THESIS

LOOKING UNDER THE HOOD: VISUALIZING WHAT LSTMS LEARN

Submitted by

Dhruva Patil

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2019

Master's Committee:

Advisor: Dr. Bruce Draper

Dr. J. Ross Beveridge
Dr. Anthony Maciejewski

ABSTRACT

LOOKING UNDER THE HOOD: VISUALIZING WHAT LSTMS LEARN

Recurrent Neural Networks (RNNs) such as Long Short Term Memory (LSTM) and Gated Recurrent Units (GRUs) have been successful in many applications involving sequential data. The success of these models lies in the complex feature representations they learn from the training data. One criteria to trust the model is its validation accuracy. However, this can lead to surprises when the network learns properties of the input data, different from what the designer intended and/or the user assumes. As a result, we lack confidence in even high-performing networks when they are deployed in applications with novel input data, or where the cost of failure is very high. Thus understanding and visualizing what recurrent networks have learned becomes essential.

Visualizations of RNN models are better established in the field of natural language processing than in computer vision. This work presents visualizations of what recurrent networks, particularly LSTMs, learn in the domain of action recognition, where the inputs are sequences of 3D human poses, or skeletons. The goal of the thesis is to understand the properties learned by a network with regard to an input action sequence, and how it will generalize to novel inputs.

This thesis presents two methods for visualizing concepts learned by RNNs in the domain of action recognition, providing an independent insight into the working of the recognition model. The first visualization method shows the sensitivity of joints over time in a video sequence. The second visualization method generates synthetic videos that maximize the responses of a class label or hidden unit within a set of known anatomical constraints. These techniques are combined in a visualization tool called *SkeletonVis* to help developers and users gain insights into models embedded in RNNs for action recognition. We present case studies on NTU-RGBD, a popular data set for action recognition, to reveal properties learnt by a trained LSTM network.

ACKNOWLEDGEMENTS

This thesis is possible due to the support of multiple people in my life. I would like to thank my advisors Dr. Bruce A. Draper and Dr. Ross Beveridge for their constant encouragement, patience and guidance in my research, giving me the freedom to explore the field of Computer Vision. I would also like to thank my committee member, Dr. Anthony Maciejewski for introducing me to the world of Robotics.

I would like to thank my family for believing in me, guiding and supporting me. Thanks to you Dada, for patiently listening to my work, and critiquing it when necessary. A special thanks to Dr. Pradyumna Narayana, who was always there to help me as a friend, an elder brother and a mentor. I am grateful for all the discussions I had with my fellow graduate students in the Computer Vision lab. Those discussions helped me get a different perspective about the problems I was trying to solve. I am grateful to the Department of Computer Science at CSU for giving me an opportunity to work on Communicating with Computers program. This work was supported by the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO) under contract W911NF-15-1-0459 at Colorado State University.

DEDICATION

I would like to dedicate this thesis to my family, friends and my mentors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

Recurrent Neural Networks (RNNs) such as Long Short Term Memory (LSTM) networks [1], Gated Recurrent Units (GRUs [2]) have been successful in many applications involving sequential data. Examples can be found in text classification [3], image and video captioning [4, 5], speech recognition [6, 7], and action and gesture recognition [8–10]. The success of these deep learning models lies in the complex feature representations they learn from the training data and encoding the temporal information. Unfortunately, the practice of deep learning is ahead of the theory. The feature representation remains a black box for the developer, who can only trust the model from the accuracy with which it labels the validation data. There is no way of summarizing what properties of the training data the network has learned with respect to individual classes in the data. This can lead to surprises when the network learns properties of the input data other than what the designer intended and/or the user assumes. As a result, we lack confidence in even high-performing networks when they are deployed in applications where the input might differ from the training data, or where the cost of failure is very high. Thus understanding and visualizing what recurrent networks have learned becomes essential.

This thesis presents visualizations of what recurrent networks, particularly LSTMs, learn in the domain of action recognition. The input to the network is a sequence of 3D human poses of a particular action. The goal of the thesis is to understand the properties learned by a network with regard to an input action sequence, and how it will generalize to novel inputs.

1.2 Current Techniques and Shortcomings

The Computer vision domain has well developed methods of visualizing layer-wise learned features for convolutional neural networks. See [11] for an up-to-date survey and [12] for an

interactive summary of visualization techniques in CNNs. However, those techniques do not apply to recurrent neural networks.

Recently, researchers studying RNNs have introduced techniques that probe what is being learned from data by changing the underlying network architecture. One such technique involves the use of attention mechanisms to study specific properties of the problem with respect to the input. These attention mechanisms have been used successfully in machine translation and image captioning [13–15]. However, this approach involves changing the original model architecture to study specific properties of the input and the problem domain. While this helps us understand the data-driven properties of the task at hand, the model-driven properties remain under explored.

Visualizations of RNN models are better established in the field of natural language processing than computer vision. Karpathy and Li presented a static visualization exploring properties of hidden units and important words in text respectively. Karpathy showed the existence of cells that keep track of long range properties like line lengths, quotes and parentheses. Li highlighted the important words in text for different models using a gradient based saliency approach. Similarly, Strobelt et al [16] studied the response of a recurrent network model to a structural pattern of words and local state changes. Y. Ming et al [17] provided a glyph based visualization displaying the association between words and different hidden state state units. Both these approaches differ from the previously stated techniques as they present a dynamic visualization of the hidden units with the input data. However, the input to all the above methods is a single character or word, embedded into a vector. This is significantly different from the input to LSTMs in an action recognition system.

The input to the LSTM in an action recognition system is a multidimensional skeleton pose sequence. Interpreting how long distance relationships within a video are modeled becomes particularly difficult. Two main approaches attempt to understand key properties learned by the LSTM model. Song et al in [9] propose a two step attention mechanism to study the spatial and temporal information in the action sequence. However, as with the attention approach, the original model architecture is altered. Zhu et al in [10] study the co-occurrence of joints in an action sequence.

While the original model is unchanged, the effect of input on the output hidden states is not explored. We aim to interpret the hidden states of a one layer LSTM model directly.

Currently, visualizations of LSTMs in skeleton based action recognition use heat maps to identify important joints and their correlations with other joints in an action. In [9], the authors use markers of different sizes and colors projected on skeletons to highlight weights of joints. However, their approach does not produce a tool for easily visualizing the skeleton inputs to the RNN model. We present a tool called *SkeletonVis* to visualize trained skeleton-based action recognition networks.

1.3 Summary of Research

This thesis presents two methods for visualizing concepts learned by RNNs in the domain of activity recognition. Activity recognition has the advantage that the inputs are sequences of 3D human poses, or skeletons. This provides a framework for visualizing results and anatomical constraints for generating synthetic inputs. Both the visualizations provide an independent insight into the working of the recognition model.

The first visualization method shows a color based importance of joints in an action sequence, also called as *sensitivity*. This approach extends the work by Li et al [18] to find important words in a text. Sensitivity is the normalized partial derivative of an output signal with respect to a given joint, where the output may either be a class label or the output of a specific hidden unit. Figure 5.2 shows class based sensitivity plot for frames for a throw sequence.

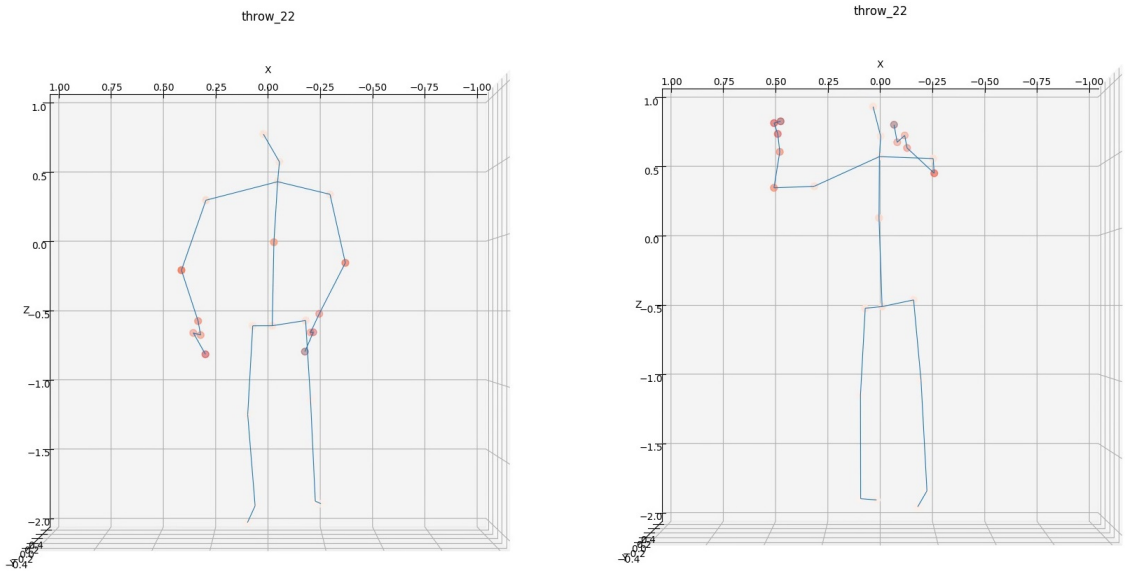

Figure 1.1: Sensitivity plot for frames of throw action. As seen in the two frames, in addition to the hands having high sensitivity, the model is sensitive to the spine joint.

We observe that the *throw* action is sensitive to the positions and motions of the arms, which is not a surprise, but is also sensitive to the upward motion of the spine. In essence, the LSTM has learned that throwing requires an upward movement of the entire body, which otherwise the user may not know.

The second visualization method generates synthetic videos that maximize the responses of a class label or hidden unit within a set of known anatomical constraints. This yields different insights from the first method. Figure 1.2 shows the original and the updated skeleton within anatomical constraints for frames for a throw sequence.

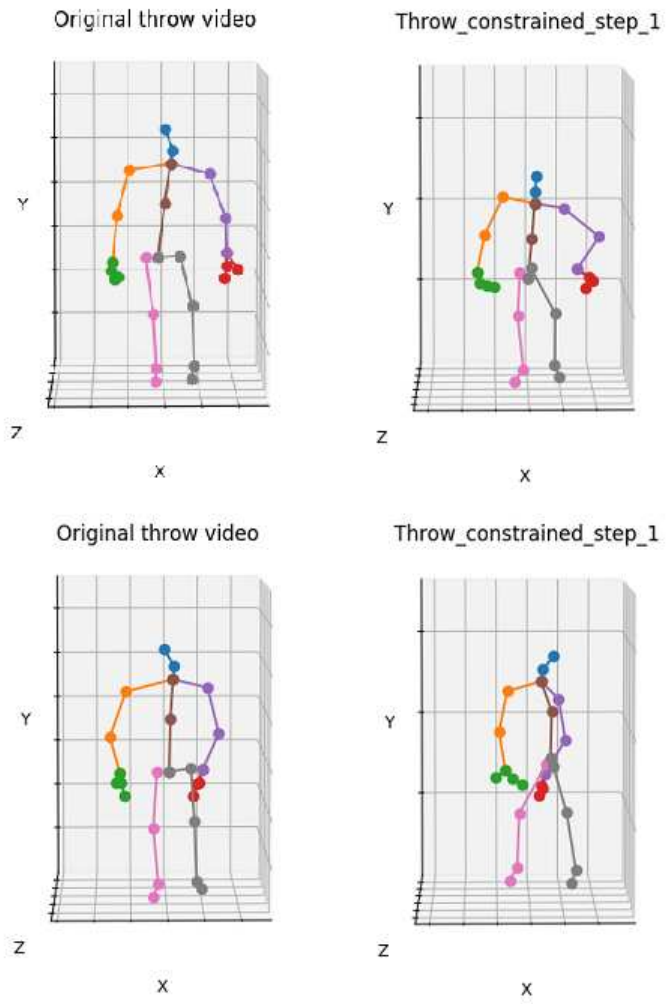

Figure 1.2: Progressive frames of a throw sequence in the original video and updated skeletons by the model within the anatomical constraints. the model favors the position of the subject to be as low to the ground as possible.

For example, the response of one hidden unit to throws is maximized when the subject begins as low to the ground as possible. The goal of such visualizations is to show users what the system has learned, and therefore how it might respond to novel inputs.

The visualization techniques presented in this paper are presented in the context of LSTMs, but can be applied to most recurrent networks, including Gated Recurrent Unit networks (GRUs [2])

and Echo State Networks (ESNs [19, 20]). For LSTMs, however, we consolidate the two visualization techniques into a tool called *SkeletonVis*.

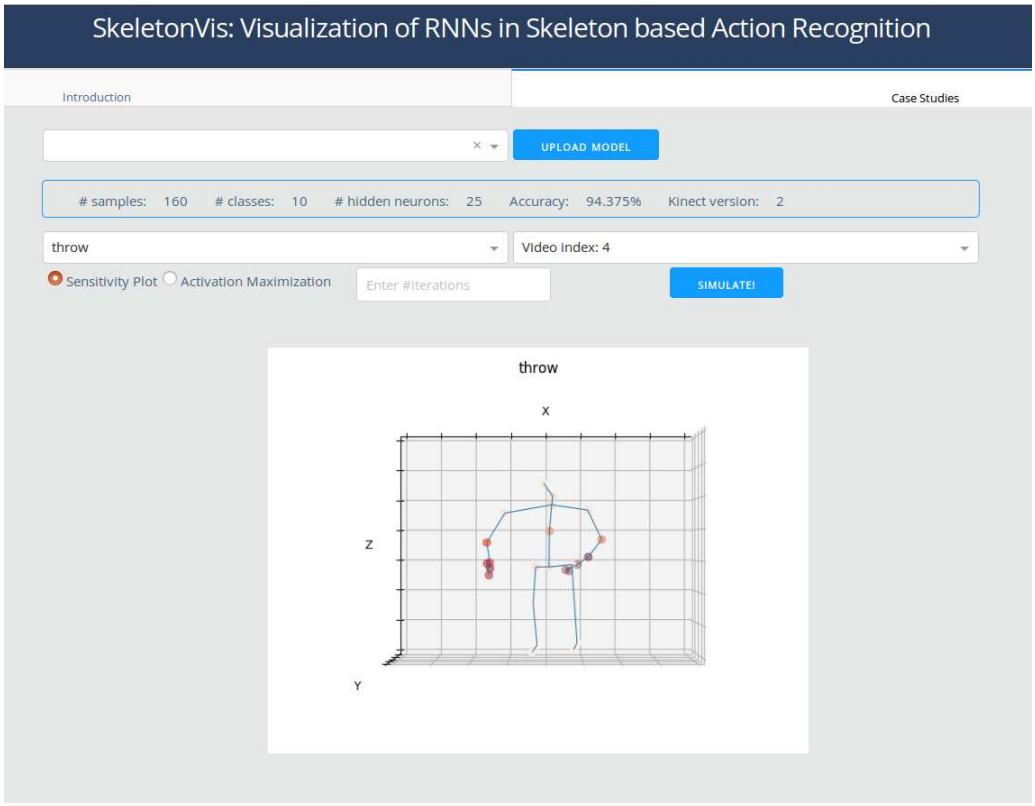

Figure 1.3: The SkeletonVis tool, as it appears to LSTM developers and users

Figure 1.3 shows a snippet of the working of SkeletonVis, as it appears to the users. This tool can be used over the web to view LSTM networks we have trained on the NTU activity data set, or downloaded and applied to LSTMs trained by users on data sets of their choice.

We also present several case studies of using SkeletonVis. The case studies show how the sensitivity and response maximization visualization techniques can be used to reveal properties of trained LSTMs.

1.3.1 Contributions

This thesis presents two methods to visualize feature representation in recurrent networks in the domain of action recognition using skeleton data. The main contributions of this thesis are:

1. A technique for visualizing the sensitivity of an LSTM class label or hidden unit response to specific joints in pose data.

2. A technique for visualizing the (synthetic) video that elicits the maximum response by a class label or hidden unit.

3. A software tool called *SkeletonVis* for visualizing LSTMs using the techniques above

4. Case studies of using SkeletonVis to probe the properties of trained networks.

1.3.2 Roadmap

The rest of the document is structured as follows: Chapter 2 discusses the visualization techniques used in convolutional neural networks and recurrent networks, anatomical constraints employed to generate synthetic skeletons and a greater discussion of related research in the field of understanding and visualizing deep learning models. Chapter 3 briefly introduces the working of recurrent networks and LSTMs in particular. Gradient based saliency and activation maximization are the two visualization approaches discussed in Chapter 4. It also describes the design and working of SkeletonVis, the visualization tool. Chapter 5 explains the case studies done on NTU RGB-D dataset, a well known dataset for action recognition. Chapter 6 concludes the thesis with a discussion and possible avenues for future research.

Chapter 2

Prior Literature

This chapter aims to provide the reader with the necessary background to understand the following chapters. Readers may want to skip to subsections pertaining to their interests. Section 2.1 explains the visualization techniques employed in understanding hidden layer mechanisms in convolutional neural networks. Section 2.2 explains the visualization techniques in recurrent neural networks, particularly in the field of natural language processing and skeleton based action recognition. The chapter concludes with section 2.3 that explains the different approaches to generate synthetic skeletons conforming to anatomical constraints of the human body.

2.1 Visualization techniques in CNN

The computer vision literature includes many methods for visualizing features learned by convolutional neural networks. There are some of well developed techniques at interpreting hidden layers and internal neurons and their interactions in CNN architectures. Of particular importance to this work, Simonyan and Zisserman use saliency maps to find class-specific properties in images [21]. Mahendran and Vedaldi in [22] introduce activation maximization to search for image patterns that maximize the neuron activations of specific layers. This thesis builds on the concepts of gradient based saliency and activation maximization. [11] and [12] present an updated and interactive summary of visualization techniques in CNNs. However, the techniques referenced above do not apply to recurrent networks; they are limited to feed-forward convolutional networks. We extend their techniques to recurrent networks.

2.2 Visualization techniques in RNNs

An advantage of visualizing features of deep learning models is the improvement in the model architectures for those problems. Researchers trying to understand the internal workings of RNNs have introduced techniques such as attention mechanism to study specific properties of the problem

with respect to the input. This approach has been used successfully in machine translation and image captioning [13–15]. However, attention mechanism modifies the original model architecture to study specific properties of the input and the problem domain. This helps understand the data-driven properties of the given task, but the model-driven properties remain under explored.

In the following sections, Section 2.2.1 presents tools for visualizations in the field of natural language processing. Section 2.2.2 explains the different methods that aid in understanding the data properties in skeleton based action recognition.

2.2.1 Visualizations in Natural Language Processing

Visualizations of RNN models are better established in the field of natural language processing than computer vision. [23] showed cells that keep track of long range text properties like line lengths, quotes and parentheses. [18] finds important words in text classification and auto encoders for different models and datasets using a gradient based saliency approach. However, this was a static visualization providing only an overall analysis. Strobelt et al developed LSTMVis [16] , a tool to analyze specific hidden state properties for a structural pattern of words. This tool gave the user the flexibility to study local state changes, but lacked scalability of the hidden state dimensions. More recently,Y. Ming et al [17] developed RNNVis to study expected responses of hidden state units to words. This technique provided a glyph based visualization of RNNs by displaying associated hidden state units and words. However, the input to all the above methods is a single character or word, embedded into a vector. This is significantly different from the input to LSTMs in an action recognition system.

2.2.2 Visualizations in skeleton-based action recognition

Skeleton based action recognition system takes a multidimensional 3D skeleton pose over time as input.Due to this, interpretation of long distance relationships within a video becomes particularly difficult. [9] designed a two step attention mechanism to study the spatial and temporal information in the action sequence. The spatial attention module focuses more on the content dependent relevant joints (foot, elbow and hand for a kicking action), while the temporal attention

module looks at time frames as the sequence progresses into the given action. This technique effectively understands key properties learned by the LSTM model, but as with the attention approach, the original model is altered.

[10] proposed the study of co-occurrence of joints in an action sequence. Though this approach shows correlations of joints without altering the model, the effect of the input on the output hidden states of LSTM is not explored. This thesis aims to interpret hidden states of a one layer RNN model directly.

Currently, visualizations of LSTMs in skeleton based action recognition use heat maps to identify important joints and their correlations with other joints in an action. In [9], the authors use markers of different sizes and colors projected on skeletons to highlight weights of joints. However, their approach does not produce a tool for easily visualizing the skeleton inputs to the RNN model. We present a tool called *SkeletonVis* to visualize trained skeleton-based action recognition networks.

2.3 Anatomical constraints for synthetic skeleton poses

Skeleton extracted by 3D pose estimation techniques from depth and/or RGB data are easily prone to distortions and noise both at the source (e.g Kinect sensor) as well as due to transformations by different models. To make the skeleton pose adhere to anatomical conditions, various constraints are imposed on it. This idea finds applications in 3D human pose estimation, generating animated human poses, or denoising the data obtained from source. Tripathy et al [24] propose a constrained Kalman filter to denoise joint coordinates obtained from the Kinect sensor. Our approach integrates the bone length constraints proposed in this paper. Dabral et al [25] models the joint angle limits and the bone length limits as a loss function that strongly penalizes joints that deviate from valid angular limits. The formulation of joint angle limits stated by the authors is used in our approach. [26] discriminate joint types (ball joints vs. hinge joints) and impose joint limits accordingly, while [27] formulate a prior to eliminate invalid poses by pose-conditioned joint angle limits. Such constraints could be added to our techniques in the future.

Chapter 3

Background

The visualization techniques presented in the next section require the reader to be familiar with recurrent networks in general and LSTMs [1] in particular. Readers who are already familiar with the basic equations in LSTMs may choose to skip this section.

Recurrent Neural Networks(RNN) are a family of neural networks used to model long temporal sequences of data. The output response $\mathbf{h_t}$ at any time instance t, is determined by the current input to the system $\mathbf{x_t}$ and the output of the previous time step $\mathbf{h_{t-1}}$. Mathematically, a vanilla RNN may be written as:

$$h_t = tanh(W(x_t; h_{t-1})) \tag{3.1}$$

where $\mathbf{W}$ is the weight matrix and $x_t; h_{t-1}$ is the concatenation of $\mathbf{x_t}$ and $\mathbf{h_{t-1}}$. The *tanh* non linear activation restricts the value of $\mathbf{h_t}$ between [-1,1]. However, this model suffers from vanishing gradients, and hence it is not able to handle long range dependencies. The LSTM architecture was designed to mitigate this problem.

Long Short Term Memory (LSTM) networks have an additional memory state called the cell state $\mathbf{c_t}$. Figure 3.1 shows the structure of a basic LSTM cell.

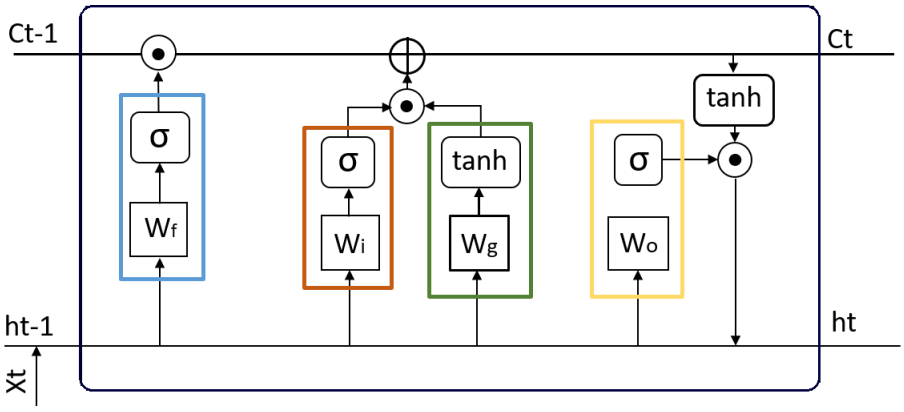

Figure 3.1: Architecture of an LSTM cell, as implemented by Equations 3.2 through 3.4

The removal of previous information and addition of current information to the cell state is regulated by linear interactions between gates **f** (forget gate), **i** (input gate) and **g** (regulator gate). The final output is obtained by the combination of the cell state and the **o** (output gate). The model architecture is mathematically expressed as:

$$\begin{pmatrix} \mathrm{i}_t \\ \mathrm{f}_t \\ \mathrm{g}_t \\ \mathrm{o}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \tanh \\ \sigma \end{pmatrix} \begin{pmatrix} \mathrm{W}_i \\ \mathrm{W}_f \\ \mathrm{W}_g \\ \mathrm{W}_o \end{pmatrix} \begin{pmatrix} \mathrm{x}_t \\ \mathrm{h}_{t-1} \end{pmatrix} \tag{3.2}$$

$$c_t = c_{t-1} \odot f_t + i_t \odot g_t \tag{3.3}$$

$$h_t = tanh(c_t) \odot o_t \tag{3.4}$$

Chapter 4

Methodology

We propose two approaches to visualize what a trained recurrent network has learned. The first is a gradient-based saliency approach that illustrates how relevant each joint is to the class label or to a particular hidden unit. This approach is inspired by the saliency maps explained in [18], which used heat maps to demonstrate the importance of words visually. The second method shows the synthetic skeleton that maximizes the hidden state activation of the class label or a particular neuron. To make these techniques easy to use, we consolidate them into a visualization tool called *SkeletonVis*. SkeletonVis allows users to gain insights into the workings of their trained models in order to increase (or decrease) their confidence in a network's abilities.

The rest of this section describes our techniques in more detail. Section 4.1 describes the design of the action recognition model and how a video sequence gets assigned a class label. Section 4.2 explains how joint saliency is calculated. Section 4.3 describes how we generate skeletons to maximize a class label or hidden state output for a particular neuron. A brief overview of the SkeletonVis tool is explained in Section 4.4.

4.1 Main network architecture

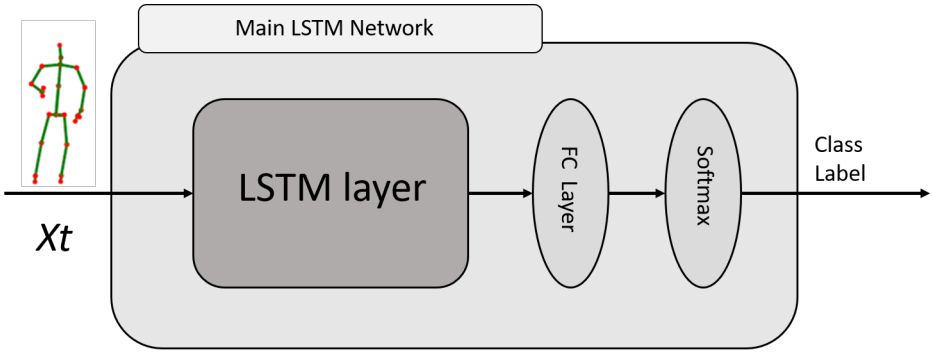

Figure 4.1: The LSTM architecture used for the experiments in this thesis. The LSTM block is as shown in Figure 3.1. It is followed by a single hidden layer converting hidden unit responses into label responses, and then a softmax layer converting label responses into probabilities.

Figure 4.1 show the architecture of a recurrent network used for skeleton-based action recognition. The input is a sequence of skeleton poses over time; the output is a vector of class label probabilities. Opening up the architecture, the recurrent network is a one-layer LSTM cell like the one shown in Figure 3.1. The Main LSTM Network architecture is similar to the one referenced in [9] and [8]. This is followed by a fully-connected (FC) layer and a softmax layer.

Consider a video sequence with T frames. Let x_t denote the input skeleton pose at time instance t. The output of the LSTM cell for this timestep is denoted by h_t. The FC layer takes as input the summation of all outputs of the LSTM's hidden units h_t over the complete sequence, expressed mathematically as:

$$H = \sum_{t=1}^{T} h^{t} \tag{4.1}$$

The FC layer has one output for every class in the data set, called logits. The softmax layer converts the logits into label probabilities. The mathematical formula for the softmax function is:

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{4.2}$$

The class label of the video sequence under consideration is the one having the maximum probability.

4.2 Gradient-based saliency

Gradients can help us understand the contribution of each individual input unit to the final output of a network. This technique has been used extensively to find localized class-discriminative visual explanations in images for CNN models [21] and to find important words in text mining [18]. In skeleton-based action recognition, saliency can help understand the contribution of every body joint to the decision about a particular class of action.

For the model shown in Figure 4.1, we first explore the gradient of the response h_u of a particular hidden unit *u* with respect to dimension *d* of joint *j*. Note that we are considering the partial derivative of h_u with respect to the pose input x_t and not the previous hidden state input h_{t-1}. We

are therefore measuring the sensitivity of a particular pose value in time, not the combined impact of a joint over time. We denote the gradient $g^{u}_{t,j,d}$ as:

$$g^{u}_{t,j,d} = \frac{\delta h_t{}^{u}}{\delta x_{t,j,d}} \tag{4.3}$$

where x_t and $h_t{}^{u}$ are the pose input and hidden state output of neuron u at time instance t respectively. The absolute value of $g^{u}_{t,j,d}$ denotes the sensitivity of the input joint to the final output hidden state. Thus, we denote sensitivity $S^{u}_{t,j,d}$ as:

$$S^{u}_{t,j,d} = |g^{u}_{t,j,d}| \tag{4.4}$$

For any particular time t, joint j and dimension d, the sensitivity $S^{u}_{t,j,d}$ is a scalar. Empirically, we note that when there are joints with very little motion across the data set, i.e. body parts that don't move, their sensitivity can become very large due to random sensor noise. Hence, we normalize sensitivity across an sequences as:

$$S'^{u}_{t,j,d} = \frac{\sigma_{x_{t,j,d}}}{\sigma_{h^{u}_{t}}} * S^{u}_{t,j,d} \tag{4.5}$$

where $S'^{u}_{t,j,d}$ denotes the normalized sensitivity for dimension d of joint j for a particular neuron u at time t. $\sigma_{x_{j,d}}$ denotes the standard deviation of the pose input $x_{j,d}$, and σ_{hu} denotes the standard deviation of h^{u} over the complete video sequence. We will refer to the normalized sensitivity $S'^{u}_{j,d}$ for the rest of this paper.

Sensitivity is a scalar value for each dimension of a joint in the skeleton pose. For a given time t, we denote the summation of sensitivities across the X, Y and Z dimensions of a joint as the final contribution of that joint in the output hidden state of a neuron. Thus:

$$S'^{u}_{t,j} = \sum_{d=1}^{3} S'^{u}_{t,j,d} \tag{4.6}$$

Equation 4.6 measures the sensitivity of a given hidden unit u to input joint j at time t. To understand the impact of a joint not just on a single hidden unit but on the overall class response we take the weighted product of the normalized joint sensitivities for a particular class. This is calculated by taking the product of the sensitivity of a neuron with its magnitude in the weight matrix column of the FC layer for the respective class. The values in the weight matrix of the FC layer indicate the final effect of a hidden unit in the classification of an input sequence. This weight matrix has dimensions (H, C) where H is the number of hidden neurons in the LSTM, and C is the number of classes in the data set. Thus, for any given class, the respective weight matrix column shows the contribution of each neuron in the classification decision. The final sensitivity map is represented as an aggregation of the weighted sensitivities of all neurons for the class under consideration and can be written as:

$$S'_{\mathrm{t,j}} = \sum_{u=1}^{H} S'^{\mathrm{u}}_{\mathrm{t,j}} * |W^{\mathrm{u}}{}_{\mathrm{c}}| \tag{4.7}$$

where $|W^{\mathrm{u}}{}_{\mathrm{c}}|$ is the magnitude of the weight of neuron u for class c. This result is visually represented in SkeletonVis as a sequential colormap with darker values for large sensitivities and lighter values for small ones.

4.3 Activation Maximization

Sensitivity visualization shows a user what body parts are having the most influence over a class label or the response of an individual hidden unit. Activation maximization, on the other hand, is a technique that generates synthetic inputs that maximize the response of a class label or a hidden unit. The idea is to warn users about inputs that the network might never have encountered but which never the less causes the network to generate a very strong response for a particular class label.

Activation maximization is implemented by hill-climbing. We begin with an input sequence that receives the class label c we are interested in studying. Starting with this input, we calculate the gradient of the fully connected layer for class c with respect to the input as:

$$d^{\mathrm{c}}{}_{\mathrm{t,i}} = \frac{\delta o_{\mathrm{c}}}{\delta x_{\mathrm{t,i}}} \tag{4.8}$$

where $\mathrm{d}^{\mathrm{c}}{}_{\mathrm{t,i}}$ denotes the gradient of the output for class c with respect to the input pose x_i at time t. Note that this gradient is obtained for all neurons in the LSTM cell for input x_i and time t and is therefore a vector of length H, where H is the number of hidden units in the LSTM.

The gradient $d^c_{t,i}$ is a weighted sum of the gradients of every hidden unit u with respect to the input pose x_i at time t. i is the dimension of the input pose x_t. This can also be written as:

$$d^{\mathrm{c}}{}_{\mathrm{t,i}} = \sum_{u=1}^{H} W^{\mathrm{u}}{}_{\mathrm{c}} * \frac{\delta h_{\mathrm{t}}{}^{\mathrm{u}}}{\delta x_{\mathrm{t,i}}} \tag{4.9}$$

The value of $d^c_{t,i}$ is used to update the input pose x_t for the next iteration. The input is updated with the gradient and passed again to the model to calculate the change in the activation. Iterating this process over a finite number of steps gives a synthetic skeleton sequence that maximizes the activation of that class. For example, Figure 4.2 shows the plot of the change in hidden state activations and the corresponding change in the gradient over 40 iterations if we run the hill climbing method over a sequence of throw action. Based on the plot, we choose the skeleton from iteration 12 as the final skeleton that maximizes the hidden state activation for throw gesture.

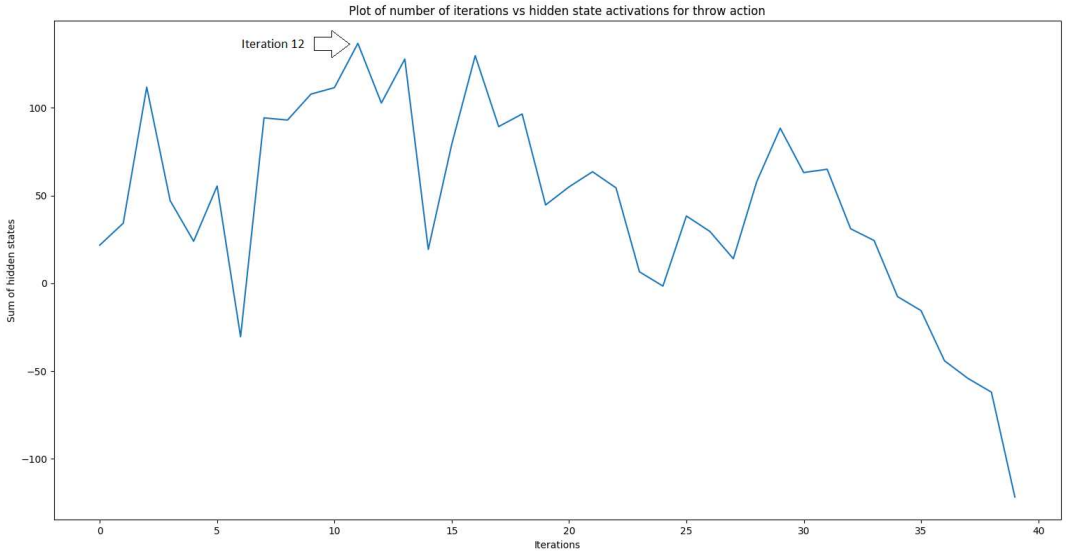

Figure 4.2: The plot of running the hill climbing method over a throw action sequence for 40 iterations. We choose the skeleton at iteration 12 as it maximizes the hidden state activation for this activation.

Unfortunately, the LSTM treats every input feature $x_{t,j,d}$ as independent. The gradient update calculated by Equation 4.9 alters the data to increase the networks response, but the result may look nothing like a reasonable human skeleton.

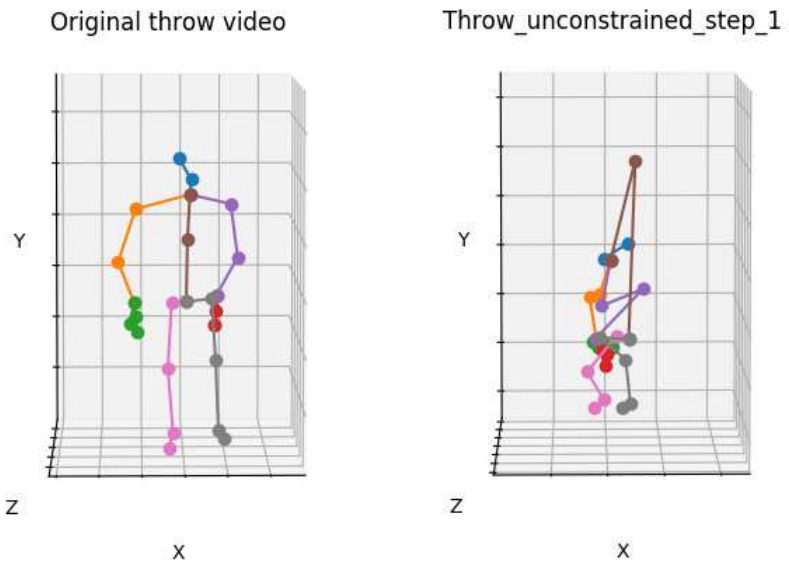

Figure 4.3: Frame (a) shows the original skeleton of a throw gesture. Frame (b) shows the skeleton updated after one iteration. Although this skeleton maximizes the hidden state activation, the result is far from a reasonable human skeleton.

Figure 4.3(a) shows the original skeleton pose of a throw action and Figure 4.3(b) shows the input pose updated with the gradient obtained by Equation 4.9. The human form is unrecognizable in this picture. The middle of the spine has been moved to the top, elongating the spine as well as giving it an unrealistic degree of curvature. Other joints have been moved in odd ways as well, resulting in a non-human shape.

In many ways, this situation is analogous to what happens when activation maximization is applied to convolutional neural networks performing image classification. Activation maximization produces "images" that fool the CNN, but look like white noise to human observers [28–30]. In our case, activation maximization produces "skeletons" that don't look like skeletons to human observers. Fortunately, in action recognition, unlike general image recognition, human anatomy provides constraints that can be used to alter how we update poses.

To produce valid skeletons, we apply two types of constraints, *bone length constraints* and *pairwise angle constraints*. We are aware that the constraints below are not exhaustive. At this stage, we are relying on a few, important constraints for conceptualization.

For a frame f, we construct a state vector s_f as:

$$\mathbf{s_f} = [x_0, x_1..x_{\mathrm{N}}, y_0, y_1..y_{\mathrm{N}}, z_0, z_1..z_{\mathrm{N}}] \tag{4.10}$$

Thus, $\mathrm{s_f}$ is a $N \times 3$ dimension vector, where N is the number of joints.

Bone Length constraints:

The bone length $\mathrm{b_{i,j}}$ between any two connected pair of joints is given by the Euclidean distance between the joints:

$$b_{\mathrm{i,j}} = \sqrt{(x_{\mathrm{i}} - x_{\mathrm{j}})^2 + (y_{\mathrm{i}} - y_{\mathrm{j}})^2 + (z_{\mathrm{i}} - z_{\mathrm{j}})^2} \tag{4.11}$$

For Kinect version 2, there are 25 joints and 24 connected joints (bones). We consider the reference bone length $b_{i,j}$ to be the mean of $b_{i,j,T}$ over the input video sequence. The bone length constraint is defined as:

$$\left\| \left(s_{\mathrm{f}} + d'\right) \cdot A_{\mathrm{i,j}} \right\| / \sqrt{2} - b_{\mathrm{i,j}} = 0 \tag{4.12}$$

where $\mathrm{A_{i,j}}$ is a 75 x 75 dimension matrix, and d' is a 75 x 1 dimension vector having the same format as s_f. For example, for joints (0,1) the A matrix will be represented as:

$$\mathbf{A}_{0,1} = \begin{array}{c} \begin{array}{ccccccc} 0 & 1 & ...25 & 26 & ...50 & 51 & ...74 \end{array} \\ \begin{bmatrix} 1 & -1 & ..0 & 0 & ...0 & 0 & 0 \\ -1 & 1 & ..0 & 0 & ...0 & 0 & 0 \\ 0 & 0 & ..1 & -1 & ...0 & 0 & 0 \\ 0 & 0 & .. -1 & 1 & ...0 & 0 & 0 \\ 0 & 0 & ..0 & 0 & ...1 & -1 & 0 \\ 0 & 0 & ..0 & 0 & ... -1 & 1 & 0 \\ 0 & 0 & ..0 & 0 & ...0 & 0 & 0 \end{bmatrix} \begin{array}{c} 0 \\ 1 \\ ..25 \\ 26 \\ ..50 \\ 51 \\ ..74 \end{array} \end{array}$$

In addition to preserving bone lengths between connected pairs, certain joint angle constraints are also imposed on skeletons.

Joint Angle Constraints:

Inspired by the joint angle limits in [25], we propose three joint constraints and four joints constraints. The conditions for three joint angle constraints are as follows: Let $\mathbf{v_{sb,sm}}$ and $\mathbf{v_{ss,\,sm}}$ be two unit vectors from spine mid to spine base and spine mid to spine shoulder respectively. We constrain the angle formed by $\mathbf{v_{sb,sm}}$ and $\mathbf{v_{ss,\,sm}}$ to be between 160°and 180°. Mathematically, this is expressed as:

$$-0.93969 \geqslant (v_{\text{sb,sm}} \cdot v_{\text{ss, sm}}) \geqslant -1 \tag{4.13}$$

We impose similar constraints on the following set of joints: angle between spine shoulder and both shoulders to be between 110°and 180°, the angle made by the spine base with the hips to be between 100°and 180°, the angle made by the wrist with the elbow and hand joint to be between 90°and 180°.

We formulate a similar angle constraint with four joints as: Let $\mathbf{v_{rh,sb}}$, $\mathbf{v_{lh,sb}}$ and $\mathbf{v_{sm,sb}}$ be three unit vectors from spine base to right hip, spine base to left hip and spine base to spine mid respectively. We define the vector $\mathbf{n_{rh,sb,\,lh}}$ as the normal vector to the plane defined by $\mathbf{v_{rh,sb}}$ and $\mathbf{v_{lh,sb}}$.

$$n_{\text{rh,sb, lh}} = (v_{\text{rh,sb}} \times v_{\text{lh,sb}}) \tag{4.14}$$

For the four joints to be in a valid position, we restrict the vectors $\mathbf{n}_{\text{rh,sb, lh}}$ and $\mathbf{v}_{\text{sm,sb}}$ to be between $0°$and $90°$. Mathematically this is written as:

$$1 \geqslant (n_{\text{rh,sb, lh}} \cdot v_{\text{sm,sb}}) \geqslant 0 \tag{4.15}$$

All the constraint equations (bone lengths, three joint angles and four joint angles) are grouped and denoted as C. We then find the update d' that optimizes:

$$\text{minimize} \quad (d - d')^2 \quad \text{subject to} \quad C. \tag{4.16}$$

The gradient d' obtained from solving the optimization equation has the same format as s_f. Hence we convert it back to the the format of the original input pose, having 3 dimensions for every joint grouped together. The rearranged update d' is denoted as follows:

$$d' = [x_0, y_0, z_0, x_1, y_1, z_1 ..x_{\text{N}}, y_{\text{N}}, z_{\text{N}}] \tag{4.17}$$

The current skeleton pose is updated with d' and iterated to hill climb in the space of valid skeletons.

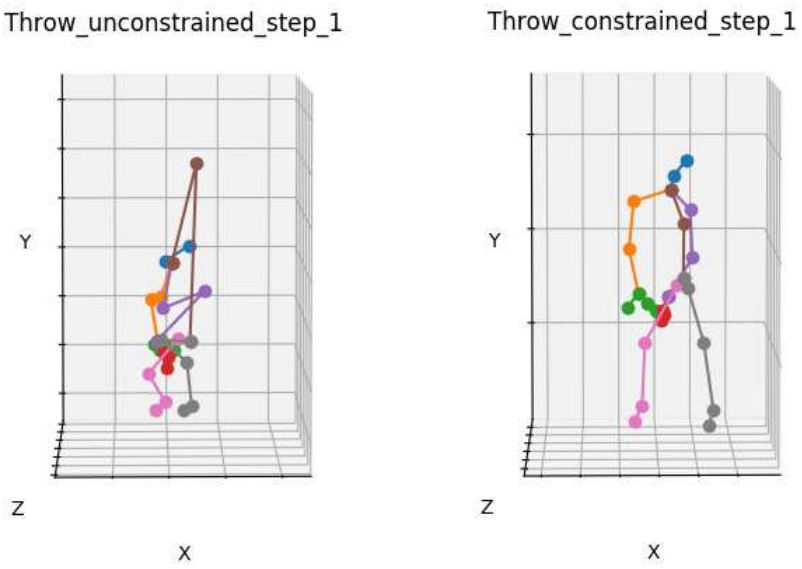

Figure 4.4: Why anatomical constraints matter. Frame (a) shows a skeleton after one iteration of activation maximization without anatomical constraints. Frame (b) shows it after one iteration with constraints.

Figure 4.4(b) shows the skeleton updated with one iteration of the constrained gradient. Equation 4.16 updates the input (i.e. the sequence of skeleton poses) so as to increase the label response while still generating skeletons that satisfy the anatomical constraints. As shown in Figure 4.4(b), the first update yields a more extreme motion that optimizes, in this case, the *throw* action. If we continue the gradient updates until convergence, we get a skeleton that maximizes the class response for the sequence. Unfortunately, the skeleton that optimizes the class response fails to look like a skeleton, despite adhering to all the constraints imposed on it.

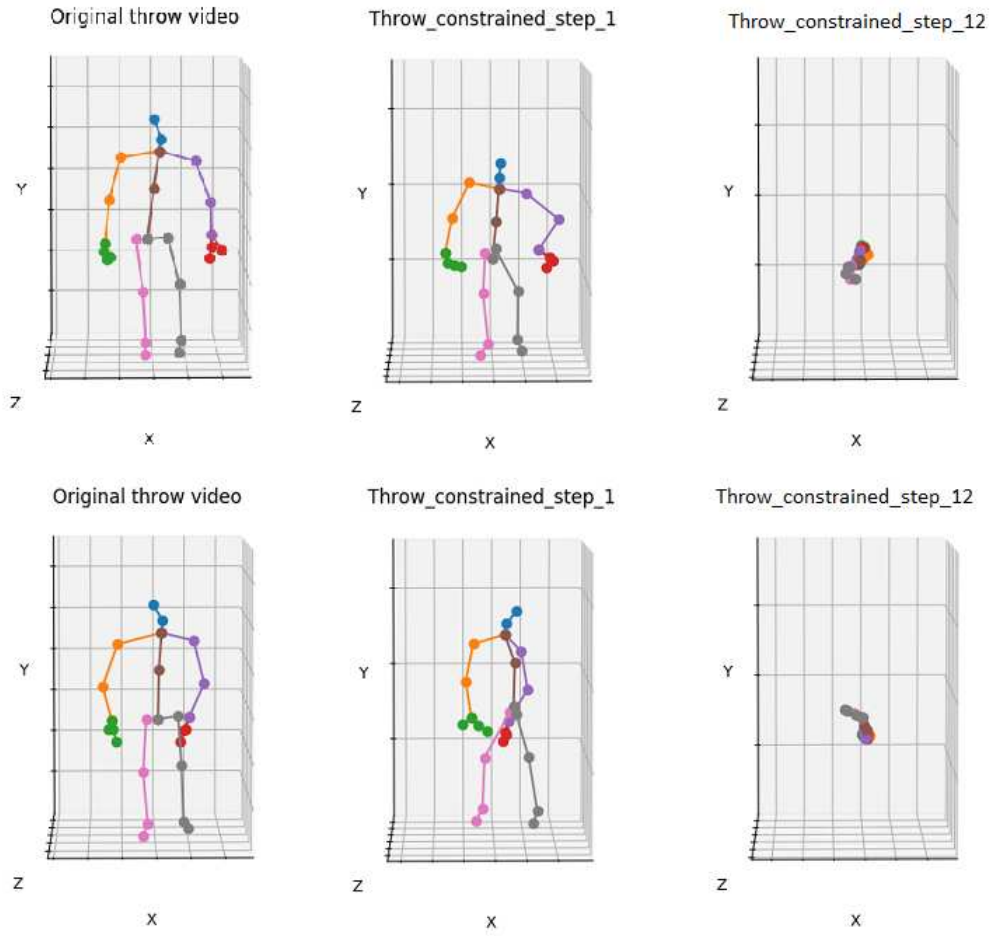

Figure 4.5: Figure shows the progression of skeleton

Figure 5.4 shows the skeleton that optimizes the response but does not look like a valid skeleton. We could add more constraints, for example, by requiring that the skeleton be supported rather than floating in mid-air, but at the moment these unrealistic optima provide a warning about inputs that generate strong false responses, while earlier stages in the optimization show us more realistic motions that still strengthen the response.

4.4 SkeletonVis

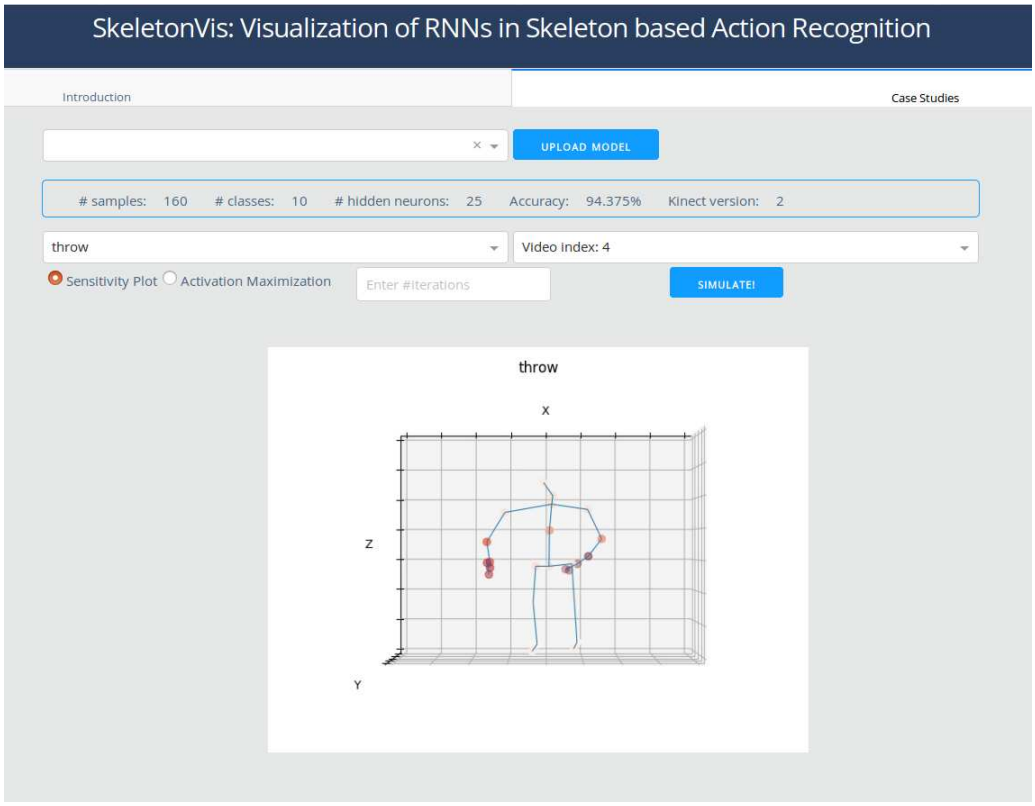

Figure 4.6: The SkeletonVis tool, as it appears to LSTM developers and users.

We aggregate sensitivity analysis and activation maximization into an interactive visualization tool called *SkeletonVis*. This tool is intended to help users better understand the models learned by LSTMs. The model is built using Tensorflow [31] and the web framework Dash [32].

SkeletonVis can be used over the web to see visualizations of our already trained networks, or it can be downloaded and run locally to examine the user's own LSTM networks. Users can log on to view the existing case studies, or they can download the source code from the CSU repository to run it locally.

Figure 4.6 shows SkeletonVis as it appears when a user logs on to xxx.edu. On top, the system summarizes the model and data information, showing the number of data samples, classes, and hidden neurons in the model, as well as the Kinect version used and the classification accuracy

of the system. The Kinect version is necessary to know how many joints there are, and in what order they appear. Users select the class or hidden unit they want to inspect, and an input video to visualize. Users also choose whether to visualize sensitivity or activation maximization, and in the case of activation maximization how many optimization iterations to apply. In the next section, we present case studies done using this tool.

Chapter 5

Case Studies

This chapter provides examples of applying the two visualization techniques on action sequences of known data sets. To achieve this, we trained an LSTM network on the well-known NTU-RGBD action recognition data set [33], and analyzed it using SkeletonVis. Although the network has a relatively good recognition performance, the focus of this work is on the analysis of the network, not the network's accuracy.

The rest of the section is structured as follows: Section 5.1 describes the NTU-RGBD dataset and the data normalization techniques used for this experiment. Section 5.2 gives an overview of the hyperparameters in the model architecture, providing information for any user to replicate the model for future work. The visualizations and discussions of sensitivity analysis on two actions, namely, *throw* and *kick* are provided in Section 5.3. The chapter concludes with Section 5.4 discussing activation maximization applied on *throw* action.

5.1 Data set specifications

This section introduces the reader to the NTU-RGBD data set, providing information about the number of classes, illustrations of video samples, etc. It also describes the normalization method used on the data samples before training for the model.

5.1.1 Description

The NTU-RGBD dataset is currently the largest data set for action recognition. It provides multi-modal data containing RGB videos, depth map sequences, 3D skeletal data, and infrared videos for each sample. It contains 56,880 video samples of 60 action classes, performed by 40 participants, seen from various viewpoints. Cross Subject (CS) and Cross View (CV) are the two standard modes of evaluation. Figure 5.1 shows the frames from a video sequence of *shaking hands* action, seen as RGB, depth, skeleton and infra-red data.

Figure 5.1: Sample frames from a video sample of two persons shaking hands. Clockwise from top left, RGB, RGB + skeleton, IR, depth modalities of the data.

Table 5.1: List of 49 single person actions in the NTU-RGBD data set.

drink water	eat meal	brush teeth	brush hair	drop	pickup	throw
sitting down	standing up (from sitting position)	clapping	reading	writing	tear up paper	wear jacket
take off jacket	wear a shoe	take off a shoe	wear on glasses	take off glasses	put on a hat/cap	take off a hat/cap
cheer up	hand waving	kicking something	put/take out something inside/from pocket	hopping (one foot jumping)	jump up	make a phone call/answer phone
playing with phone/tablet	typing on a keyboard	pointing to something with finger	taking a selfie	check time (from watch)	rub two hands together	nod head/bow
shake head	wipe face	salute	put the palms together	cross hands in front (say stop)	sneeze/cough	staggering
falling	touch head (headache)	touch chest (stomachache/heart pain)	touch back (backache)	touch neck (neckache)	nausea or vomiting condition	use a fan (with hand or paper)/feeling warm

For simplicity, we trained our network on the 49 actions that contain only a single person; there are 44,213 videos of the 49 actions. Table 5.1 shows the 49 action classes performed by a single person. Data obtained from one participant (Participant 2) is reserved as validation data, which consists of 739 samples across 49 classes, equivalent to 2% of the available training data.

5.1.2 Data Normalization

The skeleton data is normalized as specified in the dataset's reference paper. The skeleton files specified by the authors are skipped, and noisy skeletons are eliminated. The mean pose of the actor is calculated for the video, and the average position of the spine mid joint (Kinect joint index 1) is set to be the origin. The coordinate system is rotated such that the X axis is aligned with the vector from the right to the left shoulder, and the Y axis is aligned with the vector from the spine base to the spine shoulder. All 3D points are then scaled by the distance between the spine base and the spine joint.

This experiment consisted of only one person performing the action. However, for a video sequence with two persons, all transformations mentioned above are done with respect to the main actor. The main actor is decided on the basis of the highest amount of 3D body motion. Kinect sensor is prone to detect false skeletons in the background. To eliminate these noisy skeletons, we calculate the spread of the joint locations towards image axis and filter out the ones whose X spread were more than 0.8 of the Y spread, for each skeleton.

5.2 Model Specifications

For the purposes of our case studies, we trained a single-layer LSTM network on the NTU-RGBD data set. The 1 layer model consists of 150 hidden neurons in the LSTM Cell, trained with a batch size of 128. Adam optimizer [34] is used for training with an initial learning rate of 0.005. Learning rate is reduced by a factor of 10 after 100 epochs. The training was terminated after 25800 iterations.

5.3 Sensitivity analysis

The role of sensitivity analysis is to give the user an intuition about what parts of the body a class label or hidden unit is paying attention to. For most actions, we start with an idea of what joints define the action and therefore what joints the network should focus on. For example, for the *throw* action we expect the network to concentrate on the hands and elbows, while for the *kick*

action we expect it to concentrate on the knees and feet. But the whole point of sensitivity analysis is to determine whether what the network learned matches our expectations.

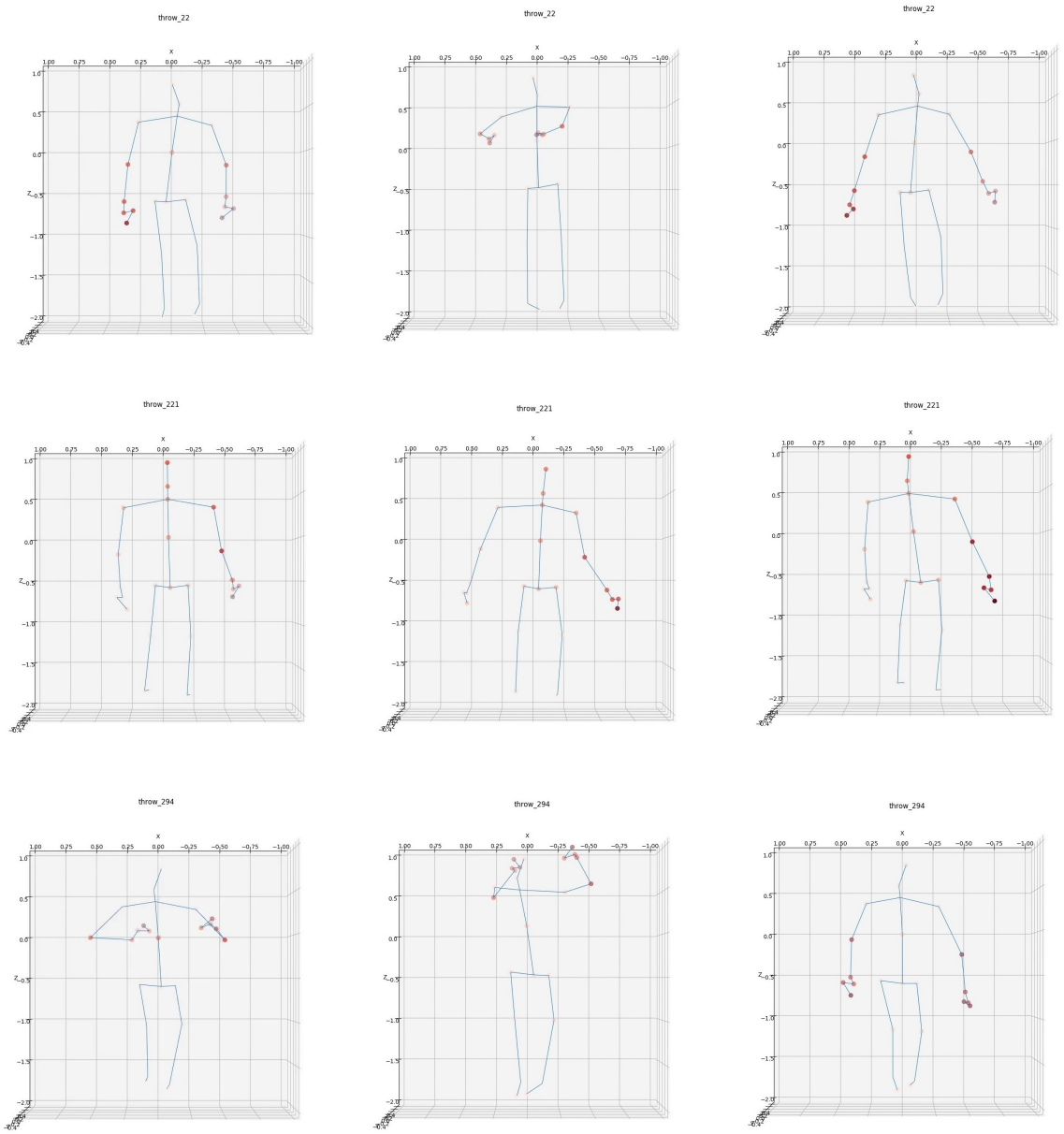

Figure 5.2: Progressive frames of sensitivity visualizations for three variations of throw action: Tow handed throw, one handed under-arm throw, two handed basketball throw. All three variations have a high sensitivity for the arm/arms that makes the throw action. The spine mid, which decides the body posture during the action has high sensitivity initially for rows 1 and 3. Row 2 has high sensitivity for head, neck and hip joints, due to slight crouching of the person during the action.

The *throw* action has three variants in the data set, namely, one handed under arm throw, two handed throw, and a two handed upward basketball throw. Figure 5.2 shows the sensitivity plot of selected frames for all the three variants of this action. As we would have expected, the arm or arms contributing to the throw action have high sensitivity. In case of the under arm throw, the dominant hand has high sensitivity, while for the two handed throws, both hands and elbows have almost equal sensitivity.

However, a less expected phenomenon seen across all these samples is the sensitivity of the mid-spine. This joint has an equal sensitivity to the hands and elbows in the initial frames of the action, and fades almost instantly. Since spine mid joint is the basis for the translation component of normalization, its average position across a video is always the origin. The *throw* label seems to be sensitive to it, however, because throw motion has a vertical movement of the torso in all the video samples. This is seen in the upward motion in both two handed, and a downward motion in the one handed throw.

For the one handed under arm throw action, the head, neck and hip joints have moderate to high sensitivity through the video sequence. This could be due to a slight crouching body posture during the throw. This indicates that, during an action, the model pays attention to those joints that alter the posture in a meaningful way. A validation of this theory can be seen from the fact that for both two handed throws, the other joints do not have a significant change in their colors.

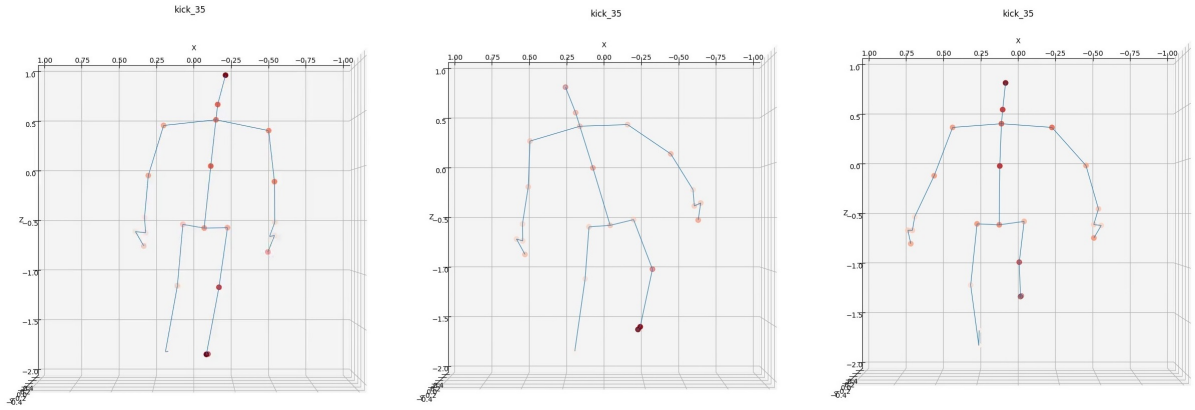

Figure 5.3: Sensitivity visualizations of frames extracted from kick action. (a) and (c) show high sensitivity to spine, shoulders, elbow and legs. (b) shows high sensitivity to foot, and low sensitivity to upper body. (a) and (c) indicate that the model is attentive to the starting pose of an action.

Figure 5.3 shows frames from a *kick* action. With kicking, we expected the attention to be focused on the knees and feet. The actual story is more dynamic. In the early frames of the video, the LSTM is sensitive to the spine, shoulders and elbows as well as the knees and left foot. As the kicking motion progresses, the LSTM becomes less sensitive to the upper body, and focuses on the foot instead. This may be because the starting pose is important to classifying *kick*, since most NTU kicks begin from a standing position. It may also be that people tend to spread their arms slightly at the beginning of a kicking motion for balance. Whatever the reason, what is likely is that a non-standing kick would probably not be recognized by this network. (*Also, the network has learned that most subjects stand on their right foot and kick with their left, so kicks with the other foot might also be missed.)

5.4 Activation Maximization

In addition to sensitivity analysis, SkeletonVis shows synthetic videos that progressively increase the hidden state activation for a particular class. We consider the example of *throw* action which is already correctly classified, and see the changes the model favors to increase its response.

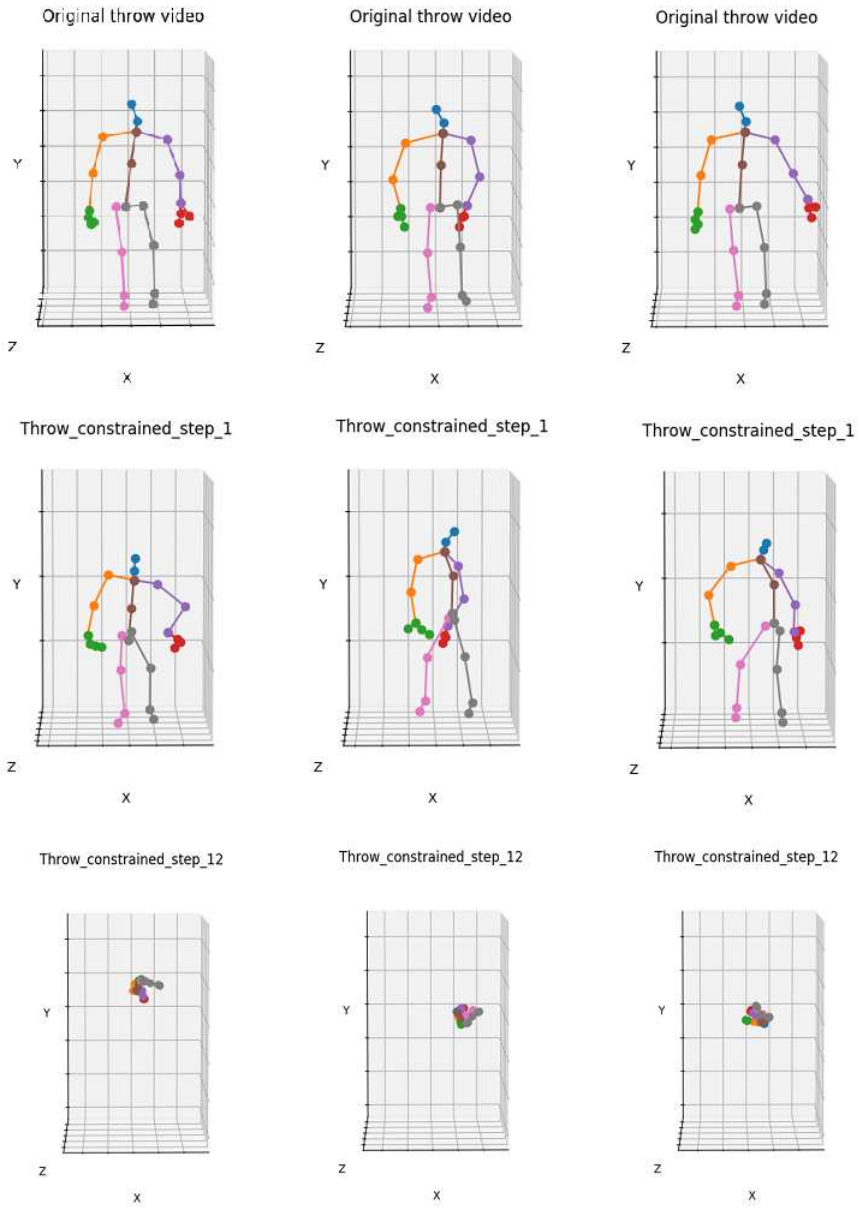

Figure 5.4: Figure shows the progression of skeleton as it gives the maximum response at iteration 12. The skeleton conforms to all constraints, but does not look like one.

Figure 5.4 concentrates on a three frames from a video of *throw* action. The first row shows skeleton frames from the original video. The middle row shows this skeleton accentuated through one iteration of our maximization algorithm. The bottom most row shows the final local optima reached after 12 iterations.

We learn different things from the second and third rows of the above figure. The second row teaches us how to improve the *throw* response by showing us what would have made the seed video even more of a "throw". In this case, the response would be stronger if the participant began in a more crouched position, with their left arm slightly more curled and their left shoulder dipped. The feet are also shown slightly more splayed, but we know that *throw* is not very sensitive to the positions of the feet, so presumably this difference is not important.

The third row of Figure 5.4, on the other hand, warns us about videos that could fool us. For the *throw* motion, activation maximization converges to a local optimum which can only be described as a floating contortionist. Although the bone and angle constraints are all satisfied, the skeleton is so contorted that the joints are practically on top of each other, and it is floating in mid-air. This is clearly not a feasible input, yet it maximizes the *throw* response. This suggests a possible source of strong but false responses.

Chapter 6

Conclusion and Future work

Recurrent Neural Networks are one of the state of the art deep learning methods for modeling sequential data. However, the internal workings of these models are often treated as black boxes by the researchers using them. This thesis tries to provide insights into the models learned by RNNs through visualizations in the domain of action recognition using skeleton data. Using NTU-RGBD, a widely popular data set for action recognition providing skeleton data, we present case studies that help us understand the details in an action performed by a human subject.

This thesis introduces two methods of visualizations of interpreting RNN models in the domain of action recognition. Using class weighted gradient based saliency of the inputs with respect to the outputs, also known as sensitivity, we explore the most relevant joints in an action. We observe that while the joints we expect to be important generally are important, there is often more to the story. For example, in our LSTM trained on the NTU-RGBD data set, for the *throw* motion, as expected, the motion of the hands has high sensitivity. However, in addition, the model has high sensitivity to the upward trajectory of the torso. Similarly, for the *kick* motion, we expect the attention to be focused on the knees and feet. Howver, we observe that the starting pose is important to classifying *kick*, with the slightly spread arms as an indicator of the starting of the action.

The second visualization method, known as activation maximization, generates synthetic videos that maximize the responses of a class label or hidden unit within a set of known anatomical constraints. The goal of this technique is to show users what the system has learned, thus giving an intuition about what the idealized from of the action looks like, and therefore how it might respond to novel inputs. For the *throw* action, the model favors the subject to be in a crouched position. At the same time, by running activation maximization to convergence we produce impossible videos that can fool the RNN, even if each skeleton pose conforms to the the stated constraints.

We aggregate these visualization techniques into an interactive visualization tool called SkeletonVis, which we are making available to the public to allow RNN developers and users to gain more insights into these enigmatic networks.

This study can be extended into many avenues in the research of interpretability of recurrent network models. One of the interesting future works is to improve the anatomical constraints underlying activation maximization. We will add temporal constraints to eliminate implausible accelerations, and volume constraints to prevent body parts from passing inside each other. We also plan to explore activation minimization, by which we mean the extent to which the motion in a video can be reduced without changing the assigned label. Lastly, the state of the art skeleton based action recognition models rely on multi-layer LSTMS rather than a single layer model. Exploring feature abstractions from higher layers would be another interesting extension to this thesis.

Bibliography

[1] S. Hochreiter and J. Schmidhuber. Long short-term memory. 1997. Neural computation, 9(8):1735âĂŞ1780.

[2] Kyunghyun Cho, Bart van MerriÂÍenboer, Dzmitry Bahdanau, and Yoshua. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. 2014. arXiv:1409.1259.

[3] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D. Trippe, and Juan B. Gutierrez. A brief survey of text mining: Classification, clustering and extraction techniques. 2017. arXiv:1707.02919v2.

[4] MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. A comprehensive survey of deep learning for image captioning. 2018. arXiv:1810.04020v2.

[5] Nayyer Aafaq, Syed Zulqarnain Gilani, Wei Liu, and Ajmal Mian. Video description: A survey of methods, datasets and evaluation metrics. 2018. arXiv:1806.00186v1.

[6] Alex Graves, Abdel Rahman Mohamed, , and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. 2013. IEEE international conference on acoustics, speech and signal processing. IEEE, 2013, pp. 6645âĂŞ6649.

[7] Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani. State-of-the-art speech recognition with sequence-to-sequence models.

[8] Pengfei Zhang, Cuiling Lan, Junliang Xing, Wenjun Zeng, Jianru Xue, and Nanning Zheng. View adaptive recurrent neural networks for high performance human action recognition from skeleton data. *arXiv:1703.08274v2*, 2017.

[9] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. 2017. In AAAI Conference on Artificial Intelligence, pages 4263âĂŞ4270.

[10] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and Xiaohui Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. 2016. Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16).

[11] Z.Qin, F.Yu, C.Liu, and X.Chen. How convolutional neural networks see the world âĂŤ a survey of convolutional neural network visualization methods. 2018. Mathematical Foundations of Computing c American Institute of Mathematical Sciences Volume 1, Number 2, May 2018.

[12] https://distill.pub.

[13] Zhihao Zhu, Zhan Xue, and Zejian Yuan. Topic-guided attention for image captioning. 2018. arXiv:1807.03514v1.

[14] Chenxi Liu, Junhua Mao, Fei Sha, and Alan Yuille. Attention correctness in neural image captioning. 2017. Association for the Advancement of Artificial Intelligence 2017.

[15] Gongbo Tang, Rico Sennrich, and Joakim Nivre. An analysis of attention mechanisms: The case of word sense disambiguation in neural machine translation. 2018. arXiv:1810.07595v1.

[16] H. Strobelt, S. Gehrmann, B. Huber, H. Pfister, , and A. M. Rush. Visual analysis of hidden state dynamics in recurrent neural networks. 2016. CoRR, abs/1606.07461.

[17] Y.Ming, S.Cao, R.Zhang, Z.Li, and Y.Chen. Understanding hidden memories of recurrent neural networks. 2017. CoRR, abs/1710.10777v1.

[18] J.Li, X. Chen, E.Hovy, and D. Jurafsky. Visualizing and understanding neural models in nlp. 2016. CoRR, abs/1506.01066v2.

[19] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science 304 (5667) (2004), 78âĂŞ80.*, 2004.

[20] H. Jaeger. The âĂİecho stateâĂİ approach to analysing and training recurrent neural networks - with an erratum note. *Tech. rep., GMD - German National Research Institute for Computer Science, Tech. Rep. (2001).*, 2001.

[21] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *Tech. rep., GMD - German National Research Institute for Computer Science, Tech. Rep. (2001).*, 2001.

[22] Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. 2016. arXiv:1512.02017v3.

[23] A.Karpathy, J.Johnson, and F.Li. Visualizing and understanding recurrent networks. 2015. CoRR, abs/1506.02078.

[24] Soumya Ranjan Tripathy, Kingshuk Chakravarty, Aniruddha Sinha, Debatri Chatterjee, and Sanjoy Kumar Saha. Constrained kalman filter for improving kinect based measurements. 2017.

[25] Rishabh Dabral, Anurag Mundhada, Uday Kusupati, Safeer Afaque, Abhishek Sharma, and Arjun Jain. Learning 3d human pose from structure and motion. 2018. arXiv:1711.09250v2.

[26] Morten Engell-NÃÿrregÃěrd and Kenny Erleben. Estimation of joint types and joint limits from motion capture data. 2009. WSCG 2009.

[27] Ijaz Akhter and Michael J. Black. Pose-conditioned joint angle limits for 3d human pose reconstruction. 2015. CVPR 2015.

[28] Abhijit Bendale and Terrance E. Boult. Towards open set deep networks. 2015. arXiv:1511.06233v1.

[29] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. 2015. arXiv:1412.1897v4.

[30] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. 2015. arXiv:1412.6572v3.

[31] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[32] Plotly Technologies Inc. Collaborative data science, 2015.

[33] A. Shahroudy, J. Liu, T.T. Ng, and G.Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. 2016. IEEE Conference on Computer Vision and Pattern Recognition, pages 1010âĂŞ1019.

[34] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. 2015. arXiv:1412.6980v9.