Illinois Institute of Technology

CS - 584 Machine Learning

Assignmnet 3 - DeepLabCut Report

DeepLabCut™:
a software package for
animal pose estimation

Submitted by : Akshay Singh

Instructed by : Prof. Yan Yan

Introduction:

DeepLabCut is an open source toolbox that **builds on a state-of-the-art animal pose estimation algorithm**. It allows training of a deep neural network by using limited training data to precisely track user-defined features, so that the human labeling accuracy will be matched.

Problem Statement:

Train a model to predict mouse location and evaluate your model.

Approach:

There are two primary sections to the project. The first part's goal is to collect approximately 125 photos of various mouse frames from a video, annotate the joints of the nose, left ear, right ear, left hip, right hip, tail base, and tail end using the DeepLabCut library, and export the data as a CSV record. The second section of the project uses neural networks to forecast where the mouse will be given the inputs from the seven joints. The "Lefttop" and "Rightdown" points centers are referred to be the mouse location ground truth.

Structure of the project:

1. Data set annotations using DeepLabCut-master tool
2. Data set cleaning
3. Building Deep learning model architectures
4. Evaluation of the model

Data Preprocessing:

The data we had after annotation had 128 rows and 39 columns. The first thing I did in the pre-processing phase was to exclude the first three rows of the dataframe because they contained only information for me and weren't necessary for the creation of a model. After that, I made the decision to divide the dataframe into two sets: one for mouse 1 and one for mouse 2. I did this because, after determining their ground truth, I wanted to combine these two dataframes vertically rather than horizontally. The x and y coordinates of the center points, determined using the formula ((x1+x2)/2, (y1+y2)/2) were then inserted as 2 new columns to both dataframes. After that, I gave all of the columns new names to make them easier to understand.

Data Analysis:

Data set consists of 125 images. Each image will have at least one mouse and maximum of 2 mouse. Annotated data consists of total 198 mouses.

```
concatenated_mouse_data["avg_cord_x"] = (concatenated_mouse_data["mouse_topleft_x"] + concatenated_mouse_data["mouse_rightdown_x"
concatenated_mouse_data["avg_cord_y"] = (concatenated_mouse_data["mouse_topleft_y"] + concatenated_mouse_data["mouse_rightdown_y"
concatenated_mouse_data
```

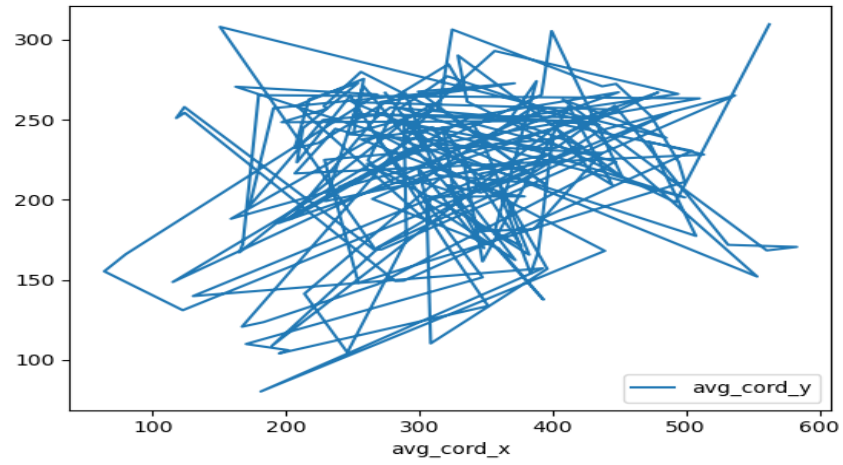
	mouse_topleft_x	mouse_topleft_y	mouse_rightdown_x	mouse_rightdown_y	mouse_nose_x	mouse_nose_y	mouse_leftear_x	mouse_leftear_y	mouse_righ
3	148.109893	206.930891	400.534239	326.859385	198.233608	240.757279	178.935832	217.950817	228.0
4	339.458237	92.515277	404.217447	232.299405	394.875179	106.913600	381.634676	107.995846	397.5
5	245.845587	154.776872	361.708926	243.011260	340.545609	205.714230	338.816245	188.918192	352.1
6	291.332599	179.191036	408.273140	241.189998	332.629619	212.319274	362.680136	205.992849	351.6
7	306.583971	171.018428	402.420756	237.941742	332.415086	210.327094	321.228789	201.385267	337.2
...	...	...	...	...	...	...	...	...	
122	358.678821	119.365051	545.861912	308.796588	387.908613	154.215957	412.079403	164.896073	415.4
123	199.498467	234.546932	458.230002	345.857810	235.939529	252.104898	228.320034	274.300817	245.2
124	225.525961	178.700097	446.110864	344.020688	286.930752	195.232156	257.645390	211.291871	280.3
125	428.022960	99.020979	556.145884	297.809640	490.103140	118.173375	479.536301	144.590473	504.6
127	490.386906	262.438823	634.555011	356.949025	556.463954	343.333149	586.498976	307.691590	554.4

198 rows × 20 columns

Plot of mouse movement

```
concatenated_mouse_data.plot(x = 'avg_cord_x', y = 'avg_cord_y')
```

```
<AxesSubplot:xlabel='avg_cord_x'>
```



```
In [63]: concatenated_mouse_data.shape
Out[63]: (198, 20)
```

```
In [27]: concatenated_mouse_data.describe()
```

Out[27]:

	mouse_topleft_x	mouse_topleft_y	mouse_rightdown_x	mouse_rightdown_y	mouse_nose_x	mouse_nose_y	mouse_leftear_x	mouse_leftear_y	mouse_ri
count	198.000000	198.000000	198.000000	198.000000	198.000000	198.000000	198.000000	198.000000	19
mean	220.390739	142.338446	427.215225	295.325803	317.861569	196.462504	314.803232	185.749820	31
std	111.501012	64.762133	107.630270	44.838847	144.111329	83.569917	135.184201	72.379429	13
min	6.508356	-0.465971	112.862820	147.742529	24.156089	1.761391	35.941286	2.146928	2
25%	138.730163	98.126517	361.205585	263.416640	198.605263	133.766254	202.426077	137.146574	20
50%	215.396813	162.967566	423.927964	309.742057	326.890044	207.568748	318.306079	195.300229	31
75%	292.131840	193.066344	517.884024	331.049370	431.669635	251.755726	408.720205	234.548545	41
max	534.017133	268.857729	634.987726	357.934836	618.406244	351.280902	616.737273	340.155510	58

Structure of ML Model:

Train and test splits were done at 70% and 30%

Test-Train Split

```
In [64]: from sklearn.model_selection import train_test_split

In [65]: X  = concatenated_mouse_data.iloc[: , 4:18]
         Y  = concatenated_mouse_data.iloc[: , 18:]
         X_train, X_test, Y_train, Y_test = train_test_split(X, Y,random_state=100, test_size=0.30)

In [66]: X_train.shape
Out[66]: (138, 14)

In [67]: X_test.shape
Out[67]: (60, 14)

In [68]: Y_train.shape
Out[68]: (138, 2)

In [69]: Y_test.shape
Out[69]: (60, 2)
```

Neural Network:

To anticipate the ground truth, I utilized the keras sequential model from the tensorflow library, a neural network model. The input layer, one of my model's three layers, contains 14 nodes that stand in for the model's 14 input variables. Here, I've utilized the relu activation function.

In [39]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [40]:
```
neural_model = Sequential()
neural_model.add(Dense(14, input_dim = 14, activation='relu'))
neural_model.add(Dense(14, activation = 'relu'))
neural_model.add(Dense(2))
```

In [49]:
```
neural_model.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_41 (Dense)            (None, 14)                210

 dense_42 (Dense)            (None, 14)                210

 dense_43 (Dense)            (None, 2)                 30

=================================================================
Total params: 450
Trainable params: 450
Non-trainable params: 0
_________________________________________________________________
```

Evaluation:

In [50]:
```
neural_model.compile(loss='mean_squared_error',optimizer='adam', metrics=['accuracy'])
```

In [51]:
```
neural_model.fit(X_train, Y_train, epochs = 100, validation_data = (X_test, Y_test))
```

```
Epoch 95/100
5/5 [==============================] - 0s 9ms/step - loss: 126.6010 - accuracy: 0.9493 - val_loss: 126.7244 - val_accuracy:
0.9167
Epoch 96/100
5/5 [==============================] - 0s 10ms/step - loss: 122.6555 - accuracy: 0.9420 - val_loss: 138.7395 - val_accuracy:
0.9000
Epoch 97/100
5/5 [==============================] - 0s 9ms/step - loss: 127.2658 - accuracy: 0.9493 - val_loss: 137.3041 - val_accuracy:
0.9000
Epoch 98/100
5/5 [==============================] - 0s 9ms/step - loss: 128.9690 - accuracy: 0.9493 - val_loss: 136.1398 - val_accuracy:
0.9167
Epoch 99/100
5/5 [==============================] - 0s 9ms/step - loss: 129.7719 - accuracy: 0.9493 - val_loss: 133.4458 - val_accuracy:
0.9000
Epoch 100/100
5/5 [==============================] - 0s 9ms/step - loss: 125.6325 - accuracy: 0.9493 - val_loss: 126.2400 - val_accuracy:
0.9333
```

Out[51]:
```
<keras.callbacks.History at 0x1db61560910>
```

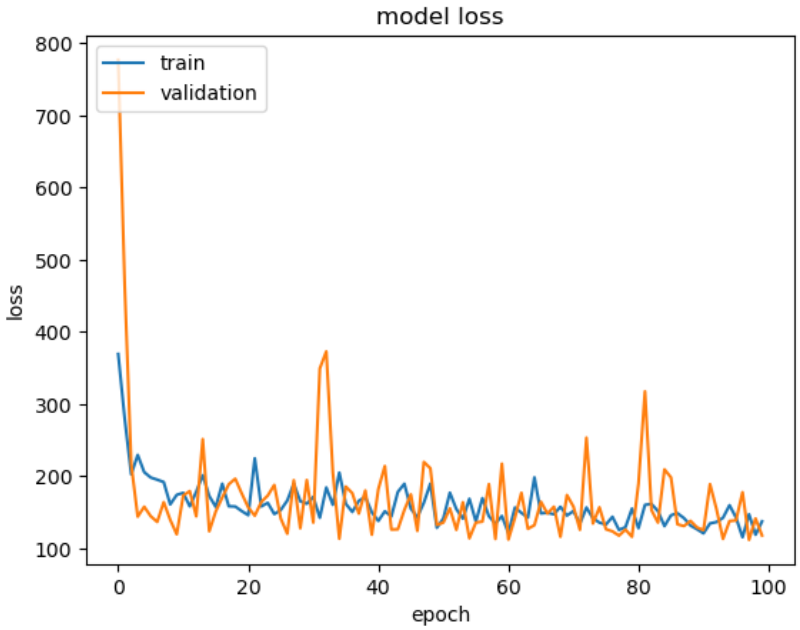
In [61]:
```
accuracy = neural_model.evaluate(X_test, Y_test, verbose=0)
print(accuracy[1]*100)
```

```
93.33333373069763
```

Result and Analysis:

Below are the results taken from running our model

```
In [43]: plt.plot(history.history['loss'])
         plt.plot(history.history['val_loss'])
         plt.title('model loss')
         plt.ylabel('loss')
         plt.xlabel('epoch')
         plt.legend(['train', 'validation'], loc='upper left')
         plt.show()
```



Conclusion:

In order to execute machine learning, I also employed layers of neural networks in this supervised machine learning model, where the data was split into training and test sets. Results came from a variety of combinations that were explored during the model training procedure. As we have already seen, our model suffers from overfitting, and in order to strengthen it, both the model and the data need to be processed further. I would also add more training and testing data.

Future Scope:

There were times when certain mouse body parts were hidden when I was annotating the images with the DeepLabCut library. In certain situations, I didn't designate that specific body part for a mouse. Instead of leaving it blank, making an educated guess as to where that body part would have been and marking it accordingly could have produced more data, increasing the number of rows, and ultimately improving the model.

Instructions for running your code on the data to reproduce your results:

Step 1. From the attached files you can use the ML-Assignment 3.ipynb.
Step 2. Run the first code block where def viz(history) is defined
Step 3. Load the annotated CSV file (A204914651_CollectedData_annotation)
Step 4: Run allthe chunks of Code
Step5: Weights are saved to achieve similar results of accuracy, attached are json file that you'll find along with ipynb