Looking Under the Hood: Visualizing What LSTMs Learn

Dhruva Patil, Bruce A. Draper, and J. Ross Beveridge

Colorado State University {dkpatil, draper, ross}@cs.colostate.edu

Abstract. Recurrent Neural Networks are a state of the art method for modeling sequential data. Unfortunately, the practice of RNNs is ahead of the theory. We lack any method for summarizing or analyzing what a network has learned, once it's trained. This paper presents two methods for visualizing concepts learned by RNNs in the domain of action recognition. The first method shows the sensitivity of joints over time. The second generates synthetic videos that maximize the responses of a class label or hidden unit given a set of 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.

Keywords: LSTM · Action Recognition · Visualizations · Saliency Maps

1 Introduction

Recurrent Neural Networks (RNNs) such as Long Short Term Memory (LSTM) networks [20], have been successful in many applications involving sequential data. Examples can be found in text classification [5], image and video captioning [13] [2], speech recognition [12][7], and action and gesture recognition [26][22][27]. The success of these deep learning models lies in the complex feature representations they learn from training data and encode as combinations of weights in memory.

Unfortunately, the practice of deep learning is ahead of the theory. There is currrently no way of summarizing what a trained RNN has learned. All that a developer knows is the accuracy with which the network labels the validation data. This can lead to surprises when networks learn 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 high. We need methods to visualize what recurrent nets learn. This paper 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 (also called skeletons). This provides a framework for visualizing results and anatomical constraints for generating synthetic inputs. The first visualization method shows the *sensitivity* of joints over time, extending the work by Li et al [15]. 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. For example, when analyzing an LSTM trained to recognize throwing motions we see that it is sensitive to the positions and motions of the arms, which is not a surprise, but also 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. 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 as case studies in the context of LSTMs, but can be applied to most recurrent networks, including Gated Recurrent Unit networks (GRUs [8]). For LSTMs, we present a visualization tool called *SkeletonVis*. 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.

In summary, the contributions of this paper are:

1. A technique for visualizing the sensitivity of an LSTM class label or hidden unit responses to specific joints in pose data.
2. A technique for generating synthetic videos that elicit maximal responses by class labels or hidden units.
3. A software tool called *SkeletonVis* for visualizing LSTMs.
4. Case studies of using SkeletonVis to probe the properties of trained networks.

2 Prior Literature

The computer vision literature includes many methods for visualizing features learned by convolutional neural networks. See [28] for an up-to-date survey and [1] for an interactive summary of visualization techniques in CNNs. This paper builds on two concepts in CNN visualizations, namely saliency maps by Simoyan and Zisserman [21] and activation maximization by Mahendran and Vedaldi [17]. This paper extends their techniques to recurrent networks.

Recently, RNN researchers have introduced techniques like attention mechanism that change the underlying network architecture to study specific properties of the input data. However, the model-driven properties still remain under explored. Diagnostic visualizations of RNN models are better established in the field of natural language processing than computer vision. See [3][15][23][25] for a brief survey. 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 an action recognition LSTM is a 3D skeleton pose over time. Interpreting how relations over time within a video are modeled is particularly difficult. The two main techniques used to understand models in skeleton based action recognition are spatio-temporal attention mechanism [22] and co-occurrence of joints [27]. The spatio-temporal approach [22] helps us understand the importance given by the model to joints and time frames in a sequence. Unfortunately, as with the attention approach, the original model is altered. The co-occurrence of joints approach reveals correlations of joints in an action sequence without changing the model, but the hidden states of the LSTM are not explored. Our approach aims to interpret hidden states directly. The approach in [15] uses saliency heatmaps to highlight the network's ability to understand negative sentiment in text. Our approach shows that the network is able to focus on the most informative joints in an action by assigning them a higher saliency.

As part of visualization, we generate synthetic skeletons that conform to anatomical constraints. 3D pose estimation techniques have a similar need. Tripathy et al [24] propose a constrained Kalman filter to denoise joint coordinates obtained from Kinect sensors. Our approach integrates the bone length constraints proposed in this paper. Dabral et al [9] model joint angle limits and bone length limits as a loss function that strongly penalizes joints that deviate from valid angular limits. While our approach does not consider a loss function minimization, their formulation of joint angle limits is used below. There are pose estimation papers that go farther: [10] discriminates joint types (ball joints vs. hinge joints). Akhter and Black [4] formulate priors to eliminate invalid poses by pose-conditioned joint angle limits. Such constraints may be added to our techniques in the future.

3 Approach

We propose two approaches to visualizing what a trained RNN has learned. The first is a gradient-based saliency approach that illustrates the relevance of a joint to a class label or hidden unit. The approach is inspired by the saliency maps in [15], which use heat maps to visually demonstrate the importance of words. The second approach shows synthetic skeletons that maximize the hidden state activations of class labels or selected neurons. 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.

Figure 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, similar to [22][26]. This is followed by a fully-connected (FC) layer and a softmax layer. The FC layer takes as input the outputs of the LSTM's hidden units h, and has one output unit for every class in the data set. The softmax layer converts the FC outputs into label probabilities.

The rest of this section describes our techniques in more detail. Section 3.1 explains how joint saliency is calculated. Section 3.2 describes how skeletons are generated to maximize a class label or hidden state output. A brief overview of the SkeletonVis tool is explained in Section 3.3.

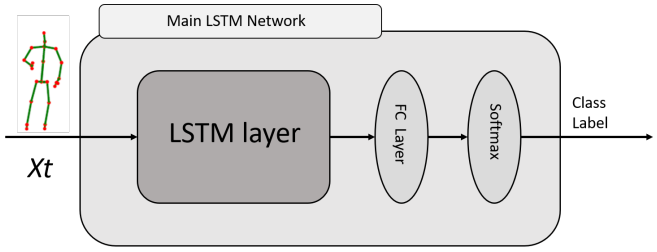

Fig. 1. The LSTM architecture used for the experiments in this paper. It is followed by a single hidden layer converting hidden unit responses into label responses, and then a softmax layer converting label responses into probabilities.

3.1 Gradient-based saliency

Gradients 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 [15]. In skeleton-based action recognition, saliency measures the contribution of every body joint to the decision about a particular class of action.

For the model shown in Figure 1, we first explore the gradient of the response h_u of 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 $\mathrm{x_t}$ and not the previous hidden state input $\mathrm{h_{t\text{-}1}}$. 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 $\mathrm{g^u_{t,j,d}}$ as:

$$g^{\mathrm{u}}{}_{\mathrm{t,j,d}} = \frac{\delta h_{\mathrm{t}}{}^{\mathrm{u}}}{\delta x_{\mathrm{t,j,d}}} \tag{1}$$

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

$$S^u_{t,j,d} = |g^u_{t,j,d}| \tag{2}$$

For any particular time t, joint j and dimension d, the sensitivity $\mathrm{S^u_{t,j,d}}$ is a scalar. 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 a sequence as:

$$S'^{u}_{t,j,d} = \frac{\sigma_{x_{t,j,d}}}{\sigma_{h^{u}_{t}}} * S^{u}_{t,j,d} \tag{3}$$

where S'$^{u}_{t,j,d}$ denotes the normalized sensitivity for dimension d of joint j for neuron u at time t. $\sigma_{x_{t,j,d}}$ denotes the standard deviation of the pose input $x_{j,d}$ over t, and $\sigma_{h^{u}_{t}}$ denotes the standard deviation of h^{u} over t. 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 the joint to the hidden state of the neuron. Thus:

$$S'^{u}_{t,j} = \sum_{d=1}^{3} S'^{u}_{t,j,d} \tag{4}$$

Equation 4 measures the sensitivity of a 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 product of the normalized joint sensitivities with their magnitudes in the weight matrix column of the FC layer for the respective class. The weights in the FC layer indicate the final effect of a hidden unit in the classification of an input sequence. The 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. 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:

$$S'_{t,j} = \sum_{u=1}^{H} S'^{u}_{t,j} * |W^{u}_{c}| \tag{5}$$

where $|W^{u}_{c}|$ 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.

3.2 Activation Maximization

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

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

$$d^{c}_{t,i} = \frac{\delta o_{c}}{\delta x_{t,i}} \tag{6}$$

where $d^{c}{}_{t,i}$ 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. This can also be written as:

$$d^{c}{}_{t,i} = \sum_{u=1}^{H} W^{u}{}_{c} * \frac{\delta h_{t}{}^{u}}{\delta x_{t,i}} \tag{7}$$

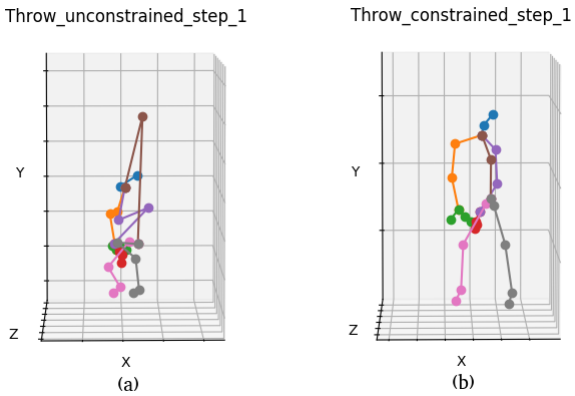

Fig. 2. Why anatomical constraints matter. Frame (a) shows a skeleton after one iteration of activation maximization without anatomical constraints. Frame (b) shows it after one iteration with constraints.

The value of $d^{c}_{t,i}$ is used to update input pose x_t for the next iteration. Unfortunately, the LSTM treats every input feature $x_{t,j,d}$ as independent. The gradient update calculated by Equation 7 alters the data to increase the networks response, but the result may look nothing like a human skeleton. Figure 2(a) shows the input pose updated according to the gradient in Equation 7. The human form is unrecognizable. The middle of the spine has been moved to the top, elongating the spine and 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 [6][18][11]. In our case, activation maximization produces "skeletons" that don't look like skeletons. Fortunately, in action recognition, unlike general image recognition, human anatomy provides constraints that can be used to alter how poses are updated.

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 rely on a few, important constraints for conceptualization.

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

$$\mathbf{s_f} = [x_0, x_1..x_{\mathrm{N}}, y_0, y_1..y_{\mathrm{N}}, z_0, z_1..z_{\mathrm{N}}] \tag{8}$$

Thus, $\mathrm{s_f}$ is a $N \times 3$ dimension vector, where N is the number of joints. The bone length $\mathrm{b_{i,j}}$ between any two connected pair of joints is given by the Euclidean distance between the joints:

$$b_{\mathrm{i,j}} = \sqrt{(x_{\mathrm{i}} - x_{\mathrm{j}})^2 + (y_{\mathrm{i}} - y_{\mathrm{j}})^2 + (z_{\mathrm{i}} - z_{\mathrm{j}})^2} \tag{9}$$

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

$$\left\| \left(s_{\mathrm{f}} + d'\right) \cdot A_{\mathrm{i,j}} \right\| / \sqrt{2} - b_{\mathrm{i,j}} = 0 \tag{10}$$

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

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

In addition to preserving bone lengths between connected pairs, certain joint angle constraints are also imposed on skeletons. Inspired by the joint angle limits in [9], 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, in this case the vectors from spine mid to spine base and spine mid to spine shoulder. We constrain the angle between $\mathbf{v_{sb,sm}}$ and $\mathbf{v_{ss,\ sm}}$ to be between 160°and 180°. Mathematically, this is expressed as:

$$-0.93969 \geqslant (v_{\mathrm{sb,sm}} \cdot v_{\mathrm{ss,\ sm}}) \geqslant -1 \tag{11}$$

Similarly we constrain the angle made by the spine top and the two shoulders to be between 110°and 180°, the angle made by the spine base with the hips to be between 100°and 180°, and the angle made by the wrist with the elbow and hand joint to be between 90°and 180°. We formulate similar angle constraints with

four joints. 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 $\mathbf{v_{lh,sb}}$.

$$n_{\text{rh,sb, lh}} = (v_{\text{rh,sb}} \times v_{\text{lh,sb}}) \tag{12}$$

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

$$1 \geqslant (n_{\text{rh,sb, lh}} \cdot v_{\text{sm,sb}}) \geqslant 0 \tag{13}$$

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:

$$\text{minimize} \quad (d - d')^2 \quad \text{subject to} \quad C. \tag{14}$$

We then add this constrained update d' to the current skeleton pose and iterate to hill climb in the space of valid skeletons. Figure 2(b) shows the skeleton updated with one iteration of the constrained gradient.

Equation 14 modifies the input (i.e. the sequence of skeleton poses) to increase the label response while generating skeletons that satisfy the anatomical constraints. As shown in Figure 2(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 still fails to look like a skeleton, despite adhering to the constraints imposed on it. Figure 5 shows a 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 strengthen the response.

3.3 SkeletonVis

We aggregate sensitivity analysis and activation maximization into an interactive visualization tool called *SkeletonVis*. This tool is intended to help users better understand models learned by LSTMs. SkeletonVis can be used over the web to see visualizations of previously trained networks, or it can be downloaded and run locally to examine the user's own LSTM networks. Users can log on to `http://www.cs.colostate.edu/~vision/skvis_toolset/index.php` to view the existing case studies or download the source code from the same site to run it locally.

Figure 3 shows SkeletonVis as it appears over the web. 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. Users select the class or hidden

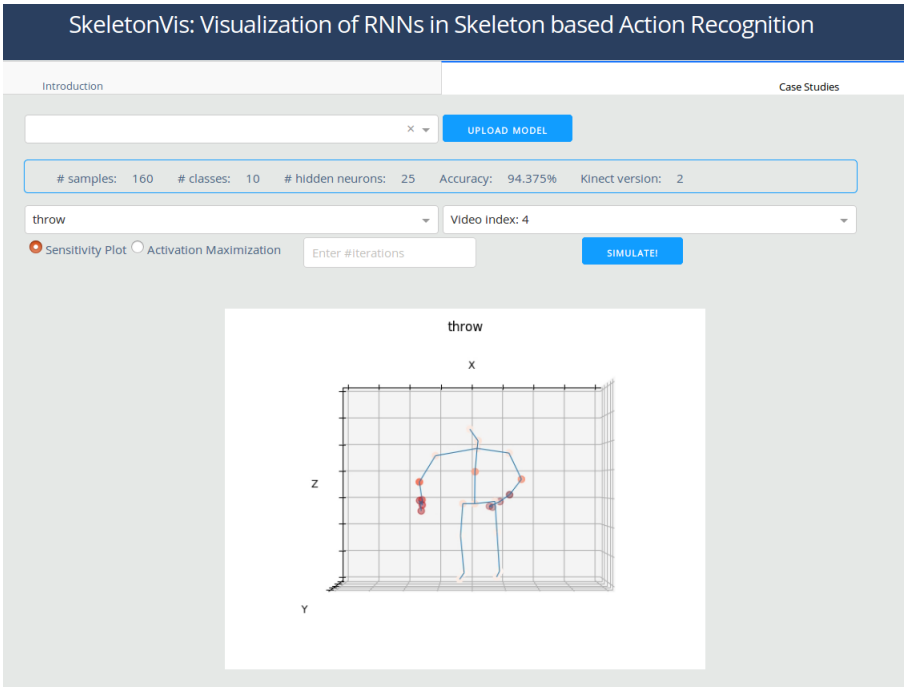

Fig. 3. The SkeletonVis tool, as it appears to LSTM developers and users.

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 activity maximization how many optimization iterations to apply. In the next section, we present case studies using this tool.

4 Case Studies

This section provides examples of using the visualization techniques described above to analyze LSTM networks. In particular, we trained LSTM networks on two data sets, NTU-RGBD action recognition data set [19] and SYSU data set [14], and analyzed what was learned using SkeletonVis. Although the network is modern and relatively good at action recognition, we remind the reader that focus of this paper is on the analysis of the network, not the network's accuracy.

4.1 An LSTM trained on NTU-RGBD Data set

For our first case studies, we trained a single-layer LSTM network on the NTU-RGBD data set [19]. This dataset contains 56,880 video samples of RGB-D videos and skeleton data captured using a Kinect v2. There are 60 action classes

performed by 40 participants and seen from multiple viewpoints. Cross Subject (CS) and Cross View (CV) are the standard modes of evaluation. For simplicity, we trained our network on the 44,213 videos of 49 actions that contain only a single person. Data obtained from one participant (Participant 2) is reserved as validation data (739 samples, equivalent to 2% of the training data). The skeleton data is normalized as specified in [19].

The model contains 150 hidden neurons, trained with a batch size of 128. The Adam optimizer [16] is used for training with an initial learning rate of 0.005. The learning rate is reduced by a factor of 10 after 100 epochs. The training was terminated after 25,800 iterations.

4.2 Sensitivity analysis

The role of sensitivity analysis is to give users 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 intuitive idea of what joints the network should focus on. The point of sensitivity analysis is to determine whether the network matches our expectations.

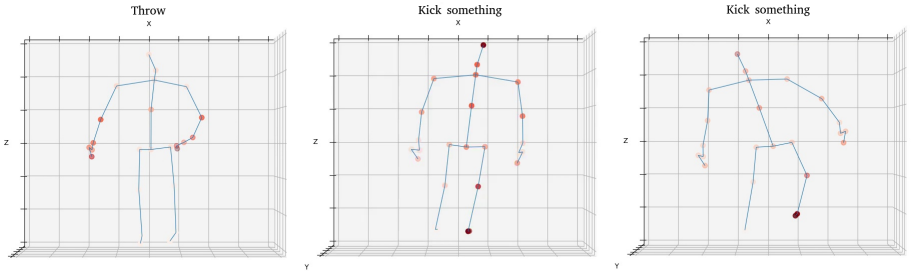

Fig. 4. Sensitivity visualizations of frames extracted from throw and kick actions. (a) shows the sensitivity of joints from an early frame of a throwing motion. (b) shows the sensitivities from an early frame of a kicking motion, while (c) shows a later frame from the same kicking video.

Figure 4(a) shows the sensitivity plot of one frame in a *throw* action. As we expected, the arms are highly sensitive. However, the mid-spine joint is unexpectedly sensitive, too. It has roughly the same sensitivity as the hands and elbows in the initial frames of the action. The *throw* label seems to be sensitive to it because throwing includes a vertical movement of the torso in all the training samples.

Parts (b) and (c) of Figure 4 show frames from a *kick* action. With kicking, we expect 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 seems to emphasize the starting pose, 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 kick for balance.

4.3 Activation Maximization

In addition to sensitivity analysis, SkeletonVis shows synthetic videos produced by activation maximization. Figure 5 concentrates on a single frame from each of two throwing videos. The actual skeleton for each frame shown on the left. The middle shows this skeleton after one iteration of activation maximization. The right shows the final local optima reached after 12 iterations.

We learn different things from the second and third columns of Figure 5. The second columns teaches us how to make the *throw* response stronger through exageration, in this case by putting the participant in a more crouched position, with their left arm more curled and their left shoulder dipped. The feet are also shown as more splayed, but we know that *throw* is not very sensitive to the positions of the feet, so presumably this difference is unimportant.

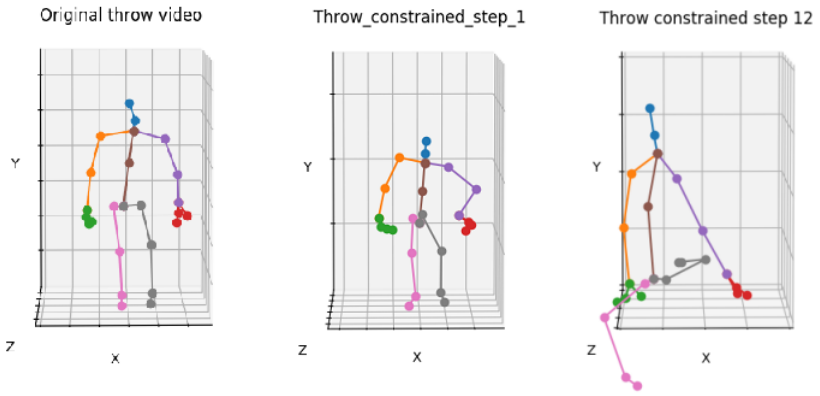

Fig. 5. Figure shows the progression of skeleton

The third column of Figure 5, on the other hand, warns us about non-sensical videos that could fool us. For the *throw* motion, activation maximization converges to a local optimum that might be described as a floating contortionist. Although the bone and angle constraints are satisfied, the skeleton is extremely contorted and floating in mid-air. This is clearly not feasible, yet it maximizes the *throw* response and suggests a possible source of false responses.

4.4 An LSTM trained on SYSU Data set

The SYSU dataset [14] contains 480 video samples of RGB-D videos and skeleton data captured using a Kinect v1, rendering skeletons with 20 joints. There are 12 action classes performed by 40 participants. We evaluate the system using one cross validation, with 20 training subjects and 20 testing subjects. Skeleton data is normalized as in the NTU-RGBD data set.

The model consists of 200 hidden neurons, trained with a batch size of 64. The Adam optimizer [16] is used for training with an initial learning rate of 0.005. The learning rate is reduced by a factor of 10 after 60 epochs. The training was terminated after 3,000 iterations.

Figure 6 shows frames from two similar actions: *mopping* and *sweeping*. To differentiate these actions, the model focuses on the movement of the legs in the sweeping action, and the lowered position of the head in the mopping action. Thus these joints have unexpectedly high sensitivities, in addition to the expected high sensitivities of the arms. The position and orientation of the spine seems to be a differentiating factor in most of the samples of the two actions.

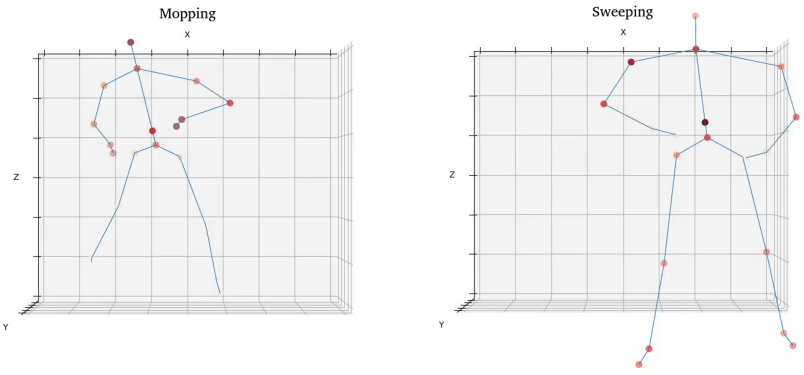

Fig. 6. Sensitivity visualizations of frames extracted from mopping and sweeping actions.

5 Conclusions and Future Work

Recurrent Neural Networks are a state of the art method for modeling sequential data. However, the internal workings of these models are often treated as black boxes by the researchers using them. This paper provide insights into models learned by RNNs through visualization. Sensitivity, reveals 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, throwing may include an upward trajectory of the torso, while kicking may be recognized in part by the starting pose. Using one (or a few) iterations of activation

maximization, we show how the response of a video can be strengthened by exaggerating a motion, thus providing intuitions about the idealized form of an action. At the same time, by running activation maximization to convergence we produce impossible inputs that can fool an RNN. We aggregate these techniques into an interactive visualization tool called SkeletonVis, which we are making available so that RNN developers and users can gain insights into these enigmatic networks.

In the future, we plan 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. Lastly, multi-layer LSTMS are becoming common, and we intend to explore feature abstractions from higher layers.

Acknowledgements

This work was supported by the U.S. Defense Advanced Research Projects Agency and the U.S. Army Research Office under contract #W911NF-15-1-0459.

References

1. https://distill.pub
2. Aafaq, N., Gilani, S., Liu, W., Mian, A.: Video description: A survey of methods, datasets and evaluation metrics (2018), arXiv:1806.00186v1
3. A.Karpathy, J.Johnson, F.Li.: Visualizing and understanding recurrent networks. (2015), coRR, abs/1506.02078
4. Akhter, I., Black, M.J.: Pose-conditioned joint angle limits for 3d human pose reconstruction (2015), cVPR 2015
5. Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B.: A brief survey of text mining: Classification, clustering and extraction techniques. (2017), arXiv:1707.02919v2
6. Bendale, A., Boult, T.E.: Towards open set deep networks (2015), arXiv:1511.06233v1
7. Chiu, C.C., Sainath, T.N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., Kannan, A., Weiss, R.J., Rao, K., Gonina, E., Jaitly, N., Li, B., Chorowski, J., Bacchiani, M.: State-of-the-art speech recognition with sequence-to-sequence models (2017), arXiv:1712.01769v6
8. Cho, K., van Merrienboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. (2014), arXiv:1409.1259
9. Dabral, R., Mundhada, A., Kusupati, U., Afaque, S., Sharma, A., Jain, A.: Learning 3d human pose from structure and motion (2018), arXiv:1711.09250v2
10. Engell-Nrregrd, M., Erleben, K.: Estimation of joint types and joint limits from motion capture data (2009), wSCG 2009
11. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2015), arXiv:1412.6572v3
12. Graves, A., Mohamed, A.R., , Hinton, G.: Speech recognition with deep recurrent neural networks (2013), iEEE international conference on acoustics, speech and signal processing. IEEE, 2013, pp. 66456649.

13. Hossain, M., Sohel, F., Shiratuddin, M.F., Laga, H.: A comprehensive survey of deep learning for image captioning (2018), arXiv:1810.04020v2
14. J.-F. Hu, W.-S. Zheng, J.L., Zhang., J.: Jointly learning heterogeneous features for rgb-d activity recognition. (2015), iEEE Conference on Computer Vision and Pattern Recognition pages 53445352
15. J.Li, Chen, X., E.Hovy, Jurafsky, D.: Visualizing and understanding neural models in nlp (2016), coRR, abs/1506.01066v2
16. Kingma, D.P., Ba, J.L.: Adam: A method for stochastic optimization (2015), arXiv:1412.6980v9
17. Mahendran, A., Vedaldi, A.: Visualizing deep convolutional neural networks using natural pre-images (2016), arXiv:1512.02017v3
18. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images (2015), arXiv:1412.1897v4
19. Shahroudy, A., Liu, J., Ng, T., G.Wang: Ntu rgb+d: A large scale dataset for 3d human activity analysis. (2016), iEEE Conference on Computer Vision and Pattern Recognition, pages 10101019
20. S.Hochreiter, J.Schmidhuber: Long short-term memory (1997), neural computation, 9(8):17351780
21. Simonyan, K., Vedaldi, A., Zisserman, A.: 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. Song, S., Lan, C., Xing, J., Zeng, W., Liu, J.: An end-to-end spatio-temporal attention model for human action recognition from skeleton data (2017), in AAAI Conference on Artificial Intelligence, pages 42634270
23. Strobelt, H., Gehrmann, S., Huber, B., Pfister, H., , Rush, A.M.: Visual analysis of hidden state dynamics in recurrent neural networks. (2016), coRR, abs/1606.07461
24. Tripathy, S.R., Chakravarty, K., Sinha, A., Chatterjee, D., Saha, S.K.: Constrained kalman filter for improving kinect based measurements (2017)
25. Y.Ming, S.Cao, R.Zhang, Z.Li, Y.Chen: Understanding hidden memories of recurrent neural networks (2017), coRR, abs/1710.10777v1
26. Zhang, P., Lan, C., Xing, J., Zeng, W., Xue, J., Zheng, N.: View adaptive recurrent neural networks for high performance human action recognition from skeleton data. arXiv:1703.08274v2 (2017)
27. Zhu, W., Lan, C., Xing, J., Zeng, W., Li, Y., Shen, L., Xie, X.: 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)
28. Z.Qin, F.Yu, C.Liu, X.Chen: How convolutional neural networks see the world 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