# Online Freezing of Gait Detection using LSTM-FCN

Cooper Ang, James Carr-Pries, Jennifer Chen, Flynn Gurnsey, and Jeffrey Ng

***Abstract*** — Freezing of Gait (FoG) is a debilitating symptom of Parkinson's Disease (PD) that causes an episodic reduction in the ability to walk forward despite an intention. The ability to predict future FoG from time-series sensor data in real-time has the potential to prevent patients from serious injury. Various machine learning and neural networks have been specifically applied to perform this task. A new dataset consisting of various multimodal sensors with FoG labels was collected from several trials performed by patients with PD. This data was then preprocessed and prepared for time series models. Various model architectures including RNNs, LSTMs, and an LSTM-FCN were then applied to this preprocessed data. By performing hyperparameter tuning with manual methods and Optuna, the LSTM-FCN model was able to achieve an F1 score of 90.4% and a specificity score of 96.1% for predicting FoG label 0.5-seconds in advance. The results from this model were compared to the other models and analyzed to extract insights into potential improvements for online FoG prediction.

***Keywords*** — LSTM-FCN, FoG, Time Series, Online Forecasting

## I. INTRODUCTION

This project aims to perform effective online FoG prediction using the latest deep learning techniques and models available for forecasting on time series data. Predicting these occurrences before they happen will allow for intervention if required to avoid any serious ramifications.

### Freezing of Gait (FoG)

Parkinson's Disease (PD) is a chronic neurodegenerative disorder that affects 10 million people worldwide [1]. A symptom caused by PD is Freezing of Gait (FoG) which is defined as an episodic reduction in the ability to walk forward despite having the intention to walk [2]. FoG is a debilitating motor symptom that can potentially lead to falls and a loss of independence [2]. In fact, patients with PD have reported a high prevalence of falls and fractures due to the consequences of FoG episodes [1]. FoG episodes are typically brief, lasting around 1-2 seconds, but can last as much as 10 seconds [1].

### Online vs. Offline

The purpose of this report is to use a deep learning-based approach to perform online prediction of FoG. The term "online"' refers to the prediction of FoG in real-time, as opposed to "offline" which does not occur in real-time but rather focuses on classifying the dataset in its entirety [3].

The task of offline FoG labelling has applications in furthering research and understanding what triggers FoG episodes. The use of neural networks to label the occurrence of FoG episodes based on a series of biomedical sensor inputs would enable researchers to pinpoint when a patient with PD started to experience FoG and stopped experiencing FoG. Researchers can use this data to develop hypotheses on FoG triggers, and establish methodologies to prevent or reduce the number of FoG episodes experienced.

This offline FoG paradigm is not the focus of this paper, however, because online prediction has a tremendous value proposition in practical applications. Performing online FoG prediction of probable episodes is a highly impactful area of research within the biomedical field. There are numerous positive potential biomedical implications of effective online FoG prediction, as sensory wearable devices can be designed to assist patients to proactively prevent the FoG, actively disrupt an FoG episode or lessen the impact of falls due to FoG. However, performing effective FoG prediction is difficult due to a variety of constraints such as the availability of quality labelled data, limitations on the number of sensors a patient can wear, and the reliability of the data collected in real-time from patients.

## II. BACKGROUND WORKS

### LSTM (Long Short-Term Memory)

All neural networks have "long-term memory" in the form of weights and biases. Recurrent Neural Networks (RNNs) are used for tasks involving sequences, and they add a "short-term" memory which lasts for the duration of the sequence. In theory, RNNs can remember indefinitely, but in practice, they forget quickly due to vanishing gradients [4]. Long Short-Term Memory (LSTM) is a type of RNN designed to reduce the vanishing gradient, to provide longer-lasting short-term memory [5]. A

schematic of the LSTM architecture is shown in Figure 1.

This diagram highlights several key components of the LSTM. The forget gate is important to determine which timesteps are important and which are not – can be forgotten. The cell state encodes an aggregate state of the entire history whereas the hidden state stores an encoding of a characterization of the previous time step. The tanh activation functions are meant to normalize encodings whereas the sigmoid activations are meant to amplify or diminish the input.

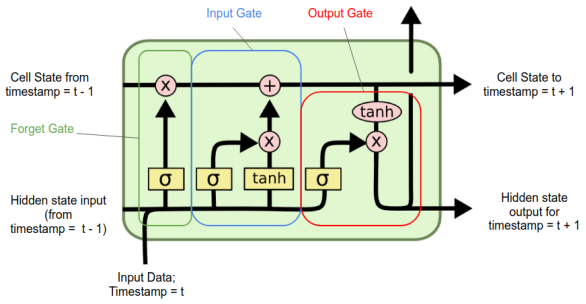


Figure 1. Schematic of LSTM cell architecture [6]

**Online FoG Baseline**

Extensive research was done on various state-of-the-art machine learning techniques on predicting an FoG event before its onset in order to gain domain knowledge on the subject matter. In order to gain a better understanding of the parameters required for FoG prediction, the paper Kleanthous. N et al. [7] was reviewed. This paper used the Daphnet FoG dataset which consists of accelerometer signals very similar to the dataset used in this project. Within this paper, techniques such as varying the size of time windows before an FoG event occurs as well as performance values were studied. Kleanthous. N et al. [7] explained that specificity and sensitivity are good ways to measure the performance of signal detection problems. Furthermore, five separate machine learning models were used to perform FoG prediction. The paper found that Support Vector Machines with Radial Basis Kernels provided the best performance with 72.34% sensitivity and 87.36% specificity - these results would inform the decisions made by the project group regarding the baseline values for model accuracies [7].

Another one of the papers reviewed was Torvi. V et al. [8] which investigated the use of various training methods using LSTM architectures for detecting FoG. This paper compares the accuracy of RNN to LSTM architectures using future prediction horizons of 1, 3, and 5 seconds. In addition, they performed a transfer learning methodology to overcome the challenge of generalizing to new patients. They first train their model using data from other subjects and then train it on data from the target subject with varying percentage ratios and with or without an additional hidden layer.

The paper describes that performing this experiment produced results showing the LSTM architecture outperforming the RNN architecture by a margin of almost 20% for each prediction horizon. Additionally, this paper also concludes that using data from the target subject drastically improves the accuracy of the model. One of the experiments performed in this paper involved pretraining the model and then using 10% of the training data from the target user to perform transfer learning. For a 1-second prediction horizon, if an LSTM architecture was used, the model would improve from a 77% prediction accuracy without using data from the target user to a 92% prediction accuracy by including data from the target user. This supports the belief that it is very difficult to generalize to a new patient's unique data. The team's main takeaways from this paper are that LSTMs outperform RNNs for FoG prediction and that including data from the target user in the training process will drastically improve the model accuracy.

**Forecasting on Time Series Data using LSTM-FCNs**

An augmented model consisting of LSTM and CNN layers was surveyed in the paper Karim. F et al. [9]. This paper proposes that Long Short Term Memory Fully Convolutional Network (LSTM-FCN) models can achieve state-of-the-art performance for time series data forecasting. The benefit of using LSTM-FCN, as described in the paper, is that it is able to extract features in a convolutional manner without requiring extensive data preprocessing or feature engineering due to the LSTM augmentation [9]. Therefore, the group considered using an LSTM-FCN for this project due to the lack of available preprocessing that is feasible to perform more effective FoG prediction over the baseline LSTM model.

## III. DATA

**Database**

The dataset selected to train the models is described in Zhang. W et al. [10]. This dataset, created in November 2022, is a multivariate time

series dataset. It is composed of data collected from 12 participants with sensors attached to them to record multimodal sensory data including electroencephalogram (EEG), electromyogram (EMG), gait acceleration (ACC), and skin conductance (SC) [10]. These 12 patients with PD were asked to perform several tasks within two different categories of tasks designed to trigger FoG while the sensor data was collected and recorded. Each patient performed each of these tasks twice for a total of four tasks per patient. However, the data was not perfectly represented within this format, so extensive data cleaning had to be performed. Altogether, 3 hours and 42 minutes of valid data were produced, including 2 hours and 14 minutes of normal gait (non-FoG) and 1 hour and 28 minutes of FoG data [10]. This dataset is biased in favour of non-FoG events and thus we have accounted for this in the use of our performance metrics and the models' loss.

**Data Cleaning**

The provided input data was first formatted in a standardized way. The number of tasks completed by each patient varied and data was stored inconsistently between them. To facilitate data manipulation, signal data was collected across all patients and all tasks, and combined into a single dataframe with 31 columns. The patients' ID and corresponding task ID were recorded, in addition to the data timestamp, EMG, ECG, Electrooculogram signals, ACC data, SC and FoG label. Due to the constraints of online FoG prediction, EEG signals were not used in the classification task due to the impracticality of obtaining EEG signals from patients in an unobtrusive manner, and were thus filtered out of the input data. To simplify notation, the two separate experiments completed by patient 8 were treated as data inputs from two separate patients (patient 8 and patient 13).

**Data Processing**

The cleaned data was then further processed to aid in the task of online classification. Since the data consisted of 1D multimodal data with values of varying magnitudes, it was decided that normalizing the data by column could possibly improve model performance. Therefore, the option of adding normalization to the data was implemented and set as a parameter that could be tuned using Optuna.

Non-overlapping sliding windows with length of $\ell$ seconds were created from the dataset. Each sliding window had a prediction horizon of 0.5 seconds. This prediction horizon was chosen as a feasible warning time to intervene when the FoG event occurs. The input data was collected using a frequency of 500Hz, so the $\ell$-seconds sliding window length would form a sequence of ($\ell \cdot 500$) input data vectors. This sequence was then used as the input for the model.

Before adding the data into the model, the non-overlapping windows of data were shuffled to make the data more uniformly distributed between patients and tasks, depending on the amount of data produced by each patient. As previously addressed, generalizing to a completely unseen patient can result in very low performance. Thus we have determined to use training and test data from all patients. Furthermore, in order to properly validate and test the model, a data split of 80% training, 10% validation, and 10% test data was used on the model. Although this is a relatively high portion of data reserved for training, we noticed that a severe limitation of our model's performance was the lack of training data and thus we chose to provide as much training data as possible while preserving the integrity of our testing and validation metrics.

## IV. MODEL

**Baseline Model**

To provide a reasonable benchmark to evaluate our final model results against, a Long Short-Term Memory (LSTM) network was implemented based on existing work done in the paper Torvi V. et al. for the purposes of FoG detection [8][11]. This benchmark model had a single LSTM layer with 50 cell units and a Dense layer with softmax activation and categorical cross-entropy loss. Although the model described in Torvi V. et al. [8] used an LSTM model with 2 layers, this was found to be ineffective, possibly due to differences in the data, so the number of layers of the benchmark model was decreased to 1. An Adam optimizer provided by Keras was used with a learning rate of 0.01, decay of 0.01, 50 training epochs, beta 1 value of 0.9 and beta 2 value of 0.999. To provide comparable results, the input data was preprocessed in the same manner as with the final proposed model. The train, validation and test set splitting was also consistent between the models. The training and validation F1 scores are shown in Figure 2. The test scores for this baseline model are shown in Table 1.

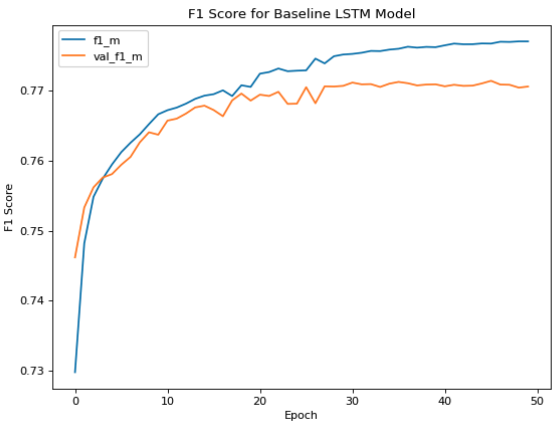


Figure 2: F1 Score of the Baseline LSTM model

**Base Model – LSTM**

In order to perform deeper analysis of the improvements of the LSTM-FCN, a basic LSTM model was created and applied to the FoG dataset. Each task is taken as an input sequence, and the elements are fed into the LSTM one at a time. Sequence lengths vary for each task and patient, ranging from 9,501 to 294,501. There is no fixed window size, so the memory is maintained for the duration of the entire sequence. The memory size is fixed however, so the network must be selective with what it chooses to remember. The architectural hyperparameters are depth (number of layers) and width (size of layers and size of memory). After trying several results, a depth of 1 and a width of 100 were found to produce good results. After the LSTM layers, there is a regular fully-connected layer of dimension (1,100) with sigmoid activation to reduce output to a scalar (probability of FoG). Binary cross-entropy loss was used, with Adam as the optimizer. The model was trained on tasks which contained at least one instance of FoG, for the patients from 1-9, and evaluated on patients 10-13.

At first, the LSTM consistently converged to always predicting 0 (indicating no FoG), due to class imbalance in the training data. This was remedied by weighting the loss based on each sample's class size. After adjusting for class imbalance, and training for 100 epochs, the model achieved a training accuracy of 83.6% and a test accuracy of 55.9%, while predicting that FoG was present 39.4% of the time. If a dummy model were created to predict FoG 39.4% of the time at random, it would achieve an accuracy of only 49.26%. Therefore, the basic LSTM achieves an accuracy that is 6 percentage points above the comparable dummy model. This shows that the basic LSTM is able to learn, but there remains much room for improvement. When applied using the same data processing steps as the other models, the training F1 score is 0.7586, while the test F1 score is 0.4416. This test F1 score is better than a dummy model with the same output distribution (predicting FoG exists 39.4% of the time at random), but worse than a dummy which predicts FoG is present 100% of the time.

**LSTM-FCN**

The nature of the data is multivariate; various signals are provided at each timestep. Based on research into similar problems and the nature of the task, it was decided that an LSTM-FCN model could yield promising results for online FoG detection [9]. A basic model was created, with various hyperparameters that were to be further tuned to achieve more optimal results.

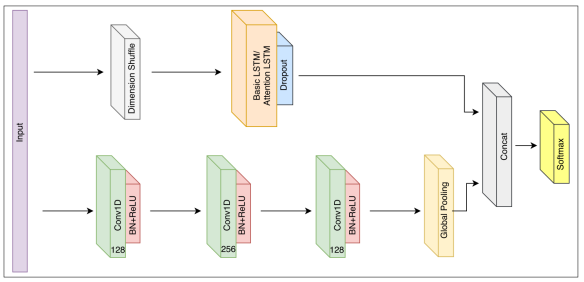


Figure 3: The LSTM-FCN Architecture (Sourced from [9])

The LSTM-FCN architecture proposed by Karim et. al. [9] consists of two branches, one using an LSTM block and the other consisting of a fully convolutional block (FCN). The FCN block contains temporal convolution layers which are used as feature extractors. Their respective filter sizes are 128, 256, and 128 with global average pooling on the last layer. For the LSTM block, the input features are first transformed by a dimension shuffle layer, which transposes the temporal dimension of the time series. Then the features enter a standard LSTM layer followed by dropout. The outputs from both branches are then concatenated together and enter a softmax classification layer for the final output. A summarized architecture is shown in Figure 3.

**Optimization and Tuning**

Once the model obtained reasonable results against the test set, attempts were made to optimize the performance. L2 regularization as well as dropout layers were incorporated and compared with base model performance. The hyperparameter optimization software Optuna [12] was also utilized to aid in tuning. Some

hyperparameters of interest that were tuned include learning rate, weight decay, batch size, number of epochs, and sliding window parameters used for data preprocessing such as the window size. The results of these tests are reported in Section V.

**Evaluation Metrics and Loss**

Due to the biased nature of our data - there are more non-FoG labels than there are FoG labels - we de-biased our loss function and the evaluation metric that was used to compare the performances of the models.

For our loss function, we chose to use the Binary Cross Entropy With Logits Loss. This loss combines the use of a sigmoid layer and the Binary Cross Entropy loss to exploit the log-sum-exp trick to improve the loss' numerical stability. To address the bias in the data, we added a weighting to the positive examples (FoG). This weight, $p_c$, was set to 1.5 since there were approximately 50% more negative samples than positive samples in the data. In the equation below $\ell(x,y) = \text{mean}(L)$, n is the number of samples, c is the class (c = 1 since it is a single label binary classification), N is the batch size and w is the weight of the loss which was set to 1.

$$\ell_c(x,y) = L_c = \{l_{1,c}, \dots, l_{N,c}\}^\top, \quad l_{n,c} = -w_{n,c}\left[p_c y_{n,c} \cdot \log \sigma(x_{n,c}) + (1 - y_{n,c}) \cdot \log(1 - \sigma(x_{n,c}))\right]$$

For our evaluation metric, we mainly compared the test F1 score of the models. The F1 score is a weighted average of precision and recall.

$$F_1 = 2\frac{\text{precision} * \text{recall}}{(\text{precision}) + \text{recall}}$$

It is commonly used as a metric for performance evaluations in tasks where the data is unbalanced. This is because it is better to look at the precision and recall - instead of accuracy - when the weights of false positives and false negatives are different. To interpret this more clearly, we can look at an extreme case where 90% of the entries are negative. If a dummy model were to predict all of the values to be negative, it would have a decent accuracy of 90%. However, the precision would be 0.1 and the recall would be 1. Thus the F1 score, of 0.18, is a more honest representation of the model's ability to learn.

We also use specificity and sensitivity performance metrics since these are standard metrics for a signal detection task like this one. Specificity is a measurement of true negatives divided by all negative samples. Specificity answers the following question: "Of all the events that are non-FoG, how many of those did the model correctly predict?". Sensitivity is the same as recall and is a measurement of true positives divided by the total number of positive samples. Sensitivity answers the following question: "Of all the events that are FoG, how many of those did the model correctly predict?".

For our use case, a higher specificity is desirable as this implies a more liberal signal detection system that is better at not missing the FoG events.

## V. RESULTS AND ANALYSIS

**Results of Hyperparameter Tuning**

Optuna is a hyperparameter optimization framework that allows users to efficiently search for the best hyperparameter values for their machine learning models. This is important because the performance of a machine learning model can be heavily influenced by the values of its hyperparameters and finding the optimal values can be a time-consuming and tedious process.

An Optuna study was conducted to search for the best hyperparameters to maximize the test set F1 score. Training for all models was conducted on the same randomly seeded training and validation set, and evaluated on the same randomly seeded test set. The Adam optimizer was used across all trials, and the model architecture did not change. For each trial of the study, the TPE (Tree-structured Parzen Estimator) algorithm [13] was used to sample a set of hyperparameters. Common hyperparameters of study included learning rate, number of epochs, weight decay, and batch; these were all evaluated in suitable ranges. Specific to the study of FoG detection, we implemented a hyperparameter search for the sliding window length, and what we call pred_val, which corresponds to how far into the future the model should try to predict the single label.

The study was conducted for 10 hours on a single Nvidia GeForce RTX 3060 Ti and resulted in 105 model candidates being trained and evaluated. The best candidate model, seen as "LSTM-FCN Tuned", is compared to the untuned variant in Table 2 and Figure 5.

Table 1: Comparison of Optuna Hyperparameters

| Hyperparameter | Untuned Value | Tuned Value |
|---|---|---|
| Learning Rate | $1 \times 10^{-3}$ | $5.6 \times 10^{-4}$ |
| Number of Epochs | 50 | 62 |
| Window Length | 500 | 11 |
| Batch Size | 32 | 64 |
| Normalization | True | True |
| *pred_val* | 250 | 250 |
| Weight Decay | None | $2.5 \times 10^{-4}$ |

Analysis of the hyperparameters was performed automatically by Optuna, which uses the trials to track the sensitivity of each parameter towards the objective, which in this case, is the test set F1 score. A summary of the importance of the searched hyperparameters can be seen in Figure 4. Interestingly the most important hyperparameter was window length, with a clear trend that lower window lengths around 10 samples (20 ms) performed the best. We hypothesize that lower window length could be attributed to creating more samples for the model to train on, thus scaling performance. As expected, learning rate and number of epochs had significant importance to allow the model to reach the lowest loss possible. Normalization of the features, pred_val, and weight decay had little importance to model performance.

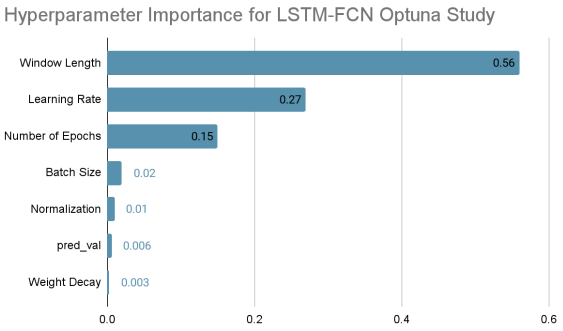


Figure 4: Hyperparameter Importance for LSTM-FCN Optuna Study

Comparing the loss curves of the untuned and tuned model, it is clear that the model was able to train for more epochs without overfitting. As a result, the model reached a lower validation loss, and testing on the hold-out test set resulted in the highest F1 score of 90.4% seen in Table 2. Model training time was 38 minutes on a single Nvidia GeForce RTX 3060 Ti.

**Comparison with Other Works**

For our experimentations we used three different models. The current state-of-the-art model that we compared against as a baseline is experimented with in Torvi. V et al. [8]. We adapted this model to work with our dataset and optimized the architecture to perform well with our dataset. We also compare our model

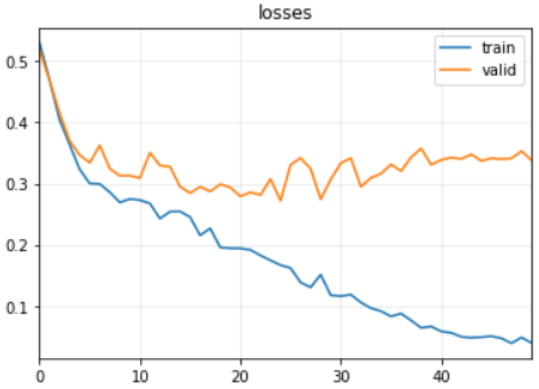


Figure 5: Loss vs Epoch graph for untuned LSTM-FCN

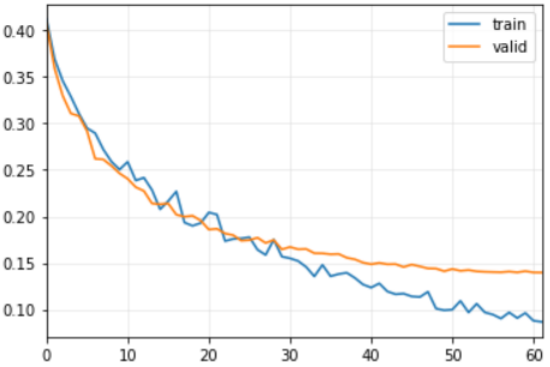


Figure 6: Loss vs Epoch graph tuned LSTM-FCN

against the single-layer LSTM. This is more to discuss the improvements our model has over a very basic implementation of an LSTM. Finally, we compare the untuned and tuned versions of our model. We report the F1 score, specificity and sensitivity for each of these methods on the test dataset in Table 2.

Table 2: Comparison of Model Performance

| | | Test F1 | Specificity | Sensitivity |
|---|---|---|---|---|
| (Torvi, 2018) | | 76.0% | 76.0% | 76.0% |
| LSTM | | 44.16% | 59.52% | 50.82% |
| LSTM -FCN | Untuned | 85.2% | 91.4% | 90.1% |
| | Tuned | **90.4%** | **96.1%** | **90.4%** |

Table 2 shows that the tuned LSTM-FCN performed the best out of all four models. Additionally, both LSTM-FCN models outperformed the state-of-the-art for this task in terms of F1 score, specificity and sensitivity.

Figure 7 shows the confusion matrix for the tuned LSTM-FCN. From this, it is clear that the model incorrectly classifies many of the examples. However, it provides a very balanced estimate of false positives and false negatives which is reassuring given the biased dataset. This implies that the model is accounting for this bias and has learned to recognize FoG events equally despite their lack of representation.

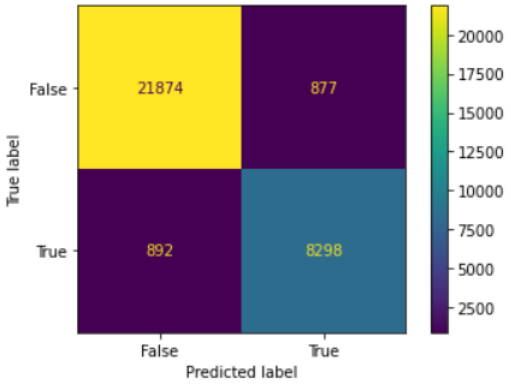


Figure 7: Confusion matrix for the tuned LSTM-FCN on the test dataset

## VI. DISCUSSION

Our new model that uses LSTM-FCNs clearly outperforms the baseline model. Additionally, since it only requires a window size of 11 samples and can predict 0.5 seconds into the future with reasonable performance, this model will be suitable for an online application of FoG detection.

**Interpretability**

With any model, but particularly with models being applied in the field of Biomedical Engineering, it is important that the models can be interpretable. Machine learning models have been notorious for being a black box that is uninterpretable [14]. When the stakes are human safety, it is important for the practitioner to understand what the model is doing proactively and to justifiably remove liability.

Figure 8 shows the feature importances for each of the features in the model. The permutation feature importance is defined as the increase in the model's loss when a single feature is randomly corrupted through a random permutation.

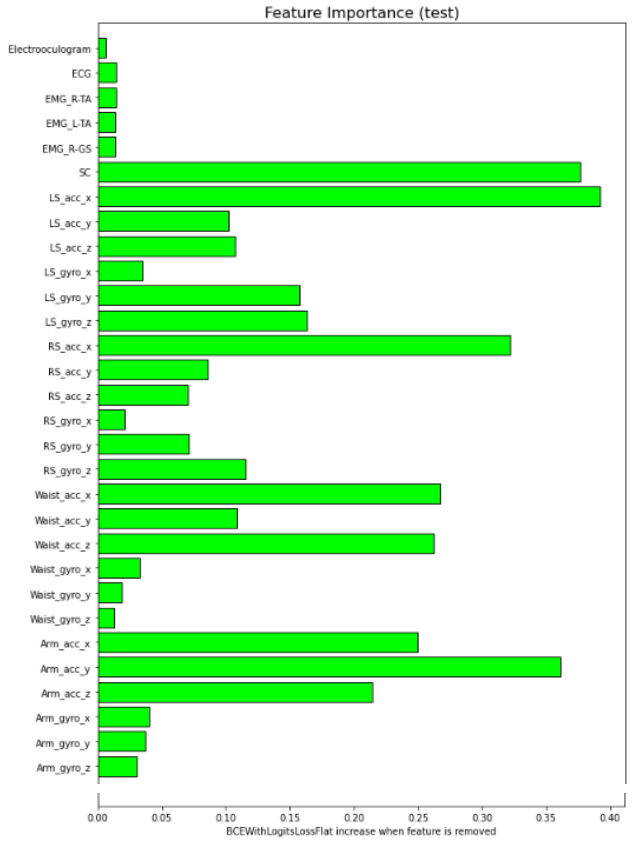


Figure 8: Feature Importances of tuned model

It is important to recognize that the model is using what we expected would be useful features to predict FoG with. It uses many of the acceleration metrics that can be used to determine if the patient is coming to a stop (start of a freezing episode) or to determine how their gait may be changing. It also, interestingly, places high importance on SC. SC has been shown to be correlated to emotion [15]. Changes in SC have been used in several FoG applications as a patient may show signs of stress or fear if they are freezing [16]. Since the model can make an accurate prediction based on SC and arm or leg acceleration, a smartwatch or anklet with our model could be a very useful device for FoG detection.

Another aspect of interpretability is determining which time steps within the window are most important to the model's prediction. Figure 9 shows the step importances for each step of the tuned model (window size 11 steps) and every 50 steps of the untuned model (window size 500 steps). The permutation step importance is defined as the increase in the model's loss when a single step is randomly corrupted.

It is interesting to note that for the untuned model with a longer time window, it relies more heavily on timesteps earlier in the window. More investigation should be done into why this occurs.

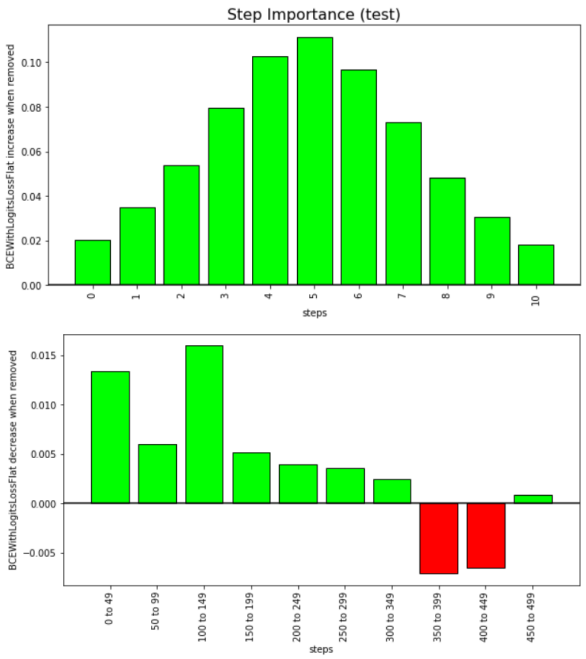


Figure 9: Step importance of the tuned (top) and untuned (bottom) LSTM-FCN

One hypothesis for this behaviour is this could be a result of a vanishing gradient within the LSTM for longer windows. The importance that the model places on earlier time steps helps to explain why a shorter time window such as 11 steps was still successful in predicting a correct FoG label.

**Limitations**

The major limitation of our experiments is the application of the model to new patients. When we originally tested our model we noticed that it was very difficult to generalize to new patients. This is likely due to the variability in each of the biomedical signals between patients. Even with some data preprocessing and normalization, we were unable to generalize. Future work should be done with respect to the work done by Torvi et al. [8]. There is evidence which suggests that the deeper layers of the network may contain more genetic features and thus the deepest layers of the model may only need to gain reasonable performance on new patients with minimal data - few-shot transfer learning or personalization. However, this work was not very successful at working with completely new patients and thus more work needs to be done for our model to generalize better to out-of-distribution patients.

Another limitation is inherent to our problem space. Online FoG prediction is much more challenging than the offline classification counterpart. A classification task has access to all of the data at once so it can use data before, during and after the FoG detection. The prediction task on the other hand must anticipate these events without knowing if they will occur. Online FoG prediction is also limited by the constraints of deployable hardware and their respective memory and compute capabilities, thus limiting the choice of the model architecture and latency.

## VII. CONCLUSION

The final implemented LSTM-FCN model was able to predict the occurrence of FoG in a patient in the next 0.5-second horizon with an F1 score of 0.9. If this model for online detection of FoG were to be paired with a wearable device, patients can be monitored throughout the day, and they could be alerted prior to an FoG episode when FoG is detected to occur in the next 0.5-second. Precautions or interventions can thus be taken to ensure the patient is safe or won't freeze to reduce fall risk. Sensory or visual cues could be used to help the patient out of their FoG episode [17]. The model will need to be further tuned and tested if it were to be used in a clinical application. Potential future optimizations that can be considered include the normalization of data for each patient to improve prediction accuracy, preprocessing of biomedical signals inputted to the model to improve their relevance in FoG detection and transfer learning techniques to generalize to new patients.

REFERENCES


[1] C. Azevedo Coste, B. Sijobert, R. Pissard-Gibollet, M. Pasquier, B. Espiau, and C. Geny, “Detection of freezing of gait in parkinson disease: Preliminary results,” *MDPI*, 15-Apr-2014. [Online]. Available: https://www.mdpi.com/1424-8220/14/4/6819.

[2] E. Heremans, A. Nieuwboer, and S. Vercruysse, “Freezing of gait in parkinson's disease: Where are we now?,” *Current neurology and neuroscience reports*. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/23625316/

[3] S. Mazilu et al., "Online detection of freezing of gait with smartphones and machine learning techniques," 2012 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops, 2012, pp. 123-130, doi: 10.4108/icst.pervasivehealth.2012.248680.

[4] S. H. F. für Informatik, S. Hochreiter, F. für Informatik, J. S. IDSIA, J. Schmidhuber, Idsia, and O. M. V. A. Metrics, “Long short-term memory,” *Neural Computation*, 15-Nov-1997. [Online]. Available: https://dl.acm.org/doi/10.1162/neco.1997.9.8.1735.

[5] S. Hochreiter, V. Profile, U. Paris, and O. M. V. A. Metrics, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 01-Apr-1998. [Online]. Available: https://dl.acm.org/doi/10.1142/S0218488598000094.

[6] C. Olah, “Understanding LSTM networks,” *Understanding LSTM Networks -- colah's blog*, 27-Aug-2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 06-Dec-2022].

[7] N. Kleanthous, A. J. Hussain, W. Khan, and P. Liatsis, “A new machine learning based approach to predict Freezing of Gait,” *ScienceDirect*, Dec-2020. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167865520303524.

[8] V. G. Torvi, A. Bhattacharya, and S. Chakraborty, “Deep domain adaptation to predict freezing of gait in patients with parkinson's disease,” *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018.

[9] F. Karim, S. Majumdar, H. Darabi, and S. Chen, “LSTM fully convolutional networks for Time Series Classification,” *arXiv.org*, 08-Sep-2017. [Online]. Available: https://arxiv.org/abs/1709.05206.

[10] Zhang, W., Yang, Z., Li, H. et al. Multimodal Data for the Detection of Freezing of Gait in Parkinson’s Disease. Sci Data 9, 606 (2022). https://doi.org/10.1038/s41597-022-01713-8

[11] alexyuan66, “Alexyuan66/Deep_Recurrent_Neural_Networks_FoG,” *GitHub*. [Online]. Available: https://github.com/alexyuan66/Deep_Recurrent_Neural_Networks_FoG.

[12] “A hyperparameter optimization framework,” *Optuna*. [Online]. Available: https://optuna.org/.

[13] J. Bergstra, Y. Bengio, R. Bardenet, and B. Kegl, “Algorithms for Hyper-Parameter Optimization.” [Online]. Available: https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[14] T. P. Quinn, S. Coghlan, V. Le, M. Senadeera, and S. Jacobs, “The Three Ghosts of Medical AI: Can The Black-Box Present Deliver?” [Online]. Available: https://arxiv.org/pdf/2012.06000.pdf.

[15] H. Miwa, S.-ichiro Sasahara, and T. Matsui, “New Mental Health index based on physiological signals at transition between arousal and sleeping state,” *2007 6th International Special Topic Conference on Information Technology Applications in Biomedicine*, 2007.

[16] F. Gravenhorst, A. Muaremi, A. Gruenerbl, B. Arnrich, and G. Troester, “Towards a mobile galvanic skin response measurement

system for mentally disordered patients," *Proceedings of the 8th International Conference on Body Area Networks*, 2013.

[17] R. Velik, U. Hoffmann, H. Zabaleta, J. F. Marti Masso, and T. Keller, "The effect of visual cues on the number and duration of freezing episodes in parkinson's patients," *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2012.