ML For Science : Mouse Action Segmentation

Jérémi Do Dinh, Marouane Jaakik, Yassine Khalfi
École Polytechnique Fédérale de Lausanne

***Abstract*—The study of behaviours between agents is an essential topic in neuroscience to understand the interactions that can occur. One of the difficulties in studying these behaviours is the data annotation process by experts to label the behaviours present. This operation requires a lot of resources, and the aim here is to investigate whether models are capable of annotating data, thus allowing more studies to be carried out on a larger sample of data. In this paper we are interested in the behaviour of a mice, to see if we are able to characterise their behaviours. Our dataset consists of videos of mice annotated by experts, and our mission will be to create models able to detect patterns of behaviour identified by the experts.**

I. Introduction

For a long time, in animal behaviour studies, annotation was done by experts, who by looking at frame by frame videos of mice were responsible for classifying each frame with a behaviour. These manual operations can only be done by experts and are therefore require a lot of resources. The aim of this project is to create an automatic video classification tool, thus overcoming this problem which is a limitation for many studies in the field of animal behaviour.

To address this issue we will rely on the CalMS21 dataset, provided by The California Institute of Technology. With the help of this dataset we will try to answer 3 tasks. TASK1 is a simple behavioural classification task. It consists in creating a model capable of classifying the different behaviours of a mouse based on a dataset of already labelled videos. TASK2 is an extension of TASK1 which takes into account the style of annotation specific to each annotator, as each annotator can annotate the files differently. Finally, TASK3 aims to create classifiers for new behaviours where the number of annotated samples is very small.

II. Dataset

The data we use was collected by the California Institute of Technology (Caltech). This dataset was constructed from videos of mice interacting and follow a standard resident-intruder assay format, where a mouse that has been in its cage for some time (the resident) is presented with the arrival of a new mouse (the intruder). After the introduction of the intruder mouse, the animals are filmed interacting under the supervision of experts who are then responsible for annotating the data. In the general case (TASK1 & TASK2), the experts annotate the data according to 4 possible behaviours: "attack", "investigation", "mount" or "other". In the case of TASK3, other behaviours may be possible, which are not named but of which we know there are 7. Videos are classified based on whether a given behavior is present, and the labels in the video are binary to represent whether a given behavior is occurring at a given frame. The videos are filmed at 30 frames per second, and the data from each frame consists of 7 keypoint coordinates per mouse. These key points are shown in Figure 1.

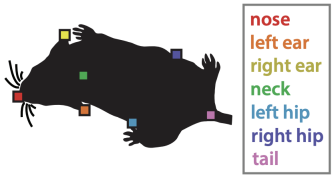

Figure 1: Illustration of the seven anatomically defined keypoints tracked on the body of each animal. [1]

There are 70, 30 and 17 videos/sequences for TASK1, TASK2 and TASK3 respectively, each of which can have a duration in the range of 2 to over 10 minutes. In the distribution of dataset labels, we observe that the majority of the annotations made by experts fall into the "other" category. More statistics are shown in Figure 2.

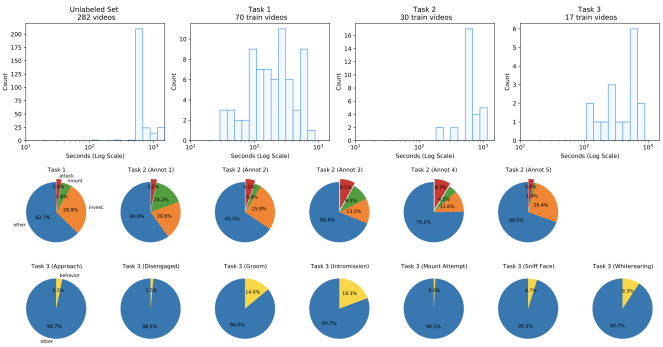

Figure 2: Statistics on the number of videos and the distribution of labels for each task [1]

III. Data pre-processing

Before we could use our data to train different models, feature engineering and pre-processing was deemed necessary to both highlight features that may be hidden by

our data at the moment, but also to enhance the data with features that, based on our observations and knowledge of the subject, may be of interest to the task at hand. In each of the following sub-sections, the changes or additions made to the data and the reason for these changes are detailed.

A. Change of origin of points

In the initial dataset, the points are referenced with respect to an origin that is situated at the upper left corner of the frame, which corresponds to one of the corners of the box where the mice are put. Therefore one of the data modifications was to change the origin of the points to the centre of the box where the mice are filmed. This will make the model invariant to the position of the mice. In addition, during our observation of the dataset, we noticed that some of the behaviours such as attacks occurred mainly at the corners of the box, so changing the origin allowed us to consider all these positions at the same distance from the centre.

B. Interpolation

The mouse coordinates are extracted from the videos at a sampling rate of 30 frames per second. Another feature enhancement we have made is the upsampling of the data to more than 30 frames per second. To go beyond 30 fps we use interpolation techniques. This allows to add any number of interpolated frames between 2 frames, and enables an increase in the size of our dataset and a reduction in the "jump" effect that can occur between 2 frames. To this extent, we use quadratic interpolation to determine the intermediate mouse coordinates and when we need to assign labels we assign the next label of an invalid frame to the interpolated frames. This treatment is very costly in resources and increases the size of our dataset considerably.

C. Distance between points

Originally in the dataset we have the coordinates of the key points of interest of each mouse as explained in Figure 1. One of the additions made is the addition of the pairwise distances between the different key points present in the dataset. Thus, in each frame for each mouse, the Euclidean distances between each of the key points are calculated, which gives us an idea of the shape of the mouse. In addition, the Euclidean distances between the points of interest of the two mice are also calculated, which gives us an idea of the proximity between the two mice. In the case of certain behaviours, the distance between two keypoints, such as the distance from nose to nose, can be characteristic. All these distances can thus give indications about the type of interactions that are taking place between the two mice.

D. Speed

Another important feature for behaviour detection is the speed of the mouse key points. On a general aspect of the problem, a fast speed of the mouse key points can mean a change in behaviour, based on our observations we can see that the mouse will suddenly speed up when attacking or that a still mouse can mean that the mouse is not producing any action. To calculate the speed of the mouse, we do this at each key point, we calculate the Euclidean distance for a key point between consecutive frames and multiply this value by the data acquisition frequency.

E. Rotation and stressing data

Due to our decision to implement a multitask architecture for all 3 tasks, it was important that the number of sequences is similar in all 3 categories. This was proven to be challenging, since task 2 & 3 provide 30 and 17 sequences respectively, which is much less than the 70 provided in task 1. To alleviate this constraint, copies of the original sequences were made, each going through a augmentation process. The augmentation process involved random rotation of the keypoints, as well as mirroring, shifting, and the addition of gaussian noise. It is was clear to us that doing this transformation to the data doesn't change the actions occurring in the sequences, however does significantly change the look of the data. This yielded additional 'artificial' sequences, and allowed us to augment the number of sequences to match the amount in task 1.

F. Bounding Box

The bounding box for each mouse is the rectangle within the frame defined my the maximum and minimum x and y coordinates of the keypoints for each mouse. Based on this measure, it is possible to obtain the area of intersection between the bounding boxes of both mice, which is the feature that was added. This feature engineering idea comes from the hypothesis that certain behaviours (for instance *mounting*) may depend on the amount of overlap between areas that the mice occupy.

G. Center of Mass and Mouse extension

The center of mass for each mouse is the point represented by the average of the x and y coordinates. This provides a concise description of the approximate position of the mice in the space. Using this feature, mouse extension was calculated for each mouse, which is the average euclidean distance of each key point to the center of mass. This allows us to estimate how compressed of decompressed both mice are, as certain actions may depend on how extended a given mouse is. For instance, it was observed that the keypoints tend shrink close to the center of mass prior to an attack. Hence the idea to include this feature in the engineered data.

H. Pipeline

The pipeline incorporates all the preprocessing necessary to transform the provided dataset to the format adequate for our model training. For TASK1, which consists of 70

sequences, no augmentation was necessary, and the data was enhanced using an interpolation factor of 3 (tripling the frame rate), and with all the other necessary features. For TASK2 and TASK2 sequences were artificially created using the augmentation techniques described above to match 70 sequences per task. The data was subsequently enhances the same way as for TASK1. For the testing data, the same techniques were used to enhance the data, however the interpolation was omitted, as it is not necessary to augment the data in the prediction phase. Finally columns of the dataset sere standardized to be in the $[0, 1]$ range.

IV. Models

A. Baseline

Many baseline models have been explored. The models consisted of very simple architectures such as LSTM, fully connected, 1D convolutional and a self attention networks.

B. Transformer

This architecture adapts an encoder-decoder flow. Given the extracted frame-wise video feature sequence, the encoder will first predict the initial action probability for each frame. Then the initial predictions will be passed to multiple successive decoders to perform an incremental refinement. Both the encoder and the decoder are composed of modules that can be stacked on top of each other multiple times. These modules consist of self attention layers followed by batch norm layers and finally a feed forward fully connected layer with a softmax head to generate the probability distribution over the action behaviours [2] [3]. Below you can find a diagram presenting the architecture of the transformer model used.

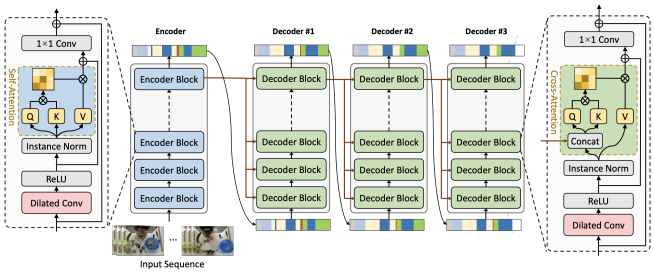

Figure 3: Architecture of Transformer model [3]

C. TCN

A TCN, short for Temporal Convolutional Network, consists of dilated, causal 1D convolutional layers, with the same input and output lengths which is ideal for generating action prediction per frame [4] [5]. We formalize our learning problem as seq2seq. One desirable quality of a seq2seq model is the ability to uncover long and short term temporal dependencies. In our case this is achieved through an adaptive receptive field which correspond to the dilation factor. The overall architecture consist of stacking several predictors sequentially each composed of several layers of dilated 1D convolutions. We call the first block predictor generators and the remaining blocks the refinement heads.

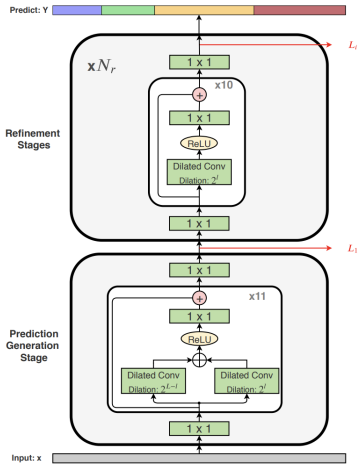

Figure 4: Architecture of MS TCN [4]

D. Multi task learning

Multi-task learning is a subfield of machine learning in which multiple tasks are simultaneously learned by a single model using shared representations to learn common ideas between a collection tasks which are often correlated. The problem we are tackling includes three tasks and is a perfect setting for MTL. The core idea is to have as many shared parameters between the 3 tasks as possible. To this extent, we use the predictor generator as a common encoder for the three tasks after which we use different refinement heads for each task. We finally have one refinement head per annotator and per new behaviour.

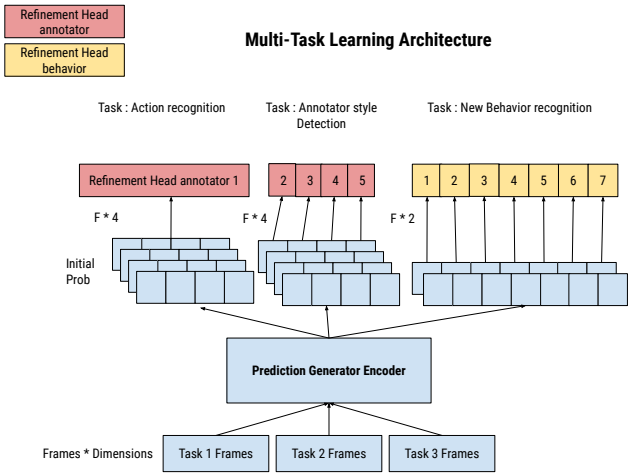

Figure 5: Architecture of Multi Task learning model

V. Results

From the different models that we have been able to build and adapt to our problem, we have been able to get different results for the separate tasks that we will present and analyse in this section. To compare our models we will use the class-averaged F1 score and the Mean Average Precision as our metrics.

A. First Results

The first results correspond to the results obtained with the models mentioned in the baseline section with the basic data, i.e. simply the mouse coordinates. From this we can obtain the following results for Task 1:

Table I: Baseline Results

Method	Average F1	MAP
Fully Connected	.659 ± .005	.726 ± .004
LSTM	.675 ± .011	.712 ± .013
Self-Attention	.610 ± .028	.644 ± .018
1D Conv Net	.793 ± .011	.856 ± .010

As we can see, we get very different results depending on the models. The model giving the best performances is the convolutional network, which is not surprising because we know that this kind of model works very well for action recognition tasks [6]. The TCN whose results we will present later takes the convolutional network architecture into account, but also considers the temporal dimension of the data, which will allow us to improve the performance.

B. Results of our models

Table II: Advanced models Results Task 1

Method	Average F1	MAP
TCN	.844 ± .01	.886 ± .01
Transformer	.85 ± .015	.89 ± .012
Top-1 entry	.864 ± .011	.914 ± .009

Table III: Advanced models Results Task 2

Method	Average F1	MAP
TCN	.8 ± .011	.848 ± .011
Transformer	.79 ± .009	.84 ± .012
Top-1 entry	.809 ± .015	.857 ± .007

Table IV: Advanced models Results Task 3

Method	Average F1	MAP
TCN	.23 ± .009	.29 ± .009
Transformer	.25 ± .018	.30 ± .016
Top-1 entry	363 ± .020	.352 ± .023

C. Multitask learning

Table V: Advanced models Results Task 3

Task	Average F1	MAP
1	.858 ± .01	.901 ± .01
2	.80 ± .01	.84 ± .09
3	355 ± .019	.34 ± .016

VI. Data Interpretation

For each single task we see that the transformer performs on average slightly better than the TCN, however it takes 10x times longer to train and requires 8x the memory on the GPUs. Inference time is also extremely slow (around 20x times slower) which makes the transformer a bad choice for real time applications. It is for those reasons that we decided to build on the TCN architecture for multi task learning.

In this case multi TCN outperformed all our models and got extremely close to the top 1 entry for this benchmark, training on multiple tasks reduces overfitting and helps smoothen the predictions especially for TASK3 where we have seen a huge improvement when using this architecture.

Fine tuning Multi Task Learning networks is extremely challenging , many design choices are available for computing the loss and methods of backpropagation along the the computational graph

We have tried summing the losses but what provided best results was a weighted sum of the losses incorporating a proportional weight to the loss generated by each task. We multiply the overall loss by 0.1 for stability.

VII. Conclusion

Action segmentation is still a challenge, and as a result we attempted to make the most use of all the data provided. Due to the variety in the annotation styles, and the little amount of data for some of the tasks, training part of the model jointly is a good solution to make the most of the similarity between the datasets. This makes sure we maximize the information can be shared between the tasks. We conclude, and the results reflect this, that a multitask architecture is the best method for training in this scenario.

References

[1] J. J. Sun, T. Karigo, D. Chakraborty, S. P. Mohanty, B. Wild, Q. Sun, C. Chen, D. J. Anderson, P. Perona, Y. Yue, and A. Kennedy, "The multi-agent behavior dataset: Mouse dyadic social interactions." https://arxiv.org/pdf/2104.02710.pdf, 2021.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin, "Attention is all you need." https://arxiv.org/pdf/1706.03762.pdf, 2017.

[3] F. Yi, H. Wen, and T. Jiang, "Asformer: Transformer for action segmentation." https://arxiv.org/pdf/2110.08568v1.pdf, 2021.

[4] S. Li, Y. A. Farha, Y. Liu, M.-M. Cheng, and J. Gall, "Ms-tcn++: Multi-stage temporal convolutional network for action segmentation." https://arxiv.org/pdf/2006.09220.pdf, 2020.

[5] L. Colin, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection." https://arxiv.org/pdf/1611.05267.pdf, 2016.

[6] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos." https://arxiv.org/pdf/1406.2199.pdf, 2014.