ECE 542- Project 3B - Body-Rocking Behaviour Recognition

Srinivas Gopalakrishnan[1] Rajaram Sivaramakrishnan[2] and Priyankha Bhalasubbramanian [3]

I. METHODOLOGY

In this project we have to devise a neural network (specifically CNN-LSTM) model for Body-Rocking Behaviour. We have obtained the data of wearable sensors on arm and wrist for 1-2 hours with which they have determined whether there is Body Rocking.

A. DATA PRE-PROCESSING

For the training data we have 10 training sets of accelerometer and gyroscope data of the sensors at arm and wrist. Since we have to use the features of the data of a continuous period for our model, we decided to convert our data into blocks of 150 x 12 data each which consists of 3 accelerometer readings, 3 gyroscope reading of the arm sensor and the same for the wrist sensor for each data point.

B. POPULATING RESPONSE VARIABLE

For the response variable, we need to do a transformation such that for every window, only one y value (detection) is extracted. There are 2 approaches to address this. One approach is to count the number of 1s (observed body rocking behavior) and the number of 0s (body rocking behavior not observed) and choose the label that has highest frequency, similar to a majority voting. The other approach is to, take the middle label of every window as the response to that particular window. For example, for the first window of 1 to 150, the 75th label will be considered. In this project we have implemented the second approach.

C. CNN FOR FEATURE EXTRACTION

We have used a Convolutional Neural Network to extract features from the data. Below is the structure of CNN used Table I summarizes the structure of our CNN model. We first had a 2D convolutional layer that takes 12 input channels and generates 96 output channels. After doing a MaxPool we feed the outputs from the first layer to the second Convolutional layer that generates 192 output channels. The next layer is a MaxPool and then finally a Dropout layer with a probability of 0.3. For both the convolutional layers we used the ReLU activation function. This network generates the features for the session data to be fed to the classifier to recognise the Body-Rocking behaviour.

Layer	I/P channels	O/P channels	Kernel
2D Conv L1	12	96	3x1
2D MaxPool	96	96	2
2D Conv L2	96	192	7x1
2D MaxPool	192	192	5
Dropout = 0.3			

TABLE I
STRUCTURE OF CNN

D. LSTM

The results obtained from [1] imply that feature learning through CNNs are more effective than hand-crafted features. Moreover when a Long Short-Term Memory model is combined with CNN to perform classifications, the detections are found to be more accurate and robust. So we have used the approach of transfer learning to use the features learnt from CNN to construct our LSTM model. The model consists of 100 hidden units with input channel size 1152 and 100 output channels. This is followed by a linear layer that generates 2 outputs corresponding to detection values 0 or 1. Finally, a softmax activation is applied that is used to select prediction corresponding to the highest value of output.

II. MODEL TRAINING AND HYPER-PARAMETERS TUNING

There are lot of hyper-parameter tuning options available for our project. We have mainly restricted ourselves to batch size, learning rate and epochs. In this project we have fixed the time interval considered for a window of data to be 3 seconds(150 accelerometer and gyroscope measures) and have used the stride length of 25 based on our previous results (Project 3a).

A. VALIDATION SETS AND TEST SET FOR EVALUATION

For the training purposes, we have totally 10 sessions of accelerometer and gyroscope data for both arm and wrist movements. We randomly picked three sessions and used two for validation and one for evaluation. While generating random numbers we also ensure that the sessions are not continuous which will give a biased result. While randomly selecting 3 numbers out of 1, 2, 3, 5, 6, 7, 12, 13, 15 and 16 we obtained 3, 12 and 7. So we proceeded with using sessions 3 and 12 for validation for hyper-parameter tuning and session 7 for evaluation of our predictions when compared to the ground truth detections for the same session 7.

So our data is split as

- **Training Set** = Session01, Session02, Session05 Session06, Session13, Session15, Session16

[1] Srinivas Gopalakrishnan is a student of the Masters program in the ECE Department at North Carolina State University
[2] Rajaram Sivaramakrishnan is a student of the Masters program in the ECE Department at North Carolina State University
[3] Priyankha Bhalasubbramanian is a student of the Masters program in the CSE Department in North Carolina State University

- **Validation Set** = Session03, and Session12
- **Testing Test** = Session07

B. BATCH SIZE

We have used batch size as one of the hyper-parameters to tune our model during training. The values of batch size that we considered are 32, 64 and 128. The below table shows the variation of validation accuracy with respect to batch size across 10 epochs.

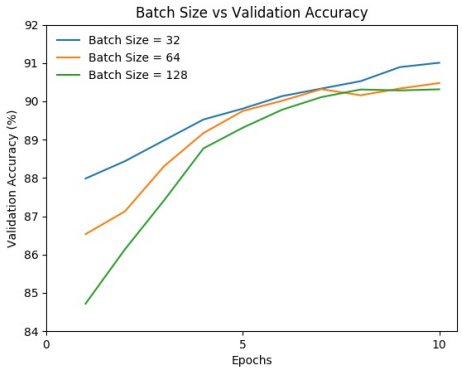

Fig. 1. Validation Accuracy vs Epochs for Batch Size Tuning

C. LEARNING RATE

We have varied the learning rate of the Adam optimizer for hyper-parameter tuning and have considered the values: 0.1, 0.01 and 0.001. We have tuned our model considering the batch size to be 32 at different optimizer learning rates for 10 epochs and the validation accuracy is as shown in Figure 2.

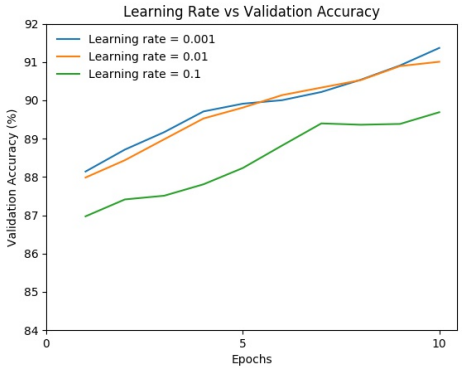

Fig. 2. Validation Accuracy vs Epochs for Learning Rate Tuning

From Figure 2, it is clear that learning rate of 0.001 and 0.01 perform significantly better than that of 0.1. We have chosen the value of 0.01 as our learning rate, despite the value of 0.001 is slightly better because the running time for learning rate of 0.001 is too high.

D. EPOCHS

We trained our model for 30 epochs and used the batch size as 32 and learning rate as 0.01 for inspecting the accuracy and loss curve of our model for training and validation sets. As expected, the model learns incrementally over every epoch and the plots also show that the training accuracy keeps increasing through every iteration. However, the validation accuracy peaks at 16^{th} epoch, which shows that the best model parameters (one that is most accurate for validation set) is obtained at 16^{th} epoch. Consequently, the loss curve shows a dip in the validation loss corresponding to the 16^{th}. Hence we would restrict to 16 epochs during the training phase of our model.

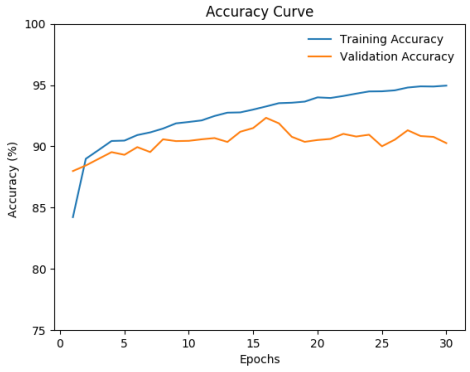

Fig. 3. Accuracy Curve of the Model

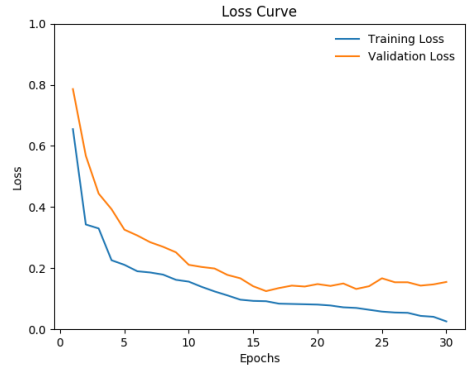

Fig. 4. Loss Curve of the Model

E. OBSERVATIONS

From the tuning of the hyper-parameters, our model was decided as

- Time Interval = 3 seconds
- Stride Length = 25
- Batch size = 32
- Learning rate = 0.01
- Number of epochs = 16

We have observed that these values give the maximum validation accuracy in optimum time. The observed prediction and ground truth detection data for Session 07 are as shown in Figure 5. The overlap clearly shows the model accuracy for which the evaluation metric is shown in next section.

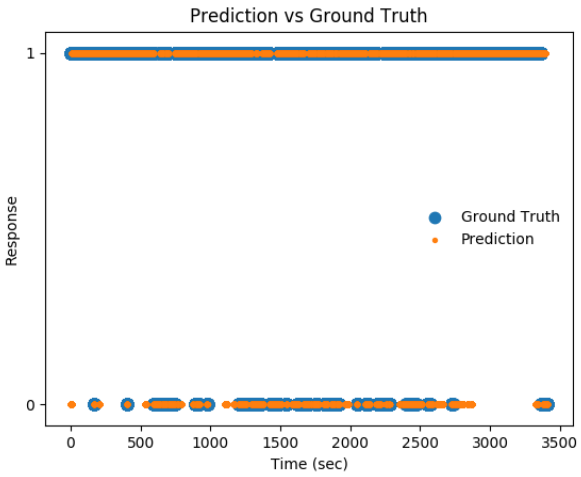

Fig. 5. Ground Truth vs Prediction for Session 07

III. EVALUATION

A. EVALUATION METRICS

During the validation phase for hyper-parameter tuning we considered the data from Sessions 7 as testing set. So, without loss of generality, we consider the same set to estimate the performance of our model using error metrics like Accuracy, Precision, Recall and F1 measure.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

where,
TP = True Positives, i.e., number of positive(1) observations predicted as positive (1)
TN = True Negatives, i.e., number of negative(0) observations predicted as negative (0)
FP = False Positive, i.e., number of negative(0) observations predicted as positive (1)
FN = False Negatives, i.e., number of positive(1) observations predicted as negative (0)
Using these measures we observe the following results:

```
srinivas@srinivas-ubuntu:~/Desktop/NNProject 3$ python3 evalfinal.py


Evaluation of Session 07
Precision: 0.7804878048780488
Recall: 0.9696969696969697
F1: 0.8648648648648649
srinivas@srinivas-ubuntu:~/Desktop/NNProject 3$
```

Fig. 6. Evaluation Result

The performance metrics correspond to the results obtained from Session 7 in Figure 6.

B. PREDICTIONS FOR TEST DATA

Finally, we train our model using all the sessions 1, 2, 3, 5, 6, 7, 12, 13, 15 and 16. Using this trained model we predict the detections for the test set consisting of Sessions 8, 9, 11 and 17. For the purpose of generating our predictions for every row of the aforementioned test sessions, we generate the features similar to how we did while training, with window of 3 seconds (150 rows), but the only difference is that the stride length was 1. So, our aim is to generate a row of features for every row of input data (except the first 150). For the first 3 seconds (150 rows) we have assumed that the predictions are 0 (no observed body rocking behavior), because the general pattern after inspecting all the detections of training sessions is that the subject does not show the behavior for almost the first 20-30 seconds. Hence, it is fair to assume that the predictions for the first 3 seconds of test set is 0. For the remaining rows we feed our model with the extracted features, generate the prediction and append it to out prediction file for that session. This is repeated for all sessions and we successfully generate our prediction results.

C. EVALUATION SCOREBOARD

The Precision, Recall and Evaluation of our Models from Project 3A and the gradual improvement made by using CNN-LSTM for this project is shown in Table II.

Model	Precision	Recall	F1 Score
SVM(Project 3A)	0.223	0.750	0.343
CNN-LSTM(4/16/19)	0.041	0.986	0.078
CNN-LSTM(4/19/19)	0.548	0.919	0.687

TABLE II
EVALUATION METRICS FOR MODELS OF PROJECT 3A AND PROJECT 3B

REFERENCES

[1] Mohammadian Rad, N., Kia, S., Zarbo, C., van Laarhoven, T., Jurman, G., Venuti, P., Marchiori, E. and Furlanello, C. (2019). Deep learning for automatic stereotypical motor movement detection using wearable sensors in autism spectrum disorders.