxPooling2D (Pool\_Size=(z, z), strides=(s, s))*

Where Pool\_Size requires the size of pooling kernel and strides requires a number of pixels in vertical and horizontal direction that are traversed in between application of individual pools.

**Fully Connected Layer**

Fully connected layer is created using below function

*Dense(total\_units)*

Where total\_units is total number which is a fully connected neurons in a specific layer.

**Dropout**

Similarly, to the activation function to apply a drop out regularization on a layer it wants to be added after it as another layer.

*Dropout(prob)*

Where *prob* is both probability that any unit is dropped and also the coefficient by which are the outputs multiplied through forward evaluation.

**Other**

Feature extraction layers are n dimensional. Specifically, Convolutional and Pooling layers are 2D (two dimensional). Classification layers that are created by fully connected layers are 1D (one dimensional). To join the two, it is required to create mappings between them. For this purpose, it needed to use following layer

*Flatten()*

which takes care of necessary connections between these layers.

**7.4 Result**

On our self-generated dataset, we achieved 99.00% accuracy on the alphabet gestures and 100% accuracy on digits. We did real time testing with different five students and estimate per user took 20 minutes time for alphabets and approximate 7 to 8 minutes for digits. We have tested with non-controlled form. Uncontrolled form means different lighting condition and different background. Confusion matrix and result accuracy graph shown in below.



*Figure 7.2. Epochs vs. validation accuracy for digits.*



*Figure 7.3. Epochs vs. validation accuracy for alphabets.*



*Figure 7.4. Confusion matrix for 0 to 9 digits.*



*Figure 7.4. Confusion matrix for A to Z alphabets.*

**Chapter 8**

**Conclusion and Future work**

**8.1 Conclusion**

One of the main purposes of the interactive virtual environments is to provide natural, efficient, and flexible communication between the user and the computer. Human gestures, including the positions and movements of the fingers, hands, and arms represent one of the richest non-verbal communication modalities allow human users to interact naturally with the virtual environment. Sign gesture can be static, where the human takes on a specific pose.

Real-time vision-based sign recognition is one of the most challenging research areas in the human computer interaction area. Vision-based sign gesture recognition can rely on generic video cameras already available on a large variety of computers, tablets, smart phones, etc. to recognize hand gestures.

This thesis proposed static sign gesture recognition system, which works under different lighting conditions with changed transformations and cluttered background. The Sign gesture recognition systems will run a more efficient and natural interaction, a modality for artistic applications. Another important application will be in the sign language recognition for the deaf people.

A First real-time visual hand gesture recognition system contains of two parts: one for hand detection and tracking using face subtraction, skin detection and contour comparison algorithms proposed in Chapter 5, and the second one which performs gesture recognition using Conventional Neural Network (CNN).

To conclude, in this thesis a real-time system was proposed that consists of three modules: hand detection and skin detection and contour comparison algorithm, gesture recognition using deep learning CNN network. Result show that the system can reach satisfactory real-time performance regardless of the frame resolution size as well as high classification accuracy of 99.00% under variable scale, orientation and illumination conditions, and cluttered background. Three important reasons affect the accuracy of the system, which are the good quality of the webcam while capturing images for a dataset, the number of the training images, and choosing Conventional Neural Network model.

**8.2 Future Work**

Sign gesture recognition still has a long way to go on the research path, especially for 2D systems. This study offers fascinating ideas for future research. Some of these possibilities are defined in this section.

**Dynamic Sign Gesture Recognition**

As this thesis focused only on static sign gesture recognition, one next step forward is to recognize the dynamic sign gesture for the ASL.

**Sign Gesture Recognition from video:**

Now a day’s videos is generally found on the internet. The idea of categorizing single frames is a start to classifying frames in videos. This can be applied in real time classifications. Extending the algorithms proposed in this thesis to video and building an automatic transcript system is an important step onward. For this purpose, it might be fascinating to explore sequential models that study the time dimension, such as recurrent neural networks and or a neural architecture combining CNNs and RNNs.

**Apply to 3 Dimension** **technique**

Now days 3Dimension cameras and sensors are very easy to available in the market and less expensive. This different type of sensor can provide much more information about the hand, making it possible to create more accurate systems for sign language real time recognition.

**Add more gesture in the dataset:**

Even though that study introduces a self-generated new dataset with a rather more gesture for American Sign Language, it still does not offer all the possible movements for American Sign Language. Videos with rotation in 3Dimension, words and expressions are examples of how this dataset can be extended.