Supervised classification of fly behaviors from pose tracking data

Adam Elodie, Des Courtils Philippine, Eyraud Pauline

***Abstract*—Animal behavior analysis is a key to behavioral neuroscience research, and its precise quantification is of paramount importance. However, manual behavior tagging turns out to be quite laborious and time consuming. In this paper, we explore some models used to classify fruit flies behaviors on video sequence, in an attempt to automate this task. Our models are trained on pose estimation data provided by DeepLabCut software, an efficient tool to perform markless pose estimation of animals. Models were optimized and their performance in predicting accurately the behavior of a fly on a test video was assessed.**

I. Introduction

Recently, the apparition of new tools in computer vision has opened a new field of possibilities for animal behavior monitoring. The applications are exciting in various fields such as neuroscience, and announces a new era in quantification of behavior. In this paper, the aim is to train a model to be able to classify fly behavior based on DeepLabCut poses across time. The study of animal behavior in research is used for instance to analyze the way animals adapt their posture in response to changes in their environment. The DeepLabCut software was presented by Mathis et al. in 2018 [3] and it prevents the use of invasive markers in animals for lab study and reached state of the art performance. Data from pose estimation of flies were provided by DeepLabCut and constitute the dataset for this project.The data given consist of folders in which there are DLC predicted poses and annotation files.

This project consists of analyzing these DeepLabcut files and implementing a model able to predict accurately fly behavior(s) per time frame.

II. Exploratory Data Analysis

Our data set consists of 145 pose tracking files from DeepLabCut videos treatments, along with hand annotated files for each time frame. Each file contains x and y positions for 25 body parts, along with the probability of this body part being occluded across 720 time steps. Each video was sampled at a rate of 20Hz, resulting in a total length of 36 seconds. In a nutshell, DeepLabCut extracted for us workable features from videos. The annotation file contains binary variables (behavior present/absent in frame t) over time. The seven behaviors annotated by an experimentalist are: arch, burrow, drag, groom, tap, egg, probosis. The category idle was added in order to characterize all the time steps for which no specific behavior is recorded.

A. Imbalanced Data

The main challenge of this data set is that some actions are more likely to occur than others. For example, the fly is in the idle state with a much higher probability than in the egg laying state.

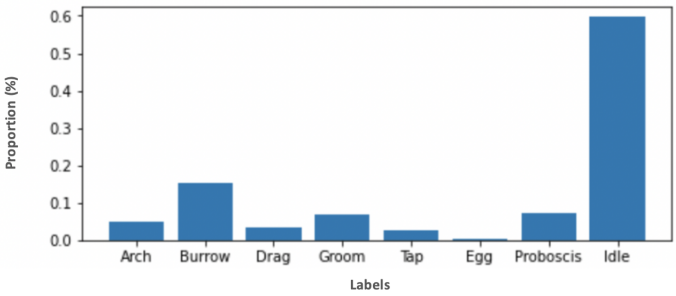

Fig. 1. Data repartition

These kinds of data sets usually generate biased scores. Indeed, the classifiers will learn how to predict the majority class and the accuracy won't reliably summarize the model performance. Indeed, a good accuracy can be still obtained even if the model predicts no occurrences at all for the event "egg", "proboscis and droom" but performs well on predicting the "idle" behavior, which occurs most of the time.

Several methods exist to balance the data set. Oversampling and undersampling won't be used in our model because these methods are computationally expensive and are more prone to over-fitting due to the duplication of data points. Besides, undersampling methods discard sometimes important data points. The first idea is to adjust the weight of each class, it is called a weighted cross-entropy loss. Instead of having equal weight for each class, the loss function is multiplied by a certain factor according to the class proportion, the majority class is multiplied by a small factor and the minority class by an higher one. Therefore, an wrong predictions on a misclassified rare example will be highly penalized.

Another approach would be to use focal loss [2] which has better results than unweighted cross entropy loss for imbalanced data. Focal Loss is just an improved version of Cross-Entropy Loss and uses the same idea. Through the modulating gamma coefficient, it handles the class imbalance problem by increasing the weight of minority classes that are easily misclassified.

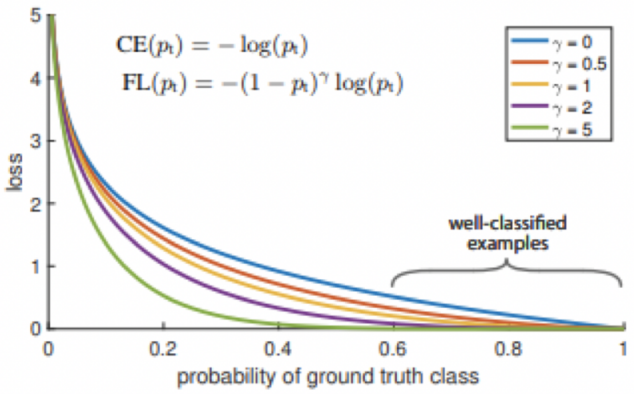

Fig. 2. Focal loss compared to Cross Entropy loss

B. Time-series sequence

Fly behavior labelling in videos is a challenging task, since we need to take into account information from previous and/or following time points. Although recurrent neural networks might appear well-suited for this task, they present two major drawbacks. First of all, their network architecture make them unable to remember long-range dependencies. Secondly, recurrent gradient multiplication due to error back propagation in the network may result in a vanishing or an exploding gradient.

C. LSTM

LSTMs or Long short term memory neural networks address those two issues. On one hand, the introduction of input, output and forget gates along with cell states enables keeping track of the most relevant elements in past history. On the other hand, additive components in the gates prevent the gradient from vanishing or exploding. Bidirectional LSTMs show even more predictive power, taking into account both previous and subsequent time steps to predict the current state. Both model types were explored. Since we needed one prediction per time frame, we design a many-to-many LSTM, where the output vector is of same shape as input vector.

D. TCN

TCN is a very interesting convolutional neural network for our problem. It has two main particularities that make it very powerful in term of time-series predictions. First, TCN has causal convolutions, meaning that each step depends only on previous steps and not on the future ones. This temporal hierarchy allows the upper layers to have access to longer input sub-sequences and learn representations at a larger time scale. Moreover, TCN can return a vector of the same size as the input. With RNN, we weren't able to return the label for each timestep but only one label for every timestep. TCN uses a 1D convolutional network architecture, where each hidden layer is of the same length as the input layer. It keeps subsequent layers of the same length as previous ones and uses dilated convolutions to avoid a linear growth of the receptive field. Dilation is a trick to enable the receptive field to grow exponentially and capture a large view of the input with few parameters and few layers. A large receptive field is central to apprehend long-range dependencies.

E. Random Forest

Random Forest is a collection of decision tree algorithms that is suitable for classification or regression problems. It is a model that is often used in time series for forecasting data. Moreover, it is a very popular model among data scientist since it is easy to interpret, stable and has generally a good performance. In the given task, random forest can capture the temporal characteristics of the data by selecting interval features rather than frame features.

III. Data preprocessing

A. Train and test sets

Since our training data set was quite small, we've investigated several simple options [6]. The first one was to flip the sign of our data to increase the number of available samples, but this didn't lead to major improvement for LSTMs models. The second one was to randomly inject small gaussian noise to already processed positions, and TCN model gave better results with this set-up (see Results section). Finally, since our models will be evaluated on a 20 minutes long video, we have created shorter videos from our training set, reducing the number of time steps available for training. We then trained LSTM and TCN models with it. Further results are discussed in Section V.

B. Preprocessing

We were interested in several aspects of data preprocessing. The first concern is data scaling. Neural networks perform better on scaled data, which improve the convergence speed. Normalization and standardization were investigated in different orders. Normalization performed on standardized data showed better results compared to merely standardized features.

The second concern is features selection. Neural networks such as CNN or LSTM learn by themselves. We have merged two labels since "arch" is an obligatory behavior for burrow, drag and tap. The idle label was added to categorize the absence of any defined behavior. For random forest, since a slight change in the training data can result in a relatively different tree structure, feature expansion was explored. Velocities along x and y axes between 2 consecutive frames were added, as well as the distance on the x axis between 2 points on the fly's body.

IV. Model training

As stated in Section II-A, we did not rely on the accuracy for parameter tuning. We optimized our models based on macro, weighted and proportional F1 scores upon cross validation. The F1 score is defined as the harmonic mean between precision : $\frac{TP}{TP+FP}$ and recall : $\frac{TP}{TP+FN}$, where TP, FP, FN stand for True Positives, False Positives and False Negatives predictions. The macro F1 score computes the unweighted mean of the F1 scores label wise, while the weighted F1 score

returns the mean weighted by the number of true instances for each label. Finally, the proportional F1 score up-weights the F1 scores for minority classes and down-weights the majority classes ones. However, one major drawback of F1 score is that it cannot distinguish between no positive event correctly predicted and the absence of a given event (true negative) correctly predicted.

A. LSTM

Several aspects were taken into consideration, looking also at F1 score for each label. Different architectures were investigated, varying the number of LSTM layers along with the number of nodes. Bidirectional LSTMs appeared quite relevant to our problem since they were applied to human activity recognition (HAR), which deals with predicting defined mutually exclusive behaviors from pose tracking sensor data [5], [4] . Dealing with our multilabel specificity, sigmoid activation function was better suited for our task than softmax since softmax enforces mutual exclusitivity of predictions.

One of our concerns was model generalization to longer videos and overfitting reduction. At first, we have tried splitting the training data into shorter windows to enforce shorter range predictions, to mimic time window length discrepancy between test and train. However, the optimal window length on cross validation turned out to be the full 720 time steps. Then, doubling the data set by adding gaussian noise to , slight L1 and L2 regularizations of the loss were added to prevent overfitting, along with dropout to allow generalization of the model. The batch size was also carefully choosen and a small batch size of 32 was favoured since it offers a regularization effect, at the cost of the running time. Moreover, it prevents the model from converging to sharp minimizers [1] which yields a better generalization.

To summarize, our results were averaged over 5 repeats to build a confidence interval of our measures. We started from a baseline LSTM model with one dense layer and one LSTM layer of 100 nodes, without any regularization nor dropout, on normal data setting (no augmentation).

B. TCN

To improve the quality of our training, we can play on a few factors such as the number of dilations or the values of dilations, loss function, activation function.

Dilations introduce holes for more coverage and allow a global view of the input with few parameters, if the dilation value is too high, we'll lose information. On the other hand,if we only have a small dilation factor, the effect of dilations will be useless, the receptive field will grow linearly and won't capture long-range dependencies. The optimal size found with cross validation is 8 and the optimal values are the following: (8,16,32,64,128,256,512,1024).

Several loss were used such as cross entropy, focal loss. Nonetheless, as we said before the loss function with the best results for our model is the focal loss. The most common activation function for the convolutional network is Relu. However, the LeakyRelu, an amelioration of the Relu activation function seems to perform better than Relu, particularly with the 'egg' behavior predictions. The value of the loss might too high, so the L1 and L2 regularizators were used in order to prevent overfitting.

C. Random forest

In order to select the hyperparameters, cross validation combined to grid search was performed. The weighted F1 score was used as reference to compare between models. The parameter chosen to measure the quality of the split at a node was the gini impurity, running faster and similarly performance-wise than the entropy. Another parameter that was studied is the time window used to extend the data, in order to select interval features. The challenge was to find a time window that had a better performance while not being too large hence bringing down the training speed. The length of the time window was determined experimentally, by running the model several times with different values. Otherwise, the most critical parameters for random forest are the number of estimators, which is the number of trees in the forest, and the maximum depth of a given tree.

The optimal models were progressively built up on a baseline model, selecting the best architecture based on macro and weighted F1 score.

V. Results

The table 3 presents the best cross validated hyperparameters for some relevant models. The metrics are given in mean over 5 runs for LSTM and TCN, and other 1 run for RF, in the following order: weighted, macro and proportional F1 scores in percentages. The best architecture tuned on cross validation and the F1 scores per label of each model for the hold out test set are presented in figure 3,

Model	Hyperparameter	F1-score
LSTM	Type: Biderectionnal one Layer	Weighted: 0.834
	Nodes: 600	Macro: 0.616
	Epochs=200	Proportionnal: 0.097
	Regularization= 1e-6	
	Dropout: 0.1	
TCN	Dilation: (8,16,32,64,128,256,512,1024)	Weighted: 0.734
	Filter_size=64; Kernel_size=2, Gamma=5	Macro: 0.441
	Epochs=200	Proportionnal: 0.096
	Regularization= 1e-6	
RANDOM FOREST	Max_depth=10	Weighted: 0.616
	N_estimators=10	Macro: 0.199
	Time window=120 time steps	Proportionnal: 0.0096

Fig. 3. Best hyperparameters for each model and evaluation of performance via variants of F1 score

VI. Discussion and model evaluation

As a side remark, we struggled to ensure perfect reproducibility of our scores for LSTM and TCN models. Our results can vary due to Google Colab GPU stochasticity, but the best models stay the same.

First of all, when looking at the F1 score per label on our test set, our models failed to capture really rare events. One instance is egg laying, which occurs in one time frame per videos, as we can notice in Figure 4. A clear link appears between the occurrence of a given behavior II-A and the corresponding F1 score. Second, we can notice that focal loss used in TCN and LSTM achieved a good overall minority labels prediction on test set, when compared to Random Forest. Indeed, there are no imbalanced specific losses for this later model, and oversampling should have been explored to cope with this issue. F1 scores on the overall test set of TCN and LSTM are quite comparable since those two models are really powerful. This can be seen through fast loss convergence (*See corresponding Jupyter notebooks in Github repository*) when training. Although, we can notice that TCN model only achieved predictions on really rare events.

For the proportional F1 score, the number of occurrences of each class in the test dataset was taken into account by weighting the F1 score of each label when doing the average. In doing so, the disparity of data is taken into account, which is not the case in 'macro' F1 score. Hence, it leads to a much lower proportional F1 score than macro F1 score.

Thirdly, one of the disadvantage of random forest is that it requires consequent computational power and resources, especially when the number of trees in the forest is increased or the data expanded. It became really hard to train this model on our computer due to RAM limits.

Finally, we tested our models on a 20 minutes long video pose tracking files sent by the hosting lab . Unfortunately, the results were quite disappointing. Our models failed to predict any of the rare behaviors. We have decided to focus on True Positives versus true label occurrences instead of F1 scores per label to compare our model predictions on this video 5. Further metrics can be explored in our Jupyter Notebooks. However, it highlights some interesting mistakes we could avoid and processing steps we should have more carefully followed. First, it might reveal overfitting, since our models were trained on too few data, with paying more attention to metrics improvement than to validation loss divergence. Even if dropout and regularizers have been used, data augmentation techniques should have been further explored. Then, too many aspects were taken into consideration at the same time. We dealt with several models and tried to cope with time series configuration and data imbalance. It would have been wiser to focus on one of those aspects.

VII. Conclusion

Novel models such as bidirectional LSTM or TCN performed well in the given classification task. Random forest seems less suited to the task, being very sensitive to the imbalanced dataset. The label-wise F1 scores of the predictions of the three model reflect the imbalance in the original dataset, which stays the most challenging aspect of this project. Through this project, we could experience the limits of different metrics such as F1 score and accuracy and struggled with understanding real world data, and realized the diversity of tools and combinations available to increase our model performance.

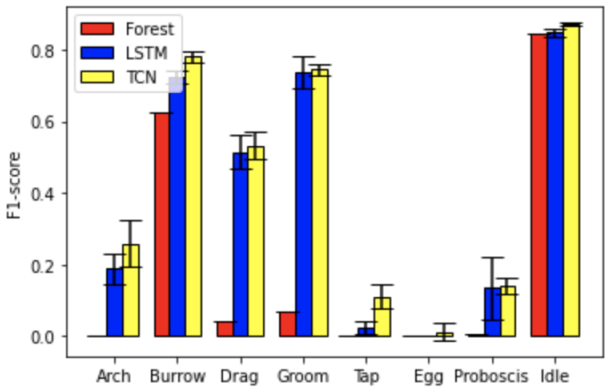

Fig. 4. F1 score per label for each model on test set

LABELS	OCCURRENCE	LSTM TP	TCN TP	RANDOM FOREST TP
Arch	950	0	**30**	0
Burrow	1307	0	**4**	0
Drag	117	0	**17**	0
Groom	250	0	**12**	0
Tap	512	**1**	0	0
Egg	151	0	0	0
Proboscis	1041	0	**62**	**1**
Idle	32139	**31877**	**14256**	**24475**

Fig. 5. Occurrence of behaviors on final video

VIII. Future Work

One improvement that could be investigated further is dealing with imbalanced data to improve Random Forest performance on rare behaviors, since this model is very sensitive to variations on training data. Models such as balanced random forest or weighted random forest [7] could be explored.

References

[1] Nitish Shirish Keskar et al. "On large-batch training for deep learning: Generalization gap and sharp minima". In: *arXiv preprint arXiv:1609.04836* (2016).

[2] Tsung-Yi Lin et al. "Focal loss for dense object detection". In: (2017), pp. 2980–2988.

[3] Cury K.M. et al Mathis A. Mamidanna P. "DeepLabCut: markerless pose estimation of user-defined body parts with deep learning". In: *Nature Neuroscience* 21 (2018), pp. 1281–1289. DOI: https://doi.org/10.1038/s41593-018-0209-y.

[4] Odongo Steven Eyobu and Dong Seog Han. "Feature representation and data augmentation for human activity classification based on wearable IMU sensor data using a deep LSTM neural network". In: *Sensors* 18.9 (2018), p. 2892.

[5] Amin Ullah et al. “Action Recognition in Video Sequences using Deep Bi-directional LSTM with CNN Features”. In: *IEEE Access* PP (Nov. 2017), pp. 1–1. DOI: 10.1109/ACCESS.2017.2778011.

[6] Qingsong Wen et al. “Time Series Data Augmentation for Deep Learning: A Survey”. In: *arXiv preprint arXiv:2002.12478* (2020).

[7] M. Zhu et al. “Class Weights Random Forest Algorithm for Processing Class Imbalanced Medical Data”. In: *IEEE Access* 6 (2018), pp. 4641–4652. DOI: 10.1109/ACCESS.2018.2789428.