Delft University of Technology

Object Classification with Radar
EE4675

Project: Classification of Different Human Activities with Radar

Authors:
Mujtaba Hassan (923008)
Yanwen Chen (5477018)
Victor Oliveira (5653797)

June 17, 2022

1 Introduction

In the last years, with the evolution of radar systems combined with the huge development of the AI field, the object classification problem got a new set of solving strategies. The use of numerous Machine Learning (ML) techniques to analyze and classify radar data (mainly Micro-Doppler signatures) has been promising. Moreover, the contexts where it can be applied are many: military, automotive, health prediction (the focus of this project), etc.

In this project, the main goal was to be able to identify (using data collected by an FMCW Radar) different activities performed by people through an ML method. After identifying the activities, a system would be capable of monitoring the activity levels and patterns of people, detecting e.g. critical events such as falls [1]. To build the Machine Learning model, a set of 6 different activities were considered: walk, sit down, stand up, pick up an object, drink water, and fall. To have a better understanding of what the radar data (shown as a spectrogram) of each activity looks like, the figure 1 is presented below.

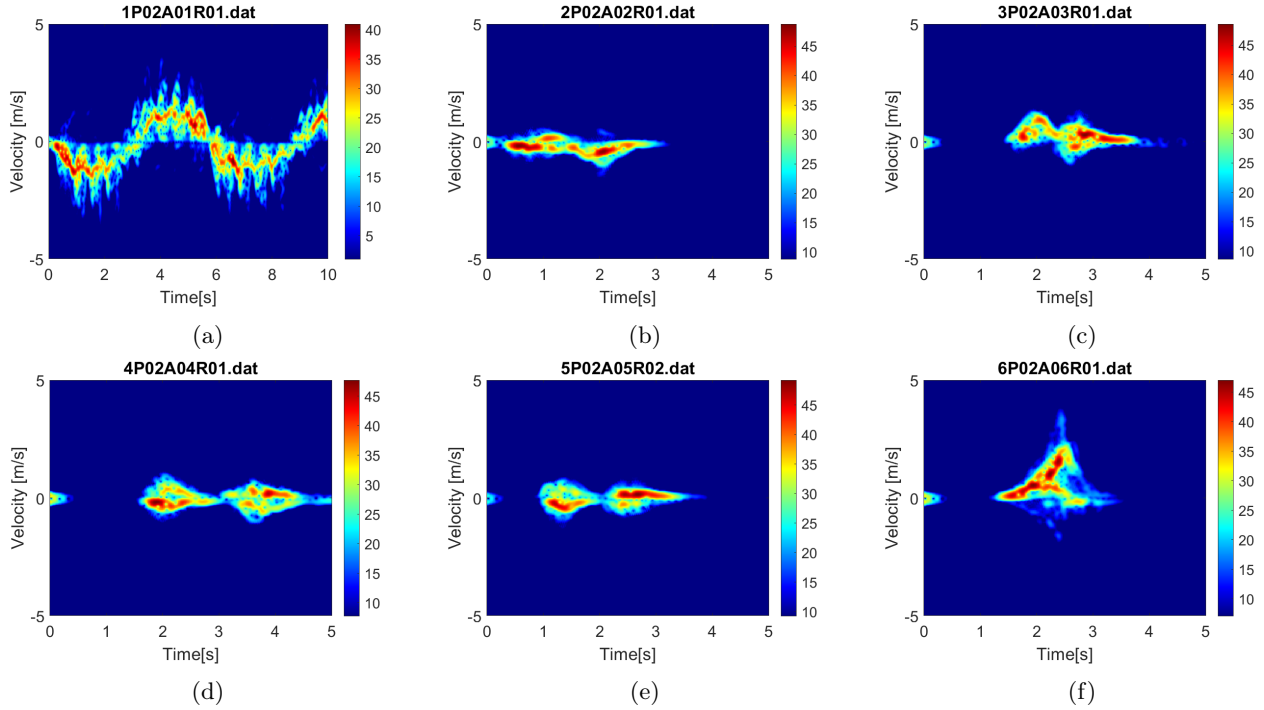

Figure 1: Examples of spectrogram for the 6 possible activities: (a) walk; (b) sit down; (c) stand up; (d) pick up an object; (5) drink water; (6) fall.

To be capable of classifying activities, the procedures of an ML pipeline had to be followed. The FMCW radar data was already collected, so no work had to be done to generate them. With the raw data in hand, data processing was necessary to get the range x time plot and the spectrogram. After that, features from the spectrogram were extracted and employed to construct the feature matrix, applied as an input in the ML classification methods. It was decided to use more than one ML technique to increase the chances of getting a better model. The (K Nearest Neighbor) KNN, (Support Vector Machine) SVM, and (Neural Network) NN were implemented and evaluated.

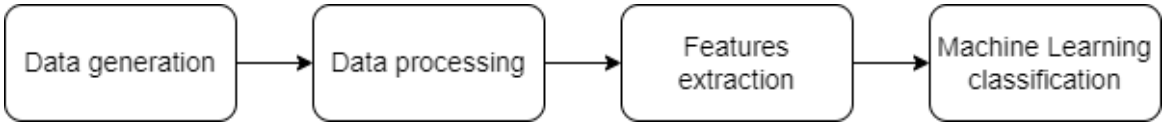

Figure 2: Machine Learning pipeline.

2 Data Processing

The input in this project is raw ADC data. This data needs to be converted into a format that can be used by a classification algorithm such as point cloud based neural network for human activity classification. We modified the code provided in the example Matlab script to generate velocity-time and range-time maps from this ADC data. In the subsequent sections, we explain the concept behind Matlab code used to obtain these radar features.

2.1 FMCW Radar

In this project, an FMCW radar is used to obtain the information about the targets. The idea behind this radar is to transmit a frequency modulated signal whose frequency is changing over time from central f_0 to cover a certain bandwidth B (as explained in Figure 3, and the reflected signals from the target are recorded by the receiving antenna. One transmission of the signal with frequency going from f_0 to $f_0 + B$ is called a chirp. The reflected signal from the target will have a complex attenuation and a certain delay which are proportional to the distance of the target from the radar. Thus, multiplying (mixing) the transmitted and received signals at the receiving end of the radar, due to the time delay, there will be two main frequencies in the obtained signal: one (comparatively smaller- called beat signal) related to the delay of the reflected signal, and a second one in the order of carrier frequency. Using a low pass filter, the carrier frequency component will be filtered out, and the beat signal will contain frequency components that are directly proportional to the distance of the possible targets.

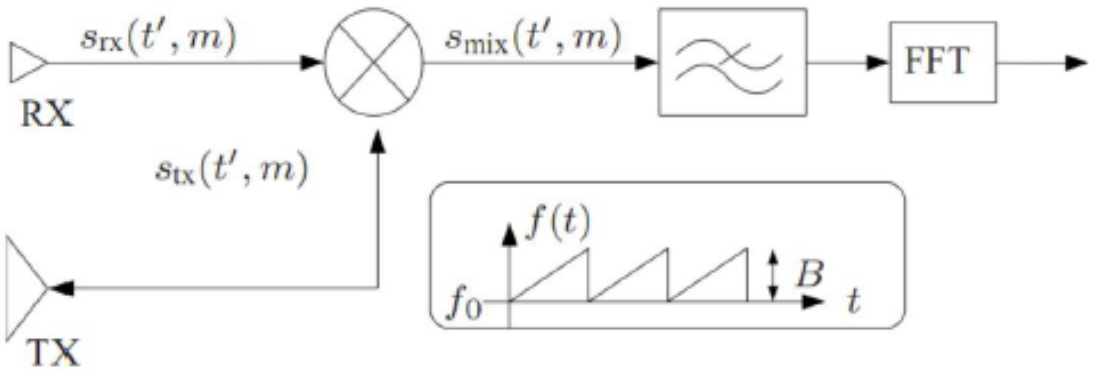

Figure 3: Schematic representation of how data is obtained using transmitted signal(TX) and received reflected signal(RX)*.

2.2 Range-Time Estimation

In order to obtain the range of the target, the following wave is sent through the transmitter:

$$S_{tx}(t) = e^{j2\pi(f_c t + \frac{\beta t^2}{2})}, t \in [0, T_s] \tag{1}$$

Where f_c is the carrier frequency, and β is the coefficient that represents the slope of the chirp of FMCW radar, it can be expressed as $\beta = \frac{B}{T_s}$, where B is the bandwidth and T_s is the chirp time. After the signal hits the target, the received signal has the following form:

$$S_{rx}(t) = \alpha \cdot e^{j2\pi(f_c(t-\tau(t)) + \frac{\beta(t-\tau(t))^2}{2})}, \tag{2}$$

Where α represents a complex attenuation on the signal, and $\tau(t) = \frac{2R(t)}{c}$ represents the round-trip time needed for the electromagnetic wave. If we assume that the velocity v_0 of the target is constant, then the range of the target is a function of time and can be represented with the formula $R(t) = R_0 - v_0 t$, where R_0 is the initial distance between the radar and target. Therefore, the time delay can be obtained:

$$\tau(t) = \frac{2R_0}{c} - \frac{2v_0 t}{c} = \tau_0 - \frac{2v_0 t}{c}, \tag{3}$$

The second part of the equation $\frac{2v_0t}{c}$ makes up the Doppler shift in frequency due to the speed of the target. Here, the assumption is made that this influence of Doppler shift is negligible compared to the beat signal, leaving us with $\tau(t) = \tau_0$. As seen in Figure 3, after the transmitted and received signals are passed through the mixer, we obtain the beat signal:

$$S_b = S_{tx}S_{rx}^* = \alpha \cdot e^{j2\pi(f_c t + \frac{\beta t^2}{2} - f_c(t-\tau(t)) - \frac{\beta(t-\tau(t))^2}{2})}, \tag{4}$$

when we substitute $\tau(t) = \tau_0$, and cancel out the same terms we get:

$$S_b = \alpha \cdot e^{j2\pi(\beta\tau_0 t + f_c\tau_0 - \frac{\beta}{2}\tau_0^2)} = \alpha \cdot e^{j2\pi(f_b t + \phi_0)}, \tag{5}$$

From this we can conclude that by analyzing the power spectrum of the beat signal and finding which frequency components are present in the beat signal, we can obtain the range and returned power from the radar to the targets. The range information can be calculated in the following manner: $f_b = \beta\tau_0 = \frac{B}{T_s}\frac{2R_0}{c} = \frac{2BR_0}{cT_s}$, from here the range is $R_0 = f_b \cdot \frac{cT_s}{2B}$. As can be seen, the range of the target is directly proportional to the beat frequency.

2.3 Velocity-Time Estimation

Between each consecutive chirp, the small displacement of the target due to the constant velocity will lead to a constant phase change along the time. This constant phase change is much more noticeable (magnified by the reciprocal of the wavelength of the carrier frequency) compared to the Doppler frequency shift. We used this change to determine the velocity. The resolution of the velocity and time depends on the scan rate and the total integration time. Since the scan rate is a fixed parameter of the radar, we can also adjust the integration time (i.e. the number of scans, which in this project is 200), but we cannot increase the integration time as much as we want due to the resolution trade-off in time and frequency. Since the fixed scan rate is 1 KHz and 200 scans are coherently integrated, the time resolution is 0.2 seconds.

For the 5.8GHz FMCW radar, the wavelength would be $\lambda = \frac{c}{f} = 5.2cm$. Although, as explained in the previous paragraph, the Doppler shift will not result in a noticeable frequency change, the phase change is significant. The phase changed can be obtained by:

$$\Delta\phi = 2\pi f_c \Delta\tau = \frac{4\pi\Delta d}{\lambda}. \tag{6}$$

For example, a velocity of $10m/s$ during a scan period of $T_c = 1ms$ will introduce a phase change of $\approx 13.8°$ between each chirp, but it will only make a range displacement of only 0.01m during one chirp. For this reason, we see that the phase shift between the peaks of the two consecutive chirps contains the information on the velocity of the target. Finally, if we substitute that the relative displacement of the target is equal to $\Delta d = v \cdot T_c$, we obtain the following formula for the velocity:

$$\Delta\phi = \frac{4\pi v T_c}{\lambda} => v = \frac{\lambda\Delta\phi}{4\pi T_c}. \tag{7}$$

2.4 Spectrogram

Despite the bulk movement of the target, objects have additional moving parts of different velocities, such as the different velocities between the limbs and torso. These so-called micro-motions can result in additional patterns in the Doppler spectrum of the received signal, which is referred to as the micro-Doppler phenomenon. This is one of the key features we can take advantage of to perform the object classification of different types of human activities.

Though FFT is the most common and efficient method to achieve frequency analysis, it is unable to provide time-dependent frequency information. Therefore, a joint time-frequency analysis that provides localized time-dependent frequency information with high resolution is needed.

One of the most prevail tool to analyze this time-varying micro-Doppler frequency of the received radar signal is the Short-Time Fourier transform (STFT). It can be represented as:

$$STFT\{x(t,\omega)\}(\tau,\omega) \equiv X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-j\omega t}dt. \tag{8}$$

Eq. 8 realizes a designed window convolution with a short segment of signal in the time domain and computes the Fourier transform separately on each time segment. The spectrogram is the magnitude of $|STFT\{x(t,\omega)|$.

Three key parameters of the STFT is the duration of the window in time, overlap between two consecutive windows, and the type of the window. In our processing, we use the default configuration of the provided script, the duration of the window is 200 samples, overlap factor is 0.95 and no additional smoothing window is applied.

3 Features Extraction

After processing the raw data and obtaining the spectrogram, the next step is to extract meaningful information from it. The combination of such extracted features will then be used to build the feature matrix - which later will be the input of Machine Learning (ML) classification methods. All the features extracted and tested during the project are presented in the upcoming sections. Note, however, that not all of them were used to construct the final feature matrix. Some features did not show good results and were discarded.

3.1 Centroid and Bandwidth

Some of the most commonly used features in object classification with radar field come from the centroid and bandwidth of spectrograms. In a Doppler x Time graph (spectrogram), the centroid tells how the weighted average Doppler frequency evolves over time. To calculate it, the equation presented in 9 is used. The bandwidth, on the other hand, gives information about the range of Doppler frequency in time, and its calculation is performed by equation 10.

$$f_c(j) = \frac{\Sigma_i f(i)\, F(i,j)}{\Sigma_i\, F(i,j)} \tag{9}$$

$$B_c(j) = \sqrt{\frac{\Sigma_i (f(i) - f_c(j))^2\, F(i,j)}{\Sigma_i\, F(i,j)}} \tag{10}$$

As an example, for the spectrogram shown in figure 4, the centroid and bandwidth plots obtained are presented below. From each of these curves, a set of parameters (mean, skewness, kurtosis, variance, etc.) can be extracted and tested as features.

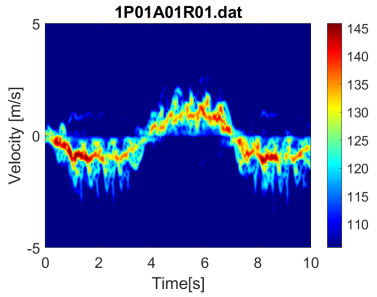

Figure 4: Spectrogram example of the first activity (walk).

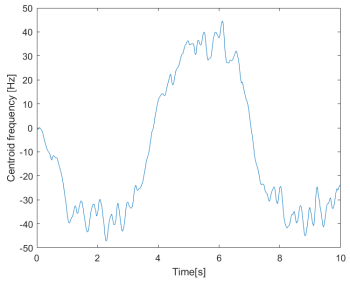

Figure 5: Centroid evolution over time.

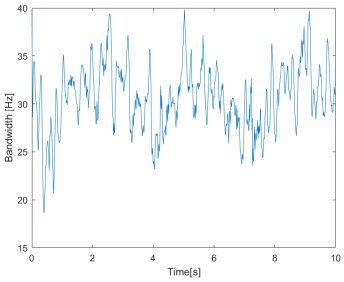

Figure 6: Bandwidth evolution over time.

However, the bandwidth has a weird behavior for activities without movements during some time interval. In figure 7, there is one of these such cases, and in figure 8 the resulting bandwidth. As it can be observed, instead of going to 0 when no movement occurs, the bandwidth goes to a very high level, causing misinterpretations. Therefore, the bandwidth parameters were not used as features.

3.2 Envelope

From the spectrograms of the different activities shown in the introduction, it is possible to see that the envelope is the main difference between them. Thus, it is likely to get good features from it.

To extract the envelope of the Doppler frequency, all the data with the same timestamp are compared to a threshold of 110 dB. The greatest frequency and the lowest frequency of that timestamp - in which its data still passes the limit - are saved. This process is repeated for all the other timestamps, so all the points of the upper and lower envelope are collected. In figures 9 and 10 it is shown a spectrogram and its envelopes.

From the envelopes, a set of information is obtained:

- The maximum frequency of the upper envelope ($f_{max,uenv}$)

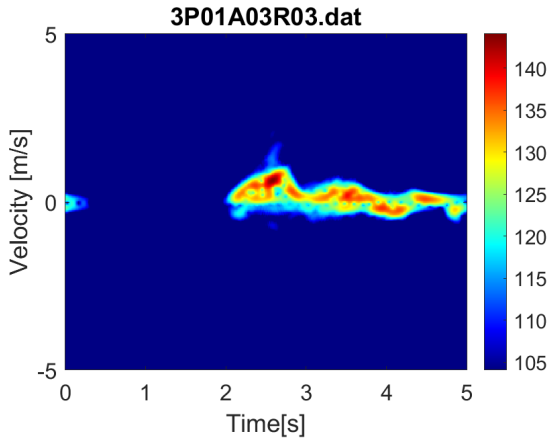

Figure 7: Spectrogram example of activity 3.

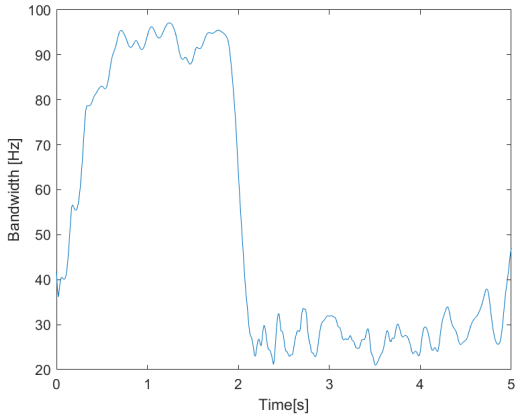

Figure 8: Bandwidth with strange behavior over time.

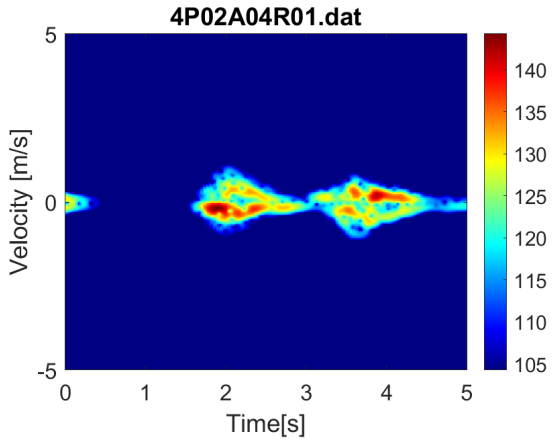

Figure 9: Spectrogram example of activity 4 (pick up an object).

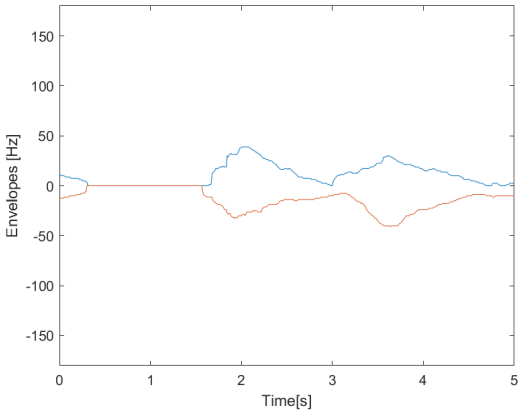

Figure 10: Upper and lower envelope over time.

- The minimum frequency of the lower envelope ($f_{min,lenv}$)
- The mean of the difference between upper and lower envelope ($\Delta f_{env,avg}$)
- The frequency variation in the timestamp with the maximum frequency ($\Delta f_{env,max}$) and with the minimum frequency ($\Delta f_{env,min}$)

3.3 Feature Standardization

Before passing the feature matrix to the classification model, it is important to guarantee that differences in the scale of the feature values will not affect the classification model. KNN (for example) uses the distance between the points as a classification method. If a feature has much higher values than the other features, its weight in the calculation of the distance will be greater and consequently its influence in the classification as well. Therefore, to ensure that all the attributes have the same importance for the model, the feature matrix should be standardized.

There are several ways of standardizing the data. Nevertheless, in this project, it was only tested two: the *zscore* method, and the quartile method. The *zscore* method is the most common form of normalizing data. It centers and scales the values to have a mean of 0 and a standard deviation of 1 (see figure 11). On the other hand, the quartile method is preferably used when the data contains many outliers. In such cases, these outliers are disregarded from the dataset, and the normalization process is done using the quartiles (see figure 12).

Hz) and the same order of magnitude, the accuracy of the models did not change considerably when changing the normalization method, even without it the accuracy did not decrease greatly.

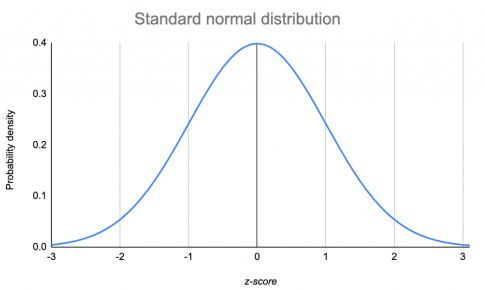

Figure 11: Standard normal distribution (zscore).

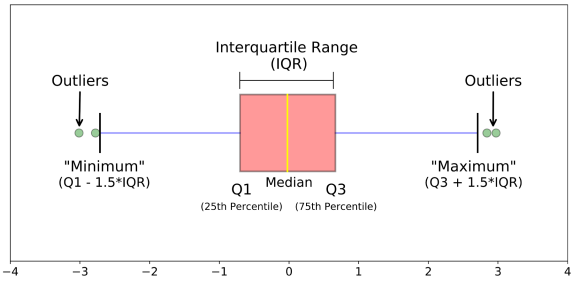

Figure 12: Distribution of a boxplot (quartile method).

3.4 Feature Visualization

To solve an object classification problem, feature visualization is not strictly necessary. Nonetheless, it is an essential step for those who want to fully understand the problem. With feature visualization, it is possible to see what is happening inside the feature matrix, and from this observation, one can state which labels are the most difficult to classify.

If only two features were used to classify the 6 activities, it would be very simple to visualize the values of each label. A normal plot (feature 1, feature 2) would be enough. However, generally, the number of features used is much higher than 2, making unfeasible a common 2D plot. Thus, to perform some sort of dimensionality reduction, the well-known t-SNE algorithm could be used. In the figure below, it is presented the t-SNE for one of the best feature configurations obtained.

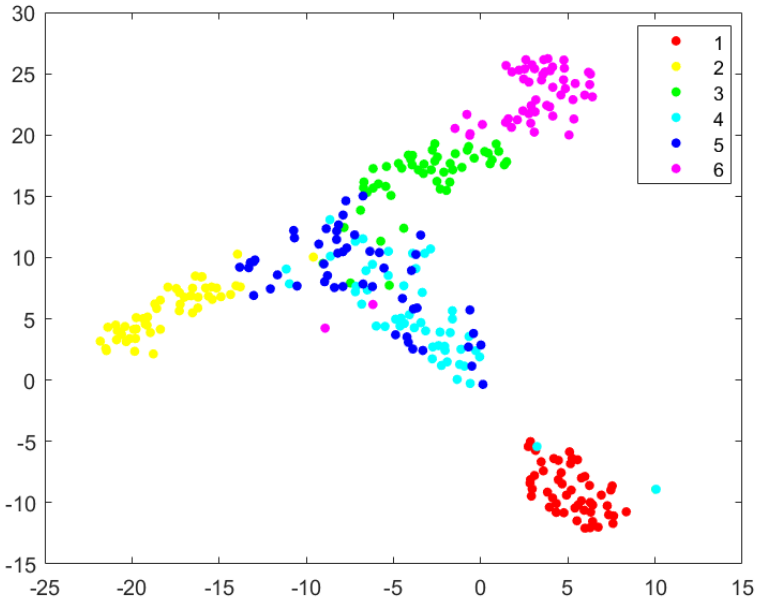

Figure 13: Visualization of the feature matrix using t-SNE algorithm.

As it can be observed from figure 13, the activities, which are the most similar regarding the feature matrix, are 4 (pick up an object) and 5 (drink water).

4 Machine Learning Classification

4.1 KNN

4.1.1 Short Theory

The KNN is a simple (probably the simplest one) and easy-to-implement classification algorithm, which assumes that data with the same label are normally near to each other. To measure this idea of proximity between two points, the distance between them is calculated (normally using euclidean distance: see equation 11). To classify a query, the KNN calculates and sorts the distances between it and all the training points. Next, it selects the K points that are closest to the query. Then, the most frequent label inside these K points is selected.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} \tag{11}$$

However, the KNN has the drawback of scaling poorly for larger datasets. In other words, it becomes considerably slow as the number of points increases. Besides that, there are faster algorithms that can produce more accurate classification results. Moreover, when using the KNN, extra care about the feature selection has to be taken. On one hand, if the number of features used is too low, it is not possible to differentiate all labels. On the other hand, a great number of not considerably good features decreases the efficiency of the good ones, diminishing the overall accuracy of the classifier. Finally, it is important to point out that before proceeding to the KNN, the feature matrix should be normalized.

4.1.2 Results

To estimate the accuracy of the KNN algorithm for the different configurations of features applied, the K-fold Cross-Validation approach was applied. The dataset utilized was the "5 February 2019 UoG Dataset" with 306 data. In general, the accuracy range was [75 - 84]%. In the end, the best configuration tested had 4 features: mean of the centroid, $\Delta f_{env,avg}$, $f_{max,uenv}$, $f_{min,lenv}$, and its accuracy oscillated between 82% to 84%. For this case, K values (number of neighbors) between 3 and 5 showed the best results - with basically the same accuracy between them.

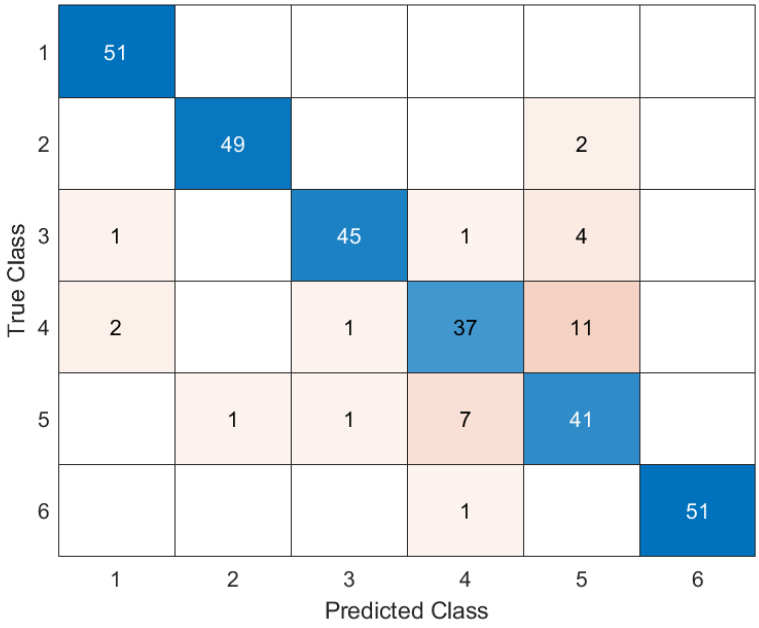

Figure 14: Confusion matrix.

The performance of the best configuration discussed above can be better interpreted by visualizing the confusion matrix presented in 14. As it can be observed, most of the mismatches happen between activities 4 and 5 (pick up an object and drink water, respectively). Looking at the figures 1d, 1e, and 13, it can be noticed that in fact, both activities have very similar spectrograms and features.

4.2 Support Vector Machine

4.2.1 Basic Theory

Support vector machine (SVM) is one of the most prevalent machine learning techniques, which provides a framework for supervised learning by creating a model by learning patterns within the input data [2]. For human target classification, as mentioned above, spectral features are commonly used for building the SVM classifiers.

SVM is essentially a non-probabilistic binary linear classifier, meaning that an SVM training algorithm builds a model that assigns new examples to one category or the other [3]. The mathematical formulation of SVM is straightforward as it is a linear technique in a higher dimensional feature space [4].

Take the basic binary classification as an example. The decision boundary between two classes will take the form[4]:

$$\mathbf{w}^T\mathbf{x} + b = 0, x \in R^M, b \in R \tag{12}$$

SVM aims to find the decision boundary that separates two classes of data points in the training set by the largest margin. The decision function can then be given as [4]:

$$\min \mathbf{w}^T\mathbf{w} \quad \text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \tag{13}$$

Here the different signs refer to the two types of classes. The principle here is to find the largest margin of the boundary that could separate the data point and limit the margin violations. This is called soft margin classification [5] compared with hard margin classification which strictly imposes the rule that all instances must be off the street and on the right side, as shown in the figure 15.The hyperparameter C indicates that low values enlarge the margin at the price of more violations, whereas high values restrict the margin.

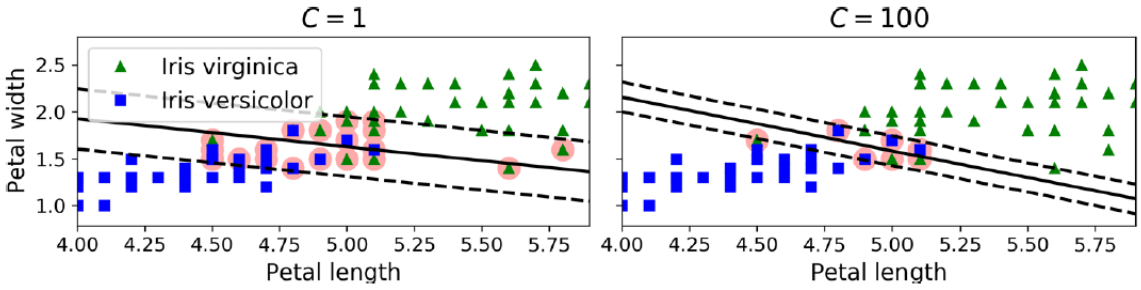

Figure 15: Large margin (left) versus fewer margin violations (right) [5] .

The illustration above uses the example of linear SVM classification. However, in real cases, many data sets are not linear separable. As a result, we use the polynomial kernel function. The function can be represented as $G(x_j, x_k) = (1 + x_j' x_k)^q$, where q indicates the order of the polynomial kernel.

4.2.2 One vs. All

As we mentioned before, the SVM is a binary classifier, which means it can only separate between two groups of data. However, our case is a multiclass classification problem, in which we need to classify 6 different kinds of human movement.

A common strategy to solve this problem is using so-called “one vs. all” strategy. This scheme is essentially training N different binary classifiers. Each one of these classifiers is trained to distinguish samples in a certain class with respect to all remaining classes. When a new sample came, the N classifiers are all run and the classifier which gives the largest value is chosen.[6]

4.2.3 Results

To estimate the accuracy of the SVM algorithm for the different configurations of features applied, the K-fold Cross-Validation approach was applied. The dataset utilized was the "5 February 2019 UoG Dataset" with 306 data. In general, the accuracy range was around 85%. Generally, the classifiers for class 4 and 5 have the lowest accuracy.The confusion matrix is shown in 16. Comparing the results with previous case, the SVM performs better than KNN, but the accuracy is still not satisfying.

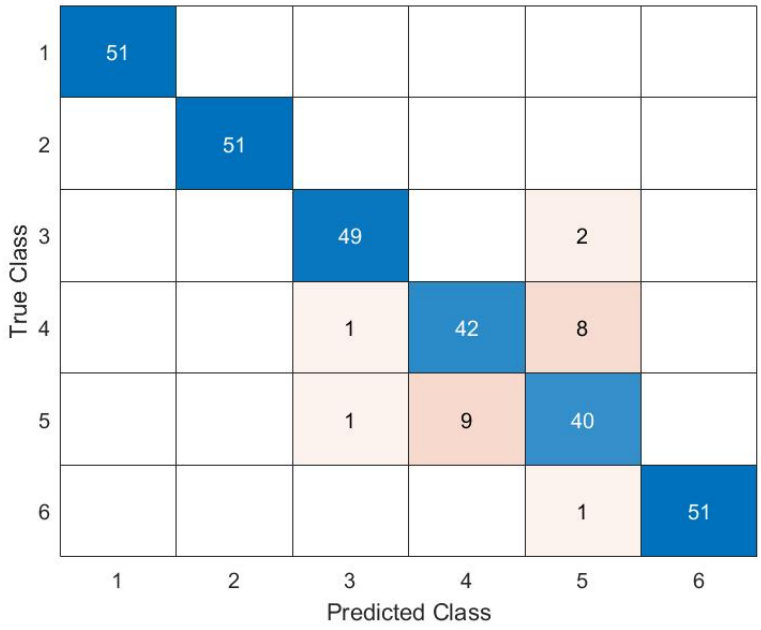

Figure 16: Confusion Matrix of SVM.

5 Point based Neural Network Classification

The goal of this portion of the project is to investigate whether point cloud is a decent representation for human activity classification task. The idea is that the point cloud extracted by detecting points from the range-velocity-time map will have a unique shape for each of the activity. This particular shape can be learned to be classified in a data-driven way using a point cloud based neural network.

The main contents of this investigation is to formulate the problem of human activity recognition as a problem of applying signal processing techniques to estimate different features present in the radar followed by converting these features into a radar point cloud. Then, to implement an off-the-shelf point cloud processing neural network for prediction and finally to explain the obtained results. In the following sub-sections, this will be described.

5.1 Pre-processing

5.1.1 Point Cloud Generation

Once we obtained the features in terms of velocity-time and range-time plots, the next step is to convert these features into a point cloud. In this respect, a CA-CFAR detector was used to detect essential points from the velocity-time plot.

Radar signals contain a lot of noise and some kind of thresholding algorithm is needed to determine the detections based on the reflected power. The threshold should be such that there should be minimum false alarms and miss detections. CA-CFAR is one way to achieve this objective whereby the probability of false alarm is set to a constant. The CA-CFAR algorithm find an adaptive threshold for each bin in the map based on the average of the values of the nearby bins. It is expected that the bins having higher reflected power than the threshold corresponds to a detection (in our case a point in velocity-time map corresponding to a shape feature) whereas a bin having lower than this value is only noise (region in velocity-time map providing no information on shape feature).

We used a two-way 1D CA-CFAR to obtain the detections from velocity-time plot. First, a horizontal 1D CFAR window was applied to obtain the useful features in time domain. This was followed by another 1D CFAR window in the vertical direction to obtain the useful features in velocity domain. Finally, the features from both these operations were multiplied to obtain the final points. We did not use a single 2D CFAR window since it gave us poor results owing to nature of velocity-time plot.

From the CFAR detector, we got two features, one is velocity and the other is time for the detected point. However, our point cloud is actually N detections with 4 features.The third feature is power which was obtained

by finding the power at each detection point. The fourth feature is the range which was found from extracting the range value from range-time plot corresponding to a detected point with a specific time value.

5.1.2 Point Cloud Visualization

As shown in Figure 17, the generated point cloud extracts the important features present in each of the range, velocity, and time axis. In this visualization example, we can see that there is a specific shape of the point cloud which is obtained due to the walking activity performed by the person whereby the range and velocity of the object continuously oscillate because of the movement of the target user away/towards the radar during the whole time duration. Also, some higher velocity components are captured as points based on the movement of hands and legs which are different than the movement of the whole body.

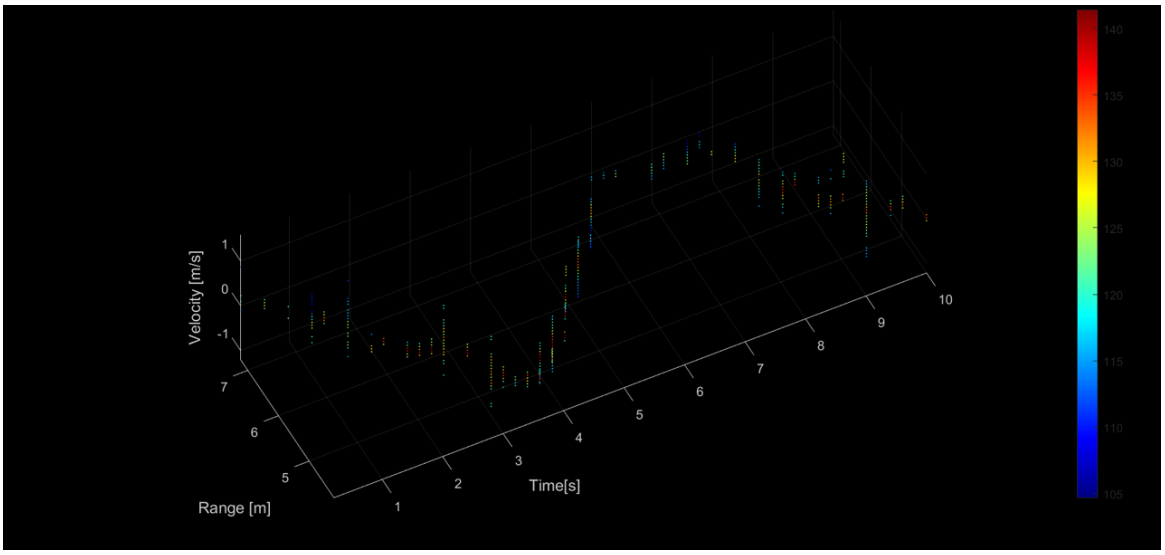

Figure 17: Visualization of the Point Cloud.

5.1.3 Saving Point Cloud to Memory

The neural network used during this experiment was implemented in python. So, we stored the point cloud generated through Matlab into the memory. To this end, the features were translated to a $n \times m$ matrix for each activity, where n denotes the points and m contains 4 values for the time (in seconds), the range (m), the velocity (m/s), and the signal power (dBm), respectively. Each matrix is written to a CSV file which can be read using python code.

5.2 Neural Network Classification

A vanilla version of PointNet [7], available online [8], was used for this project. We used a a simple PointNet because the purpose of this investigation was not to get the best results but only to determine the feasibility of point cloud based human activity classification.

5.2.1 Architecture

Figure 18 gives the architecture of PointNet. The input to the network is raw pointcloud data. It uses a shared multi-layer perceptron to extract features from the data, mapping each of the n points from input channel dimension (in this case 4), to 32 dimensions, followed by 64 dimensions, and then to 512 dimensions. Max pooling is then used to aggregate these features to obtain a global feature vector where max pooling is used since it is a symmetric function that is invariant to order of points. Finally, a fully-connected network is used to map the global features to the 6 output classification scores.

5.2.2 Training

We used all the dataset available to us for this project. We split the dataset into two parts: 80% for training and 20% for validation. Adam was used as the optimizer to train the network with a learning rate of 0.0005. Sparse

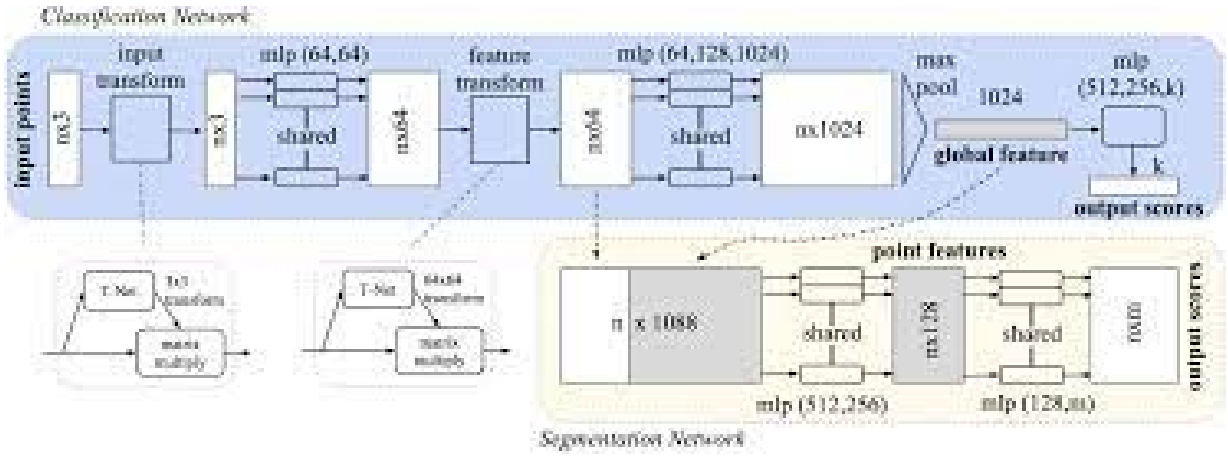

Figure 18: PointNet.

Categorical Cross Entropy was used as the loss function since we have a single value at the output specifying a classification label. The network was trained for a total of 50 epochs whereby batch size was taken as 32. It took us around 20 minutes to train the network for 50 epochs on an Intel Xeon W-2245 CPU. The total number of parameters is only 206, 886.

5.3 Experiments and Results

We trained the PointNet using the training procedure specified in the previous section. Figure 19 shows the result of the experiment (this is the best result that we achieved). Here, the orange and blue curves show the train and test accuracy respectively. The result looks quite promising keeping in view that we have a used a PointNet which is a very basic pointcloud classification network. Here, we must emphasize that our pointcloud is not a spatial pointcloud but a pointcloud that represents the shape of the activity. The result clearly shows that pointcloud representing the shape of the activity can be a good representation for classifying the different human activities.

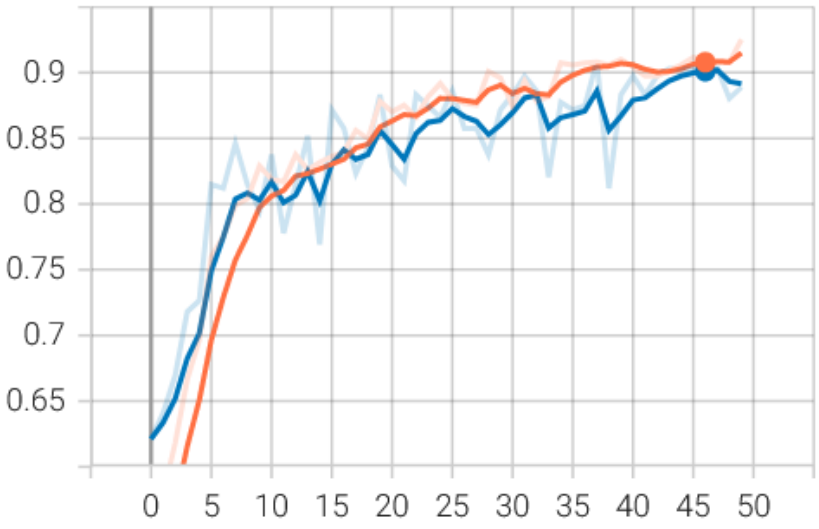

Figure 19: Accuracy on Train and Validation Data for PointNet.

We also did some analysis on the performance of the model on each of the activities. Figure **??** gives the confusion matrix results for the validation set. From the results, we can see the performance for the activities 1,2,3 and 6 is quite good. However, the network sometimes get confused between activity 4 and activity 5. Especially,the network sometimes outputs activity as activity 5 when the original activity is activity 4. If we check activities 4 and 5, they are "Bending to pick up an object" and "Drinking from a cup or glass". These activities are actually slightly similar to each other as compared to the other activities which are much different

than each other. So, this means that separating these tasks is difficult and it is expected that the network performance will be poor for these activities as compared to other activities which are easier to classify.

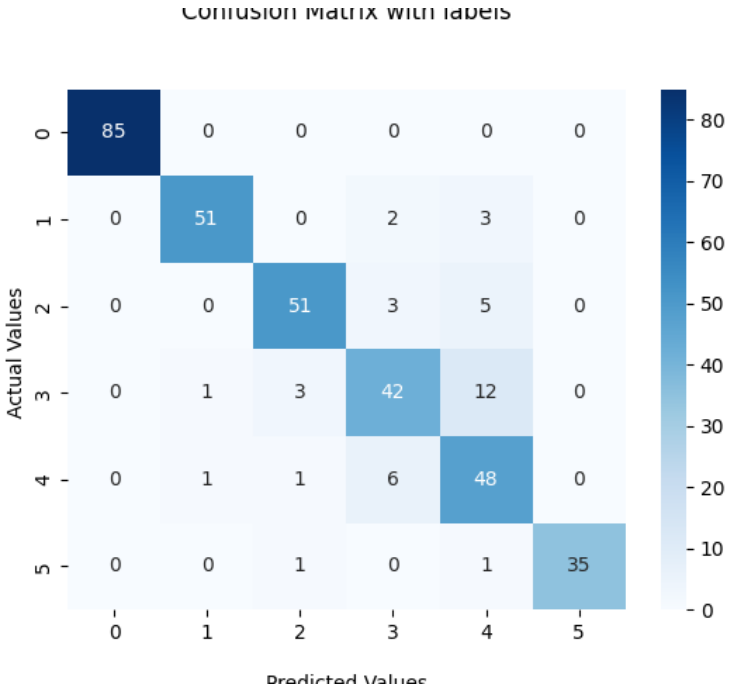

Figure 20: Confusion Matrix for Validation Set.

We also perform some other experiments taking concepts from machine learning theory. The idea is to investigate whether we can improve the results using input feature normalization and some data augmentation techniques. However, we were not successful in improving the results using these experiments.

5.3.1 PointNet with input feature normalization

In the first experiment, we investigated whether input feature normalization may help in classification performance. So, we trained a network where input features were normalized using mean and standard deviation. Figure 21 shows the results. From the figure, we can observe that even though the training accuracy improved, the testing accuracy decreased. This means that mean-standard deviation based feature normalization is not a suitable method for our data. One possible explanation can be that the feature normalization disrupts the structure of the pointcloud condensing it into a smaller range. This, in turn, may cause the data to become unrepresentative causing the neural network to overtrain on the training dataset.

5.3.2 PointNet with data augmentation

In the second experiment, we investigated the effect of data augmentation on the classification results. In this respect, we apply some data augmentation techniques from [9]. Here, three data augmentation techniques were applied: jitter, scaling and shift. Figure 22 shows the results. From the figure, we can see that data augmentation slightly decreased the performance. This result was actually not expected. However, we think that possible reason can be that we directly used the data augmentation paramaters coming from a network that use a spatial pointcloud. We think that further investigations related to what particular data augmentation techniques are essential for our pointcloud dataset may help in improving the results. This can be investigated in future work.

5.3.3 Comparison

Table 1 gives the comparison between the results achieved with different PointNet configurations. We can see that the vanilla PointNet without any feature normalziation and data augmentation gives the best performance for the validation set. The best accuracy that can be reached for the training and validation set is 95% and 88% respectively.

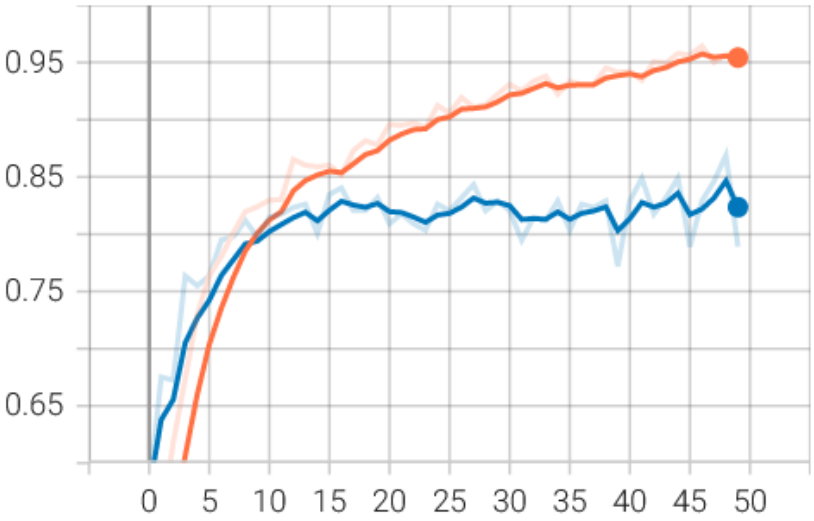

Figure 21: Accuracy on Train and Validation Data for PointNet with Feature Normalization.

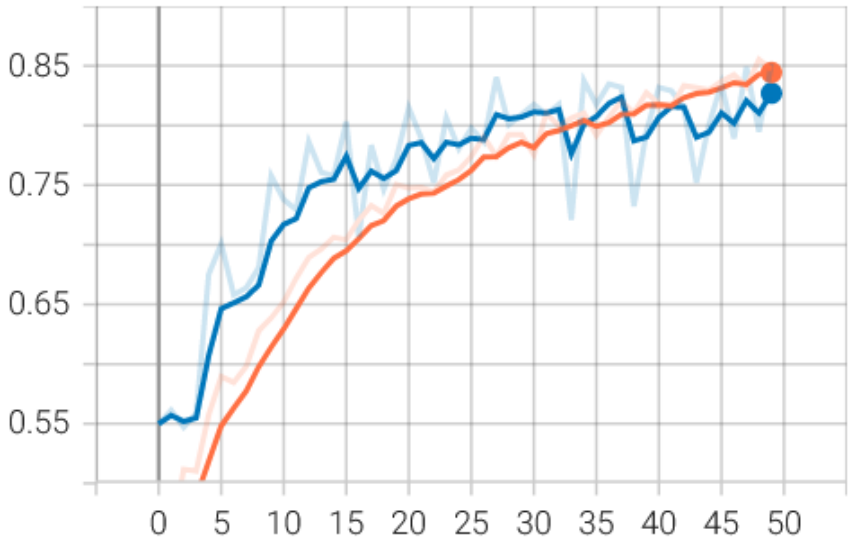

Figure 22: Accuracy on Train and Validation Data for PointNet with Data Augmentation.

Network	Training	Testing
PointNet	94%	89%
PointNet with feature normalization	95%	83%
PointNet with data augmentation	90%	84%

Table 1: Comparison of different Methods.

6 Conclusion

To tackle the problem, 3 different strategies were used: KNN, SVM, and NN. For the first and second ones, only the data processing was not sufficient. Features extraction had also to be performed to obtain the feature matrix and train the model. The first strategy (and the simplest one) had in its best configuration an accuracy of approximately 83%. Although this value was not a terrible result, it could be improved by implementing more sophisticated models.

Then, an SVM model using "One vs. All" strategy for multiclass classification was implemented. Given that the object classification problem consisted of 6 labels, the SVM model had 6 binary classifiers. The general accuracy obtained was around 85%, which was slightly better than the KNN model but still not satisfying.

The third and more complex approach was, then, developed and tested. The PointNet got in its best configuration (without feature normalization and data augmentation) an accuracy of around 89%. This result was better than the KNN and SVM results, therefore this PointNet model was the one selected to be the final model.

In general, all the models had the same difficulty: distinguish activities 4 (pick up an object) and 5 (drink water). If - somehow - a new feature was capable of clearly differentiating these activities, the overall accuracy of all models would increase substantially.

References

D. F. F. Dr Syed Aziz Shah, Dr Julien Le Kernec, "Data sheet for project: Intelligent rf sensing for falls and health prediction," pp. 1–2.

T. D. Bufler and R. M. Narayanan, "Radar classification of indoor targets using support vector machines," *IET Radar, Sonar & Navigation*, vol. 10, no. 8, pp. 1468–1476, 2016.

W. LLC, "Support-vector machine," 2022.

S. Z. Gurbuz, *Deep Neural Network Design for Radar Applications*. SciTech Publishing, 2020.

A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.", 2019.

R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *The Journal of Machine Learning Research*, vol. 5, pp. 101–141, 2004.

C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

K. Vision, "Keras vision pointnet."

C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

D. F. Fioranelli, "Object classification with radar (ee4675) - slides," vol. 6,7,8,9, 2022.