# Analysis of the contribution and temporal dependency of LSTM layers for reinforcement learning tasks

Teng-Yok Lee, Jeroen van Baar, Kent Wittenburg, Alan Sullivan
Mitsubishi Electric Research Laboratories
Cambridge, MA, USA

## Abstract

*Long short-term memory (LSTM) architectures are widely used in deep neural networks (DNN) when the input data is time-varying, because of their ability to capture (often unknown) long-term dependencies of sequential data. In this paper, we present an approach to analyze the temporal dependencies needed by an LSTM layer. Our approach first locates so-called salient LSTM cells that contribute most to the neural network output, by combining both forward and backward propagation. For these salient cells, we compare their output contributions and the internal gates of LSTM to see whether the activation of gates precedes the increasing of contribution, and how far beforehand the precedence occurs. We apply our analysis in the context of reinforcement learning (RL) for robot control to understand how the LSTM layer reacts under different circumstances.*

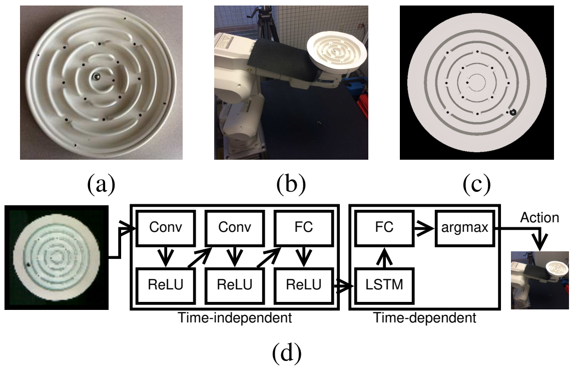


Figure 1. An overview of our robot control system with deep reinforcement learning [8]. (a): Our target task, which is moving a ball from the outer ring to the maze center. (b): The marble maze game on a robot arm. An overhead camera provides observations of the maze. (c) A rendered image of our simulator to mimic the real maze in (a). (d): Architecture of our neural network which represents a control policy for the robot.

## 1. Introduction

When the input data is sequential and time varying in nature, recurrent neural networks like RNN and LSTM (Long short-tem memory) [3] are employed to capture the time dependencies in the data. While various approaches [4, 7] have been proposed to understand these recurrent units, current work has two limitations. First, all output elements of a recurrent unit are usually treated equally, which can obscure the significant data and lead to imprecise findings. Second, existing approaches do not analyze which previous inputs of the time-varying data are contributing to the current output.

In the remainder of this paper we focus on LSTM, especially in the context of Sim2Real, i.e., transfer learning from simulation to real world data. Our goal is to gain an understanding of what role the recurrent units play and use this understanding to improve training and ultimately performance. The task we selected is for a robot to learn a circular ball-in-maze game. Using deep reinforcement learning (RL) and images as input, we train a robot control policy (parameterized by a DNN) to predict five discrete actions: four for clockwise and counterclockwise tilt around the $x$- and $y$-axis, and a fifth *No-Op(eration)* action. The goal is to bring the ball(s) into the center of the maze. For details we refer the reader to [8]. To learn in the presence of complex dynamics the network relies on an LSTM layer (Fig. 1(d)) [3]. We first train the policy on a simulation of the maze (Fig. 1(c)), and then transfer and fine-tune this policy on a real setup (Fig. 1(a, b)).

When transferring our model, we realized that the LSTM part is more difficult to transfer than previous layers. As the layers after LSTM are time-dependent, we are especially interested in how the these layers react to the time-related differences between the simulation and the real setup. These parameters can include the duration to finish an action, frame rates of the captured video frames, the delay to transmit the video frame to the neural network, the computing time to output the control signals, etc. Further, there could be more parameters unknown to us yet. If we can more comprehensively understand the LSTM layer, it can help us limit the parameters to examine.

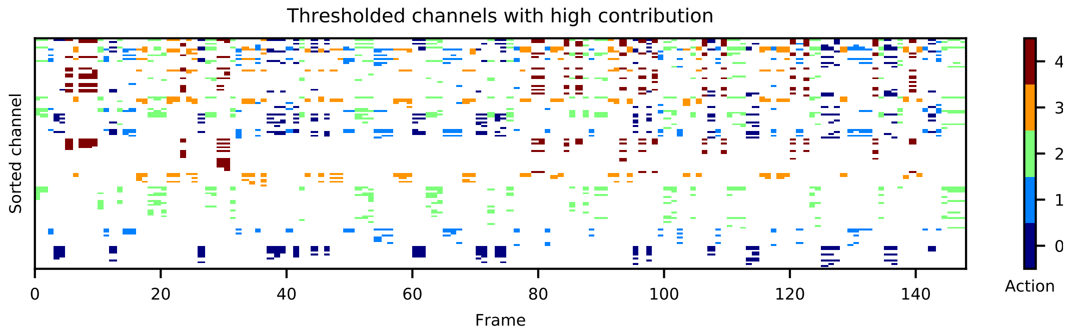


Figure 2. Highly contributing channels of our fine-tuned model on the real robot. The x-axis represents time and the $y$-axis represents the LSTM channels, which are sorted according to the contributed actions. The colors of non-white pixels represent the actions, as indicated by the color bar.

We propose an approach to address the aforementioned shortcomings in the analysis for recurrent units. Our approach first locates so-called salient LSTM channels, which contribute most to the neural network output. By combining both forward and backward propagation, we can estimate the contribution, allowing us to threshold the channels that contribute most. Given the salient channels, we can perform analysis to find the temporal dependencies between the changes of internal status of LSTM and the degree of contribution. By jointly analyzing with other information of our RL settings, we can decompose the LSTM layer into different functional units, based on its contribution in space and time. The preliminary results of our analysis show explainable behaviour that is consistent with our experiments. Then we conclude this paper with future work to further develop our approach.

## 2. Background

Here we first review the concept and terminology of LSTM. We refer readers to the original work by Hochreiter and Schmidhuber [3] for the details. An LSTM consists of two states: a cell state and a hidden state, vectors of length $n_l$, where $n_l$ represents the number of channels. The cell state captures the long term memory. Recurrency is achieved since the previous cell and hidden state are provided along with input data. The cell state is updated according to so-called *gates*. The gates include an *input gate* for the amount of the input to add to the cell state, a *forget gate* for the amount of the current cell state to retain or overwrite, and an *output gate* for the amount of the new cell state to add to the new hidden states. Our later analysis on the temporal dependencies will be based on the gates, as described in Section 4.

## 3. Channel-wise LSTM contribution

The first step of our approach is to locate LSTM channels that contribute more than other channels to the network output, i.e., the action with maximal score. A straightforward approach is checking the hidden state value of each LSTM channel [5, 2] to see which ones have larger magnitude compared to others. However, activation values alone are not sufficient to indicate what contributed to the final actions. Another approach is using backward propagation to compute the partial derivative of the network output w.r.t. the LSTM channels, similar to computing saliency maps [6, 9], where partial derivatives mainly indicate the *sensitivity*. A LSTM channel with high partial derivatives might not contribute much if its activation value is too small.

Instead, the propagation results of both directions should be jointly considered. To make the idea more concrete, it should be noted that in our model, the mapping from the LSTM channels to the action scores is achieved via a fully connected layer, which is a linear transformation. The weights of the fully connected layer can be expressed as a $n_a \times n_l$ matrix $W_a$, where $n_a$ and $n_l$ denote the number of actions and LSTM channels respectively. The relationship, at time step $t$, between the action with maximal score $a_t$, and the LSTM hidden state $h_t$ can be written as:

$$a_t = \underset{i=1\ldots n_a}{\arg\max}\, s[i] = \underset{i=1\ldots n_a}{\arg\max} \sum_{j=1}^{n_l} W_a[i,j]h_t[j]$$

In other words, the score of $a_t$ is the sum of the weighted activations $W_a[a_t, j]h_t[j]$ of all channels $j$. Considering one of the activations $h_t$ or the weight $W_a$ separately is insufficient. If one is large but the other is small, its portion to the sum is still small.

It should be noted that the weight $W_a[a_t, j]$ is also the partial derivative of the score of the $a_t$-th action w.r.t to channel $j$. Namely, the contribution of a channel to the output should consider both its forward propagated activation and the backward propagated derivative. Although this is similar to the algorithm proposed by Balu *et al.* [1] to compute saliency maps, our goals are different. Instead of computing the saliency maps on the spatial domain of the input data, our goal is to measure the contribution of channels of layers within a neural network.

Based on the weighted activation $W_a[a_t, j]h_t[j]$ of every channel $j$, we can measure its contribution to the final output as follows. We first clamp the negative weighted activation to 0 (since the output action is the one with maximal score, and thus prefers positive contributions). By denoting the clamped weighted activation of each channel $j$ as $c_t[j]$, we compute its contribution as $c_t[j]/\sum_j c_t[j]$.

Once the contribution is computed, we can filter the LSTM channels with high contributions. An example is shown in Figure 2, where a 2D color map indicates the top LSTM channels that in total contribute to more than 50% of the action score per time step. Hereafter we use the same criteria to filter the LSTM channels for later figures. In this heat map, the rows are sorted in order to place channels that contributed to the same actions together. This color map reveals several properties of the LSTM layer. The first property is sparsity: at each time step, very few LSTM channels (actually less than 10%) are needed to contribute half of the score. We can also find LSTM channels that contribute to a single action, as there exists rows of a single color.

By jointly analyzing the contribution and the input images, we can identify more properties of the LSTM channels, especially their relationship with the ball locations. In Figure 3 we plot the trajectory used in Figure 2, as the distance of the ball to the maze center over time. The steps correspond to different rings of the maze (Figure 3 bottom). By horizontally aligning the heat map of Figure 2 and the distance plot in Figure 3, we found channels that contribute only when the ball is in a specific region. We pick four channels, filter them, and show the corresponding ball locations in Figure 4. Two channels only contribute when the ball is a a small region, see Figure 4 (a) and (b). The two other channels only contribute when the ball was in a specific ring of the maze, see Figure 4 (c) and (d). Thus, with our measurement of contribution, our approach can group the LSTM channels based on the contributed actions and the corresponding inputs. This allows us to further investigate various properties per group.

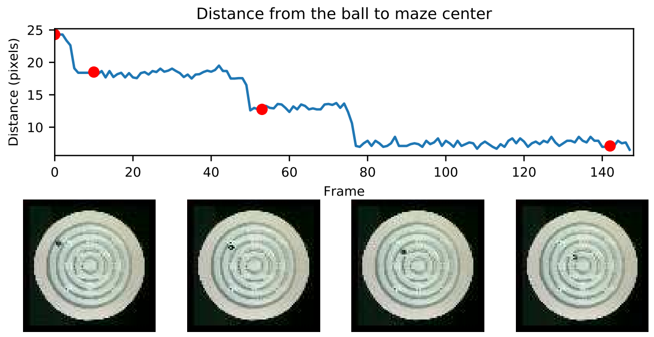
## 4. Time dependency between LSTM gates and contribution

Based on the channel-wise contribution, the next step is to examine the preceding context when the contribution of an LSTM channel increases. We are interested in how the prior LSTM internal states change, especially the *gates*, and how far ahead of the contribution they change. As the gates control the updating of the hidden state, which are transformed to the action scores, we hypothesize that there should exist correlation between the LSTM gates and the contribution.

The gating scheme of LSTM has a few interesting properties. First, each LSTM channel has its own gates to update its corresponding hidden state and cell state. Second, the values of gates are bounded. For the input and output gates, the value range is always between 0 and 1. A high value of input gate means that the new input is being added to the memory, and a higher value of output gates means that the long term cell state starts to contribute to the hidden state.

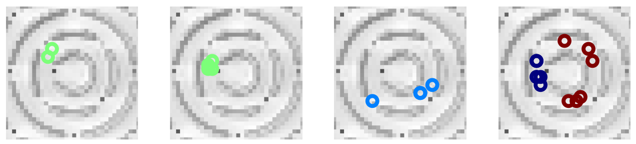
Currently we apply a simple thresholding scheme to see whether there is any offset between the time steps when the gate value goes high and when the channel starts to contribute. First, we find time intervals when the gates' value is always higher than a threshold $\theta_g$. Within a thresholded interval, says $[t_0, t_1]$, we find the first time step $t$ when its contribution is higher than another threshold $\theta_c$. If such a time step $t$ exists, the difference between $t$ and $t_0$ represents the number of time steps needed for the gates' change to be effective. We call this difference *the preceding offset*. An example is shown in Figure 5, which lists the contribution and output cell value of a channel in Figure 2. It can be seen that when this channel is filtered, there indeed exists an offset to when the output gate value starts to increase.

Meanwhile, as suggested in Section 3, the LSTM channels can contribute differently. Thus we need to measure and compare the preceding offset for all channels. Also, the contribution made by a LSTM channel can depend on the spatial contents of the image, such as the location of the ball



Step 0 Ring 1 | Step 10 Ring 2 | Step 53 Ring 3 | Step 142 Ring 4

Figure 3. (top) The distance from the ball to the maze center over time, indicating changing of rings within the maze. (bottom) Images corresponding to the four red points in the top figure. Note that the ball is in different rings of the maze.



(a) (b) (c) (d)

Figure 4. Corresponding ball locations when four channels are filtered in Figure 2. The contributed actions are encoded as the marker color based on the color bar of Figure 2.

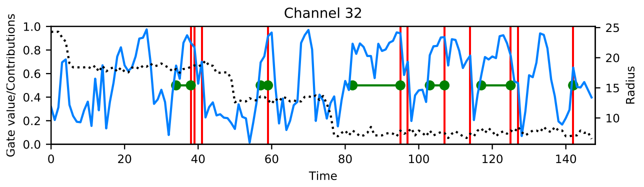


Figure 5. Comparison of the value of the output gate (blue) of a channel and its contributions. The values were measured from the same trajectory as in Figure 2 (black dotted line). The red vertical bars indicate when this channel is filtered. Each green horizontal line indicates the offset from a red bar to the preceding time step when the output gate value exceeds the threshold 0.5.

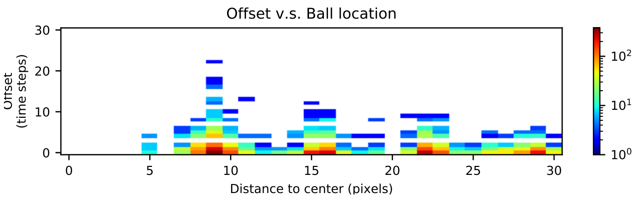


Figure 6. Joint distribution of the preceding offset and the distance from the ball to the maze center from 40 tests. The counts are color-coded according to the color map.

in our application. To comprehensively understand the preceding effect, our approach tests the model multiple times. By changing the initial location of the ball in every test, we can measure the preceding effect from different locations in the maze and see whether there exists a correlation between the length of preceding effect and the ball locations.

Figure 6 shows the joint histogram of the preceding effect and the distance from the ball to the maze center, which was measured according to all LSTM channels from 40 tests with the simulator. The color represents occurrence. Highest occurrence (in red) imply that the distances can be clustered into four groups (one per ring of the maze).

In Figure 6, we can see that when the distance to the maze center becomes small, more samples have longer preceding offsets. For instance, when the distance is 9 pixels, there could be samples with offsets more than 20 time steps, while the offsets at larger distances are 10 time steps or less. This can be explained as follows: the tilting angle to adjust the maze is fixed, therefore the ball will be accelerated faster when closer to the maze center, and thus more time steps are needed to decrease the ball velocity in order to turn. Nevertheless, we are still further evaluating this approach.

This is consistent with the finding of another experiment, where we did not pass the previous hidden states and cell states to the LSTM layer. In other words, the memory of previous time steps was totally ignored. We found that the neural network can still move the ball towards the inner rings. However, when the ball was one ring away from the center, it took longer to move the ball to cross the gate compared to the outer rings. Hence, the LSTM layer utilizes the memory to generate optimized actions when the ball is in the inner ring.

## 5. Conclusion

In this paper, we presented our approach to understand how an LSTM behaves when applied to deep reinforcement learning. By measuring the contribution made by each LSTM channel to the output action, we can understand various roles of the LSTM layer. While conventional approaches require either extra human labeling [2] or manual search of LSTM channels [4, 7], our approach can automatically filter these salient LSTM channels. From those highly contributing LSTM channels, we can further investigate other properties, such as the temporal dependency between the contribution and the changing of integral gates of LSTM. Our preliminary analysis suggests that the measured temporal dependencies are consistent with the findings of other experiments. Our future work is to conduct controlled experiments to comprehensively evaluate the measurement. The goal is to estimate the length of temporal dependency modeled by a trained LSTM, which we expect can assist the transferring of the LSTM between systems with different dynamics.

## References

[1] Aditya Balu, Thanh V. Nguyen, Apurva Kokate, Chinmay Hegde, and Soumik Sarkar. A forward-backward approach for visualizing information flow in deep networks. *arXiv:1711.06221*, 2017.

[2] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and A. T. Aude Oliva. Network dissection: quantifying interpretability of deep visual representations. In *CVPR*, 2017.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computing*, 9(8):1735–1780, 1997.

[4] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv:1506.02078*, 2015.

[5] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):91–100, 2017.

[6] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR*, 2014.

[7] Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M Rush. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 2018.

[8] Jeroen van Baar, Alan Sullivan, Radu Cordorel, Devesh Jha, Diego Romeres, and Daniel Nikovski. Sim-to-real transfer learning using robustified controllers in robotic tasks involving complex dynamics. *arXiv:1809.04720*, 2018.

[9] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833, 2014.