# PATTERN CLASSIFICATION OF HUMAN INTERACTIONS FROM VIDEOS

MUHSIN ABDUL MOHAMMED

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2018

# PATTERN CLASSIFICATION OF HUMAN INTERACTIONS FROM VIDEOS

**MUHSIN ABDUL MOHAMMED**

**THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE MASTERS OF MECHATRONICS ENGINEERING**

**FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR**

**2018**

**UNIVERSITY OF MALAYA**

**ORIGINAL LITERARY WORK DECLARATION**

Name of Candidate: MUHSIN ABDUL MOHAMMED

Registration/Matric No: KQF160012

Name of Degree: MASTER OF MECHATRONICS ENGINEERING

Title of Project Paper/Research Report/Dissertation/Thesis (“this Work”) :Pattern classification of human interactions from videos

Field of Study: Artificial Intelligence / Computer Vision

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya (“UM”), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate’s Signature Date:

Subscribed and solemnly declared before,

Witness’s Signature Date:

Name:

Designation:

## ABSTRACT

The objective of this research project is to build a machine learning model to classify human interactions from a stream of video. Being able to classify human interaction from videos is essential in the development of robotic assistance systems, video annotation, surveillance systems and many more applications. It is necessary that the algorithm performing this task needs to be robust and only relies on monocular vision systems.

In order to build a classifier capable of achieving this task, the machine learning model needs to be able to learn spatial and temporal patterns from the videos. A cascaded architecture of Convolutional Neural Networks and Recurrent Neural Networks have been created to achieve this task in this research. There have been investigations made to identify the best spatial and temporal architectures that would give the optimal result.

# ABSTRAK

Objektif dalam projek penyelidikan ini adalah untuk membina sebuah model pembelajaran mesin bagi mengelaskan interaksi manusia daripada aliran video. Berkebolehan untuk mengklasifikasikan interaksi manusia daripada video adalah penting dalam pembangunan sistem bantuan robotik, anotasi video, sistem pengawasan dan banyak lagi aplikasi yang lain. Ia adalah penting bahawa algorithma yang melaksanakan tugasan ini perlu kukuh dan hanya bergantung kepada sistem penglihatan monokular.

Bagi membina sebuah pengelas yang dapat mencapai tugasan ini, model pembelajaran mesin tersebut hendaklah mampu untuk mempelajari corak-corak spatial dan temporal daripada video. Sebuah senibina yang mengabungkan *Convolutional Neural Networks* dan *Recurrent Neural Networks* telah dibina untuk mencapai matlamat dalam kajian ini. Penyiasatan telah dibuat bagi mengenalpasti senibina spatial dan temporal terbaik yang akan memberikan keputusan yang optimum.

## ACKNOWLEDGEMENTS

Firstly I am grateful to God, the Merciful and the Almighty for all his blessings.

I would also like to express my gratitude to my supervisor Ir.Dr.Chuah Joon Huang, Head of VIP Research Group at Department of Electrical Engineering, Faculty of Engineering, University of Malaya, who has provided me with valuable insights and guidance for the fulfillment of this research.

Lastly, I want to thank my Parents and Sisters for their love, support and motivation, without which I wouldn't be where I am.

Muhsin Abdul Mohammed

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

## LIST OF SYMBOLS AND ABBREVIATIONS

| | | |
|---|---|---|
| ANN | : | Artificial Neural Networks |
| CNN | : | Convolutional Neural Networks |
| FCN | : | Fully Convolutional Network |
| fps | : | Frames per second |
| LSTM | : | Long Short Term Memory |
| ReLU | : | Rectified Linear Units |
| RNN | : | Recurrent Neural Network |
| SSD | : | Single Shot Multi-box Detector |

# CHAPTER 1: INTRODUCTION

This chapter discusses the motivation and background of this research work, followed by a statement on what would constitute as the objective or goal of this undertaking. A brief description about the structure of the thesis is also included below.

## 1.1 Background

Human activity recognition is the task of identifying what a person is doing. In case of more than one person, the task is often described as human interaction detection. As human beings, we possess a level of awareness that would let us effortlessly identify what someone is doing. However, for teaching a machine to do the same is challenging to say the least. In machine intelligence and robotics this task falls into the class of sensing problems. There has been several research approaches attempted at solving this problem (Aggarwal and Cai, 1999). For any system designed to do this task requires some form of sensor to capture the raw information. Most widely used sensors have been monocular vision, thermal imaging, depth vision systems, accelerometers, gyroscope and other inertial sensors in general. The ability to perform the task of human activity detection is crucial in the future of several applications like smart homes, robotics, gaming, virtual reality and so on. The recent developments in computation and availability of massive datasets has made computer vision based approaches along with machine learning algorithms the best route to achieve the task of human activity recognition.

## 1.2 Objective of the project

Human activity identification is a complex task due to several facts. The variation of scenarios from which the classification has to be made can be vast. The permutation of scenes will take into account, number of human beings and other objects present in the scene, number of participants in an activity, wide range of background sceneries, different human activity or interactions required to be classified, and the possibility of having more than one activity present in any given frame, especially when the classification is done on a public space. There is also the question of representing the spatio-temporal relationship in the classification model.

Due to these challenges the scope of research is narrowed down and the following assumptions are duly made:

1. At any given frame there will only be 2 people present in a frame. Therefore the task is a human to human interaction detection problem, limited to 2 agents.
2. The activities or interactions expected is limited to 5 classes. They are hand-shaking, hugging, kicking, punching, and pushing.

With the above set scope, a lot of complexities can be overcome. It is also worth noting that implementation of such a system can be scaled to involve more participants and for much broader class of activities. The set scope also makes it easier to collect and label data.

With the scope is fixed, the objectives of the research can be defined as follows:

1. To obtain a reliable labeled dataset of human to human interaction.
2. To identify acceptable performance criteria for a human interaction detection system.
3. To design a machine learning model capable of learning spatio-temporal relations.
4. To train and test the model with the labeled dataset and obtain the acceptable performance.

## 1.3 Structure of the thesis

The thesis is divided into 5 chapters including this one. Chapter 2 goes into the literature review in the area of human activity detection and also briefly discusses the background of algorithms used in this research like CNNs, LSTM and so on. Chapter 3 discusses the methodology of the research and implementation of the final model and the path that lead to it. Chapter 4 briefly presents the important results obtained from machine learning model. Chapter 5 concludes the research and presents some future directions to explore in this area.

# CHAPTER 2: LITERATURE REVIEW

The usage of advanced computer vision and deep learning algorithms to recognize and classify human movement and interaction is an active area of research.

## 2.1 Complexities present in the task

The major challenges in designing a machine learning model for the given scope is in the recognition of intra-class variations (Gao, 2011). The same action is performed differently by different people. This can be observed in the Figure 2.1. The activity performed in the image is a kick and it is performed by three different human beings. Although the task performed is the same, the pose is different in all three cases. The level of leg raised is also different. The speed at which the action is performed (which is only observable in a sequence) can also be very different from person to person. Certain techniques like keyframe extractions can be used to tackle the variations. It takes little effort for a human being to recognize these activities, but to represent all possible variation of any given task in a statistical model is clearly challenging.

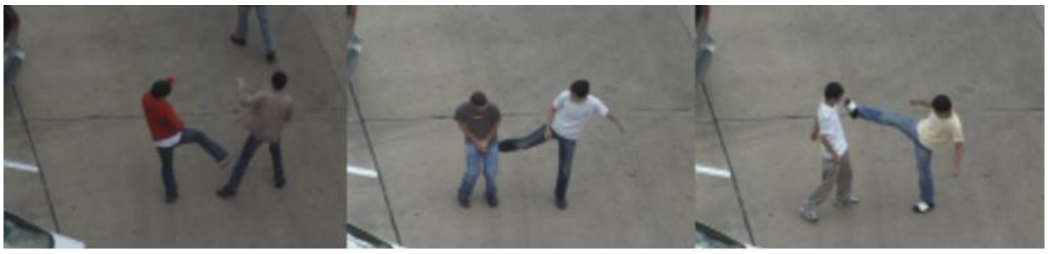


**Figure 2.1 : The action of kicking performed by three different individuals.**

### 2.1.1 Real world conditions

The other challenges are related to the real world. There could be objects used by the agents in a scene (handbags, umbrellas) or present in the scene (trees, tables, vehicles). This can occlude the subjects involved or cause clutter in background. The lighting and weather conditions can vary drastically too. The camera used for capturing the data can vary. Since the model being considered for the task accounts for spatio-temporal relations motion jitters, noise and frame rates can also have influence on the model. The model will also perform poorly on images captured on a moving camera since the model is trained on the data from a stationary camera. It is essential to ensure that the model performs accurately in all the above mentioned adversarial situations.

## 2.2 Conventional approaches in human activity detection

Three stages involved in most human activity detection systems are object segmentation, feature extraction and representation, and activity or interaction detection by classification algorithms (Ke et. al., 2013). With the advent of massive labeled datasets and computing power there are several end-to-end machine learning models which relies mostly on artificial neural networks to do all of the aforementioned tasks with minimal feature engineering required. However, these approaches requires huge datasets and computational power for training. They are also in several cases hard to interpret, and modification of particular behaviors are hard to perform. For critical tasks where human safety is important it is worth exploring the feature engineering based approaches.

### 2.2.1 **Object Segmentation**

Object segmentation as the name suggests segments the object of interest that is human beings from the background or noise. There are several parameters that needed to be considered while designing an object segmentation system, mobility of the camera is one of them. The camera can be moving or static. In a statically positioned camera, the segmentation becomes easier as the background does not move. A background model can be captured ahead of classification and used to subtract it from the foreground. This method is simple to perform and computationally less expensive. However, the resulting image is not guaranteed to be exclusively the subject of interest, as there could be other moving objects in the scene. Therefore the obtained object segmentation mask could have background objects.

Object segmentation can be further improved by incorporating the knowledge about the scene into the model. One of the methods that could be used is to model the color distribution of the background pixels as a Gaussian covariance matrix (Wren, 1995). With this information the foreground information can be grouped together by using this covariance matrix associated to that point. There can still be unwanted information like shadows and glares present. Some statistical assumptions can be made to overcome this. The chronological changes in the background can be modeled and represented as an image vector, which can be then used to subtract that from the foreground (Seki et. al., 2000).

There are several other methods used in static camera object segmentation including Gaussian Mixture Model (GMM), which all though computationally expensive can be used to describe very complex background scenes (Yoon et. al., 2003). In case of moving camera like pan-tilt surveillance cameras, drones, and other mobile robots most of the above mentioned methods would not work. A camera motion decomposition needs to be performed, which requires sensors measuring the motion of the camera and lens arrangements. This research does not use data from moving camera, but one of the advantage of using a CNN is that the area of interest can be segmented even without a complex motion model, given enough training data.

### 2.2.2 Feature extraction and representation

Once the object of interest has been segmented from the background, now the features need to be extracted and represented. The representation is crucial as it needs to account for the sequential nature of the data. Since videos are captured typically between 25-30fps, many frames might not contain relevant information. Space-time volumes (STV) captures the features from individual frames (ie., spatial information) and stack it in sequence (temporal information). In addition to STV, Discrete Fourier Transform (DFT) could also be used to represent the data. DFTs are represented as a variation of image intensities spatially (Kumari, & Mitra, 2011).

#### 2.2.2.1 Keyframe extraction

Key frame capturing has been a popular choice for temporal data classifications. The popularity of key frame extraction can be mainly attributed to its low computational cost involved in vision based analysis. To reduce the number of images to be processed by the model a key frame extraction algorithm can be used. Such an algorithm will calculate the dissimilarity between consecutive image frames (Chatzigiorgaki et. al., 2009). This type of algorithms assess the change of information, like texture, color, and other geometrical aspects. If a given frame has high dissimilarity then it is considered to be a key frame. Unsupervised clustering methods have also been used to analyze and extract keyframes (Li, & Hua, 2011).

#### 2.2.2.2 Temporal templating

Representation of human activity using temporal templates is another active area of research. In this case an action is considered as motion over time. A naive implementation of temporal templates would involve the assumption that the background is static and only the subject of interest is in motion. There are several techniques implemented to overcome this (Davis, & Bobick, 1997). Temporal templating is done by generating a binary image commonly referred to as a motion-energy image (MEI). This area of research is motivated by the observation that human beings are capable of recognizing an action from a low resolution representation with little to no information about the 3 dimensional structure of the scene.

A MEI binary image is shown in Figure 2.2. The top row shows the action in a sequence and the corresponding bottom row is a cumulative binary image of all the pixels that has changed. The final image on the bottom row is considered to have captured a robust spatial motion-distribution signal of the activity performed, in this case a person sitting down. This image has not only captured the essence of this particular action but also the direction or angle from which the activity has been observed, ie.. the resulting MEI binary image can be used to obtain both the action and the pose of the human beings. It is also worth noting that the frames displayed in the top row are not necessarily a continuous sequence captured by a camera (typical between 30 to 60 fps), it could have been passed through a keyframe extractor as discussed in the above section for computational efficiency (Gao, 2011).

Exemplar based approaches are also found in human-activity recognition. In these approaches a heuristic will be used to match a given frame with an example from the training set (Gao, 2011).

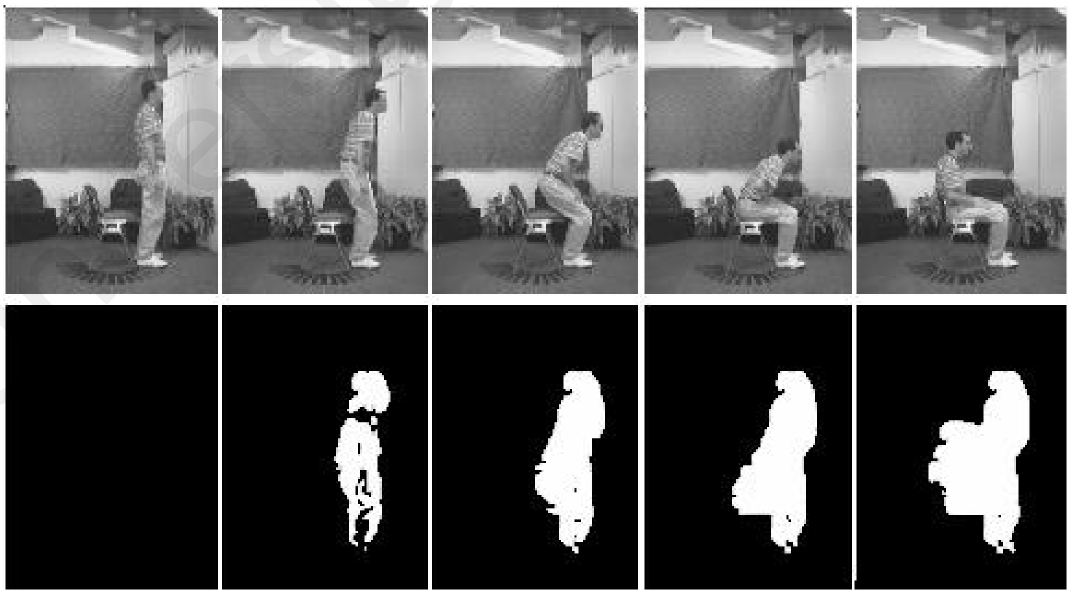
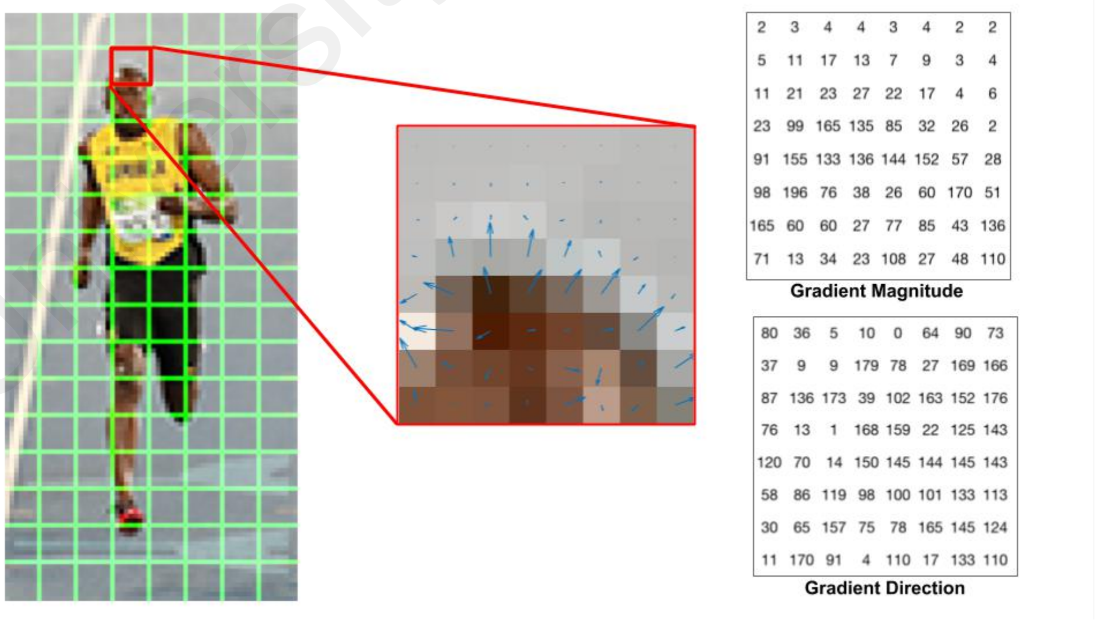


**Figure 2.2: MEI (motion energy image) of a person sitting down. 1st row is the captured images, 2nd row is the corresponding MEI**

#### 2.2.2.3 **HOG Feature extraction**

Histogram of Oriented Gradient (HOG) descriptors is a popular and widely used method for feature representation in object detection tasks. HOG descriptors are implemented by first dividing the image into smaller units or ‘cells’. A histogram of gradient detection is compiled for each cell. This is done by applying a filtering kernel like sobel or edge detector. On this the magnitude and direction of the gradient can be calculated.

The resulting matrix (as shown in Figure 2.3) for each such cell is then binned to form a histogram (Dalal & Triggs, 2005). A drawback with this method is in the choice of the cell size. Since the local descriptors are obtained on a fixed scale the object of interests cannot vary a lot in size.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | 3 | 4 | 4 | 3 | 4 | 2 | 2 |
| 5 | 11 | 17 | 13 | 7 | 9 | 3 | 4 |
| 11 | 21 | 23 | 27 | 22 | 17 | 4 | 6 |
| 23 | 99 | 165 | 135 | 85 | 32 | 26 | 2 |
| 91 | 155 | 133 | 136 | 144 | 152 | 57 | 28 |
| 98 | 196 | 76 | 38 | 26 | 60 | 170 | 51 |
| 165 | 60 | 60 | 27 | 77 | 85 | 43 | 136 |
| 71 | 13 | 34 | 23 | 108 | 27 | 48 | 110 |

Gradient Magnitude

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 80 | 36 | 5 | 10 | 0 | 64 | 90 | 73 |
| 37 | 9 | 9 | 179 | 78 | 27 | 169 | 166 |
| 87 | 136 | 173 | 39 | 102 | 163 | 152 | 176 |
| 76 | 13 | 1 | 168 | 159 | 22 | 125 | 143 |
| 120 | 70 | 14 | 150 | 145 | 144 | 145 | 143 |
| 58 | 86 | 119 | 98 | 100 | 101 | 133 | 113 |
| 30 | 65 | 157 | 75 | 78 | 165 | 145 | 124 |
| 11 | 170 | 91 | 4 | 110 | 17 | 133 | 110 |

Gradient Direction

**Figure 2.3: HOG descriptor of an image patch. Corresponding matrices of magnitude and direction to the right**

### 2.2.3 Classification algorithms

The data from video stream is first passed through a foreground extractor and then a feature extraction algorithm is used to represent the data. The feature representation now possesses both the spatial and temporal elements of the video data. At this stage a classification algorithm is required to identify the action or interaction present in the scene. Thus, the task at hand is to look at the data *(X)* and classify it into a label *(y)*.

For this task model-based systems are effective over algorithms like Dynamic Time warping (DTW). Model based algorithms can be classified into discriminative and generative models. Generative models classify a given data by attempting to simulate the process by which the data sequence can be generated. Hidden Markov Models (HMM) and Dynamic Bayesian Networks (DBN) are examples for generative models. These models learn the joint probability distribution P(X, y) from the labeled dataset. Whereas the discriminative models learn the conditional probability or posterior distribution p(X | y). Artificial Neural Networks (ANN), and Support Vector Machines (SVM) fall in the category of discriminative algorithms. There are also other algorithms like K-nearest neighbors (KNN) and Binary tree algorithms used for classification (Ke, et. al., 2013).

## 2.3 End to End classification

The end to end or behavior cloning approaches are mostly performed by supervised machine learning (ML) algorithms. These require labeled datasets. This research makes use of existing ML algorithms to achieve the task of human activity recognition. The building blocks of these classifiers are briefly discussed in the following sections.

### 2.3.1 Artificial Neural Networks

In order to properly acknowledge the strengths of Artificial Neural Networks (ANN) it is important to recognize what the traditional or alternate algorithms offer. Statistical classifiers that use Bayesian decision theory is required to have an underlying probability model. The assumed model is then used to calculate the posterior probability for the classification. Evidently the performance of these classifiers are only as good as the underlying assumptions. The fidelity of these assumptions depend on the algorithm developers understanding of the dataset. This is where the advantages of Neural Networks comes is. An ANN starts out as a placeholder model containing random parameters waiting to be modified into a classifier. This modification is fully data driven. It is not required to provide any underlying statistical distributions or functions explicitly. ANNs are universal function approximators, which means they can approximate any function with some accuracy. ANNs are also nonlinear algorithms, therefore they are capable of modeling complex real world relationships (Zhang, 2000).

A simple ANN for classifier is shown in the Figure 2.4. This is a fully connected variant or a Multilayer perceptron (MLP). The cell in each layer called a neuron, and the connecting lines are the weights. Weights are the parameters that are trained to provide the desired output. The first layer will be have neurons equal to the number of inputs and the last layer will have number of neurons corresponding to the number of classes in the output, it could also be regression units. The middle layers or hidden layers can grow length-wise or depth-wise, these are considered to be hyperparameters, which means it is left to the developer to decide. Generally the deeper the hidden layer, more complex functions the model can represent. It would also mean a bigger dataset would be required to train the model.

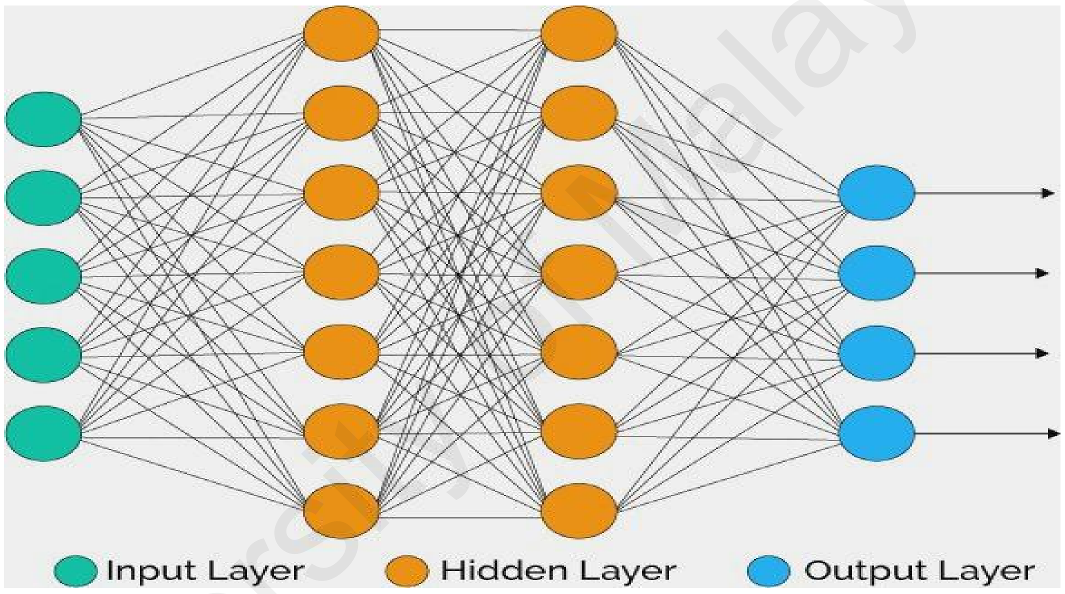


**Figure 2.4: Multilayer perceptron with two hidden layers**

#### 2.3.1.1 Feed forward network

For the given input features X ∈ Rd, and the output classes y ∈ 1,2,3,...m, the fully connected network can be represented as ƒ : X . The input data is passed on to the first layer and the resulting product with the weights W produces a linear combination. This linear combination is then added with a bias b, this is allows a function to be shifted from its position. The result is then send to an activation function for a non-linear mapping. This output represents the complete operation of one layer, and commonly called as the activation of a layer h. Then the activation h is passed on to the next layer and all the previous steps are performed for all the layers till the output. The activation for the last layer for a classifier is the softmax of the output layer neurons.

h1(X) = activation(W1 X + b1 )

h2(h1(X)) = activation(W2 (h1(X)) + b2 )

Output = h3(h2(X)) = activation(W3 (h2(X)) + b3 )

#### 2.3.1.2 Activation functions

The activation function is a nonlinear function. For this paper a ReLU or linear rectified unit is used. For an input x it outputs x if x is positive and 0 if x is negative. It is simple to program and proved to be efficient in optimization and computationally less expensive (Xu, 2015). ReLU function can be represented as in equation below.

**f(x) = max(0, x)**

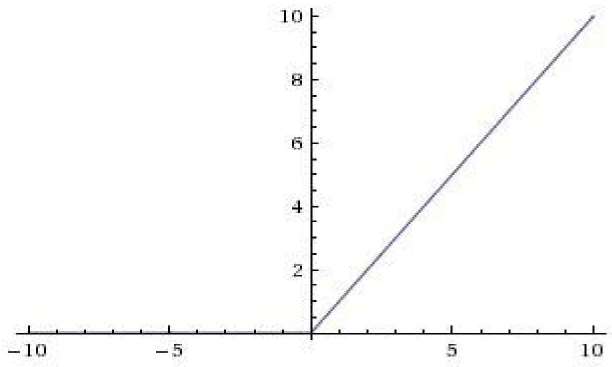


**Figure 2.5: Graph of ReLU activation function**

One downside of ReLU is that it outputs 0 for any negative value regardless of the magnitude. An alternative for this is a leaky ReLU, or Noisy ReLU. A leaky ReLU allows a small output proportional to the input when the input is negative, rather than outputting 0. Leaky ReLU can be represented as in equation below.

$$f(x) = x \quad \text{if } x > 0$$
$$f(x) = 0.01x \quad \text{if } x <= 0$$

One other type of activation function is sigmoid, it is a special case of logistic function where the input is squeezed between 0 and 1. Although it is not used as an activation function in this research it is used as a gate function. The sigmoid operation is performed using the following equation.

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

#### 2.3.1.3 Backpropagation

In any machine learning models there are three essential steps. The representation, evaluation and optimization of the model. As discussed in the previous section the ANN is started out as a randomly initialized tensors. The input data is feed-forwarded through the network to get an output. The output is then compared with the label, this is the evaluation step and there are several ways to do it, one would be to measure the root mean square error (RMSE). Once the error is calculated it is time to optimize the network, and for neural networks the most popular and effective choice of optimization is back propagation (Hecht-Nielsen, 1992).

The back propagation algorithm calculates the gradient of the loss function corresponding to each parameter using the chain rule. For classification models the cross entropy loss can be used, as show in equation below.

$$\text{Loss}(p, y) = -(y \log(p) + (1 - y) \log(1-p))$$

The gradient descent for each layer can be calculated as shown in the following equation. Where η is the learning rate, a relative small scaling factor to overcome overshooting. W is the weight parameter corresponding to that layer.

$$Wt+1 = Wt + \eta \frac{\partial Loss}{\partial Wt}$$

### 2.3.2 Overfitting

One of the challenges with an ANN is the issue of overfitting. Whereby the model extracts the residual variations or noise from the dataset, which would be irrelevant to the actual classification. This can be detected by separating the dataset into training and test sets. If the model performs well in training sets while underperforming significantly in the test set it can be due to overfitting. It is worth noting that if the model is underperforming in both the training and test sets, then it is an underfitting problem, and this can be due to lack of data. Performance is a relative metric and varies from problem to problem.

One way to overcome underfitting is to use a regularizer like L1 or L2 norm for all parameters. One other way that has been used effectively is a Dropout layer. Dropout works by randomly dropping connections of the Neural Network while training. This prevents the network from co-adapting excessively (Srivastava, et.al., 2014). The Figure 2.6 shows dropout in action.

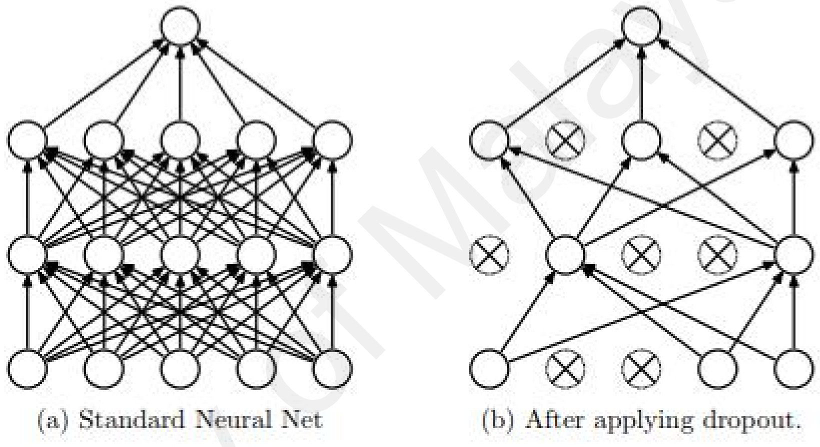


**Figure 2.6: Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped. (Srivastava et.al., 2014)**

### 2.3.3 Convolutional neural networks

The effectiveness of ANNs in classification tasks has been phenomenal, thanks to both the availability of large datasets, and high performance parallel computing. Although ANNs offer a certain degree of data agnostic performance, with modified architectures designed for specific forms of data the performance can be even superior. Convolutional Neural Networks (CNNs) are particularly developed to take advantage of the spatial representation of information in images. CNNs have been in use since the 1980s. In its earlier stage the performance has not been very promising. The discovery that the detection of light in animal receptive fields are due to the cells in visual cortex has been an inspiration for the CNN research (Hubel, & Wiesel, 1959). The seminal research of CNNs that led to an application was in the 1990s by LeCun, et. al, the particular architecture was named LeNet-5 and was used for handwriting recognition. In the mid-2000s deeper CNN (DCNN) like AlexNet inspired by LeNet-5 has shown significant improvements in classification process compared to traditional methods t (Krizhevsky et al., 2012).

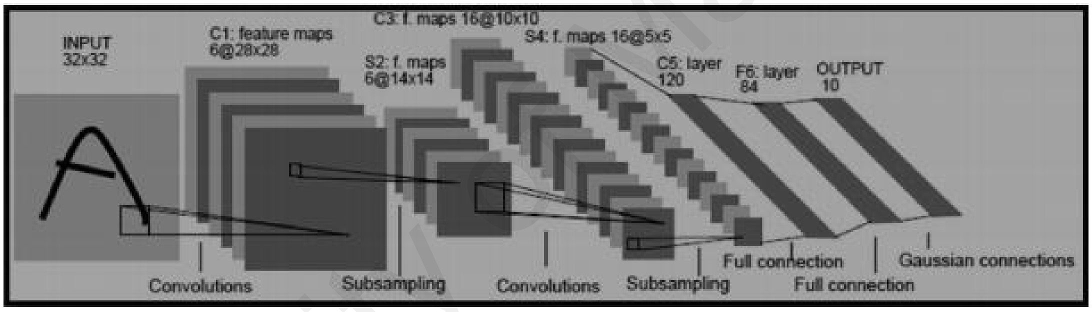


**Figure 2.7: Architecture of LeNet-5 (LeCun et al., 1998)**

In a regular neural network each input neuron is considered as individual contributor of information and therefore the spatial representation of the data is lost. Given a large dataset and complex deep neural networks, this relation can be learned by the network, but the lack of computation resources and labeled data makes it a difficult task. With CNNs the main operations are convolutions, which are done between the input of one layer and a kernel. This kernel is the parameter being learned during training. There can been several kernels deployed for each layer, and each kernel would learn a relevant feature useful for classification, like vertical edges, colors, or even complex patterns. Each kernel is the size of a small patch of 2d space and the same kernel is used as the operand for convolution throughout the input, and it is done in a sliding manner as illustrated in the Figure 2.8. This offers the advantages of sparsity in connection and thus decreased chance of overfitting, parameter sharing thus reducing the required number of parameters to be learned, translational invariance so the object can be in any part of the image, scale invariance so the object can be of varying size, and some amount of translational invariance so the object can be rotated up to some limit.

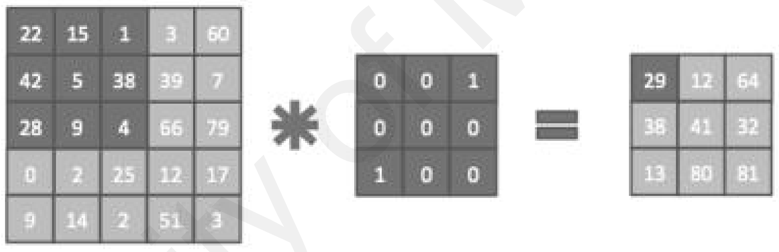


**Figure 2.8: An example of a 2-d convolution operation**

The feature value at location (i,j) in the kth feature map of lth layer, zli,j,k is calculated as:

$$Z^{l}_{i,j,k} = (w^{l}_{k})^{T} x^{l}_{i,j} + b^{l}_{k}$$

### 2.3.4 Recurrent neural networks

CNNs are used to capture the spatial information about the position and pose of the human beings in a given frame, but vanilla CNNs are incapable of representing any form of temporal relations.

Recurrent Neural Networks (RNNs) are suited for sequence models. RNNs for each time step take into account the current input and the previous one. For a simple RNN there are two parameters required to be learned, the weight at the input neuron (**$W_{xh}$**) which represents the influence of current state to the input and the recurrent neuron (**$W_{hh}$**) which represents the influence of the previous state to the current state, and (**$W_{hy}$)** the influence of current state to the output. It can be mathematically represented as in equations below. Where ht is the current state, ht-1 is the previous state and yt is the output. All three weight parameters are shared between all the inputs as shown in the Figure 2.9. The training of this type of RNN is done using variants of backpropagation through time algorithm (BPTT) (Werbos, 1990).

$$h_t = activation(\ W_{hh}\, h_{t-1} \ + W_{xh} + x_t\ )$$

$$y_t = (W_{hy}\ h_t)$$

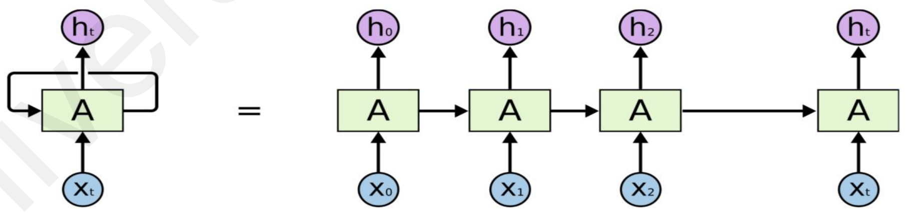


**Figure 2.9 : Recurrent Neural Network**

There are several types of architecture in the RNNs depending on the input and output size. For the application of human activity recognition we consider an architecture where input size and output size are the same (Tx = Ty), as shown in the Figure 2.9.

### 2.3.4.1 **Long short term memory (LSTM)**

RNNs are versatile in its implementation, but one of the drawbacks is that it only captures the short-term memory from the sequence. The BPTT algorithm when training tends to either blow up or vanish (exploding or vanishing gradient happens, when the timesteps are separated widely, the networks parameters drops to extremely small values or exponentially large values). Long Short Term Memory (LSTM) networks are a type of RNN adept of learning long term information. LSTMs were introduced in late 90s (Hochreiter, 1997). There are also other network designs capable of doing this, like Gated Recurrent Units (GRU) introduced by Cho, et al. in 2014. LSTMs can effectively bridge relations with sequences over 1000 steps (Gers, et.al., 1999). The underlying principle is to implement an efficient gradient based algorithm where the flow of error are forced to be constant, therefore overcoming the exploding and vanishing gradients. To achieve this, special units or gates are incorporated into the traditional RNN.

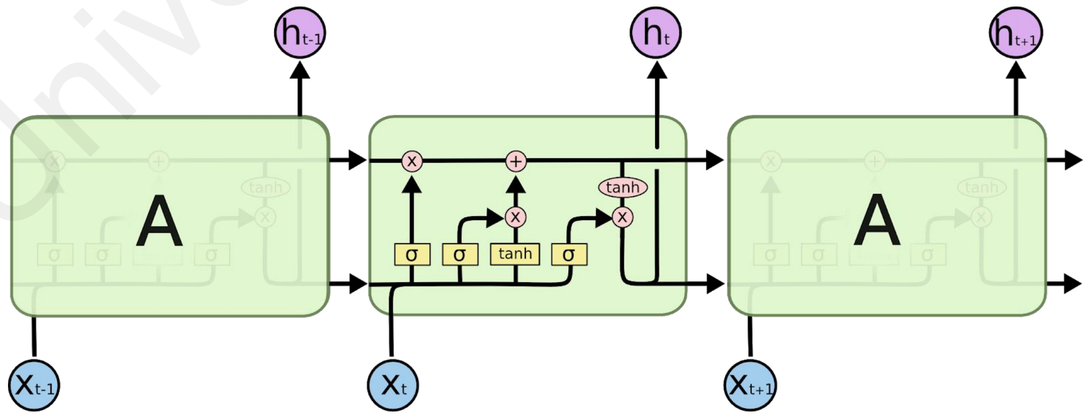


**Figure 2.10 : LSTM unit with internal gates**

There are several variants of LSTM implementations, for this research the original architecture introduced by Schmidhuber, et. al. in 1997 is used. In which for any given temporal state (the total number of states/cells are equivalent to the total number of frames in a sequence of video) there are three state variables that go into the network, the input from the current frame (xt), the output from the previous layer(ht-1) and the cell state(Ct). The cell state (Ct) keeps track of the long term behavior that needs to be preserved for the prediction of current layer and future layers. It can be information like the position of a person’s hand from a previous frame. There are two gates dictating the behavior of the cell state. These can be defined as follows:

1. Forget gate layer:

It decides what information in the cell state should be discarded. It is implemented using a sigmoid gate (as explained in the section of activation functions). The output of a sigmoid is between 0 and 1. 0 would imply complete removal of a state and 1 would mean complete retention of a state. The inputs to it are current input data and previous output. It is represented as the first gate from left in the Figure 2.10.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

2. Input gate layer:

It decides what new information should be added to the cell state. This takes place in two steps, first a sigmoid gate decides what states needed to be updated. The second is a Candidate generator (Ĉt) implemented using a tanh activator. This candidate generator comes up with a list a possible features that should be considered for future predictions.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

The forget gate layer and input gate layer updates the cell state (Ct).

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

The last gate is an output gate which generates the output (ht) for the current state. An internal output is first generated from the current input (xt) and previous output (ht-1). Then the internal output (ot) is filtered using the cell state (Ct).

$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

# CHAPTER 3: METHODOLOGY

## 3.1 Introduction

This chapter presents the research procedure undertaken in developing the machine learning classifier for human activity recognition. The first step in any supervised learning problem is to collect a labeled dataset which is sufficiently large and contains enough variation for practical learning purposes. The following section provides a brief description of the dataset used for this research

## 3.2 Dataset

The labeled dataset used for this project is the “semantic description of human activities” dataset (SDHA 2010) collected by Ryoo, et. al.. The motivation of the researchers on collecting this dataset was to explore the classification of human interactions in a realistic setting. The original dataset contains videos of six human-human interactions as shown in the Figure 3.1. They are ‘pointing’, ‘pushing’, ‘hugging’, ‘kicking’, ‘punching’ and ‘hand-shaking’. For this research only 5 classes are used, class ‘pointing’ is omitted. The dataset uses different participants, backgrounds and clothing conditions for better variety in data. The camera is not fully static producing some videos, this causes a bit of jitter and thus providing a more real life like situation. There are also other agents in some frames like pedestrians walking by or a tree branches causing a bit of occlusion.

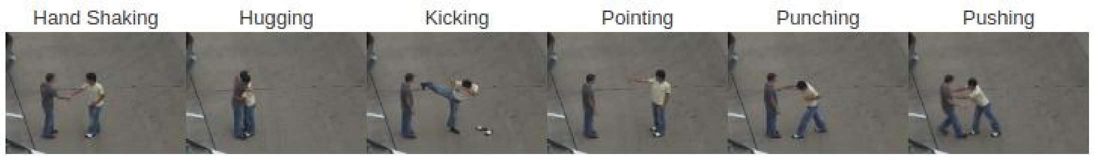


**Figure 3.1 : Sample of interactions present in the SDHA dataset (Ryoo, et.al., 2010).**

### 3.2.1 Data format

The data is collected as video sequences. The labels of interaction was provided separately. Each original video has a resolution of 720p X 480p, with 3 channels of RGB data. The video was recorded at 30fps. There is 20 sequence of videos collected for each interaction class. An interaction starts out with two people approaching from either sides of the frame and performing an action like hand-shaking. The average pixel height of the people in the frames are about 200 pixels (Ryoo, et. al., 2010).

### 3.2.2 Data preprocessing

The original data was collected at a high resolution with 3 color channels. In order to reduce the computational cost and to overcome any bias in the dataset there was many preprocessing steps used. First the video data was converted into image frames and stored in disk for efficiency while trying different models. Ffmpeg suite was used for converting video data into frames. A dense conversion of video to frame would cause 30 frames to be captured from every second of video, but after initial observation it was evident that the continuous frames at this rate did not carry much information. A keyframe extraction algorithm was initially attempted to get an optimal frame rate conversion, but it proved to be less useful as it required optimization to perform well with different backgrounds. After some trial a fixed conversion rate of 15 fps was chosen, as it seems to achieve a balance between reduced computational cost and minimal loss of temporal information. The first row of the image below shows a conversion rate at 15fps (which was used for this research) and the second row shows a conversion rate of 30fps. It is visible from the image that at 30fps very few additional information is present in consecutive frames.

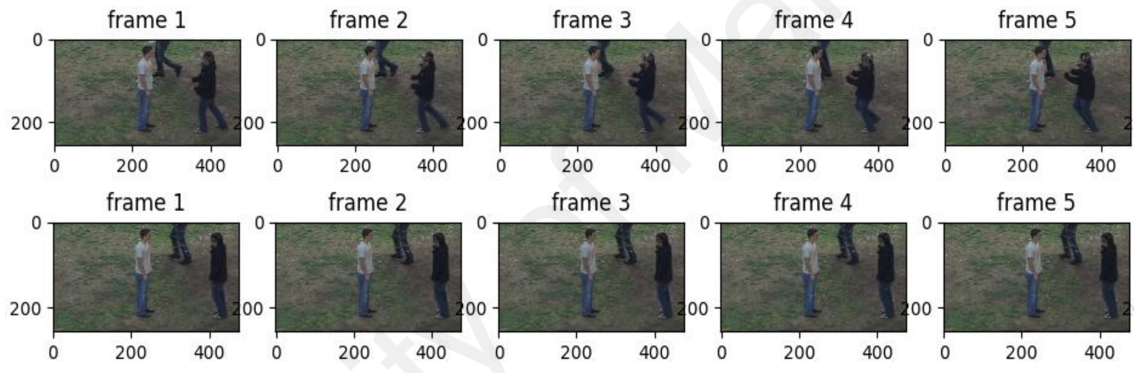


**Figure 3.2 : Top line represents frame conversion of images at 15fps, and bottom line represents frame conversion at 30fps.**

Once the frames were generated it was converted to grayscale as the color information of the dress or background is not very relevant to the classification task, and it also reduces the data to be processed by one third. The next step in preprocessing frames is to apply a Gaussian filter to blur the pixels slightly. This will help remove the noise in the images. The kernel size chosen for the Gaussian blur operation is 5X5. Then the image was reduced in scale for further computational efficiency. The scaling factor was decided by trial and error on a benchmark convolutional network model. Images of resolution below

28X28 started to decrease performance significantly. So 32X32 and 64X64 pixel images were tried out in the model. The last step in preprocessing is to normalize the pixel values between 0 and 1. This is to help the neural network from exploding gradient issues while training and to remove biases. The flow of image processing from video to processed images is represented in the image below.

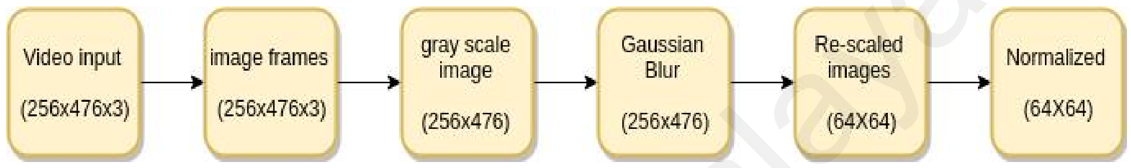


**Figure 3.3 : Sequence of steps in image preprocessing.**

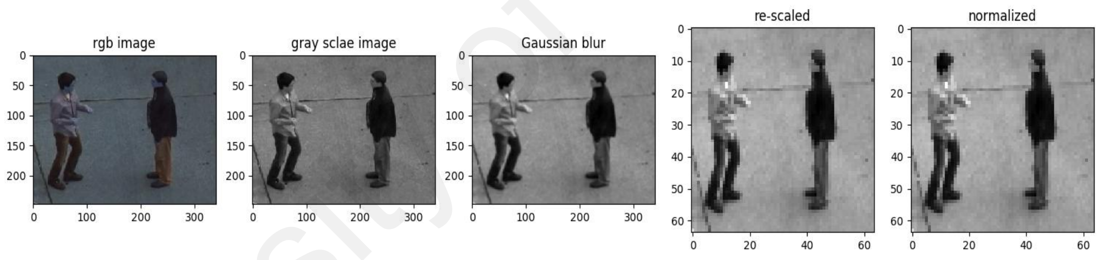


**Figure 3.4 : Sample of images from each step of preprocessing stages.**

### 3.2.3 Statistical distribution of data

For multiclass prediction problem it is important to have a balanced dataset. An imbalanced dataset will lead to a scenario where one class could have more observations and thus could lead to a biased model. As mentioned earlier for each class there is 20 instances of interaction in the dataset. Therefore in terms of sequences the data is balanced. But the time taken to perform each action can be different and therefore the duration of each instance can be varying. A simple frame count was implemented to identify if there was an imbalance in data. As shown in the Figure 3.5 the frames for each class is evenly distributed.

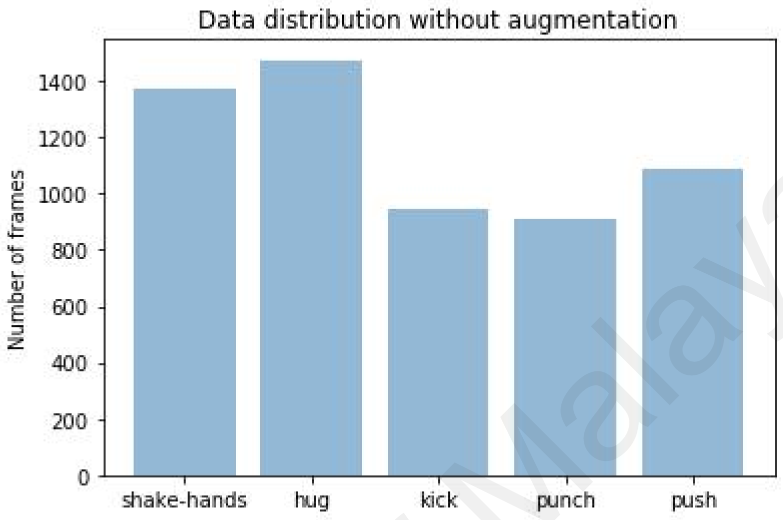


**Figure 3.5 : Data distribution of images belonging to each class.**

### 3.2.4 Data augmentation

For neural network models in order to represent complex relations multiple hidden layers are often implemented. This will cause the parameters to be learned to increase rapidly, which means more data would generally be required to train the model. However, collecting data can be costly and time consuming. One method to overcome this is by augmenting the existing data. This can also help overcome the bias issues (Perez & Wang, 2017), as the network will get acquainted to more complex scenarios, like if the agents are occluded and are not fully visible, or in scenarios where the weather or lighting conditions are different. For this research the following image augmentation techniques were used: horizontal flip, rotation, Gaussian noise, contrast variation, and random geometric occlusions. The factor by which each of the augmentations were used in the training was based on previous research in this area (Thoma, 2017). A sample of each of the aforementioned augmentation process is shown in the Figure 3.6.

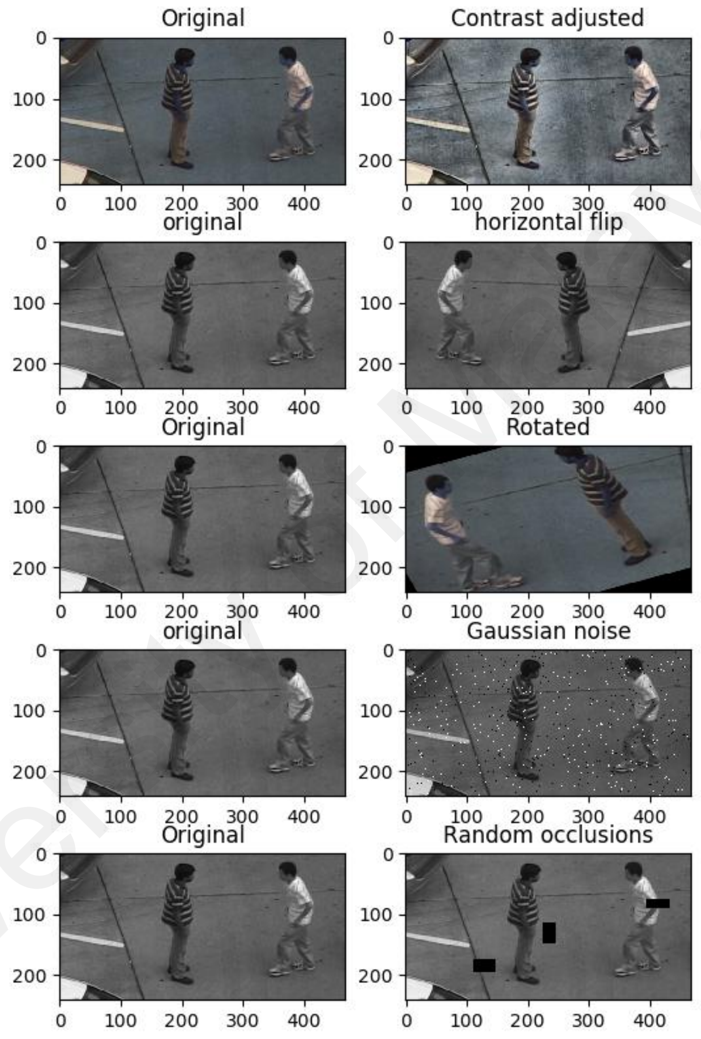


**Figure 3.6 : Sample images undergone data augmentation. From top to bottom contrast adjusted, horizontal flip, rotated, Gaussian noise, and random occlusion.**

## 3.3 Machine learning model

Once the dataset preprocessing and augmentation is done the next stage in the research was to develop the neural network model. The model developed needs to be able to capture both spatial and temporal elements in order to do the classification. Several architectures were tried before arriving at the final model.

### 3.3.1 Performance criterion

Since this is a classification problem with 5 classes, by simply choosing from a random prediction would yield 20% accuracy. If the statistical distribution of the data is considered, as discussed in the section 3.2.3, the most represented data point in the dataset is class 2 (hug), which has over 1470 instances (image frames extracted). The total number of image frames used in the training and testing process is about 5700 images. Thus, if the most represented class is chosen the accuracy will be over 25%. While the objective of the research is to achieve a model that is useful in real world scenarios, the aforementioned statistics are considered as a benchmark criteria. Any model below 30% is no better than random prediction. With this criteria in consideration several network architectures were sieved out. One of the architecture which did not meet this criteria was a single layer LSTM network with 100 cells (the input to which was a flattened vector of size 1024 of the preprocessed input image). This proved a fully dense temporal network was not good at learning the problem, which lead to the exploration of a cascaded approach, where each frames in a sequence will be spatially reduced to a lower dimension using a convnet, and this reduced or encoded representation of the complete sequence will be used to train a recurrent network. In order to encode or reduce the spatial information meaningfully two separate networks were considered. A fully-convolutional autoencoder network and a deep convolutional network followed by fully connected layers.

### 3.3.2 Spatial Network

As mentioned in the section 3.3.1, two spatially aware networks were built to identify which showed better performance in classification. The architecture and intended use of both the FCN and CNN are described in this section.

#### 3.3.2.1 Fully Convolutional Network (FCN)

A Fully Convolutional Network (FCN) as the name suggests does not have any dense layers, it is convolutional operation from start to end. The output of the network is the same dimension as the input. There are two parts in the network the encoder and the decoder, where the encoder and decoder meets is the latent layer, which is generally a much smaller vector compared to the input size (Long, et. al., 2015). This can be thought of as a compression engine. The label is same as the input during training. Such a network is shown in the Figure 3.7.

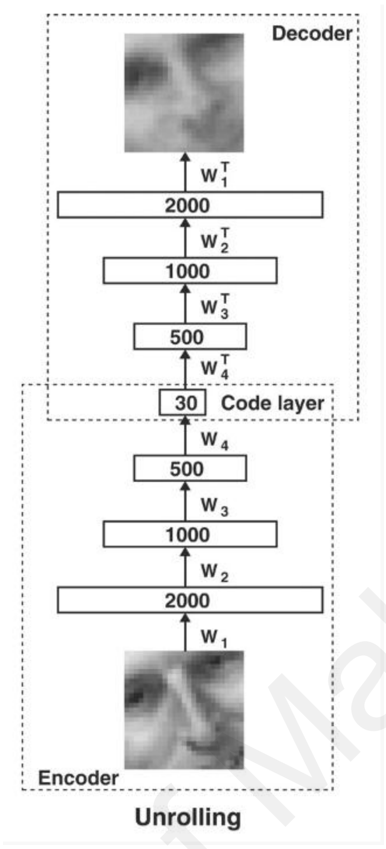


**Figure 3.7 : An autoencoder network (Hinton & Salakhutdinov, 2006)**

In this research an FCN network with 6 convolutional layers was trained, where 3 initial layers reduced the input of size (28 x 28) to a 160 sized vector. A sample of input and output images from the trained FCN is shown in Figure 3.8.

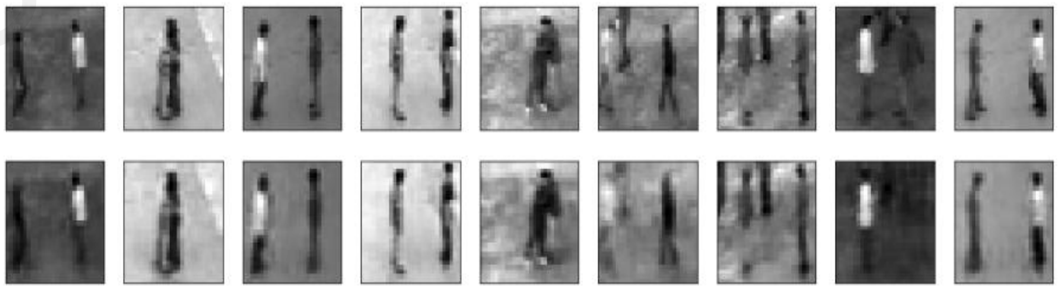


**Figure 3.8 : Top line: inputs to FCN network. Bottom line: output of FCN network to corresponding input.**

The corresponding activation of 160 values in the latent space is re-sized into a (4 X 40) vector and displayed in the Figure 3.9.

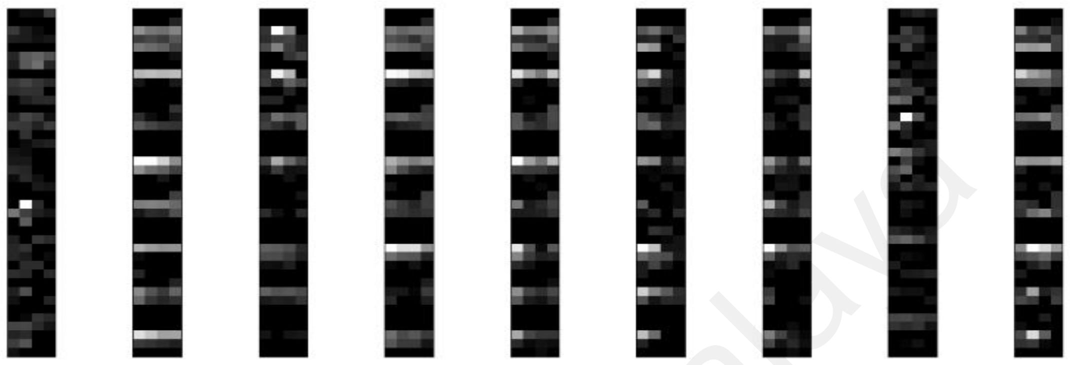


**Figure 3.9 : Output of latent layer of the FCN network. The input to which is shown in Figure 3.8**

The model was trained for 1000 epochs. The training and testing data was split with a ratio close to 7:3. The model was built and trained in Keras. The intention to build this FCN model was to extract the data from the latent layer, which is a vector of 160 activations and use it to train a recurrent network capable of capturing the temporal pattern and produce a classification of the respective human activity performed.

#### 3.3.2.2 Convolutional Neural Network (CNN)

The architecture for the convolutional neural network is as shown in the Table 3.1. The model was built and trained using TensorFlow. The model was trained for 250 epochs. The training and testing data was split with a ratio close to 7:3.

**Table 3.1: CNN architecture**

| Layer Name | Layer Description | Input Shape | Output Shape | Activation |
|---|---|---|---|---|
| Conv_1 | Convolutional Layer with 6 filters | 32X32X1 | 28X28X6 | ReLU |
| Dropout_c1 | Dropout layer, 0.5 probability | | | |
| Max_pool_1 | Max pooling layer Kernel (2X2) Strides (2X2) | 28X28X6 | 14X14X6 | |
| Conv_2 | Convolutional Layer with 16 filters | 14X14X6 | 10X10X16 | ReLU |
| Dropout_c2 | Dropout layer, 0.5 probability | | | |
| Max_pool_2 | Max pooling layer Kernel (2X2) Strides (2X2) | 10X10X16 | 5X5X16 | |
| Flatten | | 5X5X16 | 400 | |
| Fc_1 | Fully connected layer | 400 | 120 | ReLU |
| Dropout_fc1 | Dropout layer, 0.5 probability | | | |
| Fc_2 | Fully connected layer | 120 | 84 | ReLU |
| Dropout_fc2 | Dropout layer, 0.5 probability | | | |
| Fc_3 | Fully connected and final layer. | 84 | 6 | None |

This particular architecture was finalized after couple trials with hyperparameters. The dropout layers were added to reduce the model from overfitting. The intention to build this CNN model was to fully train it on individual frames of human activity and the ground truth (which class the image was obtained from) and once trained, extract the activation of layer Fc_2, which has an output vector of size 84 and use it as input to the recurrent neural network for final classification of respective human activity.

### 3.3.3 Temporal Network

The two networks trained previously, i.e. the CNN and FCN network only analyze individual frames and therefore only possessed spatial information. In order to classify a video, temporal information is essential. The network chosen for this purpose is a “Long Short Term Memory” or LSTM network. The network consisted of one LSTM layer with 100 cells followed by 4 fully connected hidden layers. The network architecture is as shown in the Table 3.2.

**Table 3.2: LSTM architecture**

| **Layer name** | **Layer description** | **Output shape** | **Activation** |
|---|---|---|---|
| LSTM cell | LSTM layer with 100 cells | 100 | |
| Fc_1 | Fully connected layer | 2000 | ReLU |
| Dropout | Dropout; probability 0.2 | | |
| Fc_2 | Fully connected layer | 1000 | ReLU |
| Dropout | Dropout; probability 0.2 | | |
| Fc_3 | Fully connected layer | 100 | ReLU |
| Dropout | Dropout; probability 0.2 | | |
| Fc_4 | Fully connected layer | 6 | Softmax |

In order to train the LSTM network in Keras the input data needs to be arranged into a vector where the first dimension is total training samples, second dimension is the number of timesteps in a sequence, and the third dimension is the number of features for each time-steps. It can be represented as [samples, time-steps, features]. One of the challenges with this arrangement is that the number of time-step in each sequence of incoming video needs to be fixed. The number of time steps is equal to the number of frames in a video, which can be varying with video length. One way to overcome this issue was to fix the number of timesteps regardless of the video length. When the video length exceeds the prefixed timesteps the inference in RNN will be split into multiple overlapping stages. When the number of frames is less than the prefixed time-steps the difference in time-steps will be filled with vectors of zeros. Training the model will learn that zero does not carry any information.

### 3.3.4 Spatio-Temporal Network

Now that two different spatial networks and one temporal network is built, it is time to combine them and train to obtain the final classification.

#### 3.3.4.1 FCN-LSTM architecture

In this arrangement each incoming frames of a sequence will be first passed through the FCN network which will encode the incoming image into a 160 (4X4X10) sized flat vector, this is shown in the Figure 3.10.

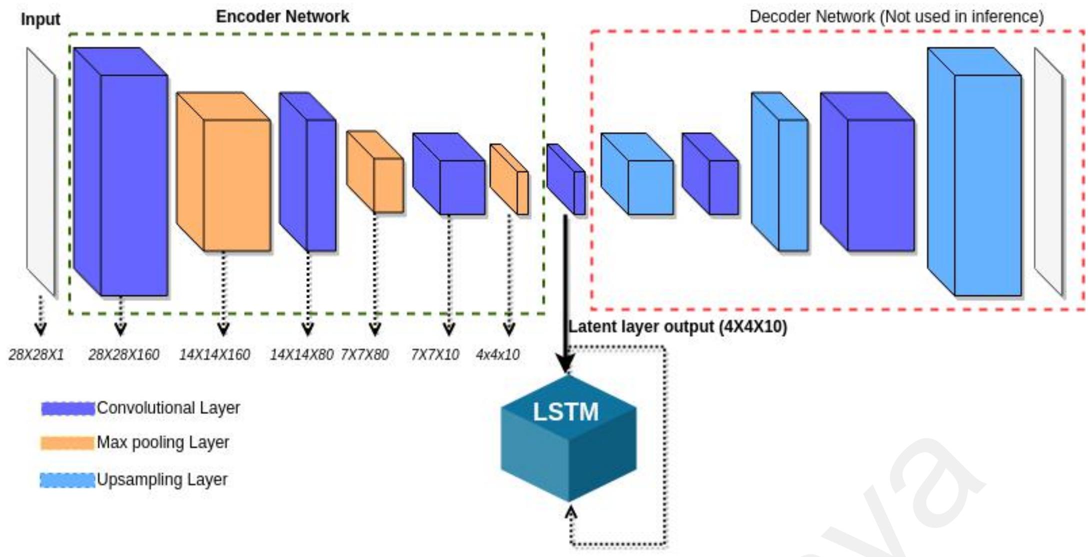


**Figure 3.10 : FCN-LSTM architecture**

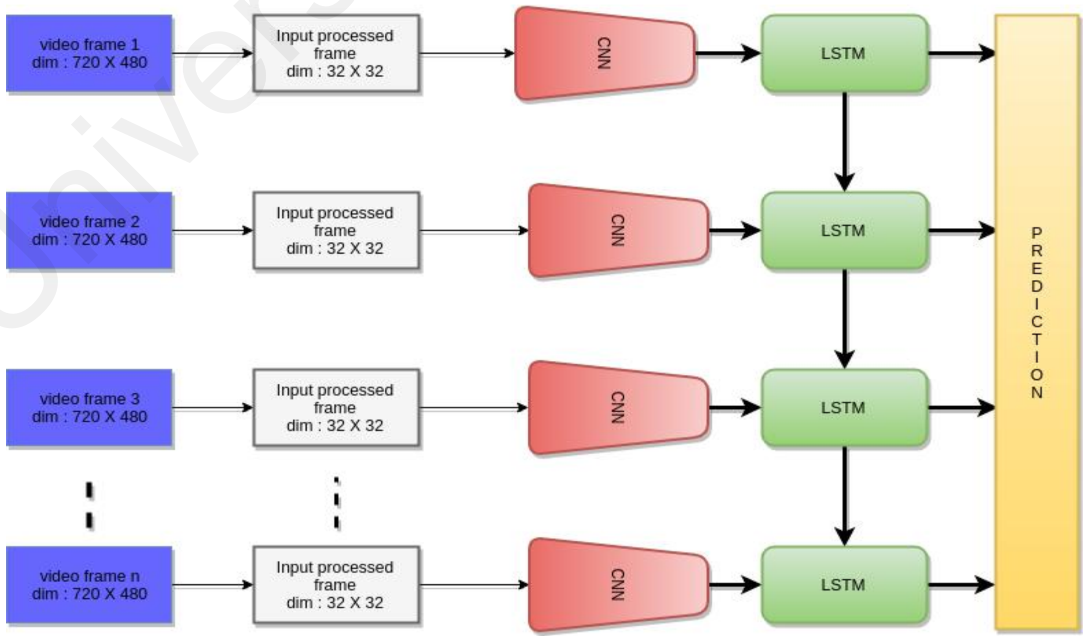
For each video sequences first the inference for all frames in that sequence will be obtained. This will be a vector of size [frame number X 160]. Since there can be varying number of frames in each video it is padded with zeros to obtain a fixed number of frames for all inputs to the recurrent network. After analyzing the distribution of frames in training size this fixed time-step or frame numbers was chosen to be 100. This is because the longest video in the training set only produced 93 frames. After training, the models performance was not satisfactory. The overall performance of the model was about 70% accuracy on the whole dataset, but with the test set (which contained 40% of the whole dataset) the accuracy was 30%, which means the model was mostly overfitting.

#### 3.3.4.2 CNN-LSTM architecture

In this setup each incoming frames went through the CNN and the activation of the second to last fully-connected layer (Fc_2) was calculated. This was a vector of 84 activations. The rest of the network is similar to the FCN and LSTM network as discussed in the Section 3.3.4.1. The same operation of padding as discussed in Section 3.3.4.1 to obtain the fixed number of frames for all input videos was also performed in this architecture. The model was trained for 25 epochs. The accuracy of the model was above 97% on the test set. The model had performed with high accuracy. The loss and accuracy of this model is included in the Chapter 4 : Results.

## 3.4 End to end inference

As the performance of the cascaded CNN and LSTM network was superior it was chosen for the building the end to end inference network. The end to end inference network take a video file as input and produces the final prediction as shown in Figure 3.10. The pipeline is visualized in the image below. The downward arrows connecting LSTM network shows the temporal relation.



video frame 1 dim : 720 X 480
Input processed frame dim : 32 X 32
CNN
LSTM
video frame 2 dim : 720 X 480
Input processed frame dim : 32 X 32
CNN
LSTM
video frame 3 dim : 720 X 480
Input processed frame dim : 32 X 32
CNN
LSTM
video frame n dim : 720 X 480
Input processed frame dim : 32 X 32
CNN
LSTM
PREDICTION

**Figure 3.11 : Complete pipeline of the CNN-LSTM human interaction recognition architecture.**

# CHAPTER 4: RESULTS

This chapter presents the results obtained from the machine learning models, the hyperparameters chosen, and the systems used for the development environment to obtain the results.

The coding for the project was mostly done using Python, and the frameworks used for building and training the neural network model was TensorFlow. TensorFlow was chosen because of the availability of several helper regularly used helper functions and its ability to train in parallel on a GPU. It was essential that the model be trained using a GPU because a lot of prototypes were tried out and the training process was extremely slow without parallel execution. The TensorFlow version used was 1.3+.

The CNN and FCN models was trained on an Amazon Web Services (AWS) computing instance for faster performance. The system specification are as follows:

Processor :Intel Xeon E5-2686 v4 (Broadwell)

GPU : NVIDIA Tesla M60 / 8 GB GDDR5

OS : Ubuntu 16.04

The LSTM network was trained on the local machine. The inference routine was also tested on the same local machine. The specs of which are as follows:

Processor : Intel® Core™ i7-7700HQ CPU @ 2.80GHz × 8

Memory : 8GB

GPU : GeForce GTX 1050 Ti/ 4 GB GDDR5

OS : Ubuntu 16.04 / 64-bit

## 4.1 CNN performance

The convolutional neural network trained for the network used 2 layers of convolution and max pooling followed by 3 fully connected layers. The final model was trained for 250 epochs with a train test split of 7:3. The accuracy on the test data was 98.5% and the accuracy on training set reached 99.4%. The model plateaued in performance at about 200 epochs. The accuracy on test set is shown in Figure 4.1. The learning rate used was 0.001, and the optimizer chosen was Adam. Adam is a variant of stochastic gradient descent and has been empirically proven to perform better than many other stochastic optimization methods (Kingma & Ba, 2014).

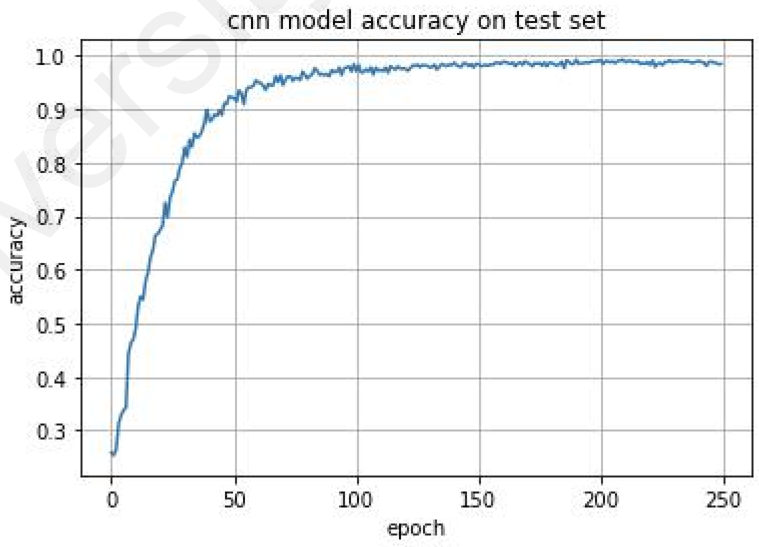
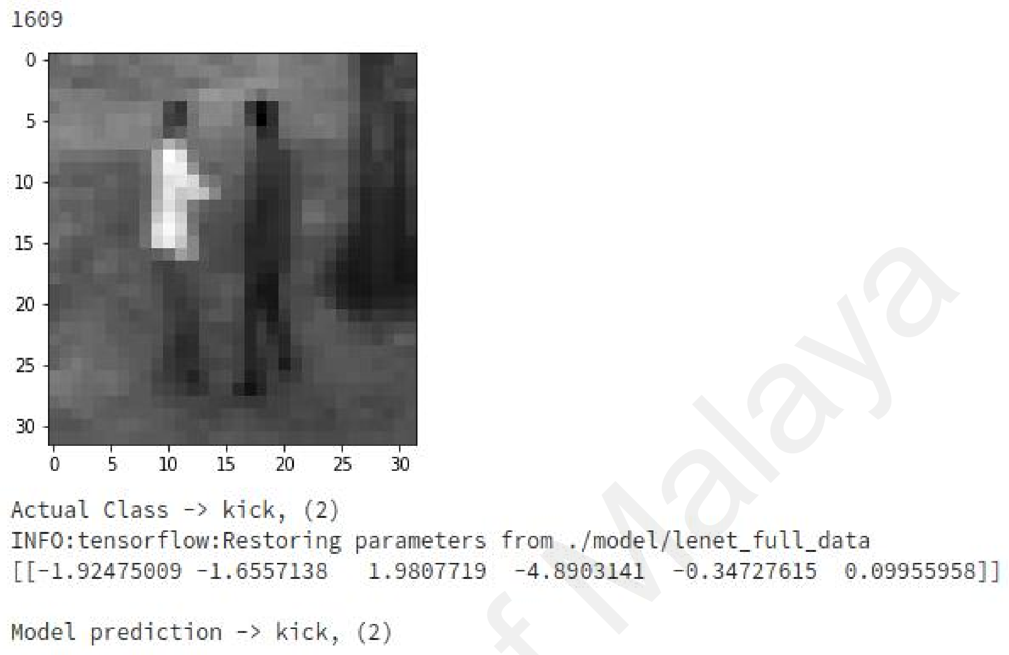


Figure 4.1 : CNN model accuracy on test set

An instance of prediction from test set is shown in the Figure 4.2 along with the output of the softmax layer.



```
1609
Actual Class -> kick, (2)
INFO:tensorflow:Restoring parameters from ./model/lenet_full_data
[[-1.92475009 -1.6557138   1.9807719  -4.8903141  -0.34727615  0.09955958]]

Model prediction -> kick, (2)
```

**Figure 4.2 : CNN model prediction on random test sample.**

## 4.2 FCN performance

The fully-convolutional autoencoder network had three convolutional layers each followed by a max pooling layer, which lead to the latent layer of vector size 160. The latent layer was followed by the decoder layer which had three convolutional layers each followed by an upsampling operation (Wang et. al., 2017). The train test split was 7:3 and the optimizer used was Adam. The network was trained for 1000 epochs. Figure 4.3 shows the training accuracy and loss of the model on test and train data.

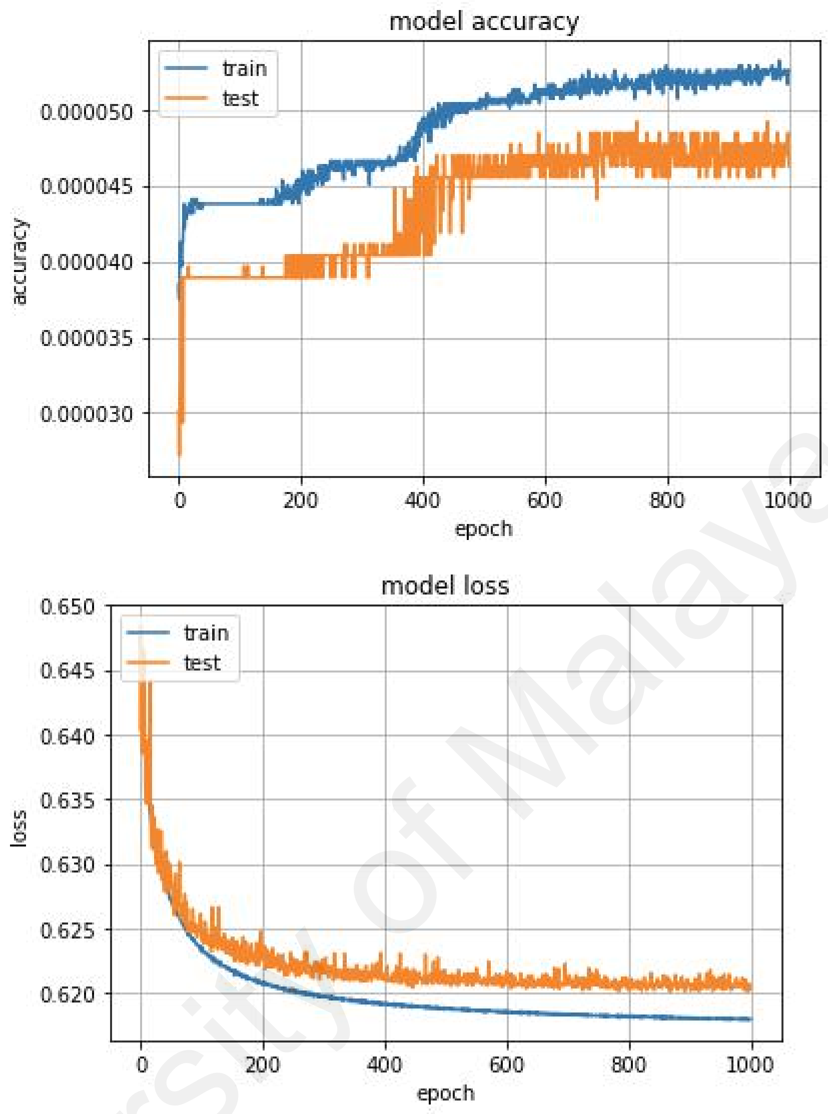


**Figure 4.3 : FCN model accuracy and loss on train and test set**

Figure 4.4 and Figure 4.5 shows the comparison in performance between the same model trained for 1000 epochs and 100 epochs.

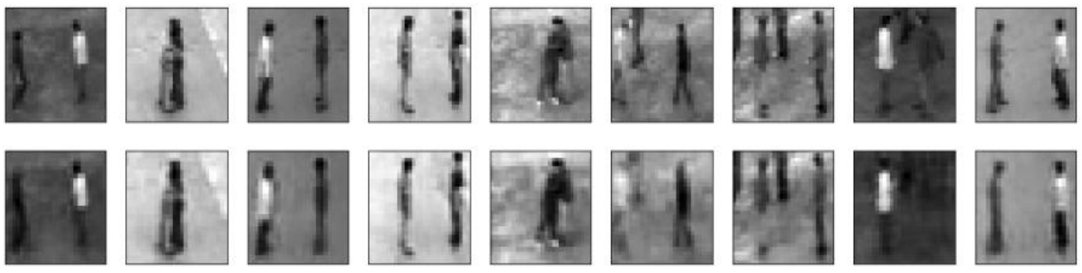


**Figure 4.4 : FCN input and output after 1000 epochs**

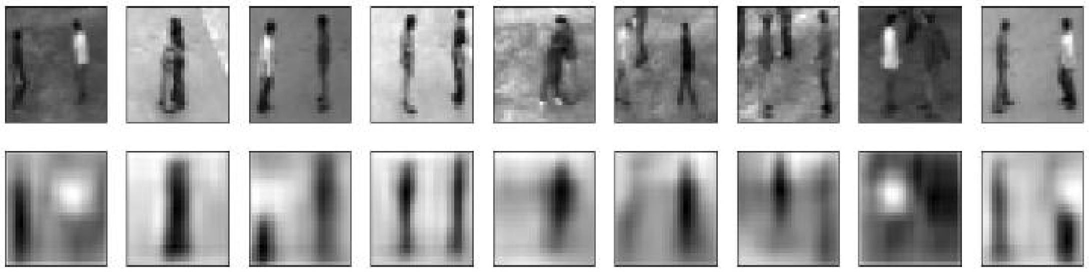


**Figure 4.5 : FCN input and output after 100 epochs**

## 4.3 LSTM performance

The last stem in the research was to train the LSTM network. In order to train the LSTM network the input data is the encoded vector from the spatial models. The output is the predicted class. Since there was two spatial models: FCN and CNN the LSTM network was trained on both.

The difference in performance is evident. The CNN followed by LSTM showed superior performance. The optimizer used in both cases were Adam, and train test split was of the ration 6:4. Both networks were trained for 25 epochs. The loss and accuracy from the first few epochs of CNN followed by LSTM is shown in Figure 4.6. The model performed close to 99% in test data.

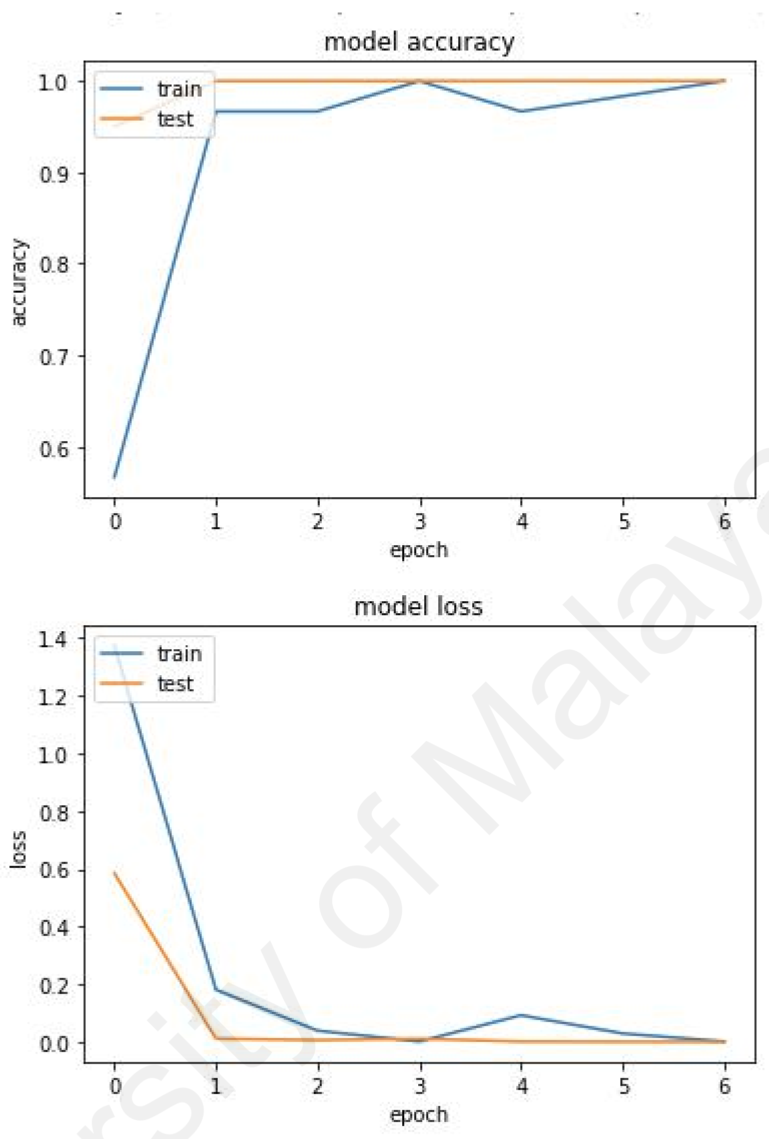
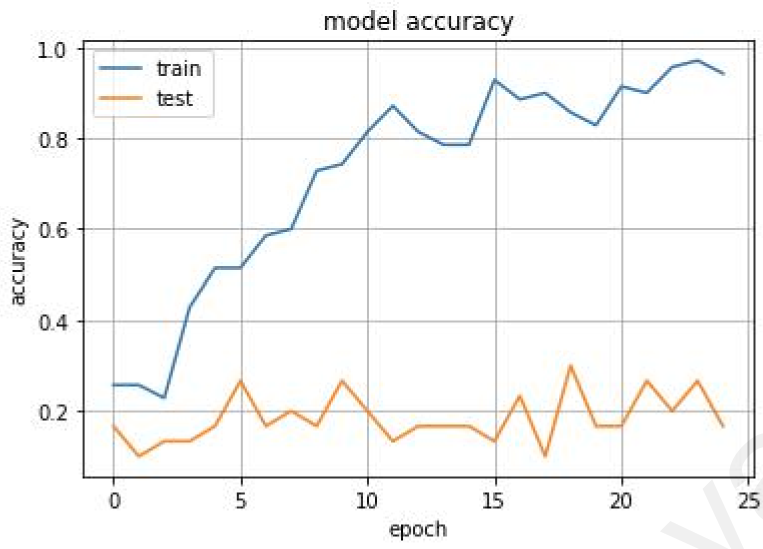


**Figure 4.6 : CNN-LSTM loss and accuracy on train and test set in first few epochs**

The accuracy and loss on the FCN followed by LSTM layer is shown in Figure 4.7. It is visible from the accuracy of the test set that the model is not learning anything about the data, it stays constant even after 25 epochs of training. At best the performance of the model on test set was about 35% which is not so much better than random prediction.



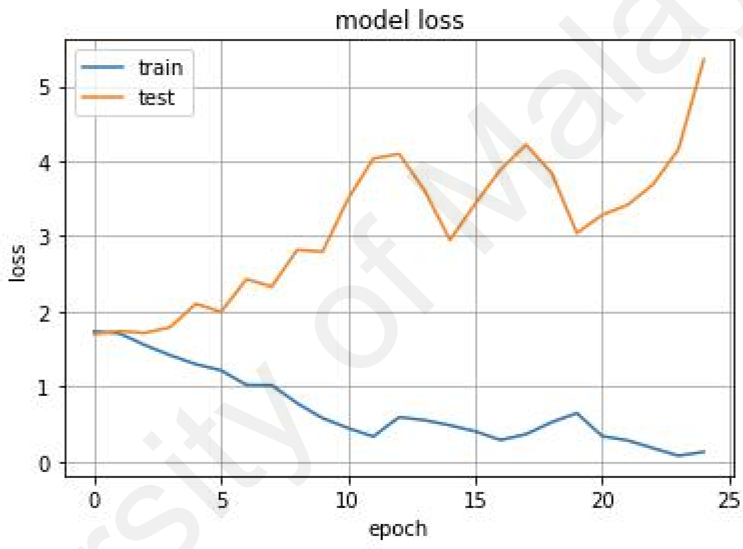


Figure 4.7 : FCN-LSTM loss and accuracy on train and test set

## 4.4 Summary

In conclusion the CNN-LSTM network has shown good performance and it is evident from the testing that it is due to the fact that the CNN network was trained on the human activity dataset, which has lead the spatial network to learn features pertaining to the classification task. FCN being an autoencoder network, was trained to compress the input into a latent space and then reproduce an output which is very close to the input image. Following this analysis it can be deduced that the FCN-LSTM network’s lack of accuracy is due to the fact that the spatial part of the network was not trained on the human activity dataset and therefore did not learn the features relating to the task of human activity classification.

# CHAPTER 5: CONCLUSION

This research has developed and tested several machine learning models on a human interaction dataset. The CNN-LSTM model has successfully learned the classification problem. This was verified by doing an inference on a test set which was never used during training. By increasing the size of the dataset to include more interactions and agents this model can be further scaled for a more general human interaction detection task.

In Chapter 2, the complexities of the tasks involved in human activity recognition was discussed, and it is evident that a feature engineering based approach would not do a good job due to edge cases and the amount of possible interactions and the settings in which these interactions take place can be vastly different. The importance of developing a robust technique for classification was also discussed.

## 5.1 Future work

There are mainly two areas in which the research can be further developed. They are discussed in this section.

### 5.1.1 Human pose estimation in spatial layer

The performance of the CNN-LSTM model was due to the fact that it was able to both analyze and include the spatial and temporal nature of the problem at hand. If there was an abundant source of labeled data  and computational resources available, perhaps it would have been possible to design a plain and massive deep neural network to perform the classification. However, since that is not the case it is important to explore innovative methods to build models which have features capable of representing the underlying statistical variance of the classification problem. For example the CNN is very good at analyzing spatial information, the weights are shared between all the patches of an image for any given kernel. This provides the advantage of spatial awareness, while being easy to train. Similar efforts can be made to identify novel methods in analyzing human activity. The human posture is a feature that is shared among all the input images in the human activity recognition dataset (Boulay, et. al, 2003). By including human pose estimation into the spatial layer the model could learn more useful features from the same amount of data.

### 5.1.2 Human recognition and boundary detection

The current model assumes the input only has one human to human interaction happening and it is not fully scale invariant, that is even if there is only one human to human interaction present but if the action space is only consuming a smaller area of the entire frame the model might not perform well. This is because of the fact that there was on average a 50% - 60% space occupancy of for every interaction recorded in the dataset. This can be overcome by using a bounding box or pixel wise semantic segmentation before running the inference of the data on the final model. SSD like networks are really fast at doing this (Liu, et. al., 2016) and by using transfer learning the model can be trained to do that with minimal modifications.

# REFERENCES

Aggarwal, J. K., & Cai, Q. (1999). Human motion analysis: A review. Computer vision and image understanding, 73(3), 428-440.

Boulay, B., Bremond, F., & Thonnat, M. (2003). Human posture recognition in video sequence. In IEEE International Workshop on VS-PETS, Visual Surveillance and Performance Evaluation of Tracking and Surveillance.

Chatzigiorgaki, M., & Skodras, A. N. (2009, July). Real-time keyframe extraction towards video content identification. In Proceedings of the 16th international conference on Digital Signal Processing (pp. 934-939). IEEE Press.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on (Vol. 1, pp. 886-893). IEEE.

Davis, J. W., & Bobick, A. F. (1997, June). The representation and recognition of human movement using temporal templates. In Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on (pp. 928-934). IEEE.

Gao, B. (2011). Exemplar-based human interaction recognition: features and key pose sequence model (Doctoral dissertation, Applied Science: School of Computing Science).

Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). Learning to forget: Continual prediction with LSTM.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. science, 313(5786), 504-507.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. The Journal of physiology, 148(3), 574-591.

Ke, S. R., Thuc, H. L. U., Lee, Y. J., Hwang, J. N., Yoo, J. H., & Choi, K. H. (2013). A review on video-based human activity recognition. Computers, 2(2), 88-131.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

Kumari, S., & Mitra, S. K. (2011, December). Human action recognition using DFT. In Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2011 Third National Conference on (pp. 239-242). IEEE.

Li, C., & Hua, T. (2011). Human action recognition based on template matching. Procedia Engineering, 15, 2824-2830.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621.

Ryoo, M. S., & Aggarwal, J. K. (2010, August). UT-interaction dataset, ICPR contest on semantic description of human activities (SDHA). In IEEE International Conference on Pattern Recognition Workshops (Vol. 2, p. 4).

Seki, M., Fujiwara, H., & Sumi, K. (2000). A robust background subtraction method for changing background. In Applications of Computer Vision, 2000, Fifth IEEE Workshop on. (pp. 207-213). IEEE.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.

Thoma, M. (2017). Analysis and optimization of convolutional neural network architectures. arXiv preprint arXiv:1707.09725.

Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., & Cottrell, G. (2017). Understanding convolution for semantic segmentation. arXiv preprint arXiv:1702.08502.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10), 1550-1560.

Wren, C. (1995). Real-time tracking of the human body. Photonics East, SPIE, 2615.

Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853.

Yoon, S., Won, C. S., Pyun, K., & Gray, R. M. (2003, March). Image classification using GMM with context information and with a solution of singular covariance problem. In Data Compression Conference, 2003. Proceedings. DCC 2003 (p. 457). IEEE.

Zhang, G. P. (2000). Neural networks for classification: a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 30(4), 451-462.