Human Activity Recognition

Nirav Patel
nkpatel8@ncsu.edu

Peeyush Taneja
ptaneja@ncsu.edu

Sneha Aradhey
svaradhe@ncsu.edu

I. METHODOLOGY

In this phase of the proj, we propose a CNN LSTM architecture to predict human activity based on the data gathered from wearable sensors. We have used Convolutional Neural Network (CNN) layers for feature extraction and recurrent neural network architecture LSTM for activity recognition. In the first phase of the project, we compared performance of classical machine learning methods where we did not use sequentiality property of the time series dataset. Recurrent Neural Networks are the state of the art algorithm for sequential data. After comparing different RNN models, model with CNN LSTM has given the best results for the test dataset.

After sampling the training dataset as mentioned in section III-B, we perform undersampling as mentioned in III-D to counter class imbalance. The input sequence is fed to cnn to learn characteristic features of the time-series data. These extracted features are later fed into LSTM model to map the internal representation of time series data to activity type. Below figure summarizes our final model.

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_44 (Conv2D)           (None, 36, 6, 128)        768
_________________________________________________________________
activation_51 (Activation)   (None, 36, 6, 128)        0
_________________________________________________________________
conv2d_45 (Conv2D)           (None, 32, 6, 128)        82048
_________________________________________________________________
activation_52 (Activation)   (None, 32, 6, 128)        0
_________________________________________________________________
conv2d_46 (Conv2D)           (None, 28, 6, 128)        82048
_________________________________________________________________
activation_53 (Activation)   (None, 28, 6, 128)        0
_________________________________________________________________
conv2d_47 (Conv2D)           (None, 24, 6, 128)        82048
_________________________________________________________________
activation_54 (Activation)   (None, 24, 6, 128)        0
_________________________________________________________________
reshape_13 (Reshape)         (None, 24, 768)           0
_________________________________________________________________
bidirectional_2 (Bidirection (None, 24, 256)           918528
_________________________________________________________________
dropout_18 (Dropout)         (None, 24, 256)           0
_________________________________________________________________
lstm_19 (LSTM)               (None, 256)               525312
_________________________________________________________________
dropout_19 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_8 (Dense)              (None, 4)                 1028
_________________________________________________________________
activation_55 (Activation)   (None, 4)                 0
=================================================================
Total params: 1,691,780
Trainable params: 1,691,780
Non-trainable params: 0
_________________________________________________________________
```

Fig. 1: model pipeline

II. TRAINING DATA ANALYSIS

Training data consists accelerometer and gyroscope values on x,y and z axis recorded by 8 different subjects and labels of the activity at each time frame i.e walking on the ground/grass, going up/down the stairs. Fig. 2 below shows the data distribution for different activities on the training dataset. It is clearly visible that the given dataset is imbalanced w.r.t labels assigned.

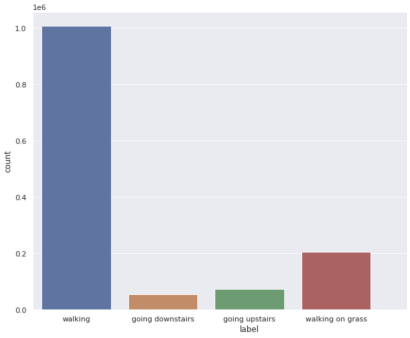


Fig. 2: Class distribution

Another observation which came to our notice is there is no correlation between accelerometer and gyroscope values as oppose to our prior assumption. This is indicated by Fig. 3.

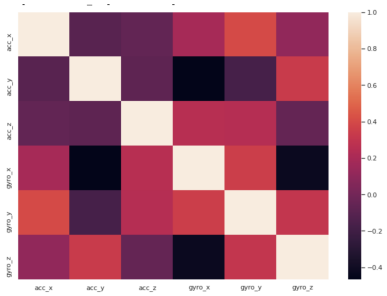


Fig. 3: Correlation heatmap

Readings of accelerometer and gyroscope follows Gaussian distribution and there are no outliers. We considered a data point to be an outlier if it does not belong to 1.5* Interquartile range. Fig.4 and Fig.5 below shows distribution of label values w.r.t mean of accelerometer and gyroscope values over all three axis. It can be seen that walking on the ground and walking on the grass are very similar activities. This is evident in our results too. To exclude any bias in the results due to the outliers or any abnormalities of the data we normalize the data before processing it further to CNN and LSTM. We have used RobustScaler function of 'sklearn' library for the same, it scales the data to be between 1st and 3rd quartiles basis the median.

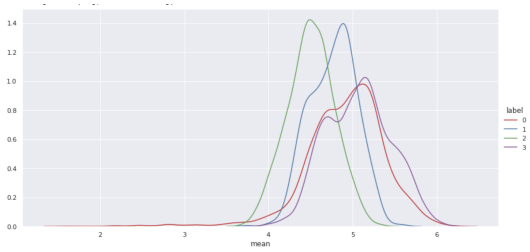


Fig. 4: Histogram over accuracy mean

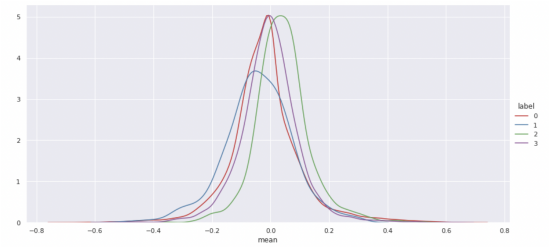


Fig. 5: Histogram over gyroscope mean

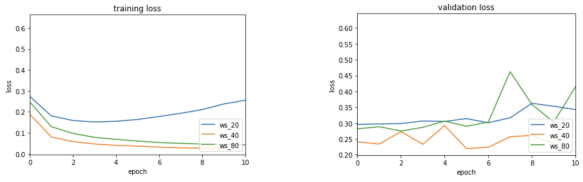
III. MODEL TRAINING

A. FEATURE EXTRACTION

We implemented 2 feature engineering techniques in this project - handcrafted features and Convolution Neural Networks. We handcrafted 12 features from the given data and fed the same tour baseline model. As expected due to limited number of features the model was not able to train fully on the data and reached maximum of **0.75** f1score. In this phase of the project, we have used CNN as a feature extractor. As discussed in Section V CNN performed better on our data and was able to capture the features of the data very well.

B. SAMPLING

We sampled our data into various time frames with window size 40 to convert the time series data into data points. For every window we calculated the mode (highest probability) of the corresponding labels for the window size. We moved the window by 1 step thus covering all the data. Choosing any step size >1 will reduce the number of data points for training. This helped smoothen any disturbances in the data without loosing on data points. It is evident from Fig. 6, window size 40 is turning out to be the best performing on both train and validation set.



(a) training loss (b) validation loss

Fig. 6: train and validation loss for window size

C. VALIDATION

We are dividing the dataset into training and validation based on different episodes. For example, subject_001_01 is considered as one single episode. The following dataset comprises 27 such episodes and we perform a train/val split of 90:10 on these episodes. Training set consists of 24 episodes and the rest 3 are in validation set. The motivation behind this kind of split is to mimic the test data we are provided with.

D. HANDLING IMBALANCE

We tried running our model on both balanced and imbalanced data. Oversampling the training data resulted in a heavier data points which were difficult to train with out computational limitations. We then employed the under-sampling technique to handle data imbalance in the given dataset. For this we used 'imblearn' library's RandomUnderSampler function, which was set to sample all the classes apart from the minority class. The table shows the comparison of model with and without under-sampling technique.

IV. MODEL SELECTION

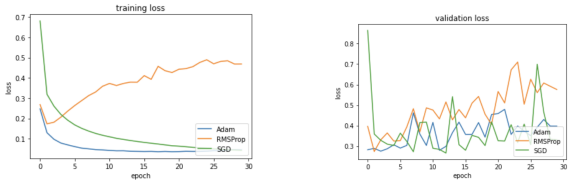
We have selected the different parameters for the model as explained below

- **Model Description:** We have used 2D Convolution network as feature extractor with LSTM as our model. The four layers of convolution network ensure the model is well trained on detailed features of the time series data. We are using LSTM layer for our model as LSTM does not suffer with vanishing gradient problems and provides reliable predictions for long sequential time series data. We have stacked LSTM layer on top of the feature extractor which helps the model to get a better context of the data by training in both directions. Further, an LSTM helps the model classify the data basis the features and the context from previous layers. We have added Dropout layers and tuned the dropout rate to avoid overfitting of the data. LSTM model is further explained below- **LSTM (Long Short Term Memory):** LSTM is a type of recurrent neural networks which is capable of learning important features in a sequence prediction problems. The LSTM gates are capable of learning the sequence of data which is important and should be maintained unlike RNN's due to which LSTM performs best on long sequential data. LSTM also improves on the vanishing gradient issue faced with RNNs. As we have sequantial data for gyroscope and accelerometer readings we are using LSTM for the classification of the same
- **Optimizer Selection:** Choosing the correct optimizer is extremely important in any deep learning model. We tested the below optimizers on our model to check for the best convergence.
  - SGD: A vanila gradient decent optimizer uses whole training dataset to update weights. Since the dataset is very large, training of the model can become very slow. Stochastic Gradient Decent solves this problem by using a single record at a time to update

the weights. Even though SGD performs better than vanila gradient decent, it is slow to converge since it has to calculate forward and back propogation for each record.

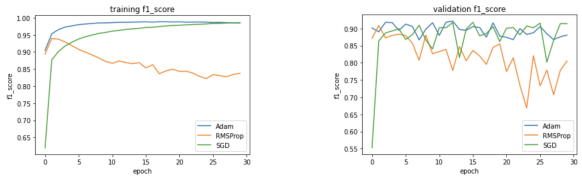
- RMSprop: Gradients of complex function can have vanishing/exploiting issue. RMSprop is a gradient based optimization technique which tackles this issue by using a moving average of squared gradients to normalize the gradients. This normalization restricts oscillations in vertical direction by decreasing the step for large gradients to avoid exploiting and increasing the step for small gradients to avoid vanishing.
- Adam: Adam optimizer combines the advantages of the 2 models RMS Prop and AdaGrad. As discussed earlier the learning rate in RMS prop is decided basis the average first moment (i.e. mean) whereas AdaGrad uses parameter based learning rate. Adam uses both these techniques along with using average of the second moments of gradients to provide a adaptive learning rate while at the same time providing stability(performing well) for even noisy data points.

  As shown in the Figure 7 and 8 , the model with Adam optimizer performs better than the other models during training phase and is much faster during training. During validation also, it seems the best optimizer overall.



(a) training loss (b) validation loss

Fig. 7: train and validation loss for different optimizer



(a) training f1 score (b) validation f1 score

Fig. 8: train and validation f1 score for different optimizer

V. EVALUATION

This section demonstrate the evaluation results we got with our proposed model on best hyper-parameters. We first captured the training and validation loss for each iteration.

From Fig 9. we noted that both training and validation loss value decreased as the epochs progressed. As the training process reach 20 epochs the training loss curve seems to flatten out which indicates that the model converged. The trajectory of validation curve doesn't suggest any kind of overfitting. Fig 10. illustrate the model performance on training and validation set in terms of accuracy. Both the curves follows a similar trend.

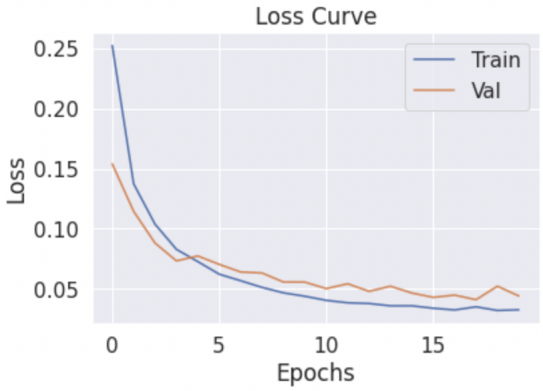


Fig. 9: Training and validation loss values plotted for each epoch

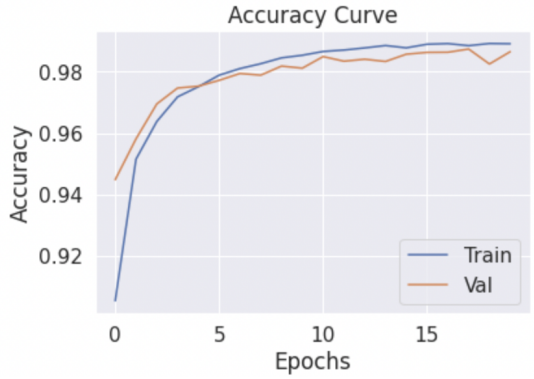


Fig. 10: Training and validation accuracy values plotted for each epoch

After concluding the training process, we calculated different metrics for each class to test our model performance on validation dataset.

	Precision	Recall	F1 Score	Accuracy
Class 0	0.99	0.94	0.97	0.94
Class 1	0.99	0.99	0.99	0.99
Class 2	0.99	0.99	0.99	0.99
Class 3	0.98	0.97	0.98	0.99

Fig. 11: DeepConvLSTM model evaluation metrics

The result shown in Fig 11. are motivating as we a got near perfect F1-score value for all the classes. This is a huge boost if we compare it with our previous report best model

	Precision	Recall	F1-Score	Accuracy
Class 0	0.89	0.84	0.83	0.87
Class 1	0.76	0.91	0.83	0.95
Class 2	0.79	0.98	0.87	0.98
Class 3	0.60	0.52	0.55	0.46

Fig. 12: Previous report best model evaluation metrics

evaluation result (Fig 12). Each class observed a significant increase in both F1 score and accuracy especifically class 3.

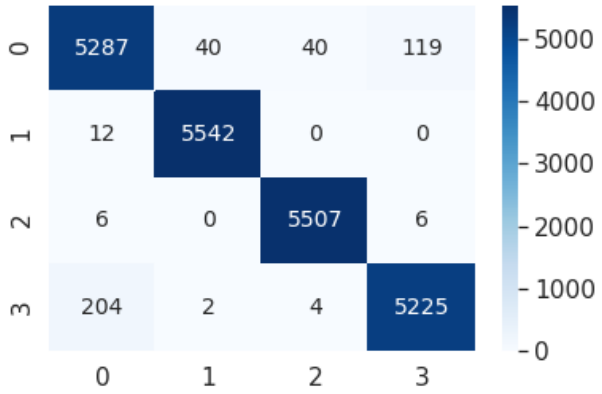


Fig. 13: Confusion Matrix for DeepConvLSTM model

Fig 14 and Fig 15 depicts an instance of predicted labels on the test set as a function of time. The sudden change of action type from class 2 to 3 and then again to 2 (timestamp 6000), gave us an interesting idea of smoothing out these sudden spikes as a post processing steps (Fig 16). With the help of this post processing step we improved our test dataset F1 score from 0.87 to 0.899.

From the above results we can agree that DeepConvLSTM outperforms classical machine learning algorithms and a regular LSTM model for the given task. Also we realise that the strategy of performing undersampling on the given dataset helped us to tackle the problem of class imbalance.

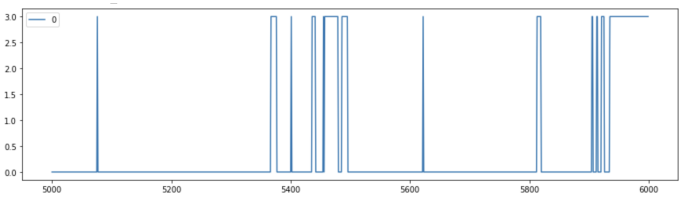


Fig. 14: Instance of predicted labels for subject_012_01

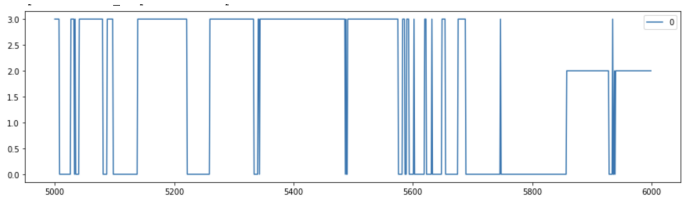


Fig. 15: Instance of predicted labels for subject_010_01

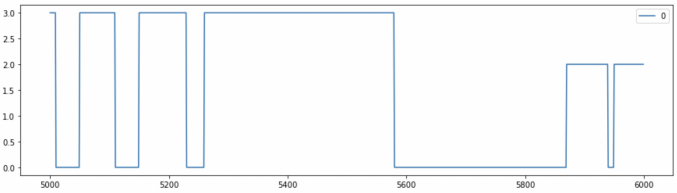


Fig. 16: Smoothing of prediction labels for subject_010_01

REFERENCES

[1] Walse, Kishor & Dharaskar, Rajiv & Thakare, V. M.. (2016). PCA Based Optimal ANN Classifiers for Human Activity Recognition Using Mobile Sensors Data

[2] Ç.Berke Erdaş*, Işıl Atasoy, Koray Açıcı, Hasan Oğul. Integrating features for accelerometer-based activity recognition

[3] Akram Bayat, Marc Pomplun, Duc A. Tran, A Study on Human Activity Recognition Using Accelerometer Data from Smartphones

[4] Andrey Ignatov, Real-time human activity recognition from accelerometer data using Convolutional Neural Networks, Applied Soft Computin

[5] Brownlee, J., 2016. Time series prediction with lstm recurrent neural networks in python with keras. Available at: machinelearningmastery. com, 18.

[6] M. Gholamrezaii and S. M. Taghi Almodarresi, "Human Activity Recognition Using 2D Convolutional Neural Networks," 2019 27th Iranian Conference on Electrical Engineering (ICEE), 2019, pp. 1682-1686, doi: 10.1109/IranianCEE.2019.8786578.