Datathon #5 Team 8 - High-fidelity Prototype

Benjamin Zhang, Xiao Yan, Arij Al Chawaf

Introduction:

Mobile health (mHealth) technologies such as wearable devices, particularly those worn on the arms, have become ubiquitous tools in promoting health and well-being. These devices, which encompass smartwatches and fitness trackers, offer a range of benefits that include remote health monitoring by health practitioners, lifestyle tracking for personal fitness and wellness and the promise of integration with the healthcare system with real-time data and insights. Coupled with advanced AI techniques like deep learning, mHealth can support preventative care including disease detection and management[1]. Using the mobile health dataset (supplied by CHL5230H): multi-dimensional time-series data acquired from wearable sensors placed on the left ankles and right lower arms of nine subjects. Movement data from sensors can support the gathering of kinematic data to understand mobility dynamics that prove useful for athletes[2] and patients[3] alike. While sensors placed on legs and the chest can provide invaluable movement data, developing prediction algorithms that rely on integrating data from sensors on arms only are likely more powerful given the pervasiveness and ease of daily use of arm sensors like smartwatches.

Research Question: Thus, our research question is, "Using only a right-lower-arm sensor data, is it possible to predict the type of action/activity performed by a subject using Recurrent Neural Networks (RNNs) and/or Long Short-Term Memory (LSTMs) network?"

Analysis:

Feature engineering: We explored the dataset using descriptive statistics. Looking at the distribution of activities for each of the nine subjects, we noted that an overwhelming number of counts fell in the 0 (Nothing) class as opposed to all other twelve movement-related classes. Thus, we opted to drop '0' as it's not predicting movement. Subject 9 had activity data in only two classes out of 12 and so we opted to drop subject 9 from further analysis also (**Figure in Appendix- Data Exploration**). Impressively, there was no missing data in the dataset, so there is no need to clean it (**Figure in Appendix- Data Exploration**). The categorical 'subject' column was converted to numerical columns using one-hot encoding. Left-ankle sensor acceleration and gyro data were dropped given our research question's focus on the use of lower-arm sensor data for the prediction of activity.

Data splitting, normalization, sequence creation and target class imbalance: In preparation for RNN and LSTM analyses, we used the train_test_split function from scikit-learn to make an approximately 80:20 split for training and testing after grouping by subject. Training set included data from subjects 8,2,6,1,3 and 4, while the testing set included subjects 5 and 7. Each set was normalized using StandardScaler within each subject grouping. Sequence creation was performed for each activity and subject using create_sequences function followed by padding. The class imbalance in the 'Activity' column in the training set was addressed using

---

[1] Bhatt P. Emerging Artificial Intelligence-Empowered mHealth: Scoping Review. JMIR Mhealth Uhealth. 2022 Jun 9;10(6):e35053. doi: 10.2196/35053.

[2] Adesida Y, Papi E, McGregor AH. Exploring the Role of Wearable Technology in Sport Kinematics and Kinetics: A Systematic Review. Sensors (Basel).

[3] Vijayan V, Connolly JP, Condell J, McKelvey N, Gardiner P. Review of Wearable Devices and Data Collection Considerations for Connected Health. Sensors (Basel). 2021 Aug 19;21(16):5589. doi: 10.3390/s21165589.

oversampling of the ‘12’ (Jump front & back) class. ***RNN:*** The data was prepared using PyTorch's TensorDataset and batches loaded for training and testing into a three layer network (including output layer) with the following hyperparameters and features: hidden_size = 100, num_classes = 12, epochs = 30, learning_rate = 0.0003, batch_size = 300, landa = 0.000001 (lambda regularization term), dropout layer (dropout1) within RNN layers and another (dropout_fc) between fully connected layers to mitigate overfitting , the “Adam” optimizer for model parameter updates and Cross-Entropy Loss function suitable for multi-class classification. The softmax activation function was used to introduce non-linearity to the model. Learning rate scheduling was implemented, reducing the learning rate every 10 epochs (learning_rate *= 0.9). In the training loop, forward pass, loss calculation, backward pass, and optimization were performed and the training accuracy and validation accuracy calculated for each epoch along with the average loss (**Figures in Appendix: RNN**). To control for dropout behavior, the model was switched between training and evaluation modes. ***LSTM:*** A PyTorch LSTM model was built with 12-class classification, 100 hidden units, 30 epochs, learning rate of 0.05, batch size of 300, and lambda regularization term= 0.001. Dropout layers in between hidden and all fully connected layers were used to mitigate overfitting. Softmax activation function and cross entropy loss function were used. Training and validation accuracies per epoch were calculated (**Figure in Appendix: LSTM**).

Findings:

Both models suffered from poor accuracy and had quite significant model instability. However, after testing completion, the RNN model performed consistently at an accuracy of 12.55% on the training data and an average of 9.5% on the testing data. Conversely, the LSTM model performed slightly better, with an accuracy of 13.6% on the training and 11.3% on the test. Observing the training epochs in these data, it’s clear that the RNN model begins to slightly overfit to the training data near the end and as a result, there appears to be accuracy trade offs in the validation dataset. However, the LSTM model seems to be able to afford a higher training accuracy while keeping more constant levels of accuracy in the validation dataset. Observing the average training loss for the RNN model demonstrates a gradual decrease in training loss with batch number, however oscillations in training loss does not appear to drop as batch iteration increases.

Conclusion

In conclusion, the results of our models are not ideal. While the LSTM model appears to have fitted better the discriminatory values of our models are relatively low. Given that there are 12 potential activities that can be done, our models only perform slightly better than random (8.3%). Furthermore, a great deal of model instability is present. This suggests that there are significant differences between the training and test dataset that continually appear to not be accounted for in our models. This makes sense. The data that the model was tested (n = 6) and trained (n=2) on had a limited number of individuals (n=8) thus, there are likely significant variations in how people perform specific actions or activities. Furthermore, we only decided to use movement data from the lower right arm to make these activity classifications. Thus, limiting the accuracy of our model when trying to predict leg-based movement data such as “cycling” or “jumping front and back”.

Suggestions for next steps

Future steps should investigate the use of the entire movement dataset and observe the importance of capturing leg-based data when discriminating between types of physical activities. Furthermore, more data should be captured on more individuals to improve the similarity between test and training datasets and reduce model instability. Thus allowing us to better determine model hyperparameters tune the models better.

Individual contributions: Arij Al Chawaf: Compiled the majority of the report and the presentation slides; coded for EDA and analysis. **Benjamin Zhang**: Coded for EDA, sequence creation, target class imbalance, RNN and LSTM models, testing, hyperparameter tuning, finalizing conclusion and findings of the report. **Xiao Yan**: Coded for data splitting, normalization and sequence creation.

Presentation Slides:
https://docs.google.com/presentation/d/1QiTdjd1Sgv3HuTEkL1AcDxXShl9GTsUBZ91n6xNLQcA/edit?usp=sharing

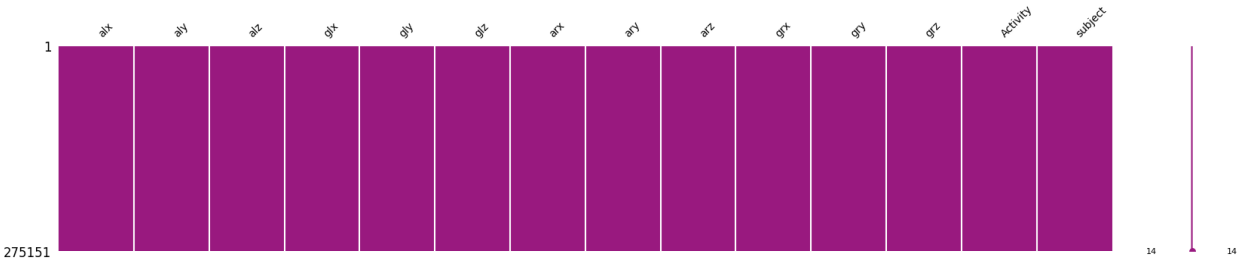
Code:
https://github.com/BennyZhangUofT/Datathon4

Appendix

Data Engineering

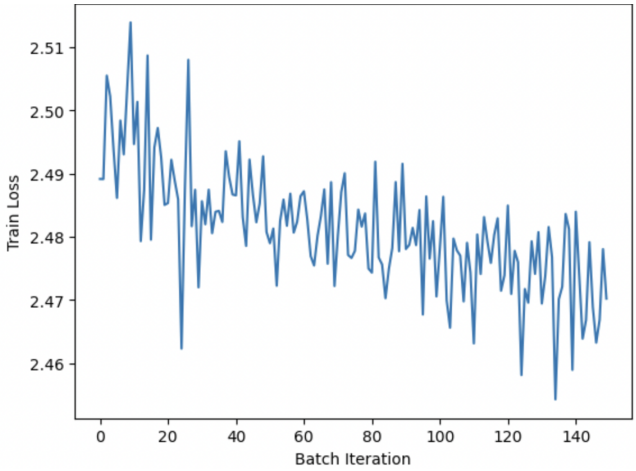
Histograms of counts for each of the 12 activities by subject.

Missing data plot

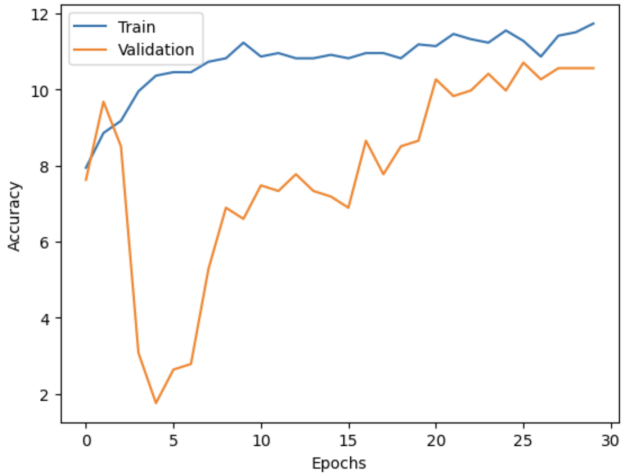


RNN

Average Train Loss per batch iteration



RNN Training and Validation Accuracy through epochs



LSTM:

LSTM Training and Validation Accuracy through epochs