FALL DETECTION IN ELDERLY USING GAIT ANALYSIS

Richa Pandya
Student# 1009959779
richa.pandya@mail.utoronto.ca

Anya Pedersen
Student# 1007637504
anya.pedersen@mail.utoronto.ca

Krish Chhajer
Student# 1005678901
krish.chhajer@mail.utoronto.ca

Nicholas Eckhert
Student# 1010037568
nick.eckhert@mail.utoronto.ca

—-Total Pages: 9

1 PROJECT DESCRIPTION

Falls are the leading cause of injury death in adults aged 65 and over, causing about 38,000 deaths and almost 3 million hospital visits in 2021 (for Disease Control & Prevention, 2024). Through the synthesis of current deep learning techniques and gait (walking patterns) based data from gyroscopes and accelerometers (located in smartphones or other portable technologies), our team proposes a new and viable solution to predicting and preventing elderly falls. Mobile devices can make this data easily accessible- however it is often complex, containing time series data and multiple sensors. Neural networks can easily extract and recognize patterns, making it a good tool for analyzing the complex sensor data and returning an accurate reading. Additionally, a neural network's ability to learn over time allows it to scale well with large amounts of data that other models may struggle with.

The team proposed using a hybrid neural network architecture of a CNN and LSTM that would receive input from sensor readings (off of the user's phone) and output information that can be used as data for future research as well as issue a notification to nearby emergency responders. The use of a neural network in this way should significantly alleviate the pressure on both the patient and the emergency responders and help fall victims to receive immediate and effective care.

This report outlines the current status of this project and showcases early results captured through implementing a baseline model and minimum viable neural network.

2 INDIVIDUAL CONTRIBUTION AND RESPONSIBILITIES

2.1 PROJECT MANAGEMENT SOFTWARE

For project progress tracking we are using SmartSheet. This allows everyone to be able to view and edit their assigned tasks and scheduling, while viewing the project in an Gantt Chart format. To communicate progress and set new goals, the team holds weekly meetings on Mondays, where minutes are recorded and posted onto a shared drive for all members to reference. For versions of code that are preliminary and in progress, Google Colab is used and edits are made on different versions between team members. Final versions of the code used are uploaded to a Github team repository.

The individual work done as well as project responsibilities assigned at the beginning of the project is presented in table 1. Project responsibilities are also defined, as specified at the beginning of the project. The task each member is responsible for includes planning timelines and dividing up the work for that particular task. This allows every member of the team to take on a leadership role, and for the group to focus on the critical part of the project all together. The deadlines for the designated tasks and the associated follow up tasks to completing the project are presented in figure 1. The immediately assigned tasks are the ones with team members specified.

Table 1: Individual Contributions and Responsibilites

MEMBER	RESPONSIBILITIES	TASKS ACCOMPLISHED/NEXT TASK
Krish	Data Collection and Testing	Acquired dataset and dataset alternatives Planned and executed sampling method and data cleaning **Add to dataset processing to incorporate labels of more classes for further activity detection**
Richa	Model Integration	Planned and designed Neural Network Structure Developed code to Format Data Input to Model Wrote CNN and LSTM Layers **Design Alternative model architectures for expanded class detection**
Nick	Baseline Model	Wrote function to create baseline model Wrote supporting functions for data processing **Test alternative phone sensors for new data acquisition**
Anya	Written Assignments	Tracking Team progress via SmartSheet and Meeting Minutes Wrote supporting functions for data processing Wrote Fully Connected Layers and Output Functions for Model **Design hyperparameter test plan for current model**

Task Name	Start	Finish	Pr...	Assigned To
Milestone 3: Final Report	07/04/24	08/02/24	23	
Improve Neural Network	07/04/24	07/29/24		
Model 1 Hyperparameter Selection	07/04/24	07/12/24		Anya Pedersen
Alternative Architectures	07/04/24	07/19/24		
Multi-Class Detection	07/04/24	07/19/24		richa.pandya@mail.utoronto.ca
Model 2 Hyperparameter Selection	07/22/24	07/29/24	68	
Introduce New Data	07/04/24	07/29/24		
New data format	07/04/24	07/29/24		
Create Multi-Class Labelled Data	07/04/24	07/19/24		krish.chhajer@mail.utoronto.ca
Designation Data Split	07/22/24	07/29/24	73	
Select / Collect new data	07/04/24	07/29/24		
Test + Select Acquisition Sensor	07/04/24	07/19/24		nick.eckhert@mail.utoronto.ca
Collected Data Testing Plan	07/22/24	07/29/24	76	
Presentation Creation + Report Planning	07/30/24	08/02/24	66	

Figure 1: Upcoming Project Tasks

2.2 Current Risks

A current risk we are facing is the feasibility of collecting our data. This would prevent us from obtaining an unbiased accuracy estimation on our model, therefore not being able to complete our project. This is due to phone sensors inherently outputting specific ranges based on the product. An alternative to this is to purchase a sensor of higher quality, which would take time and money to acquire. To mitigate this risk, we have designated it as a task to one of the teammates to do an exploratory investigation on what sensor may give us acceptable data to test with, without incurring any cost to the team (further discussed in Gathering New Test Data). To further reduce this risk, we also have accumulated alternative datasets, with data currently unseen in the training of our model .

2.3 Team Member Support

To ensure each team member is supported in their work such that the due dates are achievable, the weekly meetings provide an opportunity for the group to reflect on the progress of the project

(progress can be viewed in the SmartSheet Gantt Chart). This will allow adjustment of tasks to the critical next tasks that are at risk of not meeting internal deadlines.

3 Data Processing

Due to lack of sufficient Fall samples in the MobiFall Dataset BMI, the model has been trained on the SisFall Dataset (Angela Sucerquia & Vargas-Bonilla, 2017). Similar to the MobiFall Dataset (BMI), the SisFall Dataset (Angela Sucerquia & Vargas-Bonilla, 2017) contains accelerometer and gyroscope sensor data of 19 different types of ADL and 15 different types of Fall, performed by young and old adults. The data has been recorded using the ADXL345 and MMA8451Q accelerometers along with ITG3200 gyroscope. It has been sampled at 200 Hz and is stored in the form of text files containing the x, y and z directional components of all sensors. The procedure given below has been utilized to clean, process and extract all training and testing data.

3.1 Data Cleaning and Extraction from Text Files

Gait_Analysis_Preprocessing.ipynb contains a Python script. It reads through all the text files stored in the drive and uses string manipulation to extract the sensor readings. It converts the sensor readings from bits to the standard units and stores it in individual csv files in a "Preprocessed Data" folder in Drive. The naming convention for each text file, which contains information of the activity, subject and trail no, have been conserved in the csv files.

	Timestamp	X_Acc_1	Y_Acc_1	Z_Acc_1	X_Gyro	Y_Gyro	Z_Gyro	X_Acc_2	Y_Acc_2	Z_Acc_2
0	0.000	-0.344883	-9.848320	-0.958008	5.126953	15.075684	1.647949	-1.149609	-9.455537	0.603545
1	0.005	-0.114961	-10.078242	-0.881367	6.042480	15.747070	2.136230	-1.053809	-9.733359	0.651445
2	0.010	-0.038320	-10.346484	-0.843047	6.958008	16.601562	2.746582	-0.900527	-9.934541	0.661025
3	0.015	0.038320	-10.614727	-0.919687	7.751465	17.456055	3.479004	-0.775986	-10.174043	0.661025
4	0.020	0.076641	-10.768008	-0.958008	8.178711	18.493652	4.272461	-0.680186	-10.336904	0.603545
...	...	...	...	...	...	...	...	...	...	...
2995	14.975	-4.521797	2.567461	-9.618398	-2.868652	0.976562	0.000000	-5.230723	2.826123	-7.922725
2996	14.980	-4.521797	2.644102	-9.465117	-2.746582	0.976562	0.000000	-5.259463	2.797383	-7.874824
2997	14.985	-4.521797	2.490820	-9.541758	-2.807617	0.976562	0.000000	-5.201982	2.797383	-7.922725
2998	14.990	-4.483477	2.605781	-9.656719	-2.868652	1.098633	0.183105	-5.259463	2.835703	-7.884404
2999	14.995	-4.291875	2.529141	-9.426797	-2.929688	1.098633	0.244141	-5.182822	2.787803	-7.884404

3000 rows × 10 columns

Figure 2: Sample Fall CSV

3.2 Window Sampling

A 15 second data sample has been window sampled into 14 windows such that the window sizes are 2 seconds (400 samples @ 200 Hz) with a 1 second overlap (200 samples @ 200 Hz). This has been chosen carefully based on previous work such that the windows can capture temporal dependencies easily. Since different activities have different time length, we've employed the following techniques to adjust the time lengths before window sampling.

- **Fall Activities:** Fall samples have not been processed in any way. 15 seconds has been chosen as the benchmark time length for all samples. Certain Fall activity samples have one reading less (2999 instead of 3000 reading) so we have copied the last reading again at the end to keep the sample size consistent. This does not affect the data as most falls occur in the first 10 seconds and the last 5 seconds are mostly irrelevant data (check Data Visualisation_v1.ipynb).
- **ADLs (1-4):** These are activities such as Jogging and Walking. Since these are continuous activities (look at EDA analysis), we can easily separate these activities of time lengths of 100 seconds into five 15 second samples and process them separately. This also helps to

increase the number of such samples as each subject only does one trial for these activities (check Data Visualisation_v1.ipynb).

- **ADLs (5-6):** These activities are 25 seconds in length and contain activities going up and down the stairs. To capture movement going up and down the stairs separately, we have sampled the data such that the first 15 seconds capture one movement and the last 15 seconds capture the other movement. Thus, the 25 seconds are thus split into two such 15 second samples. Since the separation between the two movements occurs at the half mark for most samples, we don't change the data class in any way and moreover introduce irrelevant noise to add to the model's robustness (check Data Visualisation_v1.ipynb) .
- **ADLs (7-19):** These are other ADL activities of 12 seconds in length. For such activities, most of the important activity capturing data occurs at the first 10 seconds and the last few seconds only contain irrelevant data where the person is still/doing no activity. Thus, we increase the length of these samples to 15 seconds and use forward fill to fill in the NaN values in the end. This way, we don't add any new data to change the class/activity type but just increase the irrelevant data samples at the end to match the 15 second time length (check Data Visualisation_v1.ipynb).

3.3 Feature Extraction

Since it is not possible to process sensor data at 200Hz, we use window sampling to down sample our data. However, to ensure we do not lose any substantial information, we extract graph, time-domain and frequency-domain features for each window sample. This ensures greater insight into our data for our model to enhance its processing power and also not requiring significant computational resources. The features we have extracted are referenced from the UCI HAR Dataset (D. Anguita, 2013): **Mean**, **Median**, **Max**, **Min**, **Standard Deviation**, **Inter-Quartile Range**, **Jerk** (Gives the average rate of change of the data in the window), **Skewness**, **Kurtosis** (Measures the taildness of the distribution of the window data), **Energy** (Gives the average absolute (squared) value of the data in the window), **Covariance**. For each of the x, y, and z components of both the accelerometer and gyroscope data, we extract the above features to give a total of 56 features for each window sample. It is to be noted that we have only window sampled and extracted features of the ADXL345 accelerometer readings because our model is supposed to work with only one set of accelerometer readings and this sensor is more energy efficient and has a wider range (Angela Sucerquia & Vargas-Bonilla, 2017).

3.4 Processing and Merging

The last step involves processing all files in the "Preprocessed Folder" through the window sampling above and saving it in the "Processed Folder" at drive. Each new file now contains 14 windows and 56 features for each of the data samples. This contains different csv files for different activities, with the naming convention preserved and the data window sampled with features extracted. It returns two variables- data (list of all data frames) and labels (containing labels for each sample with 1 for Fall and 0 for ADL) for further reshaping and normalization as needed.

	Window	X_Acc_Mean	X_Acc_Max	X_Acc_Std	X_Acc_Min	X_Acc_IQR	X_Acc_Skew	X_Acc_Kurtosis	X_Acc_Energy	X_Acc_Jerk	...	Z_Acc_Mean.1	Z_Gyro_Max
0	1	-0.011209	3.333867	1.180099	-3.640430	1.072969	-0.629072	1.373224	1.389279	-0.210762	...	2.385254	76.538086
1	2	-0.016095	2.912344	1.311070	-4.521797	1.504072	-0.796797	1.086280	1.714866	-2.126777	...	0.439453	76.538086
2	3	0.005940	3.832031	1.455470	-4.521797	1.283730	-0.396738	1.146642	2.113131	0.191602	...	-1.145477	76.782227
3	4	0.027878	5.479805	1.808183	-4.866680	1.159189	-0.184764	1.208545	3.262128	1.571133	...	0.426636	81.848145
4	5	-0.124828	5.479805	1.705863	-4.866680	1.187930	-0.206821	1.827534	2.918276	0.076641	...	-0.218506	81.848145
5	6	0.035542	10.691367	2.354291	-8.660391	1.772314	1.261404	5.582205	5.530092	-5.633086	...	0.365601	80.444336
6	7	-3.565322	44.374922	8.112413	-42.803789	6.706055	-0.532750	9.346877	78.358236	-2.912344	...	12.981110	551.940918
7	8	-6.369602	44.374922	6.893272	-42.803789	1.005908	0.122249	19.432383	87.970238	3.391348	...	12.766876	551.940918
8	9	-5.147280	-4.636758	0.249330	-5.824688	0.421523	-0.808778	-0.436576	26.556503	0.479004	...	0.683594	3.662109
9	10	-4.929333	-4.560117	0.085077	-5.173242	0.114961	0.038899	1.295277	24.305548	0.153281	...	0.239258	2.319336
10	11	-4.820600	-4.483477	0.110448	-5.096602	0.153281	0.417441	-0.014434	23.250348	0.134121	...	0.081177	3.112793
11	12	-4.645284	-4.330195	0.135750	-5.019961	0.229922	-0.093164	-0.870819	21.597046	0.134121	...	0.210876	3.112793
12	13	-4.521893	-4.176914	0.083158	-4.790039	0.076641	-0.153532	0.986279	20.454411	0.057480	...	0.058594	1.098633

Figure 3: Fall Sample after Window Sampling and Feature Extraction

For the primary model using data and labels, the data was merged into a 3D numpy array and saved as "CNN-LSTM Data.npy" while the labels was converted into a numpy array and saved as "CNN-LSTM Labels.npy". For the SVM, the data for each sample has been flattened such that the features for each window are placed one after the other. This is because the SVM can only process 2D data in the form of samples and features. The data is then stored in a Data frame and saved as a csv file "Baseline SVM Data.csv". This data is then normalized, shuffled into batches, converted into any required shape and fed into the models for training.

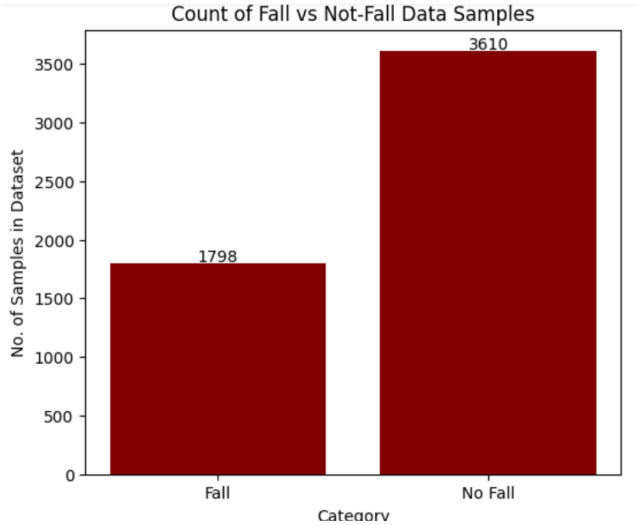


Figure 4: Final Number of Processed Samples

3.5 Gathering New Test Data

To further test the model on unseen data, the team will use the MobiFall Dataset (BMI). Similar to the SisFall Dataset (Angela Sucerquia & Vargas-Bonilla, 2017), it contains accelerometer and gyroscope readings collected from a Samsung S3 phone during Fall and ADL activities performed by adults. Similar preprocessing techniques as described above will be employed and the model's performance will be evaluated. Additionally, the team will also gather our own data using an Android mobile and the Sensor Logger app (Choi). The team will perform similar activities as the ones described in SisFall (Angela Sucerquia & Vargas-Bonilla, 2017) in a 15 second timeframe and record the sensor readings at a determined constant frequency. After that, the data will be processed in the same way and the model's performance will be evaluated.

4 Baseline Model

To establish a baseline with which to compare our model, the team has utilized a Support Vector Machine (SVM) with a linear kernel (implemented in SVM.ipynb) to classify our data. The SVM finds a hyper plane that maximizes the margin between the two nearest points of each label (in binary classification), fits the data, and then creates a decision boundary to classify based on the chosen hyperplane.

4.1 Data Reprocessing and Removing Bias

An early challenge the team faced with the baseline model was our data shape. Since the processed data was three-dimensional and SVM's only except two-dimensional data, we needed to flatten our data. The team decided to convert the time series windows and features into columns for each labeled event. This resulted in a final 5408 x 771 dataframe. After creating the new dataframe, the team removed the numbering for each sample, leaving only features in the dataframe, and shuffled

the data to remove any potential biases. Next, we normalized the data using the Standard Scaler from the Sci-Kit Learn (SKLearn) library.

4.2 Model Training and Testing

In order to fit our model to our data, we separated the reprocessed data from the fall/no-fall labels and split it into training and testing subgroups using SKLearn's split_test_train function, which divided the data into 80-percent training data and 20-percent testing. For the SVM model we chose to use SKLearn's Support Vector Classifier, and fit the model to the training data. Afterwards, we predicted the testing data's labels using our model and then scored them against the true labels. The model was able to receive an overall testing accuracy of 99.72-percent (0.28-percent error) on the remaining data by correctly classifying 718 out of 720 ADLs and 361 out of 362 falls, as seen in Figure 5.

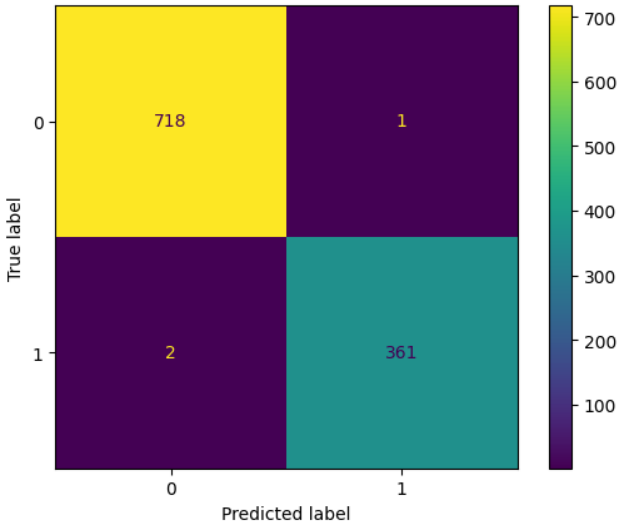


Figure 5: SVM Confusion Matrix

4.3 Visualizing the Decision Boundary

In order to visualize the decision boundary and the SVM data, the team used the dimensionality reducing technique: principal component analysis (PCA). Since PCA maintains linearity, it was used to gain a general idea of the hyperplane and its division of the data (figure 6). It can be seen that the data is separated extremely well by the SVM, resulting in the high accuracy. Note: PCA is not a perfect representation of the higher dimensional data, so the graph only gives us a general idea of the data and the SVM's decision boundary.

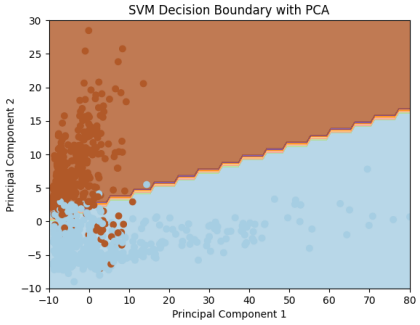


Figure 6: SVM Decision Boundary (PCA)

5 PRIMARY MODEL

The team designed the hybrid CNN-LSTM architecture to process complex time-series data from inertial sensors effectively to produce an accurate binary classification. Figure 7 details the architecture sequentially.

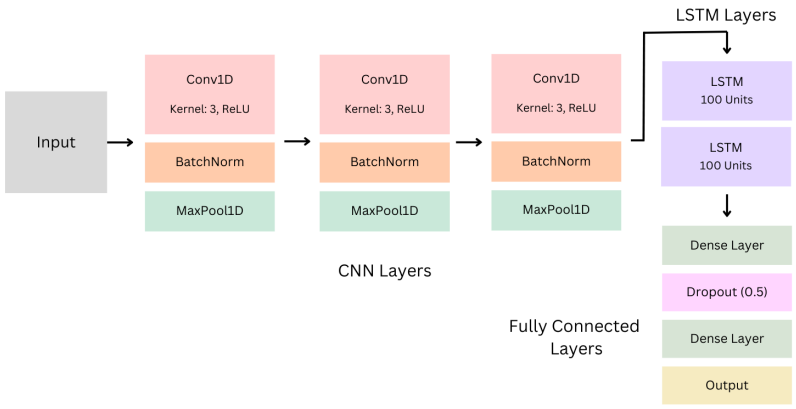


Figure 7: CNN LSTM Model Architecture

After the preprocessing process, the model accepts a 3D tensor with dimensions (sample number, time step, and feature type). This data first gets passed through three convolutional layers that each increase in complexity to detect increasingly intricate features: the first layer has 64 filters, the second layer has 128 filters, and the final layer has 256 filters. Each layer is followed by a ReLU activation function, batch normalization and max pooling (pool size of 2).

The output from the CNN then gets passed into two LSTM layers - each with 100 units. After this, the output is sent to a dense layer with 100 units and ReLU activation, a dropout layer with rate of 0.5, and the output layer is 1 unit with a sigmoid activation function to produce the accurate output.

5.1 DESIGN CHOICES AND RATIONALE

The goal of the convolutional layers is to extract spatial features from the input data. The increasing number of filters with each layer (64, 128, 256) allows the neural network to learn increasingly more complicated patterns located in the data. The batch normalization after each layer helps keep the learning process stable, while the max pooling step reduces dimensionality, computational load, and prevents overfitting.

The LSTM layers were implemented to capture long-term dependencies within the time-series data. The first LSTM layer processes the entire sequence, while the second layer summarizes the temporal information and creates a fixed-size output accordingly.

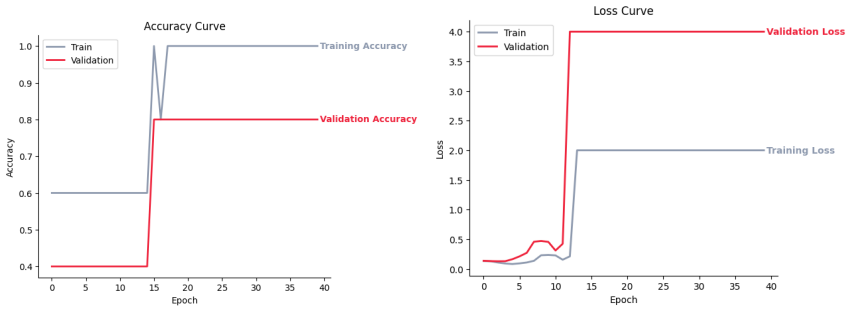
The fully connected layers interpret the high-level features extracted by the CNN and LSTM layers, and maintaining a dropout rate of 0.5 was included to prevent overfitting.The output layer is a single unit with a sigmoid activation function was used to do the final binary classification and determine whether or not a fall was detected.

With three convolutional layers, two LSTM layers, and two dense layers, this network is considered a deep, complex neural network. The increasing number of filters in the convolutional layers and the use of 100 units in the LSTM layers ensure that the model is able to learn intricate patterns in the data and effectively produce a correct prediction.

5.2 Model Performance

5.2.1 Verifying Training Function

The initial model was trained on a reduced dataset to test over fitting, or memorization of the model. It was confirmed that the model and training function are working through obtaining 100% accuracy on a reduced training dataset (5 Samples) within 200 iterations of the data (Batch size of 5, 40 Epochs). We see that the data set is memorized within 100 iterations of the data (Epoch 20). In addition to that, we can observe that the validation loss is very high and above that of training.

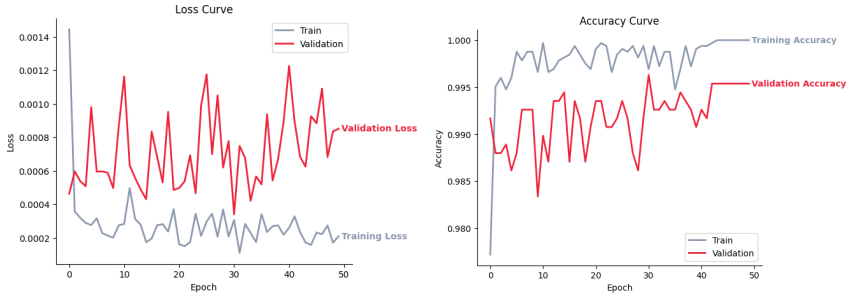


(a) Accuracy (b) Loss

Figure 8: Overfitting Small Dataset

5.2.2 Model Version 1 Performance

The data was divided to 60-percent training, 20-percent validation, and 20-percent testing. For the initial model the hyperparameters selected are specified in table 2.



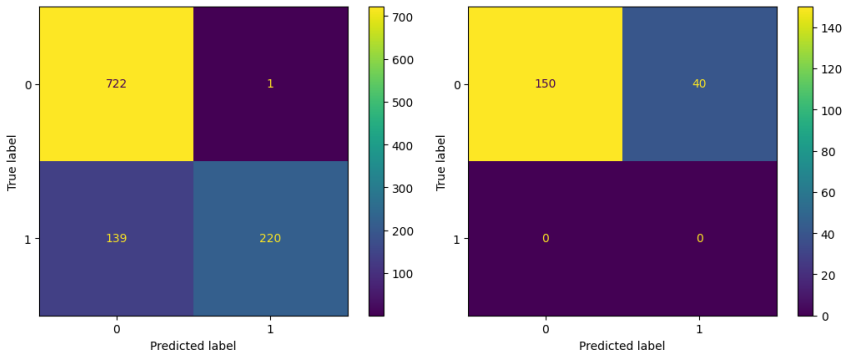
(a) Loss (b) Accuracy

Figure 9: CNN Version 1 Training

The hyperparameters are evaluated based on the progression of the loss curve and the accuracy of the model. For the first version of the model, the type 1 error count was significantly greater than the type 2 error count as displayed in the confusion matrix from the validation dataset for CNN model v1 (Test Accuracy 91%).

Table 2: Hyperparameters for Each Model Version

CNN MODEL VERSION	BATCH SIZE	LEARNING RATE	EPOCHS
Version 1	64	0.01	50



(a) Confusion Matrix (Test Dataset) (b) Confusion Matrix (D11)

Figure 10: Confusion Matrices

As the dataset is divided up into different activities, we are able to investigate what activities are able to cause what kind of failure. We have identified that a particular non-fall is the possible cause for the type 2 errors seen in our model. The identified action, coded at D11, is the action of getting up from a chair, losing balance, and sitting back down. When only looking at a test set of these samples, our model accuracy is around 73%. The corresponding confusion matrix is displayed above.

One way to circumvent this issue is to incorporate more data parameters into our input data, widening the analysis that the model can achieve on a 15 second sample. Further, more work is to be done on hyperparameter selection (Table 2).

5.3 CHALLENGES

Challenges faced primarily were around the merging of various layers. The model design has natural divisions of work between CNN layers, LSTM layers and Fully Connected Layers. Therefore when incorporating them all together, it took time not only to verify other's code, but to merge the different coding styles. Aside from that, precise planning allowed for the layers and input sizes to merge seamlessly.

6 LINKS TO GITHUB AND COLAB NOTEBOOKS

- **Gait_Analysis_Preprocessing_Setup.ipynb-**`https://colab.research.google.com/drive/1r8G4YNEqHcaZ-ZnRfnEgGmYkRrMF1YqK?usp=sharing`
- **Data_Visualisation_v1.ipynb-**`https://colab.research.google.com/drive/1XEPwOoT4W4kSRLyb-M_RuhrJ6oJQoUNq?usp=sharing`
- **Window Sampling and Features Extraction + CNN Layers.ipynb-**`https://colab.research.google.com/drive/15UIgUpZ4xrqecHJKcwrFMik1FsKyYuRp?usp=sharing`
- **SVM.ipynb-**`https://colab.research.google.com/drive/1mJhIHC20NavECZmwP0v9iwuN_4mO-dQs?usp=sharing`
- **training_v1.ipynb-**`https://colab.research.google.com/drive/1olbM_D7PF81AgQ-Ut5aiLzPOgPmIF3-i?usp=sharing`
- GitHub repository:`https://github.com/rpandya5/gaitanalysis`

REFERENCES

The mobifall and mobiact datasets — bmi. URL https://bmi.hmu.gr/the-mobifall-and-mobiact-datasets-2/.

José David López Angela Sucerquia and Jesús Francisco Vargas-Bonilla. Sisfall: A fall and movement dataset. Technical report, Universidad de Antiquia, 2017.

Kelvin Choi. Android sensor logger app. URL https://www.tszheichoi.com/sensorlogger.

L. Oneto X. Parra Jorge Luis Reyes-Ortiz D. Anguita, A. Ghio. Human activity recognition using smartphones. Technical report, UC Irvine Machine Learning Repository, 2013.

Centers for Disease Control and Prevention. Older adult falls, 2024. URL https://www.cdc.gov/falls/about/index.html#:~:text=Key%20points,injury%20death%20for%20that%20group.&text=In%202021%2C%20emergency%20departments%20recorded,visits%20for%20older%20adult%20falls.