Animal Behavior Prediction with Long Short-Term Memory

Henry Roberts
Department of Computer Science
University of South Alabama
Mobile, USA
htr1721@jagmail.southalabama.edu

Aviv Segev
Department of Computer Science
University of South Alabama
Mobile, USA
segev@southalabama.edu

***Abstract*— A foundational step in the study of any animal is the establishment of an accurate behavioral model. Building a model that is capable of defining and predicting an animal's behavior is critical to advancing ethological theory and research, however many animal models fail to be sufficiently thorough or often do not exist at all. Great pools of data are available for improving these models through recorded video of animals posted on video hosting sites throughout the internet, however these sources are largely left unused due to their sheer quantity being too much for researchers to manually observe and annotate. This paper proposes a method for efficiently converting video of animals at any length into models capable of making accurate behavioral prediction. This predictive model is developed through a data processing pipeline merging an ensemble meta-algorithm for behavior classification with a long short-term memory network for temporal pattern recognition and prediction. The application of this pipeline produced results with a higher degree of predictive accuracy compared to more traditional autoregressive techniques. These findings suggest the method has significant potential as a tool for efficiently developing new models and findings in the study of animal behavior.**

Keywords—LSTM, ARIMA, Pose estimation, Behavior modelling, Behavior prediction

I. Introduction

The cataloguing and quantitative measurement of animal behavior has been a staple of ethology and behavioral ecology since the fields' earliest conceptions in the 19th century [1]. Recent advances in artificial intelligence have further bolstered these fields with the development of machine vision and machine learning based tools for analyzing animal behavior in video [2]. Now with these modern techniques biologists have the opportunity to track and derive new information from animal behavior at a rate faster than ever before. The field of ethology, the study of animal behavior, is no stranger to these modern machine learning techniques with developments like Mathis et al.'s creation of DeepLabCut using deep, residual learning for cutting-edge animal pose estimation [3], or Charpentier et al.'s work using deep neural networks for face recognition in Mandrills [4]. Nor is the field lacking in predictive models, as explained by animal behaviorist Dugatkin in answering why mathematical theories play such a large part in ethology, mathematical models of behavior sometimes produce unexpected predictions. And these predictions can serve as a jumping-off point for empirical tests in ethology. Or if a model predicts correctly with one animal system but not others, it may produce new insights in the differences between species [5].

Many mathematical models have been made in the prediction of organism behavior, and numerous studies have produced developments in their classification and tracking. And yet, comparatively few have sought to use these methods for producing data for the implementation of behavior prediction using modern time series analysis-based models.

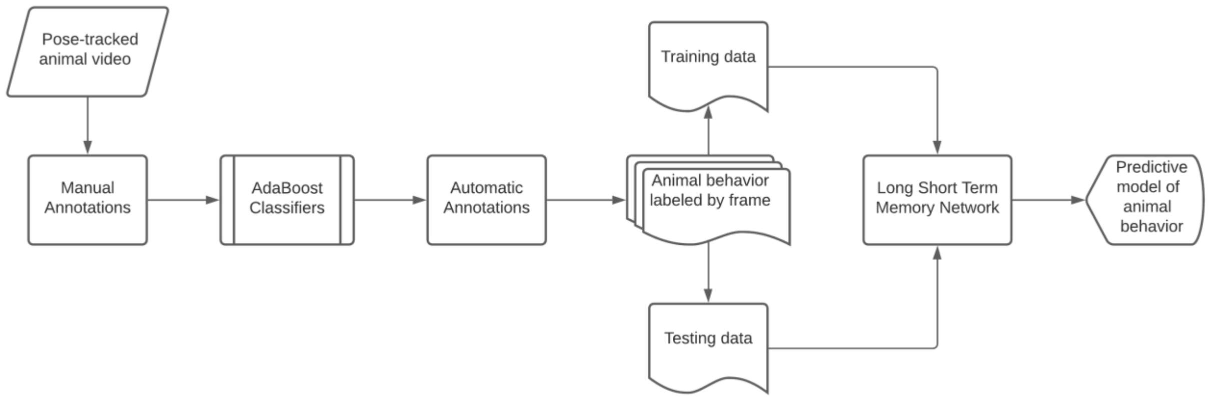
We seek to evaluate the potential for building a functional model for the prediction of an organism's behavior through the training of Long Short-Term Memory (LSTM) networks on observational data of an organism's behavior in video. The scope of this evaluation entails establishing whether such networks can provide accurate predictions of an organism's future behavior based on training on past behaviors and relevant spatial information. After making its predictions for increasingly large lengths of time-steps ahead, we calculate its accuracy in terms of mean squared error and compare it to the results of a statistical analysis model that is made for predicting on time-series data and tasked with the same goal.

This paper aims to contribute to the discussion and development of predictive organism behavioral models using modern machine learning techniques. And in order to do so we propose a potential pipeline for efficiently converting animal video and tracking data to an optimal predictive model.

In Section 2 we aim to contribute to the discussion of behavioral modeling by highlighting recent work carried out related to the field. Section 3 details our proposed predictive behavior pipeline and steps through each element involved in its creation and evaluation. Section 4 details the evaluation itself: the dataset used, the predictions made, the results of our work, and discussion on its implications. Lastly Section 5 describes our conclusions on the study and details how we will build upon it in the future.

II. Related Work

The primary focus of our study is the evaluation of deep learning and machine vision integration to form an efficient pipeline for analyzing and predicting organism behavior. Though the studies on machine vision based prediction



Pose-tracked animal video
Manual Annotations
AdaBoost Classifiers
Automatic Annotations
Animal behavior labeled by frame
Training data
Testing data
Long Short Term Memory Network
Predictive model of animal behavior

Fig. 1. Flowchart displaying each step of the predictive behavior pipeline.

modeling in behavioral ecology are limited, we do see a strong contribution from Rosenthal et al. [6] in their work in which they evaluate the complex networks of interaction between schooling fish. This is accomplished by tracking their body movements and locations in the school, calculating the field of vision for every fish within it, and using this data to find the connection between sensory input from other members of the school and their own motor response while collectively evading threats. Building out these connections as a functional model of networked behavior allows them to analyze this hidden communication within the group and accurately predict complex changes in the behavior of the school before they occur [6]. While aligning with our own study's goal of predicting organism behavior before it occurs, their methodology approaches the analysis from the bottom up, functionally modelling the individual experience of each fish in relation to its environment. In our approach we take a comparatively top down view, analyzing the animal's behavior as an external observer and training a time-series analysis model for prediction based on those observations.

The application of machine learning techniques in identifying pose in video has been a fast evolving field, developing rapidly over the past six years [7]. It began with DeepPose, an application of deep learning to human pose estimation in two dimensions [8]. Since then numerous other deep-learning pose estimators have followed, including frameworks using motion features [9], integrated training using a convolutional network and a graphical model [10], using convolutional networks for advancing object localization [11], establishing a benchmark for pose estimation and tracking [12], and human pose estimation with stacked hourglass networks [13]. DeeperCut by Insafutdinov et al. marked a milestone in producing accurate multi-person pose estimation using graph cutting guided by deep learning based body part detectors [14].

The first powerful example of machine vision for ethology is the aforementioned DeepLabCut tracking software [3]. Developed in part through the benchmarking of a selection of the features used by DeeperCut, this 2D and 3D pose tracking tool can be applied to a wide range of organisms in a multitude of different environmental contexts. From a horse to a fly to just the limbs of a mouse, DeepLabCut leverages the power of deep, residual networks to maintain accurate pose tracking on them all. The inclusion of DeepLabCut in a future predictive pipeline could lead to a greater degree of generalizability in animal tracking. However, it differs from our proposed predictive behavior pipeline in that it stops at the pose tracking. Any behavioral annotation must be carried out manually or using external software.

In the proceeding section we will detail our emphasis on behavioral classifier generation and automated behavior labeling for the production of behavioral time-series data.

III. Predictive Behavior Pipeline

In this study we implemented an integration of machine learning techniques and evaluated their potential as a pipeline for predicting animal behavior. The components of this pipeline are displayed in Fig. 1. The first stage of this framework utilized a type of machine learning meta-algorithm called AdaBoost, or adaptive boosting, used in combination with decision stumps as weak learners. We first manually annotated the behavior of our subjects frame by frame within a small portion of our pose-tracked video based on established ethograms. After this labeling we then extract the features related to each frame of video and the subjects within, building decision stump classifier functions. The adaptive boosting algorithm prunes for the resulting functions that are most accurate in replicating the manually labeled frames of behavior. The end product is a classifier for each labeled behavior that is then applied to the remaining unlabeled footage, automatically annotating the behaviors exhibited therein. Once this was done, the behavioral annotations and classifier function results were paired frame by frame to build a behavioral dataset.

Using this dataset, we were able to integrate the automated classification methodologies of the adaptive boosting algorithm with the forecasting power of Long Short-Term Memory networks. This allowed us to pipeline from unlabeled video of an animal as input to a predictive model of the animal's behavior as output. The LSTM network is an ideal architecture for this process as it was made for handling the temporal element of our behavioral dataset and could deal with the vanishing gradient problem [15], [16] typically seen in other recurrent neural networks.

Lastly, we implemented an autoregressive integrated moving average (ARIMA) model to analyze and predict using the generated behavioral annotations. The ARIMA model is well suited for fitting temporal data and forecasting future steps in time series, so it was a good fit for our behavioral data. As a well-established form of time series analysis, the ARIMA model's predictions were then compared against the LSTM's to evaluate the network's predictive significance [17]–[19].

A. Adaptive Boosting

The term "boosting" refers to a meta-algorithmic approach to producing accurate classifications using the selective combination of several less accurate classifiers, commonly called weak learners [20]. The basic form of a boost classifier is as follows:

$$F_T(x)=\sum_{t=1}^{T} f_t(x), \tag{1}$$

in which object x is taken as a input to each weak learner function f_t which then outputs a value representing the determined class of the object [21]. An output hypothesis, $h(x_i)$ is made for each sample of a training dataset by every weak learner. Every t iteration the coefficient α_t is applied to a chosen weak learner in order to make the resulting classifier's sum training error E_t minimal as shown below:

$$E_t = \sum_i E[F_{t-1}(x_i) + \alpha_t\, h(x_i)]. \tag{2}$$

In the above equation $F_{t-1}(x)$ represents a classifier that was boosted in the prior t iteration, while $E(F)$ is the error function being implemented [21]. For each of these iterations a weight $w_{i,t}$ equaling the error is assigned to the sample. Here $\alpha_t h(x) = f_t(x)$, which is the weak learner function being evaluated for inclusion in the finished boosted classifier.

In this study a form of boosting called Gentle AdaBoost is used which differs from other boosting algorithms in that its weak learning functions f_t are bounded in their step size rather than reducing the greatest amount of test error at each step [20]. The weak learners are picked to minimize:

$$\sum_i w_{t,i}\left(y_i - f_t(x_i)\right)^2, \tag{3}$$

without the minimizing coefficient α_t being used. The end result is a fast and highly generalizable boosting algorithm that is well suited for efficiently classifying through large datasets [20].

B. LSTM

Traditional artificial neural networks rely on a feedforward approach in which information is only passed forward for decision making [22]. Recurrent neural networks (RNN) differ in that they implement feedback, enabling many cycles of information processing. This looping architecture allows for the network to retain information from previous input data, allowing it to incorporate time as a dependent factor in the information that it processes [23]. Most of these networks, however, suffer from the vanishing or exploding gradient problem in which the error signal that shifts neuron weights drops to very small values or expands to extremely large values. Long Short-Term Memory networks are a form of RNN that overcome this problem through the use of additional memory block architecture featuring constant error carousels (CEC) and forget gates [22]. This ability to retain backpropagated errors through layers of the network without vanishing or exploding gradients makes the LSTM model suitable for recognizing the connections of events that happen upwards of thousands of time steps separate from each other [22].

The hidden layer of the LSTM is composed of memory block units that contain memory cells and a pair of gating units that multiplicatively influence all inputs and outputs to the memory cells in their block. The memory cells within each block contain CEC units and their activation is called the cell state. To prevent the vanishing error problems when there is a gap in input or error signals, the CECs are able to keep the local error backflow constant, showing no exponential growth or rapid shrinking. If the cell state is near zero then the gates are closed, protecting the memory block from noise and keeping the rest of the network unaffected [24].

Each step of the LSTM is evaluated in terms of units of discrete time t and entails a forward pass and backward pass. The forward pass updates every memory block and the backward pass recomputes the error function for all the weights [24]. The activation for each input gate y^{in} and output gate y^{out} is stated below.

$$net_{out_j}(\mathrm{t})=\sum_{\mathrm{m}} w_{out_j m} y^m(t\text{-}1), \tag{4}$$

$$y^{out_j}(t)=f_{out_j}\left(net_{out_j}(t)\right), \tag{5}$$

$$net_{in_j}(t)=\sum_{m} w_{in_j m} y^m(t\text{-}1), \tag{6}$$

$$y^{in_j}(t)=f_{in_j}\left(net_{in_j}(t)\right). \tag{7}$$

Here f is a range (0,1) sigmoid function, m is the source unit index, and v is the index of each memory cell within each memory block j. The state of each memory cell $s_c(t)$ is computed by a squashed, gated input to the prior time step $s_c(t\text{-}1)$ ($t > 0$):

$$net_{c_j^v}(t)=\sum_m w_{c_j^v m} y^m(t-1), \tag{8}$$

$$s_{c_j^v}(t)=s_{c_j^v}(t-1)+y^{in_j}(t)g\left(net_{c_j^v}(t)\right), \tag{9}$$

with $s_{c_j^v}(0)=0$. Here c_j^v represents the memory cell of index *v* for the memory block of index *j*. The gated input is squashed by a (-2, 2) range, centered logistic sigmoid function *g*. Each cell output y^c is computed using the output squashing function *h*, a centered sigmoid with range (-1,1), to squash each cell's internal state s^c before multiplying it by the memory block's output gate activation y^{out} [24]:

$$y^{c_j^v}(t)=y^{out_j}(t)h\left(s_{c_j^v}(t)\right). \tag{10}$$

Using a layered network topology with a standard input layer, a hidden layer of memory blocks, and a standard output layer, the formula for the output units *k* are:

$$net_k(t)=\sum_m w_{km} y^m(t-1), y^k(t)=f_k\left(net_k(t)\right), \tag{11}$$

in which *m* is the range of all units contributing to the output units, i.e. the memory cells within the memory blocks in the hidden layer and the input units [24]. The squashing function f_k is also a logistic sigmoid of range (0,1). From this base we can then extend the LSTM model to use adaptive forget gates, which implements both immediate and gradual resets of memory blocks to zero as their contents grow out-of-date in the processing of long time series data. This is done by replacing the CEC constant weight of 1 with an adaptive forget gate activation y^φ. This gate activation is calculated as:

$$net_{\varphi j}(t)=\sum_m w_{\varphi,m}\, y^m(t-1) \ ; \tag{12}$$

$$y^{\varphi_j}(t)=f_{\varphi_j}\left(net_{\varphi_j}(t)\right). \tag{13}$$

In this equation $net_{\varphi j}$ is the network's input to the forget gate. f_{φ_j} is a squashing function using the logistic sigmoid with range (0,1). The output $y^{\varphi_j}(t)$ then serves as a multiplicative weight of the self-recurrent connection in the computation of the state of each memory cell s_c, with the revised equation (when t > 0) as follows:

$$s_{c_j^v}(t)=y^{\varphi_j}(t)\ s_{c_j^v}(t-1)+y^{in_j}(t)g\left(net_{c_j^v}(t)\right), \tag{14}$$

with $s_{c_j^v}(0)=0$. The initial bias weights for the input gates and output gates are negative and for the forget gates they are positive. This means that the forget gate activation at the start of the training phase will be nearly 1.0, making its output equivalent to an LSTM memory cell without the forget gate architecture. As the training continues it then learns to reset memory blocks to zero [24].

This ability of the LSTM model to modulate the impact of data moving through its cells makes it apt for the processing and prediction of semi-stochastic data over both small and large quantities of time-steps. To that end, the behavioral dataset prepared using the gentle adaptive boosting algorithm makes for an ideal fit in the model's architecture. Preparing the data for multi-step forecasting allowed us to design an LSTM capable of predicting many frames into the future from any starting set of time-steps. This preparation entailed restructuring the dataset of behavior annotations such that for each time-step in the dataset, the behavior classified at that timestep was represented as an array of future behaviors from that point. In this preparation, the length of the array determines the number of frames forward the LSTM will output in its predictions.

3.3 Autoregressive Integrated Moving Average Model

Having an established model prepared for time-series statistical analysis ensured that after training and testing the LSTM network we were able to evaluate its significance as a tool for behavioral prediction. The autoregressive integrated moving average (ARIMA) model was selected based on our need to analyze and forecast from non-stationary time-series behavioral annotations [19].

The ARIMA model predicts future points in a dataset by applying linear functions between random error and recent prior points in time. The equation for this application is:

$$\begin{aligned} y_t &= \theta_0+\phi_1 y_{t-1}+\phi_2 y_{t-2}+\cdots+\phi_p y_{t-p} \\ &+\varepsilon_t-\theta_1\varepsilon_{t-1}-\theta_2\varepsilon_{t-2}-\cdots-\theta_q\varepsilon_{t-q} \end{aligned} \tag{15}$$

in which y_t is the value at the given time-step *t* and ε_t is the respective random error. *p* and *q* are integers that serve as the orders of the model and the parameters are $\phi_i (i = 1,2, \dots, p)$ and $\theta_i (j = 0,1,2, \dots, q)$ [17], [19]. Using an automatic optimization tool in python we found optimal parameters for the ARIMA model fit to the behavioral annotations. From there we made predictions of length equivalent to the LSTM model's and found the mean squared error (MSE) for both sets of predictions. Comparing these MSE values we described their differences, distributions, significance in terms of Pearson's correlation [25] and effect size in terms of Cohen's d [26]. As an effect size, Cohen's d gives the standardized mean between two datasets. It is found as the difference between each dataset's mean values divided by a standard deviation of the combined data.

IV. Experimental Evaluation

We processed video and tracking data in JAABA, the Janelia Automatic Animal Behavior Annotator. JAABA is a software

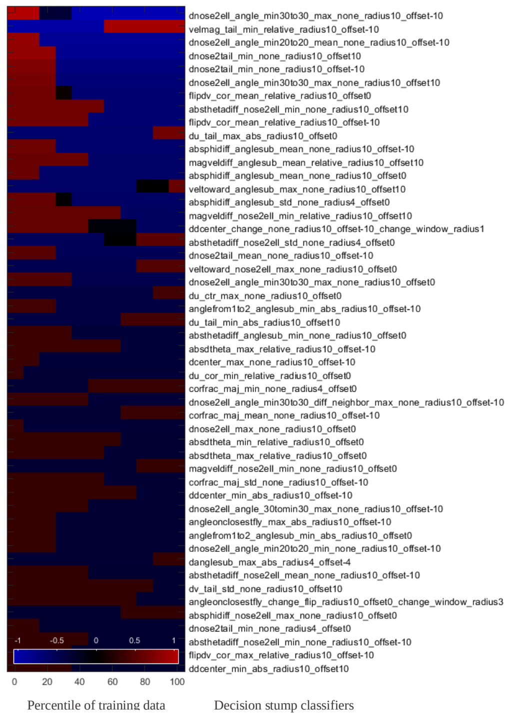


Fig. 2. A visual depiction of the behavioral classifier used for automatically labeling when a fly is chasing another fly.

implementation of adaptive boosting built for this task, using animal behavior data prepared by the Branson lab at HHMI Janelia Farm [27]. Combining the annotated behaviors and the results of the classifier functions we then built an LSTM network to train on the data and make future behavior predictions. These results were then compared to results from a standard predictive model to evaluate their significance.

A. Data Collection

The dataset used in this experiment was made by the Branson lab and the Janelia Farm Fly Olympiad Team for work on a neural activation screen of the Rubin GAL4 collection [27]. The data consists of two files. The first is a 30 frame-per-second video of 10 male and 10 female adult Drosophila melanogaster exploring a fly bowl. This video is comprised of 23,972 frames total. The second is a .mat file containing pre-computed trajectories of each fly in a JAABA compliant format [27].

B. Adaptive Boosting & JAABA

We selected JAABA, for executing the first stage of the processing pipeline. Many applications have been developed for the purpose of tracking the behavior of animals observed in video. One commonly shared drawback with these programs is that they rely exclusively on the manual input of observed behavior at each frame of video. JAABA overcomes this limitation using an implementation of the aforementioned adaptive boosting algorithm called Gentle AdaBoost [27].

Feature sets are created within JAABA after manually annotating the behaviors exhibited in a small portion of frames. These feature sets are used by decision stump classifiers to relabel the behaviors that were manually annotated. The adaptive boosting algorithm combines the most accurate amongst these in order to produce behavior classifiers that are then applied to the full set of video frames.

A visualized example of one such classifier can be seen in Fig. 2. The strings in the right column are features relative to the window view of the tracked subject. The decision stumps make binary decisions based on threshold values for each window feature. Decisions on features that align with the manually labeled data receive weight to give them greater impact in the cumulative classification decision. In Fig. 2 the classifiers are listed in descending order, with the heaviest on top. The left column in the figure visually represents the application of the behavior classification across the full video based on each window feature independently. In JAABA's GUI the automated behavioral labeling and the system's confidence levels are displayed to the user in a timeline view of the video's frames and the user can then correct and update the classifiers and their automatic labeling. Through this iterative process users are able to rapidly annotate thousands of frames of behavior with minimal manual input [27].

C. Approach

We set out to evaluate the potential for behavior prediction using a pipeline integrating the classifier generation techniques of JAABA and the time series sensitive predictive potential of LSTMs. By first processing trajectory tracked Drosophila melanogaster video in JAABA we are able to generate behavioral classifiers. The raw output of these classifier functions is then paired with the automatically annotated behavior of the Drosophila melanogaster for each frame of the video. The resulting two-dimensional dataset is then divided into training and testing data for an LSTM network. From here the framework can be evaluated in terms of its accuracy in predicting the future behavior of the fly at increasingly distant lengths of frame. We then compare the MSE of the LSTM's predictions to the MSE of predictions from ARIMA, our established statistical model for time series analysis.

While ARIMA and LSTM are both machine learning models fit to process and make predictions on time-series data, their structures differ significantly. ARIMA implements an integration between autoregression and moving average models to fit data and predict future time-steps. It evaluates each time-step as the difference between the current and prior step. In doing so it is able to make the data stationary for autoregression and make predictions with its lagged values [18]. This contrasts with the LSTM model, which fits its data and makes predictions using deep learning. The model is an enhanced version of a recurrent neural network in which constant error carrousels and forget gates enable the network to account for contexts between time-steps both near and far from each other[15], [28], [29].

D. Experiments

After loading the Drosophila melanogaster dataset into JAABA, one fly was selected and its behavior was manually labeled for 7000 frames. The annotated behaviors were based on a common Drosophila melanogaster ethogram [30] and featured: grooming, chase related movement, non-chase related movement, and remaining still. With the 7000 manually annotated frames JAABA then developed classifiers for each behavior and annotated the behavior of the fly in the remaining

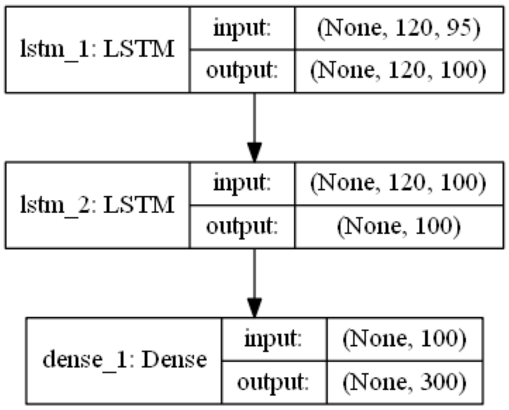


Fig. 3. A visualization of the LSTM network built for this study. The first layer receives the prior 120 time-steps, each containing 95 variables from the JAABA produced behavioral dataset. The 100 unit hidden layers form the inter-layer outputs. Finally, the dense layer outputs 300 sequential time-step predictions.

frames. The results of the behavior classifier functions and the annotated behaviors were then paired frame by frame and exported as a single two-dimensional dataset.

This dataset was then prepared for use with the LSTM network. To do this, the data was first loaded into a Jupyter notebook running Python 3.6. The four annotated behaviors were then integer encoded as 0, 1, 2, and 3. Both the integer encoded behaviors and the classifier function results were then standardized by scaling their values to a 0 to 1 range. From here each frame in the dataset was paired with a sequence of values representing the annotated behaviors for a given number of frames in the future. This quantity of frames can be configured in order to evaluate predictions closer or further out from any given frame. For this experiment its maximum prediction range was set to 300 frames out, which at 30 frames per second equals 10 seconds into the future. The standardized data was then divided with an 80/20 split into training and testing sets.

A stacked LSTM model was then defined for use in multi-step forecasting using the python deep learning API library Keras running with TensorFlow. The model was comprised of two stacked hidden layers of 100 units each and an output layer that predicts a quantity of frames as configured in the data preparation stage. In this case, 300. The model also uses a configurable input shape for determining how many prior frames it processes for each set of predictions. This was set to 120, meaning that it evaluates the 120 prior frames in the context of the model in order to predict the subsequent 300 frames. Fig. 3 provides a visualization of this network's arrangement. The model was then compiled with the Adam version of stochastic gradient descent for its optimizer and mean square error as its loss function. We then fit the model to the training data, undergoing 10 epochs with a batch size of 100, and evaluated its predictive ability.

In order to evaluate the significance of the LSTM model's predictions, the autoregressive integrative moving average model was then constructed. An auto ARIMA function was used to find optimized (p, d, q) values of (1, 1, 2), after which the model was trained on 80% of the JAABA behavior annotations. Once fit, it stepped through each frame of the remaining test data, predicting 300 frames out for each. These predictions were compiled, and their mean squared errors were found. These were then compared to the mean squared errors of the LSTM's predictions to evaluate the model's relative predictive significance. This also allowed us to analyze the LSTM's effectiveness in near future versus further out behavior predictions.

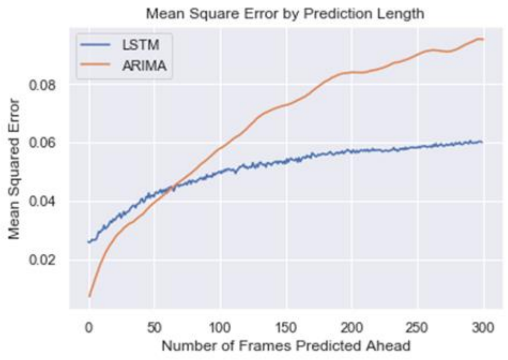


Fig. 4. The mean square error of the LSTM and ARIMA models based on the number of frames ahead they made predictions on. The ARIMA model's MSE surpasses the LSTM network at the 60th frame prediction.

E. *Results and Discussion*

As seen in Fig. 4, on average the LSTM network predicted with a higher degree of accuracy than the ARIMA model. When predicting behavior within two seconds of the current time-step, i.e. from 1 to 59 frames out, the ARIMA model demonstrates less error. After the 2nd second, however, and through to the maximum 10 seconds, the LSTM model outperforms, maintaining a consistently lower MSE.

The distributions of both models' MSE scores were tested and found to be non-normal. In Fig. 5 the distributions are displayed as a box and whisker plot. The boxes cover the second and third quartile values of each dataset, representing the middle 50% of each model's respective MSE values. The line is each set's median. The whiskers indicate the scope of the data, with the LSTM flier points representing outliers. From this depiction we can see that the ARIMA model demonstrates greater variability and overall error in its predictions compared to the LSTM model, which is more consistent in its lower error rate aside from a few even lower outliers. The Pearson's correlation between models was 0.984, suggesting a high degree of correlation. The p value was 4.28e-58, implying significance in the difference of the models' predictions with considerable certainty. Lastly, Cohen's d was found to be 0.984, indicating a large effect size.

These results supported our intuition that the LSTM network's ability to interpret context across large gaps in time would make it uniquely suited for analyzing and predicting natural processes like animal behavior. Though this study was relatively small in scope, the results of its evaluation serve as a positive indication for the expansion of our proposed pipeline for efficiently building behaviorally predictive models from video up.

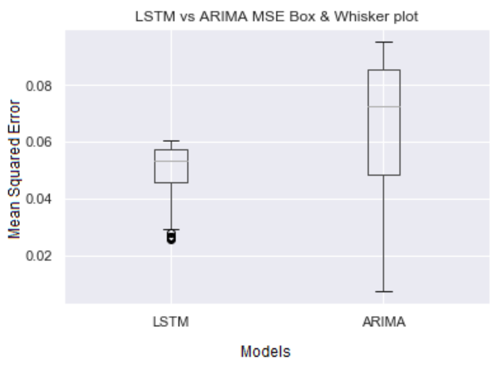


Fig. 5. Box & Whisker plot comparing the MSE range between the LSTM network and ARIMA model. We can see the LSTM has less variability in its error rate and lower MSE compared to the more traditional ARIMA.

V. Conclusions

In this study we proposed a pipeline approach for efficiently developing predictive behavioral models using a confluence of machine learning tools. We evaluated our model's accuracy in prediction and its significance against a much longer standing time-series analysis statistical model. The results of testing our proposed pipeline showed promise in that the LSTM network, trained on the JAABA annotated frames of animal behavior and classifier function results, was able to outperform the ARIMA model, maintaining a significantly lower MSE for behavioral predictions of the Drosophila melanogaster from three to ten seconds into the future. Opportunities for leveraging the advantage offered to us by big data are all around us. By developing efficient pipelines and frameworks for working with data of this scale and complexity we can collectively lower the barrier to entry on the potential insights that await us.

To that end, we plan to take action in response to our pipeline evaluation's positive results. In future work we will expand the scope of the study, aiming to fully generalize the requirements for suitable animal video and examine additional opportunities for automation in the workflow. We would also like to examine opportunities to branch this pipeline into other domains where temporal event analysis and prediction applied to large datasets may yield useful results. Expanding the scope to make gestalt predictions on the interactions within biological systems could lay the way for new insights in their function. And increasing the resolution on behavioral classification until it is near 1:1 with the subject organism could lead to new potentials in biological modeling.

References

[1] Z. Magić, "The nobel prize in physiology or medicine 2009," *Vojnosanitetski Pregled*, vol. 66, no. 11. p. 861, 2009, doi: 10.1007/bf02867268.

[2] T. D. Pereira *et al.*, "Fast animal pose estimation using deep neural networks," *Nat. Methods*, vol. 16, no. 1, pp. 117–125, 2019, doi: 10.1038/s41592-018-0234-5.

[3] A. Mathis *et al.*, "DeepLabCut: markerless pose estimation of user-defined body parts with deep learning," *Nat. Neurosci.*, vol. 21, no. 9, pp. 1281–1289, Sep. 2018, doi: 10.1038/s41593-018-0209-y.

[4] M. J. E. Charpentier *et al.*, "Same father, same face: deep learning reveals selection for signaling kinship in a wild primate," *Sci. Adv.*, vol. 6, no. 22, p. eaba3274, May 2020, doi: 10.1126/sciadv.aba3274.

[5] L. A. Dugatkin, *principles of animal behavior, 4th edition*. University of Chicago Press, 2019.

[6] S. B. Rosenthal, C. R. Twomey, A. T. Hartnett, H. S. Wu, and I. D. Couzin, "Revealing the hidden networks of interaction in mobile animal groups allows prediction of complex behavioral contagion," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 112, no. 15, pp. 4690–4695, Apr. 2015, doi: 10.1073/pnas.1420068112.

[7] M. W. Mathis and A. Mathis, "Deep learning tools for the measurement of animal behavior in neuroscience," *Current Opinion in Neurobiology*, vol. 60. Elsevier Ltd, pp. 1–11, Feb. 01, 2020, doi: 10.1016/j.conb.2019.10.008.

[8] A. Toshev and C. Szegedy, "DeepPose: human pose estimation via deep neural networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Sep. 2014, pp. 1653–1660, doi: 10.1109/CVPR.2014.214.

[9] A. Jain, J. Tompson, Y. LeCun, and C. Bregler, "MoDeep: a deep learning framework using motion features for human pose estimation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9004, pp. 302–315, doi: 10.1007/978-3-319-16808-1_21.

[10] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Advances in Neural Information Processing Systems*, 2014, vol. 2, no. January, pp. 1799–1807.

[11] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, vol. 07-12-June, pp. 648–656, doi: 10.1109/CVPR.2015.7298664.

[12] M. Andriluka *et al.*, "PoseTrack: a benchmark for human pose estimation and tracking," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5167–5176, doi: 10.1109/CVPR.2018.00542.

[13] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," 2016. doi: 10.1007/978-3-319-46484-8_29.

[14] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: a deeper, stronger, and faster multi-person pose estimation model," May 2016, vol. 9910 LNCS, pp. 34–50, doi: 10.1007/978-3-319-46466-4_3.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[16] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *30th International Conference on Machine Learning, ICML 2013*, Nov. 2013, no. PART 3, pp. 2347–2355, Accessed: Jun. 01, 2020. [Online]. Available: http://arxiv.org/abs/1211.5063.

[17] M. Khashei and M. Bijari, "A novel hybridization of artificial neural networks and arima models for time series forecasting," in *Applied Soft Computing Journal*, 2011, vol. 11, no. 2, pp. 2664–2675, doi: 10.1016/j.asoc.2010.10.015.

[18] L.-M. Liu, G. B. Hudak, G. E. P. Box, M. E. Muller, and G. C. Tiao, "Forecasting and time series analysis using the sca statistical system," 1992. Accessed: Jun. 12, 2020. [Online]. Available: http://scausa.com/SCADocs/SCAFTS_V1.pdf.

[19] P. G. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, Jan. 2003, doi: 10.1016/S0925-2312(01)00702-0.

[20] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Mach. Learn.*, vol. 37, no. 3, pp. 297–336, 1999, doi: 10.1023/A:1007614523901.

[21] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997, doi: 10.1006/jcss.1997.1504.

[22] J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Networks*, vol. 61, pp. 85–117, 2015, doi: 10.1016/j.neunet.2014.09.003.

[23] P. Le-Hong and A. C. Le, "A comparative study of neural network models for sentence classification," in *NICS 2018 - Proceedings of 2018 5th NAFOSTED Conference on Information and Computer Science*, 2019, pp. 360–365, doi: 10.1109/NICS.2018.8606879.

[24] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with lstm," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, Mar. 2000, doi: 10.1162/089976600300015015.

[25] J. Lee Rodgers and W. Alan Nice Wander, "Thirteen ways to look at the correlation coefficient," *Am. Stat.*, vol. 42, no. 1, pp. 59–66, 1988, doi: 10.1080/00031305.1988.10475524.

[26] J. Cohen, *statistical power analysis for the behavioral sciences*. Taylor & Francis, 2013.

[27] M. Kabra, A. A. Robie, M. Rivera-Alba, S. Branson, and K. Branson, "JAABA: interactive machine learning for automatic annotation of animal behavior," *Nat. Methods*, vol. 10, no. 1, pp. 64–67, 2013, doi: 10.1038/nmeth.2281.

[28] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994, doi: 10.1109/72.279181.

[29] J. F. Kolen and S. C. Kremer, "Gradient flow in recurrent nets: the difficulty of learning longterm dependencies," in *A Field Guide to Dynamical Recurrent Networks*, 2010.

[30] T. Jonsson, E. A. Kravitz, and R. Heinrich, "Sound production during agonistic behavior of male drosophila melanogaster," *Fly (Austin).*, vol. 5, no. 1, pp. 29–38, 2011, doi: 10.4161/fly.5.1.13713.