Body-Rocking Behavior Recognition

Varsha Nagarajan
Computer Science
North Carolina State University
vnagara2@ncsu.edu

Ravi Sanker
Computer Science
North Carolina State University
rsanker@ncsu.edu

Vandan Phadke
Computer Science
North Carolina State University
vphadke@ncsu.edu

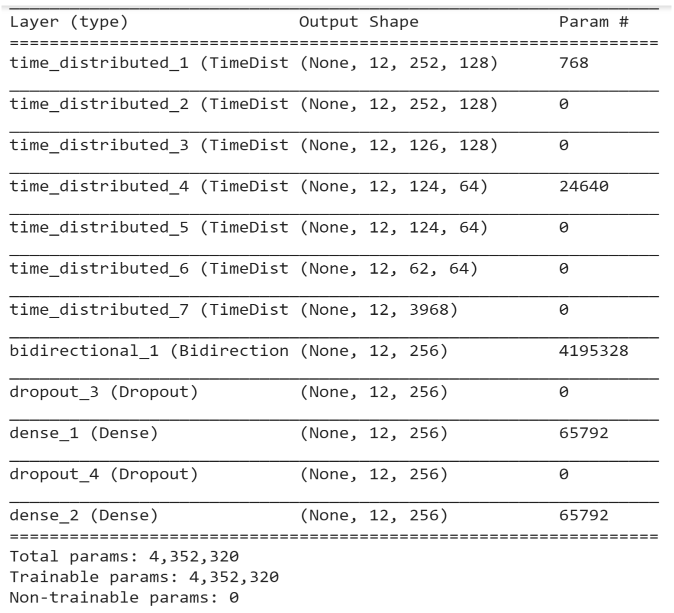
I. METHODOLOGY

In this second phase of the project, we propose a neural network based machine learning (ML) classification model for the detection of body-rocking behavior in blind subjects using inertial measurements from a wearable system. Two sensors, one on the wrist and the other on the arm, are used to record motion using accelerometers and gyroscopes. The training as well the test data consists of several sessions, each of about 1 to 2 hours long and sampled at 50 Hz, with annotations indicating when the behavior was observed.

In the first phase of this project, we presented the performance of various traditional machine learning algorithms. Having analyzed the limitations of such methods in modeling the implicit complexity of time-series data, we surveyed various deep learning models and decided to evaluate the effectiveness of Long Short-Term Memory (LSTM) networks which have produced state-of-the-art results in tasks that involve modeling of sequential data [1]. Since time-series is one form of sequential data, this study implements an LSTM network to capture and detect body-rocking behavior. In this section we present the methodology and key components that dictate our proposed model. Section II explains how the optimal parameters and hyper-parameters have been chosen for our model. Section III evaluates the performance of our model on unseen data.

We engineer the input sequences to be fed to the model by segmenting the raw sensor data over periods of time (5.12 seconds) using a fixed-length sliding window technique with 50% overlap. Since the raw data has been recorded with a sampling frequency of 50Hz, this essentially translates to data slices of 256 timesteps each. The output sequences comprise of corresponding annotations for all the 256 timesteps modeled as a multi-output RNN where we train our model to learn the outcome of every datapoint in the input sequence. The sliding window size was treated as a hyperparameter during model evaluation and 5.12 seconds seemed to work best for us. Unlike phase 1, where we extracted time and frequency domain signals and various related features from the raw sensor data, here, we let the LSTM networks perform feature engineering on their own.

After generating the input and output sequences in the required format, we now implement a CNN-LSTM model to learn the characteristic data features, and subsequently make accurate classification predictions. We design a sequential model comprising of two 1D convolutional layers distributed over time with filter sizes of $\{128, 64\}$ and kernel sizes of $\{5, 3\}$. The output of the last convolutional layer is passed through a max-pooling layer with a stride of 2. These sequences of layers are then followed by one bidirectional-LSTM layer and two-fully connected layers - the first dense layer vectorizes the output of the bidirectional-LSTM layer into a 256-dimensional feature vector; and the last dense layer comprises of neurons equal to the number of timesteps (256) each individually evaluated using binary cross-entropy loss and use a *sigmoid* activation to arrive at the final probabilities of action or not-action states. The network uses rectified linear units (ReLU) as the activation function for each of the convolutional and LSTM layers, a *tanh* activation for the fully connected layers and *sigmoid* activation for the output layer. In order to prevent overfitting, a dropout layer with a dropout rate of 0.2 is added after every convolutional layer and a dropout rate of 0.5 is added after every LSTM and fully connected layers. Also, we use an Adam optimizer with a learning rate of 0.001 and a decay of 10^{-4} and the gradients are updated via backpropagation through time (BPTT). Figure 1 summarizes our network architecture.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed_1 (TimeDist (None, 12, 252, 128)      768
_________________________________________________________________
time_distributed_2 (TimeDist (None, 12, 252, 128)      0
_________________________________________________________________
time_distributed_3 (TimeDist (None, 12, 126, 128)      0
_________________________________________________________________
time_distributed_4 (TimeDist (None, 12, 124, 64)       24640
_________________________________________________________________
time_distributed_5 (TimeDist (None, 12, 124, 64)       0
_________________________________________________________________
time_distributed_6 (TimeDist (None, 12, 62, 64)        0
_________________________________________________________________
time_distributed_7 (TimeDist (None, 12, 3968)          0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 12, 256)           4195328
_________________________________________________________________
dropout_3 (Dropout)          (None, 12, 256)           0
_________________________________________________________________
dense_1 (Dense)              (None, 12, 256)           65792
_________________________________________________________________
dropout_4 (Dropout)          (None, 12, 256)           0
_________________________________________________________________
dense_2 (Dense)              (None, 12, 256)           65792
=================================================================
Total params: 4,352,320
Trainable params: 4,352,320
Non-trainable params: 0
```

Fig. 1. Model Architecture

II. Model Training and Hyperparameter Selection

The dataset we currently use has detections for 10 sessions. In order to validate the performance of our model and subsequently tune the hyperparameters, we perform validation using a holdout set comprising of two sessions. The rest eight sessions are used to train the model. Since every session encapsulates temporal information that are correlated, we do not use time windows from the same session in both training and validation sets as we want the model to learn all trends seen over the entire range of each session. So, instead of splitting the resultant input-output sequences using a random or stratified train-test split, we dedicate two separate sessions just for validation purposes. The choice of which sessions go into training or validation sets was random and we did not observe any substantial changes in the model performance with different choices of sessions in either of the sets. As already mentioned before, we use a window-size of 256 timesteps with 50% overlap which resulted in 8850 training samples and 913 validation samples with 12 features.

The duration of time window and the percentage of overlap for generating input sequences were also treated as hyperparameters. We experimented with window sizes of 50, 100, 128 and 256 and overlap percentages of 0%, 25%, 50% and 75%. For smaller window sizes, our model was unable to capture most of the temporal information embedded in the data and we observed that our model performed best with a combination of window-size of 256 (or 5.12 seconds) with 50% overlap. Since higher overlaps did not result in significantly better results, we chose to go with 50% which produced comparable predictions in fairly lesser training time.

In addition to this, we also experimented with different models designed using various combinations of convolutional and bidirectional-lstm layers. One such experimentation was with a CNN-LSTM model comprising of three 1D convolutional layers, two bidirectional-lstm layers followed by two fully-connected layers of 256 units each with *tanh* activations and an output layer with *sigmoid* activation. We also evaluated a model consisting of three time-distributed convolutional layers with kernel sizes of {16, 8, 4} and filter sizes of {128, 64, 32}, followed by max-pooling layer and three Bidirectional LSTM layers of size {128, 128, 128}. Dropout layers with a dropout rate of 0.5 was added after every layer. On evaluating all these models, we observed that the nature of data was such that even overly complex models with sufficient regularization was not able to give any substantial improvement in prediction accuracy. Hence, we decided to work with the proposed simple model that trains faster and produces predictions with comparable accuracy. Among the other hyperparameters we tried experimenting with are the learning rate, optimizer function, batch size and the number of epochs. With higher learning rates, the validation accuracy and losses over the epochs were pretty bumpy indicating steep jumps across the local minima resulting in delayed and sometimes sub-optimal convergence. Adam optimizer with a learning rate of 0.001 seemed to work best in our case. As for the number of epochs, we trained our model for 50 epochs to note the trend of validation accuracy and losses and observed that beyond some 10 epochs, the model begins to over-fit and hence does not generalize well. So, for our experiments, we fixed on 10 epochs with a batch size of 16. Table I provides a comprehensive list of the range of values tried for different hyperparameters along with the final values that were fixed.

TABLE I
Hyperparameter Tuning Ranges

Hyperparameters	***Range***	***Final***
Learning Rate	0.1, 0.01, 0.001, 0.0001	0.001
Optimizer	RMSProp,SGD, Adam	Adam
Epochs	15, 20, 30, 50	10
Batch Size	16, 32, 64	16

III. Evaluation

For the purpose of evaluating the performance of our proposed models, we use data from Session 12 and 13 as our validation set. The results obtained are presented in Table II. The evaluation metrics used for accessing the model's performance were precision [1], recall, accuracy and F1-score. The F1-score presented in Table II was calculated using the methodology presented by Dr. Lobaton, while the one plotted in Figure 4 is one based on standard measurements without factoring in the concepts of overlap between events and detection interval. We also provide a visual representation of how our predictions compare to the groundtruth for better assessment of our model's performance.

In Figure 2, we observe that the model's training accuracy constantly increases over the number of epochs, reaching values close to 99%, while the validation accuracy fluctuates about the 85% mark. By training the model over 50 epochs, the trends in the validation accuracy and losses (Figure 3) are clearly evident, and based on our observation, we fix on the number of epochs to be 10 as beyond that we observe model over-fitting.

As mentioned before, we used Session 12 and 13 as our validation set and Figures 5 and 6 are the plots indicating our results. On observing the plots, we note that our model seems to perform decently well in detecting body-rocking behavior with few false positives. However, our experimentation with different model architectures and varied set of hyperparameters has lead us to believe that with the availability more training data, LSTMs will be able to eventually model all the complexities accurately and our results will definitely be better. Based on the classification reports and provided visual aids, we are positive that our model performs decently well and we hope to achieve good accuracy on the provided test set.

IV. Predictions

During the analysis of the validation data, it was observed that there were quite a few false positives during the initial

[1]Note: Precision and Recall values are with respect to detection of class of interest.

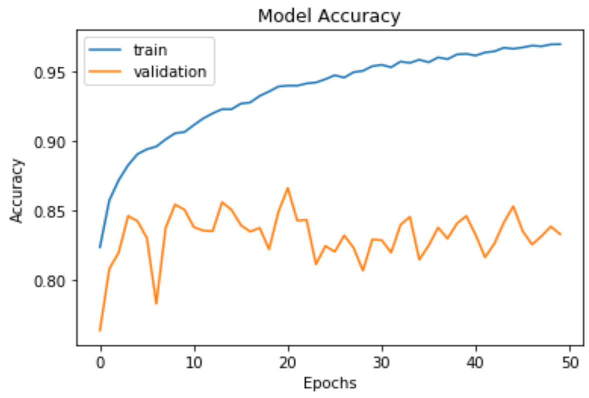


Fig. 2. Training and Validation Accuracy vs Epochs

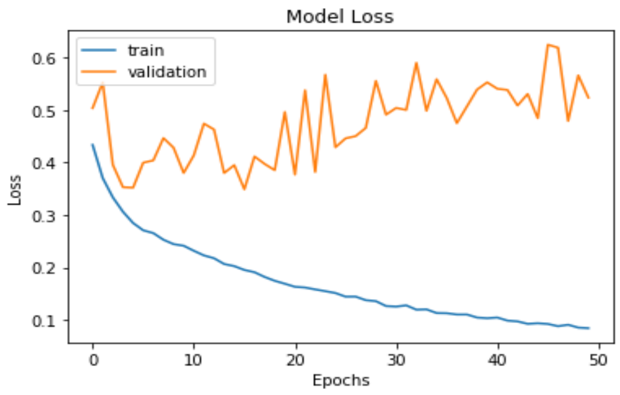


Fig. 3. Training and Validation Loss vs Epochs

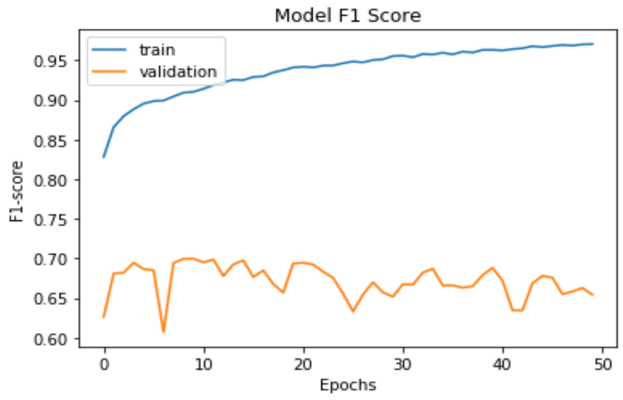


Fig. 4. Training and Validation F1 score vs Epochs

TABLE II
VALIDATION RESULTS

Session	***Accuracy***	***Precision***	***Recall***	***F1-Score***
Session 12	0.847	0.86	0.94	0.89
Session 13	0.89	0.82	1.0	0.90

stages of each session. This might be due to the fluctuations in the data during initialization of measuring devices. Since

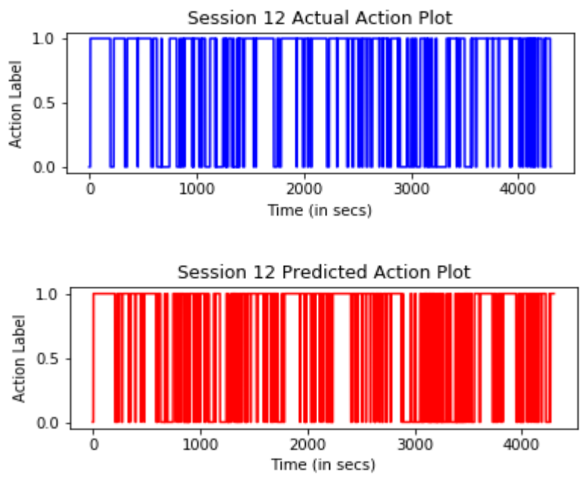


Fig. 5. Validation Plot for Session 12

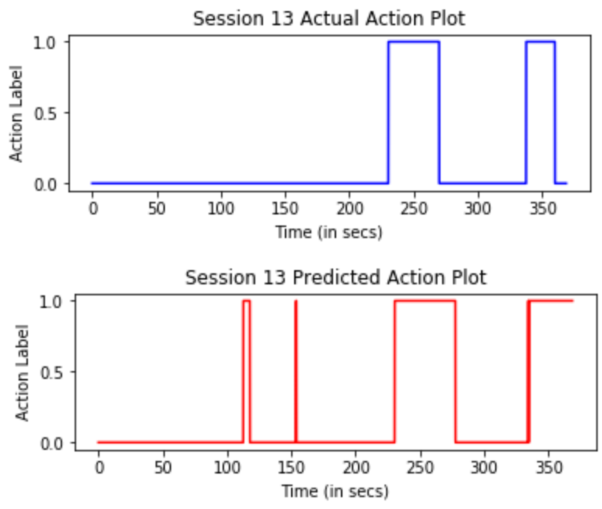


Fig. 6. Validation Plot for Session 13

these were treated as anomalies, predictions for the first few seconds of each session were zeroed out.

Additionally, as part of the post-processing of model predictions, we removed any detection interval smaller than 1 second in length. Similarly, if there were few non-body rocking behavior indicators between a huge span of the detection interval, we replaced all those as valid detection. Through post-processing, we were able to account for possible initial jitters produced by initialization of measuring devices and also avoid few aberrations in detection caused by some factors not directly related to indicators of anomalies.

References

[1] Y. Chen, K. Zhong, J. Zhang, Q. Sun, and X. Zhao, "Lstm networks for mobile human activity recognition," in *2016 International Conference on Artificial Intelligence: Technologies and Applications*, Atlantis Press, 2016/01.