
THE UNIVERSITY OF EDINBURGH

Comparative Analysis Of The Performance of DeePLabCut

Hallelujah Kebede

4th Year Project Report
Artificial Intelligence and Software Engineering
School of Informatics
University of Edinburgh

2022

Abstract

In many areas of animal behaviour research, improvements in our ability to collect large and detailed data sets are outstripping our ability to analyse them(Valletta et al., 2017). One of the ways that the we collect such data is Videography. Videography provides easy methods for the observation and recording of animal behavior in diverse settings, yet extracting particular aspects of a behavior for further analysis can be highly time consuming. DeepLabCut was created in an effort to make the process more efficient, it is a toolbox originally developed by Alexander Mathis and Mackenzie Mathis. It was developed to meet the need for better efficiency, the requirement for better results with less training data and mitigate the results from the intrusiveness of computer-based tracking methods that require marking. It has been demonstrated, in the past, to be versatile as well as having excellent tracking performance comparable to human accuracy(Mathis et al., 2018). The work below is an analysis of the performance of DeepLabCut on a video dataset and assess it compared to the performance of Actual Home Cage Analyser(Actual Analytics Ltd, Edinburgh, UK) on the same data using the accuracy of human annotators as a benchmark.

Acknowledgements

I would like to thank my amazing supervisor J Douglas Armstrong and his colleague Rowland Sillito for all the support they have provided during this time.

Table of Contents

Chapter 1

Background

1.1 Ethology

Animal behaviour is the scientific study of the behaviour of animals. It can involve the study of their evolution and natural behavior, cognitive abilities and psychological constructs, or welfare and response to stressors, among other areas of natural animal behavior (Jayne and See, 2019b). Behavioral research on animals is also carried out to model human behavior, for example in psychological studies and pharmacological models, as well as for comparative purposes to understand differences and similarities between species(Hånell and Marklund, 2014).

The discipline covers study under experimental conditions, behaviourism, or natural conditions, ethology. It has a long history, dating back over 2000 years; however laboratory behavioral research became popular in the twentieth century with the rise of behaviorism, with research using animal models to understand more about the human processes of learning and memory and the comparative abilities of animals (Jayne and See, 2019a). These experiments are conducted by monitoring the movement, fine-scale motion, social interactions , vocalizations and physiological responses of individual animals.

Historically, a large variety of species has been used for behavioral testing but rodents have always been widely used, likely since they are mammals and easy to house and breed(Jayne and See, 2019b). One of the prominent ways experiments are conducted on rodents is through home-cage monitoring which is in general, about monitoring the behaviors of animals in their home cage. It is the cage in which they are normally housed and it allows the experimenter to observe their behaviors in a 24/7 setting. This method is deemed less intrusive than other methods which require animals to be brought out of their natural contexts which results in the behavioural data including that of the response to the act as well(Mary Anne Liebert, 2021). The home-cage monitoring system also has the added advantage that a rodents' active period is during the night and, therefore, subtle changes can be picked up without stress for the animals(ibid.).

1.2 Technology and Ethology

Recent technological advances in sensor and transmitter technology coupled with data mining and computational analysis mean that large data sets can be collected (Mathis et al., 2018). As such, a range of home-cage analysis systems already exists (for review see Richardson, 2015); all offering unique features. The Technology, for data collection, which is referred to hereafter is The HCA system (Actual Analytics Ltd, UK), which allows one to monitor a cage of mice, and has been designed to fit into two rack spaces of a standard IVC rack. One half of the rig comprises an RFID reader baseplate with antennae located on predetermined locations. This provides the base for placing the individually ventilated mouse home cage. The other half, fitted within the adjacent rack space, houses an infrared camera, a computer and the appropriate power supplies. The cage sits under a plate affixed to the top of the rig which is fitted with an infrared light source allowing for continuous video capture without compromising the quality of the image(Redfern et al., 2017).

The data collected by these systems however may fail to conform to assumptions of statistical models and unknown dependencies among different variables in the data may make it difficult to accurately represent mathematically, which results in the availability of a lot of data that can't be used. Statistical models start with an assumption about the underlying data distribution (e.g. Gaussian, Poisson). The focus is on inference; estimating the parameters of the statistical model that most likely gave rise to the observed data, and providing uncertainty bounds for these estimates(Valletta et al., 2017). This, as referred to above, limits their usage in this domain. The focus of ML is typically on prediction; without necessarily assuming a functional distribution for the data, a model that achieves optimal predictive performance is identified. Its approaches can address otherwise intractable tasks, such as classifying species, individuals, vocalizations or behaviours within complex data setsibid.

Below are some of the Machine Learning methods that are used in animal behaviour studies.

Unsupervised learning

Unsupervised learning methods uncover structure in unlabelled data. Structure means patterns in the data that are sufficiently different from pure unstructured noise. Structure can be discovered by visualizing the data after reducing its dimensionality (**dimensionality reduction)**, identifying groups of observations sharing similar attributes (**clustering**) and/or determining the distribution of the data (density estimation).

The rationale behind dimensionality reduction is simple; although the collected data may seem high dimensional, the structure of the data can be represented by fewer variables. Reducing the dimensionality of a problem is achieved by mapping the original data to a new feature set, feature extraction (Burges, 2010, Zhang et al., 2015 , Lee and Verleysen, 2007), and/or selecting a subset of attributes, feature selection (Liu and Motoda, 2007). Feature extraction deals with finding representations for high-dimensional data sets. The objective is to construct new features from the original measured variables that accentuate the inherent patterns in the data and are nonredundant.

Clustering

The goal of clustering is to find groups that share similar properties. The data in each group should be similar (minimize intracluster distance), but each cluster should be sufficiently different (maximize intercluster similarity). There are three major types of clustering methods:

- partitioning, where the feature space is divided into k regions;
- hierarchical, where small clusters are iteratively merged into larger ones, or vice versa; and
- model-based, where multivariate statistical distributions are fitted.

Valletta et al., 2017 Arguably the most widely used partitioning clustering method (e.g. to group behavioural modes in little penguins; Zhang et al., 2015). The feature space is divided into k regions as follows.

1. Choose k centroids (at random or using some prior knowledge).
2. Compute the distance between centroids and each data point.
3. Assign each data point to the closest centroid.
4. Compute new centroids; the average of all data points in that cluster.
5. Repeat steps 2 to 4 until data points remain in the same cluster or a maximum number of iterations is reached.

Supervised learning

Like traditional statistical models (e.g. generalized linear models), supervised learning methods identify the relationship between an outcome and a set of explanatory variables. An example of a supervised learning method is Descision Trees.

Decision Trees

Decision trees are simple and intuitive predictive models, making them a popular choice when decision rules are required (Hutchinson and Gigerenzer, 2005). They can handle both categorical and continuous data, and are fairly robust to outliers.

A decision tree is constructed as follows.

1. Find the yes/no rule that best splits the data with respect to one of the features.
2. The best split is the one that produces the most homogeneous groups.
3. Repeat steps 1 and 2 until all data are correctly classified or some stopping rule is reached.

Examples where these methods are employed include in vocalization studies. Vocalizations can be recorded remotely permitting assessments of population size and species composition, individual behavioural and inter/intraspecific interactions (Blumstein et al., 2011) . ML has been applied to classify and count particular elements or syllables (Acevedo et al., 2009). Early work used ML techniques to adjudicate similarity between calls based on sets of such elements (Tchernichovski et al., 2000). These

approaches can also discern differences in calls. Classification of calls from different species and subspecies is robust (Fagerlund, 2007, Kershenbaum et al., 2016) and permits assessment of community structure (e.g. Grigg et al., 1996; Brandes, 2008).

ML was also used in a comparative assessment of welfare across multiple laboratory populations of mice (Chesler et al., 2002). A potential novel use of ML would be to detect emotional state in animals based on facial expression, body posture or vocalization. Such techniques have already been used in humans looking at facial (Michel and El Kaliouby, 2003), physiological (Shi et al., 2010), vocal (Shami and Verhelst, 2007) and gestural (Castellano, Villalba, and Camurri, 2007) cues of emotions. ML also permits integration of multiple sets of these cues to further enhance emotion detection (Caridakis et al., 2007).

1.3 Deeplabcut

For the computational analysis of fine-grained behavior, pose(*the geometrical configuration of multiple body parts*) estimation is often a crucial step. The gold standard for pose estimation in the field of motor control is the combination of video recordings with easily recognizable reflective markers applied to locations of interest, which greatly simplifies subsequent analysis and allows tracking of body parts with high accuracy. However, such systems can be expensive and potentially distracting to animals, and markers need to be placed before recording, which predefines the features that can be tracked. This mitigates one of the benefits of video data: its low level of invasiveness(Mathis et al., 2018). Another alternative is training regressors based on various computationally derived features to track particular body parts in a supervised way. Training predictors based on features from deep neural networks also falls in this category.

One of the best algorithms is known as DeeperCut which in the past has been demonstrated to perform within human-level labeling accuracy with a small(200) number of training images(ibid.).This is possible as a result of transfer learning: the feature detectors are based on extremely deep neural networks, which were pretrained on ImageNet, a massive dataset for object recognition. Here we focus on a subset of DeeperCut: its feature detectors, which are variations of deep residual neural networks (ResNet) with readout layers that predict the location of a body part(ibid.). To distinguish the feature detectors from the full DeeperCut, we refer to this autonomous portion as DeepLabCut. DeepLabCut is a deep convolutional network combining two key ingredients from algorithms for object recognition and semantic segmentation: pretrained ResNets and deconvolutional layers. It is provided as an open source software package which is also named DeepLabCut. The toolbox uses the feature detectors from DeeperCut and provides routines to :

1. extract distinct frames from videos for labeling,
2. generate training data based on labels,
3. train networks to the desired feature sets, and
4. extract these feature locations from unlabeled data.

The typical use case would be for an experimenter to extract distinct frames from videos and label the body parts of interest to create tailored part detectors. Then, after only a few hours of labeling and training the network, can be applied to novel videosMathis et al., 2018.

1.4 Actual Analytics

A similar software for analysing video data will be discussed which is used by the data collection technology mentioned above(The HCA system). The software package called ActualHCA-Capture (Actual Analytics Ltd, UK) is used to capture readings from the baseplate antennae as well as synchronized video for subsequent validation work. The baseplate and video data are amalgamated using ActualHCA Analysis tool(Actual Analytics Ltd, UK)(Redfern et al., 2017).

In experiments conducted using the system the video data is used to validate the data collected from the baseplates using Actual HCA Capture™ (Actual Analytics Ltd, Edinburgh, UK) which reads and analyses the raw data generated by Actual HCA Analyser™. Video footage is analysed using motion detection for cage level events (e.g., video-based motion detection) or to extract individual behavior(ibid.). The data capturing phases of experiments done using the HCA system involves human annotators to check accuracy of the automated detection. While the Actual HCA Capture also has good perormance with the HCA system this paper analyses the accuracy it provides verusus the accuracy of DeepLabCut as an automatic detection mechanism on the same video dataset using human-level performance as a benchmark.

Chapter 2

Introduction

2.1 Deeplabcut

The software toolbox at the centre of this study is DeepLabcut. As mentioned in previous sections the software allows for pre-training activities like frame extraction from videos and creation of training data by enabling labeling of features in those frames. The tool also enables training of models pre-trained by Imagenet to have the last two layers of their neural net to be trained, forming a model for later use. It also provides tools to evaluate the network which serves as an informative checkpoint as to whether the model is good enough to move forward to either pose-estimation or tracking steps.

2.1.1 Installation

The installation instructions varied depending on the choice of preferences for work flow, operating system type and resources that were being used for the experiment. The experiment was conducted according to the below workflow:

- Use the DeepLabcut GUI to create project file
- Use the DeepLabcut GUI to extract frames from training videos
- Use the DeepLabcut GUI to label features
- Use COLAB notebooks to perform all training and post-training activities

This particular workflow was helpful in getting the project started quickly and familiarising with the toolbox as the GUI abstracted away some of the functions which one could use to preform the pre-training activities while offering an interactive peek into their documentation.

The resources chosen for this experiment included a CPU pre-training and GPU 's for all the activities afterwards. The GPUs were accessed through the COLAB notebooks which incorporate a cloud computing feature. Throughout the experiment there were attempts to use either CPU or GPUs from other sources , but these GPU's proved to be better interms of efficiency. The Operating system throughout the experiment was a MacOs.

The installation instructions also featured recommendations for the ideal software to use for various purposes and this was followed when the installation procedure was the performed, as follows:

2.1.1.1 Environment setup

- Install Anaconda to install a Python version of later than 3.7
- Install Deeplabcut with GUI support using :

```
pip install 'deeplabcut[gui]'
```

2.1.1.2 Run Environment

- In the terminal cd into location Deeplabcut was downloaded
- cd into Deeplabcut/conda-environments
- run:

```
conda env create -f DEEPLABCUT_M1.yaml
```

The environment can now be run from anywhere using:

```
conda activate DEEPLABCUT_M1
```

Running the environment proved to be one of the challenges eventhough the installation instructions were followed as outlined above. The environment would run in other contexts and a look at the discussionboards for Deeplabcut indicated that the 'base' environment has to be deactivated before activating the DEEPLABCUT environment to run.

2.1.2 GUI

The GUI is launched in the above activated environment using:

```
pythonw -m deeplabcut
```

.

Once launched the GUI has a User interface that has a form like structure to enter:

- Name of file
- Name of experimenter

After filling in these values that are used to name the folder and are directly entered into the configuration file the GUI has a button to load training videos into the folder and several tick boxes to mark whether the experiment is multi-animal and whether to copy videos into the folder. In this experiment the videos were all multi-animal.

The configuration file mentioned above is a yaml document to keep track of the setup of the experiment detailing the animals(rats and later mice in this case),features and other

experimental variables. The next step in the experiment was to use the Edit config file button to access the config file and setup the individuals/animals that are tracked in the videos under 'individuals:' which intrinsically sets the number of rats/mice. The other configuration is of the features/bodyparts on the the rats/mice that we want to be tracked i.e snout, tailbase ... which go under the 'multianimalbodyparts:' section.

There is also a 'uniquebodyparts:' section which is used when not all the animals have a certain feature and it needs to be included in the experiment. Features like these help in the later tracking phase as unique features help identify animals better.

The least straightforward section to setup in the earlier stages of the experiment was the 'skeleton:' section. This includes all the bodyparts listed in the 'multianimalbodyparts:' section, but it sets out the features in how they are connected to each other. An approach in this experiment is to decide on features and sketch out the connection graph using features as nodes and use the diagram as a guide to fill the section. The input is in the form

- - The feature/bodypart a connection starts from

  - The feature/bodypart the connection ends at

There is also the option to change the number of frames to be extracted from one video which is set to 20 by default in the 'numframes2pick:' section.

Once the config file has been setup. The frame extraction is started by pressing the extract frames button on the top of the GUI which opens up a window to setup parameters for extraction like the number of frames if not set in the configuration file or . This takes some time depending on the video size/length and outputs frames from the video to be used for labeling.

Labelling is by far the most time-consuming task in this experiment. The similarity in the animals in the video and the size of them for the mouse videos was a factor, but in general labelling for the purpose of pose-estimation and tracking required following through the video to do the tracking manually and locate the animal and it's features to label. The only optimisation being, to have the both open both the video and GUI in the same window. Labelling is also the most important as the accuracy of both the pose-estimation and tracking output are highly dependent on the quality of the labelling. In fact, a low quality in the labelling phase could mean animals aren't tracked at all.

This task is either done in-house or outsourced in larger projects. In this experiment, considering the amount of data ,just had each training set labelled by the experimenter.

To commence training the folder is uploaded to Google Drive where it can be accessed by a notebook.

2.2 Training

As mentioned above the training and following activities happen within COLAB notebooks which requires changes to the configuration file. The paths to videos and the

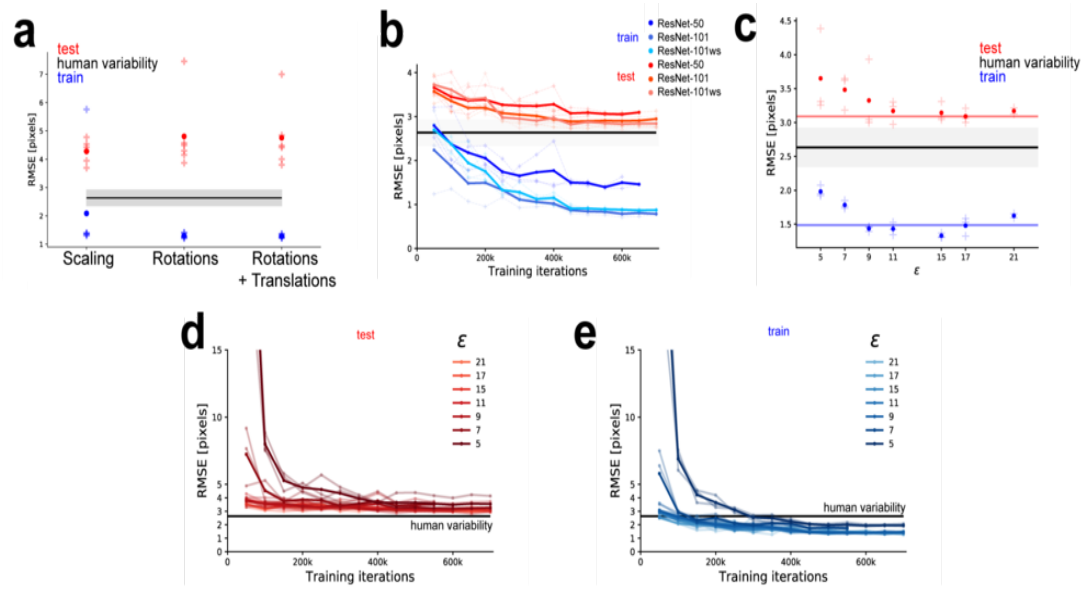


Figure 2.1: Resnet50 performance

configuration file set out in the document during creation have to be changed to the path the folder will be accessed from.

The path '/content/Drive' is added in place of the location the folder used to have locally. The Drive is then mounted with the instructions:

```
from google.colab import drive
drive.mount('/content/drive')
```

.

The training phase involves two functions:

createmultianimaltrainingdataset - which takes arguments filename and the name of the imagenet pre-training data type to use. The Neural net used in this work is the 'resnet50'. This was chosen as the result of the figures obtained by (Mathis et al., 2018) which suggest that it is the better in terms of speed and accuracy.

trainnetwork - which takes arguments filename and the number of iterations. The number of iterations is another crucial parameter. It influences the following steps as it has a correlation with the test error as shown below. The function returns a learning stats file that logs the improvements in various accuracy metrics per the number of iterations set in the 'saveiters' parameter of the function.

2.3 Evaluation

The Evaluation step has consists of two functions

evaluatenetwork - which returns the useful figures such as the training and test errors of the model.

extractsaveallmaps - returns visual representations of accuracy measures. These images are useful to identify whether the training has resulted in a good model or not. In the case that these images with labels do not reflect a good enough accuracy there is an intern=mediate function filterpredictions that finds outlier values and their corresponding frames. These frames can then be relabeled and used in later iterations of the training.

2.4 Pose Estimation

This step is when Deeplabcut starts to produce results, the first function is 'analyzevideos' which takes unseen/test videos as input and returns the frame by frame labeling by the model in multiple formats one of which is csv. That outputs in this format will be the main source of data in the analysis.

A complementary function to this is the 'createvideoswithalldetections' which outputs the video above with all the labels and is a very good tool to make the descision to move forward onto tracking as the quality of the labels(the consistency and accuracy) is what we turn into tracks.

2.5 Tracking

Tracking is the last step in this process.

the first function used is 'convertdetections2tracklets' which takes the detections from the analysis step above and retrieves the labels per frame . The function takes the value set in the configuration file for slider length to get the number of frames to extract per second.

The last function is 'stitchtracklets' this function takes ones particular input , the 'trackmethod' which is can either be pre-set in the configuratio file(reccmended as it might cause errors during this step) or in the function. This is set to ellipse by default but tDeeplabcut has several options including the general 'box' method. The experiments were run using the default setting as the Mathis et al., 2018 have demonstrated that it has better performance overall. The function returns tracking data by identity and labeled parts per frame.

Link to COLAB Notebook:

`https://colab.research.google.com/drive/1zgtNOW2gTeOXU-OGZwT8ydiHjB3Y3u-I?usp=sharing`

Chapter 3

Analysis

3.1 Parameters and Accuracy

All the functions mentioned in the previous chapter have that parameters serve multiple purposes from determining either the quality or even presence of outputs to helping evaluate the accuracy of the models produced. The first step in conducting this analysis is therefore, understanding how these parameters work with the data. The data is 2D rat and mouse video data. The The data presents challenges in that there isn't much of it as initially there were 2 mouse videos and later 8 rat videos. The form(2D) of the video also had an effect on the labelling process as 2D recordings mean missing data is inevitable for some frames as parts of the animal would be out of view in them.

Some solutions suggested to fix these issues include guessing where the missing label could be and opting for inaccurate outputs rather than potentially no outputs. The lack of many videos was later re-mediated, but Deeplabcut, as mentioned in Mathis et al., 2018 has been proven to have good performance with small data sets therefore the preliminary experiments are also included.

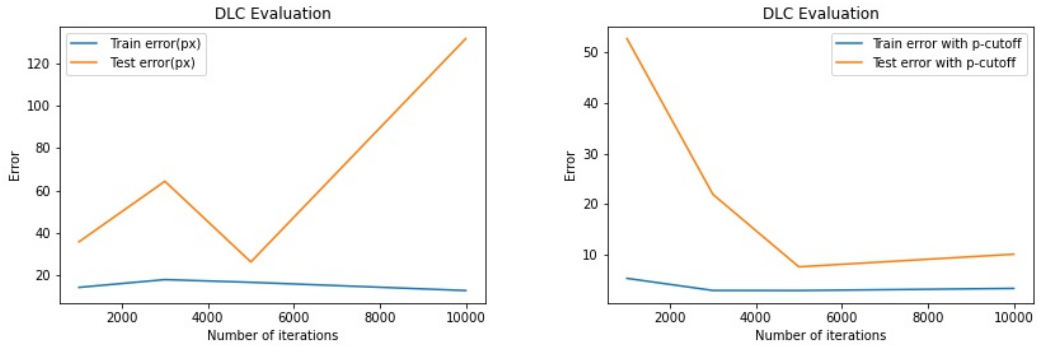
Given the data by ibid. the 'resnet50' is used the remaining parameter that is determinant to a DeepLabcut model's is the number of iterations. Therefore for the rest of the experiment models that result from different training iterations on the same data are considered as separate trackers.

To observe the effects of the number of iterations the Deeplabcut workflow described in the previous part was used, up until the evaluation which is where the relevant output is obtained.

The output file named dist_(number of iterations of training).csv contains the Root mean squared error of the tracker on labelling the original training frames. One of the preliminary experiments was done on data with 20 training frames where

```
Percentage of missing data : 9.65
Frames data is missing from frames: [4, 3, 6, 8, 12, 13, 19, 7, 22, 9,16]
```

The following investigation on the effect of training iteration was conducted.



(a) without p-cutoff (b) with p-cutoff

Figure 3.1: Iteration vs Error

3.2 Training and Test Errors

One of the outputs of the evaluation step is a Combined Evaluation file that records the training and test errors of the model. This was recorded for several models with many iterations.

The iterations were run on separately created projects initially and this was also one of the time- consuming bits of the project, possibly due to the design of the workflow.

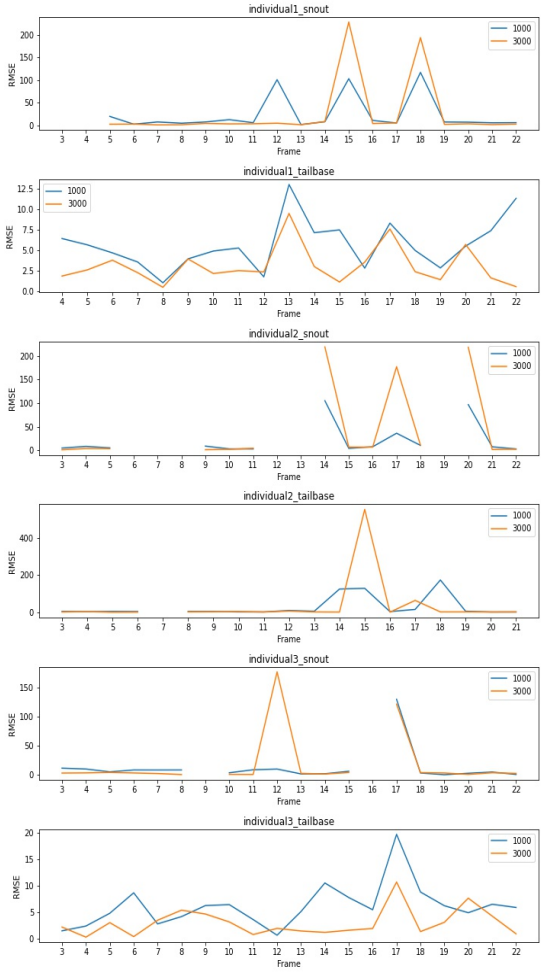
The first results showed an unexpected result of non-monotonic test error relation to the number of iterations(which are recorded in fig 3.1). This prompted an investigation into whether the data set size was the problem with the errors. The values in the evaluation file however were not only of training and test errors, but had values adjusted by another parameter in the configuration file, p-cutoff. This parameter was in the training function that was set by a default value of 0.6. It was used to filter predictions by looking at the confidence of the tracker. These values yielded a graph(the one on the right fig 3.1). This trend matched the assumptions made as well as the trend in Mathis et al., 2018 that the test error decreases with training iterations and the training error remains almost constant.

3.2.1 Data and Error

Deeplabcut also provides details on the models error rate by relabeling the training data frames. The output file has a frame by frame labelling and confidence measure that could better inform the decision to move forward with the model to pose estimation.

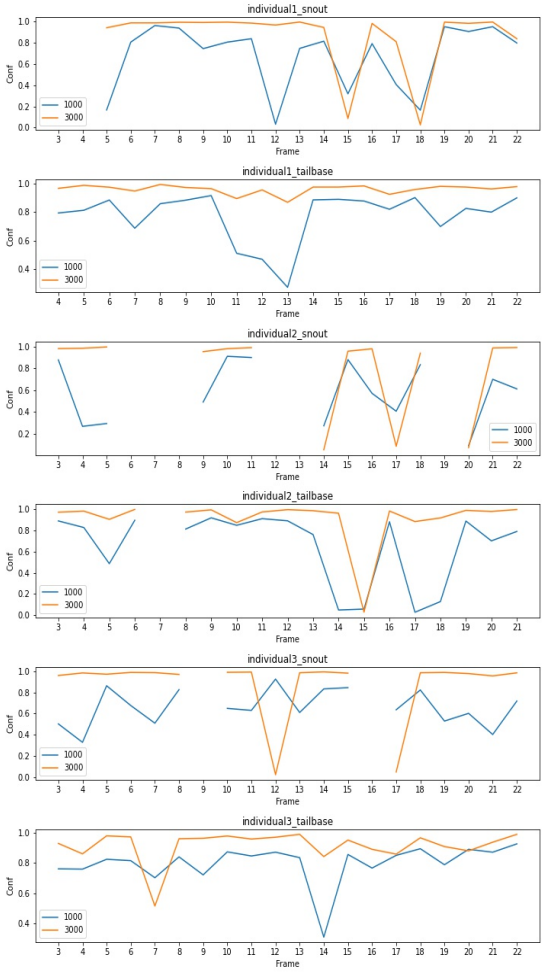
One of the basic problems in the experiments was that models would usually do well at this stage. They would then move forward and produce labelled videos which would visually appear to produce good results and later fail in the tracking phase by producing tracklets the last function cannot build tracks out of or result in tracks that have a majority of missing data.

The analysis below is done to investigate the effect of training iteration and missing data on the model accuracy and confidence.



This image represents the errors of two models on the same data, but varying (1000 and 3000) iterations per frame. The main observations from this image are:

1. Wherever there is data the model with 3000 iterations mostly does better than model with a 1000
2. Where the data is missing in the input the change in iteration does not change a model's ability to predict in that frame
3. The lack of data for one some labels in a frame seems to exacerbate errors in training with more iterations
4. For frames with consistently labeled data the performance gets better with the increasing iteration



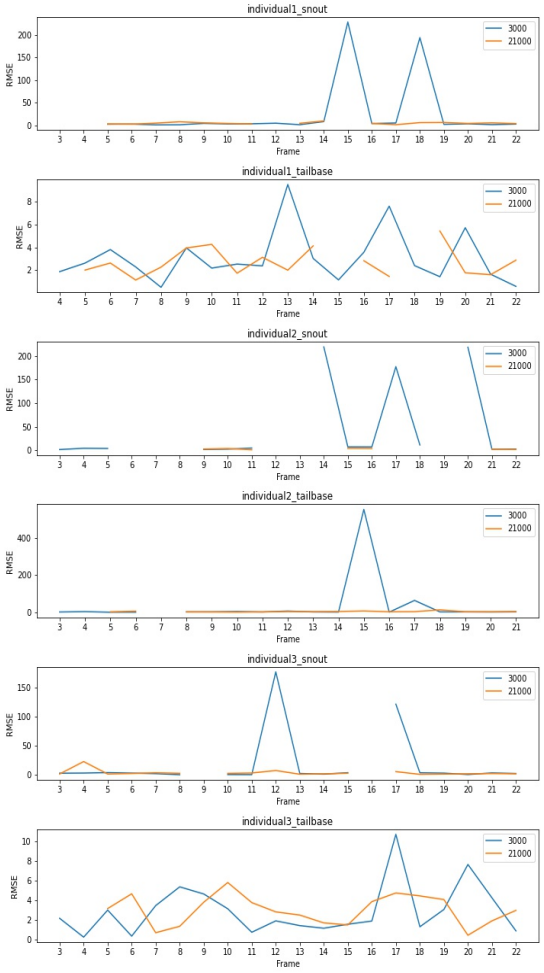
This image represents the confidence of two models on the same data, but varying (1000 and 3000) iterations per frame. The main observations from this image are:

1. Wherever there is data the model with 3000 training iterations mostly has better confidence than model with 1000 training iterations
2. Where the data is missing in the input the change in iteration does not change a model's confidence in the prediction, as it remains 0 for that label in that frame
3. The lack of data for one some labels in a frame seems to lower the confidence of the model in predictions in those frames in general
4. The higher iteration model is however getting higher RMSE with generally low confidence

Both models have a however have the property that whenever they make high RMSE predictions they make them with low confidence, hence the graphs are almost reflections of one another.

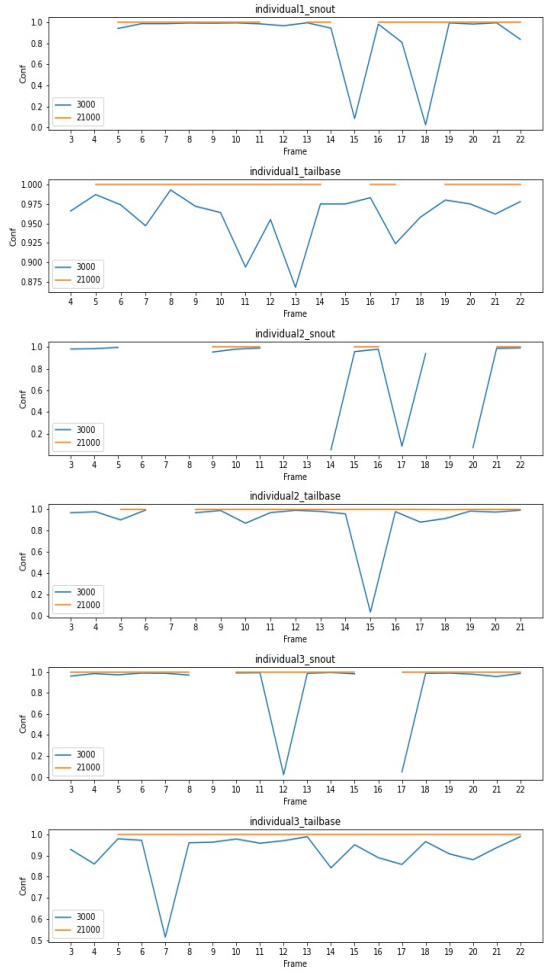
In looking for the solution for model's that couldn't perform tracking one of the resources used was the Deeplacut user support. One of their recommendations when faced with the tracking model inaccuracies was to increase the number of iterations.

The recommended number of iterations was a minimum of 20,000 therefore, the training was done again at 21,000 iterations to observe the changes.



This image represents the errors of two models on the same data, but varying (3000 and 21000) iterations, per frame. The main observations from this image are:

1. Wherever there is data the model with 21000 training iterations mostly does better than mode with 3000 training iterations
2. The performance of model with 21000 iterations does not seem to be affected by missing data as the other



This image represents the confidence of the of the two models on the right. The main observations from this image are:

1. Wherever there is data the model_21000 has higher(almost always perfect) confidence than model_3000
2. Where the data is missing in the input the change in iteration does not change a model's confidence to predict in that frame, it remains 0
3. The lack of data for one some labels in

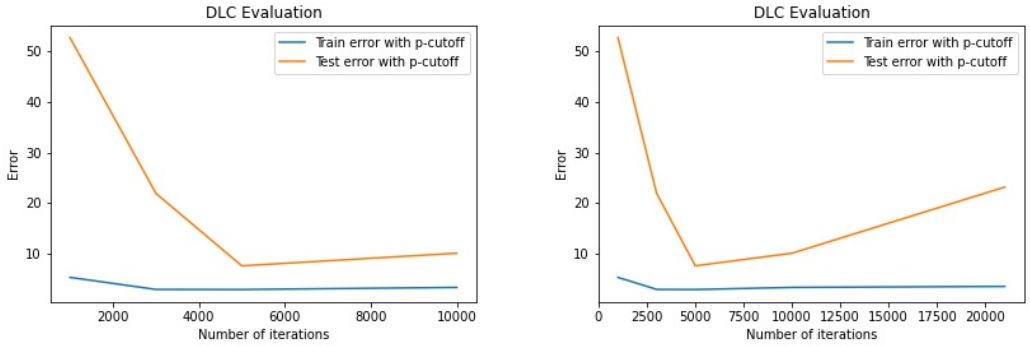
model was in the previous comparison
3. The increase in iterations does not enable the model to make any predictions for missing data
4. The higher iteration model does however still have higher RMSE for some frames where labels are missing

a frame has no effect on the confidence of the higher iteration model
4. The model with 21000 training iterations makes all the high and low RMSE predictions with the same confidence(1)

The model with 21000 training iterations had a confidence level which was really high and yet has some error values higher than that of the 3000 iteration model. It was also not producing tracklets which meant it was not improving.

Using the insight from the training error the next step was to look into how the model was performing on test data, which is the most relevant measure for performance comparison.

The approach was to train models on several iterations and observe how it fits with the trend in test error in fig 3.1 and the result is recorded in fig 3.2. This shows an increase of test error for the model which is a clear indication of over-fitting - A characteristics of models that perform well on training data and poorly on any data not seen beforehand.



(a) Error trend before

(b) Error trend after

Figure 3.2: Iteration vs Error

None of the above models succeeded in building tracks, however and more research led to another solution which was to increase the number of tracked bodyparts on the mice. Another set of experiments was run by increasing the number of body parts one of which is shown below.

The data used had properties:

```
Percentage of missing data : 52
Frames data is missing from frames: {3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17, 18, 19, 20}
```

The frames recorded are those with at least one label missing.

The data wasn't the best as gathered from the previous analysis and better percentage of missing data(at least 50) would be ideal, but an effect of labeling more body parts in 2D data is the possibility that some would be out of sight increases as well. Although in later experiments more guesses were made in the process of labeling this experiment was conducted before that solution was trialled.

This model was able to generate tracklets from test videos despite having a lot of missing data in the output as well. The tracks had:

```
Percentage of missing data : 82.86
```

Proving that the number of bodyparts tracked improved the ability of a model to track despite the significant loss in label data and track data.

The lessons from the above experiments are as stated below:

- Missing data in labeling phase will crop up at all other stages therefore guessing the location of labels(within reason) would be a better option than leaving unannotated.
- Labelling more(3 or more) bodyparts will improve tracking performance
- The performance of trackers could be improved on the same data by increasing iterations although this is not a strictly linear relationship.

3.3 Experiment

Unlike the preliminary experiments this experiment generates a tracker(a model that have that are guaranteed to produce tracks) by applying all the lessons. This experiment also differs in that the data used is opposite to the one preliminary experiment lower in quality and larger in size as opposed to the two higher quality videos.

The training data consisted of four 30 min mouse videos that were in '.flv' format which turned out to be incompatible with Deeplabcut and had to be converted into into a suitable format '.mp4' in this case.

The labeled data had:

```
Dataset 1:
Percentage of missing data : 47
Frames data is missing from frames: {3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22}

Dataset 2:
Percentage of missing data : 46
Frames data is missing from frames: {3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22}

Dataset 3:
Percentage of missing data : 44
Frames data is missing from frames: {3, 4, 5, 6, 7, 8, 9, 10, 11,
```

```
12, 13, 14, 15, 16, 17, 18, 20, 21, 22}

Datset 4:
Percentage of missing data : 40
Frames data is missing from frames: {3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
13, 14, 15, 16, 17, 18, 19, 20, 21, 22}
```

The first trail was 10000 training iterations which still had high missing data percentage in the output:

```
  Percentage of missing data : 81.28
```

It's training and test performance are:

```
Training Error(px) :3.22
Test Error(px) : 12.6
with p-cutoff(0.6)
```

The evaluation measures used so far measure how good a tracker is compared to its data i.e labeler. As stated in (Bernardin and Stiefelhagen, 2008),we need an objective measure of how this tracker would do compared to other trackers. One of the intuitive metrics suggested by (ibid.) to remedy that are :

(1) The multiple object tracking precision (MOPT):

Which is total error in estimated position for matched object-hypothesis pairs over all frames, averaged by the total number of matches made. It shows the ability of the tracker to estimate precise object positions.

$MOTP = \sum_{i\,t} d_t^i / \sum_t c_t$

MOTP is similar to RMSE used in DeepLabcut, because the naming used is not strictly the mathematical definition. RMSE in Deeplabcut is actually the MSE averaged by the number of matches which is similar to MOTP, the difference being that MOTP is summer up before averaging. To evaluate how good the tracker is against the labeler using the labels as ground truth the RMSE value is sufficient, but to evaluate it against another tracking system (Actual Analytics) the outputs of the tracking systems need to be matched.

Actual Analytics is a centroid tracker and has a single output for the location value of an animal per unit time while DEEPLABCUT tracks multiple body-parts and has multiple outputs. To get outputs that are similar, but also express the tracking ability of the systems DEEPLABCUT was extended as is described in the next chapter. The main aim is to aggregate deeplabcut data to obtain distance values for animal trajectories which Actual analytics also achieves, to make a comparison.

Chapter 4

DLC Extension

4.1 Introduction

The application was constructed out of scripts that were used during the analysis of the distance data from Deeplabcut. It was intended to allow Deeplabcut users to further process their output after tracking. It extracts missing data amount and distance data from the output. It can also find frames in which a specific animal behavior is being displayed.

4.2 Functional Requirements

4.2.1 Stakeholders

The main stakeholders will be DLC users who would like to get the distance data from the generated tracks and identify single body-part distance characterised behaviours.

4.2.2 System states

On users:

The app will request user input the name of file to be aggregated. The app will take the output tracking file name as input from user. The app will return distance for tracked body-parts per individual/animal. The app will also allow the user to identify frames for specified behaviours.

4.2.3 Use cases

A user requests distance data:

- Primary actor: User
- Supporting actor(s): App
- Precondition: App prompts the user to enter file name.

- Summary of interaction(s): The App asks the user to provide the Deeplabcut output filename. The user provides the name of the file and presses enter key to run the application. The application gives the distance, missing frames(and percentage of missing frames) as the output.
- Trigger: User presses 'Enter' key
- Guarantees: Success guarantees The distance_log,Track_log and output_log files have been written to. Minimal guarantees Same as above in Success guarantees. Failure guarantees The system either alerts the user of errors with filename,format... or exceptions.
- Main Success Scenario : User enters filename The application writes to the appropriate files The application checks or notifies the user that the writes have been successful.

A user requests behaviour data:

- Primary actor: User
- Supporting actor(s): App
- Precondition: App prompts the user to enter name of behaviour, bodypart involved and distance parameter
- Summary of interaction(s): The App asks the user to to enter name of behaviour, body part involved and distance parameter. The user provides the values and presses enter key to run the application. The application writes the frames where the behaviour is happening to the BehaviourFrames_log file
- Trigger: User answers yes to prompt for behaviour data request
- Guarantees: Success guarantees The BehaviourFrames_log file has been written to with the appropriate data. Minimal guarantees Same as above in Success guarantees. Failure guarantees The system either alerts the user of errors with filename,format... or exceptions.
- Main Success Scenario : User enters values The application writes to the appropriate file The application checks or notifies the user that the write has been successful.

The above cases are illustrated in fig 4.1.

4.2.4 Non-functional requirements

The application:

Shall make all prompts for data entry clear and understandable Shall be reliable or shall not be prone to unexpected crash or fail behaviour

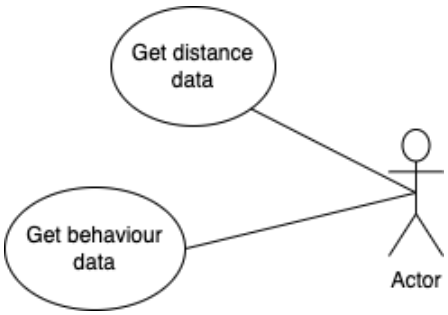


Figure 4.1: Use case Diagram

4.3 Design

Based on the requirements set above the application design with regards to the class association and dependencies is displayed below.

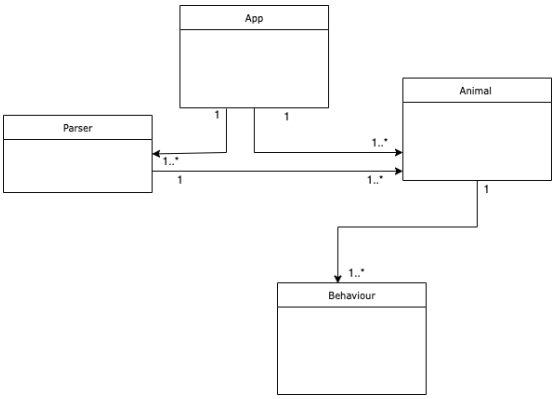


Figure 4.2: Class Diagram(Association and Dependencies)

The App class uses the Parser class to extract the data from the file provided by the user. The Parser uses this data to create the individual Animal objects within the file along with the data contained in the file about them. The App then uses this to calculate the distance for the individuals per body-part tracked. The App will also uses the data to find frames within which distance based behaviours are happening on request.

The functions and features of these classes are outlined in the Class diagram in fig 4.3.

4.3.1 High Level Description

Assumptions made and impact on design:

- Input is generated in '.csv' as the application only accepts '.csv ' file. The output from DeepLabCut is usually in '.h5' and can be converted using a DeepLabcut function, so the extra step is assumed to have been done.
- The output file is generated using the default DeepLabcut 'numframes' value which is 25 frames per second. The value of TIME_BINS is set with that assumption in mind as it is used to aggregate data within specific time frames

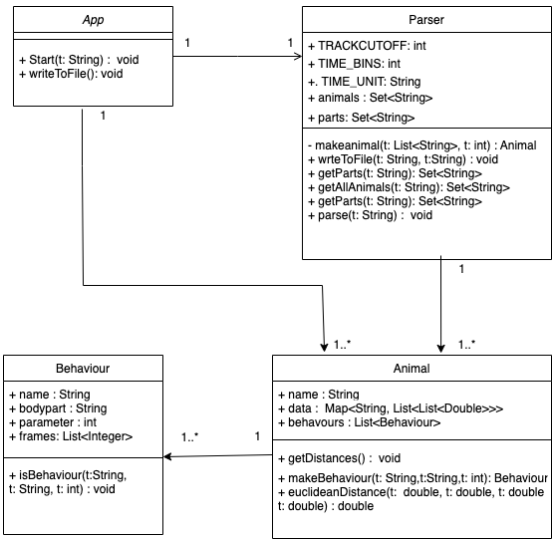


Figure 4.3: Class Diagram

- The user has domain knowledge in animal behaviours that are identified by distances between body parts. This assumption arises due to the lack of domain knowledge to the extent of choosing parameters and also because the knowledge gathered did not present these parameters in pixel units which the app is calculating the distance in.

The app was used to generate distance data initially, this was then intended to be compared to the distance data from Actual Analytics. The values turned out to be too far from each-other and the reason was found to be that Actual Analytics tags record distances travelled by tracking the animal in it's environment. This meant that Actual Analytics distances were in physical units like 'mm' and Deeplabcut obtained data from videos, or images to be exact, and the distances the extension calculated are in pixels(have no unit). Unless the scaling of these images was known the two values would not match.

The experiment had to be redesigned to obtain similar data from both systems. The solution was to have the Actual Analytics system as a labeler and use the previously found metrics to evaluate the performance of Deeplabcut against it. This was done by:

- Obtaining video with overlayed tag data
- Using the tags to label the identity of the individuals
- Using RMSE values as done in the previous experiment and compare that to the RMSE obtained against human-labeler

The distance extension was also applied later by using the outputs to check whether the tag and Deeplabcut returned the same measure for distance travelled.

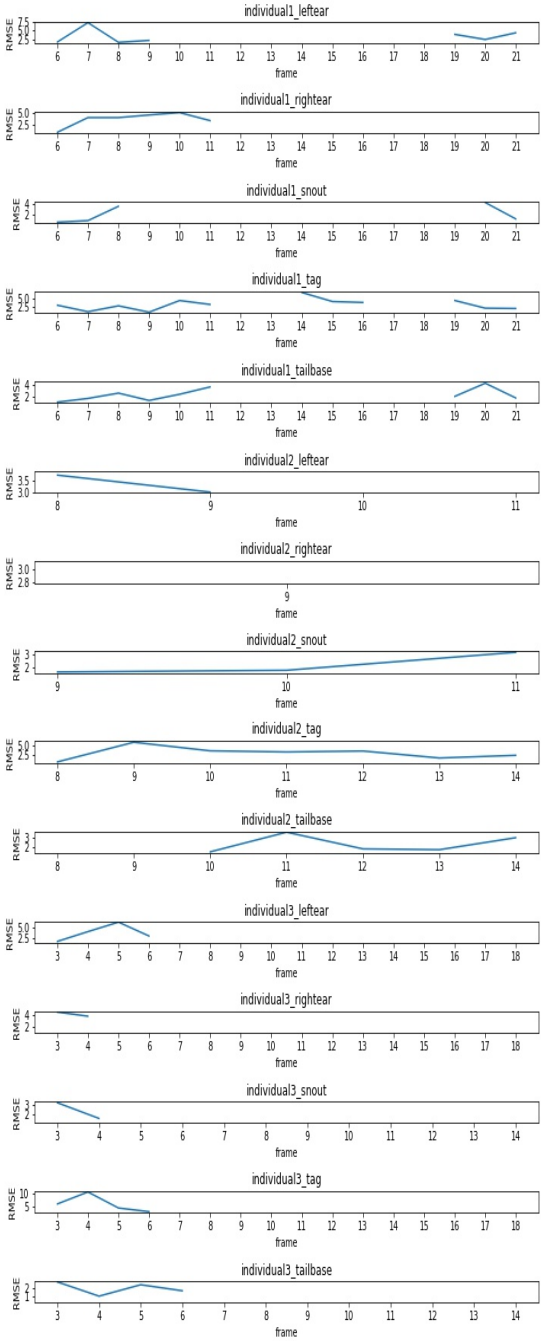
This was done by:

- Obtaining video with overlayed tag data
- Labelling the tag as part of the animal skeleton

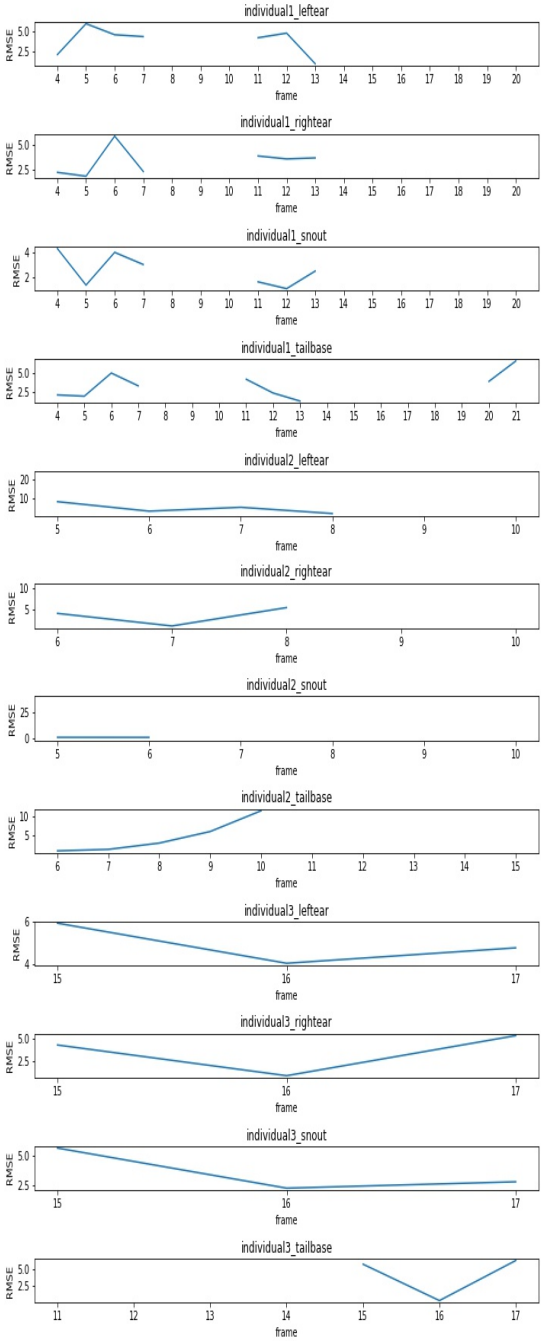
- Obtaining the distance measure using extension
- Visualising comparison

The first comparison values are as below:

Note that the tag has also been labelled in the second tracker data to achieve the second objective and comparison is made using the common values.



This image represents the errors of a model with 10000 training iterations on mouse data mentioned in the last experiment, but with the approach of the redesigned exper-



This image represents the errors of a model with 10000 training iterations on mouse data mentioned in the last experiment. The observations from this image

iment where the tag is also labeled. The observations from this image are.

1. The RMSE's span between 1 and 7$\dot{5}$ exclusive of the tags
2. It has a bodypart completely unlabeled (right_ear of individual_2)
3. The area under the graphs(which we stated was similar to MOTP in chapter 3) is less that on the left

are.

1. The RMSE's range from 30 to 0
2. The model does label wherever there is data, however
3. The area under the graphs exceeds those on the right in most cases

Assuming constant labeler inaccuracy. The experiment results show an improvement in the RMSE with labeling guided by RFID tags. The investigation into the distance travelled by tag vs distance travelled by bodypart was conducted by:

- comparing the distance travelled by all body parts to tag
- Choosing closest match
- Calculating the MSE with that body part

The bodypart found to be closest was the tailbase and has MSE:

- individual_2 25611.623913753985
- individual_1 88556.12622445569
- individual_3 74590.7245296464

Chapter 5

Conclusion

The experiment set out to find the performance of Deeplabcut and Actual Analytics compared to human labeler performance. The former was achieved to a great extent in early experiments and the latter was not found to be directly measure able as the data from Actual Analytics did not incorporate human labeling as it was designed to replace that need to begin with. The experiment was redesigned to look at at what the performance of Deeplabcut would be compared to a human labeler, a human labeler aided by Actual Analytics and lastly an indirect comparison between the distances moved by the tag and the rest of the animal body to investigate the latter of the original aims(compare Deeplabcut performance against Actual Analytics).

The last of these aims was not satisfactorily explored as either the data wasn't matching in units or the tag indicator was not a good measure of distance travelled by the animal as indicated by the last result.

Bibliography

Acevedo, Miguel A et al. (2009). "Automated classification of bird and amphibian calls using machine learning: A comparison of methods". In: *Ecological Informatics* 4.4, pp. 206–214.

Bernardin, Keni and Rainer Stiefelhagen (2008). "Evaluating multiple object tracking performance: the clear mot metrics". In: *EURASIP Journal on Image and Video Processing* 2008, pp. 1–10.

Blumstein, Daniel T et al. (2011). "Acoustic monitoring in terrestrial environments using microphone arrays: applications, technological considerations and prospectus". In: *Journal of Applied Ecology* 48.3, pp. 758–767.

Brandes, T Scott (2008). "Automated sound recording and analysis techniques for bird surveys and conservation". In: *Bird Conservation International* 18.S1, S163–S173.

Burges, Christopher JC (2010). *Dimension reduction: A guided tour*. Now Publishers Inc.

Caridakis, George et al. (2007). "Multimodal emotion recognition from expressive faces, body gestures and speech". In: *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, pp. 375–388.

Castellano, Ginevra, Santiago D Villalba, and Antonio Camurri (2007). "Recognising human emotions from body movement and gesture dynamics". In: *International Conference on Affective Computing and Intelligent Interaction*. Springer, pp. 71–82.

Chesler, Elissa J et al. (2002). "Identification and ranking of genetic and laboratory environment factors influencing a behavioral trait, thermal nociception, via computational analysis of a large data archive". In: *Neuroscience & Biobehavioral Reviews* 26.8, pp. 907–923.

Fagerlund, Seppo (2007). "Bird species recognition using support vector machines". In: *EURASIP Journal on Advances in Signal Processing* 2007, pp. 1–8.

Grigg, Gordon et al. (1996). "Monitoring frog communities: an application of machine learning". In: *Proceedings of eighth innovative applications of artificial intelligence conference, Portland Oregon*, pp. 1564–1569.

Hånell, Anders and Niklas Marklund (2014). "Structured evaluation of rodent behavioral tests used in drug discovery research". In: *Frontiers in behavioral neuroscience* 8, p. 252.

Hutchinson, John MC and Gerd Gigerenzer (2005). "Simple heuristics and rules of thumb: Where psychologists and behavioural biologists might meet". In: *Behavioural processes* 69.2, pp. 97–124.

Jayne, Kimberley and Adam See (2019a). "Behavioral Research on Captive Animals: Scientific and Ethical Concerns". In: *Animal Experimentation: Working Towards a Paradigm Change*. Brill, pp. 517–547.

– (2019b). "Chapter 21 Behavioral Research on Captive Animals: Scientific and Ethical Concerns". In: *Animal Experimentation: Working Towards a Paradigm Change*. Leiden, The Netherlands: Brill, pp. 517–547. ISBN: 9789004391192. DOI: https://doi.org/10.1163/9789004391192_022. URL: https://brill.com/view/book/edcoll/9789004391192/BP000026.xml.

Kershenbaum, Arik et al. (2016). "Disentangling canid howls across multiple species and subspecies: structure in a complex communication channel". In: *Behavioural processes* 124, pp. 149–157.

Lee, John A and Michel Verleysen (2007). *Nonlinear dimensionality reduction*. Springer Science & Business Media.

Liu, Huan and Hiroshi Motoda (2007). *Computational methods of feature selection*. CRC Press.

Mary Anne Liebert, Inc. (2021). *Home-Cage Monitoring and Its Effects on Research Capability and Outcomes*. URL: https://docplayer.net/154787396-Home-cage-monitoring-and-its-effects-on-research-capability-and-outcomes.html.

Mathis, Alexander et al. (2018). "DeepLabCut: markerless pose estimation of user-defined body parts with deep learning". In: *Nature neuroscience* 21.9, pp. 1281–1289.

Michel, Philipp and Rana El Kaliouby (2003). "Real time facial expression recognition in video using support vector machines". In: *Proceedings of the 5th international conference on Multimodal interfaces*, pp. 258–264.

Redfern, William S et al. (2017). "Automated recording of home cage activity and temperature of individual rats housed in social groups: The Rodent Big Brother project". In: *PloS one* 12.9, e0181068.

Shami, Mohammad and Werner Verhelst (2007). "An evaluation of the robustness of existing supervised machine learning approaches to the classification of emotions in speech". In: *Speech communication* 49.3, pp. 201–212.

Shi, Yuan et al. (2010). "Personalized stress detection from physiological measurements". In: *International symposium on quality of life technology*, pp. 28–29.

Tchernichovski, Ofer et al. (2000). "A procedure for an automated measurement of song similarity". In: *Animal behaviour* 59.6, pp. 1167–1176.

Valletta, John Joseph et al. (2017). "Applications of machine learning in animal behaviour studies". In: *Animal Behaviour* 124, pp. 203–220.

Zhang, Jingjing et al. (2015). "Extending the functionality of behavioural change-point analysis with k-means clustering: a case study with the little penguin (eudyptula minor)". In: *PloS one* 10.4, e0122811.