

Homework 3

Mouse location prediction using annotations

Dec 2nd, 2021

CS584 – Machine Learning

Submitted by: - Devansh Goel (A20490554)

Description

This homework aims to train a model to localize the mice in the video. Mouse locations are first annotated using DeepLabCut tool. DeepLabCut is an open-source toolkit for marker less pose estimation of animals. It is based on Convolutional Pose Machines and uses deep learning to estimate the 3D positions of body landmarks from 2D images. The toolkit can be used to automatically annotate images for a variety of tasks, including tracking the movement of animals, monitoring behaviour, and studying biomechanics.

DeepLabCut tool is used to make 7 joints: “nose”, “left ear”, “right ear”, “left hip”, “right hip”, “tail base”, “tail end”. It is also used to mark 2 corners, “left top” and “right bottom” of the bounding box. The mouse is inside the bounding box constructed from 2 corner point.

Finally, a neural network is used to predict the mouse location.

Data Analysis

The inputs of the model are 7 joint coordinates.

mouse1 topleft x, mouse1 topleft y: x and y coordinates of top left corner of mouse 1

mouse1 rightright x, mouse1 rightright y: x and y coordinates of right bottom corner mouse 1

mouse1 nose x, mouse1 nose y: x and y coordinates of nose mouse 1

mouse1 leftear x, mouse1 leftear y: x and y coordinates of left ear mouse 1

mouse1 rightear x, mouse1 rightear y: x and y coordinates of right ear mouse 1

mouse1 leftHip x, mouse1 leftHip y: x and y coordinates of left hip mouse 1

mouse1 rightHip x, mouse1 rightHip y: x and y coordinates of right hip mouse 1

mouse1 tailBase x, mouse1 tailBase y: x and y coordinates of tail base mouse 1

mouse1 tailEnd x, mouse1 tailEnd y: x and y coordinates of tail end mouse 1

mouse2 topleft x, mouse2 topleft y: x and y coordinates of top left corner of mouse 2

mouse2 rightright x, mouse2 rightright y: x and y coordinates of right bottom corner mouse 2

mouse2 nose x, mouse2 nose y: x and y coordinates of nose mouse 2

mouse2 leftear x, mouse2 leftear y: x and y coordinates of left ear mouse 2

mouse2 rightear x, mouse2 rightear y: x and y coordinates of right ear mouse 2

mouse2 leftHip x, mouse2 leftHip y: x and y coordinates of left hip mouse 2

mouse2 rightHip x, mouse2 rightHip y: x and y coordinates of right hip mouse 2

mouse2 tailBase x, mouse2 tailBase y: x and y coordinates of tail base mouse 2

mouse2 tailEnd x, mouse2 tailEnd y: x and y coordinates of tail end mouse 2

mouse1 center x, mouse1 center y: x and y coordinates of center mouse 1

mouse2 center x, mouse2 center y: x and y coordinates of center mouse 2

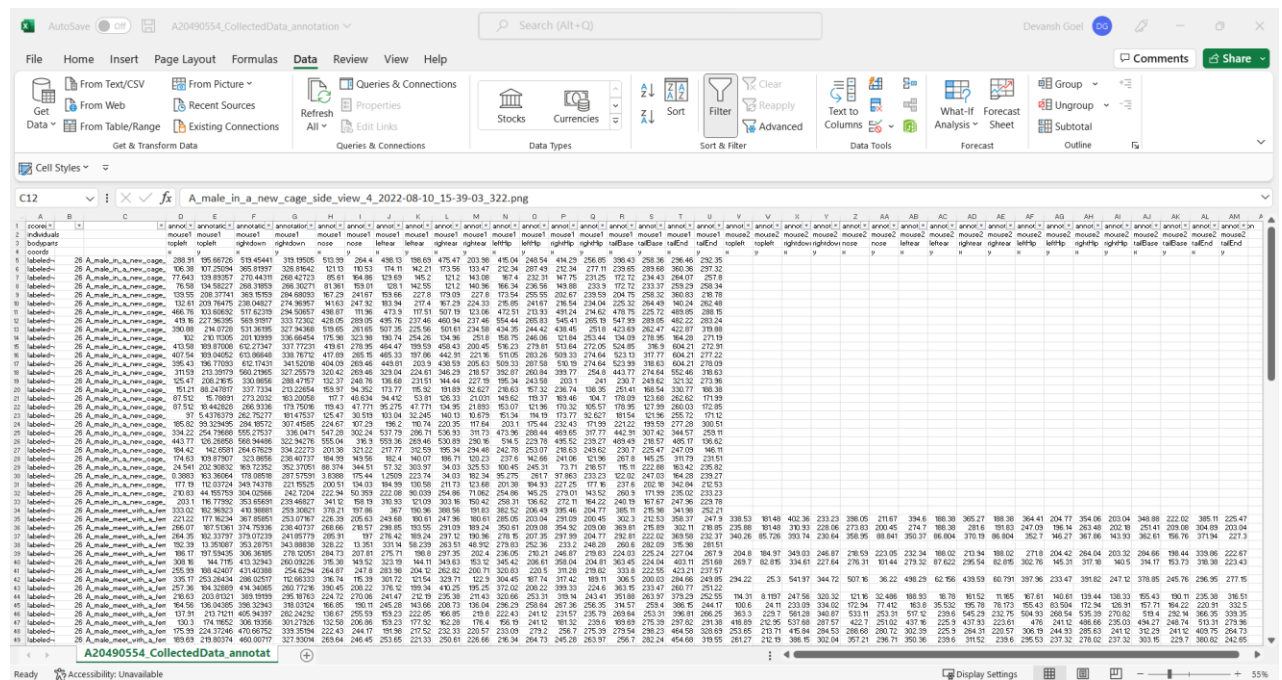
The mouse location ground truth is defined as the average of the corner coordinates of the bounding box, i.e., center = $((x_1+x_2)/2, (y_1+y_2)/2)$.

The data is split into testing and training sets. 70% of the data is used for training and 30% for testing and evaluation. The neural network is used to predict the ground truth which indicates the location of the mouse.

Structure of the model used

Data Collection:

The deep lab cut tool us used to annotate the mouse frame by frame. After annotating the mouse with 7 data points (“nose”, “left ear”, “right ear”, “left hip”, “right hip”, “tail base”, “tail end”) a csv file is created as shown below.



This file consists of all the annotated points in the form of coordinates.

Preparing Data:

- a) **Data Cleaning:** - The following columns are dropped: 'scorer', 'Unnamed: 1', 'Unnamed: 2'. Columns are renamed as shown below for better understanding.

```
df = df.drop(columns=['scorer', 'Unnamed: 1', 'Unnamed: 2'])
df.rename(columns={'annotation': 'mouse1 topleft x', 'annotation.1': 'mouse1 topleft y',
'annotation.2': 'mouse1 rightdown x', 'annotation.3': 'mouse1 rightdown y',
'annotation.4': 'mouse1 nose x', 'annotation.5': 'mouse1 nose y',
'annotation.6': 'mouse1 leftear x', 'annotation.7': 'mouse1 leftear y',
'annotation.8': 'mouse1 rightear x', 'annotation.9': 'mouse1 rightear y',
'annotation.10': 'mouse1 leftHip x', 'annotation.11': 'mouse1 leftHip y',
'annotation.12': 'mouse1 rightHip x', 'annotation.13': 'mouse1 rightHip y',
'annotation.14': 'mouse1 tailBase x', 'annotation.15': 'mouse1 tailBase y',
'annotation.16': 'mouse1 tailEnd x', 'annotation.17': 'mouse1 tailEnd y',
'annotation.18': 'mouse2 topleft x', 'annotation.19': 'mouse2 topleft y',
'annotation.20': 'mouse2 rightdown x', 'annotation.21': 'mouse2 rightdown y',
'annotation.22': 'mouse2 nose x', 'annotation.23': 'mouse2 nose y',
'annotation.24': 'mouse2 leftear x', 'annotation.25': 'mouse2 leftear y',
'annotation.26': 'mouse2 rightear x', 'annotation.27': 'mouse2 rightear y',
'annotation.28': 'mouse2 leftHip x', 'annotation.29': 'mouse2 leftHip y',
'annotation.30': 'mouse2 rightHip x', 'annotation.31': 'mouse2 rightHip y',
'annotation.32': 'mouse2 tailBase x', 'annotation.33': 'mouse2 tailBase y',
'annotation.34': 'mouse2 tailEnd x', 'annotation.35': 'mouse2 tailEnd y'}, inplace=True)
```

The data frame values are converted to float values.

- b) **Calculation of ground Truth:** The ground truth is calculated using the below formula.

Ground Truth

```
df['mouse1 center x'] = (df['mouse1 topleft x'] + df['mouse1 rightdown x'])/2
df['mouse1 center y'] = (df['mouse1 topleft y'] + df['mouse1 rightdown y'])/2
df['mouse2 center x'] = (df['mouse2 topleft x'] + df['mouse2 rightdown x'])/2
df['mouse2 center y'] = (df['mouse2 topleft y'] + df['mouse2 rightdown y'])/2
```

- c) **Handling null values:** For this assignment null values are handled in 2 ways: - Filling NULL values with 0 and removing the rows which contain NULL values.
- d) **Feature Scaling:** Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization. For this assignment the variables have been scaled with the range of -1 to 1.
- e) **Testing and training subset:** The data is divided into testing (70%) and training subset (30%).

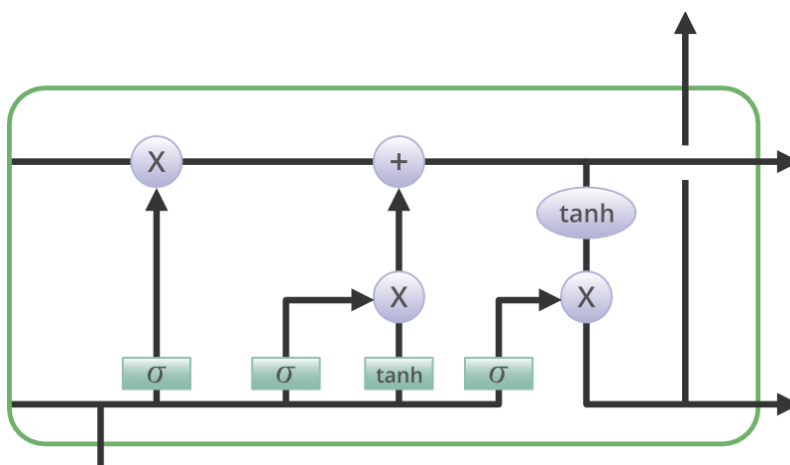
Choosing a Model:

Model used in this Assignment: LSTM

Long Short-Term Memory is a kind of recurrent neural network. In RNN output from the last step is fed as input in the current step. It tackled the problem of long-term dependencies of RNN in which the RNN cannot predict the word stored in the long-term memory but can give more accurate predictions from the recent information. As the gap length increases RNN does not give an efficient performance. LSTM can by default retain the information for a long period of time. It is used for processing, predicting, and classifying on the basis of time-series data.

Structure of LSTM

LSTM has a chain structure that contains four neural networks and different memory blocks called **cells**.



Information is retained by the cells and the memory manipulations are done by the **gates**. There are three gates: -

1. **Forget Gate:** The information that is no longer useful in the cell state is removed with the forget gate. Two inputs x_t (input at the particular time) and h_{t-1} (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use.
2. **Input gate:** The addition of useful information to the cell state is done by the input gate. First, the information is regulated using the sigmoid function and filter the values to be remembered similar to the forget gate using inputs h_{t-1} and x_t . Then, a vector is created using \tanh function that gives an output from -1 to +1, which contains all the possible values from h_{t-1} and x_t . At last, the values of the vector and the regulated values are multiplied to obtain the useful information.
3. **Output gate:** The task of extracting useful information from the current cell state to be presented as output is done by the output gate. First, a vector is generated by applying \tanh function on the cell. Then, the information is regulated using the sigmoid function and filter by the values to be remembered using inputs h_{t-1} and x_t . At last, the values of the vector and the regulated values are multiplied to be sent as an output and input to the next cell.

Neural Network Configuration:

Given that the output layer is the result layer, this layer has 1 neuron present by default. The number of neurons in each layer is configured as follows:

- **Input layer:** The number of neurons in the input layer is calculated as follows:

Number of features in the training set + 1

In this case, as there were 7 features in the training set to begin with, 8 input neurons are defined accordingly.

- **Hidden layer:** One hidden layer is defined, as a single layer is suitable when working with most datasets. The number of neurons in the hidden layer is determined as follows:

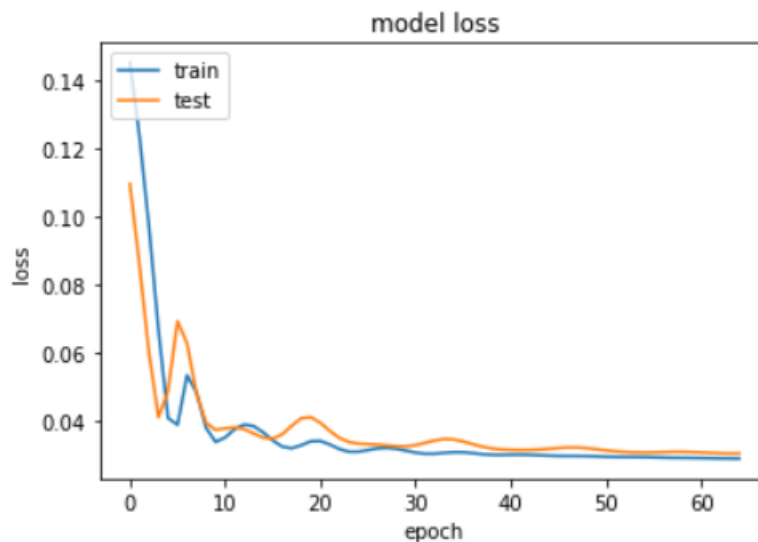
Training Data Samples/Factor * (Input Neurons + Output Neurons)

A factor of 1 is set in this case, the purpose of the factor being to prevent overfitting.

- **Output layer:** As this is the result layer, the number of nodes depends on the output required.

Epochs: 65 epochs have been specified for our model. This means that we are essentially training our model over 65 forward and backward passes, with the expectation that our loss will decrease with each epoch, meaning that our model is predicting the value of y more accurately as we continue to train the model.

Model loss plot:



As seen from the above figure, Both the training and validation loss decrease in an exponential fashion as the number of epochs is increased, suggesting that the model gains a high degree of accuracy as our epochs (or number of forward and backward passes) is increased.

The predictions generated using the features from the validation set can now be compared to the actual values from that validation set.

The predictions are scaled back to the original scale:

Predicted Values

```
predictions = scaler_y.inverse_transform(predictions)
predictions
```

L2 loss function is used for model evaluation.

Experiments

There are total 6 experiments performed in this assignment. These are listed below:

- Prediction of position of mouse 1 with null values as 0
- Prediction of position of mouse 2 with null values as 0
- Prediction of position of mouse 1 after dropping null rows
- Prediction of position of mouse 2 after dropping null rows
- Prediction of position of mouse 1 and mouse 2 with null values as 0
- Prediction of position of mouse 1 and mouse 2 after filling the null values with column mean

Overall Results:

L2 loss is used for model evaluation

The loss for the above 6 experiments is listed below:

Final L2 losses for 6 experiments performed

```
: print("L2 Loss of mouse 1 with null values as 0: " + loss_mouse_1_null_0)
print("L2 Loss of mouse 2 with null values as 0: " + loss_mouse_2_null_0)
print("L2 Loss of mouse 1 after dropping null rows: " + loss_mouse_1_null_drop)
print("L2 Loss of mouse 2 after dropping null rows: " + loss_mouse_2_null_drop)
print("L2 Loss of mouse 1 and mouse 2 with null values as 0: " + loss_mouse_1_mouse_2_null_0)
print("L2 Loss of mouse 1 and mouse 2 after filling the null values with column mean: " + loss_mouse_1_mouse_2_null_column_mean)
```

```
L2 Loss of mouse 1 with null values as 0: 994203.3563294889
L2 Loss of mouse 2 with null values as 0: 3050383.2687006993
L2 Loss of mouse 1 after dropping null rows: 151522.07052295344
L2 Loss of mouse 2 after dropping null rows: 524716.8041368042
L2 Loss of mouse 1 and mouse 2 with null values as 0: 5241100.221822384
L2 Loss of mouse 1 and mouse 2 after filling the null values with column mean: 1954149.0905619296
```

Instructions to run and general outline of the code

The code is divided into 6 experiments as mentioned in the “Experiments” section.

These experiments are executed in the same mentioned order.

- 1) There are 3 main functions in the code namely: -
 - def lstm (): - this function is used to fit the model by defining the input, hidden and output layers. This function uses a custom loss function which represents l2 loss of the model.
 - def featureScaling ():- This function is used for scaling the input parameters.
 - def plotting (): - This function is used plot the model loss.
- 2) All the experiments use the above 3 functions to predict and calculate the model loss.
- 3) The project files are stored in below folder hierarchy
 - data: The produced annotation files. (Contains A20490554_CollectedData_annotation.csv file)
 - src: All program files. (Contains ML Assignment 3 Code. ipynb file)
 - doc: the written report in pdf file. (Contains Assignment 3 report)
- 4) The first step to run the code is to download and save the path of the A20490554_CollectedData_annotation.csv file.
- 5) Open Jupiter Notebook and upload the program file ML Assignment 3 Code. ipynb
- 6) After the file is successfully uploaded run the file by clicking on cell -> run all.
- 7) The final results are displayed at the end of the program execution.

Conclusion

Machine learning with its modern advancements is being used more and more for predictions and data analysis. One example is presented in this report. Deep lab cut tool is used to annotate the mouse locations using a video divided into many frames. After annotations are done the data is fed into a neural network (LSTM network is used for this assignment) for training and prediction.

The model is trained to predict the ground truth (center location of the mouse) by dividing the data into training and testing subsets. There are 6 experiments preformed in this assignment, and for each experiment model loss (l2) is calculated and compared.