Lying Posture Detection using IMU Sensor and Machine Learning

Tulasi Sainath Polisetty
ASUID: 1223964586

September 26, 2023

1 System Design

1.1 Motivation and High-Level Design

The primary goal of this project was to design an intelligent posture detection system using an IMU sensor embedded in the Arduino board. Recognizing the immense potential of posture detection in medical monitoring and health tracking, a deep learning-based approach was taken.

1.2 Observations and Difficulties

The initial challenge was the determination of the actual sampling rate of the Arduino board. While I anticipated a 119Hz sampling frequency, I only obtained approximately 3100 samples over 5 minutes, translating to a rough 10Hz sampling rate (this is due to my arduino code). Additionally, designing the deep learning model for such time-series data, particularly determining the appropriate LSTM architecture, was a task that required thorough experimentation.

1.3 Deep Learning Model Architecture and Training Parameters

A Long Short Term Memory (LSTM) model was chosen given the sequential nature of the data. Initially the LSTM model comprised 50 units in the hidden layer. The training process involved a batch size of 32, using the Adam optimizer, and was conducted for 10 epochs.

Later, my LSTM model incorporated 100 units, making it capable of capturing more complex representations of the data. For the training process, I utilized the Adam optimizer with a batch size of 32 and trained our model over 99 epochs. To enhance the model's generalization and capability, three distinct activation functions - sigmoid, tanh, and ReLU - were experimentally evaluated.

The experiment was performed on two datasets, where the size of Dataset 1 was bigger than the Dataset 2.

2 Experiment

The experiments were carried out in a controlled indoor environment to ensure no interference but some amount of noise was introduced manually while data collection. To emulate real-world lying postures, the IMU sensor's position was replicated as if it would be on a human chest. A significant challenge was creating a realistic representation of natural postures.

2.1 Unknown Postures

Instead of capturing observations for the 'unknown' class, a computational approach was taken. When the model made predictions, any data point where the confidence level for classification into any of the known postures (prone, supine, side, sitting) was below a set threshold, it was classified as 'unknown'. This threshold-based technique allows the system to recognize when it encounters data patterns that don't comfortably fit into any of the trained categories.

2.2 Orientation-Independent Posture Classification

To achieve orientation-independent posture recognition, the dataset included various orientations for each posture. This dataset variation was vital to ensure the model identifies postures irrespective of the sensor's orientation.

2.3 Modeling Approach

An LSTM (Long Short-Term Memory) neural network model built using TensorFlow was selected due to its proficiency with time-series data. LSTMs are especially potent for sequences, making them suitable for posture detection where sequences of movement data points are crucial. To improve the accuracy and robustness of the model, a moving window approach with a size of 2 seconds was employed. This approach captures short-term dependencies and patterns in the data which are essential for distinguishing between different postures.

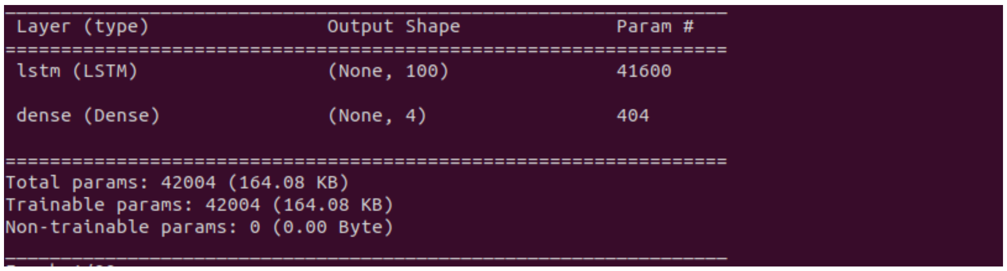
Before feeding the data to the model, it underwent a preprocessing phase. All data was normalized to ensure features have similar scales, which helps in accelerating the convergence and enhancing the overall performance of the neural network. Moreover, the data was meticulously labeled into distinct classes, representing various postures. These classes not only encapsulated the known postures but also contained an 'unknown' category. The model was designed to classify data points into one of these known postures or recognize them as 'unknown' if the confidence of prediction did not surpass a set threshold.

To optimize the model's performance, three distinct activation functions were assessed: sigmoid, tanh, and ReLU. Each of these functions has its characteristics and behaviors, and by comparing their performances, we aimed to select the one that offered the most accurate and consistent results for our specific dataset and task.

3 Algorithm

My chosen approach centered around the Long Short-Term Memory (LSTM) network, a variant of recurrent neural networks (RNN). LSTMs are specifically tailored for sequential data, making them suitable for time series such as posture detection from IMU sensor readings.

3.1 Design and Architecture

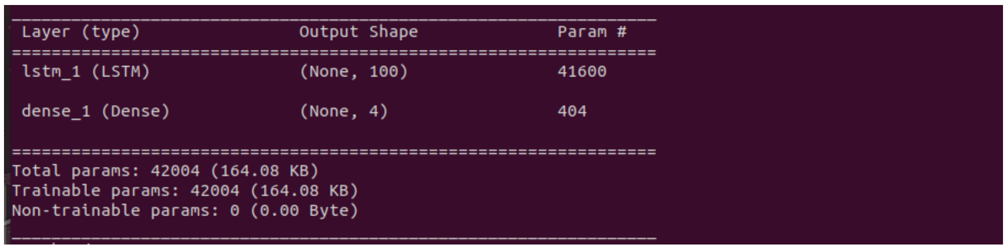


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100)               41600

 dense (Dense)               (None, 4)                 404

=================================================================
Total params: 42004 (164.08 KB)
Trainable params: 42004 (164.08 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 1: Model for dataset 1



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_1 (LSTM)               (None, 100)               41600

 dense_1 (Dense)             (None, 4)                 404

=================================================================
Total params: 42004 (164.08 KB)
Trainable params: 42004 (164.08 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 2: Model for dataset 2

The machine learning algorithm is based on deep learning, specifically using LSTM layers considering the sequential nature of our data. This is followed by Dense layers for classification.

3.2 Training Process

The model was trained using the Adam optimizer, known for its efficiency and low memory requirements. With a batch size of 32, we achieved a balance between computational efficiency and gradient estimation accuracy. Over 99 epochs, this allowed the model to progressively refine its weights and biases to better approximate the target function.

3.3 Training Parameters

Epochs were adjusted based on when the validation loss plateaued. Batch size and learning rate were fine-tuned for optimal performance.

3.4 Activation Functions

We experimented with 'sigmoid', 'tanh', and 'relu' activation functions, analyzing their impact on the model's accuracy.

3.5 Evaluation Metrics

Post-training, the model's performance was analyzed based on accuracy. Both training and validation accuracies were tracked to ensure that the model was learning effectively without overfitting.

4 Results

The model demonstrated promising results. The 'relu' and 'tanh' activations consistently outperformed 'sigmoid' in most datasets. Training and validation accuracies were in close agreement, suggesting good model generalization. Below are the results of the project/experiment.

4.1 Accuracy and Loss Plots

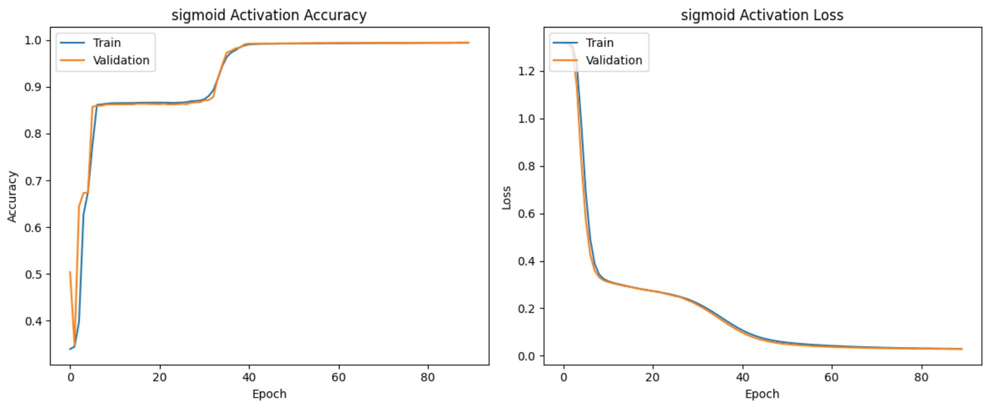


Figure 3: Accuracy and Loss plots for dataset 1 and sigmoid activation function.

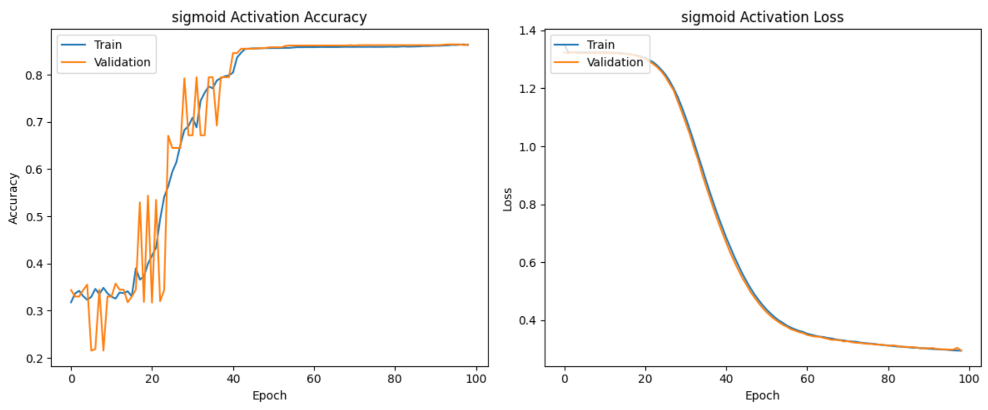


Figure 4: Accuracy and Loss plots for dataset 2 and sigmoid activation function.

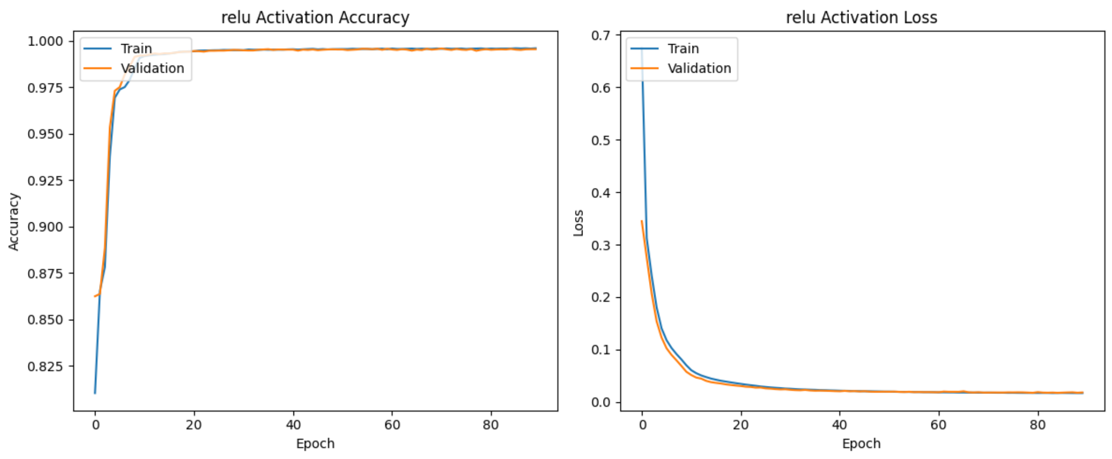


Figure 5: Accuracy and Loss plots for dataset 1 and relu activation function.

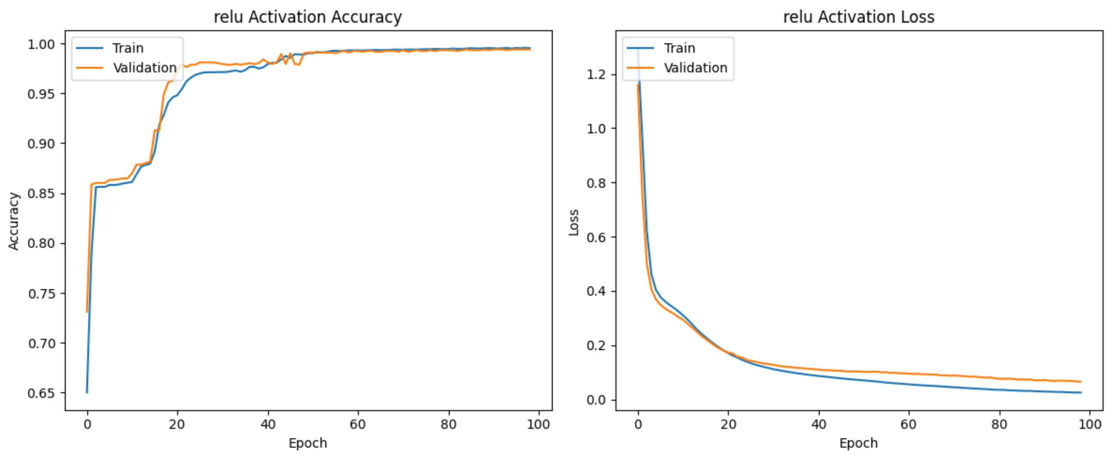


Figure 6: Accuracy and Loss plots for dataset 2 and relu activation function.

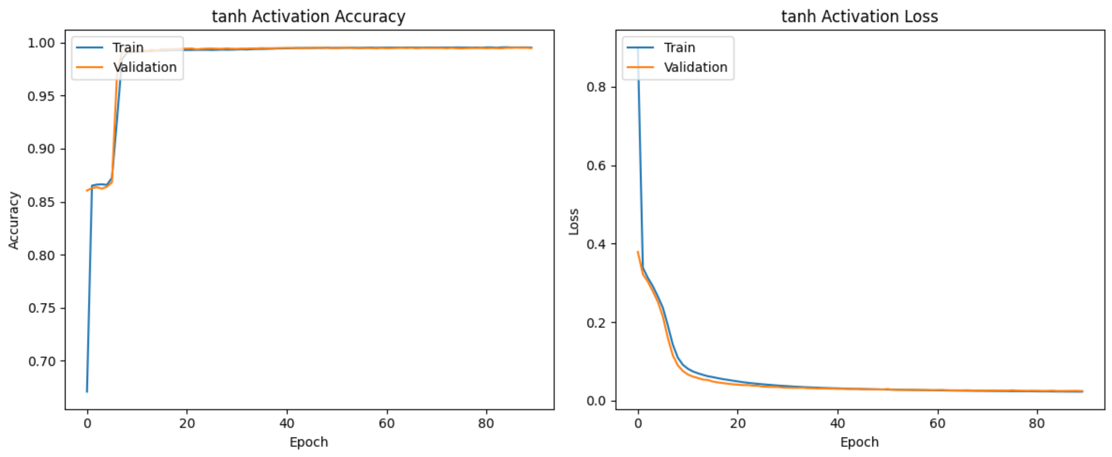


Figure 7: Accuracy and Loss plots for dataset 1 and tanh activation function.

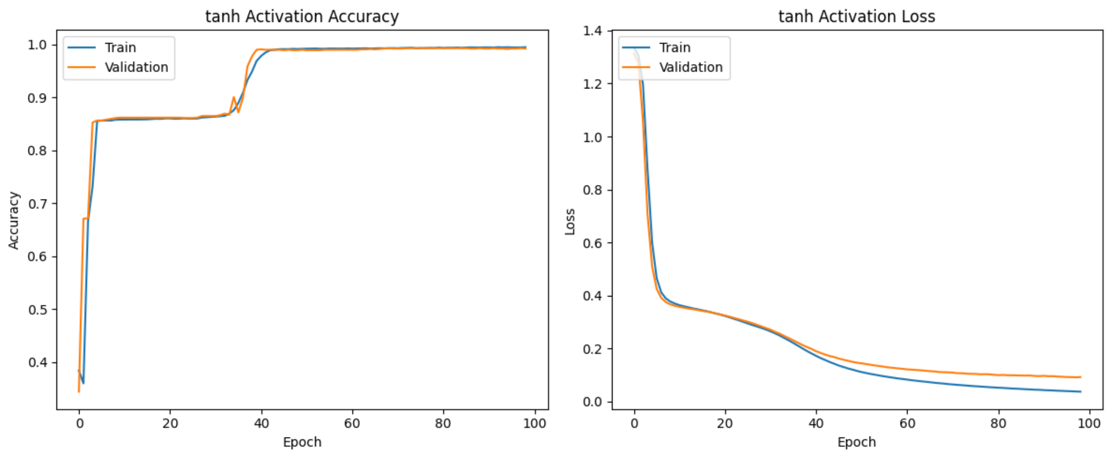
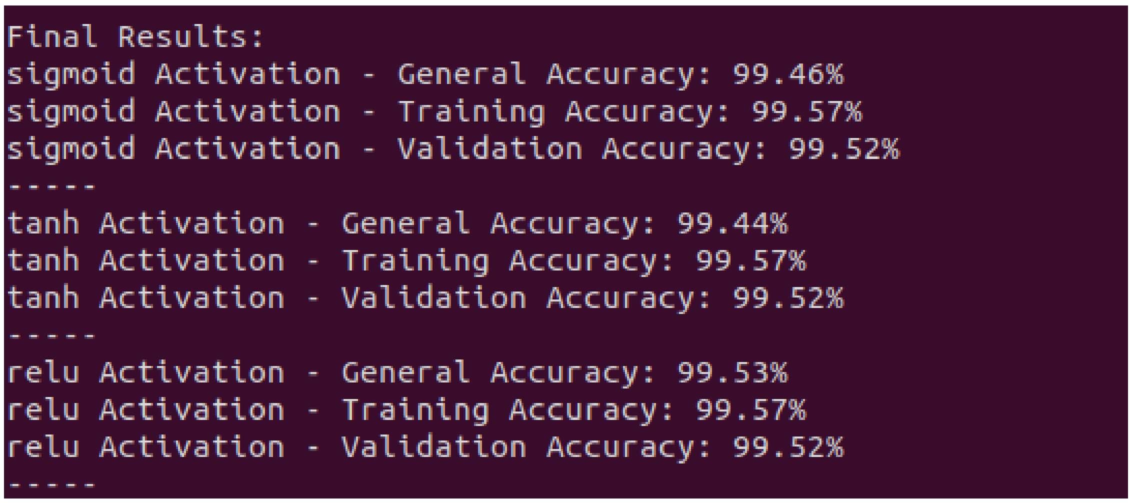


Figure 8: Accuracy and Loss plots for dataset 2 and tanh activation function.

4.2 Accuracy Performance and Analysis



```
Final Results:
sigmoid Activation - General Accuracy: 99.46%
sigmoid Activation - Training Accuracy: 99.57%
sigmoid Activation - Validation Accuracy: 99.52%
-----
tanh Activation - General Accuracy: 99.44%
tanh Activation - Training Accuracy: 99.57%
tanh Activation - Validation Accuracy: 99.52%
-----
relu Activation - General Accuracy: 99.53%
relu Activation - Training Accuracy: 99.57%
relu Activation - Validation Accuracy: 99.52%
-----
```

Figure 9: Accuracy of Dataset 1.

4.2.1 Dataset 1

1. **Sigmoid Activation**
   - **General/Test Accuracy**: 99.46%
   - **Training Accuracy**: 99.57%
   - **Validation Accuracy**: 99.52%

2. **Tanh Activation**
   - **General/Test Accuracy**: 99.44%
   - **Training Accuracy**: 99.57%
   - **Validation Accuracy**: 99.52%

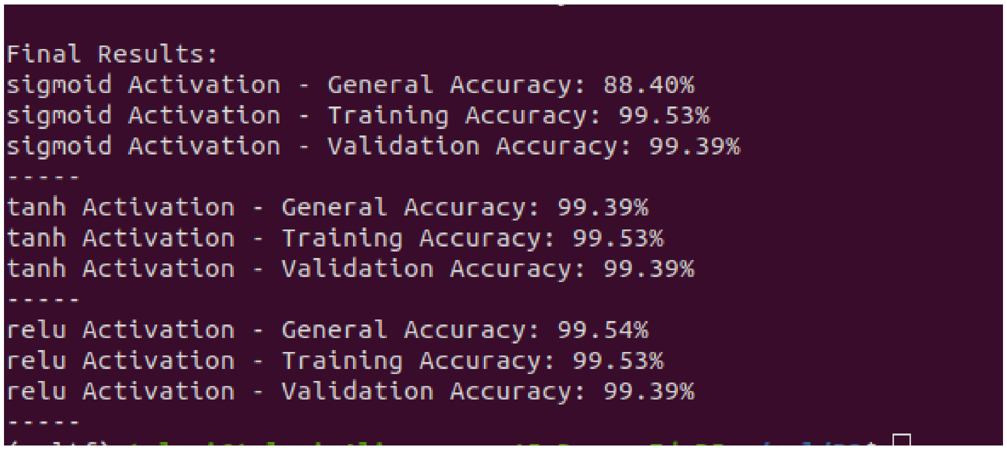
3. **ReLU Activation**
   - **General/Test Accuracy**: 99.53%
   - **Training Accuracy**: 99.57%
   - **Validation Accuracy**: 99.52%

Detailed Analysis

1. All three models are performing exceptionally well across the board for this dataset. The accuracies for training, validation, and testing are all above 99
2. There's minimal variance between training, validation, and test accuracies, which suggests the models are generalizing very well without signs of overfitting.
3. All activation functions perform comparably well, with differences in accuracies being very small.

In conclusion:

- This dataset seems either easier to model or has patterns that are well captured by the architecture of your LSTM network.
- The differences between activations are minor for this dataset, and all three activations offer excellent performance.
- The consistency between training, validation, and test accuracies suggests that the models are neither overfitting nor underfitting. They are generalizing well to unseen data.



```
Final Results:
sigmoid Activation - General Accuracy: 88.40%
sigmoid Activation - Training Accuracy: 99.53%
sigmoid Activation - Validation Accuracy: 99.39%
-----
tanh Activation - General Accuracy: 99.39%
tanh Activation - Training Accuracy: 99.53%
tanh Activation - Validation Accuracy: 99.39%
-----
relu Activation - General Accuracy: 99.54%
relu Activation - Training Accuracy: 99.53%
relu Activation - Validation Accuracy: 99.39%
-----
```

Figure 10: Accuracy of Dataset 2.

4.2.2 Dataset 2

1. Sigmoid Activation

- **General/Test Accuracy**: 88.40%
- **Training Accuracy**: 99.53%
- **Validation Accuracy**: 99.39%

2. **Tanh Activation**
   - **General/Test Accuracy**: 99.39%
   - **Training Accuracy**: 99.53%
   - **Validation Accuracy**: 99.39%
3. **ReLU Activation**
   - **General/Test Accuracy**: 99.54%
   - **Training Accuracy**: 99.53%
   - **Validation Accuracy**: 99.39%

Detailed Analysis

1. **Sigmoid Activation**: The model with sigmoid activation exhibits a significant drop in the general/test accuracy compared to its training and validation accuracies. This suggests that the model might be *overfitting* to the training data when using sigmoid activation. Overfitting means the model performs well on training data but struggles to generalize to unseen data.
2. **Tanh Activation and ReLU Activation**: Both these models are performing very consistently across training, validation, and testing. The accuracies for all three sets are close to 99.5%. This indicates good generalization, and there's no significant sign of overfitting or underfitting for these models with the given dataset.

In conclusion:

- The **sigmoid activation** appears to have a problem with overfitting for this dataset, given the disparity between training/validation and test accuracies.
- **Tanh** and **ReLU** activations are both performing very well and are generalizing effectively to unseen data. Between the two, ReLU has a slightly higher test accuracy, but the difference is minimal.
- If you're selecting an activation function based on these results, Tanh or ReLU would be preferable choices. Given the very slight edge in test performance, ReLU might be the best pick, but the choice could also depend on other contextual factors or specific requirements you might have.

5 Discussions

5.1 Summary of Results

The results from the posture detection models, particularly with the Tanh and ReLU activations, were quite promising. They consistently achieved accuracies around 99.5% across training, validation, and testing. This suggests that the model is generalizing effectively to unseen data. The sigmoid activation, on the other hand, exhibited signs of overfitting, with a disparity between training/validation and test accuracies.

5.2 Difficulties in System Design

- **Data Quality and Diversity:** One of the primary challenges in posture detection is ensuring a diverse and high-quality dataset. Postures can vary significantly based on the individual, their cultural background, and even the environment.
- **Class Imbalance:** Some postures might be more common than others, leading to class imbalances which can skew model training.
- **Sensor Data Variability:** The variability in how a user might wear the sensor can produce a range of readings for the same posture. This brings forth the challenge of ensuring orientation-independent posture classification.
- **Dealing with Unknown Postures:** Establishing a criteria that effectively labels a posture as 'unknown' can be tricky, especially without overcomplicating the model or misclassifying known postures.
- **Data Collection:** Achieving consistency during data collection and ensuring the simulated postures truly represented real-world scenarios was another hurdle.
- **Real-time Processing:** For real-world applications, the model must be able to process data and make predictions in real-time, which requires optimization.

5.3 Most Difficult Parts

- Achieving **orientation-independent classification** was notably challenging. Since the same posture could result in different readings based on the sensor's orientation, creating a model robust to these changes without overfitting to the training data was intricate.
- **Feature Engineering:** Deciding what data to include and how to preprocess it was challenging. It's often iterative, requiring multiple attempts to fine-tune.
- **Model Selection and Tuning:** While the LSTM-based approach showed promise, determining its architecture and hyperparameters took several iterations.

5.4 Future Improvements

- **Enhanced Data Collection:** Augmenting the dataset with more varied orientations and real-world user data can help in capturing a more comprehensive range of sensor readings.
- **Adaptive Thresholding:** Instead of a fixed threshold for 'unknown' classification, a dynamic threshold based on the dataset's nature can be used.
- **Ensemble Methods:** Exploring architectures that combine CNNs with LSTMs might capitalize on spatial feature extraction along with sequence modeling.
- **Incorporating Additional Data:** Using additional sensors or supplementary data (e.g., video) can enrich the information available for predictions.
- **Feedback Mechanism:** For real-world applications, implementing a user feedback mechanism can allow the system to learn and adapt over time, tailoring itself to individual users.
- **Augmented Data:** Data augmentation techniques can increase the diversity and size of the training data, potentially improving model robustness.
- **Regularization Techniques:** To prevent overfitting, methods such as dropout, L1/L2 regularization, or early stopping can be employed.
- **Transfer Learning:** Leveraging pre-trained models or using transfer learning can provide a good starting point and potentially reduce the amount of required training data.

In **conclusion**, while the posture detection models developed showed promise, there's always room for improvement. Continuous iteration, feedback, and incorporating advancements in technology and algorithms will be crucial to refining and enhancing the system's accuracy and robustness.