ROBOTIC INTELLIGENCE USING DEEP LEARNING ON GRAPHICALLY STRUCTURED DATA

A *Thesis Project* for
the Degree of *Bachelor of Technology*

by

Sudarshan Kamath - 150103069
Diptanshu Agarwal - 150103029

To the Department of Mechanical Engineering,
Indian Institute of Technology Guwahati
May, 2019

Certificate

It is to certify that the work contained in the bachelor thesis report titled "***Robotic Intelligence using Deep Learning on Graphically Structured Data***", by *Sudarshan Kamath* and *Diptanshu Agarwal*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Prof. Shyamanta M Hazarika
Department of Mechanical Engineering
Indian Institute of Technology Guwahati
May, 2019

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Sudarshan Kamath
Roll No. 150103069

Diptanshu Agarwal
Roll No. 150103029

Approval Sheet

The project entitled "***Robotic Intelligence using Deep Learning on Graphically Structured Data***" is approved for the degree of Bachelor of Technology in Mechanical Engineering.

Prof. Shyamanta M Hazarika
Department of Mechanical Engineering
Indian Institute of Technology Guwahati
May, 2019

Acknowledgement

This is to acknowledge that this bachelor thesis report was written under the guidance of *Prof. Shyamanta M Hazarika.* The entire direction of this project from the ideation to the execution has been supervised by *Prof. Hazarika* and we are thankful for his help and support.

Sudarshan Kamath
Diptanshu Agarwal
Department of Mechanical Engineering
Indian Institute of Technology Guwahati
May, 2019

Abstract

Deep Recurrent Neural Network architectures, have often been used with a high level intuitive spatio-temporal structure for robotic intelligence applications. Spatio-temporal graphs are a popular tool for imposing such high-level intuitions in the formulation of real world problems, however they require a large amount of data and are unable to model complex systems. In this project, we propose an alternative for spatio-temporal graphs in the form of Temporal Activity Graphs which provide a more intuitive high level structure that is, feedforward, fully differentiable, and jointly trainable. We would be modifying the Temporal Activity Graph for usage in one particular field of Robotic Intelligence namely, Human Activity Recognition and would also be providing an insight into the applications of this approach in other intelligent systems. Further, we would be using a new form of RNN architecture called the sRNN to model the TAG into a deep learning system which is more intuitive and would make the encoding of sequential information more efficient. We would also be looking into adding additional features while forming the TAG in order to ensure better encoding of the raw sequential data. These features would be procured with the help of Human Skeletal Data. Finally, we will be implementing alternative classifying techniques in order to obtain higher accuracies based on the neural network output.

Contents

List of Figures 9

1. **Introduction** 10-13

    1.1 Motivation. 10-12
    1.2 Objectives 12
    1.3 Thesis Overview 13

2. **Literature Review** 14-32

    2.1 Qualitative Spatio-Temporal Reasoning Paradigm 15-19
        2.1.1 Extended CORE9 15
        2.1.2 Temporal Activity Graphs 15-18
        2.1.3 Illustrative Example 18-19
    2.2 Learning Paradigm 19-31
        2.2.1 Recurrent Neural Networks 20-22
        2.2.2 Long Short Term Memory 22-25
        2.2.3 Structured RNN Architecture 25-26
        2.2.4 Random Forests 27-28
        2.2.5 Siamese Networks 29-30
        2.2.6 Regularisation 30-31
    2.3 Skeletal Data 31-32

3. **Working Pipeline** 33-34

4. **Experiments and Evaluation** 35-45

    4.1 Data Extraction 35-36
    4.2 Preliminary Architecture 36-37
    4.3 Preliminary Analysis 37-39
    4.4 Intermediate Architecture 39-40
    4.5 Intermediate Analysis 40-41

4.6 Final Architecture 41-42
4.7 Final Analysis 42
4.8 Local Testing 43-44

5. Conclusion 45
Future Work 46-47

References 48-50

List of Figures

1.1 *Handshake* from UT-Interaction dataset 11

2.1 TAG depicting a video activity 16
2.2 TAG corresponding to *Handshake* Activity 19
2.3 RNN 20
2.4 RNN depicting Loss Function 21
2.5 RNN depicting Back Propagation 22
2.6 *Tanh* and its derivative function 23
2.7 LSTM 24
2.8 LSTM with gates 25
2.9 sRNN from st-graph 26
2.10 Random Forest 27
2.11 Decision Tree 28
2.12 Siamese Networks 30
2.13 Human Skeletal Data 31
2.14 Skeletal Data Extraction at IITG 32

3.1 Working Pipeline 33

4.1 SBUKI dataset 36
4.2 Primary sRNN Architecture. 36
4.3 Intermediate Architecture 39
4.4 Final Architecture 41
4.5 Correctly labelled examples at IITG 43
4.6 Falsely labelled examples at IITG 43

Chapter 1

Introduction

Robots are programmable machines which are usually able to carry out a series of actions autonomously, or semi-autonomously. In the 21st century, robotics goes hand to hand with Artificial Intelligence(AI). AI is a broad field involving developing philosophies to complete tasks which would otherwise require human intelligence. AI algorithms can tackle learning, perception, problem-solving, language-understanding and/or logical reasoning. Artificially intelligent robots are the bridge between robotics and AI. These are robots which are controlled by AI programs.

Many robots are not artificially intelligent. Up until quite recently, all industrial robots could only be programmed to carry out a repetitive series of movements. Non-intelligent robots are quite limited in their functionality. AI algorithms are often necessary to allow the robot to perform more complex tasks. One such complex task is Human Activity Recognition.

1.1 Motivation

Human Activity recognition(HAR) is the problem of predicting the movement of a person based on sensor data, such as an accelerometer/video in a smartphone. An example of the importance of HAR would be its utilization in a factory where Human Machine Interaction is a common phenomena. If one can design an intelligent system which is able to predict the safety of the maneuvers performed by the Human, one can ensure a reduction in the number of accidents on the factory floor. Another real life example[1][2] of HAR is China's project to implement a nationwide surveillance system which aims to predict crimes before they can occur by analysing the behavior of the persons of interest. Such surveillance systems are of importance to most countries and hence a lot of research work has been dedicated into finding solutions to HAR which would make the public surveillance systems almost completely autonomous. As an example, a sequence of frames for an instance of the handshake activity from the *UTI* dataset[3] is shown in the figure below. Learning

an interaction model for handshake would allow a HAR system to recognise the handshake activity in any video thereafter.

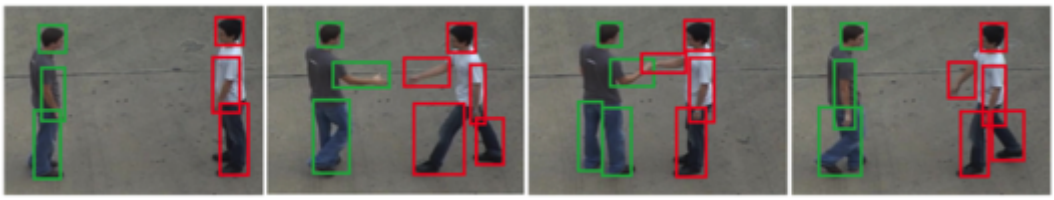

Figure 1.1 : *Handshake* from the UT-Interaction Dataset

Having understood that this field is of incredible importance, various methods have been developed to provide AI solutions for HAR. We studied these approaches and with the help of the state-of-the-art, designed a new system which combines important elements of multiple first in line research methodologies with the hope of increasing the efficiency and accuracy of the current state-of-the-art.

Spatio-temporal graphs[4](st-graphs) are a popular general tool for representing high-level spatio-temporal structures. But spatio-temporal graphs need large amounts of training data and are often too simple to handle complex representations of high level spatio-temporal data involving extended objects. Extended objects have also been shown to be effective for the representation of Human Activities[3], hence we would be using the same for our representation of humans in video frames. Using extended objects we define an activity in the form of a Temporal Activity Graph(TAG). Nodes in the graph correspond to components of interacting objects, labels on the edge describe the qualitative spatial relations. The set of sequences over different time points express the relational changes between the different objects of an activity. When human bodies are abstracted as extended objects, the graph structure allows a neat way to keep track of how the spatial relations are evolved between the body parts.

Neural Networks have been widely accepted as an important tool for solving tasks involving prediction, recognition, classification and much more. RNNs have proven their capability on many end-to-end learning tasks which are sequential in nature, especially when used with the LSTM[5] architecture. However, they lack a high-

level intuitive spatio-temporal structure though they have been shown to be successful at modeling long sequences. Therefore, augmenting a high-level structure with learning capability of RNNs leads to a powerful tool that has the best of both worlds. We propose a sRNN architecture which is modified to work with the TAG so as to provide a high level intuitive structure which would perform better that unstructured (plain-vanilla) RNNs.

Human Skeletal Data is commonly accepted to be a feature rich entity for representation of video data where the relative movement of human joints encode most of the information required for classification[6]. Hence we would be adding Skeletal data to enhance our feature vector.

Alternative classification strategies such as Random Forests[7] are known to prevent overfitting. Siamese Networks[8] are commonly used when the size of the datasets are relatively smaller as compared to the conventional ones. Hence, we would be adapting these ideas into our sRNN architecture and selecting the combination which provides the best results.

1.2 Objectives

Our objective is to design and implement a Recurrent Deep Learning Architecture based on ideas motivated by the sRNN architectures and commision them to work in synchronisation with the modified Temporal Activity Graphs which have been suitably tailored to represent video activities in a feature rich manner.

We would be implementing our architecture on pre-existing datasets such as the SBUKI and ME datasets. We shall also be creating our own examples based on video data captured at IITG.

We would also be utilising the ideologies behind ‘one-shot learning’ algorithms in order to achieve better classification accuracies. We will be comparing this with a commonly used alternative to the Softmax classifier, i.e. the Random Forest Classifier and would hence perform a comparative analysis in order to understand the pros and cons of both these methods.

1.3 Thesis Overview

All end to end deep learning projects involve at least 2 parts. One is the data processing which involves all the steps from procuring raw data to generating necessary features and structuring it to be fed into the Deep Learning toolbox. The second part is the deep learning toolbox itself which can be modified and optimised to provide the best results over a particular dataset. We break our thesis into two main parts. One is the modification of the raw data from the respective datasets to form the Temporal Activity Graph(TAG)[3] and simultaneously modify the features of the TAG to enhance the encoding process. Second is the formulation of the modified sRNN[4] architecture, made fit to use the TAG as data input. Our reasons for choosing these architectures have been explained in the appropriate sections.

Efforts have been made to make this thesis more or less self-sufficient. The sections are a step-by-step guide into our project and they describe the entire project pipeline.

Chapter 2 consists of the literature review which focuses on the on-going works in this field. It also provides some mathematical basis to the AI based algorithms used by us. It acts as a guide for the further sections.

Chapter 3 tells the reader about our working pipeline and our approach towards the problem in hand and the methodology utilised to solve it.

Chapter 4 contains all of the experiments which we have performed and the results obtained. We have also analysed the results and have provided reasons behind the experiments undertaken. Each of the results have been tabulated for comparisons. Also the state-of-the art results have been provided for the reader to juxtapose with ours.

Chapter 5 discusses the future direction for anyone who is willing to work on our thesis. It provides a few suggestions and a few factors to keep in mind before proceeding further. It also summarises our thesis and concludes the entire document. It is followed by the references.

Chapter 2

Literature Review

The use of LSTMs[5] for making RNNs has been widely accepted as an effective encoding structure for capturing long term dependencies. RNNs have previously been used for anticipation of driver activities and maneuvers by researchers[9]. RNNs have also been used in low-power devices for tasks such as HAR using sensory data[10]. Use of sRNNs and st-graphs has also been discussed thoroughly by researchers[4]. They were applied for studying the tasks of human object interactions, human motion modelling, driver activity anticipation and much more. However the st-graph uses a single label for each node and edge. While this might be a convenient simplification, it is often unable to model complex features underlying each of the nodes and hence becomes ineffective for tasks such as HAR which required high level semantic features to represent each node for the learning process to be efficient. For such features, researchers have used the TAG[3] with the extended CORE9[11] integrated representation, which captures topological, directional, size, distance and motion related information in a compact form between a pair of extended objects. More details about sRNNs, extended CORE9 and TAGs have been provided in their respective sections. However TAG was designed to be used with Support Vector Machines(SVM)[3], hence sRNNs may fail to encode the features represented in a TAG for an activity. We would be improving the feature space to overcome this. Additional features would be induced with the use of skeletal information from the human bodies in video frames as explained later. This approach has often been used by researchers as an effective method to increase the quality of the feature space[6]. It was also observed that the outputs from the dense layers of the modified sRNN were not being aptly classified by a simple softmax classifier and hence we have modified our classification methodology. We have branched into two classification strategies, one, utilising a random forest and two, using a Siamese network classifier. Both methodologies have been proven to provide better classification results for a given feature vector. Further details have been described in their respective sections.

An insight into the working of the deep learning building blocks has also been provided to make this thesis self-sufficient.

2.1 Qualitative Spatio-Temporal Reasoning(QSTR) Paradigms

This section deals with the ideologies and reasonings which help us understand and represent the data in a feature rich manner by encoding the underlying spatio-temporal features qualitatively.

2.1.1 Extended CORE9[3][11]

CORE9 allows encoding of interesting spatial information between a pair of rectangles as an integrated representation[11]. Topological, directional, size, distance, and motion related information is captured by CORE9 in a compact form. An extension of CORE9 that was proposed to compute spatial information between a pair of extended objects is known as Extended CORE9[12]. Here, an extended object is defined as a collection of components, where an axis-aligned minimum bounding rectangle is used to approximate each component. It has been argued that human bodies for the purpose for HAR have been better abstracted by extended objects. Component relations and spatial relations are used to express binary spatial relation corresponding to a pair of extended objects. The spatial relations between components of one extended object with components of the other are known as component relations. Whole relations capture the general spatial relations between the extended objects. A general recursive algorithm is used to compute the component and whole relations opportunistically. Computation of topological, directional and distance relations for a pair of extended objects has been discussed within the Extended CORE9 framework. Thus, an activity is represented using the computed component relations.

2.1.2 Temporal Activity Graphs(TAGs)[3]

A TAG represents an activity by capturing the sequence of relations amongst the interacting objects. Each object in an activity depicting video is abstracted as an

extended object. Specific points in time are represented by the various frames of the activity video.

Definition 2.1.2.1: A set of sequences of whole relations and component relations over a set of time points between a pair of extended objects is known as an activity.

The intuition that change of relations between extended objects over the sequence of time points is fairly distinctive for every activity forms the basis for this representation. We represent such an activity using a temporal graph as shown in the following figure :

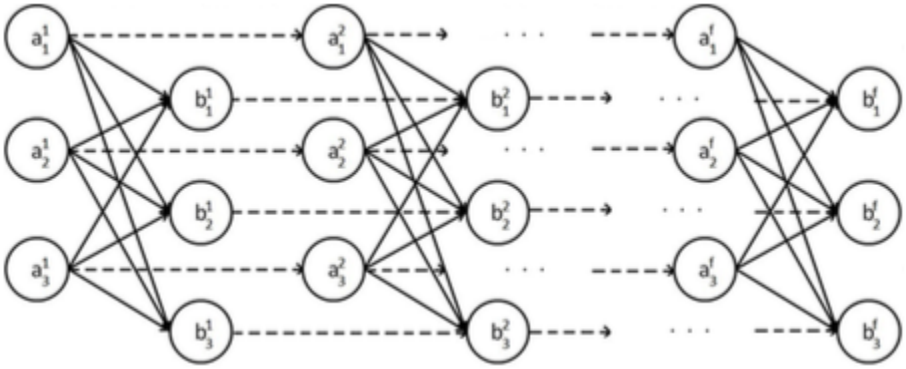

Figure 2.1 : A temporal graph depicting a video activity consisting of *f* frames[3]

Definition 2.1.2.2: A temporal activity graph *G* is formally denoted by a 5-tuple *(X, T , V , E_s , E_t)*, where,

- *X = A U B U ...,* where *A, B, ...* are any number of extended objects involved in the activity such that

A = {a_i |i = 1...n and *n* is the number of components in *A}*,

B = {b_j |j = 1...m and *m* is the number of components in *B}* and so on.

- *T : X x N → {0, 1}* is the time function. Here, *T (a_i , t) = 1* iff component a_i appears in the video activity at time point *t*. Here, a_i is a component of the extended object *A*.
- *V = {a^t_i |a_i ∈ X and T (a_i , t) = 1}* is the set of vertices.

- $E_s = \{(a_i, b_j, t) \mid a_i, b_j \in X$ and $T(a_i, t) = T(b_j, t) = 1\}$ is the set of directed spatial edges. Further, an edge label is associated with each spatial edge that is denoted by (a_i, b_j, t).
- $E_t = \{(a_i, t, t + k) \mid a_i \in X, T(a_i, t) = T(a_i, t + k) = 1, T(a_i, t + 1) = \ldots = T(a_i, t + k - 1) = 0\}$ is the set of directed temporal edges.

Semantically, $T(a_i, t) = 1$ iff A appears in the video at frame number t.

An activity involving two extended objects, A and B, is depicted as a TAG in figure 2; here, *a1, a2, a3* and *b1, b2, b3* are the components of A and B respectively. The spatial edges correspond to solid edges and are labeled using the spatial relations between the respective components at the specific time point.

Definition 2.1.2.3: A three tuple <*top-dir-dis*>, denoted by $\varepsilon(a^t_i, b^t_j, t)$, is the edge label between a^t_i and b^t_j, where *top* is the topological relation, *dis* is the distance relation and *dir* is the directional relation between a^t_i and b^t_j.

The temporal changes that appear between a^t_i and a^u_i are represented by dashed links, such that component a_i appears in the video at time point t and reappears at some subsequent time point u $(t < u)$.

a_i does not appear at any time point between t and u in the video if t and u are not consecutive. The changes in the spatial relations (obtained from the solid edge labels) between components a_i and b_j over the entire duration of the activity as recorded in the video, can thus be described by traversing along the temporal edges. Such a description can be given as a sequence of edge labels and we term it a *label sequence*.

Definition 2.1.2.4: In a TAG, G, the sequence of edge labels $<\varepsilon(a_i, b_j, 1), \varepsilon(a_i, b_j, 2), \ldots \varepsilon(a_i, b_j, t)>$ is a label sequence $ls_{i,j}$ between components a_i and b_j.

The change in spatial relations between a pair of components over time is described by a *label sequence*. The corresponding edge label, $\varepsilon(a_i, b_j, t)$ is replaced by a *NULL* value in the label sequence $ls_{i,j}$ if at any time point t, either of the components a_i and

b_j do not appear in the video. Every activity is characterised by the set of label sequences of the corresponding TAG. Comparison between the respective sets of label sequences can establish the similarity between two activities.

In the following section we will illustrate the concepts of TAG, *label sequences* and *edge-labels*.

2.1.3 Illustrative Example

A sequence of three video frames obtained from a sample *Handshake* activity in the *UT interaction* dataset, as shown in Fig. 1 is considered here. Each human body is an extended object of at most *five* components. The activity sample here involves two human bodies *A* and *B*. The following figure depicts the TAG for the activity. In the figure, components of *A* are labeled a^t_x where *x* refers to the component identifier and *t* refers to the corresponding time point; similarly components of *B* are labeled b^t_x. Here, $x \in \{h, rh, lh, rl, ll\}$; *h* refers to *head*, *rh* refers to *right hand*, *lh* refers to left hand, *r* refers to *right leg* and *lh* refers to *left leg*.

In the TAG shown in figure 3, 3-tuples topological directional and distance relations computed using Extended CORE9 are the spatial *edge labels*. The edges corresponding to a particular component are not drawn if that particular component does not appear in the frames due to occlusion. We obtain the label sequence, i.e. the sequence of edge labels for *t = 1, 2, 3, 4*, for every pair of component of *A* and *B* .

In this example, <*DR-W-Away*>, <*DR-W-Near*>, <*PO-W-Conn*>, <*DR-W-Away*> represent the label sequence for the components a_{rh} and b_{rh}. The edge labels between a_h and b_h do not change over time; the label sequence is <*DR-W-Away*>. All *label sequences* between components a_{lh} and b_x are NULL, because the component a_{lh} does not appear in any of the time points.

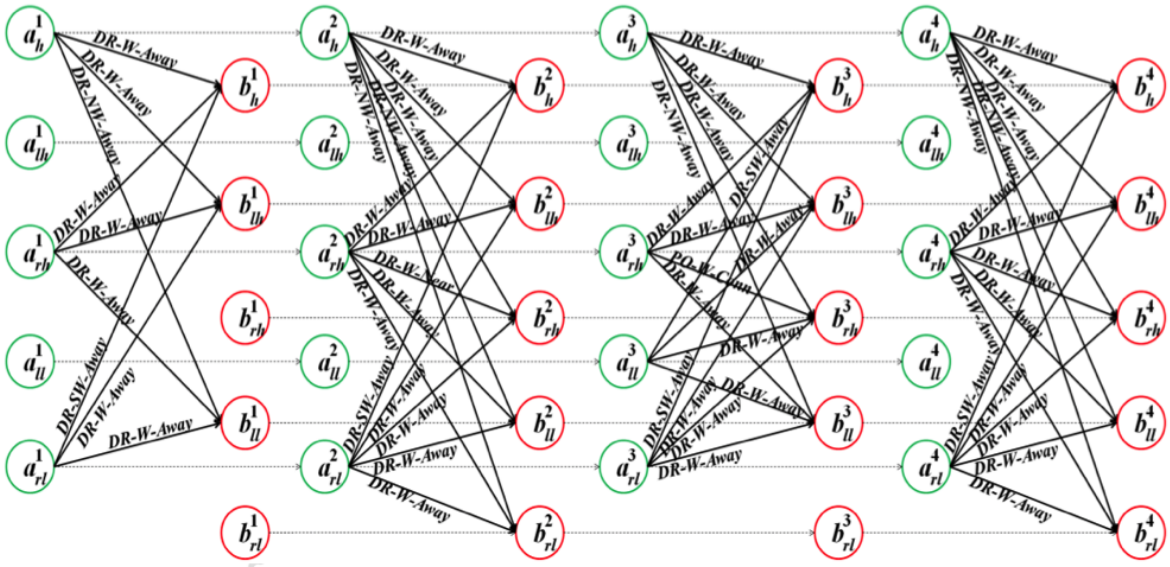

Figure 2.2 : TAG for the sequence of frames in Fig. 1 that corresponds to *Handshake* activity from the *UT-Interaction Dataset*[2]

2.2 The Learning Paradigm

In recent years, with the introduction of GPUs, high performance computations have been easier and hence machine learning has found its application in almost all tech related fields. Since our problem statement majorly details with sequential data, hence we have provided a basic introduction to the fundamentals of RNNs as well as the LSTMs used to form them. Subsequently we have also briefly provided the structural details and algorithms used in Random Forest as well as Siamese Network Classifiers.

Neural Networks can be thought of as highly non-linear function approximators. Further details on Neural Networks can be found in numerous articles, research papers and blogs [13][14]. Large amounts of data is fed to Neural Networks which then learn a suitable function which maps the input to the output classes. This is quite similar to curve fitting and if we are to overly simplify the math, we can say that Neural Networks and Least Square Fitting Algorithms basically perform the same operations, except for the fact that Neural Networks are able to work with highly non-linear functions.

2.2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are robust and powerful type of neural networks and belong to the most promising algorithms out there at the moment because they are the only ones with an internal memory. RNNs are able to remember important things about the input they received because of their internal memory. This enables them to be very precise in predicting what's coming next. Hence they have been preferred in learning sequential data.

The information cycles through a loop in a RNN. While making a decision, it takes into consideration what it has learned from the inputs it received previously along with the current input. The image below illustrates an unrolled RNN. On the left, you can see the RNN, which is unrolled after the equal sign. Note that there is no cycle after the equal sign since the different timesteps are visualized and information gets passed from one timestep to the next. This illustration also shows why a RNN can be seen as a sequence of Neural Networks.

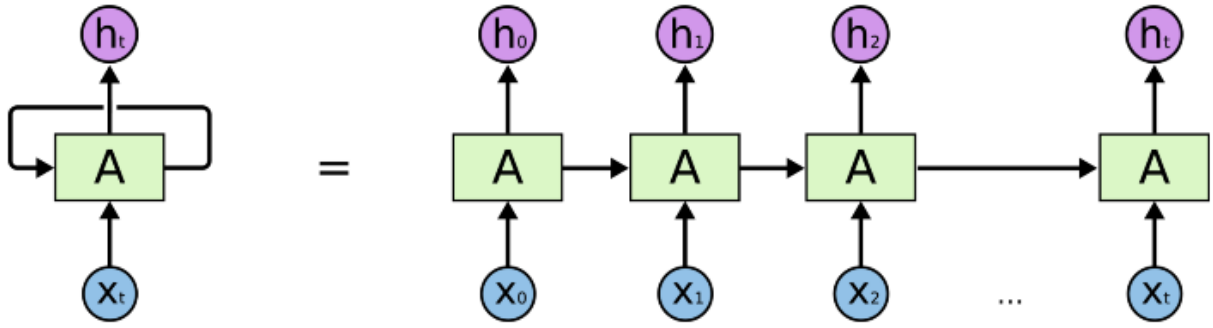

Figure 2.3 : RNN rolled over time steps[15]

Here during training time, the outputs $h_{0,} \ldots h_t$ generate error signals $E_{0,} \ldots E_t$ based on the chosen loss function. A loss function is basically a measure of the deviation of the acquired output from the expected output.

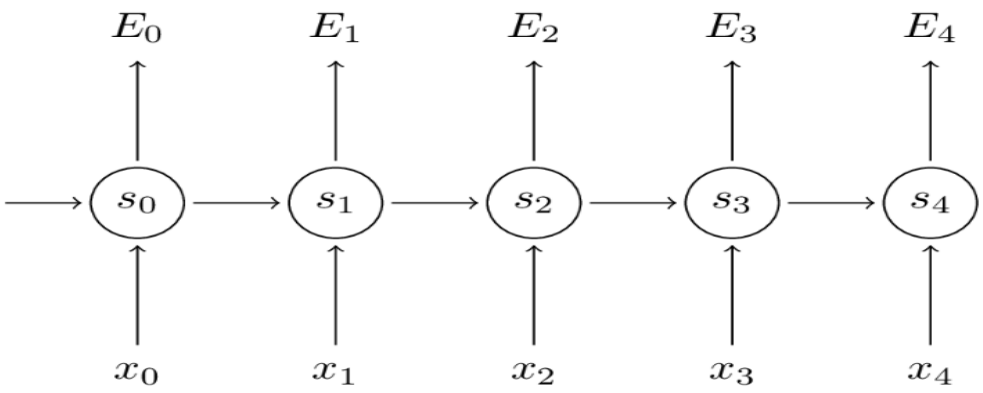

Figure 2.4 : RNN depicting Error Signals generated after computation of Loss functions[16]

Following the generation of error signals, the parameters at each time step in *A,* which are $s_{0,} \ldots s_t$ respectively, are used to update the internal weights. Weights are nothing but parameters which can be learnt based on the input data. For eg. In the least square fitting algorithm of a straight line $y = mx + c$, the parameters are m and c. Here, s can be written as :

$$s_t = tanh(U * x_t + W * s_{t-1}) \quad (2.1)$$

where U and W are the internal weights which are to be updated. The output at each time step h_t can be written as :

$$h_t = softmax(V * s_t) \quad (2.2)$$

where V is the weighing also an internal weight.

An update to weights (eg. W) can be written as $\partial E/\partial W$, and similarly for other weights. The total error E is calculated as the sum of errors $\sum E_i$. Therefore a combination of the chain rule and the summation rule for derivatives is used to perform what is known as backpropagation, to update the weights. The basic update rule is as follows :

$$W_{new} = W_{old} - \alpha * \partial E/\partial W \quad (2.3)$$

So for example, for the error in time step 3, the weights are updated in accordance to the following diagram :

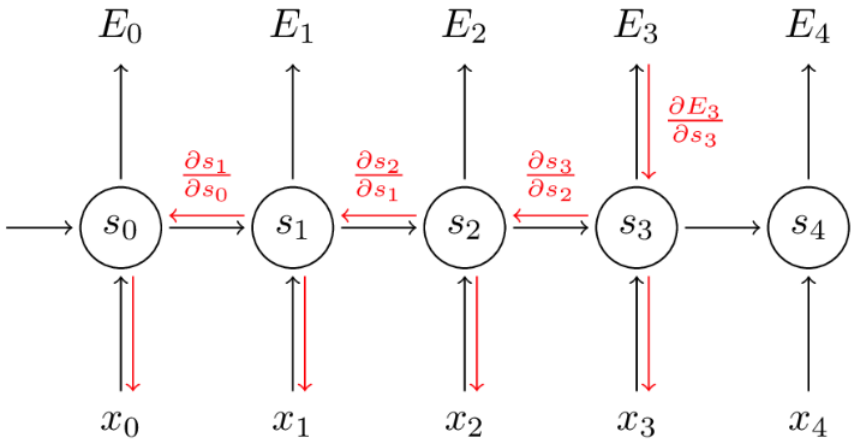

Figure 2.5 : RNN depicting propagation of partial derivatives over time steps[16]

Finally the updates from each error signal, are used to modify the internal weights. This super-position of partial derivatives is the only difference between back-propagation in RNNs and Feed-Forward Neural Networks.

Detailed algorithms and formulae can be found in many of the recent publications and have not been mentioned here to avoid deviations [17].

However it is to be noted that RNNs are suitably used when the temporal bindings of a data sequence hold more importance than the spatial bindings of individual nodes in the sequence. Hence a modification is required if one is looking to suitably encode spatio-temporal data where both spatial and temporal features hold a reasonable significance.

2.2.2 Long Short Term Memory (LSTM)

One major issue faced by RNNs is the problem of vanishing gradients. Traditionally, RNNs were unable to appreciably learn long term dependencies and hence predictions were made based on data features of nodes which were temporally close.

Mathematically this can be explained with the help of the chain rule deployed for backpropagation :

$$\partial E/\partial W = \sum (\partial E_3/\partial h_3) * \partial h_3/\partial s_3 * (\Pi \partial s_j/\partial s_{j-1}) * \partial s_k/\partial W \qquad (2.4)$$

The absolute value of the above Jacobian Matrix has an upper bound of 1. Intuitively this can be seen as the *tanh* activation maps all values into a range between -1 and 1, and the derivative is also bounded by 1:

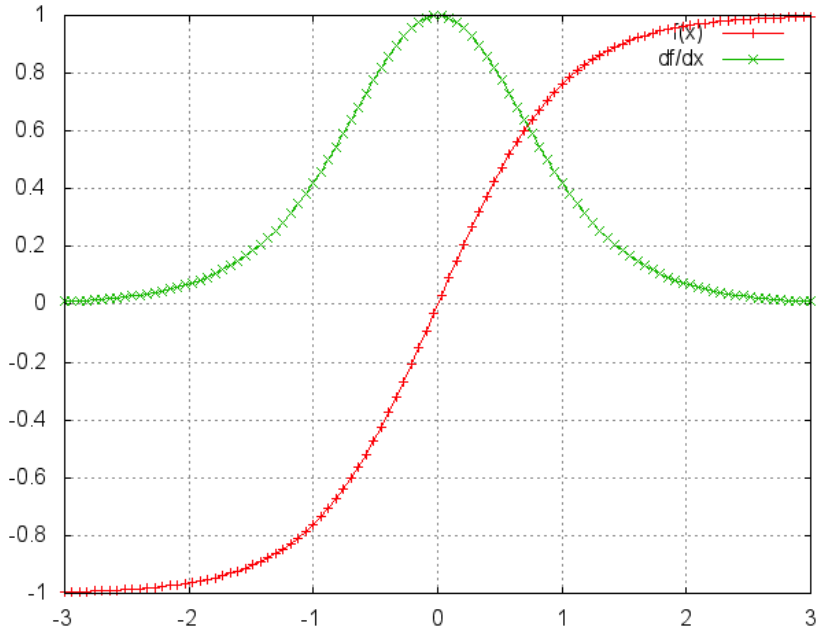

Figure 2.6 : *tanh* and its derivative function [16]

The derivative approaches zero at both ends and here the neurons or nodes are said to be saturated. Thus small values in matrix and matrix multiplications lead to gradient values to shrink very fast and hence gradient contributions from 'far away' steps turn out to be almost zero. Hence long term dependencies are not learnt.

To tackle this, the Long Short Term Memory was introduced and has been widely accepted as an efficient logic block to solve the vanishing gradient problem. This is because LSTMs contain their information in a memory, that is much like the memory of a computer because the LSTM can read, write and delete information from its memory. This memory can be seen as a gated cell, where gated means that the cell

decides whether or not to store or delete information (e.g if it opens the gates or not), based on the importance it assigns to the information. The assigning of importance happens through weights, which are also learned by the algorithm. This simply means that it learns over time which information is important and which not.

An illustration of a typical LSTM layer which also forms the basic building block for our sRNN architecture, is given below:

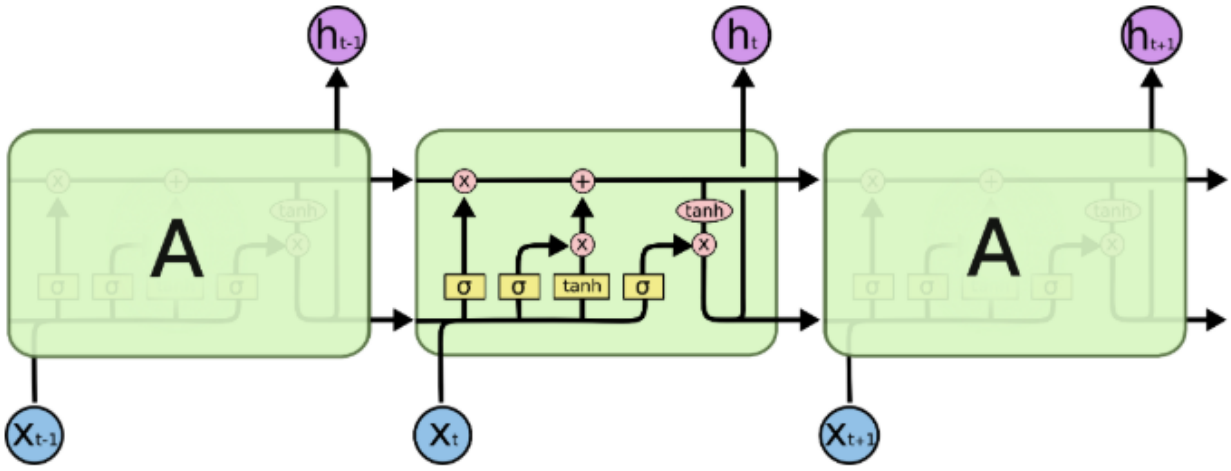

Figure 2.7 : Inside Long Short Term Memory [15]

The equations for each of the gates are given as follows :

$$i_t = \sigma(w_i * [h_{t-1}, x_t] + b_i) \quad (2.5)$$

$$f_t = \sigma(w_f * [h_{t-1}, x_t] + b_f) \quad (2.6)$$

$$o_t = \sigma(w_o * [h_{t-1}, x_t] + b_o) \quad (2.7)$$

Where i_t represents input gate, f_t represents forget gate, o_t represents output gate, represents the sigmoid function, w_x represents the weight for node x, h_{t-1} represents the output of the previous LSTM block, x_t is the current input and b_x are the biases for the respective gates.

An LSTM block in terms of gates is shown below :

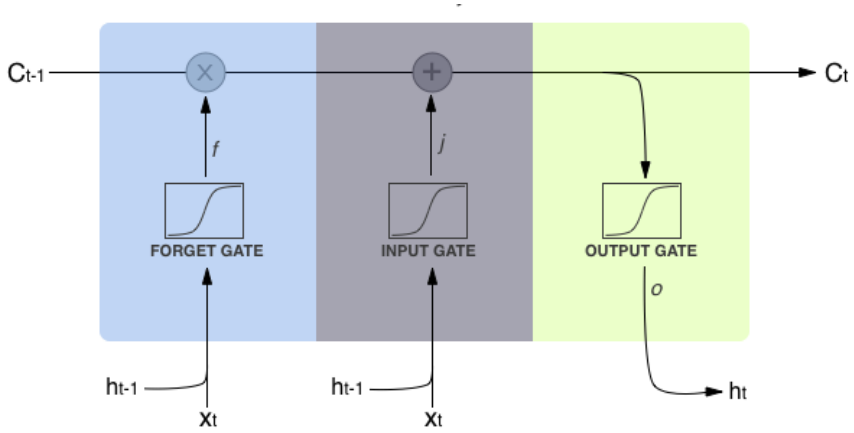

Figure 2.8 : LSTM depicted with the help of gates [15]

Gated mechanisms are hence very efficient in solving the vanishing gradient problem and therefore we have chosen LSTMs in all further RNN examples. Further details about the LSTMs' working can be found in the references.

2.2.3 Structured RNN(sRNN) Architecture[4]

The sRNN is a combination of multiple RNNs structured in a manner so as to provide semantic meaning to the data which is encoded by each network. Instead of getting a single RNN to predict the activity label, the task is broken down into simpler, easier to identify components. The following figure shows the example of the sRNN architecture proposed by the original authors.

In the figure below, an activity is showcased as a sequence of video frames at times *t-1, t* and *t+1* respectively. Instead of a temporal activity graph, the authors have encoded spatio temporal features, using a st-graph which is simultaneously converted into a factor graph which is finally fed into the sRNN architecture.

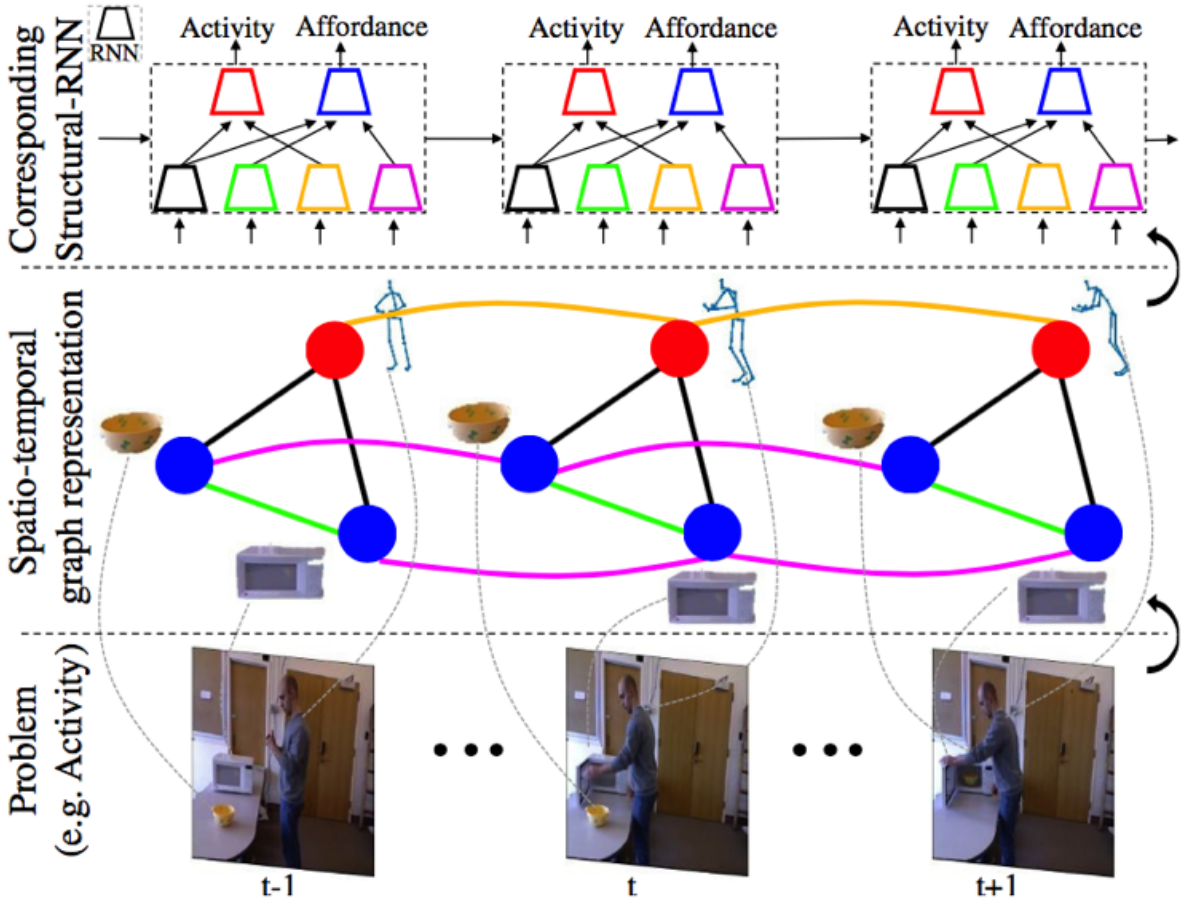

Figure 2.9 : From st-graph to sRNN for an example problem. (Bottom) Shows an example activity (human microwaving food). (Middle) st-graph capturing spatial and temporal interactions between the human and the objects. (Top) Schematic representation of the sRNN architecture automatically derived from st-graph. [4]

The first layer of the sRNN corresponds to the edges depicting, for example, the features encoded by the relative orientation between two object nodes. The node labels represent the human activity and object affordance respectively, encoded by the node features which represent the human and object poses.

Having gained an intuitive idea about how the sRNN is formed, we modify this to be used with the modified TAG, which we believe would provide better results over the state-of-the-art.

2.2.4 Random Forests

The softmax classifier only works well when there is one class output whose probability is close to one and the rest of the outputs are negligible. A standard softmax classifier is given as follows :

$$\sigma(z)_j = e^{z}_j / \sum e^{z}_k \text{for } j = 1, \ldots., K \text{ and } z = (z_1, \ldots., z_k) \in R^K \qquad (2.8)$$

There is no consideration given to the classes which are close contenders and fall short due to slight differences in output probabilities. Hence, instead of naively utilizing a traditional classifier, we trained a random forest to predict output classes based on the last dense layer feature vector.

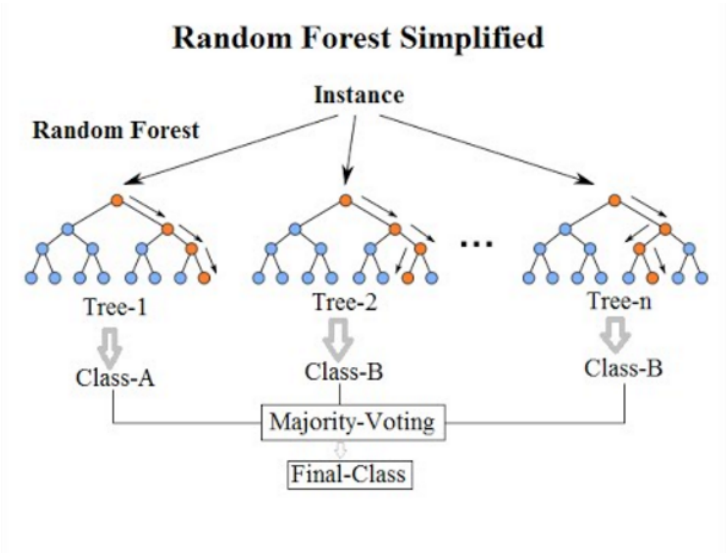

Figure 2.10 : Random Forest [18]

The random forest is a collection of decision trees and is an extremely convenient supervised learning methodology, traditionally used to prevent overfitting. A decision tree is a flow-chart type structure in which each internal node represents a test on an attribute. Based on the outcome, the example is classified into one of the branches of the node, where it is subject to further tests. Given below is an example of a decision tree which classifies 17 examples of 8 classes :

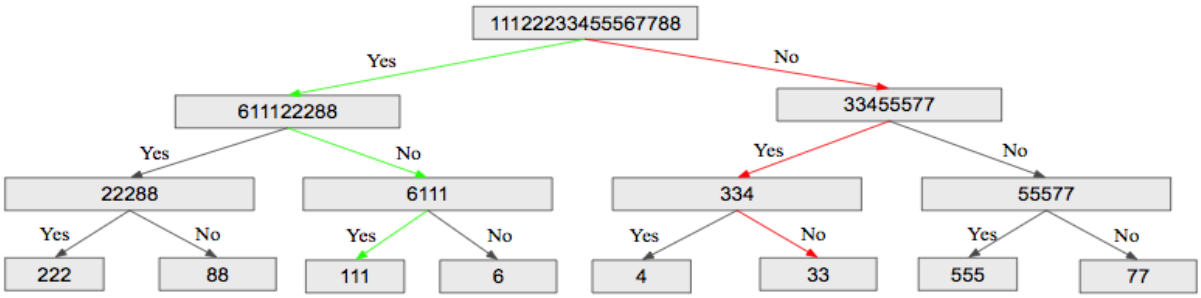

Figure 2.11 : Decision Tree example

Each of the boxes represent a test on the attributes of the examples. Such decision trees are collectively trained by randomly choosing a train-test split from a given set of examples and the trained decision trees each provide their individual output class results which are collected as votes. The class with the maximum number of votes is selected as the output class. Random forests prevent overfitting as they are formed by randomly selected decision trees.

In order to decide what tests are to be undertaken at each decision node, the information gain is calculated. The information gain is basically the measure of the knowledge while splitting a parent node into child nodes. It is given by :

Information Gain = Entropy of parent node - ∑(Weighted Entropy of children)

Here, entropy is the measure of randomness in a subset of data. The more the number of classes in a subset, the larger the entropy. A subset containing just one class of data is known as a pure subset. The formula for entropy for a node of a decision tree is given as follows :

$$Entropy = \sum -p_i * log\ (p_i) \tag{2.9}$$

Here, p_i is the probability of picking the i^{th} class in the subset. Thus pure subsets have zero entropy and decision trees are formed such that the child nodes have lower entropy than the parents.

2.2.5 Siamese Network

One of our main objectives is to achieve a high prediction accuracy using very small datasets. Similar problems are faced by tech firms implementing Facial Recognition softwares. A person's face has to be recognised from multiple angles, under different lighting, exposure conditions using just a few sample images. This problem is also called as the 'one-shot learning' problem[19].

In order to tackle this, researchers have used what has been known to be the Siamese network. In a Siamese network instead of using a explicit classifier to predict the output class, two neural networks outputs are compared, one based on the new example and one based on the ground truth image. The feature vector which is the closest to the ground truth feature vector is labelled similar to the ground truth class. A major advantage of this approach is that during training, negative examples can be paired with positive examples in the loss function. This forms what can be intuitively understood as the distance metric in the feature space. Typically loss is given by :

$$L = - y * log\, p + (1 - y) * log\, (1 - p) \tag{2.10}$$

Where y is the class label and p is the predicted probability.

Using a positive and a negative example, loss can be defined as :

$$L_{tot} = L_{+} + L_{-} \tag{2.11}$$

Thus, a large number of training pairs are generated. An illustrative diagram of a Siamese network is shown below :

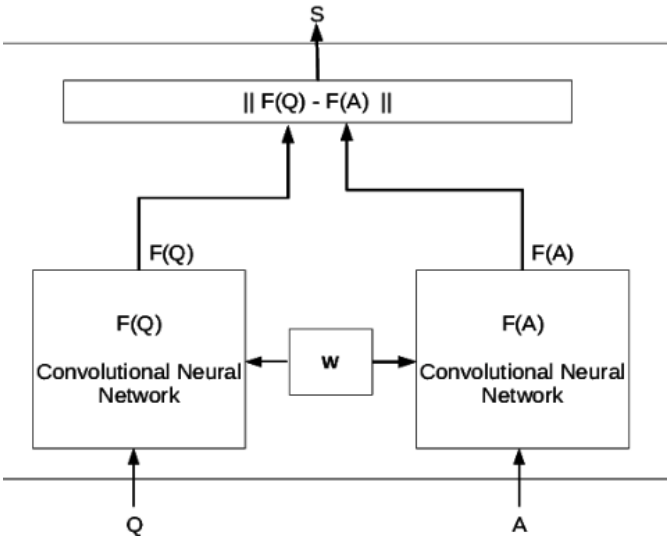

Figure 2.12 : Siamese Network [20]

The two networks have shared weights and hence there is very little computational complexity.

2.2.6 Regularisation

Mathematical modelling commonly faces the challenge of overcoming overfitting i.e. reducing variance without reducing test accuracies. Overfitting is high when the neurons in a neural network are able to model any non-linear complex function over the set of input train examples. Thus, instead of providing a reasonable approximation, the function is custom made to just fit over all the training data points. This is unnecessary and needs to be kept in check. Various methods have been suggested in order to overcome overfitting. One of them is Regularisation, which has proven to be highly effective. Regularisation poses as an upper limit to the maximum values that neurons can undertake and hence it ensures that the neurons do not overly tune themselves to learn misfitting examples and outliers. Thus new examples are more likely to be classified correctly.

We will be using the L2 regularisation[21] which is a basically a modification to the loss function and can be written as follows :

$$L_2 = \lambda * \sum \Theta_i^2 \quad (2.12)$$

Where $\Theta_i s$ are the weights to be regularised and λ is a hyperparameter set by us.

2.3 Skeletal Data

Skeletal data has often been used to increase the quality of the feature space. It is an efficient encoding for video data where human bodies are objects of prime importance. Skeletal data helps represent human bodies quite dependably with high quality standards as it is not affected by the brightness, background and other characteristics of the video.

A typical data skeleton from the SBUKI dataset consists of *13* joints which were used to form minimum bounding boxes to represent the *arms, legs, head* and *torso.*

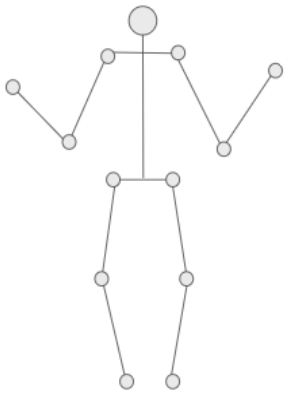

Figure 2.13 : Human Skeletal Data

We have used the open-source library *OpenPose* to extract skeletal information from custom videos taken at the IITG. The images below portray skeletal tracking for a *Handshake* activity. As one can see, sometimes under low lighting conditions, the non-extended objects are not mapped. But since the interacting parts such as hands are mapped and skeletal data from these parts are procured, the RNN is able to sufficiently learn features for classification. We have not been able to do this in real-time since the videos have been shot using a mobile camera at *720p, 30fps* with frames taken at suitable intervals, and the computation requires a GPU based PC.

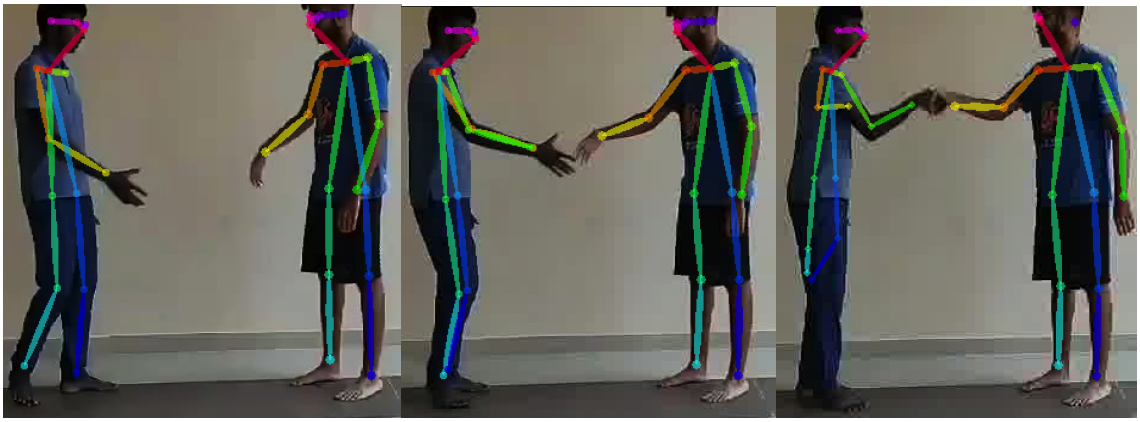

Figure 2.14 : Skeletal data extraction and mapping on video frames in IITG

Chapter 3

Working pipeline

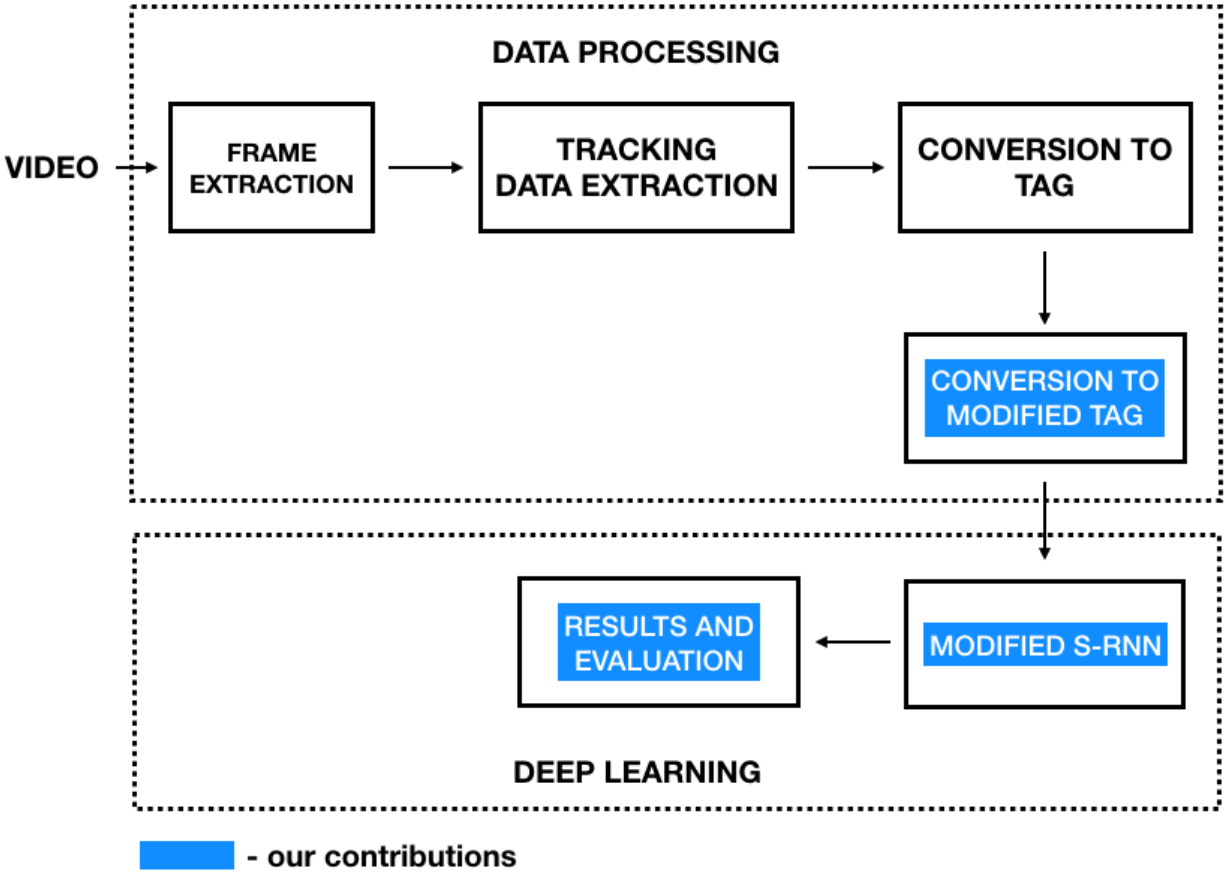

Figure 3.1 : Working pipeline for our Thesis Project

The above Fig. 16 summarises our project. Our contributions would mainly consist of modifying the TAG to have a higher and richer feature space for encoding information from the extracted video frames, modifying the sRNN to be compatible with our modified TAG and finally evaluating and optimizing the results. In addition, we would also be looking into providing insights into how our methodology can be extended to other domains of robotic intelligence.

The other boxes such as *Video, Frame extraction, Tracking Data Extraction and Conversion to TAG* has been done by third party applications. The video data is available online as a part of the *ME, SBUKI* and other datasets. Frame extraction is done based on frames of importance. For example, if a particular frame is taken as data input at time *t*, there is usually a few seconds buffer before the next video frame is taken as a part of the data in time step *t+1*. This is done to avoid redundant data input. Frames taken very close together in time encode almost the same data points and hence provide very little new information about the Activity. Tracking data can be extracted in the form of bounding box information. Conversion to TAG is done with the help of extended CORE9 as well as other methods provided in [3].

The results and evaluation section contains multiple subsections. This includes addition of Skeletal Data to improve the feature space, modifications to the sRNN based on Skeletal input, selecting suitable classifiers such as Siamese Networks and Random Forests to improve classification results. This section involved continuous integration and development. The hyperparameters at each level are optimized to provide the best results. This was done following common hyperparameter conventions, intuitive restructuring and meta analysis of results.

Chapter 4

Experiments and Evaluation

This section serves to guide the reader through our experimental protocols, methodologies and would hence provide details about each stage of our experimental analysis. It also contains details regarding all the additional contributions that we have done.

4.1 Data Extraction

We had based our initial experiments on two datasets, the *Mind's Eye* Dataset and the *SBU Kinect* Dataset.

The *Mind's Eye* Dataset is a result of *Prof. B. A. Draper's* efforts at the *Colorado State University's Department of Computer Science*. It consists of *11* activity classes and a total of *121* examples. Further details about the *Mind's Eye* Project can be found on their website[22].

The *SBUKI* dataset is a relatively larger dataset created at the *Stony Brook University* by *Yun et al*.[3] We have used a smaller subset of the dataset consisting of *8* activity classes and a total of *283* examples. For advanced models, we have used evaluations metrics on *SBUKI* only, because *ME* Dataset has very few examples for Deep Learning and the accuracies were saturated at unacceptable levels.

The *SBUKI* dataset also provides skeletal joint data and therefore we did not use *OpenPose* to extract the data from video frames. The following figure depicts images from the *SBUKI* dataset being converted to skeletal data and depth maps. More information is available on their website[23].

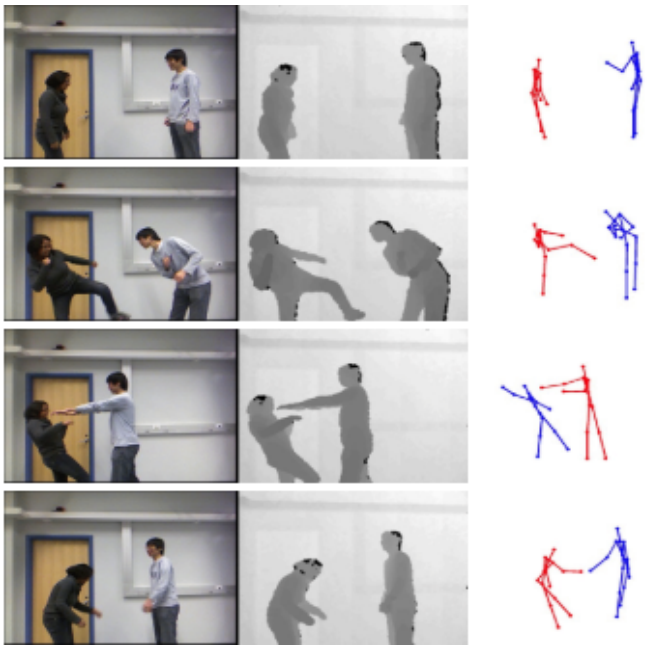

Figure 4.1 : Activity examples from SBUKI dataset along with Skeletal data extraction

We have processed these datasets and structured them to form TAGs as well as skeletal bounding box data. The TAG was formed using the extended CORE9 architecture on tracking data obtained from raw images in both the datasets.

4.2 The Preliminary Architecture

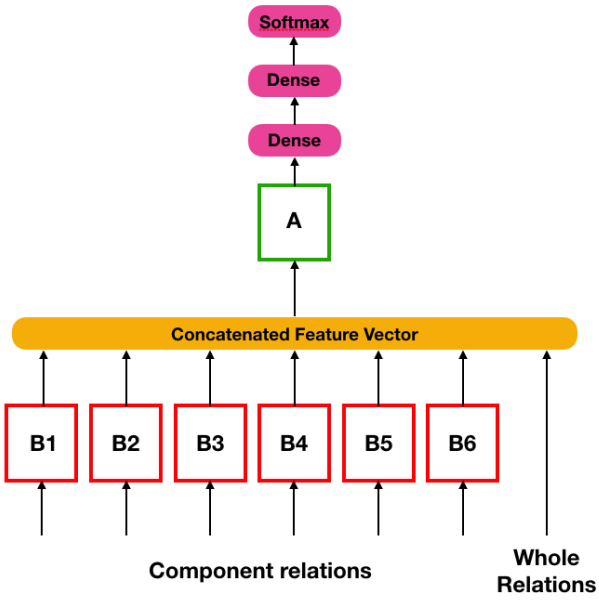

Figure 4.2 : Depicts the modified sRNN architecture which takes as input the TAG

The RNNs *B1* to *B6* encode component relations. The Bi^{th} RNN encodes the features corresponding to all the relations held by the i^{th} element of the extended object *b* in Fig 2. The features for one time step are fed sequentially into the RNN. Their outputs are then concatenated along with the whole relation obtain for that time step. Finally this new encoded feature vector is passed on to the RNN *A* which is responsible for associating an activity to the entire sequence of video frames from *t = 0* to *t*. This activity label is obtained through the softmax layer after the output of the RNN is passed through two dense fully connected layers with *64* and *32* nodes respectively.

An individual RNN consists an *LSTM* unit with *L2* regularization running with a lambda value of *0.2*. The dense layer consists of *64* nodes with the same *L2* regularization. Following this, we have added batch normalization as it is a well accepted methodology to induce stability into neural networks. Batch normalization reduces the mean of the inputs to the next layer zero and equates the variance to unity[24]. *ReLU* activation has been used as it removes the vanishing gradient problem[25]. Finally a *softmax* layer has been used to get the output class of the activity.

All the experiments were performed for *5* iterations, with randomly selected test and train splits and the results after *100* epochs were averaged. It was ensured that there were reasonable number of training examples from each activity class.

4.3 Preliminary Analysis

The results from two of the datasets have been tabulated in the following tables :

ME Dataset		**SBUKI Dataset**	
Activity	**Accuracy**	**Activity**	**Accuracy**
Approach	33.33%	Approaching	30.29%
Carry	93.33%	Departing	31.42%
Catch	93.33%	Pushing	48.49%
Collide	40.00%	Kicking	66.94%
Drop	60.00%	Punching	63.67%
Follow	53.33%	Exchanging	77.43%
Hold	81.67%	Hugging	49.91%

Kick	55.00%	Handshaking	24.63%
Pick Up	60.00%	**Overall Accuracy : 44.2 %**	
Push	73.33%		
Throw	56.66%		
Overall Accuracy : 68 %			

Primary Results

We have not used the *URFD, UTI* dataset as they have very few activity classes. An intuitive guesstimate suggests that it would either overfit the dataset or never really learn the features as the amount of data is extremely low. For a deep learning model to work, it is generally recommended to have a large number of positive and negative examples. Hence we abandon these datasets for now.

Analysing the results, we can see that the model is *severely underfitting* for many activities. For comparison, we have the following tables showcasing overall accuracies from the state-of-art :

	Datasets	
Methods	**ME**	**SBUKI**
TAG + Skeletal Information based Kernel	73.63 %	81.91 %
TAG + Interestingness based Kernel	65.45 %	78.01 %

Comparison of classification accuracies with other approaches in literature

The results do not reach a level of expectation from deep learning. However, it is to be noted that we have very few positive examples for each class in the datasets. In

fact, our aim is to make deep learning work for such datasets and this is part of ongoing research.

We have not calculated *Precision, Recall* and *F1 score* for now as the accuracies are too low to compare anyways. However, we can see that certain activities in the *ME* dataset such as *Carry, Catch* and *Hold* have extremely good accuracies. In *SBUKI* dataset, *Kicking*, which is one of the more difficult accuracies to classify(as most of the nodes of the TAG are *NULL* due to the fact that only the legs enjoy rich relationship labels provided by the extended CORE9), has a high accuracy. This means that the TAG does have enough information in it for the sRNN to encode and classify for these activities. Also it provides us a reassurance that the sRNN is indeed encoding the features as per our expectations.

In order to tackle this, we have added skeletal data.

4.4 The intermediate architecture

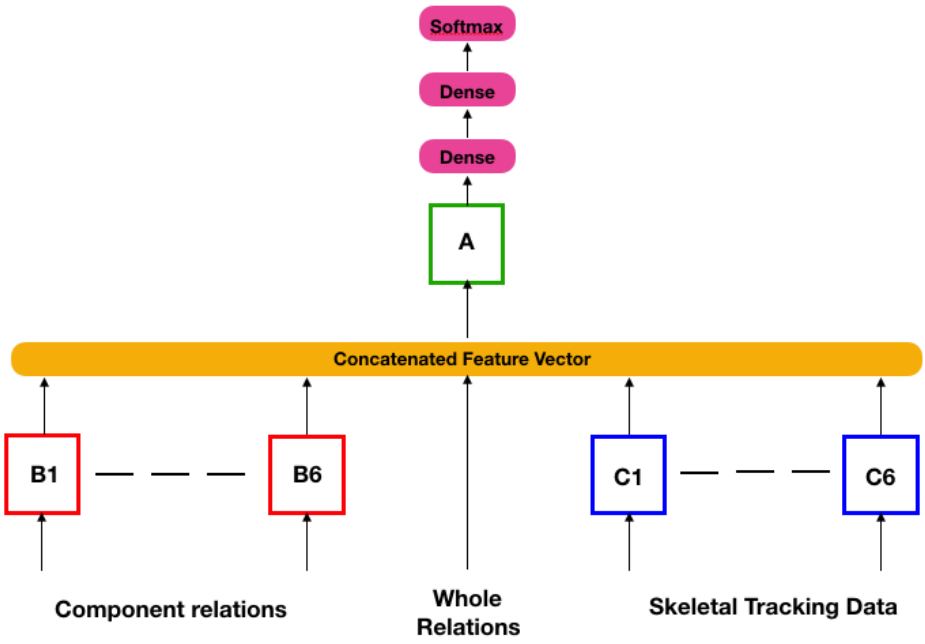

Figure 4.3 : Depicts the intermediate modified sRNN architecture which takes as input the TAG as well as skeletal data

The Fig. portrays our first modification to the preliminary architecture. RNNS *C1 ... C6* take as input the skeletal bounding box data of *hands, legs, torso* and *head*. Thus the concatenated feature vector is formed using both the TAG as well as the skeletal data. The regularisation parameters and the number of nodes are similar to the first architecture. The Skeletal RNNs contain the same number of nodes as the RNNs taking in inputs from the TAG.

4.5 Intermediate Analysis

The results from the above architecture have been summarised in the following table :

SBUKI Dataset	
Activity	**Accuracy**
Approaching	93.29%
Departing	83.42%
Pushing	50.49%
Kicking	83.94%
Punching	23.67%
Exchanging	56.43%
Hugging	40.91%
Handshaking	34.63%
Overall Accuracy : 58.33 %	

Intermediate Results

We can see that the accuracies have certainly increased and the regularisation has also provided additional impetus to the classification technique. However there is still a large room for improvement. In order to make this physically deployable, we need an accuracy of close to *80%* in our test sets.

We realised that although the final dense layer is feature rich, the *softmax* classifier is unable to heuristically classify the activities. Hence we will be branching into two alternative classifying strategies. The Random Forest and the Siamese Network.

4.6 The final architecture

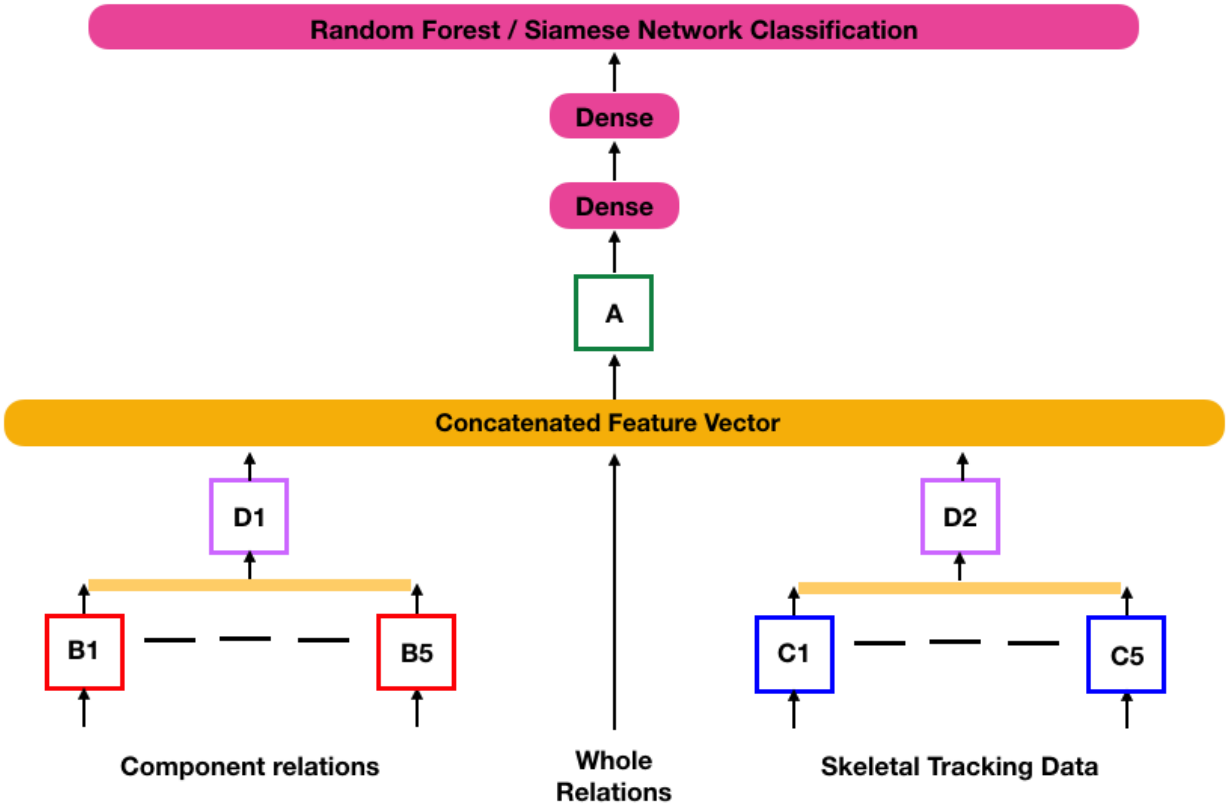

Figure 4.4 : Depicts the final modified sRNN architecture with Random Forest/Siamese Network

We removed the tracking data from the *torso* and reduced the number of RNNs in the initial layer. We realised that the accuracies do not change in doing so. This also preserves precious computation time. Instead we first use two RNNs *D1, D2* to encode the outputs from the first layer coming from the TAG and the skeletal data respectively. Their outputs are then passed on to form a concatenated feature vector with the whole relations from the TAG. Finally, we removed the softmax and first added a Random Forest Classifier which we trained. Then we used the Siamese Network approach to classify the outputs. All the regularisation parameters remained

the same. The new RNNs *D1* and *D2* have *64* neurons each. The number of trees to be used in the Random Forest were set to *100*.

4.7 Final Analysis

The results from the final architecture are summarised as follows :

SBUKI Dataset		
Activity	**Random Forest Accuracy**	**Siamese Network Accuracy**
Approaching	90.29%	89.11%
Departing	95.42%	93.88%
Pushing	88.49%	98.15%
Kicking	84.94%	84.94%
Punching	66.67%	95.17%
Exchanging	66.67%	67.43%
Hugging	49.91%	51.55%
Handshaking	69.63%	70.80%
Overall Accuracy	70.2%	77.65%

Final Results

The Siamese Network manages to give us very high accuracies for the minimal amount of data being used for training which elucidates its use in one-shot learning. The Random Forest also gives a slight improvement in results but the accuracies are not as high as the Siamese Network.

Although the accuracy from the Random Forest based architecture is quite high we need to keep in mind that the state-of-the-art neural networks work well on multiple datasets. In our approach, we cannot use extremely small datasets nor can we generalise our results to other datasets for now. This is one limitation to be kept in mind.

4.8 Local Testing

We used our final architecture on a few video examples which we captured at IITG. The labelling results have been shown in the images below :

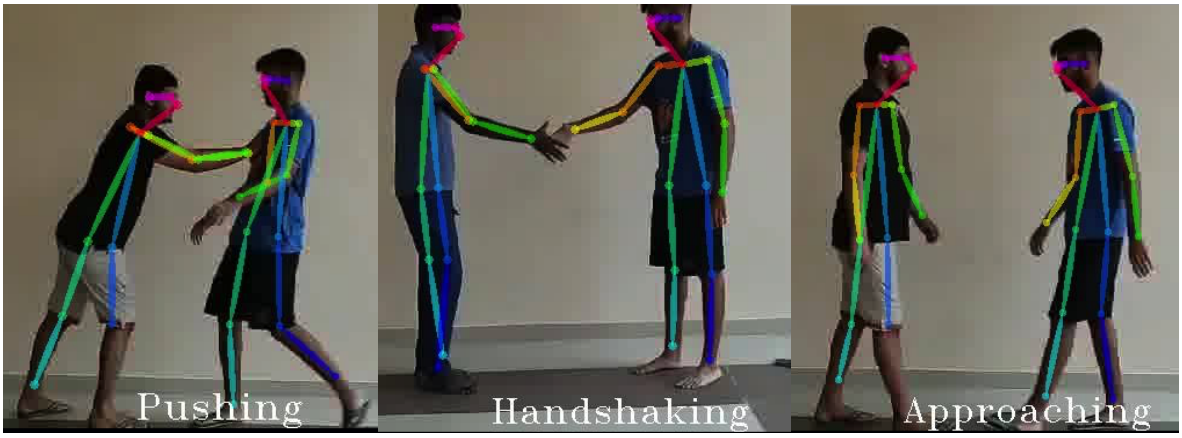

Figure 4.5 : Positively Labelled Examples at IITG

The above figures depict correctly labelled activities. We have noticed that a large percentage of the videos were labelled correctly. However there were a few instances of mislabelling as shown below :

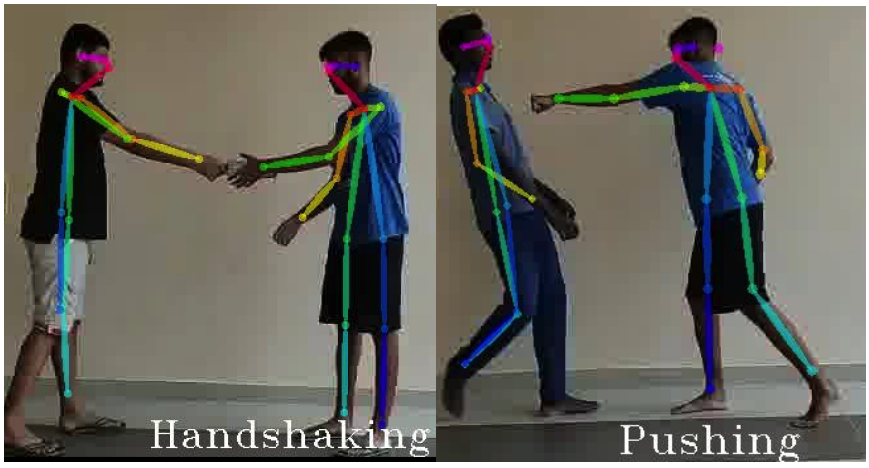

Figure 4.6 : Negatively Labelled Examples at IITG

Intuitively, the mislabeling is understandable as it is difficult to differentiate between an Exchanging and a Handshaking activity. Further tuning is necessary in order to

avoid this. Similarly, a pushing and a punching activity has similar skeletal data before the onset of physical contact in punching. Therefore it is understandable why the classification is wrongly done.

Chapter 5

Conclusion

To conclude, in accordance to our intuition, the study of the use of sRNNs for Human Activity Recognition seems to be a promising field.

- TAGs are able to scale and represent the humans in the video frames to a considerable extent, providing an alternative to st-graphs. Accordingly, some of our results are even better than the state of the art.
- A large fraction of the results consisting of just TAGs as inputs were unsatisfactory. However, this was expected as the TAG was designed to work with Support Vector Machines.
- In order to achieve better accuracies, we added skeletal data as an input to the sRNN along with the TAG. This increased the classification accuracy with slight increase in computational complexity.
- Finally, it was noted that introduction of alternative classification strategies such as Random Forests and Siamese Networks provide a huge boost to the accuracies. The best accuracies were obtained using Siamese Networks.

An easy way to improve the accuracy of our results would have been to acquire more data. However our aim was to construct a highly efficient and scalable reproduction of the TAG with a larger and improved feature space which can model each component of the extend object as well as its relations with other components and hence nullify the need for larger datasets.

Deep learning is an incredibly powerful, simple and effective tool, with the only downside of requiring large amounts of data. However, with this thesis, we can positively conclude that structuring neural networks serves as an alternative to larger datasets and thus the field holds a promising scope for future studies.

Future Work

After obtaining the results, we realise that although some of our results are challenging the current state-of-the-art, we need to optimize and refactor our method in order to create a more generalised scheme which can be applied to all datasets to procure satisfactory results. Thus, we strive to find the best combination of sRNN architecture based on the modified TAG which would hopefully provide better results than the state of the art.

An end-to-end deep learning project can be thought of as an old television with multiple knobs for tuning. It is very difficult to get the right combination of tuning parameters to optimize the accuracy of the task in hand. However, having known that TAGs do work successfully with SVMs and sRNNs have achieved high quality results over other datasets, we believe that with the right set of features, we can make this combination work together. Although increasing the size of the dataset would definitely provide us better results, one would want to find a methodology to tune a neural network with small, to-the-point data sets.

- The current TAG is formed using just topological, direction and distance relations. Various other features such as optical flow and much more can be introduced into the graph's feature space. Currently, the activities seem too complex to be encoded into the above mentioned labels. One needs to add new feature parameters to this feature vector in the hope of encoding each of the complex tasks in a unique but simple manner.
- Another field which might be open to exploration is the modification of the proposed sRNN architecture. However we need to keep in mind that although TAGs break down humans into extended objects, it is computationally expensive to allot one RNN to each edge relation as proposed by the original authors of the sRNN architecture. They were also able to utilise the advantage of factor sharing which is currently not being implemented in this project. These fields are nascent and unexplored.
- With the addition of skeletal data, the results have improved quite drastically. The Siamese Network has provided us with the best results so far in combination with the skeletal data. Additional tuning of the Siamese network

with better similarity comparators might lead to a steep rise in classification accuracies. A Siamese Network in itself contains multiple methodologies for tuning and optimization of hyper-parameters. Use of the triplet loss function might provide an additional impetus to the learning curve.

- We have also not added any data manipulation techniques to help augment and increase the data input. This might not only lead to additional increase in the training set using the same number of examples, but also might help prevent overfitting if suitable generalisation algorithms are used. However it is to be kept in mind that this would increase the training time.

A link to our github repository containing well-documented code has been provided in [26] for anyone who wishes to recreate our experiments and work on it.

References

[1] Raleigh, H. (2018, November 5). From thefederalist.com: *https://thefederalist.com/2018/11/05/china-rolling-massive-population-surveillance-system-world/*
[2] Ma, A. (2018, April 29). From businessinsider.in: *https://www.businessinsider.in/China-is-building-a-vast-civilian-surveillance-n%20etwork-here-are-10-ways-it-could-be-feeding-its-creepy-social-credit-system/articleshow/63959324.cms*
[3] Kalita, S. (2018). *Temporal Activity Graph Based Human Activity Classification*, In *Proceedings of the 11th Indian Conference on Computer Vision, Graphics and Image Processing 2018*
[4] Jain, A., Zamir, A. R., Savarese, S., & Saxena, A. (2016). Structural-RNN: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5308-5317).
[5] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.
[6] Pham, H. H., Khoudour, L., Crouzil, A., Zegers, P., & Velastin, S. A. (2018). Exploiting deep residual networks for human action recognition from skeletal data. *Computer Vision and Image Understanding*, *170*, 51-66.
[7] Bickel, P., Diggle, P., Fienberg, S., Gather, U., Olkin, I., & Zeger, S. (2009). Springer Series in Statistics.
[8] Koch, G., Zemel, R., & Salakhutdinov, R. (2015, July). Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop* (Vol. 2).
[9] Jain, A., Singh, A., Koppula, H. S., Soh, S., & Saxena, A. (2016, May). Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*(pp. 3118-3125). IEEE.
[10] Ravi, D., Wong, C., Lo, B., & Yang, G. Z. (2016, June). Deep learning for human activity recognition: A resource efficient implementation on low-power devices. In *2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN)* (pp. 71-76). IEEE.

[11] Kalita, S., Karmakar, A., & Hazarika, S. M. (2018). Efficient extraction of spatial relations for extended objects vis-à-vis human activity recognition in video. *Applied Intelligence*, *48*(1), 204-219.
[12] Cohn, A. G., Renz, J., & Sridhar, M. (2012, May). Thinking inside the box: A comprehensive spatial representation for video analysis. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.
[13] Nielsen, M. A. (2015). *Neural networks and deep learning*(Vol. 25). San Francisco, CA, USA:: Determination press.
[14] Karpathy, A. (2019, April 25) From skymind.ai:]
https://skymind.ai/wiki/neural-network
[15] Olah, C. (2015, August 27) From colah.github.io:
https://colah.github.io/posts/2015-08-Understanding-LSTMs/
[16] Britz, D. (2015, October 8) From wildml.com:
http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/
[17] Chen, G. (2016). A gentle tutorial of recurrent neural network with error backpropagation. *arXiv preprint arXiv:1610.02583*.
[18] Koehrsen, W. (2017, December 27) From medium.com:
https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d
[19] Koch, G., Zemel, R., & Salakhutdinov, R. (2015, July). Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop* (Vol. 2).
[20] Das, A., Yenala, H., Chinnakotla, M., & Shrivastava, M. (2016). Together we stand: Siamese networks for similar question retrieval. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Vol. 1, pp. 378-387).
[21] Ng, A. Y. (2004, July). Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning* (p. 78). ACM.
[22] Draper, B. (2016) From cs.colorado.edu:
https://www.cs.colostate.edu/~draper/MindsEye.php
[23] Yun, K., Honorio, J., Chattopadhyay, D., Berg, T. L., & Samaras, D. (2012, June). Two-person interaction detection using body-pose features and multiple

instance learning. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (pp. 28-35). IEEE.
[24] Normalization, B. (2015). Accelerating deep network training by reducing internal covariate shift. *CoRR.–2015.–Vol. abs/1502.03167.–URL: http://arxiv. org/abs/1502.03167*.
[25] Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines vinod nair.
[26] Kamath, S., Agarwal, D. From github.com: *https://github.com/sudz123/HAR.ai*