***SeiSense* - How to expand seizure monitoring to fitness trackers***Nova IMS – EDSA (Deep Learning and Neural Networks)*

**Group 08**Frederico Monteiro 20211786; Maria Correia 20211155; Mariana Gil 20211043; Manuel Félix 20211333; Vasco Pombo 20211301

**Abstract**This report presents a proposed add-on feature for the *SeizAlarm* application that aims to detect seizures in people not diagnosed with epilepsy. By using machine learning techniques to determine the position (or state) of the user and resorting to already known details such as irregular motion and abnormal heart rates, this new feature called *SeiSense* would be able to accurately detect a preemptive generalised seizure fit. Both the native application and this new feature would rely heavily on gathered data from smartphones or wearables, equipped with both gyroscope and accelerometers. The data team resorted to Neural Network models, specifically Multilayer Perceptron models, to train and successfully fit a model that would identify a user’s state from a pool of six possible, with a minimum accuracy established at 80%. To achieve this, the data team used multiple iterations with varying hidden layer settings and optimizers, with their current succession resulting in an accuracy of 0.9386 and a precision of 1.0 for the Laying activity outcome.

**Tags:** machine learning, MLP, seizure detection, API, add-on feature, Neural Network models, gyroscope, accelerometer.

1. **Introduction**

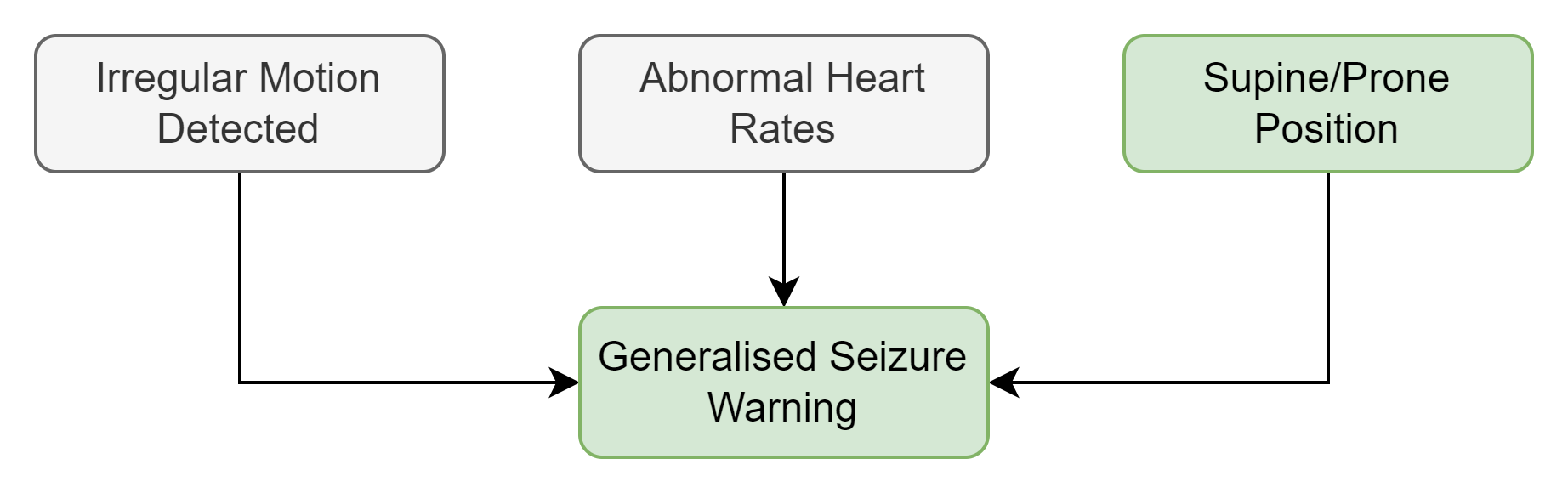
Epilepsy is one of the most common conditions affecting the brain, with an estimated number of cases in the vicinity of 50 million worldwide. With most documented cases happening in young children or elderly people (over 65 years old), it is expected that in the US alone, 1 in every 100 people will experience a single unprovoked seizure. However, even with the majority of documented cases pointing to recurring cases that end up being monitored, there are still episodic seizures that result from different, often physical causes, usually referred to non-epileptic seizures (NES).

Apart from continuous monitoring and lifelong medication, early detection and length between events are essential to monitor patient wellbeing. For a successful diagnosis of a seizure, it is often important to identify the type of seizure: whether it is a focal or generalised one. Focal seizures tend to include auras, symptoms that warn the patient that a seizure is about to occur, with most common auras including mood swings, visual and hearing abnormalities or changes in the sense of smell. Generalised ones often lead to loss of consciousness, with muscle spasms and jerking reactions.

As a continuous effort to detect early onset signs of seizure fits, several applications have been built that monitor both erratic movements and elevated heart rates monitored by users, such as *SeizAlarm*. By including customizable heart rates per user and by specialising in detecting seizure-like motion through their proprietary algorithm, they are able to preemptively warn both users and caregivers that a seizure is imminent. To achieve this, an accelerometer is usually used, either in a smartphone or wearable format, to determine rapid variations in movement.

The motivation for this project comes from an add-on feature envisioned by the exploration of *SeizAlarm*, where our data science team focused on two main objectives:

* the inclusion of users ranging from 18 - 65 years old, with the main objective being the preemptive detection of seizures in people who are not diagnosed with epilepsy and already being monitored;
* the creation of an added feature to detect whether the user would be in a prone/supine position, a result of a generalised seizure with loss of consciousness.



*Image 1 - Flowchart for Generalised Seizure Warnings*

This would not only make the application more accessible and marketable, but would also allow the possibility to connect to different active applications such as Google Fit, Apple Health, Fitbit or Strava, allowing for an emergency message to be triggered outside of the normal *SeizAlarm* application, via API as an example.

Our business case for this project will be the following:

* As a part of a collaboration with *SeizAlarm*, we would not focus on detecting irregular motion nor heart rate, as their algorithm currently monitors them. Instead, our project would rely on determining accurately the position of a user.
* As we are developing an add-on for an existing application, we should focus on exploring existing physical features from smart devices (IoT, smartphones or wearables) to determine the position of the user. The safest choices will be the accelerometer or gyroscope hardware embedded.
* By choosing to open this information as an API, our underlying business model would also change: *SeizAlarm* would be able to reach outside of their usual market share by including virtually anyone (instead of restricting itself as a monitoring application for diagnosed epileptic patients). As such, our team would suggest that the name of the application (or add-on feature) would differ from *SeizAlarm*, with the proposed name being *SeiSense* (patent pending).

1. **Motivation**

As the project developed, our data team quickly understood that by identifying a user’s position based on a smartphone/wearable would add value to *SeiSense’s* strategy, and set out to gather information that would sustain our pitch. We resorted to a Human Activity Recognition dataset available on Kaggle, that would require a Machine Learning algorithm, potentially based around Neural Networks. The plan to proceed was the following:

1. Find a dataset that would match our target demographic, with the purpose of identifying the user’s position (referred to as state);
2. Create a model by understanding the most relevant features that would be able to recognise the different states based on user parameters, derived from the accelerometer and gyroscope embedded in the device;
3. Determine a realistic expectation to validate the model both in terms of accuracy and precision.
4. Prepare a use-case for applicability and a roadmap for direct integration with *SeizAlarm* or via API with third-party users.

The proposed dataset was built from 30 participants within an age bracket of 19-48 years, which is relevant for our target demographic outside of the most common range in epilepsy. These participants carried a waist-mounted smartphone, with the dataset gathering information about the embedded accelerometer and gyroscope. Each variable was recorded and resulted into a classification of six different activities performed, resulting in a state. Both accelerometer and gyroscope variables were separated and tagged, with their consequent labels:

* “t” variables represent time domain, and “f” represent frequency domain;
* “Body” for body motion components of the accelerometer/gyroscope, “Jerk” for erratic body motion components, “Mag” for magnetometer components and “Gravity” for gravitational ones;
* “XYZ” represents the relevant axis (*image 5*).

Being an ambitious data team (and also to compensate for the restricted number of users that compiled our dataset), we sought to achieve a minimum of 85% of accuracy in our model. As a by-product, we strived to maximise precision, ideally above 90%. These will constitute our benchmarks for evaluating the models later on.

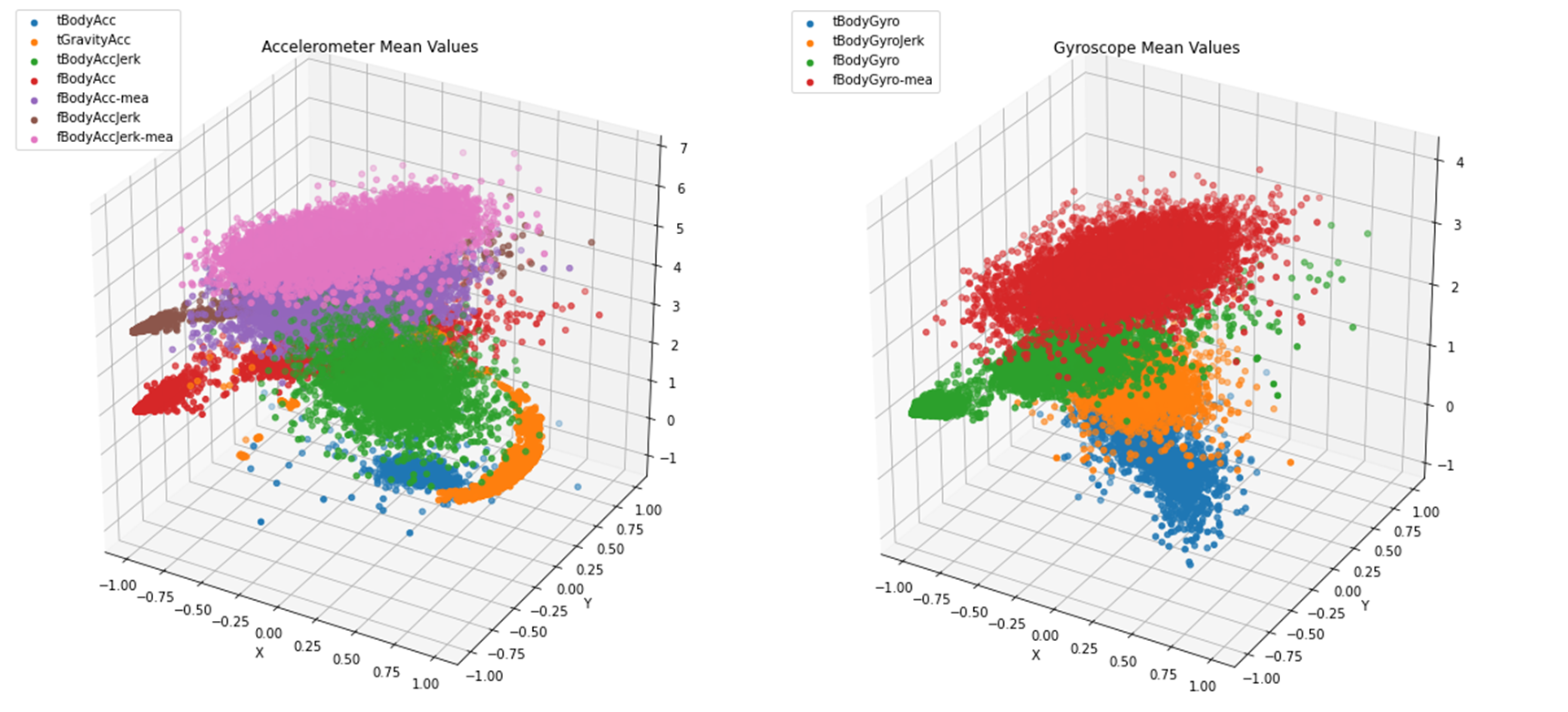
1. **Data**

The combination of variables described above result in a total of 563 distinct variables, with a main purpose of identifying which relevant ones will determine the *Activity* outcome. No data cleaning processes are required for this dataset, as no duplicates nor missing values were found. The provided dataset was identified as balanced, meaning that the train/test datasets are evenly split between each of the distinct activities, which means that we are able to directly assess the different train/test datasets without any data pre-processing. A total of 10.302 entries were gathered, with a 70/30% train/test split. Prior to the model train, our team extracted a validation set from the train set in a proportion of 20%.

Our data team set on three pivotal questions:

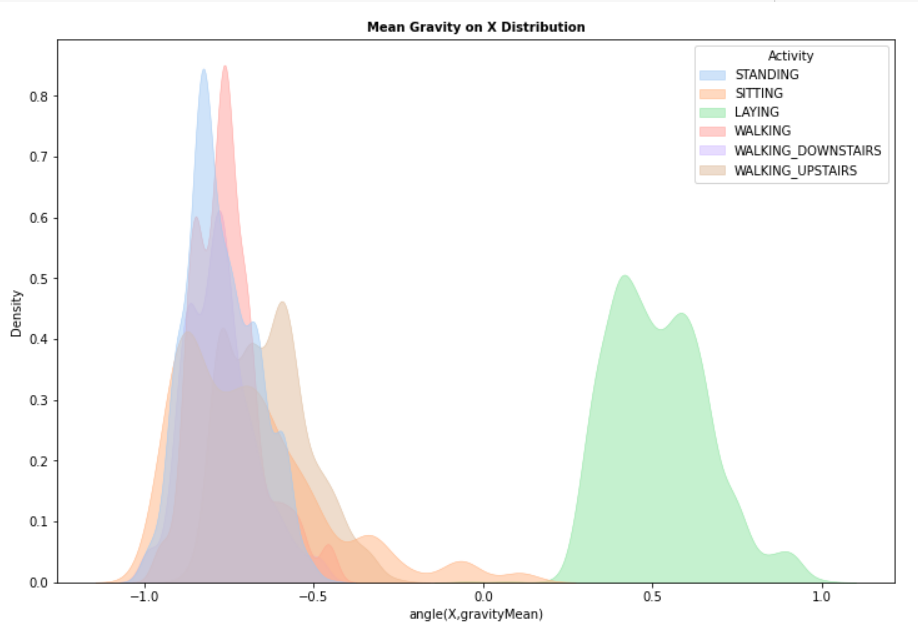
* Sensor importance between accelerometer and gyroscope: the application *SeizAlarm* already incorporates the wearable/smartphone accelerometer into their algorithm, so we should try to maximise reach by incorporating both accelerometer and gyroscope if possible;
* Gravity (specified axis) for *Activity* **Laying**: based on our understanding of the dataset, our team can foresee the connection between gravity and a specific axis (X or Y-axis) to determine whether the user is in a prone/supine position, to indicate loss of consciousness;
* Stationary *vs* Movement: is there a clear distinction between stationary and movement in the model to prevent overlap with the erratic motion detection from *SeizAlarm*.

After exploring the dataset, a clear tendency favours accelerometer over gyroscope for determining walking as an outcome. At a glance, by assessing mean values per variable related to each device, we failed to determine whether this could be expanded to stationary states, such as **Standing, Laying** and **Sitting**, and should be considered in order to prevent skewness.



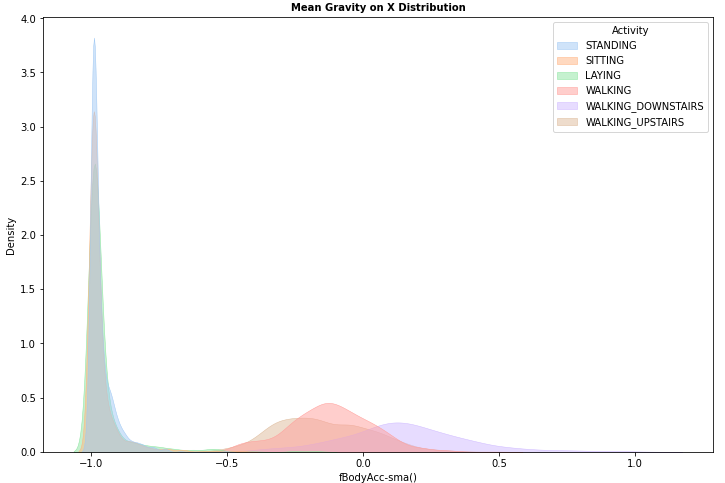
*Image 2 - Accelerometer and Gyroscope mean values. The scatter plot in each axis makes it hard to identify a specific pattern.*

As expected, values for the Gravity variable are biassed when paired with the X-axis for **Laying**, which translates to the user being in prone/supine position. On both Y and Z-axis, the recorded variations are also more significant for **Laying** but are not explanatory of the prone/supine positions, as the recordings are more centred around negative angle values.



*Image 3 - Mean Gravity on X-axis distribution based on accelerometer angle, with a clear distinction on* ***Laying****.*

Furthermore, stationary states can also be determined by resorting to the signal magnitude area gathered via frequency (or *sma()*) from the accelerometer’s body towards a specific axis. Similar to **Laying**, a clear split can be identified between stationary states and moving states when assessing the signal emitting around the X-axis, which corroborates the findings mentioned previously.



*Image 4 - Mean Gravity on X-axis distribution based on accelerometer’s signal magnitude area, with a clear distinction between stationary and moving states.*

1. **Model**
   1. **Reducing Dimensionality**

Knowing a bit more about the dataset, our team set out to create the proposed model. The first task would be centred around the variables and how to reduce them. The second aspect would revolve around the type of model to apply and its functions, optimizers and hidden layers. Finally, the proposed metrics to evaluate each model and their fitness.

To address dimensionality, we resorted to Principal Component Analysis (PCA), a technique that seeks to explore the correlation structure of the original variables in a data set. The underlying idea behind PCA is to generate new variables, called principal components (PCs), constructed as linear combinations of the data set original variables, representing a smaller number of uncorrelated variables that can summarise the information in the larger original data set. Thus, two of its main purposes are to be used as a data-reduction technique and to unveil new correlations in the original data set, being often used in exploration and pre-processing stages, and to attain better computational performances when dealing with very large data sets.

The resulting accuracy after deploying PCA with 50 variables resulted in a high tendency for Accelerometer (12 out of the top 15 variables), 7 of which related to erratic body movement components, which led us to try a broader approach with AutoEncoders. An AutoEncoder is a type of neural network, often used for unsupervised learning, specifically for the task of dimensionality reduction and feature learning. It is composed of encoders and decoders, the former mapping the input data to a lower-dimensional representation (acting as a bottleneck) and the latter mapping the representation back to the original dimension. Its purpose is to reconstruct the original input from the compressed representation.

An easier way of thinking about AutoEncoders is as a non-linear version of PCA. Both PCA and autoencoders attempt to find the most informative linear/non-linear combination of the input features. Also, they can be helpful for feature selection because they learn a compressed representation of the input data that tries to capture the most informative and relevant features of the input. During training, the autoencoder learns to ignore the noise and less relevant features in the input data, and instead focuses on the features that are most important for accurate reconstruction. This means that the bottleneck representation of the input data can be used as a new set of features that are more informative and relevant than the original features. As an outcome, the accuracy obtained from the AutoEncoder features (0.985) was higher than the one obtained with the PCA (0.943), while also relying on accelerometer and gyroscope variables analogously.

* 1. **Model Assessment**

As all successful models, our team attempted to baptise it as something overly complicated such as Tensorial Usable and Relative Directional model, but ultimately decided against it due to lack of creativity and the outcoming acronym.

We chose to implement a multi-layer perceptron (MLP) model for the classification task. MLPs are a type of feedforward neural network that are usually applied within multi-class classification problems due to their ability to learn non-linear decision boundaries. MLPs require:

* activation functions;
* optimizers
* hidden layers

In the setup of multi-layer perceptrons (MLPs), we opted for the rectified linear unit (ReLU) as the activation function. ReLU is a common choice due to its computational efficiency and ability to improve the model's learning capabilities. It evaluates to the input when it is positive, reverting to 0 when it is negative, adding non-linearity to the model and enabling the network to learn complex decision boundaries. Additionally, the ReLU activation function can help to mitigate the vanishing gradient problem, a common issue when training deep neural networks with sigmoid or tanh, two distinct activation functions.

When designing the architecture for the MLP training, we needed to choose the right optimizer to employ in order to adjust the network's weights. There are various optimizers available, each with distinct strengths and weaknesses, with some of the most common ones mentioned in by various authors being Stochastic Gradient Descent, RMSprop and Adam.

* Stochastic Gradient Descent (SGD): A simple yet effective optimizer that iteratively updates the weights by taking the negative gradient of the loss function with respect to the weights.
* RMSprop: A variant of SGD that uses a moving average of the squared gradients to adjust the learning rate for each weight, potentially accelerating convergence by making larger updates for weights that are changing slowly and smaller updates for those that are changing quickly.
* Adam: An advanced optimizer that combines the ideas of RMSprop and SGD with momentum, using an exponential moving average of the gradients to scale the learning rate and an exponential moving average of the squared gradients to scale the learning rate for each weight. It also features an adaptive learning rate mechanism, which may improve performance further. Adam is known to be an efficient and robust optimizer, and it's considered by many a good default choice for MLPs, because it can adapt well to different types of data and problems, and it's not very sensitive to the selection of the learning rate hyperparameter. The model that performed best, and therefore was chosen, had Adam as its optimizer.

The selection of the number of hidden layers and the number of neurons per layer can significantly affect the performance of a MLP model. Generally, increasing the number of hidden layers and neurons per layer can improve the model's capacity and enhance its ability to capture more intricate decision boundaries, thus leading to better performance on the training data. However, this also increases the risk of overfitting, which occurs when the model becomes too intricate and starts to capture the noise in the data rather than the underlying pattern.

Our model experienced six different variations, alternating between the hidden layer structure and the used optimizer. As for the activation function, ReLU was always used, due to its quick conversion and faster training time. The outcoming models tested were the following:

* Model 1 - 2 hidden layer structure: [1000,500] and RMSprop optimizer
* Model 2 - 3 hidden layer structure: [1000,500,250] and RMSprop optimizer
* Model 3 - 2 hidden layer structure: [1000,500] and Adam optimizer
* Model 4 - 3 hidden layer structure: [1000,500,250] and Adam optimizer
* Model 5 - 2 hidden layer structure: [1000,500] and SGD (learning rate = 0.01) optimizer
* Model 6 - 3 hidden layer structure: [1000,500,250] and SGD (learning rate = 0.01) optimizer
  1. **Evaluation**

The outcome of these six variations would be benchmarked in terms of accuracy and precision. During the exploration phase, our team decided that our minimum thresholds should be 85% in accuracy with maximum precision possible (ideally above 90%).

* Accuracy: measure of how often the classifier makes the correct prediction. It is the ratio between correct predictions and total number of predictions and can be defined as: (Number of Correct Predictions) / (Total Number of Predictions).
* Precision: measure particularly relevant in cases where the cost of false positives is high. It is the ratio between the number of true positive predictions and total positive predictions (both true and false) and can be defined as: (Number of True Positives) / (Number of True Positives + Number of False Positives).

In this particular scenario, false positives want to be avoided as much as possible. By creating SeiSense as a built-in feature for third party providers, we want to only add to their “experience”, meaning we only want to issue a seizure warning when we are absolutely certain of it. The experience of highlighting potentially fake seizures would lead to a loss in confidence in already established platforms, which would ultimately hinder customer experience and work against the set business model.

In terms of model accuracy:

* Model 1 - validation accuracy of 0.9701
* Model 2 - validation accuracy of 0.9674
* Model 3 - validation accuracy of 0.9646
* Model 4 - validation accuracy of 0.9803
* Model 5 - validation accuracy of 0.9694
* Model 6 - validation accuracy of 0.9667

The Adam optimizer was responsible for both the best overall validation accuracy (in model 4) and worst overall validation accuracy (in model 3), both measured against the validation dataset. However, this can also be prone to overfitting, so a more conservative approach should be taken, such as understanding accuracy against unseen data. In this sense, model 3 ended up having a better prediction in unseen data, with an accuracy of 0.9386, whereas model 4 had it under 0.9. We then tested model 3 for precision regarding **Laying** activity models, where it scored 1.0.

To help visualise different iterations in model 3, we resorted to TensorBoard, a web-based tool provided with TensorFlow. It allows users to interactively visualise and analyse the training of their machine learning models. It can display various metrics, such as accuracy and loss, as well as visualisations of the model's structure and its performance over time. It also supports visualising and comparing different runs of the same model, which was useful for comparing the results of different hyperparameter settings or optimizers in successive iterations. It allowed us to have a better understanding of how the model achieved a better accuracy with every epoch, by how much it improved per epoch and the best results per iteration.

We deemed model 3 as the best overall model, with both metrics performing well above the expected mark, thus concluding that it would solve the initial proposed metrics.

1. **Conclusion and further work**

Our team prepared itself prior to the meeting with *SeizAlarm*, as we knew we had a winning formula on our hands: the model had been tried and tested. The first step would be to reiterate the applicability: now the application would be able to be expanded to non-diagnosed users of any age group, meaning virtually anyone using a phone could get a seizure warning without any prior monitoring. Specifically with *SeizAlarm*, we would work closely with them to embed this feature into their platform, as it would also benefit their ongoing algorithm.

Our next steps would be the following:

* Optimise our proposed hyperparameters within model 3, to potentially reduce overfitting even further. This was not deem a priority as we were exceeding our initial targets, but we could try to strive for better accuracy values, such as 95%;
* Expand our database to further solidify our findings and check whether our model can correctly predict user states. Cross-referencing people with epilepsy in the mix could also enhance it, to try to develop specific patterns combining it with the **Laying** state;
* With the integration with *SeizAlarm* complete, we would be able to export it as an API directly to fitness trackers, as they engage directly in heart rate monitoring. As such, if the fitness tracker could detect abnormal heart rates while in unusual states, further diseases could be investigated.
* A V1.1 deployment of *SeiSense* as a seizure tracker with fitness trackers, which would include additional features such as voice activated commands for emergency calls. A running gag within the team was a link to an exorcism application, should the emergency call be made in Latin, albeit not directly in line with the proposed business plan.

**References**

***Introduction:***

* [*https://www.who.int/news-room/fact-sheets/detail/epilepsy*](https://www.who.int/news-room/fact-sheets/detail/epilepsy)
* [*https://www.epilepsy.com/what-is-epilepsy/understanding-seizures/who-gets-epilepsy*](https://www.epilepsy.com/what-is-epilepsy/understanding-seizures/who-gets-epilepsy)
* [*https://www.hopkinsmedicine.org/health/conditions-and-diseases/epilepsy/evaluation-of-a-firsttime-seizure*](https://www.hopkinsmedicine.org/health/conditions-and-diseases/epilepsy/evaluation-of-a-firsttime-seizure)
* [*https://www.epilepsy.com/tools-resources/device-wiki/seizalarm-seizure-detection-mobilewatch-application*](https://www.epilepsy.com/tools-resources/device-wiki/seizalarm-seizure-detection-mobilewatch-application)

***Motivation:***

* *Joana Raquel Cerqueira da Silva (2013), “Smartphone Based Human Activity Prediction”.*

***Data:***

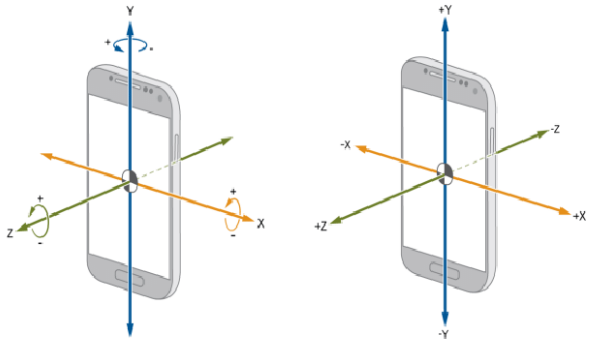
* [*https://www.kaggle.com/code/essammohamed4320/human-activity-recognition-scientific-prespective/notebook#Word-about-the-dataset*](https://www.kaggle.com/code/essammohamed4320/human-activity-recognition-scientific-prespective/notebook#Word-about-the-dataset)

***Model:***

* *Hotelling, H. (1933), “Analysis of a complex of statistical variables into principal components”****.***
* *Dor Bank, Noam Koenigstein, Raja Giryes (2021), “AutoEncoders”.*
* *"Multilayer feedforward networks are universal approximators" by Hornik et al. (1989)*
* *"Rectifier Nonlinearities Improve Neural Network Acoustic Models" by Maas et al. (2013)*
* *"On the Convergence of Adam and Beyond" by Reddi et al. (2018)*
* *"Deep Learning" by Bengio et al. (2015)*

***Other references (bonus for reading):***

******

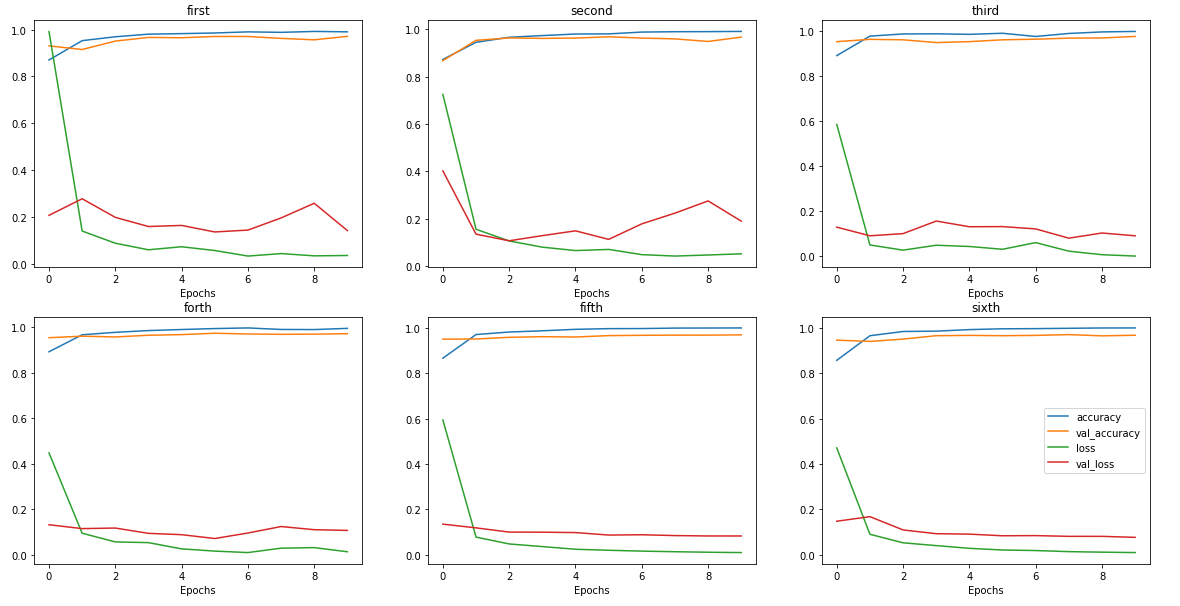
**Annexes**

**Accelerometers** is an electronic sensor that measures the acceleration forces acting on an object, in order to determine the object’s position in space and monitor the object’s movement. **Gyroscope** is a device that can measure and maintain the orientation and angular velocity of an object. These are more advanced than accelerometers. These can measure the tilt and lateral orientation of the object whereas an accelerometer can only measure the linear motion.

*Image 5 - XYZ axis on a smartphone*

*Table 1 - Top 15 PCA variables (gyroscope variables in* ***bold****)*

| 280 - fBodyAcc-min()-Z  359 - fBodyAccJerk-min()-Z  95 - tBodyAccJerk-min()-Z  **438 - fBodyGyro-min()-Z**  226 - tGravityAccMag-arCoeff()4 | 231 - tBodyAccJerkMag-sma()  515 - fBodyAccMag-kurtosis()  520 - fBodyBodyAccJerkMag-min()  228 - tBodyAccJerkMag-std()  227 - tBodyAccJerkMag-mean() | **175 - tBodyGyroJerk-min()-Z**  507 - fBodyAccMag-min()  **502 - BodyGyro-bandsEnergy()-25,48**  517 - fBodyBodyAccJerkMag-std()  83 - tBodyAccJerk-mean()-Z |
| --- | --- | --- |

*****Image 6 - Variation of each model iteration per epoch.*