Human activity and emotion recognition from RGB videos using deep learning

Advisor: PROF. ANDREA BONARINI

Master Graduation Thesis by:

ARNAB DEY
Student Id n. 893291

Academic Year 2019-2020
Riconoscimento di attività ed emozioni dai video RGB tramite l’apprendimento profondo

Relatore: PROF. ANDREA BONARINI

Tesi di Laurea Magistrale di:
ARNAB DEY
Matricola n. 893291

Anno Accademico 2019-2020
I am dedicating this thesis to my parents for their kindness, devotion, and endless support during my study. Their selflessness will always be remembered.
I wish to express my sincere appreciation to my supervisor, Professor Andrea Bonarini, he convincingly guided through the hard challenges I faced during this thesis. Without his persistent help, the goal of this project would not have been realized. The technical contribution of ‘AiRLab’ during the thesis is truly appreciated. Without their support and resources, this project could not have reached its goal.
# CONTENTS

Abstract                                                                                               xv  
1 INTRODUCTION                                                                                          1  
2 STATE OF THE ART                                                                                     5  
  2.1 Activity Recognition ................................................. 5  
    2.1.1 Human Action Datasets ........................................... 7  
  2.2 Emotion Detection .......................................................... 8  
    2.2.1 Conventional FER Approaches ...................................... 9  
    2.2.2 Deep Learning-Based FER Approaches .............................. 10  
    2.2.3 FER Datasets .......................................................... 11  
3 PROBLEM FORMULATION                                                                                   13  
  3.1 Activity recognition ................................................... 14  
    3.1.1 Skeleton detection .................................................. 14  
    3.1.2 Skeleton data encoding ............................................ 14  
    3.1.3 Classification ...................................................... 14  
  3.2 Emotion detection .......................................................... 15  
    3.2.1 Face detection ....................................................... 15  
    3.2.2 Pre-processing ....................................................... 15  
    3.2.3 Classification ....................................................... 15  
4 PROPOSED SOLUTION FOR ACTIVITY RECOGNITION                                                            17  
  4.1 Selection of activity .................................................... 17  
  4.2 Dataset .......................................................................... 17  
  4.3 Skeleton detection .......................................................... 21  
    4.3.1 PoseNet .................................................................... 22  
  4.4 Sequence to image ........................................................... 26  
    4.4.1 Skeleton to RGB image transformation .............................. 27  
  4.5 Image classification ....................................................... 29  
    4.5.1 Model architecture .................................................... 32  
    4.5.2 Transfer learning ..................................................... 37  
  4.6 Prediction ....................................................................... 39  
5 PROPOSED SOLUTION FOR EMOTION RECOGNITION                                                            41  
  5.1 Selection of Emotion ....................................................... 41  
  5.2 Dataset creation and pre processing ................................. 42  
  5.3 Image Classification ....................................................... 46  
    5.3.1 Transfer learning ..................................................... 46  
    5.3.2 Model architecture .................................................... 47  
6 SYSTEM ARCHITECTURE                                                                                   49  
  6.1 Activity Recognition dataset creation ........................................................................ 49  
  6.2 Activity recognition training ................................................................................ 51
6.3 Emotion recognition dataset creation and pre-processing . . . . 51
6.4 Emotion recognition model training . . . . . . . . . . . . . . 52
6.5 System architecture . . . . . . . . . . . . . . . . . . . . . . . 52

7 EXPERIMENTAL RESULTS OF ACTIVITY RECOGNITION 55
7.1 Data Representation . . . . . . . . . . . . . . . . . . . . . . . 55
7.2 Comparison between stretching and repeating encoding tech-
iques . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
7.3 Comparison between 17 keypoints and 14 keypoints for encoding
7.4 Comparison between cross-subject and cross-view . . . . . . . 59
7.5 Comparison between 2 encoding techniques using score and
average of keypoints for 3rd channel for image . . . . . . . . . . . . 60
7.6 Model Comparison . . . . . . . . . . . . . . . . . . . . . . . . . 61
7.7 Compare 3 types of training . . . . . . . . . . . . . . . . . . . . 62
7.8 Compare 6 different models performance . . . . . . . . . . . . 63
7.9 Model Training . . . . . . . . . . . . . . . . . . . . . . . . . . . 64

7.9.1 ResNet . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
         7.9.1.1 Confusion matrix . . . . . . . . . . . . . . . . . . . . . . 64
         7.9.1.2 Precision matrix . . . . . . . . . . . . . . . . . . . . . . . 66
         7.9.1.3 Recall Matrix . . . . . . . . . . . . . . . . . . . . . . . . . 68
7.9.2 Densenet . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68
         7.9.2.1 Confusion matrix . . . . . . . . . . . . . . . . . . . . . . . 70
         7.9.2.2 Precision matrix . . . . . . . . . . . . . . . . . . . . . . . 70
         7.9.2.3 Recall matrix . . . . . . . . . . . . . . . . . . . . . . . . . 70

7.10 Real time performance . . . . . . . . . . . . . . . . . . . . . . 72

8 EXPERIMENTAL RESULTS OF EMOTION RECOGNITION 73
8.1 Dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73
8.2 Training . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73
8.3 Test Result . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
         8.3.1 Confusion Matrix . . . . . . . . . . . . . . . . . . . . . . . . 75
         8.3.2 Precision Matrix- . . . . . . . . . . . . . . . . . . . . . . . . 77
         8.3.3 Recall matrix . . . . . . . . . . . . . . . . . . . . . . . . . . . 78

9 DISCUSSION 81

10 CONCLUSIONS AND FUTURE WORKS 83

BIBLIOGRAPHY 85
| Figure 1.1 | Activity Recognition process: It takes video as input and return activity as output. | 2 |
| Figure 1.2 | Emotion Recognition process: it takes image of a face as input and return emotion as output | 2 |
| Figure 2.1 | Categories and sub-categories of activity recognition approaches based on sensor data | 6 |
| Figure 2.2 | A typical CNN-based FER system. The image also shows different types of layers present in a CNN. | 9 |
| Figure 2.3 | Basic structure of an LSTM. Each LSTM module consists of 4 activation function as shown in the picture with yellow boxes. Picture from: Colah’s blog [76] | 10 |
| Figure 3.1 | Robot Assistant in a shopping mall, example of an activity and emotion detection application in everyday life. | 13 |
| Figure 4.1 | Samples from NTURGB+D dataset. Picture taken from: NTURGB+D website[75] | 18 |
| Figure 4.2 | Pose confidence score vs keypoint confidence score. Picture taken from: PoseNet website[74] | 23 |
| Figure 4.3 | Seventeen pose keypoints detected by PoseNet. Picture taken from: PoseNet website[74] | 23 |
| Figure 4.4 | Single person pose detector pipeline using PoseNet. Picture taken from: PoseNet website [74] | 24 |
| Figure 4.5 | Relationship with output strip and performance and accuracy. Picture taken from: PoseNet website [74] | 25 |
| Figure 4.6 | Heatmap and offset vector. Picture taken from: PoseNet website[74] | 25 |
| Figure 4.7 | Mapping between XYS and RGB space. left picture shows axis corresponding to X position, Y position, and score of the keypoints. | 26 |
| Figure 4.8 | The picture shows 3 steps to generate RGB image from skeleton data. | 26 |
| Figure 4.9 | Transformation of motion capture sequence into an RGB image and intermediate steps. | 29 |
| Figure 4.10 | Images generated using two different encoding techniques. | 30 |
| Figure 4.11 | Basic structure of CNN used for image classification task. | 31 |
LIST OF TABLES

Table 2.1  Some of the most popular RGB(2D) human action
datasets available. ................................. 7
Table 2.2  Some of the most popular RGB-D(3D) human action
datasets available. ................................. 8
Table 4.1  NTURGB+D Action Recognition Datasets ..................... 17
Table 4.2  Activities related to medical conditions ......................... 19
Table 4.3  Activities related to Mutual Actions / Two Person
Interactions .................................................. 19
Table 4.4  Activities related to Daily Actions ............................. 20
Table 5.1  Four datasets we used to collection human emotion data. 43
Table 7.1  Comparison between 2 encoding methods based on
validation accuracy obtained by the models. The table
shows effect of stretching and repeating methods on
validation accuracy ................................. 58
Table 7.2  Comparison between 2 encoding methods(17 and 14
keypoints) based on validation accuracy obtain by the
models ..................................................... 59
Table 7.3  Comparison between cross-view and cross-subject eval-
uation methods. ............................................ 60
Table 7.4  Comparison of model performance when we used 3rd
channel as score of keypoints and average of X and Y
position of Keypoints. ................................. 60
Table 7.5  Comparison of validation accuracy achieved by differ-
et models while using same configuration and data. . 61
Table 7.6 Comparison among deep retraining, shallow retraining, and training from scratch techniques. Table shows validation accuracy achieved by 6 models with 3 techniques. .......................................................... 62

Table 7.7 Table shows training time, prediction time, and model size of 6 different models in case of 17, and 60 classes activity classification .................................................. 63
In this thesis, we have proposed a new methodology to recognize human activities and emotions based on RGB videos, which take advantage of the recent breakthrough made in the field of deep learning. This method uses an image classification approach to recognize human activities and emotions. We can divide this problem into two sub-problems: activity recognition, and emotion recognition. We have used the transfer learning technique for both problems. Human activity and emotion recognition gained popularity in recent years because of the wide use of digital cameras in daily life and their potential for human-computer interaction, and robotics applications.

In this work, a solution is proposed requiring only the use of RGB video instead of RGB-D videos to recognize human activity and emotion. This work shows a different approach based on the conversion of RGB video data into 2D images and image classification. From a stream of RGB videos, a two-dimensional skeleton of 17 joints for each detected body part is extracted with a DNN-based human pose estimator called PoseNet. Then, skeleton data are encoded into red, green, and blue channels of an image. A different way of encoding data was studied and compared.

We used different state-of-the-art deep neural network architectures to classify human activities and compared them. Based on related works, we have chosen to use image classification models: SqueezeNet, AlexNet, DenseNet, ResNet, VGG, Inception, and retrained them to perform action recognition. For all the experiments for activity recognition, the NTU RGB+D database was used. The highest accuracy was obtained with ResNet 88.19%, which outperformed all the previous works.

The second part of the problem is the detection of facial expressions from RGB videos. Based on the previous study, we have used image classification techniques based on deep learning. Before doing classification, we have applied the OpenCV face detection function to recognize faces in the wild. Cropped image of the face used as an input to our retrained VGG16 emotion detection model based on deep neural network. The highest accuracy is obtained with VGG16 (85.06%) which is comparable to any other state of the art approaches.
In questa tesi, abbiamo proposto una nuova metodologia per riconoscere le attività e le emozioni umane basate sui video RGB, che sfruttano i recenti risultati nel campo dell’apprendimento profondo. Questo metodo utilizza un approccio di classificazione delle immagini per riconoscere le attività e le emozioni umane. Possiamo dividere questo problema in due sottoproblemi: riconoscimento dell’attività e riconoscimento delle emozioni. Abbiamo usato la tecnica di apprendimento per trasferimento (transfer learning) per entrambi i problemi. Il riconoscimento dell’attività umana e delle emozioni hanno guadagnato popolarità negli ultimi anni a causa dell’ampio uso delle fotocamere e del loro potenziale utilizzo per l’interazione uomo-computer e per applicazioni di robotica. In questo lavoro, viene proposta una soluzione che richiede solo l’uso di video RGB anziché video RGB-D per riconoscere l’attività umana e le emozioni. Questo lavoro si basa su diversi approcci basati sulla conversione di dati video RGB in immagini 2D e classificazione delle immagini. Da un flusso video RGB, uno scheletro bidimensionale di 17 articolazioni per ogni parte del corpo viene estratto con uno stimatore di posa umana basato su DNN chiamato PoseNet. Quindi, i dati dello scheletro vengono codificati nei canali rosso, verde e blu di un’immagine. Sono stati studiati e confrontati diversi modi di codificare i dati.

Abbiamo usato diverse architetture di reti neurali profonde disponibili allo stato dell’arte per classificare le attività umane e confrontarle. Sulla base di questo studio dei lavori correlati, abbiamo scelto di considerare diversi modelli di classificazione delle immagini: SqueezeNet, AlexNet, DenseNet, ResNet, VGG, Inception e li abbiamo ri-allenati per eseguire il riconoscimento dell’azione. Per tutti i test per il riconoscimento delle attività, è stato utilizzato il database NTU RGB+D. La massima precisione si ottiene con ResNet 88.19%, che supera le prestazioni di tutti i lavori precedenti.

La seconda parte del problema è il rilevamento delle espressioni facciali dai video RGB. Sulla base dello studio precedente, abbiamo utilizzato tecniche di classificazione delle immagini basate sull’apprendimento profondo in questo lavoro, abbiamo applicato la funzione di rilevamento dei volti di OpenCV per riconoscere i volti in ambiente non strutturato e li abbiamo usati come input per il nostro modello di rilevamento delle emozioni VGG16 basato su una rete neurale profonda. La massima precisione si ottiene con VGG16 (85,06%), che è paragonabile ad altri approcci allo stato dell’arte.
INTRODUCTION

Many Researchers aimed at implementing machines to understand human activity and human emotions. It has been very popular area of research because of the lucrative application in different fields. Despite all the previous research, still this is a very challenging and active area for research. In our work we have tried to solve some of the related problems.

Activity recognition and emotion detection are important research areas in computer vision that have a wide range of applications like: video surveillance, robotics, and human computer interaction. In recent years, development of real-time skeleton detection algorithms, deep neural networks, and powerful hardware allow us to include activity recognition and emotion detection in various applications. The task is to recognize human activity and emotion frame by frame in real-time video stream. Activity recognition can be a major challenge because of diversity of movements, the complexity of motion capture, and availability of large datasets. Same activity can be done in different ways by different persons, in different contexts so it is always challenging to create good datasets of human activity. In case of facial emotion detection, the same emotion can be different for different people and based on different situations.

Very recently, human pose estimation from 2D RGB videos have also been studied with deep CNN methods [1–4] and pre-trained models like PoseNet¹, and OpenPose² gained popularity. However, it seems there is still lack of effective method to deal with this kind of 2D skeleton data to recognize human activity. Recently deep learning methods have achieved great success in high-level computer vision tasks such as image recognition, classification, detection, and segmentation, etc. As for the skeleton-based action recognition problems, Recurrent Neural Networks (RNNs) widely adopted [5–7]. RNNs could effectively extract the temporal information and learn the contextual information well. However, RNNs tend to overemphasize the temporal information especially when training data is insufficient, thus leading to overfitting problems [7].

We present a new method to process RGB videos to perform activity recognition (Fig:1.1). The first step to recognize activity is to extract motion from the videos. So we will extract the skeleton of the people in the videos. Skeleton extraction can be done by using PoseNet, a machine learning model that allows extracting a 2D skeleton of 17 joints for each detected human

¹ PoseNet–https://www.tensorflow.org/lite/models/pose_estimation/overview
² OpenPose–https://github.com/CMU-Perceptual-Computing-Lab/openpose
**Figure 1.1:** Activity Recognition process: It takes video as input and return activity as output.

body. Then motion sequence can be converted to RGB images as explained in [8]. Finally, we used those images to classify activity using a deep neural network. We exploit the advantages of deep neural network by transforming the skeleton based activity recognition task into an image classification task. The motion parameters are encoded in R, G, and B channels to generate an image.

The second part of the problem is to recognize emotion from RGB images as shown in the process(Fig:1.2). For this part of the problem, we have gathered four datasets with images of faces categorized into 7 different emotions, both human and animated images. We have used a transfer learning approach with VGG16 architecture to train a model that recognizes emotion from RGB images of the face.

*Figure 1.2:* Emotion Recognition process: it takes image of a face as input and return emotion as output.

We have divided our thesis into 8 chapters. The first chapter is "Introduction" 1, it will give an overview of the problem. The second chapter is "State of the art" 2, where we will discuss the previous research done on these
topics. The next chapter is "problem formulation" 3, where we will discuss the activity and emotion detection problems in detail. Forth chapter, "Proposed Solution for Activity Recognition" 4 will describe the solution approach we took to solve activity recognition problems. In chapter "Proposed Solution for Emotion Recognition" 5, we will discuss about solutions about emotion recognition. In next chapter "system architecture" 6, we will discuss technical details about our solution. The seventh chapter is "Experimental Results of Activity Recognition" 7, which will compares the results and experiment we have done for activity recognition. In chapter eight "Experimental Results of Emotion detection" 8 we discussed results and experiments of emotion detection. In chapter nine "Discussion" 9 we will analyze the results we found from the experiments. Chapter ten 10 describe future work and conclusion of the thesis.
STATE OF THE ART

Human activity and emotion recognition have been very widely researched topics because of their potential use cases. The researchers have been trying to solve problem of activity and emotion recognition with different approaches since long time. In this chapter we will present some of the research results achieved so far.

2.1 ACTIVITY RECOGNITION

*Activity recognition aims to recognize the actions and goals of one or more agents from a series of observations on the agents’ actions and the environmental conditions.*

— Wikipedia

Here, when we say “activity recognition” we mean “human activity recognition” based on image sensors. Activity recognition, in general, may concern different aspects like: plan recognition, goal recognition, intent recognition, and behavior recognition.

Activity recognition problem can be solved with different approaches like: logic and reasoning, probabilistic reasoning, Data mining based approach, and sensor-based approaches. Vision-based activity recognition is one kind of sensor-based activity recognition.

Figure 2.1 shows different types of approaches to solving the activity recognition problem based on visual data. Human activity recognition can be divided into two categories: Single and hierarchical. Hierarchical approaches can be statistics, syntactic, or Description based. On the other hand, a single-layered approach can be further divided into Spacetime or sequential. We are going to discuss these approaches in detail later in this chapter. With the development of deep learning algorithms in the last couple of years, image and video classification started to have great success. Some of the researchers tried to take advantage of it by transforming skeleton-based activity recognition tasks to image classification [10]. Many works have exploited the RGB-D (Red, Green, Blue, and Depth) data for activity recognition. Review [9] by Aggarwal and Xia summarized different techniques for activity recognition using 3D data. The motion parameters can be encoded into Red, Green and Blue channels and action sequences become an RGB Image [8]. Then we can use neural networks for image classification to recognize activity.

1 https://en.wikipedia.org/wiki/Activity_recognition
Figure 2.1: Categories and sub-categories of activity recognition approaches based on sensor data

Laraba et al. [8] encoded X, Y and Z position of each joint into R, G, B channels. They obtain an accuracy of 74.24% for the cross-subject and 75.74% for the cross-view action recognition using NTU RGB+D dataset [11] with CNN. With the same technique, Du et al. [12] obtain an accuracy of 100% with Berkeley MHAD dataset [13].

Laraba et al. [10] encoded X, Y, and C values of each joint as R, G, B channel where X and Y are the 2D coordinates of the joint and C is the confidence score of each joint. They obtain an accuracy of 83.317% for cross-subject and 88.780% for cross-view with retraining ResNet152 on NTU RGB+D dataset.

In [14], Ding et al. encoded different features into RGB images. They tested 5 different features: joint-joint vectors, joint-joint distance, joint-line distance, joint-joint orientations, and line-line angles. They generated 5 different RGB images with 5 different methods and trained 5 different CNNs. The output score is a combination of all 5 methods. They obtained an accuracy of 82.31% with NTU RGB+D dataset [11].

In [15], Ke et al. suggested another representation of skeleton data. They encoded 3D skeleton data into a gray-scale image containing spatio-temporal information. They obtain accuracy of 84.83% for cross-view and 79.57% for cross-subject with NTU RGB+D dataset [11].

In [16], Li et al. propose to using skeleton data by dividing them into five body part, 3D coordinates of joints of each body part is concatenated in a
vector and encoded into an RGB image. They obtained 84.6% for cross-subject and 90.9% for cross-view with NTU RGB+D dataset [11].

In [17], Li et al proposed a method to rearrange and select important skeleton joints automatically. The order of the joints influences accuracy. They obtain the accuracy of 83.2% for cross-subject and 89.3% for cross-view using NTU RGB+D dataset [11].

2.1.1 Human Action Datasets

There are quite a lot of datasets available for the human activity recognition. In [43], Singh et al. provided a comparison between most popular human action datasets. These datasets can be divided in two categories: RGB(Table:2.1) and RGB-D (Table:2.2) datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Year</th>
<th>Modality</th>
<th>Application domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>KTH (Laptev and Lindeberg)</td>
<td>2004</td>
<td>Gray</td>
<td>Human action recognition in real outdoor conditions</td>
</tr>
<tr>
<td>Weizmann (Gorelick et al.)</td>
<td>2005</td>
<td>RGB</td>
<td>Human action recognition</td>
</tr>
<tr>
<td>Hollywood 2 (Marszaek et al.)</td>
<td>2009</td>
<td>RGB</td>
<td>Realistic actions recognition from movies</td>
</tr>
<tr>
<td>YouTube 8M (Haija et al.)</td>
<td>2016</td>
<td>RGB</td>
<td>Human activity recognition, human interaction</td>
</tr>
<tr>
<td>Something–Something (Goyal et al.)</td>
<td>2018</td>
<td>RGB</td>
<td>Human–object interaction</td>
</tr>
<tr>
<td>NTU RGB+D (Shahroudy et al.)</td>
<td>2016</td>
<td>RGB</td>
<td>Human Action Recognition</td>
</tr>
</tbody>
</table>

Table 2.1: Some of the most popular RGB(2D) human action datasets available.
### Table 2.2: Some of the most popular RGB-D (3D) human action datasets available.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Year</th>
<th>Modality</th>
<th>Application domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>i3DPost Multi-View (Gkalelis et al.)</td>
<td>2009</td>
<td>RGB, Depth</td>
<td>Human action/interaction and behaviour</td>
</tr>
<tr>
<td>Berkeley MHAD (Ofli et al.)</td>
<td>2013</td>
<td>RGB, Depth, skeleton</td>
<td>Human behaviour Recognition</td>
</tr>
<tr>
<td>Hollywood 3D</td>
<td>2013</td>
<td>RGB, Depth</td>
<td>Natural action recognition in movies</td>
</tr>
<tr>
<td>UTD-MHAD (Chen et al.)</td>
<td>2015</td>
<td>RGB, Depth, skeleton</td>
<td>View-invariant human action recognition</td>
</tr>
<tr>
<td>NTU RGB+D (Shahroudy et al.)</td>
<td>2016</td>
<td>RGB, Depth, skeleton, IR sequences</td>
<td>Human Action Recognition</td>
</tr>
<tr>
<td>PRAXIS GESTURE (Negin et al.)</td>
<td>2018</td>
<td>RGB, Depth</td>
<td>Elderly care assistive activities</td>
</tr>
<tr>
<td>PKU-MMD (Liu et al.)</td>
<td>2017</td>
<td>RGB (image and video), Depth, skeleton, IR sequences</td>
<td>Multi-modal action recognition</td>
</tr>
</tbody>
</table>

2.2 Emotion Detection

In [18], Ko has given a brief review of different techniques used in the field of facial expression recognition (FER) over past decades. Although various sensors such as electromyograph (EMG), electrocardiogram (ECG), electroencephalograph (EEG), and a camera can be used as an input for FER input, Camera is the most promising because it provides more clues and it is easier to use. A conventional FER system is made of 3 main parts:

- face and facial component detection
- feature extraction
- expression classification
In contrast to traditional approaches using handcrafted features, deep learning has emerged as a general approach to machine learning, yielding state-of-the-art results. A deep learning-based approach reduces complexity of conventional 3 step based approach by enabling end-to-end learning [19]. Among all the deep learning models, Convolutional neural network (CNN) is the most popular for this type of problem. Figure 2.2 shows the procedure used by CNN-based FER approaches.

**Figure 2.2:** A typical CNN-based FER system. The image also shows different types of layers present in a CNN.

### 2.2.1 Conventional FER Approaches

For conventional FER systems, the main focus is detecting the face region and extracting geometric features, appearance features, or a hybrid of geometric and appearance features on the target face. For geometric features, the relationship between facial components is used to construct a feature vector for training [20, 21]. In [21], they used two types of geometric features based on the position and angle of 52 facial landmark points.

As an example of using global features, Happy et al. [22] utilized a local binary pattern (LBP) histogram of different block sizes from a global face region as the feature vectors, and classified various facial expressions using a principal component analysis (PCA).

For hybrid features, some approaches [22, 23] have combined geometric and appearance features to complement the weaknesses of the two approaches and provide better results.

Apart from FER of 2D images, 3D and 4D recordings are increasingly used in expression analysis research. Some of the researchers [24–28] also tried to recognize facial expressions or emotions using infrared images instead of visual light spectrum images (VIS).
2.2.2 Deep Learning-Based FER Approaches

In recent years, there has been a breakthrough in deep learning algorithms including CNN and recurrent neural networks (RNN). These deep learning algorithms have been used in many fields including facial expression recognition. The main advantages of the deep learning-based approaches are complete removal of physics-based approach or pre-processing by enabling end-to-end learning. CNN has achieved the state of the art results for facial expression recognition.

In [29], Breuer and Kimmel used the CNN visualization technique to understand a model learned using various FER datasets. Jung et al. [30] used two different types of CNN to extract temporal appearance features from an image sequence and extract temporal geometric features from facial landmark points.

Figure 2.3: Basic structure of an LSTM. Each LSTM module consist of 4 activation function as show in the picture with yellow boxes. Picture from: Colah’s blog [76]

In [31], Zhao et al. proposed deep region and multi-label learning (DRML). A hybrid approach combining a CNN with long short-term memory (LSTM) was developed. LSTM is a type of recurrent neural network (RNN) capable of keeping track of arbitrary long-term dependencies in input sequences. LSTM has a chain-like structure as shown in figure 2.3. Kahou et al. [32] proposed a hybrid RNN-CNN framework for propagating information over a sequence using a continuously valued hidden layer representation. The hybrid CNN-RNN architecture facial expression analysis outperformed previous CNN in the 2015 Emotion Recognition in the Wild (EmotiW) Challenge [33].

Kim et al. [34] utilized representative expression-state (e.g. the onset, apex, and offset of the expression), which can be specified in facial sequences regardless of the expression intensity.

In [35], Chu et al. proposed a multi-level facial AU detection combining spatial and temporal features. In [36], Hasani et al. proposed the 3D Inception-ResNet architecture followed by an LSTM. Graves et al. [37] used an RNN
to consider the temporal dependencies present in the image sequences. This study proved that a bidirectional network provides significantly better performance than a unidirectional LSTM.

2.2.3 FER Datasets

In the field of facial expression recognition, various datasets have been collected, the most popular being reported in Table: 2.2.3:

<table>
<thead>
<tr>
<th>Database</th>
<th>Data Configuration</th>
<th>Web Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>CK+ [38]</td>
<td>• 593 video sequences.</td>
<td><a href="http://www.consortium.ri.cmu.edu/ckagree/">http://www.consortium.ri.cmu.edu/ckagree/</a></td>
</tr>
<tr>
<td></td>
<td>• 123 subjects from 18 to 30 years old.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Image resolutions of 640X480, and 640X490.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 230 human subjects (130 females and 100 males, mean age 23).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Image resolution of 3000X4000.</td>
<td></td>
</tr>
<tr>
<td>FERG-DB [40]</td>
<td>• 55767 images corresponding to 7 categories.</td>
<td><a href="https://grail.cs.washington.edu/projects/deepexpr/ferg-db.html">https://grail.cs.washington.edu/projects/deepexpr/ferg-db.html</a></td>
</tr>
<tr>
<td></td>
<td>• six stylized animated characters.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• High resolution images.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 39 human subjects from 19 to 54 years old.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• High resolution images.</td>
<td></td>
</tr>
</tbody>
</table>
• 213 images of seven facial emotions.

JAFFE [42]  • Six emotion adjectives by 60 Japanese subjects.

• Image resolution of 256X256.

http://www.kasrl.org/jaffe_info.html
In this chapter, we are going to discuss in details motivation and goals for this thesis. Let’s start from the beginning: why did we decide to work on this problem? The idea of this thesis began with idea of an assistance robot in a shopping mall. We wanted to design a robot which could assist and interact with shoppers or, general public that can be found in a large space like a shopping mall. To do this, a robot needs to understand humans and environment and interact with them. To make the interaction more natural and contextual, the robot needs to understand physical and emotional state of the person.

What is the best way to understand the state of the person other than analyzing current activity and emotion? To make natural interaction, understanding emotions of the person is very important during the conversation to respond accordingly. We need to understand the current activity of the person so that the robot can approach the person appropriately. So we decided to make a system to recognize human activity and emotions.

Other than this particular use case there can be a thousands of situations where a system like this can be very useful, like surveillance, monitoring, abnormal activity detection, etc. This type of system can be also useful for
human-computer interaction. As we have discussed before you can divide the problem into two sub-problems.

3.1 ACTIVITY RECOGNITION

In this section, we are going to discuss problems related to activity recognition. Considering the potential use case for our application, some activities are more important than others. For our application we have selected the following activities: sit down, stand up, cheer up, hand waving, phone call, play with the phone, point to something, check the time, run on the spot, push, follow, and support somebody. Our objective is to detect these activities real-time in the wild. We focused on recognition of the specific activities, real-time performance, and low computation power for robotics implementation.

Activity detection can be divided into 3 parts: skeleton detection, skeleton data encoding, classification.

3.1.1 Skeleton detection

The first step of the process is skeleton detection from RGB videos. Here skeleton means a real-time multi-person system to jointly detect human body, hand, facial, and foot keypoints on single image. Skeleton is generally defined by key points of the body; the number of key points can vary among 17, 18, or 25 depending on the specific algorithm. Skeletons are necessary to track human keypoints to identify activity.

3.1.2 Skeleton data encoding

The next step of the recognition activity is to encode the skeleton data in some format, like RGB images so that we can take advantage of deep learning techniques to classify activities. In the previous chapter, we have seen that different researchers used different approach to encode the skeleton data. Finding the right encoding technique is very important for this problem.

3.1.3 Classification

In this part, we have to classify the activity using the encoded data. We will use a deep learning technique for classification which we need to train with activity dataset. We need to identify appropriate architecture among the all available networks. We need to consider also the needed computation power, the classification time, and performance of the model in real-time. We can not use large models because they will require a lot of time for classification, which will slow down performance significantly.
3.2 EMOTION DETECTION

The second part of our problem is emotion recognition from the RGB video. This is a quite straightforward problem where we need to detect faces and identify the emotion. We can divide the problem into 3 subproblems: Face detection, pre-processing, classification.

3.2.1 Face detection

We need to detect faces in the wild before doing the facial emotion recognition. The face detection algorithm will detect faces in each frame, which will feed to the next step for emotion detection. For our application, we need to have a very efficient face detection algorithm.

3.2.2 Pre-processing

After face detection, we will select the face corresponding to the human skeleton used for activity recognition which will ensure that we are detecting emotion for the same person. We need to resize the image and convert it to a gray scale image before using it as a CNN input.

3.2.3 Classification

In this part, we need to create a model able to classify emotions. For this project, we are going to use a deep learning approach. We need to compare different architectures to find the best one for our application. In this case, performance of the model and prediction time are also need to be considered.
In this chapter, we are going to discuss our proposed solutions in detail. As we have discussed before we are going to divide our problem into two parts: activity recognition and emotion recognition. We are going to start with activity recognition.

Aim of our project is to identify predefined activities in the wild. While we can use different approach to solve this problem, we have decided to use deep learning based image classification method from RGB video. We are going to use skeleton detection algorithm to extract skeletons from videos and encode the skeleton sequences in the RGB image. Finally, we use that RGB image for classification of the activities. So let’s dive deep into the each part of the process in details.

### 4.1 Selection of Activity

We are going to focus on a subset of activities, relevant to our application. The activities we are going to focus on are: Throwing things, sitting down, standing up, clapping, tear up a paper, taking off a shoe, cheer up, hand waving, kicking something, jumping up, phone call, play with phone, taking selfie, etc. Even though we are only interested on these activities we are going to train our model with all classes of popular NTURGB+D [11] dataset and compare it with results obtained in other researches.

### 4.2 Dataset

As we have motioned before, we are going to use NTURGB+D [11] dataset for our project. NTURGB+D Action Recognition Datasets is created by Rapid-Rich Object Search Laboratory of Nanyang Technological University. There are two versions of the dataset: "NTU RGB+D" Dataset and "NTU RGB+D 120" Dataset, whose characteristics are reported in Table 4.1.

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Number of Classes</th>
<th>Number of Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTU RGB+D</td>
<td>60</td>
<td>56,880 video samples</td>
</tr>
<tr>
<td>NTU RGB+D 120</td>
<td>120</td>
<td>114,480 samples</td>
</tr>
</tbody>
</table>

*Table 4.1: NTURGB+D Action Recognition Datasets*

These two datasets both contain:

- RGB videos
- Depth map sequences
- 3D skeletal data
- Infrared (IR) videos

Each dataset is captured by three Kinect V2 cameras concurrently. The resolutions of RGB videos are 1920x1080, depth maps and IR videos are all in 512 × 424, and 3D skeletal data contains the 3D coordinates of 25 body joints.
at each frame. The actions in these two datasets can be divide in three major categories: daily actions, mutual actions, and medical conditions, as shown in the tables below.

<table>
<thead>
<tr>
<th>A41: sneeze/cough</th>
<th>A42: staggering</th>
<th>A43: falling down</th>
</tr>
</thead>
<tbody>
<tr>
<td>A44: headache</td>
<td>A45: chest pain</td>
<td>A46: back pain</td>
</tr>
<tr>
<td>A103: yawn</td>
<td>A104: stretch oneself</td>
<td>A105: blow nose</td>
</tr>
</tbody>
</table>

**Table 4.2**: Activities related to medical conditions

<table>
<thead>
<tr>
<th>A50: punch/slap</th>
<th>A51: kicking</th>
<th>A52: pushing</th>
</tr>
</thead>
<tbody>
<tr>
<td>A53: pat on back</td>
<td>A54: point finger</td>
<td>A55: hugging</td>
</tr>
<tr>
<td>A56: giving object</td>
<td>A57: touch pocket</td>
<td>A58: shaking hands</td>
</tr>
<tr>
<td>A59: walking towards</td>
<td>A60: walking apart</td>
<td>A106: hit with an object</td>
</tr>
<tr>
<td>A107: wield knife</td>
<td>A108: knock over</td>
<td>A109: grab stuff</td>
</tr>
<tr>
<td>A110: shoot with gun</td>
<td>A111: step on foot</td>
<td>A112: high-five</td>
</tr>
<tr>
<td>A113: cheers and drink</td>
<td>A114: carry object</td>
<td>A115: take a photo</td>
</tr>
<tr>
<td>A116: follow</td>
<td>A117: whisper</td>
<td>A118: exchange things</td>
</tr>
<tr>
<td>A119: support somebody</td>
<td>A120: rock-paper-scissors</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4.3**: Activities related to Mutual Actions / Two Person Interactions

---

2 Actions labeled from A1 to A60 are contained in "NTU RGB+D", and actions labeled from A1 to A120 are in "NTU RGB+D 120".
<table>
<thead>
<tr>
<th>Activity 1</th>
<th>Activity 2</th>
<th>Activity 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>drink water</td>
<td>eat meal</td>
<td>brush teeth</td>
</tr>
<tr>
<td>brush hair</td>
<td>drop</td>
<td>pick up</td>
</tr>
<tr>
<td>throw</td>
<td>sit down</td>
<td>stand up</td>
</tr>
<tr>
<td>clapping</td>
<td>reading</td>
<td>writing</td>
</tr>
<tr>
<td>tear up paper</td>
<td>put on jacket</td>
<td>take off a jacket</td>
</tr>
<tr>
<td>put on a shoe</td>
<td>take off a shoe</td>
<td>put on glasses</td>
</tr>
<tr>
<td>take off glasses</td>
<td>put on a hat/cap</td>
<td>take off a hat/cap</td>
</tr>
<tr>
<td>cheer up</td>
<td>hand waving</td>
<td>kicking something</td>
</tr>
<tr>
<td>reach into pocket</td>
<td>hopping</td>
<td>jump up</td>
</tr>
<tr>
<td>phone call</td>
<td>play with phone/tablet</td>
<td>type on a keyboard</td>
</tr>
<tr>
<td>point to something</td>
<td>taking a selfie</td>
<td>check time (from watch)</td>
</tr>
<tr>
<td>rub two hands</td>
<td>nod head/bow</td>
<td>shake head</td>
</tr>
<tr>
<td>wipe face</td>
<td>salute</td>
<td>put palms together</td>
</tr>
<tr>
<td>cross hands in front</td>
<td>put on headphone</td>
<td>take off headphone</td>
</tr>
<tr>
<td>shoot at basket</td>
<td>bounce ball</td>
<td>tennis bat swing</td>
</tr>
<tr>
<td>juggle table tennis ball</td>
<td>hush</td>
<td>flick hair</td>
</tr>
<tr>
<td>thumb up</td>
<td>thumb down</td>
<td>make OK sign</td>
</tr>
<tr>
<td>make victory sign</td>
<td>staple book</td>
<td>counting money</td>
</tr>
<tr>
<td>cutting nails</td>
<td>cutting paper</td>
<td>snap fingers</td>
</tr>
<tr>
<td>open bottle</td>
<td>sniff/smell</td>
<td>squat down</td>
</tr>
<tr>
<td>toss a coin</td>
<td>fold paper</td>
<td>ball up paper</td>
</tr>
<tr>
<td>play magic cube</td>
<td>apply the cream on face</td>
<td>apply the cream on hand</td>
</tr>
<tr>
<td>put on bag</td>
<td>take off bag</td>
<td>put object into bag</td>
</tr>
<tr>
<td>take an object out of bag</td>
<td>open a box</td>
<td>move heavy objects</td>
</tr>
<tr>
<td>shake fist</td>
<td>throw up cap/hat</td>
<td>capitulate</td>
</tr>
<tr>
<td>cross arms</td>
<td>arm circles</td>
<td>arm swings</td>
</tr>
<tr>
<td>run on the spot</td>
<td>butt kicks</td>
<td>cross toe touch</td>
</tr>
<tr>
<td>side kick</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 4.4:** Activities related to Daily Actions
Even though the NTURGB+D dataset has skeleton data, we decided not to use it as we are going to use a separate method(PoseNet) to create a skeleton during the prediction phase. PoseNet is better and more efficient, which is more suited for real-time applications. We will use PoseNet method to create a skeleton of the training data from RGB videos to make the model more consistent.

4.3 SKELETON DETECTION

Pose Estimation is a general problem in Computer Vision where we detect the position and orientation of an object. This usually means detecting key point locations that describe the object.

In this thesis, we will focus on human pose estimation, where it is required to detect and localize the major parts/joints of the body (e.g. shoulders, ankle, knee, wrist, etc.).

Human pose estimation can be of two types:

- 2D Pose Estimation - Estimate a 2D pose \((x,y)\) coordinates for each joint from an RGB image.
- 3D Pose Estimation - Estimate a 3D pose \((x,y,z)\) coordinates a RGB image.

Human pose estimation is a challenging task because of strong articulations, small and barely visible joints, occlusions, clothing, and lighting changes make it a difficult problem.

There are two main approaches for tackling multi-person detection, pose estimation and segmentation. The top-down approach starts by identifying and roughly localizing person instances using a bounding box object detector, followed by single-person pose estimation or binary foreground/background segmentation in the region inside the bounding box. The bottom-up approach starts by localizing identity-free semantic entities (individual keypoint proposals or semantic person segmentation labels, respectively), followed by grouping them into person instances.

Some of the deep learning-based approaches are:

- DeepPose [44]
- Efficient Object Localization Using Convolutional Networks [45]
- Convolutional Pose Machines [28]
- Human Pose Estimation with Iterative Error Feedback [46]
• Stacked Hourglass Networks for Human Pose Estimation [47]
• Simple Baselines for Human Pose Estimation and Tracking [48]
• Deep High-Resolution Representation Learning for Human Pose Estimation [49–51]
• PoseNet [52]
• OpenPose [1–3, 28]

We have decided to use PoseNet in our application because it needs less computation power and much faster than other methods available. We have used pre-trained model of PoseNet3 provided by Google.

4.3.1 PoseNet

PoseNet can be used to estimate either a single pose or multiple poses, meaning there is a version of the algorithm that can detect only one person in an image/video and another version that can detect multiple persons. At a high level, pose estimation happens in two phases:

• An input RGB image is fed through a convolutional neural network.
• Either a single-pose or multi-pose decoding algorithm is used to decode poses, pose confidence scores, keypoint positions, and keypoint confidence scores from the model outputs (see Figure: 4.2).

Pose—PoseNet will return a pose object that contains a list of keypoints and an instance-level confidence score for each detected person.

Pose confidence score—this determines the overall confidence in the estimation of a pose. It ranges between 0.0 and 1.0. It can be used to hide poses that are not deemed strong enough.

Keypoint—a part of a person’s pose that is estimated, such as the nose, right ear, left knee, right foot, etc. It contains both a position and a keypoint confidence score. PoseNet currently detects 17 key points illustrated in Figure: 4.3.

Keypoint Confidence Score—this determines the confidence that an estimated keypoint position is accurate. It ranges between 0.0 and 1.0. It can be used to hide key points that are not deemed strong enough.

Keypoint Position—2D x and y coordinates in the original input image where a key point has been detected.

The single-pose estimation algorithm. The high level structure of the process is reported in Figure: 4.4.

3 https://github.com/google-coral/project-posenet
The PoseNet algorithm trained both a ResNet [53] and a MobileNet [54] model. While the ResNet model has higher accuracy, its large size and many layers would make the load time and inference time less-than-ideal for any real-time application. We went with the MobileNet model as it is designed to run on mobile devices.

Conveniently, the PoseNet model is image size invariant, which means it can predict pose positions on the same scale as the original image, regardless of whether the image is downscaled. This means PoseNet can be configured
to have a higher accuracy at the expense of performance by setting the output stride we’ve referred to above at runtime.

The output stride determines how much we are scaling down the output relative to the input image size. It affects the size of the layers and the model outputs. The higher the output stride, the smaller the resolution of layers in the network and the outputs, and correspondingly their accuracy. In this implementation, the output stride can have values of 8, 16, or 32. In other words, an output stride of 32 will result in the fastest performance but lowest accuracy, while 8 will result in the highest accuracy but slowest performance, as summarized in Figure 4.5.

When PoseNet processes an image, what is returned is a heatmap along with offset vectors that can be decoded to find high confidence areas in the image that correspond to pose keypoints. The Figure:4.6 captures at a high-level how each of the pose keypoints is associated with one heatmap tensor and an offset vector tensor. Both of these outputs are 3D tensors with a height and width that we’ll refer to as the resolution. The resolution is determined by both the input image size and the output stride.

Heatmaps—Each heatmap is a 3D tensor of size resolution x resolution x 17, since 17 is the number of keypoints detected by PoseNet. For example, with an image size of 225 and an output stride of 16, this would be 15x15x17. Each slice in the third dimension (of 17) corresponds to the heatmap for a specific keypoint. Each position in that heatmap has a confidence score, which is the probability that a part of that keypoint type exists in that position. It can be thought of as the original image is broken up into a 15x15 grid, where the heatmap scores provide a classification of how likely each keypoint exists in each grid square.

Offset Vectors—Each offset vector is a 3D tensor of size resolution x resolution x 34, where 34 is the number of keypoints x 2. With an image size of 225
Figure 4.5: Relationship with output strip and performance and accuracy. Picture taken from: PoseNet website[74]

Figure 4.6: Heatmap and offset vector. Picture taken from: PoseNet website[74]

and an output stride of 16, this would be 15x15x34. Since heatmaps are an approximation of where the key points are, the offset vectors correspond in location to the heatmap points, and are used to predict the location of the key points by traveling along the vector from the corresponding heatmap point.
The first 17 slices of the offset vector contain the x of the vector and the last 17 the y. The offset vector sizes are on the same scale as the original image.

After the image is fed through the model, the algorithm performs a few calculations to estimate the pose from the outputs. The single-pose estimation algorithm, for example, returns a pose confidence score which itself contains an array of keypoints (indexed by part ID) each with a confidence score and x, y position.

4.4 Sequence To Image

This is one of the most important parts of the project, encoding skeleton data generated from video into an image. We will generate an image from the skeleton sequence. To transform a sequence into an image, skeleton data are normalized and then mapped into the RGB space (see Figure: 4.7). As a result, the high dimensionality of motion capture (mocap) sequences is reduced to 2D color images.

![Figure 4.7: Mapping between XYS and RGB space. Left picture shows axis corresponding to X position, Y position, and score of the keypoints.](image)

The RGB space is an alternative domain to explore data by mapping X, Y coordinate and S score into RGB components. Figure: 4.8 illustrates the different steps needed to transform a 3D skeleton sequence into RGB image.

![Figure 4.8: The picture shows 3 steps to generate RGB image from skeleton data.](image)
4.4.1 Skeleton to RGB image transformation

Let $S(v(f))$ be a sequence of body skeletons, where $V(f) = v_1(f), \ldots, v_N(f)$ denotes a set of body joint locations, $N$ is the number of joints, and $f$ is the index of the frame. For each body joint $i = 1, \ldots, N$, $v_i = (x_i, y_i, s_i), \forall (x_i, y_i, s_i) \in \mathbb{R}^3$. The sequence $S(V(f))$ can be represented in a matrix form as follows:

$$S(V(f)) = \begin{pmatrix}
  x_1(1) & \cdots & x_1(F) \\
  \vdots & \ddots & \vdots \\
  x_N(1) & \cdots & x_N(F)
\end{pmatrix}$$

where $F$ is the number of frames.

In this work, we map the values of $S(V(f))$ onto the RGB domain by normalizing all the values between 0 and 255. First, we extract the $X$, $Y$, and $S$ matrices from $S(V(f))$ and process each one separately. $S(V(f)) = (X, Y, S)$, where

$$X = \begin{pmatrix}
  x_1(1) & \cdots & x_1(F) \\
  \vdots & \ddots & \vdots \\
  x_N(1) & \cdots & x_N(F)
\end{pmatrix}$$

$$Y = \begin{pmatrix}
  y_1(1) & \cdots & y_1(F) \\
  \vdots & \ddots & \vdots \\
  y_N(1) & \cdots & y_N(F)
\end{pmatrix}$$

$$S = \begin{pmatrix}
  s_1(1) & \cdots & s_1(F) \\
  \vdots & \ddots & \vdots \\
  s_N(1) & \cdots & s_N(F)
\end{pmatrix}$$

Then, for each $x_i(f), y_i(f), s_i(f), i = 1, \ldots, N, f = 1, \ldots, F$, we compute $r_i(f), g_i(f), b_i(f)$, respectively the red, green, and blue values as follows:

$$r_i(f) = 255 \ast \frac{x_i(f) - \min(X)}{\max(X) - \min(X)}$$

$$g_i(f) = 255 \ast \frac{y_i(f) - \min(Y)}{\max(Y) - \min(Y)}$$

$$b_i(f) = 255 \ast \frac{s_i(f) - \min(S)}{\max(S) - \min(S)}$$

The minimum and maximum values of each matrix ($X$, $Y$, and $S$) are $\min(X)$, $\min(Y)$, $\min(S)$ and $\max(X)$, $\max(Y)$, $\max(S)$, respectively. We obtain the new matrices as follows:

$$R = \begin{pmatrix}
  r_1(1) & \cdots & r_1(F) \\
  \vdots & \ddots & \vdots \\
  r_N(1) & \cdots & r_N(F)
\end{pmatrix}$$
\[ G = \begin{pmatrix} g_1(1) & \ldots & g_1(F) \\ \vdots & \ddots & \vdots \\ g_N(1) & \ldots & g_N(F) \end{pmatrix} \]

\[ B = \begin{pmatrix} b_1(1) & \ldots & b_1(F) \\ \vdots & \ddots & \vdots \\ b_N(1) & \ldots & b_N(F) \end{pmatrix} \]

where \((r_i, g_i, b_i) \in [0, 255]^3\)

From these matrices, we create a single RGB image, where each matrix represents a channel in the final image. Figure 4.9 shows a transformation of a random sequence of one joint and 101 frames (4.9a) into a 17 × 101 pixel RGB image 4.9c.

The length of the video sequences is very large compared to the number of key points, for example, a sequence of 3 sec and 17 key points will create an image of 17 × 90.

In the case of a high-precision system, the frame rate will be more so the width of the image will be different in different systems. The sequence that will be used for classification will have a different frame rate depending on the system. This is a problem for classification because the size of the input will be different. We resize the image using a bi-cubic interpolation [55] to have fixed size 255 × 255 for all the sequences. According to H.S., H.L., and K.N. [55] enlarging an image makes more loss than shrinking it. To avoid this loss we will create a square image by replicating each row \(m\) times, where \(m\) is calculated as follows:

\[ m = \text{floor} \left( \frac{F}{N} \right) \]

In this example \(m=6\). Each row repeats 6 times, which gives an image of 102 × 90 pixels, then the interpolation is applied to fix the size to 255 × 255. Each distinctive row in the image represents a sequence of one marker, where the order of the markers are from 0 to 17.

We have used two methods to create the 255 × 255 image by repeating each row \(m\) times:

- First method: each row is repeated \(m\) times one after the other like is shown in Figure: 4.10b.

- Second method: all 17 rows are repeated \(m\) times together as a group, one group after each other, like in figure: 4.10a.

Other than these two encoding techniques we also have tried two different encoding techniques based on the number of keypoints to consider for the image classification, as Aubry et al. suggested in [10]. We are going to create 4 different datasets based on these encoding techniques.
4.5 Image Classification

We need to classify the images generated from videos to recognize the human activity. Also, we need to compare the different frameworks for image classification. By encoding the video sequence to single images we made the activity recognition task an image classification task. Now we need to classify those images in different activity classes. Image classification, which can be
defined as the task of categorizing images into one of several predefined classes.

In recent years, deep learning models that exploit multiple layers of nonlinear information processing, for feature extraction and transformation as well as for pattern analysis and classification, have been shown to overcome the challenges of classical approaches.

Among them, CNNs\(^4\) [56, 57] have become the leading architecture for most image recognition, classification, and detection tasks [58]. Despite some early successes [56, 57]; LeCun et al. [59]; Simard, Steinkraus, and Platt [60], deep CNNs (DCNNs\(^5\)) were brought into the limelight as a result of the deep learning renaissance Hinton, Osindero, and Teh [61], [62], and [63], which was fueled by GPUs, larger data sets, and better algorithms [64–67]. Several advances such as the first GPU implementation and the first application of maximum pooling (max pooling) for DCNNs have all contributed to their recent popularity.

The most significant advance, which has captured intense interest in DCNNs, especially for image classification tasks, was achieved in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 (Russakovsky et al., 2015), when the winning entry, by Krizhevsky et al. (2012), used a DCNN to classify approximately 1.2 million images into 1000 classes, with record-breaking results.

CNN’s are feed forward networks in that information flow takes place in one direction only, from their inputs to their outputs. Just as artificial neural networks (ANN) are biologically inspired, so are CNNs. An image is an input directly to the network, and this is followed by several stages of convolution and pooling. Thereafter, representations from these operations feed one or

---

\(^4\) Convolutional neural network
\(^5\) Deep convolutional neural network
more fully connected layers. Finally, the last fully connected layer outputs the class label as shown in figure 4.11.

**CNN architecture:**
CNN’s are generally made of 3 parts

- **Convolutional layers**– The convolutional layers serve as feature extractors, and thus they learn the feature representations of their input images. The neurons in the convolutional layers are arranged into feature maps. Each neuron in a feature map has a receptive field, which is connected to a neighborhood of neurons in the previous layer via a set of trainable weights. Inputs are convolved with the learned weights to compute a new feature map, and the convolved results are sent through a nonlinear activation function.

- **Pooling layers**– The purpose of the pooling layers is to reduce the spatial resolution of the feature maps and thus achieve spatial invariance to input distortions and translations. Max-pooling aggregation layers propagate the maximum value within a receptive field to the next layer. Formally, max-pooling selects the largest element within each receptive field.

- **Fully connected layers**– Several convolutional and pooling layers are usually stacked on top of each other to extract more abstract feature representations in moving through the network. The fully connected layers that follow these layers interpret these feature representations and perform the function of high-level reasoning.

We are going to train and compare results of our image classification task with the following CNNs:

- ResNet
- AlexNet
32 Proposed solution for activity recognition

- VGG
- SqueezeNet
- DenseNet
- Inception

4.5.1 Model architecture

In this chapter, we are going to discuss the architecture of different CNN we are going to use.

- ResNet[53]— Introduce by He et al. A residual neural network (ResNet) is an artificial neural network (ANN) of a kind that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers. Typical ResNet models are implemented with double or triple layer skips that contain non-linearities (ReLU) and batch normalization in between. There are 5 versions of ResNet based on number of layers: ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152.

- AlexNet[68]— AlexNet is the name of a CNN, designed by Krizhevsky, Sutskever, and Hinton. AlexNet competed in the ImageNet Large Scale Visual Recognition Challenge on September 30, 2012. The network achieved a top-5 error of 15.3%, more than 10.8 percentage points lower than that of the runner up. The original paper’s primary result was that the depth of the model was essential for its high performance, which was computationally expensive, but made feasible due to the utilization of graphics processing units (GPUs) during training. AlexNet contained eight layers; the first five were convolutional layers, some of them followed by max-pooling layers, and the last three were fully connected layers. It used the non-saturating ReLU activation function, which showed improved training performance over tanh and sigmoid.

AlexNet is considered one of the most influential papers published in computer vision, has spurred many more papers published employing CNNs and GPUs to accelerate deep learning. As of 2019, the AlexNet paper has been cited over 47,000 times.

- VGG[66]— VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000
4.5 IMAGE CLASSIFICATION

Figure 4.12: Left: the VGG-19 model as a reference. Middle: a residual network with 152 parameter layers. Right: a feed-forward neural network for reference.

classes. The input to the conv layer is of a fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3x3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations, it also utilizes 1x1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of Conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3x3 Conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the Conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2x2
Figure 4.13: An illustration of the architecture of AlexNet, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network’s input is 150,528-dimensional, and the number of neurons in the network’s remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Figure 4.14: VGG16 model structure. Different color boxes represent different types of layers. It takes 224x224x3 images as input.

pixel window, with stride 2. Three Fully-Connected (FC) layers follow a stack of convolutional layers (which has a different depth in different architectures): the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks. All hidden
layers are equipped with the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response Normalisation (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time.

- **SqueezeNet**[69]—SqueezeNet is the name of a deep neural network for computer vision that was released in 2016. SqueezeNet was developed by researchers at DeepScale, University of California, Berkeley, and Stanford University. In designing SqueezeNet, the authors’ goal was to create a smaller neural network with fewer parameters that can more easily fit into computer memory and can more easily be transmitted over a computer network. SqueezeNet was originally released on February 22, 2016. This original version of SqueezeNet was implemented on top of the Caffe deep learning software framework. Shortly thereafter, the open-source research community ported SqueezeNet to several other deep learning frameworks. As of 2018, SqueezeNet ships “natively” as part of the source code of several deep learning frameworks such as PyTorch, Apache MXNet, and Apple CoreML. Besides, 3rd party developers have created implementations of SqueezeNet that are compatible with frameworks such as TensorFlow.

- **DenseNet**[70]—DenseNet architecture can be considered as a new architecture; it is a logical extension of ResNet. ResNet architecture has a fundamental building block (Identity) where you merge (additive) a previous layer into a future layer. The reasoning here is by adding additive merges we are forcing the network to learn residuals (errors i.e. diff between some previous layer and the current one). In contrast, DenseNet paper proposes concatenating outputs from the previous layers instead of using the summation. In DenseNet, each layer obtains additional inputs from all preceding layers and passes on its feature-maps to all subsequent layers. Concatenation is used. Each layer is receiving a “collective knowledge” from all preceding layers. Since each layer receives feature maps from all preceding layers, the network can be thinner and compact, i.e. many channels can be fewer. The growth rate $k$ is the additional number of channels for each layer. DenseNets have several compelling advantages: they alleviate the vanishing gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters.

- **Inception**[71]—The GoogLeNet builds on the idea that most of the activation in a deep network are either unnecessary (value of zero) or redundant because of correlations between them. Therefore the most
efficient architecture of a deep network will have a sparse connection between the activation, which implies that all 512 output channels will not have a connection with all the 512 input channels. There are techniques to prune out such connections which would result in a sparse weight/connection. But kernels for sparse matrix multiplication are not optimized in BLAS or CuBlas (CUDA for GPU) packages which render them to be even slower than their dense counterparts.

GoogLeNet devised a module called inception module that approximates a sparse CNN with a normal dense construction. Since only a small number of neurons are effective as mentioned earlier, the width/number of the convolutional filters of a particular kernel size is kept small. Also, it uses convolutions of different sizes to capture details at varied scales (5x5, 3x3, 1x1).
Figure 4.16: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Figure 4.17: Two different types of Inception modules used for Inception network.

4.5.2 Transfer learning

Transfer learning (TL) is a method in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, the knowledge gained while learning to recognize cars could apply when trying to recognize trucks. This area of research bears some relation to the long history of psychological literature on transfer of learning, although formal ties between the two fields are limited. From the practical standpoint, reusing or transferring information from previously learned tasks to learn new tasks has the potential
to significantly improve the sample efficiency of a reinforcement learning agent.

In practice, very few researchers train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a ConvNet on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest.

- **ConvNet as a fixed feature extractor.** Take a ConvNet pretrained on ImageNet, remove the last fully-connected layer (this layer’s outputs are the 1000 class scores for a different task like ImageNet), then treat the rest of the ConvNet as a fixed feature extractor for the new dataset. In an AlexNet, this would compute a 4096-D vector for every image that contains the activations of the hidden layer immediately before the classifier. We call these features CNN codes. It is important for performance that these codes are ReLUd (i.e. thresholded at zero) if they were also thresholded during the training of the ConvNet on ImageNet (as is usually the case). Once you extract the 4096-D codes for all images, train a linear classifier (e.g. Linear SVM or Softmax classifier) for the new dataset.

- **Fine-tuning the ConvNet.** The second strategy is to not only replace and retrain the classifier on top of the ConvNet on the new dataset but to also fine-tune the weights of the pre-trained network by continuing the backpropagation. It is possible to fine-tune all the layers of the ConvNet, or it is possible to keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portion of the network. This is motivated by the observation that the earlier features of a ConvNet contain more generic features (e.g. edge detectors or color blob detectors) that should be useful to many tasks, but later layers of the ConvNet becomes progressively more specific to the details of the classes contained in the original dataset. In the case of ImageNet for example, which contains many dog breeds, a significant portion of the representational power of the ConvNet may be devoted to features that are specific to differentiating between dog breeds.

- **Pretrained models.** Since modern ConvNets take 2-3 weeks to train across multiple GPUs on ImageNet, it is common to see people release their final ConvNet checkpoints for the benefit of others who can use the networks for fine-tuning. For example, the Caffe library has a Model Zoo where people share their network weights.

Fine-tune and type of transfer learning. This is a function of several factors, but the two most important ones are the size of the new dataset (small or
big), and its similarity to the original dataset (e.g. ImageNet-like in terms of the content of images and the classes, or very different, such as microscope images). Keeping in mind that ConvNet features are more generic in early layers and more original-dataset-specific in later layers, here are some common rules of thumb for navigating the 4 major scenarios:

- New dataset is small and similar to the original dataset. Since the data is small, it is not a good idea to fine-tune the ConvNet due to overfitting concerns. Since the data is similar to the original data, we expect higher-level features in the ConvNet to be relevant to this dataset as well. Hence, the best idea might be to train a linear classifier on the CNN codes.

- New dataset is large and similar to the original dataset. Since we have more data, we can have more confidence that we won’t overfit if we were to try to fine-tune through the full network.

- New dataset is small but very different from the original dataset. Since the data is small, it is likely best to only train a linear classifier. Since the dataset is very different, it might not be best to train the classifier form the top of the network, which contains more dataset-specific features. Instead, it might work better to train the SVM classifier from activations somewhere earlier in the network.

- New dataset is large and very different from the original dataset. Since the dataset is very large, we may expect that we can afford to train a ConvNet from scratch. However, in practice, it is very often still beneficial to initialize with weights from a pre-trained model. In this case, we would have enough data and confidence to fine-tune through the entire network.

We have decided to train our model times to compare different approaches: train the whole network from the scratch, train only the last few layers (Shallow retraining) and fine-tune the whole network (deep retraining).

4.6 Prediction

We have used our trained model on the NTURGB+D dataset for predicting the activity from the video. Firstly, we take 3 seconds of video and compute the skeleton by PoseNet, then we classify the image created from the skeleton data with our model. The model returns an activity with a probability score.
Figure 4.18: Block diagram of prediction steps. As an input we will provide the video and it will output classification result.
In this chapter, we are going to discuss our proposed solutions for emotion recognition. Emotion recognition is second part of our project, in previous chapter we have discussed about first part of the project.

Facial emotions are important factors in human communication that help us understand the intentions of others. In general, people infer the emotional states of other people, such as joy, sadness, and anger, using facial expressions and vocal tones. According to different surveys, verbal components convey one-third of human communication, and nonverbal components convey two-thirds. Among several nonverbal components, by carrying emotional meaning, facial expressions are one of the main information channels in interpersonal communication. Therefore, it is natural that research of facial emotion has been gaining a lot of attention over the past decades with applications not only in the perceptual and cognitive sciences but also in effective computing and computer animations.

The second part of our project is aimed at classifying facial expressions from RGB video. It is a 3 step process: face detection, pre-processing, and classification.

For emotion recognition, we use deep learning based approaches. In the next sections, we will discuss the method in detail. Our objective is to predict the expression of the human face in real-time as fast and as accurately as possible. Constraints:

- **Latency:** Given an image, the system should be able to predict the expression immediately and transfer the result. Hence, there is a low latency requirement.

- **Interpretability:** Interpretability is important for still images but not in real-time. For still images, the probability of predicted expressions can be given.

- **Accuracy:** Our goal is to predict the expression of a face in the image as accurately as possible. Higher the test accuracy, the better our model will perform in the real world.

### 5.1 Selection of Emotion

Humans have a wide range of emotions and some of them are exclusive to some group of people. So in this project, we are going to focus on most
basic human emotions propose by Ekman in [73]: ANGER, DISGUST, FEAR, HAPPY, NEUTRAL, SAD, SURPRISE.

These are also the most useful and general emotions more or less everyone uses. Even though these seven emotions can be also divided into subcategories based on the intensity of the emotion but we will stick with these seven categories for now.

5.2 DATASET CREATION AND PRE PROCESSING

However, our goal here is to predict human expressions, but we have trained our model on both human and animated images. Since we had only approx 1500 human images which are very few to make a good model, so we took approximately 9000 animated images and leverage those animated images for training the model and ultimately do the prediction of expressions on human images. For better prediction, we have decided to keep the size of each image $350 \times 350$.

We have downloaded data from 4 different sources, reported in Table:5.1. We have a total of 10596 images. Out of which 1496 are human images and 9100 are animated images.
5.2 Dataset Creation and Preprocessing

<table>
<thead>
<tr>
<th>Database</th>
<th>Data Configuration</th>
<th>Web Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cohn-Kanade AU-Coded</td>
<td>• includes 486 sequences</td>
<td><a href="http://www.consortium.ri.cmu.edu/ckagreement/">http://www.consortium.ri.cmu.edu/ckagreement/</a></td>
</tr>
<tr>
<td>Expression Database</td>
<td>• 97 posers</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Each sequence begins with a neutral expression and proceeds to a peak expression.</td>
<td></td>
</tr>
<tr>
<td>FERG-DB[40]</td>
<td>• 55767 images corresponding to 7 categories.</td>
<td><a href="https://grail.cs.washington.edu/projects/deepexpr/ferg-db.html">https://grail.cs.washington.edu/projects/deepexpr/ferg-db.html</a></td>
</tr>
<tr>
<td></td>
<td>• six stylized animated characters.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• High resolution images.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 39 human subjects from 19 to 54 years old.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• High resolution images.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Six emotion adjectives by 60 Japanese subjects.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Image resolution of 256X256.</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: Four datasets we used to collection human emotion data.

Some of the examples of human images are reported in Figure: 5.1 and some examples of animated images are reported in Figure: 5.2.

As there is a lack of availability for the human dataset we had to use both human and animated data for training our model. Even though we have used both human and animated images for the training of our model, we have used the human and animated images separately from the beginning. We have also calculated the accuracy and performance matrices for both categories separately to evaluate how our model performs with human and animated images.
Overview of Initial Pre-Processing of Images:

- We have separated human and animated images right from the beginning and stored them in seven different folders namely: anger, disgust, fear, happy, neutral, sad, surprised.

- Remember, that there are two folders for each expression. One for human, and one for animated. In total, there are 14 folders.
• We made separate data frames for each expression. Therefore, there will be a total of 14 dataframes. 7 dataframes corresponding to seven expressions for human images. Similarly, 7 dataframes corresponding to seven expressions for animated images.

• Then combined all of the dataframes of humans. There are 1496 human images in total. After this, we combined all of the dataframes for animated images. There are 9100 animated images in total.

• So, now we have two dataframes, one for human images and another one for animated images.

Here, we have followed 5 simple steps for processing and changing our images such that they become suitable for feeding to the model for training.

• Step-1: Converting images to gray-scale.
• Step-2: Detect face in the image using OpenCV HAAR Cascade\(^1\).
• Step-3: Crop the image to the face.
• Step-4: Resize the image to 350 × 350.
• Step-5: Finally save the image.

After processing the images, we got images like the ones reported in Figure:5.3. Here, our result does not depend on the color of the image. Further-

![Figure 5.3: some samples of cropped images from our dataset](https://www.docs.opencv.org/trunk/db/d28/tutorial_cascade_classifier.html)

more, many human images were inherently black and white. So, to make

---

\(^1\) [https://www.docs.opencv.org/trunk/db/d28/tutorial_cascade_classifier.html](https://www.docs.opencv.org/trunk/db/d28/tutorial_cascade_classifier.html)
all of the images the same we decided to convert the rest of the images to
gray-scale so that during training, the model should treat all of the images
equally regardless of their color.

5.3 IMAGE CLASSIFICATION

In this case, emotion detection is an image classification problem because
we are going to classify images in 7 different classes for the detected faces
in each frame of the video. We will choose the face corresponding to the
skeleton detected by the first part. Image classification is a very common and
well-researched topic in computer science. In recent years image classification
obtains very good results with deep learning approaches. In our thesis, we
are going to use a convolutional neural network to classify images.

5.3.1 Transfer learning

We will use the transfer learning technique to classify images. Creating
a Convolution Neural Network from scratch is not an easy task. All the
more so, the problem exacerbates when we don’t have computation power
because convolution itself is a computationally quite expensive operation.
Even though we don’t have many images, creating a CNN layer and applying
convolution operation would take a lot of time.

So, to save ourselves from this over-head we decided to use transfer learn-
ing. Transfer learning is a concept according to which we can transfer the
learning of other pre-trained models to our data. Instead of training our
custom neural network, we can use other popular pre-trained models and
pass our data to those models and ultimately get the features for our images.

Inherently, the convolution layer generates features for the images. It
applies convolution operation on each pixel of the images and ultimately
generates ‘n’ dimensional array which is nothing but learned features of
the images. The final features of the image which we get at the end of
the convolution neural network are known as bottleneck features. These
bottleneck features are the learned features of the images which are then fed
to the MLP which acts as a top-model. This MLP then reduces loss function
and updates the weights in MLP and kernels/filters in CNN.

Now, for our task, we have chosen the VGG-16 pre-trained neural network
to generate bottleneck features. VGG-16 network contains 16 layers out of
which 13 layers are convolution layers. This neural network is well trained
on ImageNet[72] dataset which contains millions of images. Fortunately,
this VGG-16 trained network is available in Keras. So, we have loaded this
pre-trained VGG-16 network and loaded ImageNet weights in it.
5.3.2 Model architecture

It needs to be remembered, that here we haven’t included the top MLP part of VGG16, because we will build our custom top MLP model later. We have loaded the VGG16 network only till the bottleneck features part. The diagram in Figure 5.4 explains this clearly. We have passed each image one-by-one through this network, generate bottleneck features, and stored them in NumPy array. We just have to use predict function of VGG16 and generate bottleneck features for the images.

Figure 5.4: VGG16 model architecture used for generating bottleneck features from input image.
Finally, for all of our 10596 images, we have generated bottleneck features and saved them in a disk. In this way, we have used transfer learning of VGG16 model for our task.

Now, we already have bottleneck features for each of our images. Now our task is to create a top-model means MLP model which will take the bottleneck feature of each image one-by-one and reduce Multi-Class Log-Loss/Cross-Entropy loss. For this, we have designed the following Neural network.

```python
model = Sequential()
model.add(Dense(512, activation='relu', input_dim = input_shape))
model.add(Dropout(0.1))
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(64, activation='relu'))
model.add(Dense(output_dim = 7, activation='softmax'))
```

As you can see above we have 5 dense fully connected layers. All contain relu activation units. First contains 512 activation units, second contains 256 activation units, third contains 128 activation units, fourth contains 64 activation units and the fifth layer is the output layer which contains 7 softmax units. These softmax units are nothing but generalization of logistic regression to a multi-class setting. In a nutshell, it is a multi-class log loss. It will generate 7 probability values corresponding to seven classes. The sum of all the probability value is one. This result will then feed to final cross-entropy loss which is minimized through back-propagation. In this way, our MLP model will be trained enough to classify facial expressions in the images.

In the above model, you must have observed that we have used a very small to no dropout rate. Initially, we began with a ‘0.5’ dropout rate between the first four layers. But, after 12 epochs we observed that our training and validation loss was not reducing. We gradually decrease our dropout rate and we observed that our both training loss and validation decreased and thereby our training and validation accuracy increased.
In this chapter we are going to discuss the algorithms and configurations used for our experiments in details. As it is a machine learning project it has a life cycle like the one reported in Figure 6.1.

We can divide the project into two subparts: Activity and Emotion recognition. Each machine learning project can be divided into two phases: training phase and deployment phase. The training phase can also divide into dataset creation and training.

6.1 ACTIVITY RECOGNITION DATASET CREATION

The activity recognition part takes video and predict activities. Activity recognition is based on the deep learning model. Like any other machine learning and deep learning problems this also has the dataset creation or the pre-processing part. For our problem dataset creation and pre-processing part involved generating images form the videos. The images used for training are created by encoding skeleton data generated by PoseNet.

We have explained the process of encoding skeleton data to images in detail in previous chapter. For creating the dataset we have used a server with Nvidia 1080ti GPU. We have created a function called `image_generator` which takes video path as input and return encoded image. The image generator takes the video path as input then initializes `VideoCapture`. Until the video capture is open it processes each frame as follows. Firstly, it passes the frame
to posenet and gets the pose score, keypoint score, and key point coordinate as return from posenet. Then we convert the key point coordinates in the image scale. we create the X, Y and S matrix as discuss in the previous chapter. After processing all the frames of the video we normalize the X, Y, and S matrix and Stack them as R, G, and B channel of an image.

After we generated all the images corresponding to all the videos in the NTURGB+D dataset. We split the images in different datasets. We created different datasets based on the number of classes and according to dataset benchmark evaluation rules. We have selected 17 classes important to our application out of 60 classes. NTURGB+D dataset has a standard evaluations rule for benchmark, they defined precise criteria for two types of evaluation method.

• **Cross-Subject Evaluation**–We split the 40 subjects into training and testing groups. Each group consists of 20 subjects. For this evaluation, the training and testing sets have 40,320 and 16,560 samples respectively. The IDs of the training subjects in this evaluation are: 1, 2, 4, 5, 8, 9, 13, 14, 15, 16, 17, 18, 19, 25, 27, 28, 31, 34, 35, 38. remaining subjects used for the testing set.

• **Cross-View Evaluation**–We have picked all the samples of camera 1 for testing and samples of cameras 2 and 3 for training. For this evaluation, the training and testing sets have 37,920 and 18,960 samples, respectively.

Other than these categories we also have 2 different encoding techniques(stretching and repeating) and number of keypoints(17 and 14) we have used for encoding.

The different datasets we have created are:

- Stretching, 60 classes, 17 keypoints, and Cross-Subject(S\_60\_17\_CS)
- Repeating, 60 classes, 17 keypoints, and Cross-Subject (R\_60\_17\_CS)
- Stretching, 60 classes, 14 keypoints, and Cross-Subject(S\_60\_14\_CS)
- Repeating, 60 classes, 14 keypoints, and Cross-Subject(R\_60\_14\_CS)
- Stretching, 17 classes, 17 keypoints, and Cross-Subject(S\_17\_17\_CS)
- Repeating, 17 classes, 17 keypoints, and Cross-Subject(R\_17\_17\_CS)
- Stretching, 17 classes, 14 keypoints, and Cross-Subject(S\_17\_14\_CS)
- Repeating, 17 classes, 14 keypoints, and Cross-Subject(S\_17\_14\_CS)
- Stretching, 60 classes, 17 keypoints, and Cross-View(S\_60\_17\_CV)
Repeating, 60 classes, 17 keypoints, and Cross-View(R_60_17_CV)
Stretching, 60 classes, 14 keypoints, and Cross-View(S_60_14_CV)
Repeating, 60 classes, 14 keypoints, and Cross-View(R_60_14_CV)
Stretching, 17 classes, 17 keypoints, and Cross-View(S_17_17_CV)
Repeating, 17 classes, 17 keypoints, and Cross-View(R_17_17_CV)
Stretching, 17 classes, 14 keypoints, and Cross-View(S_17_14_CV)
Repeating, 17 classes, 14 keypoints, and Cross-View(S_17_14_CV)

We have performed our training and experiments with these 16 datasets. Mostly we will focus on the dataset for 17 classes because it’s more important for our application.

6.2 Activity Recognition Training

For the training of our models, we have used PyTorch library and python languages with mlflow\(^1\) library to track all the experiments and save the models. We have used TORCHVISION models for transfer learning. We have done 3 different types of training for each model.

- **Deep retraining**– In this technique, we have fine-tuned the whole pre-trained models.
- **Shallow retraining**– In this technique, we have just trained the last few layers of the pre-trained models.
- **Training from scratch**– In this technique, we have trained the models from scratch without using pre-trained models.

6.3 Emotion Recognition Dataset Creation and Pre-processing

The emotion recognition part is implemented using the TensorFlow and Keras library. For emotion recognition, we have used deep learning based approach. We have a fine-tuned VGG16 model for this purpose. As with any machine learning project, this project also has dataset creation and pre-processing part. We have used human and animated images. We have separated the human and animated images from the beginning and stored them separately. 7 folders were corresponding to 7 emotions, so we created 14 folders in total, 7 for human emotion and 7 for animated emotion. Then we generated 14 data frames from those folders. We have converted all the human emotions

\(^1\) https://github.com/mlflow/mlflow
dataframes to a single dataframe the same for the animated dataframes. We have split the data in the train, test, and validation sets. We got 6 datasets after the splitting, 3 for the human data and 3 for the animated data. For the training part only, we have combined the human and animated images, while for test and validation, we kept the human and animated data separate. By doing so we have the flexibility to monitor the performance of our model in the human and animated image constantly.

After that, we had only five dataframes: train, human test, human validation, animated test, and animated validation.

6.4 EMOTION RECOGNITION MODEL TRAINING

We have used a transfer learning approach for the training. We used the pre-trained VGG16 model. We get the bottleneck features from the pre-trained model. These bottleneck features are the learned features of the images which are then fed to the MLP which acts as a top-model. This MLP then reduces loss function and updates the weights in MLP and kernels/filters on CNN. We used the pre-trained VGG16 network to generate bottleneck features for all the 5 dataframe we have created. After that, we will have 5 dataframe with bottleneck features corresponding to 5 dataframes. Then we train the 5 dense fully connected layers we have created with the bottleneck feature dataframe generated from the training dataframe. For validation and testing, we use separate dataframe for human and animated images. Which allowed us to measure the performance of our model both for human and animated images separately.

6.5 SYSTEM ARCHITECTURE

![Figure 6.2](image)

**Figure 6.2:** Block diagram of our system to detect human activity and emotion in real time

In this chapter, we are going to discuss our end-to-end system, depicted in Figure:6.2, to recognize human activity and emotion.
• **PoseNet:** As an input to the system we feed a RGB video or video stream directly from a camera. The first module in the pipeline of our system is PoseNet. PoseNet module detects human skeletons in each frame of the video. PoseNet module returns the key points and scores of the key points of each frame.

• **Sequence-to-Image:** After the skeleton detection by the PoseNet module, the Sequence-to-image module generates RGB images by encoding skeleton data. The sequence-to-image module takes 90 frames or 3 sec video to generate each image.

• **Activity Recognition:** The image generated by the Sequence-to-image module feed as input to the CNN model. The trained CNN model predicts the activity present in the video sequence.

• **Face Recognition:** The face recognition process happens parallel to the sequence-to-image module. We have used Haar-cascade Detection by OpenCV to detect faces in each frame but we have also take into account the skeleton provided by PoseNet to extract the face correspond to the skeleton. Module output the resized image of the face.

• **Emotion Recognition:** In this module, we feed the image of the face to the trained CNN model. The CNN model predicts an emotion based on the face.

• **Compute Output:** In this module, we take into account the predicted emotion and activity predicted by the previous modules. We apply some logical reasoning to discard errors or low probability predictions. Finally this module output the video with overlap text mentioning activity and emotion.

For the skeleton detection from the videos, we have used the pre-trained model PoseNet developed by Google. We have also used USB accelerator Google Coral, the onboard Edge TPU co-processor capable of performing 4 trillion operations (tera-operations) per second (TOPS), using 0.5 watts for each TOPS. Google Coral helped to achieve very good performance to detect skeleton in the video in real-time.
The proposed method for activity and emotion recognition can be divided into two sub parts: activity and emotion recognition. We are going to discuss both parts one by one. The proposed system takes RGB videos as input and returns activity and emotion detected in the video. In this chapter, we present experiments and results of activity recognition. As both parts for activity and emotion recognition are based on deep learning approaches, we are going also to focus on different data pre-processing techniques, training results, and model comparisons.

In this section we are going to discuss about our activity recognition method. We have used transfer learning-based image classification algorithms. We have designed various training datasets and tests to compare the results of different models and the performance of different data generation methods. In this section, we are going to present a detailed comparison of different experiments and their results.

7.1 Data Representation

The first thing we are going to focus on in this section is data representation. We have put lot of effort to create different data encoding techniques and distribution of the data. The different encoding techniques we have used based on sequence to image creation are:

First method – We get the keypoints from each body part for each frame and get number of keypoints $\times$ number of frames $\times$ 3 (3 for $x$, $y$, and $s$ value of the keypoints) matrix. Now we stretch the rows of the matrix by replicating each row $f$ ($f = \text{floor}(\text{number of frames} / \text{number of key}$

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{image.png}
\caption{Image generated by stretching method}
\end{figure}

by replicating each row $f$ ($f = \text{floor}(\text{number of frames} / \text{number of key}$

55
Second method – We get the keypoints from each body part for each frame and get number – of – keypoints × number – of – frames × 3 (3 for x, y, and s value of the keypoints) matrix. Now we repeat the number – of – keypoints × number – of – frames × 3 matrix row wise f (f = floor( number of frames / number of key points) ) times to create almost square matrix as depicted in Figure:7.2. We call this method repeating(R) for quick reference from now on.

![Image generated by repeating method](image)

**Figure 7.2:** Image generated by repeating method

Other than the different techniques to generate the image, we have two different encoding techniques based on the number of keypoints:

- Firstly, we have considered all 17 keypoints returned by the PoseNet.
- Secondly, we have considered only 14 keypoints. We have not considered 3 keypoints(nose, and two eyes) because we are more interested about activity of the human. These 3 key points doesn’t have strong influence on physical activity and movements.

We have tried another type of the encoding technique where we have replaced the score of the keypoints with average of X and Y coordinate of the keypoints to create images from key points of the skeletons. We have found out that the performance is better when we use the score of the keypoint as 3rd channel of the RGB image instead of average of X and Y coordinate. We are going to compare all the different tests later in this section.

PoseNet returns the skeleton as the order shown in the diagram presented in Figure:7.3. As we have explained above we have tested with 17 and 14 joints and encoded them in following order:

- **Order 1 =** 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
- **Order 2 =** 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
Lastly, we have done all the experiments with either 60 classes or 17 classes and all the datasets were split according to the cross subject and cross view methods to ensure avoidance of data leakage. Data leakage, also called target leakage, is a problem that affects several stages of the machine learning life cycle, from data collection to model evaluation. Data leakage in supervised learning is an unintentional introduction of information about the target which should not be available to learn from.

For the cross view each class had 632 and 316 samples for training and validations respectively. In case of the cross-subject data each class had 672 and 276 samples respectively for training and validation.

### 7.2 Comparison between Stretching and Repeating Encoding Techniques

In this section, we are going to compare the two encoding techniques: stretching and repeating. We are going to see how these techniques affect the models by comparing 3 models.

---

**Figure 7.3:** Schematic representation of 17-joint skeleton created by PoseNet
In Table 7.1 we compare the stretching and repeating techniques, we have used 3 deep learning models for comparison namely ResNet, DenseNet, and SqueezeNet. For these experiments we have used all pre-trained models and fine tuned the whole model with our datasets. To give a general feedback about the results, we have shown the validation accuracy for each case.

We have used all 60 classes and 17 classes to compare the results.

If we analyze the results, we can notice that in every case the stretching encoding technique gave better accuracy than the repeating technique. If we specifically consider the ResNet model trained for 17 classes got 5% higher accuracy in case of stretching technique compared to repeating technique.

We can conclude that the stretching technique performs better than the repeating technique in all the cases.

<table>
<thead>
<tr>
<th>number of class</th>
<th>model</th>
<th>stretching</th>
<th>repeating</th>
<th>stretching</th>
<th>repeating</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>ResNet</td>
<td>88%</td>
<td>83%</td>
<td>86%</td>
<td>83%</td>
</tr>
<tr>
<td>17</td>
<td>DenseNet</td>
<td>87%</td>
<td>83%</td>
<td>86.5%</td>
<td>83%</td>
</tr>
<tr>
<td>17</td>
<td>SqueezeNet</td>
<td>82%</td>
<td>77%</td>
<td>78%</td>
<td>77%</td>
</tr>
<tr>
<td>60</td>
<td>ResNet</td>
<td>79%</td>
<td>74%</td>
<td>78%</td>
<td>72%</td>
</tr>
<tr>
<td>60</td>
<td>DenseNet</td>
<td>80%</td>
<td>74%</td>
<td>77%</td>
<td>73%</td>
</tr>
<tr>
<td>60</td>
<td>SqueezeNet</td>
<td>70%</td>
<td>64%</td>
<td>67%</td>
<td>62%</td>
</tr>
</tbody>
</table>

Table 7.1: Comparison between 2 encoding methods based on validation accuracy obtained by the models. The table shows effect of stretching and repeating methods on validation accuracy.
7.3 Comparison between 17 keypoints and 14 keypoints for encoding

PoseNet returns 17 keypoints for each human in the frame. We have considered 17 and 14 keypoints to create two different kinds of datasets and compared the effect on the final models to understand the importance of the number of keypoints used for encoding.

<table>
<thead>
<tr>
<th>number of class</th>
<th>model</th>
<th>Cross-subject</th>
<th>Cross-view</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>SqueezeNet</td>
<td>82%</td>
<td>78%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>81%</td>
<td>77.7%</td>
</tr>
<tr>
<td>17</td>
<td>DenseNet</td>
<td>87%</td>
<td>86%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>86%</td>
<td>85%</td>
</tr>
<tr>
<td>17</td>
<td>Inception</td>
<td>87%</td>
<td>85%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>85%</td>
<td>85%</td>
</tr>
</tbody>
</table>

Table 7.2: Comparison between two encoding methods (17 and 14 keypoints) based on validation accuracy obtained by the models

In Table 7.2 we have compared accuracy obtained by three models using two different datasets respectively generated with 17 and 14 keypoints. All three models SqueezeNet, DenseNet, and Inception were pre-trained and we have fine-tuned the whole network using our datasets.

If we analyze the results, we can notice that using all 17 keypoints for encoding the models perform a little better than only using 14 keypoints.

In all three models we manage to get 1% more accuracy with using more key points for encoding. With this result, we can conclude that using all keypoints will result in better accuracy.

7.4 Comparison between cross-subject and cross-view

Cross-subject and Cross-view are different types of splitting we have used for creating data sets. In short, in case of the cross-subject we use video of different subjects for training and different subjects for validation to ensure there is no data leakage from training set to validation set. For cross-view we used different camera angles for training and different camera angle for validation to ensure the model will work irrespective of camera angles.
<table>
<thead>
<tr>
<th>number of class</th>
<th>model</th>
<th>Cross-subject</th>
<th>Cross-view</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>ResNet</td>
<td>88%</td>
<td>86%</td>
</tr>
<tr>
<td>17</td>
<td>DenseNet</td>
<td>87%</td>
<td>86.5%</td>
</tr>
<tr>
<td>17</td>
<td>SqueezeNet</td>
<td>82%</td>
<td>78%</td>
</tr>
</tbody>
</table>

**Table 7.3:** Comparison between cross-view and cross-subject evaluation methods.

Table 7.3 shows validation accuracy of three models with respect to cross-subject and cross-view. All of the experiments for this purpose have been done with 17 classes. We can see that for cross-view the accuracy is lower compared to the cross-subject. In these 6 experiments, ResNet performed best.

### 7.5 Comparison between 2 Encoding Techniques Using Score and Average of Keypoints for 3rd Channel for Image

In this section, we are going to compare two types of encoding techniques. First, when we encoded the skeleton data to an RGB image we used the score of the key points to generate the 3rd channel of the RGB images. Second, when we created RGB images from skeleton data we have used average of the X and Y to generate 3rd channel of the RGB image instead of using score.

<table>
<thead>
<tr>
<th>number of class</th>
<th>model</th>
<th>Cross-subject</th>
<th>Cross-view</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>SqueezeNet</td>
<td>82%</td>
<td>81%</td>
</tr>
<tr>
<td>17</td>
<td>DenseNet</td>
<td>87%</td>
<td>87%</td>
</tr>
<tr>
<td>17</td>
<td>ResNet</td>
<td>88%</td>
<td>87%</td>
</tr>
</tbody>
</table>

**Table 7.4:** Comparison of model performance when we used 3rd channel as score of keypoints and average of X and Y position of Keypoints.
In Table 7.4 we have shown how these 2 encoding techniques affect accuracy of the models. For this comparison we have used 3 models. All of the experiments were performed on 17 classes. There was no difference in the datasets other than encoding of 3rd channel in two different ways. So from this comparison we can clearly see that the accuracy is little better when we use the 3rd channel of the RGB images to encode score of the key points. The cross-view average technique achieved more accuracy than the score technique, for cross-subject it is the opposite.

7.6 MODEL COMPARISON

In this section, we are going to discuss and compare about the different CNN architectures. We have used ResNet, AlexNet, VGG16, SqueezeNet, DenseNet, and InceptionV3 pre-trained models. We have used transfer learning techniques to train all the models. For comparison of different models we have trained all the models with same dataset using deep retraining method. We have ensured that all the models trained exactly with same dataset to understand how the model architecture affect the accuracy of the models.

<table>
<thead>
<tr>
<th>number of class</th>
<th>model</th>
<th>cross-subject</th>
<th>cross-view</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>ResNet</td>
<td>88%</td>
<td>86%</td>
</tr>
<tr>
<td>17</td>
<td>AlexNet</td>
<td>82%</td>
<td>78%</td>
</tr>
<tr>
<td>17</td>
<td>VGG16</td>
<td>84%</td>
<td>81%</td>
</tr>
<tr>
<td>17</td>
<td>SqueezeNet</td>
<td>82%</td>
<td>78%</td>
</tr>
<tr>
<td>17</td>
<td>DenseNet</td>
<td>87%</td>
<td>86%</td>
</tr>
<tr>
<td>17</td>
<td>Inception</td>
<td>87%</td>
<td>85%</td>
</tr>
</tbody>
</table>

*Table 7.5:* Comparison of validation accuracy achieved by different models while using same configuration and data.

In Table 7.5 we have compared the validation accuracy of the models. In this experiment we have used ResNet152, DenseNet201, and InceptionV3 versions of ResNet, DenseNet, and Inception respectively. As we can see ResNet achieves the best accuracy among all the models. All the models
perform well for cross-subject than cross-view. We have used deep retraining which means tuning the whole network.

## 7.7 Compare 3 Types of Training

Transfer learning can be compared to learning a language. For someone who already knows several languages, it is easier to learn a new one compared to someone who only speaks one language. Thanks to previous knowledge, the learning process is easier. It is the same for neural networks. We start with models designed for image classification. The network is already able to classify images, and with some adaptation of the parameters it is also capable to classify motion sequence images.

<table>
<thead>
<tr>
<th>number of class</th>
<th>model</th>
<th>Cross-subject</th>
<th>Cross-view</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Deep retraining</td>
<td>Shallow retraining</td>
</tr>
<tr>
<td>17</td>
<td>ResNet</td>
<td>88%</td>
<td>54%</td>
</tr>
<tr>
<td>17</td>
<td>AlexNet</td>
<td>82%</td>
<td>46%</td>
</tr>
<tr>
<td>17</td>
<td>VGG16</td>
<td>84%</td>
<td>42%</td>
</tr>
<tr>
<td>17</td>
<td>SqueezeNet</td>
<td>82%</td>
<td>48%</td>
</tr>
<tr>
<td>17</td>
<td>DenseNet</td>
<td>87%</td>
<td>58%</td>
</tr>
<tr>
<td>17</td>
<td>Inception</td>
<td>87%</td>
<td>50%</td>
</tr>
</tbody>
</table>

**Table 7.6:** Comparison among deep retraining, shallow retraining, and training from scratch techniques. Table shows validation accuracy achieved by 6 models with 3 techniques.

The table 7.6 shows comparison between different types of training methods. The table shows accuracy of validation set for 6 different models achieved with 3 different type of training. The table shows deep retraining of the mod-
els constantly performed way better than shallow, and training from scratch method. Among 3 types of training techniques we have achieved worse performance with training the models from scratch because we do not have enough data to train a deep learning network from scratch.

7.8 COMPARE 6 DIFFERENT MODELS PERFORMANCE

As we have mentioned before that performance is key factor for this project so it is very important to have an efficient model. In table 7.7 we have showed the training time, prediction time, and model size of different models we have used for activity recognition.

<table>
<thead>
<tr>
<th>Model name</th>
<th>Training time (min)</th>
<th>Prediction time (sec)</th>
<th>Model size (Mb)</th>
<th>Training time (min)</th>
<th>Prediction time (sec)</th>
<th>Model size (Mb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet</td>
<td>192</td>
<td>0.089</td>
<td>223</td>
<td>624</td>
<td>0.0898</td>
<td>223</td>
</tr>
<tr>
<td>AlexNet</td>
<td>20.1</td>
<td>0.017</td>
<td>218</td>
<td>96</td>
<td>0.017</td>
<td>218</td>
</tr>
<tr>
<td>VGG16</td>
<td>47.4</td>
<td>0.056</td>
<td>513</td>
<td>196</td>
<td>0.057</td>
<td>513</td>
</tr>
<tr>
<td>SqueezeNet</td>
<td>26.2</td>
<td>0.063</td>
<td>2.9</td>
<td>90</td>
<td>0.072</td>
<td>3</td>
</tr>
<tr>
<td>DenseNet</td>
<td>108</td>
<td>0.157</td>
<td>71</td>
<td>384</td>
<td>0.168</td>
<td>71</td>
</tr>
<tr>
<td>Inception</td>
<td>132</td>
<td>0.044</td>
<td>94</td>
<td>504</td>
<td>0.048</td>
<td>948</td>
</tr>
</tbody>
</table>

Table 7.7: Table shows training time, prediction time, and model size of 6 different models in case of 17, and 60 classes activity classification

From the table we can see that SqueezeNet has lowest prediction time and model size. ResNet and DenseNet has very high training time compare to other models. All the experiments for this table done with 17 keypoints, cross-subject, and straching configuration method. One Nvidia 1080 ti GPU used for trainings.
7.9 MODEL TRAINING

In this section we will dive deep into the training process and results. We are only going to discuss experiment with ResNet and DenseNet. Even though we have trained 6 different types of CNN architectures, ResNet and DenseNet performed consistently well, so we focus only on these two.

7.9.1 ResNet

We have discussed the Resnet architecture in details in previous chapters. We have used pre-trained model of ResNet provided by Torchvision\(^1\) library. Torchvision provides different versions of the ResNet models:

- ResNet-18
- ResNet-34
- ResNet-50
- ResNet-101
- ResNet-152

The numbers in the name of the model represent number of layers present in that model. In our experiments we have used ResNet-152 because it provides most accurate results than other versions. Figure:7.4 shows the training accuracy and validation accuracy of the model during training time. As we can see training accuracy started very low, then increased gradually and saturated around 90%. While the validation accuracy starts from 60% then achieve max 88% accuracy. Figure:7.5 shows the training and validation loss of the model during training time. Train loss started from 2 and then model achieved lowest loss of 0.4 around 43rd epoch. The validation loss gone down slowly and achieved 0.2 loss. We can notice that at the beginning of the training, for some epochs, the model performs better with validation set than with training set. It is because we are using pre-trained model and the models use dropout or similar generalization techniques which are used during the training, but in validation we use the whole model without dropout or generalization techniques.

7.9.1.1 Confusion matrix

In the field of machine learning and specifically the problem of classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically

\(^1\) [https://pytorch.org/docs/stable/torchvision/models.html](https://pytorch.org/docs/stable/torchvision/models.html)
a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in a actual class while each column represents the instances in an predicted class (or vice versa). The name stems from the fact that it makes it easy to see if the system is confusing two classes. It is a special kind of contingency table, with
two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).

A table of confusion (sometimes also called a confusion matrix), is a table with two rows and two columns that reports the number of false positives, false negatives, true positives, and true negatives. This allows more detailed analysis than mere proportion of correct classifications (accuracy). Accuracy is not a reliable metric for the real performance of a classifier, because it will yield misleading results if the data set is unbalanced (that is, when the numbers of observations in different classes vary greatly).

The figure 7.6 shows the confusion matrix for ResNet. We have used the confusion matrix to create recall and precision for our classes.

7.9.1.2 Precision matrix

To get the value of precision we divide the total number of correctly classified positive examples by the total number of predicted positive examples. High Precision indicates that an example labeled as positive is indeed positive (small number of false positives). Precision is given by the relation:

\[
\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}
\]

Figure 7.7, showing precision matrix obtained from confusion matrix of ResNet. From the precision matrix we can see what are the classes where
our model have less precision. As we can see the class Ao31 and Ao32 which represent "point to something" and "taking a selfie". Both of the actions have kind of same hand movements. Except for these 2 classes, our model has quite good precision.
7.9.1.3 Recall Matrix

Recall can be defined as the ratio of the total number of correctly classified positive examples divided by the total number of positive examples. High Recall indicates that the class is correctly recognized (small number of false negatives).

Recall is given by the relation:

\[
\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}
\]

Recall (also known as sensitivity) is the fraction of the total amount of relevant instances that were actually retrieved. Recall in this context is also referred to as the true positive rate or sensitivity.

Figure 7.8 shows the recall for each class. Recall is the ratio of correct positive predictions to the overall number of positive examples in the dataset. Recall matrix shows how good is our model to predict that a sample belongs to a class. From the matrix we can see that the A031 has lowest recall of 0.66 because model predicted many of the A031 samples as A032 class because model could not distinguish these two classes very well. To increase the recall either we could add more samples for both the classes or we could increase the weight for these 2 classes in future training.

The precision and recall prove that our model is very good for predicting all the 17 classes.

7.9.2 Densenet

Densenet is a very popular network for image classification task. We have discussed the architecture of the DenseNet in a previous chapter. In this section we are going to focus on experiments and results. As all other experiments we have done, also here we have used pre-trained models by torchvision. We have used transfer learning, more specifically deep re-training technique in this case. Torchvision provides some versions of Densenet models:

- Densenet-121
- Densenet-161
- Densenet-169
- Densenet-201

For this experiment we have used Densenet-201 version, it has 201 layers. Figure 7.9 shows training and validation accuracy with respect to epochs. We can see that training accuracy increased slowly and reached maximum of 92% by 50th epoch. Where validation accuracy started quite high then
quickly reached saturation. We can see that in first couple of epochs validation accuracy is more than the training accuracy because of the generalization techniques and pre-trained model use. 7.10 figure shows training and validation loss with respect to epochs. The training loss went down gradually.
with epochs where validation loss increased little bit towards the end which means it started over fitting.

**Figure 7.11:** confusion matrix of DenseNet

### 7.9.2.1 Confusion matrix

Figure: 7.11 represents the confusion matrix of validation data. The confusion matrix shows how many samples were predicted correctly by our model. We have calculated also recall and precision matrices from the confusion for further investigation of model performance.

### 7.9.2.2 Precision matrix

Figure: 7.12 shows precision of each class. As we have seen in our ResNet model the class A031 and A032 had lowest precision. In DenseNet also those 2 classes has lowest precision of 0.71 and 0.74. The precision of class A032 is better in case of DenseNet compared to ResNet. While ResNet performed better in case of class A031. All other classes have very good precision.

### 7.9.2.3 Recall matrix

Figure: 7.13 shows the recall matrix created from the confusion matrix of our DenseNet model. We can see the classes A031 and A032 have lowest recall in this case also. A031 has recall of 0.78 and A032 has recall of 0.74. If we compare the recall of these 2 classes with the ResNet, we will see that A031
recall is higher in DenseNet at the expense of precision, and the opposite for A032.
7.10 REAL TIME PERFORMANCE

In this section we will take look at the real-time performance of the Activity recognition module of our system. The activity recognition module takes a video as input and return activity as output. Before using the model for prediction we need to extract the keypoints for each frame using Posenet and generate image from those keypoints. As Posenet generate keypoints in 21fps and most of the videos are 30fps, we will have some delay from that. Performance of 3 different models are as follows:

- **ResNet:** Video with 114 frames took 5.36606 sec when Posenet produced keypoints 21 fps.
- **DenseNet:** Video with 114 frames took 5.4146 sec when Posenet produced keypoints 21 fps.
- **SqueezeNet:** Video with 114 frames took 6.2022 sec when Posenet produced keypoints 21 fps.
- **AlexNet:** Video with 55 frames took 4.1868 sec when Posenet produced keypoints 21 fps.
- **VGG:** Video with 55 frames took 4.3259 sec when Posenet produced keypoints 21 fps.
- **Inception:** Video with 55 frames took 3.4857 sec when Posenet produced keypoints 21 fps.
In this chapter, we are going to discuss about the our emotion recognition method. As we have mentioned before, we have used deep learning techniques for emotion recognition.

### 8.1 Dataset

As all machine learning problems we need data to train our models. While working and doing research about the topic we discovered that there is not enough human data to train a model from zero. So we have decided to use transfer learning. We have collected data from 4 different datasets: CK+[38], FERGDB[40], IMPAFACE3D[41], JAFFE[42].

We decided to combine the human image and animated image for the training because of lack of dataset of human images. We had total 9107 animated images and 1758 human images in total combining all 7 classes. We partitioned the data in training, test, and validation sets. The distribution of the animated data is shown in Figure:8.1. All seven classes in the animated images have same number of images because in the actual dataset there are lot more samples available, but we did not wanted too many animated images while having very less images for human dataset. In Figure:8.2 we shown the split of the data into train, test, and validation set. As you can see there is very limited number of human images available for training. We have evaluated our model separately for the human and animated images. All the human and animated data were pre-processed and converted to the gray scale images of size 350x350 pixels.

### 8.2 Training

We have used transfer learning technique with pre-trained VGG16 model. In Figure:8.3, we can see the training accuracy of created MLP model(top layers of the model), which takes the bottleneck features as input and return probability of 7 classes. As we can see, the training accuracy increased slowly with time. We have trained our model for 20 epochs. The training accuracy saturates after very few epochs. If we analyze the cross validation accuracy for human and animated images, we will see that accuracy for the animated images is way more than that for human images because of the large number of available samples for animated images, but less number of subjects. The
Experimental results of emotion recognition

**Figure 8.1:** Animated image dataset

**Figure 8.2:** Human image dataset

Figure 8.4 shows the value of loss with respect to each epoch. In our model, we have used very small to no dropout rate. Initially, we began with ‘0.5’ dropout rate in between first four layers. However, after 10 epochs we observed that our training and CV loss was not reducing. We gradually decreased the dropout rate and we observed that both training loss and CV decreased and
8.3 Test Result

As mentioned above we have kept test data of both animated and human images separate so that we can test our model on human and animated images separately. After using the model for human test data we have got an accuracy of 85.06%.

8.3.1 Confusion Matrix

To analyze our model carefully we have created two confusion matrices for human test set (Figure: 8.5) and animated test set (Figure: 8.6). First thing we can notice from the confusion matrix for human data is that the number of samples in each class is quite different, like "happy", "surprise", and "sad" have large number of sample as compared to "fear" or "angry". This can tell us that our model will predict "happy" or "surprise" better than "fear". On the other hand we can see for animated dataset every class has same number of
Figure 8.4: training loss with respect to epoch

Figure 8.5: Confusion matrix for human test set

samples. Also from confusion matrix we can see the number of false positive, false negative for each classes.
Figure 8.6: Confusion matrix for Animated test set

8.3.2 Precision Matrix–

Figure 8.7: Precision matrix for human test set

For our model we have created 2 different precision matrices reported in Figure:8.7 and Figure:8.8, respectively for human and animated test sets.
Firstly, if we look at the precision matrix of the human test set we can see that precision is really good for “disgust” and “fear”, while “sad” has lowest precision.

If we now take a look at the precision matrix for the animated images, we will see all seven classes have perfect precision. So for all the animated classes the model perform excellent.

8.3.3 Recall matrix

We have generated two recall matrices respectively reported in Figure:8.9 and Figure:8.10, for human and animated test sets. If we take a look at the recall matrix for the animated test set, we will notice that all the classes have high recall. Now if we move on the recall matrix for the human test set, first thing we notice is that the class ”surprise” has highest recall along with ”happy” and ”sad”. On the other hand ”fear” has lowest recall of 0.61.

The class ”fear” has low recall and high precision for human test set, which means we miss lot of positive examples, but predicted positive ones are indeed positive.

High recall, low precision: This means that most of the positive examples are correctly recognized (low false negatives) but there are a lot of false positives.

Low recall, high precision: This shows that we miss a lot of positive examples
(high false negatives) but those we predict as positive are indeed positive (low false positives)

The class "sad" has high recall and low precision, which means most of the positive examples are correctly recognized but there are a lot of false positive. Model perform well on the animated images proved by the high recall and precision for all the classes in the test set.

![Recall Matrix](image)

**Figure 8.9:** Recall matrix for human test set

We have analyzed the emotion detection model in details with all the matrix and results and we can see that lack of data for the training affects our model quite a lot.
Figure 8.10: Recall matrix for Animated test set
DISCUSSION

In this chapter, we are going to take a deep dive into our experiments and comparisons we have done. We are going to discuss the insights we have found from the results. As you have seen our project is divided into 2 parts: activity recognition and emotion recognition. We have conducted a lot of experiments and compared different techniques, mostly for the activity recognition part.

Let’s start with experiments on the activity recognition part of our project. Some of the best data encoding techniques we have found by comparisons and experiments are stretching, use score for 3rd channel of RGB image, and 17 key points. The stretching technique worked better than a repeating technique for encoding skeleton data into images. The use of all 17 key points results in better accuracy than just using 14 key points for the activity recognition purpose. The results of the experiments show that models achieved better accuracy with stretching and 17 key points. From this observation, we can say that 17 key points provide more information to a model, useful to better prediction.

We have used both scores, and the average of key points for the 3rd channel of the image generated from the video sequence. Both of the techniques performed well, but the score of the key points provided much more information and helped to achieve a little better accuracy. The experiment results of table 7.4 show using score is a better encoding technique.

We have used cross-subject and cross-view evaluation methods mentioned by the NTURGB+D dataset for the evaluation of our models. We have used some of the most popular image classification algorithms for our experiments. We have decided to use a wide range of models in terms of the number of layers present and techniques used. We have shown that all the models performed considerably better in case of deep retraining than shallow retraining and training from scratch. Among all the models we have used, the best accuracy was achieved by ResNet. In our application, where efficiency is important, Squeezenet can be also used even though it is less accurate, but it is more efficient than ResNet. Table 7.7 shows the performance of the models in prediction time where AlexNet achieved the lowest prediction time.

We have also shown ResNet and DenseNet models in detail. Confusion matrix, precision, and recall of both models gave us a better understanding of weaknesses and strengths.

In the end, we have shown the real-time permanence of activity recognition part of the project while using different models as the predictor. We can
conclude from results that a large part of the delay was caused by sequence to image generation module. To improve the performance of the whole activity recognition part we have to improve sequence to image generation module.

Emotion detection is a very challenging task and it was more challenging because of the scarce availability of data. This is why we decided to use also animated data along with human data for our project. The number of human data was much lower as compared to animated images. But we evaluated our model separately on human and animated data to make sure our model also performs well on human images. The accuracy of the model can be increased by providing more human data for training. Emotion recognition part of our system is very simple, it takes an image of faces and predicts emotion. The real-time performance of the emotion recognition part only depends on the prediction time of the VGG model.

The performance of the whole system depends on both parts of the system. Activity recognition is the most important part of the system as it takes the most computation power and time compared to the emotion recognition part.
CONCLUSIONS AND FUTURE WORKS

This thesis addresses the problem of human activity and emotion recognition. An effective, yet simple, system is proposed to recognize activity and emotion from videos in real-time. Such a system allowed us to use powerful deep learning-based image classifiers to recognize activity and emotions.

This work demonstrates that the data extracted from traditional RGB videos contain sufficient information to train image classifiers to recognize human activities and emotions. We obtain better performance w.r.t. the state of the art algorithms. These results show any video can be used for activity and emotion recognition without the use of depth or motion sensors.

This work demonstrates that the data extracted from traditional RGB videos contain sufficient information to train image classifiers to recognize human activities and emotions. We obtain better performance w.r.t. the state of the art algorithms. These results show any video can be used for activity and emotion recognition without the use of depth or motion sensors.

PoseNet is a very recent technology which shows improved accuracy, performance, and compatibility compared to other well-known pose estimation algorithms like OpenPose.

Based on an intensive study of the related work, we chose to use image classifiers based on deep neural networks for activity and emotion detection. In the case of activity recognition, we converted the motion sequences into RGB images so that we can use existing convolutional neural networks designed for image classification. Cropped images of faces were used for recognizing emotions with convolutional neural networks. We have tasted 6 different image classifier models: SqueezeNet, AlexNet, DenseNet, Resnet, Inception, and VGG.

Different data representations have also been tested for activity recognition. Among all the data representations, encoding of X, Y coordinates and the confidence score of 17 joints into RGB channels gave the highest accuracy. Also, different image classifiers have been tested for activity recognition. While in case of emotion detection we have used animated and human images to train image classification model.

The highest accuracy reached during this study is 88.19% for activity recognition and 85.06% for emotion recognition, which is better than previous state-of-the-art approaches based on RGB videos.

The experimental results have shown the efficiency of the system. In future works, we aim to improve our system’s performance. We can try new data encoding techniques and image generation methods for activity recognition.
Also, we can do more research and experiments to find a more efficient neural network for our purpose. It would be interesting to try other techniques like recurrent neural networks for activity and emotion recognition. The performance of the system can be improved with a lot more experiments and efficient programming techniques in future.


