Recurrent Fully Convolutional Networks for Multi-Target Tracking

Edward Li

Introduction

Researchers all around the world are now collecting more and more data and of increasingly varied types. What would have taken analysis by hand just under a decade ago is now possible with automated computer analysis, thanks in part to advancements in machine learning. This has allowed researchers to, instead of collecting kilobytes of easily analyzed data, to instead collect terabytes upon terabytes of highly unstructured data, like video. Due to this increase in data collection, the algorithms used to analyze this data must be as accurate and robust as possible, making it fast and easy for scientists to analyze their data without getting stuck in technical details of a field outside their own.

One particular type of data analysis that is becoming increasingly important, especially in the neuroscience community, is the markerless tracking of animals in video. Oftentimes, kinematics data, like those of head fixed mice, or position and orientation data, like that of free-moving animals, is crucial to the analysis of topics in neuroscience.

Many such algorithms exist for this multi-target video tracking, including DeepLabCut (Mathis et al., 2018) or LEAP [LEAP Estimates Animal Pose] (Pereira et al., 2019). However, these algorithms, using deep fully convolutional neural networks (FCNN), are inherently optimized for the problems of single-frame image analysis, having little capacity for the learning of temporal relationships. This becomes especially apparent when tracked features in a video are temporarily occluded by other objects, producing holes in tracking that can render a given trial unusable or inaccurate. It is apparent that an algorithm, robust to occlusion and temporally aware, would be much better suited for the problem of multi-target tracking in scientific applications.

One other network type that is particularly suited for temporal data are termed recurrent networks. Such a network's output is not only dependent on the current input, but also reliant on previous network inputs. This allows such a network to save a hidden state inside the network, allowing for better predictions through time in multiple domains, like language understanding, weather forecasting, and speech. A popular type of recurrent network is composed of multiple types of recurrent layers called Long Short-Term Memory [LSTM] (Hochreiter & Schmidhuber, 1997). However, these networks are often not well suited to target tracking, as the nature of their inputs is inherently flat. Recently, a new type of layer has been introduced, termed Convolutional LSTM [ConvLSTM] (Shi et al., 2015), which replaces 1-dimensional operations in traditional LSTM with spatial data preserving convolutional layers. This has been successfully used in video segmentation tasks (Valipour et al., 2016). This allows for the combination of both spatially and temporally aware features, a very promising combination for multi-target tracking.

In this study, the LEAP FCNN network will be combined with the ConvLSTM layer to improve occlusion resistance. Mouse kinematics data will be used to train this network. It is predicted that adding the ConvLSTM layer to this tracking network will result in lower mean distance error when paws are occluded.

Hypothesis

If a Convolutional Long Short-Term Memory module is added to the LEAP Fully Convolutional Network, resistance to tracking target occlusion will be significantly increased, allowing for more accurate tracking.

Materials and Methods

Data Collection and Processing

To train a robust occlusion resistant tracking network, data must contain multiple tracking targets as well as multiple points when targets are occluded. To accomplish this, mouse kinematics data was collected. Data is composed of side video feeds (336x280 resolution at 250 fps) of headfixed mice running on top of a stationary wheel (seen in Figure 2). Lighting conditions were kept as dark as possible to ensure mouse comfort. Video data was acquired from an unpublished dataset collected by Richard Warren at Sawtell Lab in Columbia University. Ground truth paw location data was obtained through an existing DeepLabCut (Mathis et al., 2018) FCNN teacher network, previously trained on 1047 hand-labeled frames from the video dataset. Ground truth data consisted of four paw locations, as well as two locations on the tail (midpoint, base point). Data consisted of 16 sessions, each with approximately 300000 frames each. 250 frames were randomly chosen from each session and split into two sets: a training set and a test set (80%/20%).

Results

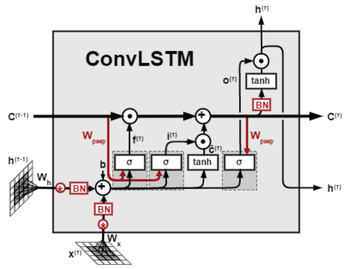

Figure 1: Convolutional Long Short-Term Memory Module Structure
A diagram of a Long Short-Term Memory module. Inputs enter from the bottom, while two hidden states from the previous timestep enter from the left. Data is processed through three sigmoid gates, allowing for the updating of the hidden state and providing output: a forget gate, input gate, and output gate. Outputs exit through the top, while new hidden states exit through the right.

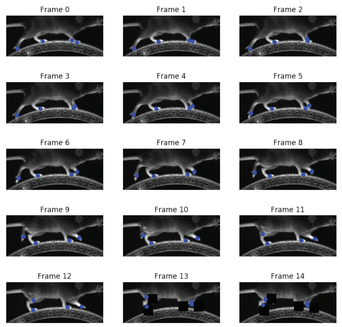

Figure 4: Artificial Occlusion Visualization on Recurrent Network
A series of 15 frames was found in the test set where no natural occlusion occurred. Manual occlusion was applied on the final two frames by placing a black square with side length 20 at the locations of each paw as determined by the ground truth data. It is evident that some tracking is retained, although is not entirely accurate when paws are occluded.

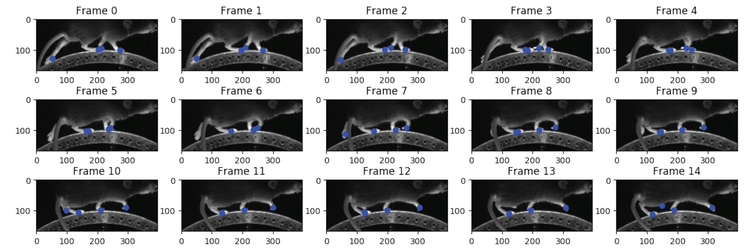

Figure 2: Natural Occlusion Results in Non-Recurrent Network
A series of 15 frames was located in the test set where natural occlusion occurred. Specifically, the tail in this case occludes one of the paws. The non-recurrent FCNN network was run on this series of 15 frames, and paw locations were extracted by finding the maximum confidence value across heatmaps of all 4 paws. Data was plotted with Matplotlib. It is evident that paw tracking is lost on frames 3, 4, 5, 6, 8, 9, and 13, when the back paw is occluded.

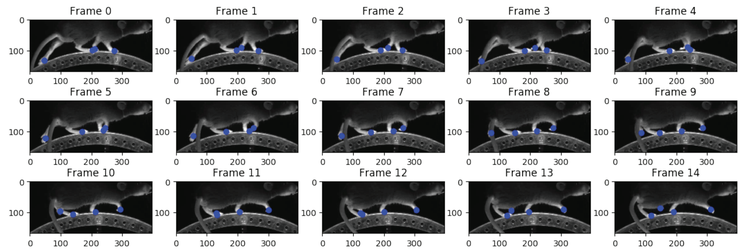

Figure 3: Natural Occlusion Results in Recurrent FCNN Network
A series of 15 frames was located in the test set where natural occlusion occurred. Specifically, the tail in this case occludes one of the paws. The recurrent FCNN network was run on this series of 15 frames, and paw locations were extracted by finding the maximum confidence value across heatmaps of all 4 paws. Data was plotted with Matplotlib. It is evident that paw tracking is retained even when occluded by the tail or the mouse body.

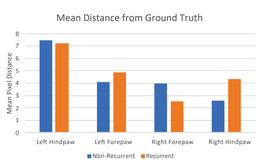

Figure 5: Mean Distance from Ground Truth
All frames in the test set were evaluated on both the recurrent and non-recurrent networks. Mean distance from ground truth was plotted. No statistical difference was found.

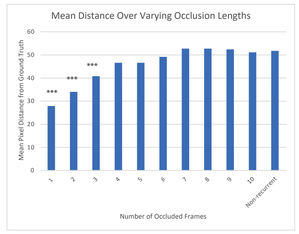

Figure 6: Mean Distance Over Varying Occlusion Lengths
50 sets of 15 frames were tested with varying 20x20 black artificial occlusions over different lengths of time. *** = p<0.001.

Discussion

It is evident from the data that the hypothesis (recurrent networks help with occlusion resistance) is largely supported. This can first be seen by comparing Figures 2 and 3, illustrating a natural occlusion event by the mouse tail. In Figure 2, the non-recurrent network loses paw tracking for a hindpaw during frames 3, 4, 5, 6, 8, 9, and 13, as the fully convolutional network is unable to find a paw location suitably similar to a hindpaw. However, in Figure 3, it is evident that all tracking is retained, even when paws are occluded. This proves extremely promising, as both the location data and the velocity data for each paw seems to be accurate even through occlusion. This implies that the extra recurrent layer present in the recurrent network is trained to act like a pseudo-Kalman filter, keeping an internal representation of paw location and velocity information to substitute when paws are occluded.

This similar trend of occlusion resistance extends to Figure 6, where mean pixel errors are compared between non-recurrent networks and recurrent networks for varying lengths of occlusion. It can be seen that 1, 2, and 3 frame occlusions perform significantly better ($p<0.001$) in recurrent networks than in the non-recurrent network. This strongly quantitatively suggests that a tracking improvement can be obtained through the use of ConvLSTM.

Additionally, recurrent fully convolutional networks appear to perform well when no occlusions are occuring. In Figure 5, the average distance error is compared between the non-recurrent and recurrent networks when no artificial occlusion is applied. No statistical difference was observed when conducting T-tests across non-recurrent and recurrent networks for all four paws. Therefore, non-recurrent and recurrent networks perform comparably well when tracking targets are not occluded. This proves promising, as no performance hit needs to be taken for improved occlusion performance.

However, some potential issues were identified in the performance of the recurrent network that must be addressed in future work. In Figure 6, artificial occlusion was carried out to the maximum length of 10 frames. However, only the first 3 frames were statistically more accurate than a non-recurrent network. Currently, the recurrent network only has a visible window of 15 frames (60 ms). It seems that it would strongly help if a larger window size is used in the future. Additionally, future work must be conducted to improve the occlusion error. While occlusion error was statistically lower than a non-recurrent network, much room still exists for improvement.

Materials and Methods (cont.)

Network Selection

Networks used in this project are based off the LEAP fully convolutional multi-target tracker (Pereira et al., 2019). This network consists of 15 layers, 11 being convolutional, 2 being max pooling, and 2 being deconvolutional. The non-recurrent network was initialized with 16 filters for the first set of convolutional layers, 32 for the next set, 64 for the third set, and 32 for the last set. The last layer contained 6 outputs, one for each paw and two for the tail. To modify the LEAP network into a recurrent network, a ConvLSTM2D layer was added immediately after the second max pooling layer. This layer contained a 32 filters, a kernel size of 3, and "same" padding. Existing layers before and after the ConvLSTM2D layer were wrapped in a TimeDistributed layer from Keras, a deep learning library. 15 timesteps were used to construct the network.

Network Training

Prior to training, input data was normalized with a mean of 0.257 and a standard deviation of 0.288. The non-recurrent network was trained for 20 epochs on a GeForce GTX 1070 graphics card with 8GB of VRAM. Mean squared error loss was used with the non-AMSGrad variant of the Adam optimizer. A batch size of 32 was used, as no timeseries data was used. The recurrent network was trained for 20 epochs, also optimizing for mean squared error loss with the non-AMSGrad variant of the Adam optimizer. However, as 15 timesteps were used per training example, the maximum batch size achievable was 8. Networks were saved after training was completed. No significant improvement in loss or mean distance error was observed after approximately 10 epochs, so networks only trained for 10 epochs were used to minimize overfitting.

Occlusion Analysis Methodology

To analyze occlusion resistance of the recurrent FCNN network, natural occlusions were found in the testing data, where paws would be temporarily occluded by tails or an obstacle. Only the location of the 4 paws were used in occlusion analysis, as the location of the tail midpoint and base point were found to be too noisy to yield any useful data. In order to obtain quantitative data about occlusion performance, 50 collections of 15 frames were selected from the test set. Varying levels of manual occlusion were applied by blacking out a square of 20 pixels surrounding each paw, determined by the ground truth data. Mean distance from ground truth was collected and compared between the non-recurrent and recurrent network by Student's T-Test, conducted as two-tailed and paired.

Conclusion

In conclusion, the use of recurrent fully convolutional networks is extremely promising. It is clear that occlusion resistance is vastly improved when convolutional LSTM layers are added to the LEAP network. This resistance is present both in natural occlusion instances (Figure 2 and 3) and in artificial occlusion (Figure 6). In addition, this network remains highly accurate in normal tasks (Figure 5). This implies that recurrent fully convolutional networks may be extremely useful into the future, allowing researchers around the globe to convert unstructured video into highly structured target location data, ushering in more subtle and deeper analyses enabled by large datasets.

However, future work is still needed, as the occlusion accuracy still remains low, albeit better than a non-recurrent network. In the future, more analysis should be conducted to determine what cases of occlusion are best handled, as well as more experimentation with other classes of FCNN for potentially better performance. Finally, a pipeline still remains to be built, so other researchers can take advantage of this new class of network in their own work.

References

Hochreiter, S., & Schmidhuber, U. (1997). Long Short-Term Memory. Neural Computation. Retrieved March 2, 2019.

Mathis, A., Mamidanna, P., Cury, K. M., Abe, T., Murthy, V. N., Mathis, M. W., & Bethge, M. (2018). DeepLabCut: Markerless pose estimation of user-defined body parts with deep learning. Nature Neuroscience,21(9), 1281-1289. doi:10.1038/s41593-018-0209-y

Pereira, T. D., Aldarondo, D. E., Willmore, L., Kislin, M., Wang, S. S., Murthy, M., & Shaevitz, J. W. (2018). Fast animal pose estimation using deep neural networks. doi:10.1101/331181

Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., & Woo, W. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. Retrieved March 2, 2019.

Valipour, S., Siam, M., Jagersand, M., & Ray, N. (2017). Recurrent Fully Convolutional Networks for Video Segmentation. 2017 IEEE Winter Conference on Applications of Computer Vision (WACV). doi:10.1109/wacv.2017.11