Università Politecnica delle Marche
Ph.D. Course in Information Engineering
Curriculum in Biomedical, Electronics and Telecommunication Engineering

---

Application of algorithms for system performance increasing in ICT

Ph.D. Dissertation of:

DEIVIS DISHA

Thesis Advisor:

Prof. ENNIO GAMBI

Ph.D. Course Coordinator:

Prof. FRANCO CHIARALUCE

XXXV edition - new series

Università Politecnica delle Marche
Ph.D. Course in Information Engineering
Facolty of Information Ingineering
Via Brecce Bianche — 60131 - Ancona, Italy

Acknowledgements

To my beautiful wife Blerina, my son Hansel, my Professor Ennio Gambi and all my Family.

Abstract

Different algorithms used in ICT are described in this dissertation regarding artificial intelligence, deep learning, and genetic algorithms to process signals in communication technologies with the purpose of increasing system performance.

Advances in microwave and millimeter-wave systems have enabled remote sensing techniques previously utilized in long-range applications to be utilized in relatively close-range applications such as detection and categorization of human presence and measurement of human attributes thank to the huge bandwidth and the short time of signal transmission. The design of a WSN homogeneous network with hierarchical structure is demonstrated with the priority of covering an environment with minimal cost, high connection, and maximum longevity.

In the first part will be presented different techniques to process the micro-Doppler signals coming from automotive radars regarding classification and tracking of the objective gained from the information of the target. The second part will present the comparison between machine learning and deep learning algorithm used for human activity classification purpose and getting the best results in terms of performance, accuracy etc. The last part is the demonstration of an enhanced genetic algorithm for improving performance in customizable hierarchical wireless sensor networks choosing weight combination to generate the most performable topology conceivable.

Contents

List of Figures

List of Tables

Chapter 1

Introduction of FMCW radar, ML & DL networks

1.1 Introduction

Advanced Driver Assistance Systems (ADAS) is one of the primary research aims in automotive technology, which has resulted in more sophisticated sensors, one of which is Frequency Modulated Continuous Wave (FMCW) Radars. To perform good measurements and support assistance systems, Radars need huge bandwidth and small Chirp Repetition Time (CRT) but these features can be powerful also to extract micro-Doppler (mD) signals. To classify this signals Modern Machine Learning (ML) algorithms provide a highly strong tool where exists various techniques to extract features and classify signals with varying performances and computational costs.

1.2 MIMO-FMCW radar principle

The key principles underpinning the MIMO-FMCW principle, which is at the heart of the processing chain presented in this study, are briefly described in this section. MIMO radar is based on the homonym idea from communications, in which a collection of M transmitters and N receivers provides $M \, x \, N$ separate propagation channels [1]. The main advantage is that $M + N$ physical components may be used to produce $M \, x \, N$ virtual channels.

In radar, there is a contraposition between distributed and colocated MIMO. The radar cross-section (RCS) of the target for each propagation path in the former may be treated as independent random variables, allowing for improved detection performance and the extraction of additional geometrical information due to spatial variety. In the colocated instance, waveform diversity can be used to improve spatial resolution with large virtual apertures, or array signal processing methods can be used to boost *SNR*. When compared to the standard phased array, which corresponds to the single-input multiple-output (*SIMO*) scenario, a *MIMO* radar may boost the degrees of freedom up to *M* times by employing orthogonal waveforms. Multiplexing the signals in time, frequency, or coding can achieve orthogonality.

1.3 Machine learning and deep learning

The major characteristics of deep learning algorithms are reviewed and compared to standard machine learning approaches in this part, introducing the most significant ideas and techniques with an emphasis on CNN-based image classification structures and their principles. Finally, the applications of such approaches in radar signal processing are discussed. Machine learning is a branch of research that is commonly included in the broader idea of artificial intelligence, which includes computers capable of addressing issues traditionally associated with human intellect. Machine learning, as opposed to rule-based systems, in which a set of instructions is programmed to produce a particular output given a specific input, is a family of algorithms capable of learning general rules to generate predictions or choices from incoming data without being explicitly programmed.

Regression, classification, density estimation, anomaly detection, machine translation, and many more tasks are among them. Significant progress has been made in recent decades, driven by the need for automation of processes and decision making without human interaction across a wide range of applications. Machine learning is frequently divided into two categories: supervised learning and unsupervised learning. In the supervised pattern, the algorithm is trained using samples of an input vector x and an associated label y, and it learns to predict the value y from the input x, often by estimating $p(y|x)$. Classification tasks are an example of an application that supervised algorithms may do. Unsupervised learning approaches, on the other hand, aim to learn the hidden structure from unlabeled data: with an input vector x of data, the algorithm attempts to learn the probability distribution $p(x)$ or certain distribution attributes. This method is commonly used for problems like as density estimation and clustering.

The distinction between these groups is not officially defined, as the borders are occasionally blurred, and certain techniques, such as reinforcement learning or semi-supervised learning, do not fit well into any of them. In any event, all machine learning algorithms and approaches are based on the premise that machines can learn from data in order to better at a specific job, and they typically comprise three components: representation, evaluation, and optimization. The first idea relates to the work of describing the input data in a formal language that the computer can understand, i.e. a collection of characteristics that reflect the input data. The other two concepts deal with how to map the input data represented by a set of features to a desired output: evaluation establishes a loss or scoring function that

objectively defines the model's performance (mean squared error, likelihood), and optimization includes the strategy to achieve the best possible score in the defined objective function (gradient descent, linear programming). In traditional machine learning methodologies, the representation of the input data is generally accomplished by a human in order to give the machine with some type of structure in the data, and the computer then learns from that preprocessed data automatically (Fig. 1). Previous research on both sides, data representation and learning process automation (assessment and optimization), has resulted in machine learning's extraordinary success in a wide range of applications. The representation of the data is therefore an important aspect of the machine learning pipeline, because alternative ways of expressing the input information can have a significant impact on the algorithm's performance for a specific job. In voice recognition, for example, numerous algorithms for feature extraction (and combinations of such features) have been presented during the last decades, some of which are better suited for certain tasks (chroma vector, spectral rolloff, etc) [2]. The similar problem occurs in computer vision, where a number of feature extraction approaches (histogram of directed gradients, scale-invariant feature transform, etc.) [3] are commonly used for natural picture categorization. In this vein, the process of selecting and/or designing the features that best represent or model the input data for a specific application (representation) is a task that necessitates extensive specific-domain knowledge, not only in a specific field of study (e.g., audio signal processing), but also in very specific applications (e.g. text-independent speaker recognition, unstructured audio classification for environment recognition

etc.). Furthermore, when working with complicated data, underlying correlations or interactions between variables within the data are not always obvious, and human feature engineering is frequently done by trial and error. As a result, it is a time-consuming procedure that is frequently far from optimal.

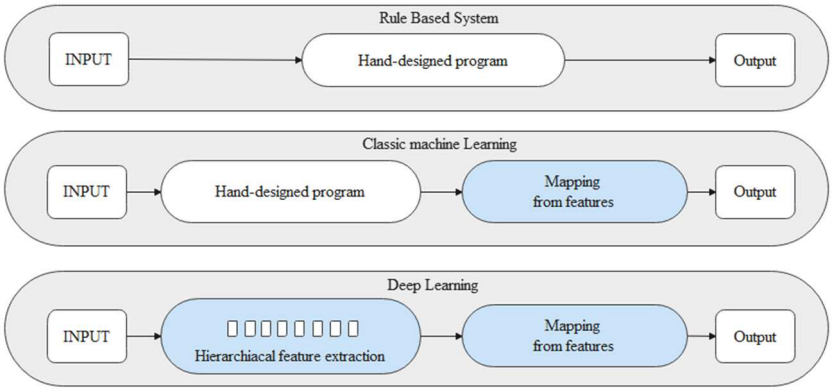


Figure 1. Approaches in AI systems. Blue boxes indicate components that learn from data.

In this context, deep learning, a subset of machine learning techniques, entails teaching the computer not only the mapping between the feature space and the output, but also how to represent the input data as a set of features. In this manner, the algorithm seeks the best representation of the input data in order to obtain the optimum performance for a certain application or job. The learned features are optimum in the sense that they strive to minimize an objective function for a specific application by including the representation step into the optimization problem. As a result,

these algorithms are especially well suited for unstructured data, such as photos, audio, or text, where there are a large number of sources of variation, making it challenging to automatically extract high-level abstract characteristics that appropriately describe the data. Deep learning is believed to be scalable across domains as compared to other techniques focused on feature design since, in theory, no substantial specific-domain expertise is required.

The competence required in a specific area to create a good model is replaced by the requirement of a large enough data collection to extract generic characteristics from raw input data. In this sense, there is a paradigm shift from model-driven to data-driven techniques, where highly complicated characteristics may be learnt automatically given enough data. One of the important aspects of deep learning approaches is how they learn how to represent complicated notions. The technique is founded on the concept of a hierarchical representation, in which complicated properties are described in terms of simpler ones. Higher layers in the hierarchy reflect higher degrees of abstraction. Learning may be viewed as an inductive process in which a group of specific instances is examined in order to construct a general representation by extracting successive features of increasing complexity at different levels of abstraction. Deep architectures may represent more complicated concepts because they can integrate numerous intermediary characteristics hierarchically through different degrees of abstraction, as indicated by the number of these intermediate representations or layers. Tasks connected to visual perception, such as recognizing or categorizing items in a picture, are typical instances of these notions. A picture contains a wealth of information

in the form of color, textures, and spatial relationships between neighboring regions, but the input data is displayed in 2-D as a collection of raw pixels. Lower layers of a CNN may train to recognize fundamental patterns based on geometrical primitives and basic features such as edges, blobs, or color information, while intermediate layers combine these elements to generate more sophisticated patterns such as corners, contours, or textures. As input flows to higher layers, the network learns more abstract representations, such as object components with distinguishing characteristics. In the case of recognizing a car in an image, the network would learn to represent the wheels or the contour of the chassis by combining the intermediate representations, until the upper layers decide that the detected parts of the image are consistent with the idea of a car, based on previous experience gained by observing a number of images of cars during training. The ability to acquire big abstract concepts hierarchically from smaller components is a fundamental aspect of the human brain's learning process and one of the reasons for such algorithms' strong capacity for dealing with complicated representations. As a result, deep learning has been effectively implemented in a wide range of applications with extremely favorable outcomes, beating earlier techniques in machine learning in many situations. Deep learning has gotten a lot of attention from researchers in recent years, which has resulted in a wide range of solutions based on various architectures and techniques. In this section, we will go through the fundamental concepts of artificial neural networks and a similar design, a convolutional neural network, which will be used in the context of this study to classify radar mD images.

1.3.1 Artificial Neural Networks

Artificial neural networks (ANN's) are computing systems build with a set of interconnected nodes that process information responding in a dinamic way to external inputs. In these systems, information can flow from the input to the output with connections just in one way, or the nodes might have feedback connections or loops between them. The former are known as feedforward networks, while the latter are known as recurrent neural networks and will not be investigated further in this work since they are suitable for temporal series and other applications that are outside the scope of this research. The goal of feedforward networks is to estimate a function *g* in order to obtain the required response *y* from a collection of inputs *x*, such that $y = g(x)$. This is accomplished by merging a number of fundamental units known as perceptrons, which execute simple operations. The perceptron is sometimes viewed as a simplified model of a neuron that, in essence, creates a binary output from an input vector *x* and a set of learning parameters, notably weights *w* and bias *b*, as follows:

$$g_0(x) = \begin{cases} 0 \;\; if \; x^T w + b \leq 0 \\ 1 \;\; if \; x^T w + b > 0 \end{cases} \tag{1}$$

Because the perceptron's operation is linear, a nonlinear activation function $f(z)$ is added at the output to mimic complex functions. The logistic function, hyperbolic tangent, and rectified linear units are examples of activation functions (*ReLU*). Non-linear functions can therefore be approximated by merging many neurons in various layers. Indeed, multilayer feedforward networks are referred to as universal approximators because they may

approximate any function of any complexity [4], [5]. This is a significant observation since it demonstrates the strength of neural network representation. The origins of *ANNs* are claimed to be biological neural networks. Although biological and modern artificial neural networks have considerable inherent differences, there are certain commonalities that might be exploited to build specific comparisons between the two ideas. For example, in terms of the dynamics of individual neurons, a certain parallelism (synapses and weights, axons and element outputs, etc.) may be established between the elements, as can similarities in the network structure in terms of neural connection. However, there are significant distinctions between biological and artificial neural networks that necessitate a rigorous comparison. In any event, the concept of a biologically inspired artificial neural network was first established in 1943, when Warren McCulloch, a neurophysiologist, and Walter Pitts, a mathematician, built a rudimentary neural network using electrical circuits while researching the functioning of neurons in the brain.

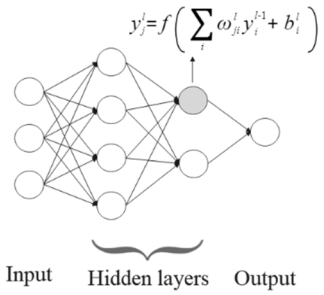


Figure 2. Schematic representation of a neural network

The output of the j-th neuron at the layer l is the function f (activation function) of the outputs of the layer $l - 1$ and the weights and bias of the layer l. The layer l has j hidden units, while layer l -1 has i hidden units. The notion of a perceptron and its training was presented in the 1950s, followed by the first successful attempts at applying neural networks to real engineering issues, which drew significant attention and financing [6]. However, development in AI came to a halt at the end of the 1960's when the perceptron technique was challenged for being incapable of scaling to multilayer neural networks. The notion of using backpropagation to train multilayer neural networks rekindled interest in neural networks in the 1980's, among other things. The first allusions to the notion of CNN's may be found in a work by Fukuhsima [7] from that time period, in which an architecture for visual pattern recognition was presented with comparable properties to contemporary deep CNNs. The so-called second wave of neural network research lasted until the 1990s, when it was supplanted by interest in other more effective approaches such as support vector machines [8]. However, significant advances were made during this time, such as the use of stochastic gradient descent and backpropagation to train deep networks, as described in a paper by Yann LeCun et al. Around 2006, Hinton proposed the concept of deep belief networks and unsupervised pre-training, ushering in the third wave of neural network research [9]. Around that time, the idea of deep learning was established to stress the relevance of depth in neural networks. The design developed by Krizhevsky et al. [10], which won the Imagenet Large Scale Visual Recognition Challenge in 2012, was the turning point that

finally drew significant attention to neural networks and deep learning. The suggested design made use of graphic processing units (GPU's) to train the model, as well as two approaches that are standard building elements in contemporary CNN architectures, namely dropout and *ReLU's*. Deep learning approaches have subsequently gained popularity and are the subject of extensive study in a wide range of applications such as computer vision, natural language processing, and so on. The causes for the current proliferation of these approaches are most likely due to the abundance of data in the form of high-quality labelled datasets, as well as access to a massive amount of unlabeled data from various sources. Parallel to this, the rise in processing capacity, particularly the ability to parallelize computing with GPU's, has resulted in deeper and more complicated architectures that can be trained with very large data sets. Furthermore, the popularity of deep learning has garnered significant attention and resources from academic and industry sectors, resulting in a large community of researchers, software platforms, and constant algorithm and technique innovation.

1.3.2 Convolutional Neural Networks

Convolutional neural networks are a type of neural network that has one or more convolutional layers to exploit the 2-D structure (or greater dimensionality) of the input data, such as pictures, audio signals, and so on. Because of their ability to recognize patterns and express complex ideas, these architectures have received a lot of attention in the field of computer vision. CNN-based architectures are regarded state-of-the-art in applications such as image classification [10], [11], semantic segmentation [12], object detection

[13], and so on. Another major area of research is the application in natural language processing tasks such as machine translation [14], speech recognition [15], and many others. Other fields have also taken use of the possibilities of these algorithms for various objectives such as drug discovery [16], recommender systems [17], market prediction [18], and so on. CNN's are inspired by the mammalian visual cortex, where specific cells are sensitive to partly overlapping sub-regions that compose the vision field. While simpler cells operate as filters to identify specific patterns in such places, more sophisticated cells provide higher order responses, resulting in the visual perception [19]. In contrast to CNN's, the input data of a fully connected neural network is provided as a vector that is converted through a succession of layers, each of which has a collection of components that are connected to the output of the preceding layers. In the case of a picture as input, each pixel has a weight, and the succeeding levels of the architecture are linked to all of the preceding layer's parameters. When dealing with 2-D data, such as pictures, the fully connected technique has the drawback that the number of parameters scales quadratically with the size of the input and it ignores the property of local spatial correlation of images. As a result, it is impossible to train such structures with several hidden layers.

Convolutional neural networks, on the other hand, are founded on two fundamental concepts: local connection and shared parameters. Convolutional layers are created by combining feature maps that cover a certain receptive field. The spatial correlation within a particular region may be used by establishing locally linked layers, so that the input of a neuron in a given layer is derived from a subset of units with contiguous receptive fields

from the preceding layer. As a result, neighboring areas that produce separate patterns can be identified.

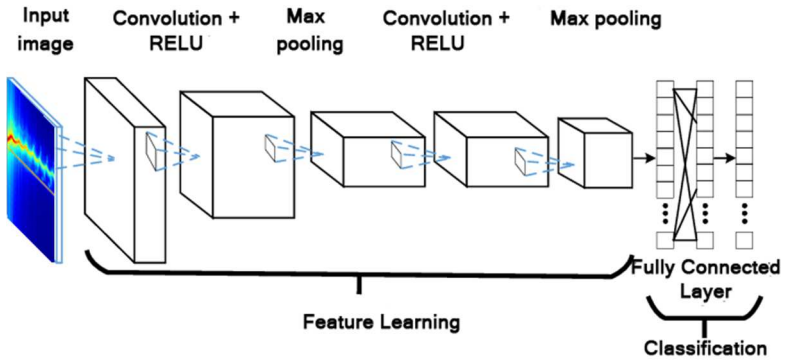


Figure 3. Schematic representation of a CNN with convolution, pooling and fully connected layers (activation functions are omitted in the representation)

The parameters of the convolutional layers, on the other hand, are shared over the full visual field, based on the assumption of stationarity, i.e. the picture statistics are location-invariant. That is, the convolutional layers learnt kernels can recognize patterns that exist at various places in the picture. These qualities allow for a significant reduction in the number of network parameters as compared to a fully connected neural network, enhancing learning efficiency and improving generalization as the number of degrees of freedom is lowered. For example, a neural network with *40K* hidden units and an input picture of 200 *x* 200 pixels would have roughly 2 billion parameters, but a convolutional neural network with feature maps of 10 *x* 10 would only need 4 million parameters for the same input.

1.3.3 Architecture of a CNN

CNNs are extremely adaptable designs that can exhibit a variety of topologies based on a variety of parameters like as the application, the nature of the input data, and many more. Furthermore, as the field progresses, new designs and methodologies are presented on a regular basis, with further variants. The next section reviews the fundamentals of a standard CNN architecture for image classification, which is the application that will be explored in this study. A typical CNN design has the shape seen in Fig. 3 and is often built of numerous construction components, which we will briefly introduce:

- Convolutional layers:

Convolutional filters are the foundation of CNN's. The goal is to recognize certain patterns using a collection of learnable kernels. The filters cover a certain region (receptive field) and span into a third dimension known as depth. A bank of filters is employed in a convolutional layer to identify as many features as the number of kernels. The filter bank is therefore a 4-D tensor of size $S\ x\ S\ x\ D\ x\ N_l$, that is, a bank of N_l filters of kernel size $S\ x\ S$ and depth D. The depth dimension must be the same as the preceding convolutional layer's number of features, N_{l-1}. When working with a color picture (RGB) as input data, for example, the first convolutional layer has a depth of three to correspond to the number of color channels. The size O of the output map on each picture dimension may be described in terms of the convolution's $z_{padding}$ and stride L_{stride} as, where P is the input dimension size.

$$0 = \frac{P - S + 2Zpadding}{L_{stride}} + 1 \quad (2)$$

If the convolution parameters are set so that the output is the same size as the input map *P x Q*, the output value of the feature map of a particular feature n_l at position p, q in the $l - th$ layer may be written as :

$$z_{P,q}^{n_l} = \sum_{d=0}^{D} \sum_{i=0}^{S} \sum_{j=0}^{S} w_{i,j}^{n_l} y_{d,(p+1),(q+j)}^{l-1} + b^{n_l} \quad (3)$$

where $w_{i,j}^{n_l}$ represent the weights of the kernel and b^{n_l} is the bias. Convolutional layers with common parameters exhibit translation equivariance, which means that translating the input features results in an equivalent translation of the output.

- Non-linear activation function:

In order to describe non-linear functions in a neural network, a non-linear activation function is applied at the output of the neurons. Because the convolution is a linear operator, the same logic applies, and an activation function *f(·)* is applied to the convolution's output:

$$y_{p,q}^{l} = f\left(z_{p,q}^{l}\right) \quad (4)$$

In order to estimate the gradient of the cost function via backpropagation, the activation function must be differentiable. Different activation functions can

be employed in neural networks and CNN's. Two activation functions that will be utilized are: the sigmoid function

$$f(z) = \frac{1}{1 + exp(-z)} \tag{5}$$

And the Rectified Linear Unit (*ReLU*)

$$f(z) = max(0, z) \tag{6}$$

ReLU is presently the default suggestion in deep learning architectures owing to its enhanced convergence outcomes and lower processing cost when compared to other functions such as sigmoid or hyperbolic tangent [10].

- Pooling layers:

Pooling layers are added after the activation functions to lower the spatial size of the activation maps (i.e., the feature maps after non-linear activation). A pooling layer of size s_p separates the activation maps into $s_p \; x \; s_p$ areas and extracts a specific statistic from each. Such statistics include average pooling, L-2 norm pooling, and simply choosing the highest value (max-pooling). As a result, the number of parameters in the following layer is decreased by a factor of about s_p, resulting in a significant reduction in computing effort. As the picture proceeds through successive convolutional layers and pooling processes, semantic information is retrieved at the price of spatial information, i.e. the spatial resolution of the activation maps is reduced. This is a useful quality in classification jobs, as it ensures that the absolute location

of the features in an image does not affect the classification's result. When recognizing an automobile in a picture, for example, the classifier should be able to recognize it whether it is in the center or a corner of the image. In this sense, pooling layers make the feature representation insensitive to tiny translations, because the result of the pooling operation is the same as long as the detected feature is within the pooling region.

- Fully connected layers:

According to the concept depicted in Fig. 1, the convolutional and pooling layers of a CNN for classification tasks may be viewed as feature extractors, whereas the top layers conduct the mapping from the feature space to the output scores. Although any classifier, such as an SVM [20], can be utilized, a multilayer perceptron is frequently employed since it integrates nicely with a CNN for end-to-end training with many classes. It is commonly referred to as a Fully Connected (*FC*) layer in this context. The high level information retrieved in the convolutional layers in the form of an input volume are translated into a vector of features that is utilized to conduct classification in *FC* layers. This is accomplished by linking all of the activations in the preceding layers with the nodes in the *FC* layer in one or more intermediary stages, so that the number of nodes at the output equals the number of classes in the classification. The high level features are thus mapped to the output classes via a set of learnable weights in the multilayer perceptron. The softmax function (also known as multinomial logistic regression) generalizes the logistic function to K classes and maps input data to output values within a range of [0 ; 1], thus the softmax regression scores,

or equivalently, the output scores for each class, may be understood as probabilities:

$$\begin{pmatrix} P(y=1|x;w) \\ . \\ . \\ . \\ P(y=K|x;w) \end{pmatrix} = \frac{1}{\sum_{k=1}^{K} \exp\,(w^T x)} \begin{pmatrix} \exp\,(w_1^T x) \\ . \\ . \\ . \\ \exp\,(w_K^T x) \end{pmatrix} \tag{7}$$

The cross-entropy loss is the loss function associated with the softmax function:

$$J(w) = -\frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} (y^m == k)\, log \frac{exp^{(w_k^T x)}}{\Sigma_{i=1}^{K} exp(w_i^T x)} \tag{8}$$

where M is the number of samples and (y^m== k) equals 1 if the m-th training sample belongs to class k and 0 otherwise.

- Training a CNN:

The supervised neural network training concept consists of changing the network's parameters w to minimize the cost function $J(w)$ given a training data set with M labelled samples $\{(x^1 y^1)\,, \ldots (x^M y^M)\}$, where x^m represents the input features of dimension *N* of the *m-th* sample, and $y^m \epsilon \{1,2, \ldots, K\}$ the labels. This concept extends to CNN's with no loss of generality. The kernels of the convolutional layers and the weights of the fully connected layers are the learnable parameters in a CNN. The standard technique for locating local minima of the cost function is based on gradient methods [21], in which the

gradient of the cost function $\nabla_w J\ (w)$ with respect to the network parameters is computed, and then these parameters are updated in the opposite direction. There are several methods for estimating the gradient of the cost function. On the one hand, batch gradient descent estimates the gradient using the entire training set, whereas stochastic gradient descent approximates it iteratively with each individual training set. While the former requires enough memory to retain the data set and can be quite sluggish depending on the size of the set, the latter is faster but the objective function fluctuates significantly due to frequent updates. As a result, an intermediate solution, in which the gradient is approximated using a tiny batch of training data (hence the name minibatch gradient descent), is usually selected. Because the variation in the parameter update is decreased, this strategy becomes more efficient by employing matrix multiplication and produces more steady convergence. The parameters of mini-batch gradient descent are updated as follows:

$$w = w - \alpha \nabla_w J\left(w; x^{(m:m+n_{batch})}, y^{(m:m+n_{batch})}\right) \tag{9}$$

It is necessary to define two hyperparameters: the batch size n_{batch} and the learning rate α. The batch size is determined by the application and architecture. Batch sizes are typically between 32 and 256. Because it defines the magnitude of the step in the direction of the negative gradient, the learning rate is an important hyperparameter. A high learning rate might cause missed local minima and failure to attain convergence, whereas a low number increases the runtime until convergence. The momentum approach is typically used to accelerate convergence and eliminate oscillations around a local minimum, quickening convergence by adding a portion of the value

from the previous step to the update vector [22]. Two sets of parameters, the weights and the bias, must be learned in a neural network. In the *l*-th layer the weights are represented by a matrix W^l and the bias b^l. Updated parameters will be :

$$W_{j,i}^{l} = W_{j,i}^{l} - \alpha \frac{\partial}{\partial W_{j,i}^{l}} J\left(W, b\,;\, x^{(m:m+n_{batch})}, y^{(m:m+n_{batch})}\right) \tag{10}$$

$$b_{j}^{l} = b_{j}^{l} - \alpha \frac{\partial}{\partial b_{j}^{l}} J\left(W, b\,;\, x^{(m:m+n_{batch})}, y^{(m:m+n_{batch})}\right) \tag{11}$$

- Stochastic gradient descent

The gradients in equations (10) and (11) must be approximated in order to update the parameters. Backpropagation, an efficient approach for computing the gradient of the cost function with respect to the network parameters, is generally used in a neural network [23]. The procedure is divided into two steps: first, the cost function is evaluated in a forward pass of the input values through the network with randomly set parameters (the initialization of the weights might affect training convergence). The gradient is calculated in the second step by propagating the mistake from the output layer backwards. Using the chain rule, the goal is to analyze each node in the network's contribution to the final mistake. The intermediate values corresponding to each node's activations are saved during the forward pass. The activation of the *j-th* node in the *l-th* layer is provided by :

$$y_j^l = f\left(z_j^l\right) = f\left(\Sigma \omega_{ji}^l y_i^{l-1} + b_j^l\right) \tag{12}$$

The cost function is evaluated after a complete forward pass, and the contribution to the error created by the output nodes (layer L) is determined. This may be written as the ratio of the change in the cost function as a function of the output activations:

$$\delta_j^L = \frac{\partial}{\partial y_j^L} J(W, b) f'\left(z_j^L\right) \tag{13}$$

or equivalently:

$$\delta^L = \nabla_y J \odot f^1\left(z^{(L)}\right) \tag{14}$$

Where $\odot$ signifies the Hadamard product or element-by-element multiplication. Similarly, the contribution to the error created in layer L-1 is calculated using δ^L and the weights in that layer. In general, the error at the layer l is represented as a function of the error determined in the next layer (l + 1) and its weights:

$$\delta^l = [(W^{l+1})^T \delta^{l+1}] \odot f'(z^l) \tag{15}$$

As a result, the partial derivatives of the cost function with respect to the parameters in any layer l can be defined in terms of the activations in layer l - 1 and the previously computed error δ^l :

$$\frac{\partial}{\partial W_{ij}^{l}} J(W, b; x, y) = y_i^{l-1} \delta_j^{l} \tag{16}$$

As a result, the error in the layer l is a function of the errors in the layers l+1, l+2, ... L, which are assessed in turn using the chain rule by combining the equations (15) and (16). After executing backpropagation with the training samples of one batch, mini-batch stochastic gradient descent estimates the gradient and updates the network parameters. The batches are created by splitting the dataset at random. An epoch of training is defined as a full pass of all the samples in the training set, separated into batches. The network is trained for as many epochs as necessary to achieve convergence.

1.3.4 CNN in radar signal processing

Due to the requirement of analyzing incoming radar echoes to extract useful information, the fields of machine learning and radar processing have a significant overlap. Machine learning applications in remote sensing have already been the subject of much study [24]. Statistical signal processing is a clear illustration of how some of the techniques available in the machine learning framework may be used. For example, detecting radar targets in clutter is a well-known challenge in the field, in which specific statistics from an otherwise unknown distribution (e.g., marine clutter) are extracted to produce an acceptable false alarm rate. In this regard, machine learning offers relevant techniques for learning the statistical distribution from data [25], [26]. In general, the theoretical foundation of machine learning in its traditional meaning has been applied to a variety of subdisciplines and

applications in the radar field, including cognitive radar [27], automated target identification [28], parameter estimation [29], and many more. As a result of the recent success of deep learning techniques, advancement in this area is also infiltrating the radar field, with an increasing variety of applications and approaches. To provide a brief overview of recent development in this field, we split the challenges in radar signal processing that employ deep learning algorithms into three categories, as stated in [30]: sensing, processing, and recognition. We concentrate on the most recent group because it is the topic of this work and constitutes the vast bulk of research at the intersection of deep learning [44] and radar. Sensing refers to the techniques utilized to obtain data at a fundamental level, and deep learning is employed to optimize this process. Although the use of machine learning or deep learning models in this context is not easy, certain techniques in the area of waveform design and cognitive radar have been presented [31], [32]. The second category includes issues involving parameter estimates, detection, and nonlinear modeling. Due to practical constraints, several detection and estimation issues in radar are frequently tackled as optimization problems under the assumption of a linear forward model. Because neural networks are universal approximators, they are a good solution for handling non-linear and inverse issues. Problems with SAR imaging [30], modeling multipath effects or interference [33], and clutter [26] are some instances in this category. However, the primary use of deep learning techniques in radar is centered on detection and classification challenges. As previously said, one of the main notions of deep learning [45] is its capacity to automatically extract characteristics from data, which is a significant benefit in the context

of radar. For feature engineering, traditional machine learning algorithms rely on domain expertise. In the context of feature extraction from natural photos, for example, manual feature engineering necessitates prior knowledge and skill in order to select the image attributes that contain the most discriminative features. With sufficient prior training, a human operator can naturally interpret these photos and impose some prior knowledge based on human vision and extract discriminative features. However, manual feature extraction is particularly difficult in the classification of radar pictures in particular, or signals in general, because the data format is difficult to grasp or counter-intuitive. As a result, automatic feature extraction with the added benefit of hierarchical representation is an appropriate strategy for dealing with data provided in a manner that may be difficult for a person to comprehend. Deep learning for automated recognition in SAR imaging is one of the most obvious but intriguing applications.

In [34], a CNN trained on the MSTAR dataset (Moving and stationary target acquisition and identification) achieves a 99 percent average accuracy in the classification of objects on SAR pictures. [35] describes a natural extension based on polarimetric SAR. A deep learning strategy for ISAR picture classification has also been proposed, similar to the work published in [36], which employs an unsupervised approach based on autoencoders to extract features. Another application that has received a lot of interest is the use of deep learning to classify micro-Doppler (mD) signatures. Additional Doppler shifts caused by vibration, rotation, or movement of non-rigid parts have been shown to convey relevant information for automatic target recognition [37]. Given the difficulty of obtaining features from the mD

spectrogram, deep learning offers an intriguing approach. Several human activities, for example, are classified in [38] by training a CNN with the relevant mD characteristics. In addition, the authors of [39] provide a CNN pre-trained with an unsupervised technique to differentiate distinct patterns of human gait that are hardly visible in mD spectrograms. Other CNN-based applications include mD signature classification of hand movements in a human-machine interaction [69], classification of UAVs from spinning propeller mD signatures [70], [71], and signature extraction using deep learning-based segmentation [63]. Automotive radar is one area that is projected to benefit from deep learning, because to the desire in better driver aid systems (ADAS). Radar technology, in addition to laser and optical sensors, is regarded as a significant component of these systems. Aside from range and velocity, deep learning algorithms can extract more sophisticated information in order to generate a semantic picture of the vehicle's surroundings. For example, CNN's may be used to classify static items on the street using gridmap representations [64], recurrent neural networks can be used to distinguish between stationary and moving targets [65], and fully connected networks can be used to identify ghost targets [66]. Detection and classification of people, cyclists, and animals that may occur in the vehicle's vicinity has also been proposed utilizing CNN's and autoencoders [67]. The scarcity of labelled data to train deep models is a fundamental restriction in the implementation of deep learning methods in radar. Data-driven techniques need a large collection of training data in order to avoid overfitting (learning specific properties of the training set) and generalize appropriately to new test data. For example, in the field of computer vision, a massive quantity of

labeled data is accessible in the form of standardized datasets for various applications such as classification, semantic segmentation, and so on. In radar, however, there is no such thing, and in order to train somewhat complicated models, a large enough amount of labelled training data for a specific application is required, which is frequently expensive in terms of time and labor. As a result, the models employed in radar image classification are not as sophisticated as those used in other fields in terms of network complexity and depth. Despite these constraints, there are various ways for reducing the effect of overfitting when there is a scarcity of data. In [48], for example, an unsupervised pre-training method based on convolutional autoencoders is presented in the context of classification of human gait using mD signals. They employ a limited dataset to initialize the network parameters, resulting in higher convergence. In [49], the authors apply transfer learning to consider mD signatures by pre-training a model using a huge database of natural photos and fine-tuning the parameters with a smaller collection of mD images. When the training set is small, it is demonstrated that such an initiation produces better results than starting from scratch. Another approach for providing regularization is data augmentation, which involves enriching the training set artificially by executing transformations on the input data that are compatible with hypothetical transformations present in the test data. In [68], for example, the authors guarantee translation invariance in the model by inserting translations on the MSTAR dataset training data for SAR image classification. While an effort is required to develop standardized data sets in the context of radar, its feasibility is restricted due to the wide range of hardware and preprocessing methodologies

available depending on the application. As a result, when training complex deep learning models for radar applications, the effort should be focused not only on the data, but also on the algorithm side, in order to establish a framework that allows for the creation of complex but still general models that can be applied in the radar field. Although integrating deep learning techniques from other fields such as computer vision is a good starting point, further study is needed to adapt these models to the nature of radar data. In this vein, incorporating strong priors into the model can help to simplify the optimization process and decrease the need for labelled training data. By incorporating domain-specific and application-specific bias, such priors may be included on two separate planes. The former relates to information gained from a particular field, in this example, the qualities of radar signals.

Chapter 2

Artificial Intelligence in radiofrequency signals

In Chapter 1, we learnt that the mD is an effect that is independent of radar technology and that it may be extracted using a variety of signal processing techniques. [40, 41, 42, 43] provide examples of how mD may be utilized with Ultra-Wide Band (UWB) and CW radars. The mD in mm Wave FMCW radars may be extracted in two methods, which will be explored in this chapter. Modern Machine Learning (ML) techniques provide a highly strong tool for a classification issue; there are various approaches to extract features and categorize signals with varying performances and computational costs.

This chapter will show several pipelines that have been developed and will evaluate their effectiveness and performance in various cases.

2.1 Pre-processing FMCW signals

In Chapter 1, we discussed MIMO-FMCW radar princliple and how the radar system transmits this data to a computer. Because the sensor has four receivers, the beat signals are limited to four and are referenced to the device's MIMO arrangement. Only one beat signal can be utilized to analyze and identify the mD, but by adding the four-receiver lines, we can enhance the signal-to-noise ratio and get better results. Following that, the samples are rearranged into a complicated matrix, with samples of a single chirp stored

along the rows and samples of distinct chirps saved along the columns. The obtained matrix comprises the global acquisition samples.

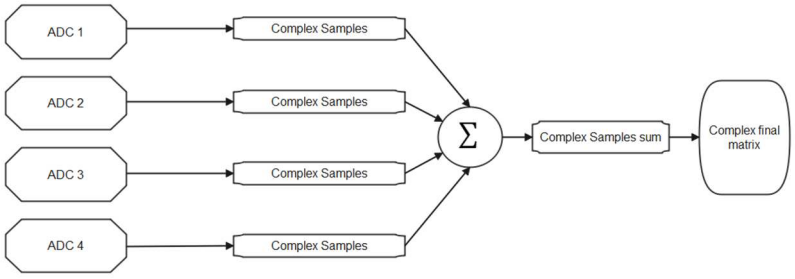


Figure 4. Pipeline used to obtain the initial complex matrix

Two types of matrices are used to store the original mD information. In the first the information is compressed along the time wich is called range-Doppler map and the other along the range where we extract the information with a Short Time Fourier Transform (STFT) and is stored in a Doppler-time map. We can use both of them for classification problems. A bi-dimensional FFT is used to calculate the range-Doppler map. This process is shown in the image below:

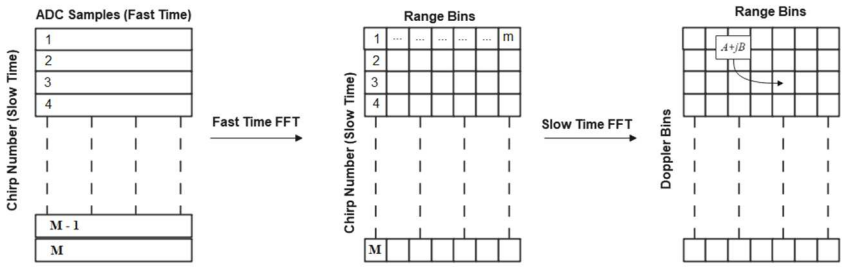


Figure 5. Calculation process of the range-Doppler map

The equation of the mathematical model is:

$$R_D = \left| \sum_{m=0}^{M} \sum_{n=0}^{N} r[m,n] * exp\left(-j2\,\pi \left(\frac{mk}{M}\right)\left(\frac{nl}{N}\right)\right)\right| \tag{17}$$

where *r[m,n]* are the elements of the data matrix, *m* and *k* are the indexes in range from 0 to *M* – 1 and *n* and *l* indexes from 0 to *N* -1, being *M* the number of fast time elements, while *N* is the number of slow time elements. The technique is slightly different in the second map, but the beginning complex matrix is the same. To obtain the Doppler-time map, the fast-time axis must be along the columns and the slow-time axis must be along the rows, and the first step is to execute a FFT along the fast-time. This technique generates a range-time map, and an example is shown in Fig. 6. From this map, a STFT, the most popular time-frequency representation [64], may be calculated along the slow-time. The equation given for the STFT is:

$$STFT_x[k,n] = \sum_{r=-\infty}^{+\infty} x[r]w[n-r]e^{-j2\pi rk/N}, k = 0,1,\ldots,N-1 \tag{18}$$

where n represents a discrete index of time, $w[\cdot]$ is a window function and *k* the discrete index of frequency. In reality, the STFT is the Fourier transform of a signal amplified by a sliding window over time. A trade-off between time and frequency resolution must be determined, and overlapping the windows can help as this procedure attempts to lengthen the duration to increase

frequency resolution [65]. Fig. 7 depicts the procedure for obtaining this second sort of map. The STFT is done in each row of the range-time map, and the resulting matrix represents the gathered mD at each distance. We derive the Doppler-time map of the global acquisition by adding the acquired matrices. Both range-Doppler and Doppler-time maps are constructed of complex numbers, and we must compute the module to produce a "image" from which we may extract features. Each categorization process will be carried out in figures that simply display module information. Fig. 8a and 8b show some examples of these outcomes for an awake individual. We may now describe how to categorize mD signals based on this starting point.

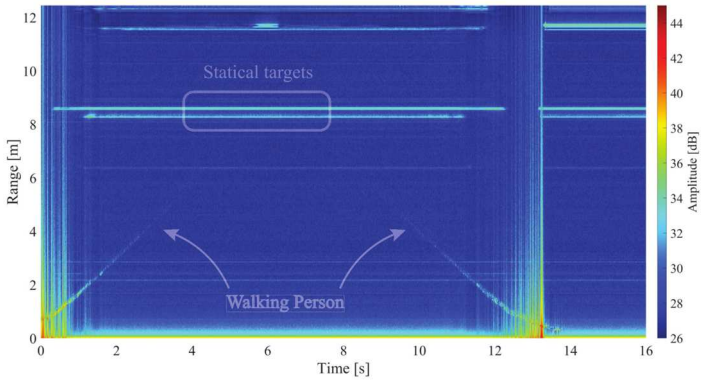


Figure 6: Example of range-time map for a walking person

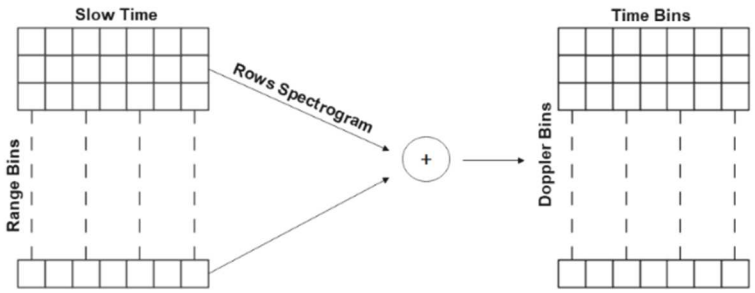
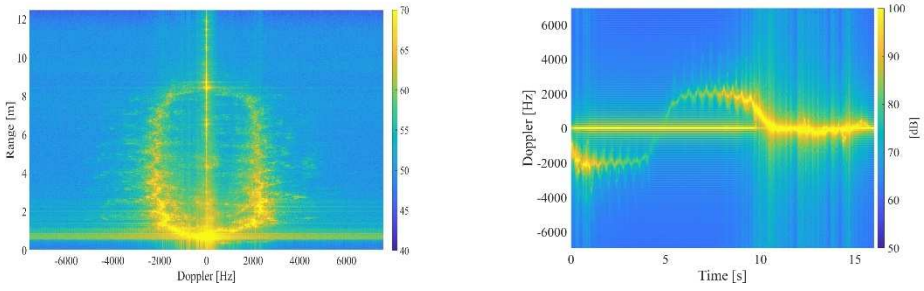


Figure 7: Calculation process of the Doppler-time map

2.2 Classification Framework

In this section, we discuss the initial pipeline that was developed, which is based on dimension reduction techniques, as well as the classification algorithms that were utilized to discriminate between the acquisitions under consideration. In terms of feature extraction, we use two alternative approaches to minimize data dimensionality: Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE). Both maps created from radar signal processing will be referred to as amplitude images. We get a vector by using dimensionality reduction algorithms to these pictures, that is, the principal components derived from PCA and the major dimensions supplied by the-SNE will serve as features vectors.



(a) (b)

Figure 8: Example of (a) range-Doppler map and (b) Doppler-time map

For example, suppose we have a collection of N photos. I_n of dimension [l x m], with n = 1...,N, are initially vectorized row-wise and grouped to generate a training set $X = [x^{(1)},..., x^{(N)}]^T$, where T is the transposition operator; rows of X correspond to observations and columns to variables. Each row's j-th element is then normalized using the following equation:

$$\bar{x}_j = x_j - \mu_j = x_j - \frac{1}{N}\sum_{n=1}^{N} x_j^{(n)} \tag{19}$$

2.2.1 Principal Component Analysis (PCA)

Principal component analysis (PCA) [46, 47] is a multivariate approach that examines a data table in which occurrences are represented by numerous quantitative dependent variables that are inter-correlated. Its purpose is to extract the significant information from statistical data and express it as a set of new orthogonal variables called principal components, as well as to display the pattern of similarity between observations and

variables as spots on spot maps. PCA is mathematically based on the eigen-decomposition of positive semi-definite matrices and the singular value decomposition (SVD) of rectangular matrices. Eigenvectors and eigenvalues determine it. Eigenvectors and eigenvalues are integers and vectors that are related with square matrices, respectively. They form the eigen-decomposition of a matrix, which examines its structure, such as correlation, covariance, or cross-product matrices.

In practice, doing PCA is pretty straightforward. Create a *m x n* matrix from a data collection, where *m* is the number of measurement kinds and *n* is the number of trials. Subtract the mean from each measurement type, or row x_i. Calculate the SVD or the covariance eigenvectors. It was discovered that there were numerous fascinating applications of PCA, among which multivariate data analysis and picture compression are utilized alternately in everyday life, deliberately or unwittingly.

2.2.2 Description of the principal components method

The primary goal of the principal component analysis is to reduce the size of the observation space in which provided items are analyzed. The reduction is achieved by generating new linear combinations of variables that characterize the items under consideration. These combinations, known as primary components, must meet specific mathematical and statistical requirements. The principal components technique begins with an observation matrix *X*, in which column vectors indicate observations that characterize an item in relation to random variables $X_1, X_2, \ldots, X_p$.

$$X = \begin{bmatrix} x_{11} & x_{22} & \cdots & x_{1n} \\ x_{21} & x_{12} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{p1} & x_{p2} & \cdots & x_{pn} \end{bmatrix} \tag{20}$$

Each column vector represents a *p*-dimensional location in space. Because the observation matrix *X* is compiled for a subset of the entire population (numbers *p* and *n* are finite), the variance-covariance matrix *S* derived from random variable observations is an estimator of the general variance-covariance matrix, whereas the vector of mean values *I* is an estimator of the general vector *U*. As previously stated, the aim of the principle components approach is to find linear combinations with the lowest variance. Thus, the task simply consists of replacing the initial set of variables with their linear combinations, i.e. new variables with particular attributes. These new variables are known as principle components and are denoted as follows:

$$V = A'X \tag{21}$$

where *V* - new matrix of the new variables, *A* is a matrix of orthonormal eigenvectors of matrix *S* and *X* is the observation matrix. After solving determinantal equation (22) from transformation of (21) is achievable:

$$|S - lI| = 0 \tag{22}$$

where *S* is a variance-covariance matrix of order (*p x p*), *l* is the determinantal equation's characteristic root, and *I* is a unit matrix of order (*p x p*). Equation

(22) is a polynomial of degree p with respect to an unknown l, hence it has p roots that may be sorted in the following way:

$$l_1 \geq l_2 \geq l_3 \geq \cdots \geq l_p \geq 0 \tag{23}$$

Because each root l_i has an orthonormal column eigenvector A_i, the variable V_i produced from equation (22) has the largest value l_i (maximum variance) and is referred to as the first main component. Because the sum $l_1 + l_2 + \cdots + l_p = t_r S$ and equals the sum of the variances of matrix S (i.e $(\sigma_{11} + \sigma_{22} + \cdots + \sigma_{pp})$, $l_1, l_2, \ldots, l_p$ denotes the proportion of variability of certain main components in the overall variance of matrix S. When we look at the quotients, we can see that :

$$\frac{l_1}{tr\,S}100, \frac{l_2}{tr\,S}100, \ldots, \frac{l_p}{tr\,S}100, \tag{24}$$

we obtain the percent share of each component in matrix *S*′s variance. The technique for calculating main components is designed in such a way that this is a diminishing sequence, indicating that $l_1 \,/\, t_r S100$ is the greatest quantity. Quantity l_2 relates to variable V_2, and is hence referred to as the second major component. There are clearly as many primary components as there are initial variables. Each root l_i corresponds to a column vector A_i, such that

$$(S - l\,I)A_i = 0 \quad or \quad SA_i = l_i A_i \tag{25}$$

Because vectors $A_1, A_2, \ldots, A_p$ are orthonormal, that is

$$A'A_i = 1, \qquad A_jA_i = 0 \ \ for\ i \neq j, \tag{26}$$

And they satisfy equations (22) and (25) we have :

$$A_i'SA_i = l_i, \qquad A_i'SA_i = 0\ for\ i \neq j, \tag{27}$$

$$I = \ A_1A_i' + \ldots + A_pA_p' \tag{28}$$

and

$$S = \ l_1A_1A_1' \ + \ l_2A_2A_2' + \ldots + l_pA_pA_p' \tag{29}$$

A spectral decomposition of a matrix S is what expression (29) is. The essential attribute of the new variables (in contrast to the previous variables) is their lack of correlation. The ith component's variance is l_i, or :

$$Var(A_iX) = \ l_i \tag{30}$$

where

$$Cov(A_iX, A_jX) = \ 0 \ \ for\ i \neq j \tag{31}$$

Because the major goal of principle components analysis is to reduce the dimensions of the observation space, it is vital to select at some point how many additional variables should be taken into account for future investigation. The ratio of distinctive roots to the trace of the matrix is thought to aid in decision making. For example, if the equation $l_1 \ / \ t_rS100$ has a large value (e.g., 90%), the set of initial variables is substituted with the first component V_1. When the ratio is not too high, the following components are

considered. Naturally, the exclusion of some components from further analysis cannot be based entirely on the researcher's subjective appraisal of the $l_1 \ / \ t_r S100$ quotient, but must be the outcome of component testing. The interpretation of the components is an essential topic in Principal Components Analysis since it helps discover which initial variables have the biggest shares in the variance of certain principal components after the observation space has been reduced. This information may be acquired by applying the coefficients of determination that have been established between the components and the beginning variables. It should be noted that the components' meaning varies somewhat depending on whether S or R is employed. Principal components generated from the variance-covariance matrix *S* are interpreted. The following equation defines the coefficient of correlation between the *i'th* component and the *j'th* starting variable:

$$r_{ij} = \frac{a_{ij}\sqrt{l_i}}{S_j} \tag{32}$$

As a result, the coefficient of determination has the following form:

$$r_{ij}^2 = \frac{a_{ij}^2 l_i}{S_j^2}, \tag{33}$$

where a_{ij}^2 is the square of the eigenvector element A_i represents the *i'th* component and the *j'th* starting variable, l_i represents the variance of the *i'th* component, and S_j^2 represents the variance of variable *j*. On the basis of (29) and (30), and using the variances and eigenvectors of all the primary

components, the variance-covariance matrix S may be reconstructed. Naturally, the product $l_1 A_1 A_1'$ has the most influence on this reconstruction. Furthermore, in matrix $l_1 A_1 A_1'$, for example, the components on the main diagonal represent estimates (provided by the first component) of the variance of the *j'th* starting variable, which may be derived using a generic expression:

$$\hat{V}arX_j = l_1 a_{ij}^2 \tag{34}$$

The following parts of the spectral decomposition are estimates of covariance, with the last element of $l_1 A_1 A_1'$ (matrix) bringing the estimated variance and covariance up to real values. Given (34) may be expressed in the following way :

$$r_{ij}^2 = \frac{\hat{V}arX_j}{VarX_j} = \frac{\hat{S}_j^2}{S_j^2} \tag{35}$$

The coefficient of determination between component *i* and variable *j* is clearly defined as a ratio of variable *j's* estimated variance to its true variance. If we examine any (*i'th*) matrix from the spectral decomposition, we may calculate the major diagonal by adding the components on the main diagonal.

$$\sum_{j=1}^{p} \hat{S}_j^2 = \sum_{j=1}^{p} l_i a_{ij}^2 \tag{36}$$

$$\sum_{j=1}^{p} \hat{S}_j^2 = l_i \sum_{j=1}^{p} a_{ij}^2 \tag{37}$$

Get :

$$\sum_{j=1}^{p} \hat{S}_j^2 = l_i \tag{38}$$

Thus, we can calculate the variance of the *i'th* component by adding up the estimated variances of specific variables. This connection can serve as a theoretical foundation for component interpretation. (24) can also be written using (38), as :

$$\frac{\sum_{j=1}^{p} \hat{S}_j^2}{trS} 100 \tag{39}$$

Although expression (39) assumes the biggest value for the first component, this metric should be utilized with caution. According to (38), practically the whole variance of the *i'th* component is made up of the estimated variance of a single variable, such as one with a large absolute value in comparison to the other ones, i.e. one with a high variance. As a result of the need for a skilled construction of the observation matrix, the variables of which should be of a comparable order of measure (39), the following dependency is recommended for use in component interpretation :

$$w = \frac{\sum_{ij}^{p} r_{ij}^2}{p} 100 \tag{40}$$

where *p* is the number of observation matrix variables and r^2 is the coefficient of dependence between the *i'th* component and the *j'th* starting variable.

equation (40) displays the percentage of variation accounted for by the *i'th* component for all variables. The findings of measure (40) applied to components obtained from the covariance matrix are often lower than those of measure (39), since in practice, one component (e.g., the first) seldom accounts for more than 50% of the variance of all variables included in the observation matrix. When a variable in the observation matrix outperforms all others in terms of value, expression (39) returns a high value for the first component and expression (40) returns a low value. This is because the variance of the component in this case is determined by a single variable, which may or may not be the most significant one (its importance is determined purely by the units of measurement used). As can be seen, expression (40) depicts the real contribution of the *i'th* component to the variance of all variables. It is worth noting that the numerator of (40) is simple to compute since the following equation is true:

$$\sum_{j=1}^{p} r_{ij}^2 = l_i \sum_{j=1}^{p} \frac{a_{ij}^2}{S_j^2} \tag{41}$$

The component accounts for the variance of all the starting variables if and only if:

$$\sum_{j=1}^{p} r_{ij}^2 = l_i \sum_{j=1}^{p} \frac{a_{ij}^2}{S_j^2} = p \tag{42}$$

Interpretation of principal components derived from the correlation matrix R. Disagreements between assessments of main components generated using equations (39) and (40) do not exist if they are derived from a correlation matrix (i.e. using normalized beginning variables). This is because, as a result of an adequate adjustment of the covariance matrix, the following dependencies hold:

$$r_{ij} = a_{ij}\sqrt{l_i} \tag{43}$$

$$r_{ij}^2 = a_{ij}^2 l_i \tag{44}$$

$$\sum_{i,j}^{p} r_{ij}^2 = \sum_{i,j}^{p} a_{ij}^2 \, l_i \tag{45}$$

$$\sum_{i,j}^{p} r_{ij}^2 = l_i \tag{46}$$

Or

$$\frac{l_i}{trS}100 = \frac{\sum_{i,j}^{p} r_{ij}^2}{p}100 \tag{47}$$

As a result, when using the correlation matrix, it is feasible to use either measure (39) or measure (40) to determine what proportion of the variation of all the initial variables is accounted for (in percent) by the *i'th* component (40). In the case of the variance-covariance matrix, such flexibility does not exist.

2.2.3 t-distributed Stochastic Neighbor Embedding

t-SNE is and unsupervised machine learning algorithm for vizualization developed by Laurens van der Maaten and Geoffrey Hinton [97]. Uses a non-linear dimensionality reduction strategy that focuses on keeping highly similar data points close together in lower-dimensional space. t-SNE computes the similarity between two points in a low-dimensional space using a heavy-tailed Student t-distribution rather than a Gaussian distribution which helps to handle crowding and optimization issues. Outliers have no effect on the t-SNE. The high-dimensional Euclidean distances between datapoints x_i and x_j are converted into conditional probabilities *P(j|i)* through t-SNE. Based on the fraction of its probability density under a Gaussian centered at point x_i, x_i would choose x_j as its neighbor. σ_i denotes the Gaussian variance centered on datapoint x_i. A pair of points' probability density is proportional to their similarity. *p(j|i)* will be quite high for local data points and microscopic for distant data points. To obtain the final similarities in high dimension space we should symetrize the conditional probabilities in high dimension space. As seen below, conditional probabilities are symmetrized by averaging the two probabilities.

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{48}$$

Based on the pairwise similarity of points in high dimensional space, we can map each point in high dimensional space to a low dimensional map. The two-dimensional or three-dimensional map will be used for the low-dimensional map.

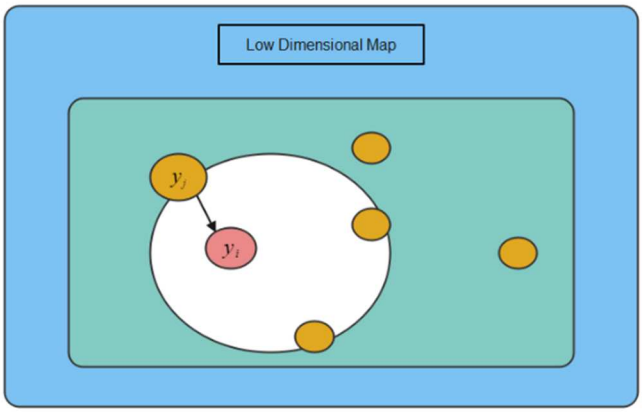


Figure 9 : Data in low dimensional space

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq 1}(1 + \|y_k - y_l\|^2)^{-1}} \tag{49}$$

The low-dimensional datapoints y_i and y_j correspond to the high-dimensional datapoints x_i and x_j . Computing the conditional probability *q(j|i)*, which is comparable to *P(j]i)*, centered under a Gaussian centered at point y_i, and then symmetrize the probability. Then using gradient descent and Kullback-Leibler divergence [98], it is found a low-dimensional data representation that minimizes the mismatch between *Pij* and *qij* (KL Divergence)

$$\sum_i KL(P_i \,||\, Q_i) = \sum_i \sum_j \; p_{j|i} \, log \frac{p_{j|i}}{q_{j|i}} \tag{50}$$

P_i depicts the conditional probability distribution for point x_i over all other data points. Given map point y_i , Q_i represents the conditional probability distribution across all other map points. t-SNE uses gradient descent [99] to optimize points in lower dimensional space. The reason why we should use the *KL* divergence [98] is that when we reduce the *KL* divergence, q_{ij} becomes physically equal to P_{ij}, and the data structure in high dimensional space is equivalent to the data structure in low dimensional space. If P_{ij} is big, then we require a large value for q_{ij} to represent local points with more similarity, according to the *KL* divergence equation. If P_{ij} is tiny, we require a lower value for q_{ij} to represent distant local locations.

2.2.4 Machine Learning Classification algorithms

In terms of classification, we propose using k-Nearest Neighbor (k-NN) and Support Vector Machines (SVM), which are both supervised and non parametric techniques. k-NN [50, 51] is an instance-based approach, which means it does not learn a model directly. It instead chooses to memorize the training examples, which are then employed as "knowledge" during the forecasting phase. In practice, this implies that the algorithm will only utilize the training examples to deliver a response when a query is performed in the database (i.e., when requested to supply a label with an input). As a disadvantage, this algorithm incurs both a storage cost during the training phase, due to the need to store a potentially massive dataset, and a computational cost during the prediction phase, because the classification of a given observation necessitates the vision and/or analysis of the entire dataset. The k-NN is a classification algorithm. In the context of

classification, the k-NN method effectively determines a majority vote among the k-nearest neighbors to a given unknown instance. A distance metric, often the Euclidean distance, between two data points defines their closeness. SVM [70] method classifies data by establishing a linear or non-linear decision boundary to distinguish various groups. It projects the data through a non-linear function to a higher dimension space, elevating them from their original space to a future place with an indefinite dimension. SVM employs kernels to carry out this procedure, the most common of which is the Gaussian kernel. Fig. 10a and 10b show a graphical illustration of these two classifications.

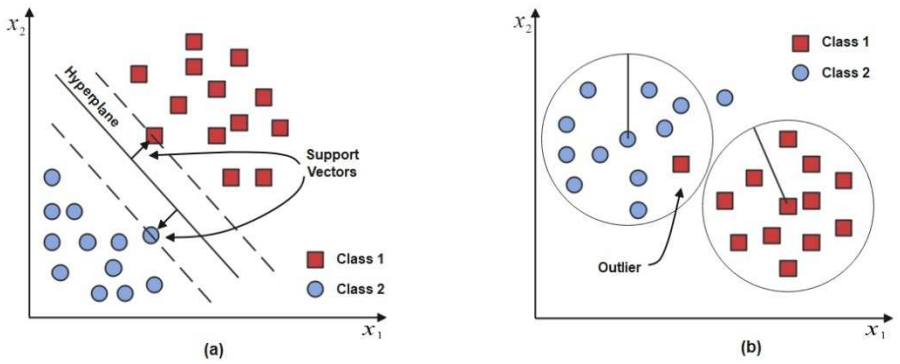


Figure 10: Two class classification example (a) SVM (b) kNN, in this case d1 and d2 are the distances metrics

2.2.5 SVM Support Vector Machine

The support vector machine algorithm's goal is to find a hyperplane in an N-dimensional space (N is the number of characteristics) that clearly classifies the data points.

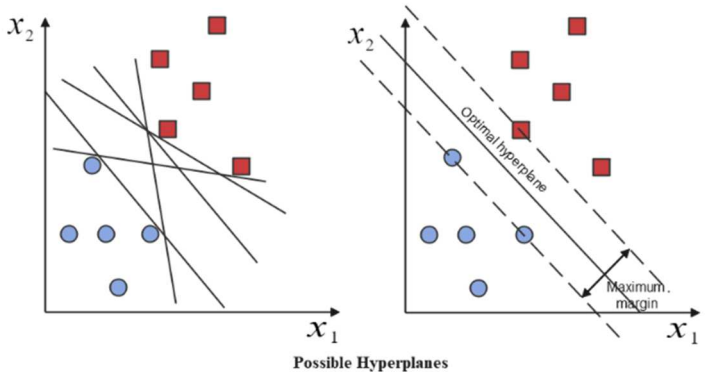
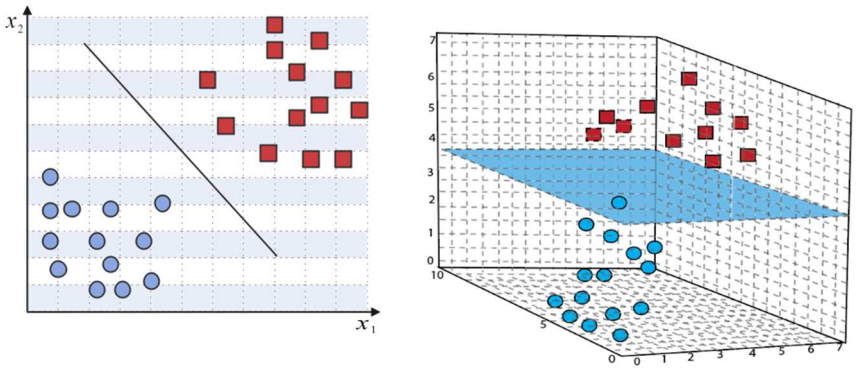


Figure 11: Possible Hyperplanes

There are several hyperplanes that might be used to split the two groups of data points. Our goal is to discover a plane with the greatest margin, i.e. the greatest distance between data points from both classes. Maximizing the margin distance gives some reinforcement, allowing subsequent data points to be categorized with more certainty.



(a) (b)

Figure 12: Hyperplane in 2D and 3D feature space. (a) A hyperplane in R^2 is a line, (b) A hyperplane in R^3 is a plane

Hyperplanes are decision boundaries that aid in the classification of data items. Different classifications can be assigned to data points that lie on either side of the hyperplane. Furthermore, the size of the hyperplane is determined by the number of features. When the number of input features is two, the hyperplane is just a line. When the number of input characteristics reaches three, the hyperplane transforms into a two-dimensional plane. When the number of characteristics exceeds three, it becomes impossible to imagine.

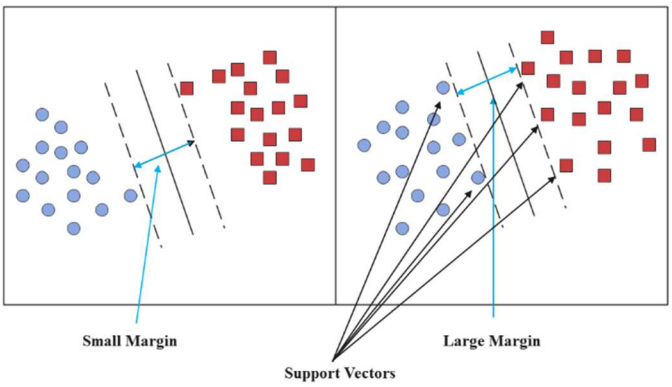


Figure 13: Support Vectors

Support vectors are data points that are closer to the hyperplane and impact the hyperplane's location and direction. Using these support vectors, we optimize the classifier's margin. The location of the hyperplane will vary if the support vectors are removed. These are the points that will assist us in developing our SVM. In SVM, we take the output of the linear function and identify it with one class if it is larger than 1, and another class if it is less than 1. Because the threshold values in SVM are modified to 1 and -1, we get this reinforcing range of values ([-1,1]) that works as margin.

2.2.6 KNN K-Nearest Neighbors Algorithm

The k-nearest neighbors (KNN) technique is a straight-forward supervised machine learning approach that may be used to address classification and regression issues. The KNN algorithm presumes that comparable objects exist nearby. Classifies data points based on their similarities. It makes a "informed judgment" on what an unclassified point

should be classed as based on test results. KNN is a non-parametric method that is also an example of lazy learning. It is non-parametric because it makes no assumptions. Rather than assuming a conventional structure, the model is built solely from the data provided to it. The term "lazy learning" refers to the algorithm's inability to make generalizations. This implies that applying this strategy requires little training. As a result, when employing KNN, all of the training data is also used in testing. Simple to use. Calculation time is short. Makes no assumptions about the data. The accuracy is determined on the quality of the data. It is necessary to determine the best k value (number of nearest neighbors). Poor at categorizing data points that are on a border that may be labeled one way or the other.

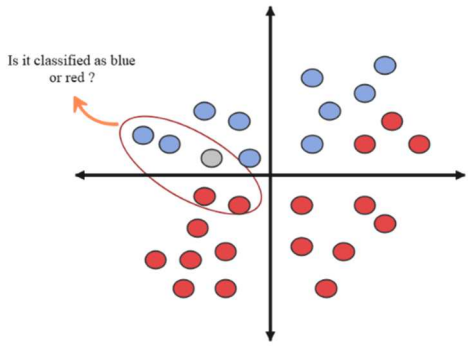


Figure 14: k-NN classification principle

The initial step in developing kNN is to convert data points into feature vectors, or their mathematical value. The method then determines the mathematical distance between these points mathematical values. The

Euclidean distance, as demonstrated below, is the most often used method for calculating this distance.

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \quad (51)$$

kNN uses this formula to calculate the distance between each data point and the test data. It then calculates the chance that these points are similar to the test data and classifies them depending on which points have the highest probabilities. To summarize kNN works by calculating the distances between a query and all of the instances in the data, then picking the number of examples (K) closest to the query and voting for the most frequent label (in the case of classification) or averaging the labels (in the case of regression).

2.3 Experimental Implementation Test

Four classification pipelines may be created by combining the previously outlined approaches. To assess their performance, we need to create a dataset of mD signals, and for these tests, we select to identify different types of walking [62] activities. The dataset, which can be accessed in [71], comprises of six distinct types of activities carried out by 29 people who repeat each activity numerous times. The subjects walk sans shoes. The participants walk without any limitation or pattern, and each subject was simply instructed to walk in a "slow" or "rapid" manner, without defining the number of steps or the time necessary to finish the exercise, in order to obtain

data as realistic as possible. In addition, acquisitions from participants of various heights and weights were gathered to give a collection that encompassed a wide range of attributes. The acquisitions are carried out in our department's corridor, with the subjects walking in front of the radar system, which is mounted on tripods. Fig. 15 depicts the test region.

Figure 15: Hallway used for the acquisitions

In each acquisition, the subject performs the action by moving [52] away from the radar and then returning, always in front of the radar system. The activities under consideration are as follows:

- Walking slow;
- Walking slow with hands in pockets;
- Walking fast;
- Walking with hiding a metallic bottle;

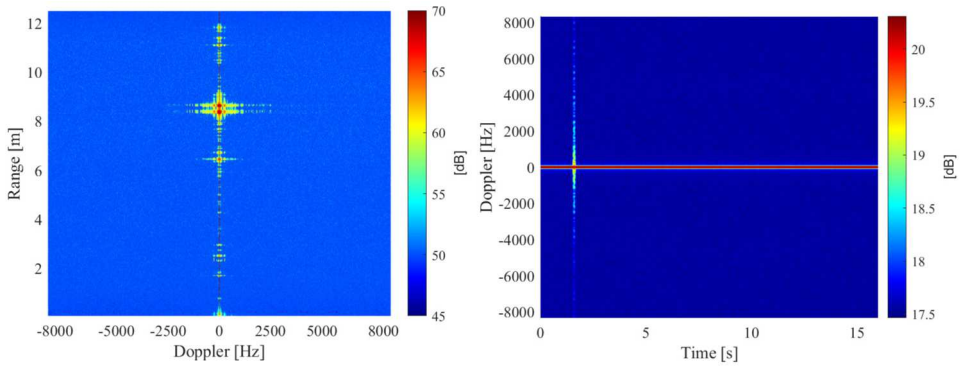
- Limping;
- Walking slowly and swinging hands.

We chose to employ only the first three tasks in the work reported in [63] and [100]. This subset of the dataset is made up of nineteen people who repeated each task three times for a total of 168 distinct acquisitions. The difference in walking speed is slight and relies on the individual being tested, who interprets it subjectively. In general, the average speed measured for a brisk walk is approximately 2 m/s, whereas the average speed measured for a sluggish walk with both free hands or hands in pockets is around 1.2 m/s. The maps used in the classification procedure are created as previously stated; for the Doppler-time map, the STFT function utilizes windows of 512 samples, with an overlap of 98 percent when a Hann window is employed. The radar configuration utilized for these experiments is detailed in Tab. 1.

Table 1: Radar parameters

Parameter	Value
f_{start}	77 GHz
Slope	60.012 MHz / μs
t_{idle}	100 μs
ADC Valid Start Time	6 μs
f_s	10 Msps
t_{ramp}	60 μs
$n_{samples}$	512
n_{frame}	400
No. of chirp per frame	128
Periodicity	40 ms
Used Radar Bandwidth	3.6 GHz

A preliminary analysis was performed on the backdrop without a person, as shown in Fig. 16a and 16b. Because the test area is the same for all individuals, just one measurement was taken. This investigation shows that the backdrop has no effect on the data, thus we can ignore it in the movement classification.



(a) (b)

Figure 16: Analysis on the background in absence of subjects using (a) RangeDoppler map and (b) Doppler-Time map.

In Fig. 17, we present an example of a person walking in various directions, exhibiting both range-Doppler maps (on the left) and Doppler-time maps (on the right). It is feasible to see that slow and quick walks are plainly distinguishable on the maps. Maps associated with a leisurely walk with hands in pockets show a little less obvious Doppler effect as compared to free hands, as predicted, although this effect is hardly discernible.

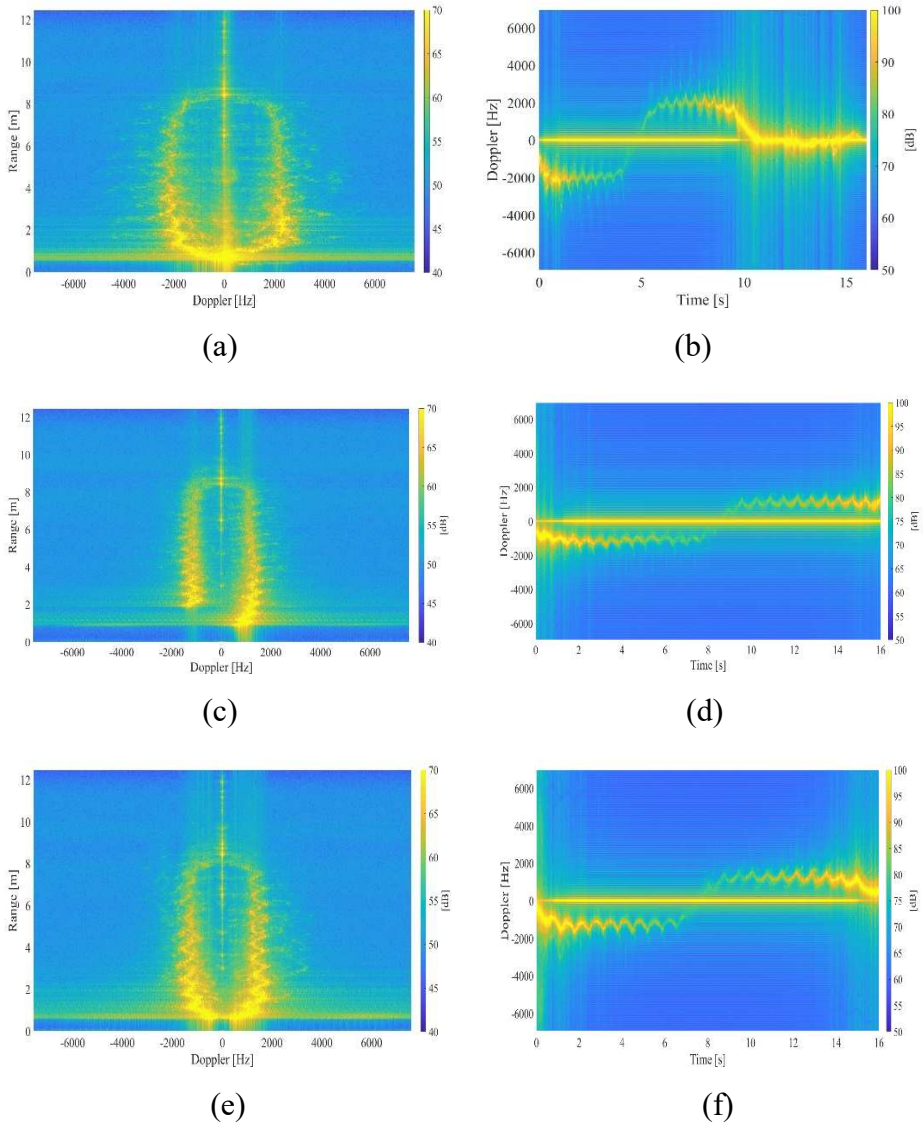


Figure 17: Example of a person walking fast ((a) and (b)), slowly with hands in pockets ((c) and (d)) and slowly ((e) and (f))

Images are created using the data collected after the radar signal has been processed. Because their original dimensions are too large to manage, all matrices have been reshaped to the same dimension [195 x 119], with the result acquired by applying a mean between neighboring pixels of the image. To further reduce dimensionality and extract characteristics from pictures, the PCA [57] and t-SNE algorithms were applied to the data individually. The matrices are vectorized, resulting in a matrix with each row representing a range-Doppler map or a Doppler-time map [56]. Figures 18a and 18b illustrate the classification accuracy obtained by utilizing a different number of main components with the a k-NNclassifier and a SVM algorithm. We used a Gaussian kernel for the SVM. The value of k for the k-NN and the kernel for SVM were set using a leave-one-out cross-validation approach that tries to minimize validation error. Each sample of the dataset is alternately chosen as a validation set, with the remaining portion being the training set. As a result, each sample is only utilized once for training and once for validation. The algorithm's results for odd values of k between 1 and 49 are presented in Fig. 19, where k equal to 1 results in an inaccuracy of roughly 2.4 percent. In Tab. 2, the validation error achieved by different kernels is provided in percentage, pointing the option to the usage of linear kernel in our case.

Table 2: Results of the leave-one-out cross validation for support vector machine (SVM) with different kernels.

Kernel	Linear	Gaussian	Polynomial
Error Validation (%)	4.46	17.26	33.33

We define the measures that will be used to assess the performance of the suggested approaches here. They are based on the so-called confusion matrix, which has columns that reflect expected values for each class and rows that represent actual values. The most often used metric is accuracy, which is defined as :

$$A_{CC} = \frac{TP + TN}{TP + TN + FP + FN'} \tag{52}$$

Where TP and TN represent true positives and true negatives, respectively, and FP and FN represent false positives and false negatives. It reflects the proportion of photos that have been classified to the right category. Another helpful statistic is accuracy (also known as positive predictive value), which is the ratio of successfully predicted positive observations to total expected positive observations, or :

$$P = TP \,/\, (TP + FP) \tag{53}$$

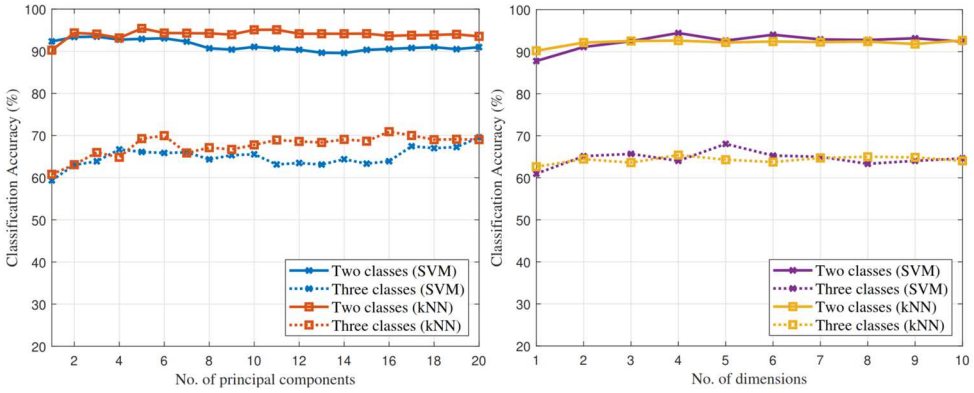
Recall (also called as sensitivity) is the ratio of accurately predicted positive observations to all observations in the actual class.

$$R = TP \,/\, (TP + FN) \tag{54}$$

The F1 score is the harmonic mean of accuracy and recall, with 1 being the best number (perfect precision and recall). As a result, this score considers both false positives and false negatives, as shown below.

$$F1 = 2RP \,/\, (R + P) \tag{55}$$

Sixty percent of purchases are utilized for training, with the remaining for testing. The findings were averaged across 100 classification results obtained by randomly selecting training and test sets. We only examine two groups here, which correspond to the slow and rapid stroll. Interestingly, the number of main components (or dimension in the case of t-SNE), which corresponds to the number of features in this scenario, has a minor influence on classification performance. The use of the PCA or t-SNE algorithms to extract features from pictures yields relatively similar results, despite the fact that t-SNE was initially meant to reduce data to two or three dimensions and becomes very sluggish for greater values. Furthermore, we get the same findings with both range-Doppler and Doppler-time maps.



(a) (b)

Figure 18: Comparison of classification accuracy achieved by SVM and kNN considering 2 and 3 classes, applying (a) Principal Component

Analysis (PCA) and (b) t-distributed stochastic neighbor embedding (t-SNE)

In Tables 3 and 4, we exhibit the classification results in terms of confusion matrices produced by applying classification to two and three different classes, respectively. In the first table, measures of slow walk and slow walk with hands in pockets have been combined into a single class, however in the second table, they have been separated into two distinct groups.

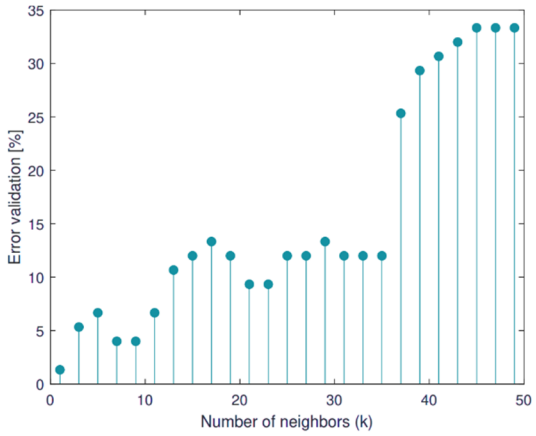


Figure 19: Results of the leave-one-out cross-validation for the k-Nearest Neighbor (kNN)

As expected, differentiating free hands from hands in pockets is a far more difficult challenge than identifying different walking styles. In the first case, the best accuracy obtained is approximately 72 percent, and red boxes indicate the presence of a number of misclassified examples, despite the fact that the fast walk is recognized from the other activities with a high precision (87.5 percent); SVM methods appear to achieve better performance than

KNN algorithms. Instead, in the latter situation, we have a great accuracy of more than 93 percent. In Tabs. 3 and 4, we highlighted a high number of accurate detections in green, whereas a large number of misclassified samples is emphasized in red.

Table 3: Confusion matrix obtained applying SVM and kNN(into parentheses) on two classes, considering 5 principal components, acc = 93.5 %.

True / Predicted	S	F
Slow Walk (S)	110 (109)	2 (3)
Fast Walk (F)	9 (8)	47 (48)

Table 4: Confusion matrix obtained applying SVM and kNN (into parentheses) on three different classes, considering 9 principal components, $acc_{SVM} = 72\%$, $acc_{KNN} = 66.7\%$.

True / Predicted	S	F	SH
Slow Walk (S)	33 (32)	2 (1)	21 (23)
Fast Walk (F)	4 (5)	49 (48)	3 (3)
Slow Walk with Hands in Pockets (SH)	16 (22)	1 (2)	39 (32)

In Tab. 5, we present an overview of the findings acquired by various research focusing on the classification of walking activities using radar data, highlighting the greatest accuracies obtained. The symbol [*] signifies the current work. Reference [64] considers 7 types of activities, namely walking backwards, limping, depressed, elderly, excited, holding the arm, and walking in a zigzag, and the radar used is an UltraWide Band; Reference [65] considers a FMCW radar, and the activities examined are crawl, creep on

hands and knees, walk, jog, and run. Although the distinction between walking slowly and swiftly is less obvious than in the other activities, we demonstrate that our method can obtain higher accuracy. Furthermore, we take into account a greater number of individuals who move differently from one another, proving the applicability of our strategy in a practical situation. With a specific accuracy of 42.42 percent, the action of holding the arm while walking [64], which is similar to our instance of walking slowly with hands in pockets, could not be distinguished from the others.

Table 5: Comparison of different radar based methods for human walking classification.

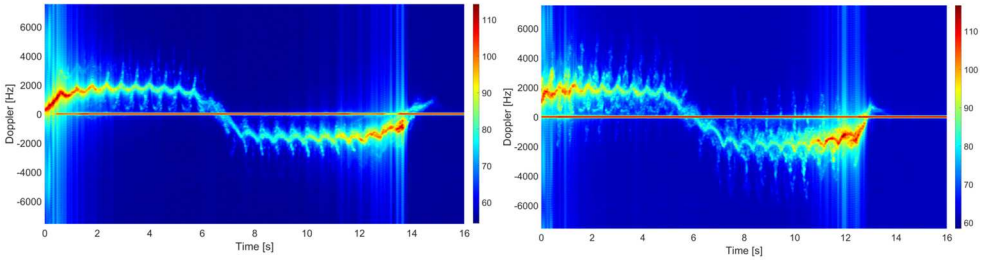
	Radar Type	**Nr. of activities**	**Dataset dimension**	**Algorithm**	**Best accuracy**
[*]	FMCW mmWave	2	19 subjects, 168 acquisitions	PCA/t-SNE + k-NN/SVM	93.5%
[*]	FMCW mmWave	3	19 subjects, 168 acquisitions	PCA/t-SNE + k-NN/SVM	72%
[64]	Ultra Wide Band	7	8 subjects, 280 acquisitions	PCA + SVM	89.1%
[65]	FMCW mmWave	5	3 subjects, 95 acquisitions	CV/TV + SVM	91%

The subjectivity and personal speed interpretation of the completed tests is the primary cause of mistake in our categorization model. A consistent duration or number of steps during the experiment would almost certainly increase system performance, but this is not a practical circumstance. The similar method may be used to the other portion of the dataset where the other activities are more varied. This work may be found in [66], and it will be

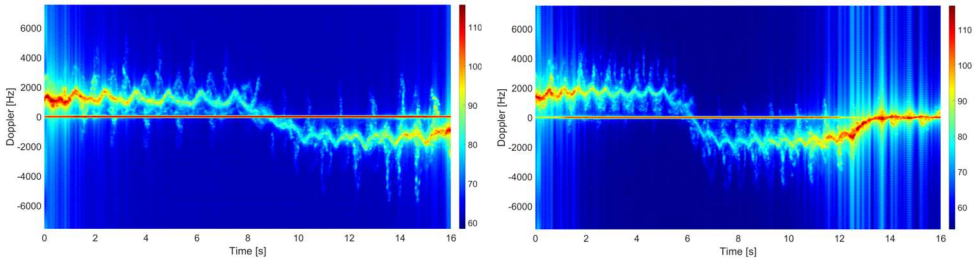
reported and debated throughout the rest of this section. Only PCA and k-NN are studied in this scenario because the other pipelines produce comparable findings. Four distinct activities, divided into four classes, were investigated:

- Slow walk with non-swinging hands [NS] (55 samples);
- Limping [L] (20 samples);
- Slow walk with swinging hands [S] (20 samples);
- Fast walk [F] (36 samples).

In terms of the limping exercise, individuals were instructed to walk with a limp. This led in numerous distinct styles of walking, as each subject carried out the request in their own unique way. In order to get data that was as realistic as feasible, no instructions were given on the other type of walk. Doppler-time maps for four activities done by the same person are presented in Fig. 20. The variations between the several types of walks analyzed may be seen on these maps, due to the great sensitivity of the radar employed. Swinging hands (Fig. 20b), for example, provide a more noticeable effect on Doppler than the non-swinging case (Fig. 20a), but limping (Fig. 20c) produces a considerably different map. The results produced with the range-Doppler map are discussed further below.



(a) (b)



(c) (d)

Figure 20: Doppler-time maps of a person walking slowly with non swinging hands (a), slowly with swinging hands (b), limping (c) and rapidly (d)

As stated in the last test, data received from radar signal processing [55] are considered as pictures, and all matrices are first reshaped to the same size [180 *x* 71], then their dimension is further decreased using PCA. As previously, the value of k was determined using a leave-one-out cross-validation procedure, with a sample of the dataset serving as a validation set and the remainder serving as a training set. As a result, each sample is only utilized once for training and once for validation. The algorithm's results for odd values of k between 1 and 49 are presented in Fig. 21, where a value of k equal to 1 leads to the lowest error, amounting to 1.33 percent.

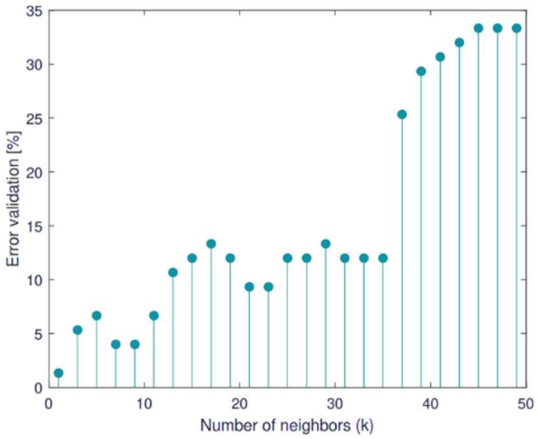


Figure 21: Results of the leave-one-out cross-validation for the k-NN for the second dataset

During the classification process, 70% of the acquisitions are utilized for training, while the remaining 30% are used for testing. To lessen the reliance of the findings on the training set selection, the results were averaged over 50 classification results obtained by selecting training and test sets at random. The confusion matrix derived by evaluating 9 major components and then applying k-NN is shown in Tab. 6. We achieve an average accuracy of 96.1 percent, with exceptional precision, particularly in recognizing the slow walk with non-swinging hands and the quick walk.

Table 6: Confusion matrix obtained applying k-NN on four activities, considering 9 principal components. Accuracy 96.1%.

True/Predicted	*NS*	*L*	*S*	*F*
NS	98.91%	0	0	1.09%
L	3.67%	91.33%	0	5%

S	0	4.33%	95.67%	0
F	1.5%	0	0	98.5%

This second component of the dataset yields results comparable to the first. It is feasible to differentiate two separate actions performed by a person with high performance, but this lowers when we try to distinguish comparable activities such as walking slowly and slowly with hands in pockets. The experimental evaluation confirms the effectiveness of the constructed pipelines, but also highlights the key disadvantage. To boost data correlation with an overhead in data processing, the feature extraction approach must be performed to all datasets. As a result, in the next part, we investigate several methods for extracting features from maps, including ad hoc creation as well as a Deep Learning [54] technique.

2.4 Different classification approaches

This section will examine several methods for classifying mD signals. This study will be based on the previously stated dataset and will employ just three activities: rapid walking, slow walking, and slow walking with hands in pockets. Figure 22 depicts the various pipelines.

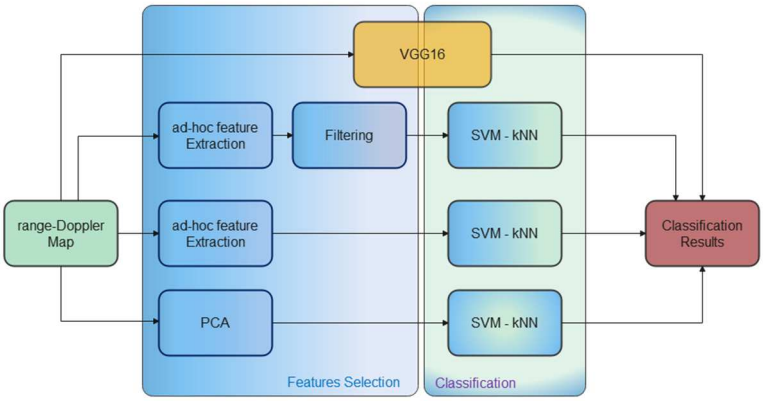


Figure 22: Flowchart of the different pipelines compared

The goal of this research is to determine whether it is possible to attain comparable classification results using reduced compute complexity strategies. As a result, we compare three distinct sorts of approaches:

- PCA + k-NN or SVM;
- ad-hoc feature extraction + k-NN or SVM;
- Deep Learning (VGG16).

All comparisons will be based on range-Doppler maps, as we demonstrate how this and Doppler-time maps provide similar results. To enhance the dimension of the dataset, the range-Doppler map is divided into two halves, yielding two maps of dimension [256 *x* 25600] from each acquisition (for the remainder of the chapter, we refer to this matrix as "map").

2.5 PCA complexity analysis

The result of PCA is a *Y*-dimensional matrix of main components. The initial components of each vector y in the resultant matrix Y include the highest potential variability of the original variables, allowing for the selection of a limited number of variables that convey the most information. The classification algorithm can then utilize Matrix Y as input. In terms of PCA computing cost, we examine a dataset made up of L samples (or photos in our instance). Before entering the PCA, each picture is vectorized and has a dimension of [M x N]. As a result, the dataset dimension is [L x MN]. The construction of the covariance matrix, which takes in the order of $O(L^2 \cdot MN)$, is the major computing expense of PCA. This covariance matrix is then used to compute eigen values and eigen vectors, which has a complexity on the order of $O(L^3)$. PCA's overall complexity is thus on the order of $O(L^2MN + L^3)$.

2.6 Parameters extraction from range-Doppler maps

PCA use often necessitates a significant computing cost. Furthermore, in order to acquire the greatest possible correlation between samples, the algorithm must be applied to the complete dataset, and whenever a new sample is examined, the procedure must be restarted. An alternate approach for feature extraction from range-Doppler maps can be offered for this purpose. On each map, we may compute several characteristics that can be utilized as features. The parameters computed from the range-Doppler maps are as follows:

1. Maximum Doppler value: the parameter max_D is derived by taking the maximum intensity value for each row of the Range-Doppler map and dividing it by the number of rows.

$$max_D^{(i)} = max\big(R_D(i)\big) \tag{56}$$

where i is the row index and $R_{Doppler}(i)$ is the chosen row. The rows, in reality, include information about the target's velocity, and the maximum specifies the Doppler value for each evaluated distance. Figure 23 depicts an example of the computation method. All of the $max_D^{(i)}$ items at the end can be collected into a vector named $\overline{max_D}$.

2. Mean Doppler value: The mean values of the Doppler are assessed for each row of the Range-Doppler matrix, using a procedure similar to that used to compute the maximum. These numbers are added together to form a vector called $\overline{\mu_D}$, and each element is computed as

$$\mu_D^{(i)} = \frac{\sum_{m=1}^{M} m \cdot R_D(i,m)}{\sum_{m=1}^{M} R_D(i,m)} \tag{57}$$

where i is the row number used in the calculation and m is the column index of the single element. Figure 24 shows an example of $\overline{\mu_D}$ for various types of acquisition.

3. Doppler values variance: Because of the dispersion of Doppler values within the map, the technique used to assess this parameter changes slightly from the preceding ones. Figure 25 shows an example of a normalized distribution of values within the same range distance.

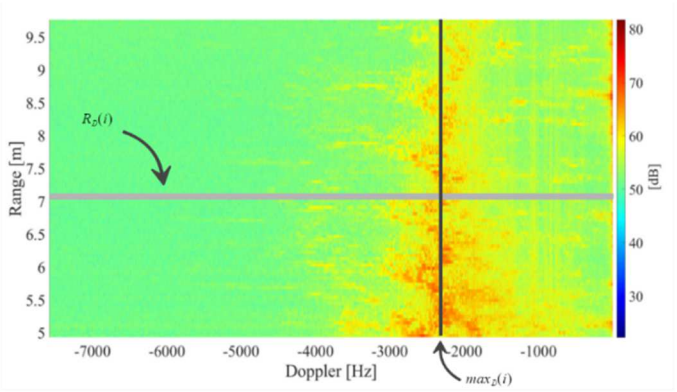


Figure 23: Extraction process of the Maximum Doppler value $max_D^{(i)}$

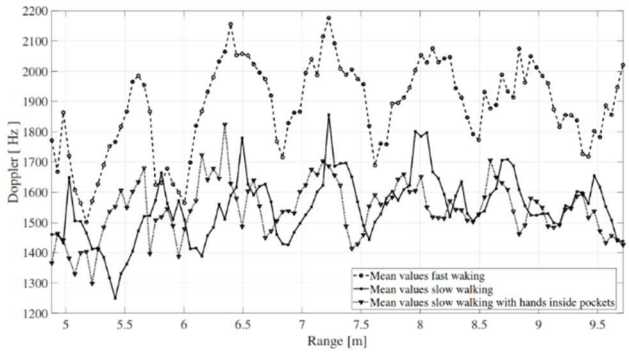


Figure 24: Comparison between the calculated values $\overline{\mu_D}$ for different acquisitions

The highest Doppler intensity values within the map are clustered near the zero-Doppler, both for the positive and negative parts. This effect is caused by sensor parameter calibration, which allows for detection of a

higher velocity than the desired effective value. On the computation of the $\sigma_D^{2(i)}$, this impact creates imbalanced results that are not centered on the mean values; as a result, the following procedure is applied to the distribution of values:

- The greatest amplitude value is retrieved for each row of RD, for the first 10000 Doppler bins in the event of negative Doppler values and the final 10000 in the case of positive Doppler values.
- By setting the values to zero, any amplitude values below the determined threshold are ignored.

The outcome of the previously mentioned technique is depicted in Fig. 26. Only at this stage can we calculate the variance σ_D^2 as below:

$$\sigma_D^{2(i)} = \frac{\sum_{m=1}^{M} m^2 \cdot R_D(i,m)}{\sum_{m=1}^{M} R_D(i,m)} - (\mu_D^{(i)})^2 \tag{58}$$

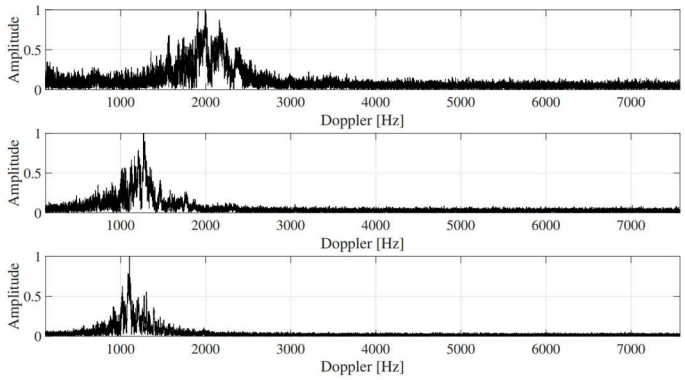


Figure 25: Doppler distribution for three different activities at the same distance. The values are re-scaled for a better comparison.

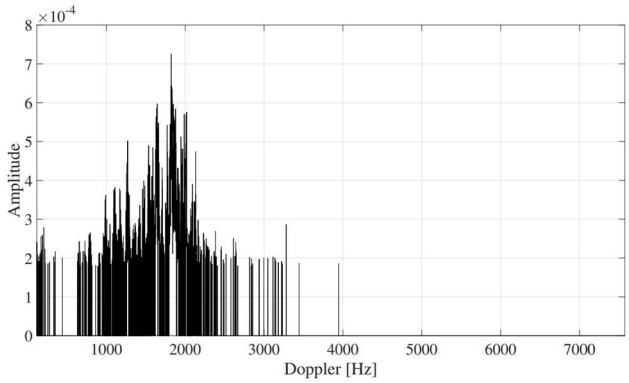


Figure 26: Effect of the applied threshold

The vectors obtained can be treated as signals, and each sample is calculated on a slow-time row within the Range-Doppler map. As a result, the sampling frequency and chirp time are related. Finally, a Butterworth filter is used to these vectors to soften the signal's rapid changes. Figure 27 shows an example of calculating the Maximum Doppler value before and after filtering for the dataset's fast and slow walk activities. Looking at equations (56), (57), and (58), we can see how much it costs to extract each parameter, that can be simply calculated as follows:

1. Maximum Doppler value: $O(L \cdot N)$;
2. Mean Doppler value: $O(L \cdot N \cdot 2M)$;
3. Variance of Doppler values: $O(L \cdot N \cdot 2M)$.

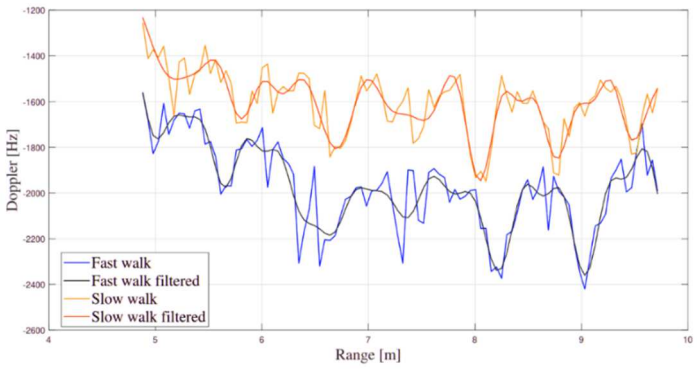


Figure 27: Effect of the Butterworth filter on the computation of the Maximum Doppler value for the fast walk and the slow walk activities.

When the sum of these costs is compared to the PCA complexity, it is clear that the second way is computationally less expensive.

2.7 VGG16 neural network classification

VGG16 proved to be a defining moment in humanity's attempt to make computers "see" the world. For decades, a lot of work has been invested into enhancing this capacity under the discipline of Computer Vision (CV). VGG16 is one of the important inventions that laid the way for a number of subsequent advances in this sector. The model's concept was presented in 2013, while the actual model was submitted in 2014 during the ILSVRC ImageNet Challenge. The ImageNet Wide Scale Visual Recognition Challenge (ILSVRC) was an annual competition that assessed picture classification (and object identification) methods on a large scale. In contrast to the enormous receptive fields in the first convolutional layer, this model recommended using a relatively modest 3 x 3 receptive field (filters)

throughout the whole network with a stride of 1 pixel. The notion of applying 3 *x* 3 filters consistently is something that distinguishes the VGG. Two successive 3 *x* 3 filters yield a 5 *x* 5 effective receptive field. Similarly, three 3 *x* 3 filters yield a 7 *x* 7 receptive area. As a result, the decision functions become more discriminative. Second, it greatly decreases the number of weight parameters in the model. If the input and output of a three-layer 3 *x* 3 convolutional stack comprise C channels, the total number of weight parameters is 3 * 32 C2 = 27 C2. When compared to a 7 × 7 convolutional layer, 72 C2 = 49 C2 is required, which is nearly double the number of 3 x 3 layers. This may also be viewed as a regularization of the 7 x 7 convolutional filters, forcing them to decompose through the 3 x 3 filters, with non-linearity introduced in-between by way of ReLU activations. This would lessen the network's proclivity to over-fit throughout the training process. The network's persistent use of 3 x 3 convolutions makes it incredibly simple, beautiful, and easy to deal with.

2.7.1 VGG16 Architecture and Uses

On the ImageNet dataset, VGG16 was shown to be the top performing model.

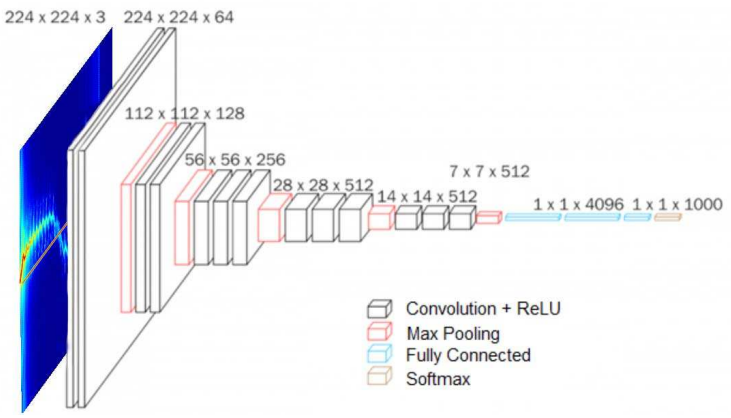


Figure 28: VGG16 Architecture

Any of the network settings considers the input to be a 224 by 224 picture with three channels R, G, and B. The only pre-processing performed is to normalize the RGB values for each pixel. This is accomplished by removing the average value from each pixel. Following ReLU activations, the image is sent through the first stack of two convolution layers with a very tiny receptive area of 3 *x* 3. These two layers each include 64 filters. The convolution stride is set at 1 pixel, and the padding is also set to 1 pixel. The spatial resolution is preserved in this setup, and the size of the output activation map is the same as the dimensions of the input picture. The activation maps are then pooled spatially over a 2 *x* 2-pixel window with a stride of 2 pixels. This reduces the size of the activations by half. As a result, the activations at the bottom of the first stack are 112 *x* 112 *x* 64.

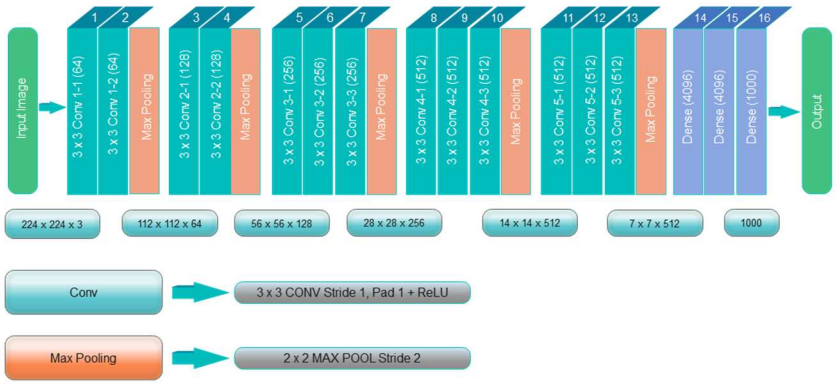


Figure 29: VGG16 Architecture model layers

The activations are then routed via a second stack that is identical to the first but with 128 filters instead of 64. As a result, the final size is 56 *x* 56 *x* 128 after the second stack. The third stack consists of three convolutional layers and a maximum pool layer. The stack's output size is 28 *x* 28 *x* 256 due to the 256 filters used. Following that are two stacks of three convolutional layers, each with 512 filters. Both of these stacks will produce 7 *x* 7 *x* 512 as their output. The stacks of convolutional layers are followed by three fully linked layers with a flattening layer in between. The first two layers have 4096 neurons apiece, while the last fully connected layer acts as the output layer, with 1000 neurons matching to the ImageNet dataset's 1000 potential classifications. The output layer is followed by the Softmax activation layer, which is utilized for categorical categorization. Despite the fact that this is a very basic, attractive, and simple to use approach, it does have certain limitations. This model has over 138 million parameters and is over 500MB in size [72]. As a result, the model's use is severely limited, particularly in

edge computing, where the inference time is longer. Second, no precise metric exists to address the issue of disappearing or ballooning gradients. This issue was solved in GoogLeNet by employing inception modules, and in ResNet by using skip connections. Despite the introduction of many new and improved scoring models since VGG was first suggested, VGG16 [73] continues to pique the curiosity of data scientists and researchers worldwide. A few examples of practical applications for VGG16:

- Image Recognition or Classification – VGG16 can be used to diagnose diseases using medical imaging such as x-rays or MRI. It may also be used to recognize street signs while driving.
- Image Detection and Localization —It can perform admirably in image detection use cases. In fact, it was the 2014 ImageNet detection competition winner (where it ended up as first runner up for classification challenge)
- Image Embedding Vectors — After removing the top output layer, the model may be trained to generate image embedding vectors that can be utilized for problems such as face verification using VGG16 inside a Siamese network.

2.7.2 VGG16 in transfer learning

In this situation, we will propose using the VGG16 network [58, 59] to construct a model that categorizes actions based on the classes in the dataset. Transfer learning [69] can be used to avoid training the model from beginning when working with a dataset with a small dimension. Although neural networks require a large dataset for training, it is feasible to use a smaller

dataset by applying transfer learning [69]. Transfer learning is very beneficial when working with minimal amounts of data since it allows you to extract network weights from pre-trained models and transfer them to other networks, which saves money on training new neural networks. Range-Doppler maps are used as training input, and the model built on the ImageNet Large Visual Recognition Challenge dataset [70] is then retrained on the specified dataset using the approach outlined in [69]. The computational cost of a convolutional network for a single picture is $O(F_I MNmnF_O)$, where each input feature map is of size $(M \cdot N)$, spatial two-dimensional kernels are of size $(m \cdot n)$, and F_I, F_O are the input and output channels within a layer, respectively [71].

2.8 Machine learning Classification

Following feature selection, classification is carried out using the k-NN and the SVM, which have distinct computing costs. As previously stated, the parameters for both methods were optimized using a leave-one-out crossvalidation procedure, with the purpose of minimizing the validation error. As a result of this evaluation, k is set to one and a linear SVM is chosen. Following is a complexity analysis of the k-NN algorithm:

In general, a k-NN [60] has a computational cost in the order of $O(Ly + kL)$, assuming that k is determined previously. The training set's cardinality is represented by L, while the dimension of each sample is represented by y. y might be the PCA result or the vector that contains the extracted parameters.

The overall cost of NN algorithms may be simply calculated by following the steps below:

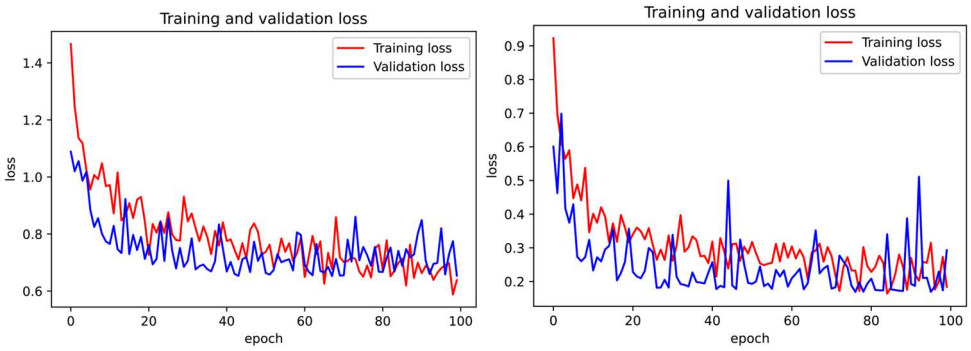
- Distance computation: computes the distance of a new observation from each sample of the training set. The cost required for each computation is in the order of *O(y).*
- Label assignment: by looping through the full training set, the k samples with the smallest distance value are chosen. The label applied to the new observation is the same as that of the majority of the k samples chosen. By looping through the training set observations, each iteration of the second step incurs a cost in the order of *O(L),* resulting in an overall step cost in the order of *O(k L).*

SVM computational complexity: Because linear SVM is essentially a single inner product, its complexity is equal to *O(L).* Kernel SVM's often have a greater cost, which is determined by the kernel used and the number of supporting vectors s_v. The cost is $O(s_v L)$ for most kernels [61], including polynomial and Gaussian. There is an approximation for SVM's with a Gaussian kernel that decreases the complexity to $O(L^2)$. As a result, linear SVM has a lower cost than k-NN, although kernel SVM may have a larger complexity. The cost of categorization must then be added to the cost of the feature selection technique under consideration.

2.9 Deep neural networks results

The dataset is splited into three parts: 80% for training, 10% for validation, and 10% for testing. In Fig. 30a and 30b, we illustrate the training and validation loss for the three independent classes and when a slow walk

and a slow walk with hands in pockets are combined, respectively, based on 100 training steps, or *epochs*. The training loss represents how well the model fits the training data, whereas the validation loss represents how well the model fits new data. Cross-validation is used to assess them. In Fig. 30, values less than 0.75 are seen after 80 epochs, and tests with more than 100 epochs tend to overfit. When just two activities are examined, the performance improves and the validation loss takes values between 0.2 and 0.3 on average for a limited number of epochs. Table 7 displays some additional classification metrics for our model for each evaluated class, namely accuracy, recall, and F1-score. It is feasible to see that both the slow and fast walk groups attain good accuracy but low recall, whereas the slow pocket class exhibits the reverse trend.



(a) (b)

Figure 30: Training and validation loss of VGG16 neural network for a) three and b) two classes.

2.10 Comparison of several techniques

As previously stated, we extract three parameter vectors from the range-Doppler maps, reflecting the maximum, mean, and variance of the Doppler values. After getting these vectors for all acquisitions in the dataset, they are averaged to yield three features, or six if the positive and negative halves of the range-Doppler maps are considered. Because the participants do the same action when moving away from and returning to the radar, we may divide the acquisition into two parts and increase the number of features.

Table 7: VGG16 results for the three activities

	Precision	Recall	F1-score
Fast	1	0.57	0.73
Slow	1	0.43	0.60
SlowPocket	0.6	0.86	0.71

There are 171 rows and 3 (or 6 obtaining negative part) columns in the resultant matrix. When doing the classification job, we may utilize less resources because the matrix has been decreased. In the categorization based on these characteristics, 60% of the acquisitions are utilized for training, while the remaining 40% are used for testing. The findings were averaged across 50 classification results obtained by randomly selecting training and test sets. Tables 8 and 9 show the classification accuracy attained by the various methodology described in the preceding sections, while Table 10 shows a summary of the processing expenses for each of the proposed

strategies. It is vital to remember that the VGG16 cost applies to a single picture, not a set of images.

Table 8: Accuracy achieved by the proposed methods for two activities

Method	Classification Algorithm	Accuracy
PCA	SVM	93.50%
PCA	kNN	93.50%
Ad-hoc Features Extraction	SVM	84.20%
Ad-hoc Features Extraction	kNN	86.18%
Ad-hoc Features Extraction + filtering	SVM	94.20%
Ad-hoc Features Extraction + filtering	kNN	94.20%
Deep Learning	VGG16	93.75%

Table 9: Accuracy achieved by the proposed methods for three activities

Method	Classification Algorithm	Accuracy
PCA	SVM	72%
PCA	kNN	66.70%
Ad-hoc Features Extraction	SVM	63.70%
Ad-hoc Features Extraction	kNN	66.10%
Ad-hoc Features Extraction + filtering	SVM	73.70%
Ad-hoc Features Extraction + filtering	kNN	73.70%
Deep Learning	VGG16	66.67%

Table 10: Computational costs of the considered methodologies for feature selection. *Note that the cost for the VGG16 refers to a single image.

Method	Computational cost
PCA	$O(L^2MN + L^3)$
Ad-hoc Features Extraction	$O(LN(1 + 4M))$
VGG16	$O(F_1MNmnF_0)$ *

Looking at the table, it is clear how parameter extraction combined with filtering produces better results than other approaches while also requiring the least processing cost. Surprisingly, the use of filtering has a significant

influence on the results, assisting in the improvement of performance. Deep learning [53] enhances the accuracy of the PCA plus classification technique for both two and three activities, but it is surpassed by parameter extraction with filtering. This can be attributed to the short dimension of the dataset under consideration, since the VGG16 network performs better in the presence of a big training set.

2.11 Conclusion

In this chapter, we demonstrate how an automobile radar may be utilized to categorize targets by utilizing their mD features. These devices, when combined with appropriate signal processing and a classification pipeline, can yield good results. We also demonstrate how alternative feature extraction algorithms may be employed and compare the outcomes and computational costs. We also use a more complicated neural network to improve the comparison. Based on the results, we show how an ad hoc feature extraction strategy achieves the same classification results but at a reduced computational cost. Of course, this methodology can only be utilized in a specific instance (classifying different walking patterns), whereas the other ways are more generic.

Chapter 3

Genetic algorithms in WSN

3.1 Introduction

LEACH [91] is one of the most common adaptive clustering routing systems. A Wireless Sensor Network (WSN) is made up of many sensor nodes that are grouped in a network in a given region with the primary goal of autonomously performing certain activities such as event detection, physical parameter measurements, and target object tracking. Technological improvement in electronics-related fields, particularly advancements in embedded systems, has made it feasible to raise the dependability, capabilities, and efficiency of sensor nodes while decreasing their size and cost [74]. The use of WSN benefits such as dynamic self-organizing characteristics and decentralized functioning via wireless communication has drastically enhanced the use of WSNs in many various fields. Commercial applications [75], safety systems [76], healthcare detection systems [77], wearable sensor health monitoring [78], and environmental monitoring systems [79] are the main groupings. A large range of applications with even broader performance requirements has led in the creation of a wide range of protocols with multiple changeable parameters [80]. WSN varies from standard wireless networks in several aspects, including restricted capacity nodes, severe energy limits, and application-specific features. When hierarchical architectures are considered, numerous research for different

clustering approaches are created. These techniques strive to deliver more accurate clustering by minimizing the number of clustering stages. As a result, Nazari et al. [81] proposed a novel bottom-up hierarchical clustering technique that use the intersection point as a linking criterion. This method produces more accurate clustering results since none of a data point's nearest neighbours are overlooked. WSN topology design is critical prior to network deployment because of the influence of network organization on overall system performance. Sensor nodes are typically installed in deterministic or heuristic contexts such as residences, industries, residential buildings, or hospitals. Alternatively, they are deployed at random in uncontrolled environments, such as battlefields, poisonous zones, and disaster-affected areas. As a result, some research has focused on multi-level clustering algorithms as strategies for optimizing data gathering. To optimize packet transmission and decrease latency, they employ Ant Colony Optimization (ACO) [82] or multiple Traveling Salesman Problem (mTSP) [83]. The heuristic WSN design technique in our work is based on Genetic Algorithms (GA), which is an optimization tool that mimics natural selection and genetics. GA is often used to find a global maximum or minimum in a search region containing several local maximums or minimums. Nonlinear optimization methods, including the GA, have already been used to optimize application-specific network deployment [84], as well as numerous hierarchical routing protocols comparable to LEACH [85]. Although there are various tools for constructing a WSN, most of them do not consider the chosen communication protocol or simply neglect network organization. The various solutions proposed do not consider the performance of a WSN.

Tinker, SensDep, and ANDES are three of the most researched deterministic and deductive WSN design tools. Tinker [86] is a high-level design tool for sensor networks that uses simulated data streams based on real sensor network models to decide which data processing algorithms to apply. It does not need (or enable) users to define details like routing algorithms or retransmission rules, allowing system designers to swiftly iterate between different broad concepts before fleshing out the details of the one that appears to be the most promising. SensDep [87] includes many solution methodologies to maximize sensor network cost and coverage as a software design tool. It employs a deductive technique to build a list of relevant network models that match the application environment as well as the specifications of the accessible sensor nodes and gateway. This program also takes into consideration the effects of the environment on network traffic generation. Designing a WSN is difficult due to energy limitations, specific properties based on the field, and the aim of the application. The optimum design of the WSN prior to implementation in the environment is crucial and frequently necessitates compromises between opposing objectives. Another new technique is provided in [88], which proposes a Dynamic Load Balance Clustering Mechanism (DLBCM) that not only analyses the CM loading, but also monitors the energy usage of the CM in each cluster. To maintain the overall network architecture more stable and efficient, CM re-election and cluster reconfiguration are avoided on a regular basis. The lifespan of WSN will thereafter be extended. There are further four weights to consider: residual energy, CPU usage, node communication bandwidth, and distance to the cluster's centre. Their values alter depending on the relevance of each item

in various specialized applications, and in some circumstances, these weights can be adjusted to zero. We analyse the challenges raised above and offer a WSN-based design method based on GA that can assist WSN designers in configuring parameters to achieve the desired performance prior to system deployment. As a result, a deductive design tool based on GA is proposed for the topology of WSNs with hierarchical structure. In our application, GA functions as a WSN topology design tool, autonomously producing a hierarchical cluster-based network organization based on each node's position in the distribution region, operation status, and cluster structure of the active nodes. The functioning of the GA-based design tool is limited by application-specific design requirements as well as network parameters like as network coverage, connectivity, energy consumption efficiency, and network longevity. The main purpose of this work is to present the technique used to generate the topology of a WSN based on the design criteria, as well as to evaluate the performance of the network produced in relation to the application needs. Thus, it is meant to build the topology and establish the function of nodes among hundreds or thousands of methods of assembling the network to maximize the specific design parameters.

3.2 Related Works

Clustering is often a very efficient approach [89],[90], in which sensor nodes are combined to create a cluster that is administered by the Cluster Head (CH). The CH collects the data, compresses it, and transfers it to the sink. As a result, the nodes communicate less than when data is transmitted straight to the sink. Although most cluster-routing protocols seek to evenly

balance the load between sensor nodes, by using a probabilistic model to select the CH node each round, they fail to ensure that the chosen node is the best available. There is still much space for development. Although there are numerous clustered-based protocols accessible in the present literature, just a few well-known LEACH-based protocols are covered here due to the interest of our study.

3.3 Cluster-based routing protocols

LEACH's functioning, like that of other hierarchical protocols, is divided into two phases: setup and steady data transmission. During the setup phase, the CH is chosen from among the available sensor nodes using a probabilistic model, and many clusters are built dynamically. During the continuous data transmission phase, sensor nodes in each cluster send data to the specialized CH, which compresses it and transfers it to the target sink node. The LEACH protocol elects the CH nodes and re-establishes the clusters on a regular basis, ensuring that the energy dissipation of each node in the network is reasonably consistent. Despite the fact that the LEACH methodology spreads the load evenly across each CH, there are certain disadvantages. For starters, there is no assurance that the chosen CH is the best option. For example, if the chosen CH is located near the network's boundary, other nodes may use more energy sending the message to CH. It is also impossible to predict the number of CH’s who will be elected in each round. Several protocols and approaches based on LEACH have been suggested throughout time, generally with a significant improvement in network lifespan [92]. Cluster-based routing algorithms normally focus on

maximizing network energy efficiency, however when WSN's are utilized for a specific application, there may be several QoS requirements.

3.4 Genetic algorithm based routing protocols

The majority of GA applications in WSN's are concerned with optimizing lifetime and energy usage. The enhancement is accomplished by using a GA-based algorithm in practically every operational step of WSN's, including node distribution, network coverage, clustering, and data aggregation, in order to provide a satisfying set of performance parameters for various WSN's organizations. LEACH-GA [93] was one of the first GA-based adaptive clustering methods described. The goal of this protocol is to optimize the CH selection probability model in order to achieve significant network lifespan improvement. The suggested GA-based protocol is based on LEACH and functions substantially identically to the existing LEACH protocol. It essentially contains a set-up and a steady-state phase for each cycle in the protocol, which function identically as stated at LEACH, but it varies in that this protocol incorporates an extra preparation phase in its operation. This is done only once before the first round's setup process.

In the preparation phase, before the first round of network operation, all nodes undertake CH selection based on a random model. As a result, each sensor node creates a random number from the interval [0, 1], and the produced value, along with the node ID and geographical position, is transmitted to the Base Station (BS). Only sensor nodes with values greater than a particular threshold are considered CH candidates. As the BS receives signals from all nodes, it employs a GA-based searching method to determine

the best likelihood of nodes becoming CH. The selection probability for each node is determined by reducing the overall energy consumption necessary to complete one loop. At the completion of the preparation phase, BS sends an advertisement message to all nodes with the optimal produced probability values in order to build clusters in the next set-up phase. The methods of following set-up and steady-state stages in each round are the same as in LEACH. In other words, the preparation phase creates the probability values for the set-up phase's CH selection, resulting in lowest energy use. The proposed GA-based adaptive clustering technique achieves optimal energy usage, hence extending network lifetime. Other LEACH-based clustering techniques include Genetic Algorithm Based Energy Efficient Clusters (GABEEC) [94], enhanced LEACH [95], and C-LEACH [96]. All of the presented strategies are aimed at enhancing the CH selection procedure in order to optimize the distribution of energy resources.

3.5 Enhanced Algorithm

There are four major phases to applying the GA to a problem. The initial step is to code the problem, and throughout this process, the structure of a potential solution is formed in the genome. The genome is a collection of binary or alphanumeric characters that the GA must alter in order to create candidate solutions that are more optimum. The second stage is to construct the fitness function, which has a direct influence on both the quality of the produced solutions and the complexity of the GA. In our scenario, the fitness function is made up of a collection of algorithms whose objective is to estimate the parameters of the candidate's topologies. Finally, the fourth stage

is to identify the genetic operations that influence variety, the quality of created solutions, and the convergence of the group of candidate solutions toward the global or local maximum or minimum. Recombination and mutation are two genetic pathways. GA's are deductive procedures based on random search, which implies that the algorithm searches a field of probable solutions for a global maximum or minimum. For a number of causes, the GA may prematurely converge to a local maximum in many circumstances. This can result from selecting the incorrect genetic procedures. Another explanation is how the problem is integrated into the fitness function, which is the mechanism that steers the algorithm toward optimum local or global solutions. As a result, before applying GA to an issue, it is critical to first assess and deconstruct the problem in such a way that it can be integrated into a fitness function. To use the GA in topology construction, the parameters of the network to be optimized should be specified as fitness function variables.

3.5.1 WSN model

We consider a homogeneous WSN with nodes structured in a hierarchical cluster-based network paradigm. Each node can function as a CH node or a sensor node. CH's in a WSN collect data from sensor nodes and transfer the aggregated data to a BS at regular intervals known as operation rounds. Because CH's have a high energy consumption due to their complex and demanding activities, choosing optimal CH's to improve network lifespan is a fundamental difficulty in WSN topology design in order to increase network longevity. The nodes of the WSN will be deployed across a two-dimensional square network, X *x* X units represented on Fig. 31. The

region is split into grids by a preset Euclidian distance, and nodes are put at the intersections of the grids. The detection range of CH's (green circles) is $\sqrt{2\ /\ 2}$ units, while their transmission range is $2\sqrt{2}$ units. As shown in Fig 31, the following nodes are active: Low Range Node (LRN-cyan circles) and High Range Node (HRN-blue circles). The detecting and transmission range of LRN's is $\sqrt{2\ /\ 2}$ units, and the total operation energy per round is the lowest of all conceivable states. The HRN has a transmission and sensing range of $\sqrt{2}$ units, which is twice that of the LRN, and its total operation energy each round is more than in the previous condition. Inactive Nodes (IN-nodes, X) do not conduct any processes at all, hence their energy consumption is zero. Active sensor nodes in the LRN and HRN are separated for energy optimization, coverage, and overlap reduction. To save energy, nodes near congested areas and CH prefer to switch to low range mode.

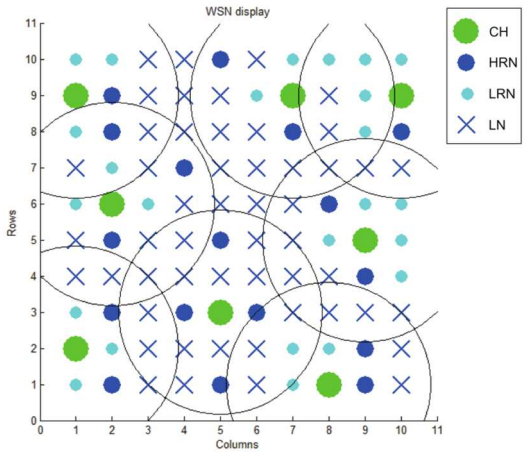


Figure 31: Network model layout

Nodes located in an uncovered region or away from the CH, on the other hand, tend to function in the HRN mode to provide coverage and preserve connections with the CH. Active nodes outside of the service region, known as Out of Range Nodes (ORN), are unable to interact with the CH. They simply need energy for operation, and because the values measured cannot be transferred to the CH, it is more economical to pass them in sleep mode.

3.5.2 WSN design parameters

As previously stated, the goal of the WSN topology design tool is to simultaneously optimize several application specific network performance parameters such as area coverage, connectivity, energy efficiency, and lifetime through cluster formation and changing the operation state and position of the nodes in the monitoring area. The effectiveness of network distribution is measured by area coverage. It is critical in practically every WSN design to obtain total coverage of the region at the lowest possible implementation cost. As a result, the design tool will favour topologies with fewer sensor nodes and greater area coverage. The total uncovered surface parameter U_S, is used to assess and evaluate area coverage. The total uncovered surface is calculated as

$$U_s = \frac{\sum U_a}{X \cdot X} \tag{59}$$

where U_a is a region with an inactive node present but uncovered by any nearby node, and X is the complete network deployment area's height and

width parameters. The GA method employs a deductive technique to find the best topology for the application at hand by selecting and continuously combining the best performing topologies from randomly generated groups. The fitness function assesses the quality of generated or selected topologies by assessing network parameters such as coverage, number of sensor nodes per CH, total average energy of the system, minimum and maximum energy for nodes, residual energy, number of nodes out of coverage, number of overlaps, and total network lifetime. The fitness function is applied to each person, i.e., topology, of a population of a specific generation, and a fitness value is assigned based on the results of the corresponding parameters and weights. In this scenario, the GA acts as the function of minimizing the fitness function, which implies that topologies with lower fitness values will be favored by the selection function to be picked in order to recombine them to produce young individuals from the following population. The implementation of the global optimization toolbox via Matlab's genetic algorithms consists of three main steps: establishing the fitness function, identifying genetic operations, and building the sequence of genome characteristics depending on the challenge. Binary coding was defined to represent all network nodes. The following syntax code line was used to enable the GA toolbox:

$$[P, FitVal] = ga(@FitnessFunction, Individual\_Size, options)$$

where "*FitnessFunction*" is the fitness function that evaluates population individuals for each generation. "*Individual_Size*" specifies an individual's genome length, which in our instance is:

$$G_L = 2N^2 \quad (60)$$

The method works by using genetic processes to favor the recombination of individuals with the lowest fitness value. Topologies with optimum performance characteristics for application needs can thus be generated through minimizing of the fitness function.

The functions implemented by the user determine and initialize the variables of the fitness function, as well as the methods of their assessment for each individual, since they are specific depending on the situation being handled. The GA of the global optimization toolbox functions as a minimization function for the fitness function in the design application of a WSN topology. The toolbox function serves as a minimization function for the fitness function. Step 7 presented in appendix A shows how, after forming a population, the GA determines the value of fitness for each individual in the population using the function of the weighted sum of the network parameters. Following that, the algorithm functions through genetic operations, preferring the recombination of individuals with the lowest fitness value. Topologies with optimum performance characteristics for application needs can thus be generated through minimizing of the fitness function.

3.5.3 Performance and application requirements

Priority number one before implementing the current network in the environment is a compromise between several WSN criteria such as connection, coverage, and energy efficiency. This function is performed

independently by the algorithm, which determines the particular position of the nodes in the network, their statuses or possible roles, and organizes the nodes into clusters. Before using the design tool, we must first examine the target network's capabilities and needs, as well as prioritize each of the performance characteristics. The user can allocate priority to each parameter in the network design tool based on the weights it assigns to those in the fitness function of GA. The user must also define network attributes such as node power capacity, communication radius, coverage radius, operational and communication energy expenses per round, communication radius, and the unit of surface on which the network would monitor for the application in question. The implementation of the network design method can begin after the performance parameter weights and network characteristics have been determined. The algorithm execution circumstances, including genetic operations, their configuration, and termination criteria, are determined experimentally throughout the algorithm's development. Table 11 summarizes the network characteristics for the application in issue.

Table 11: Network design criteria

WSN design criteria	**Value**
Surface for coverage	10 x 10 unit of surface
Maximum number of nodes	100
Energy capacity	1000 unit of energy
Operational energy for LRN	4 unit of energy
Operational energy for HRN	8 unit of energy
Operational energy for CH	16 unit of energy
Transmission radius for LRN	$\sqrt{2}/_{2}$ unit of length
Transmission radius for HRN	$\sqrt{2}$ unit of length
Transmission radius for CH	$2\sqrt{2}$ unit of length

Coverage radius for LRN	$\sqrt{2}/_2$ unit of length
Coverage radius for HRN	$\sqrt{2}$ unit of length
Coverage radius for CH	$\sqrt{2}/_2$ unit of length
Communication energy per round	0,6 * d^2 (d – distance between two nodes)

Priorities must be established when the capacity of the nodes and the features of the network to be created have been determined. As a result, the weight coefficients of each parameter in the fitness function must be determined. GA is a conventional evolutionary algorithm that is based on randomness. As a result, applying it to the simultaneous optimization of a number of interconnected performance characteristics yields a diverse set of alternative solutions. Selecting the solution set that best matches the application is a difficult procedure with no established strategy. For any application, there may be a set of weight combinations that can provide outcomes that satisfy or do not meet the majority of the criteria. The purpose at this stage is to pick parameter weights by testing different combinations of variable coefficients on the fitness function in order to discover the best performing combination for the specified application. We will use eight characteristics to identify the weighting combination that provides the best set of answers. However, five of them are particularly essential and will be used as criteria for evaluating the performance of produced network topologies. The first criterion used is the simulated environment's uncovered surface (U_s), which is assessed in relation to the entire area of the environment. Weight combinations that decrease the uncovered territory will be deemed more beneficial. The second criteria is the residual energy (E_R) in the active nodes after the network fails, which reflects the efficiency of employing the active nodes' energy capacity. A low value for this parameter

implies that we are efficient in our use of network resources. The third criteria is longevity (L_T), which is one of the most essential factors for assessing the network's energy efficiency. In order to properly execute the defined monitoring functions, the network must run in the testing environment for a specific amount of time. The final two criteria concern connectivity: the Node Degree (ND) once the network is unplugged and the amount of overlaps (O_v). ND estimates how much the network's connection is affected by the death of a CH in the network and how much network recovery is achievable, whereas O_v determines the efficiency of node distribution between clusters.

3.5.4 Results of the network design algorithm

Designing the best topology of a WSN involves a collection of performance factors that are connected to one another, and the major goal is to find a compromise between these parameters in order to fulfill application requirements. Deterministic strategies are ineffective in these applications; hence, the use of GA as a deductive method for creating the topology with the highest performance was proposed and implemented. The network's performance parameters are determined using a model of WSNs with hierarchical organization and homogenous nodes. Performance factors are incorporated as heavily weighted by fitness function variables, organized by influence on coverage, energy efficiency, and connection parameters. Each performance characteristic can be allocated a weight based on its significance in a specific application. The GA is used to minimize the fitness function in order to generate performing solutions. The weights of each parameter in the WSN design process are assigned randomly at the start based on the

application requirements. Their work is completed by testing and eliminating inefficient scenarios for the application. Table 12 displays the performance parameter symbols and their expected values from the GA. After simulating several topologies with optimal performance, 100 individual tests were done for each combination instance, and the mean results are given in Table 13. The first test involves all parameters having the same unit weight. Following that, necessary actions were made, raising or reducing the weight value based on the outcomes of the specified parameters. It is repeated in the following scenarios until the combination that produces the best answer is discovered. Except for the maximum and lowest energy, which have no effect at the start of the simulation, all weight coefficients are units at the start. Because coverage is a priority in our application, the value 0.3 of U_S does not match our criterion. As a result, in the second situation, U_S value is 2, enhancing its influence on fitness. There is now better coverage and minor adjustments to other metrics, but O_v is still high and not optimum for our situation, indicating that sensor nodes are distributed inefficiently among clusters. As a result, O_v is set to 2 to restrict the value of overlaps. The coverage condition is somewhat modified based on the results of the next instance, and the overlaps criterion is fulfilled in this situation.

Table 12: Performance parameters with their respective weight coefficients

Performance parameter	Weight coefficient	Objective of GA
Uncovered surface (U_s)	A_1	Min.
Sensor nodes density for CH (S_pC)	A_2	Max.
Avarage energy (E_{mA})	A_3	Min.
Number of non-connected nodes *(SOR)*	A_4	Min.
Number of overlaps (O_v)	A_5	Min.
Maximum energy (E_{max})	A_6	Max.

Minimal energy (E_{min})	A_7	Min.
Lifetime (L_T)	A_8	Max.

Table 13: Results of combinations of weights coefficients

No.	Weight coefficient								Parameters of performance				
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	U_s	L_T	E_R	ND	O_v
1	1	1	1	1	1	0	0	1	0.3	43	614	38	0.4
2	2	1	1	1	1	0	0	1	0	42	621	39	0.6
3	2	1	1	1	2	0	0	1	0.1	40	631	38	0.1
4	2	2	1	1	2	0	0	1	0.2	37	662	41	0.5
5	2	0.5	1	1	2	0	0	1	0.3	41	618	33	0.2
6	2	0.5	1	1	2	0	0	2	0.5	45	577	33	0.6
7	2	0.1	1	1	2	0	0	2	1	48	446	19	0.7
8	2	1	1	1	3	0	0	1	0.2	39	639	37	0
9	2	1	2	1	2	0	0	1	0.2	40	621	34	0.1
10	2	0.5	2	1	2	0	0	1	0.5	43	494	18	0.3
11	2	1.5	1	1	0.5	0	0	1	0	42	631	43	1.1
12	2	0.5	1	1	1.5	0	0	1	0.1	42	606	34	0.5
13	2	1.5	0.5	1	1	0	0	1	0	40	641	41	0.4
14	2	0.5	1.5	1	1	0	0	1	0	46	478	21	1.1
15	2	1	0.5	1	1.5	0	0	1	0.1	40	635	39	0.2
16	2	1	1.5	1	0.5	0	0	1	0	43	606	38	1.8
17	2	1	1	1	2	0	0	1	0.4	44	595	36	0.4
18	2	1	1	1	2	0	1	1	0.2	40	636	38	0.1
19	2	1	1	1	2	0.5	0	1	0.2	42	612	36	0.1
20	2	1	1	1	2	0	0.5	1	0.1	40	632	37	0.2

Changing the weight value of S_pC or the maximum power parameter is one way to optimize sensor node dispersion. The algorithm will prefer to generate topologies with a uniform distribution and fewer CH. Thus, the strategy for selecting the optimal coefficient combination for the network under consideration is implemented in this manner. According to the simulation, the instance 19 is the combination of weights that offers the most effective parameter compromise for obtaining a full-coverage WSN. As can be observed, unlike in previous circumstances, it is set to a relatively low

value of A_6; as a result, the number of sensor node connections per CH is limited. This weight combination allows for an increase in L_T and a more efficient use of energy resources. The clustering of sensor nodes is particularly efficient for full-coverage of the environment, consistent power consumption amongst CHs, and avoiding overlaps. Table 14 summarizes all of the dimensions and characteristics of ideal case 19, based on an average test of 100 cases.

Table 14: The average parameters of the most optimal topology

Network parameter	Values
Number of CH	7
Number of HRN	18
Number of LRN	24
Number of IN	51
Uncovered surface (U_s)	0.2
Sensor nodes density for CH (S_pC)	6
Number of overlaps (O_v)	0.19
Maximum energy (E_{max})	24
Minimal energy (E_{min})	1
Lifetime (L_T)	42
Average energy remaining after disconnection of the network	611.3
ND before network disconnection	42
ND after network disconnection	36

Based on the provided topology findings, it is clear that just 49 active sensors out of 100 are required for comprehensive coverage of the surroundings. The nodes are distributed around the environment in seven clusters, with six sensor nodes each CH. Because there is no overlap, the distribution of sensor nodes per CH is fairly consistent and efficient (0.19). Furthermore, homogeneous sensor node distribution allows for balanced power consumption between CH and nodes with greater power consumption.

In comparison to other investigated scenarios, the network with the produced topology has optimum L_T since it can function for 42 cycles before being disconnected. GA are logical strategies that seek a global maximum or minimum in a space with several options. As a result, the greatest answer in our instance would be to keep things as simple as possible. To determine whether we truly constructed the topology with the best performance for the given circumstance, we must examine the algorithm's performance. The advancement of the fitness value of the people formed throughout the implementation of a deductive algorithm may be used to evaluate its performance. This assessment technique may be insufficient to avoid or identify premature convergence, and hence monitoring of specific application parameters throughout algorithm implementation is required to determine whether or not the parameter values converge to appropriate values for the application. It is sufficient to assess the advancement or regression of performance parameters for the topology of a WSN to establish if the employed algorithm is effective in developing a viable solution or not. In most circumstances, if the algorithm converges early, we are working with a restricted search space. To avoid it, simple actions such as raising the pace or modifying the mechanism of mutation, changing the method of recombination, choosing individuals for recombination, increasing the number of people per population, and executing the algorithm over more generations can be taken. Figure 32 depicts the minimizing of the number of overlaps (O_v), as expected in Table 12.

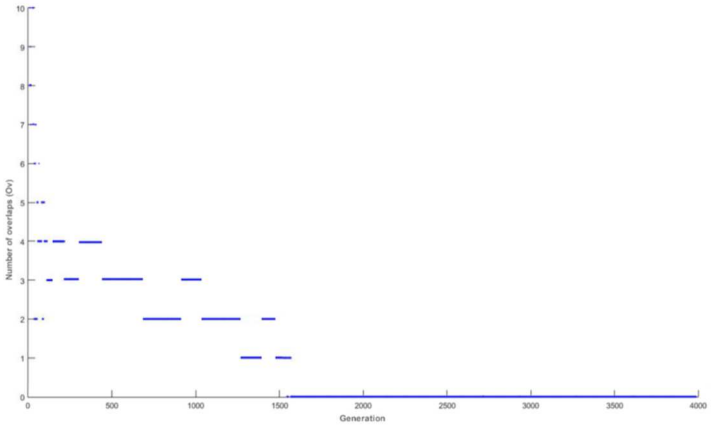


Figure 32: Number of overlaps (O_v)

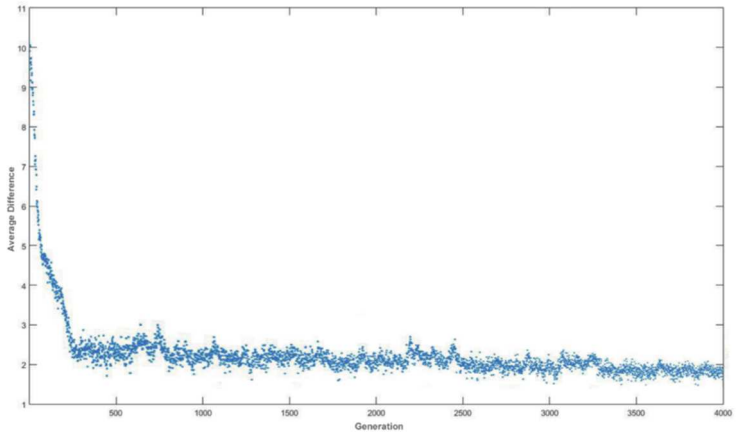


Figure 33: Average difference of the value of fitness function between nodes during the application of GA.

The selection and setup of genetic procedures are carried out experimentally using evidence based on performance parameter progress. Figure 32 depicts the minimization of the number of overlaps (O_v), as anticipated in Table 12,

during the algorithm's execution using the combination number 19 of the weights (Table 13). Fig. 32 results confirm once again the results given in Table 14, where the average number of overlaps (O_v) is 0.19. We have a convergence of population individuals difference for the fitness function based on the findings of the GA applied to the design of a WSN, as shown in Fig. 33. The value of this difference is very high at the beginning of the algorithm, when the first generation is generated randomly, indicating that randomly generated solutions are far from the optimal required solution, and the average difference in the value of fitness function among individuals in the population is quite high. After a few dozen generations, the algorithm begins to converge towards more acceptable solutions with lower fitness function values. However, even after convergence, the difference between individuals in the population remains large, implying that the search space is large enough to allow for the production and selection of the best solutions by genetic operations. The algorithm has improved over generations until it reaches a point when the difference between people in fitness value reduces and there is no improvement in the fitness function's value of the best individual. At this point, we can declare with certainty that the population has converged and we are either very near to or have found the best feasible answer. Observing the degree of difference between people and the advancement of the average fitness value is a rather good way of judging whether or not we have discovered the best feasible answer. However, this strategy is frequently insufficient since we lack data on the progress of other metrics. As far as we know, fitness value advancement can also result from the improvement of a single parameter of the fitness function with a high

weight ratio in respect to other parameters, while other parameters may not change or may deteriorate. Because network characteristics are directly reliant on the application, WSN design is a process that needs consideration of both application requirements and wireless sensor network restrictions. The distribution of joints in the environment, as well as the status of their functioning and cluster structure, has a significant influence on the efficiency of communication functions, environmental monitoring, and energy usage. The network topology design process has an impact on network performance and must be completed before it can be implemented in the environment. Due to environmental limits and needs, designing WSN's is sometimes a difficult task that entails striking a balance between opposing performance factors. After the 3000'th generation, the average fitness value and network performance characteristics in successive populations stalled and stayed constant, with very minor variations in some situations. As a result, the genetic algorithm is programmed to end after 4000 generations.

3.6 Conclusion

The design of a WSN homogeneous network with hierarchical structure is demonstrated in this third chapter, with the priority of covering an environment with minimal cost, high connection, and maximum longevity. Designing a network necessitates determining the best compromise between performance parameters, with the priority of each parameter regulated by the weight coefficients in the fitness function. We proved that our system can find the most effective weight combination through continuous testing and case selection, resulting in a topology with network parameter values that fulfill

the application constraints. Finally, the chosen weight combination may be used to generate the most performable topology conceivable. The optimization of communication between nodes can be simulated in future efforts. The design criterion may involve the selection and assessment of routing algorithms' efficiency. This is possible by implementing a network performance simulation and evaluation function for specific hierarchical routing protocols. Another feature that may be included is the simulation of a network breakdown and the testing of the performance of several network recovery mechanisms in order to pick the best one.

List of Publication

1. G. Ciattaglia , A. De Santis , D. Disha , S. Spinsante , P. Castellini , E. Gambi. Performance Evaluation of Vibrational Measurements Through mmWave Radars. 2020 IEEE 7th International Workshop on Metrology for AeroSpace (MetroAeroSpace). Proceedings, art. no. 9160237, pp. 160-165. DOI: 10.1109/MetroAeroSpace48742.2020.9160237

2. E., Zanaj, E. Gambi, B. Zanaj, D. Disha. Customizable Hierarchical Wireless Sensor Networks Based on Genetic Algorithm. International Journal of Innovative Computing, Information and Control, Volume 16, Number 5, October 2020, ICIC International c 2020 ISSN 1349-4198.

3. E. Zanaj, B. Zanaj, E. Gambi, D. Disha. Simulation of UWSNs performance evaluation. International Journal of Emerging Trends in Engineering Research. Volume 8. No. 7, July 2020, ISSN 2347-3983, https://doi.org/10.30534/ijeter/2020/48872020

4. Zanaj E, Gambi E, Zanaj B, Disha D, Kola N. Underwater Wireless Sensor Networks: Estimation of Acoustic Channel in Shallow Water. Applied Sciences. 2020; 10(18):6393. https://doi.org/10.3390/app10186393

5. E. Zanaj, D. Disha, S. Spinsante and E. Gambi, "A Wearable Fall Detection System based on LoRa LPWAN Technology," in Journal of Communications Software and Systems, vol. 16, no. 3, pp. 232-242, July 2020, doi: 10.24138/jcomss. v16i3.1039

6. G. Ciattaglia, A. De Santis, D. Disha , S. Spinsante , P. Castellini , E. Gambi. Performance Evaluation of Vibrational Measurements Through mmWave Radars. 2021 MDPI (Multidisciplinary Digital Publishing Institute) Remote Sensing, 13 (1), art. no. 98, pp. 1-20.

7. L. Senigagliesi, G. Ciattaglia, D. Disha, E. Gambi. Classification of Human Activities based on Automotive Radar Spectral Images Using

Machine Learning Techniques: A Case Study. 2022 IEEE Radar Conference, Doi:10.1109/RadarConf2248738.2022.9764217. pp. 1-6, (RadarConf22)

8. G. Ciattaglia, A. De Santis, D. Disha, E. Gambi. Patients' behavior monitoring inside a hospital garden: comparison between RADAR and GPS solutions. 8th EAI International Conference, HealthyIoT 2021: IoT Technologies for Health Care pp 139–152, https://doi.org/10.1007/978-3-030-99197-5_12

9. D. Disha, L. Senigagliesi, E. Gambi. Comparison of deep learning techniques for human activity classification based on feature extraction from range-Doppler maps. The 1st International Conference on Information Technologies and Educational Engineering (ICITEE21). ISBN: 978-9928-329-52-3

10. Iadarola G., D., Disha, De Santis, A. Spinsante, S., Gambi, E. Global Positioning System measurements: comparison of IoT wearable devices. 2022 IEEE 9th International Workshop on Metrology for AeroSpace (MetroAeroSpace), DOI: 10.1109/MetroAeroSpace54187.2022.9855994

11. Ciattaglia, G., Senigagliesi, L., Disha, D., De Santis, A., Gambi, E. Experimental Evaluation of Mutual Interference in Automotive Radars. 2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring), DOI: 10.1109/VTC2022 Spring54318.2022.9861024

Bibliography

[1] J. Hampton, Introduction to MIMO Communications. Cambridge University Press New York, USA, 2014.

[2] T. Kinnunen and H. Li, “An overview of text-independent speaker recognition: From features to supervectors,” Speech Communication, vol. 52, pp. 12–40, 2010.

[3] G. Kumar and P. K. Bhatia, “A detailed review of feature extraction in image processing systems,” 2014 Fourth International Conference on Advanced Computing Communication Technologies, pp. 5–12, February 2014.

[4] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” Mathematics of Control, Signals, and Systems, vol. 2, pp. 303–314, 1989.

[5] K. Hornik and H. Stinchcombe, M.and White, “Multilayer feedforward networks are universal approximators,” Neural Networks, vol. 2, pp. 359–366, 1989.

[6] J. Schmidhuber, “Deep learning in neural networks: An overview,” Neural networks : the official journal of the International Neural Network Society, vol. 61, pp. 85–117, 2015.

[7] K. Fukushima, “Neocognitron: A hierarchical neural network capableof visual pattern recognition,” Neural Networks, vol. 1, pp. 119–130, 1988.

[8] C. Cortes and V. Vapnik, “Support-vector networks,” Machine Learning, vol. 20, pp. 273–297, 1995.

[9] G. Hinton, S. Osindero, and Y. Teh, “A fast learning algorithm for deep belief nets,” Neural computation, vol. 18, pp. 1527–54, August 2006.

[10] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet classification with deep convolutional neural networks,” Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105, 2012.

[11] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv: 1409.1556, 2014.

[12] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, “A survey on deep learning techniques for image and video semantic segmentation,” Applied Soft Computing, vol. 70, 05 2018.

[13] K. Zinal and R. Monali, “A review: Object detection using deep learning,” International Journal of Computer Applications, vol. 180, pp. 46–48, 03 2018.
[14] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. Dauphin, “Convolutional sequence to sequence learning,” International Conference on Machine Learning, 2017.
[15] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional neural networks for speech recognition,” IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 22, no. 10, pp. 1533–1545, October 2014.
[16] I. Wallach, M. Dzamba, and A. Heifets, “Atomnet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery,” arXiv: 1510.02855, 2015.
[17] S. Zhang, L. Yao, and A. Sun, “Deep learning based recommender system: A survey and new perspectives,” arXiv: 1707.07435, 2017.
[18] A. Borovykh, S. Bohte, and C. Oosterlee, “Conditional time series forecasting with convolutional neural networks,” arXiv: 1703.04691, 2017.
[19] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” Journal of Physiology, vol. 160, 1962.
[20] Y. Tang, “Deep learning using linear support vector machines,” 2013.
[21] L. Bottou, F. Curtis, and J. Nocedal, “Optimization methods for largescale machine learning,” SIAM Review, vol. 60, pp. 223–311, 2018.
[22] N. Qian, “On the momentum term in gradient descent learning algorithms,” Neural networks : the official journal of the International Neural Network Society, vol. 12, no. 1, pp. 145–151, 1999.
[23] D. Rumelhart, G. Hinton, and R. Williams, “Learning representations by back-propagating errors,” Nature, vol. 323, pp. 533–536, 1986.
[24] D. J. Lary, A. H. Alavi, A. H. Gandomi, and A. L. Walker, “Machine learning in geosciences and remote sensing,” Geoscience Frontiers, vol. 7, no. 1, 2016.
[25] D. Callaghan, J. Burger, and A. K. Mishra, “A machine learning approach to radar sea clutter suppression,” 2017 IEEE Radar Conference (RadarConf), pp. 1222–1227, May 2017.

[26] G. Lopez-Risueno, J. Grajal, and R. Diaz-Oliver, “Target detection in sea clutter using convolutional neural networks,” Proceedings of the 2003 IEEE Radar Conference, pp. 321–328, May 2003.
[27] Metcalf, S. D. Blunt, and B. Himed, “A machine learning approach to cognitive radar detection,” pp. 1405–1411, May 2015.
[28] Q. Zhao and J. C. Principe, “Support vector machines for sar automatic target recognition,” IEEE Transactions on Aerospace and Electronic Systems, vol. 37, no. 2, pp. 643–654, April 2001.
[29] A. Turlapaty and Y. Jin, “Parameter estimation and waveform design for cognitive radar by minimal free-energy principle,” 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 6244–6248, May 2013.
[30] E. Mason, B. Yonel, and B. Yazici, “Deep learning for radar,” 2017 IEEE Radar Conference (RadarConf), pp. 1703–1708, May 2017.
[31] A. M. Elbir, K. V. Mishra, and Y. C. Eldar, “Cognitive radar antenna selection via deep learning,” arXiv: 1802.09736, 2018.
[32] F. Seleim, A.and Paisana, J. A. Arokkiam, Y. Zhang, L. Doyle, and L. A. DaSilva, “Spectrum monitoring for radar bands using deep convolutional neural networks,” 2017 IEEE Global Communications Conference, pp. 1–6, 2017.
[33] T. Wheeler, M. Holder, H. Winner, and M. J. Kochenderfer, “Deep stochastic radar models,” 2017 IEEE Intelligent Vehicles Symposium (IV), pp. 47–53, 2017.
[34] S. Chen, H. Wang, F. Xu, and Y. Jin, “Target classification using the deep convolutional networks for sar images,” IEEE Transactions on Geoscience and Remote Sensing, vol. 54, no. 8, pp. 4806–4817, Aug 2016.
[35] Y. Zhou, H. Wang, F. Xu, and Y. Jin, “Polarimetric sar image classification using deep convolutional neural networks,” IEEE Geoscience and Remote Sensing Letters, vol. 13, no. 12, pp. 1935–1939, Dec 2016.
[36] X. He, N. Tong, and X. Hu, “Automatic recognition of isar images based on deep learning,” pp. 1–4, Oct 2016.
[37] V. C. Chen, The micro-Doppler effect in radar. Artech House, 2011.
[38] Y. Kim and T. Moon, “Human detection and activity classification based on micro-Doppler signatures using deep convolutional neural networks,” IEEE Geoscience and Remote Sensing Letters, vol. 13, no. 1, pp. 8–12, Jan 2016.

[39] M. S. Seyfioglu, A. M. Ozbayoglu, and S. Z. Gurbuz, “Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities,” IEEE Transactions on Aerospace and Electronic Systems, vol. 54, no. 4, pp. 1709–1723, August 2018.
[40] Y. Kim and B. Toomajian, “Hand gesture recognition using microDoppler signatures with convolutional neural network,” IEEE Access, vol. 4, pp. 7125–7130, 2016.
[41] B. K. Kim, H. Kang, and S. Park, “Drone classification using convolutional neural networks with merged Doppler images,” IEEE Geoscience and Remote Sensing Letters, vol. 14, no. 1, pp. 38–42, Jan 2017.
[42] J. Martinez, D. Kopyto, M. Schuetz, and M. Vossiek, “Convolutional neural network assisted detection and localization of UAVs with a narrowband multi-site radar,” 2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM), pp. 1–4, April 2018.
[43] J. Martinez and M. Vossiek, “Deep learning-based segmentation for the extraction of micro-Doppler signatures,” 2018 European Radar Conference (EURAD), 09 2018.
[44] J. Lombacher, M. Hahn, J. Dickmann, and C. Wohler, “Potential of radar for static object classification using deep learning methods,” 2016 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility, pp. 1–4, 2016.
[45] C. Grimm, T. Breddermann, R. Farhoud, T. Fei, E. Warsitz, and R. Haeb-Umbach, “Discrimination of stationary from moving targets with recurrent neural networks in automotive radar,” 2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility, pp. 1–4, 2018.
[46] J. Martinez, R. Prophet, J. C. Fuentes, R. Ebelt, M. Vossiek, and I. Weber, “Identification of ghost moving targets in automotive scenarios with deep learning,” 2019 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility, pp. 1–4, April 2019.
[47] K. N. Parashar, M. C. Oveneke, M. Rykunov, H. Sahli, and A. Bourdoux, “Micro-Doppler feature extraction using convolutional autoencoders for low latency target classification,” 2017 IEEE Radar Conference (RadarConf), pp. 1739–1744, 2017.

[48] M. S. Seyfioglu and S. Z. Gurbuz, "Deep neural network initialization methods for md classification with low training sample support," IEEE Geoscience and Remote Sensing Letters, vol. 14, pp. 2462–2466, 2017.
[49] J. Park, J. Rios, T. Moon, and Y. Kim, "Micro-Doppler based classification of human aquatic activities via transfer learning of convolutional neural networks," Sensors, vol. 24, 2016.
[50] J. Ding, B. Chen, H. Liu, and M. Huang, "Convolutional neural network with data augmentation for sar target recognition," IEEE Geoscience and Remote Sensing Letters, vol. 13, no. 3, pp. 364–368, March 2016.
[51] Y. Wang, Q. Liu, and A. E. Fathy, "Cw and pulse–doppler radar processing based on fpga for human sensing applications," IEEE Transactions on Geoscience and Remote Sensing, vol. 51, no. 5, pp. 3097–3107, 2012.
[52] Y. S. Koo, L. Ren, Y. Wang, and A. E. Fathy, "Uwb microdoppler radar for human gait analysis, tracking more than one person, and vital sign detection of moving persons," in 2013 IEEE MTT-S International Microwave Symposium Digest (MTT). IEEE, 2013, pp. 1–4.
[53] R. Trommel, R. Harmanny, L. Cifola, and J. Driessen, "Multi-target human gait classification using deep convolutional neural networks on micro-doppler spectrograms," in 2016 European Radar Conference (EuRAD). IEEE, 2016, pp. 81–84.
[54] H. G. Doherty, R. A. Burgueño, R. P. Trommel, V. Papanastasiou, and R. I. Harmanny, "Attention-based deep learning networks for identification of human gait using radar micro-doppler spectrograms," International Journal of Microwave and Wireless Technologies, vol. 13, no. 7, pp. 734–739, 2021.
[55] S. K. Mitra and Y. Kuo, Digital signal processing: a computer-based approach. McGraw-Hill New York, 2006, vol. 2.
[56] L. Cohen, Time-frequency analysis. Prentice hall, 1995, vol. 778.
[57] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 374, no. 2065, p. 20150202, 2016.
[58] R. Bro and A. K. Smilde, "Principal component analysis," Analytical methods, vol. 6, no. 9, pp. 2812–2831, 2014.

[59] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip et al., "Top 10 algorithms in data mining," Knowledge and information systems, vol. 14, no. 1, pp. 1–37, 2008.
[60] J. C. Bezdek, S. K. Chuah, and D. Leep, "Generalized k-nearest neighbor rules," Fuzzy Sets and Systems, vol. 18, no. 3, pp. 237–256, 1986.
[61] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," IEEE transactions on neural networks, vol. 12, no. 2, pp. 181–201, 2001.
[62] E. Gambi, G. Ciattaglia, A. De Santis, and L. Senigagliesi, "Millimeter wave radar data of people walking," Data in brief, vol. 31, p. 105996, 2020.
[63] L. Senigagliesi, G. Ciattaglia, A. De Santis, and E. Gambi, "People walking classification using automotive radar," Electronics, vol. 9, no. 4, p. 588, 2020.
[64] J. Bryan, J. Kwon, N. Lee, and Y. Kim, "Application of ultra-wide band radar for classification of human activities," IET Radar, Sonar & Navigation, vol. 6, no. 3, pp. 172–179, 2012.
[65] S. Björklund, H. Petersson, and G. Hendeby, "Features for micro-doppler based activity classification," IET radar, sonar & navigation, vol. 9, no. 9, pp. 1181–1187, 2015.
[66] L. Senigagliesi, G. Ciattaglia, and E. Gambi, "Contactless walking recognition based on mmwave radar," in 2020 IEEE Symposium on Computers and Communications (ISCC). IEEE, 2020, pp. 1–4.
[67] K. Simonyan and A. Zisserman, "Very deep convolutional networks for largescale image recognition," arXiv preprint arXiv:1409.1556, 2014.
[68] M. Mahdianpari, B. Salehi, M. Rezaee, F. Mohammadimanesh, and Y. Zhang, "Very deep convolutional neural networks for complex land cover mapping using multispectral remote sensing imagery," Remote Sensing, vol. 10, no. 7, p. 1119, 2018.
[69] S. L. Rabano, M. K. Cabatuan, E. Sybingco, E. P. Dadios, and E. J. Calilung, "Common garbage classification using mobilenet," in 2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM). IEEE, 2018, pp. 1–4.
[70] http://https://www.image-net.org/

[71] P. Maji and R. Mullins, “On the reduction of computational complexity of deep convolutional neural networks,” Entropy, vol. 20, no. 4, p. 305, 2018.
[72] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.
[73] Nash, Will & Drummond, Tom & Birbilis, Nick. (2018). A review of deep learning in the study of materials degradation. npj Materials Degradation. 2. 10.1038/s41529-018-0058-x.
[74] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, A survey on sensor networks, IEEE Communications Magazine, 2002.
[75] L. C. Tagliabue, F. R. Cecconi, N. Moretti, S. Rinaldi, P. Bellagente and A. L. C. Ciribini, Security assessment of urban areas through a GIS-based analysis of lighting data generated by IoT sensors, Appl. Sci., vol.10, no.6, DOI: 10.3390/app10062174, 2020.
[76] A. Pedersen and T. F. Brustad, A study on hybrid sensor technology in winter road assessment, Safety, vol.6, no.1, DOI: 10.3390/safety6010017, 2020.
[77] M. Al-Zubaidie, Z. Zhang and J. Zhang, REISCH: Incorporating lightweight and reliable algorithms into healthcare applications of WSNs, Appl. Sci., vol.10, no.6, DOI: 10.3390/app10062007, 2020.
[78] A. Kristoffersson and M. Lind´en, A systematic review on the use of wearable body sensors for health monitoring: A qualitative synthesis, Sensors, vol.20, no.5, DOI: 10.3390/s20051502, 2020.
[79] D. M. Doolin and N. Sitar, Wireless sensors for wildfire monitoring, Proc. of SPIE on Smart Structures & Materials, pp.477-484, 2005.
[80] L. Tang, Z. Lu and B. Fan, Energy effcient and reliable routing algorithm for wireless sensors networks, Appl. Sci., vol.10, no.5, DOI: 10.3390/app10051885, 2020
[81] Z. Nazari, M. Nazari and D. Kang, A bottom-up hierarchical clustering algorithm with intersection points, International Journal of Innovative Computing, Information and Control, vol.15, no.1, pp.291-304, 2019.
[82] M. A. Sayeed and R. Shree, Optimizing unmanned aerial vehicle assisted data collection in cluster based wireless sensor network, ICIC Express Letters, vol.13, no.5, pp.367-374, 2019.
[83] J. Ma, S. Shi, X. Gu and F. Wang, Heuristic mobile data gathering for wireless sensor networks via trajectory control, International Journal

of Distributed Sensor Networks, vol.16, no.5, DOI: 10.1177/1550147720907052, 2020.
[84] L. J. G. Villalba, A. L. S. Orozco, A. T. Cabrera and C. J. B. Abbas, Routing protocols in wireless sensor networks, Sensors, vol.9, no.11, pp.8399-8421, 2009.
[85] J. D´ıez-Gonz´alez, R.´Alvarez, N. Prieto-Fern´andez and H. Perez, Local wireless sensor networks positioning reliability under sensor failure, Sensors, vol.20, no.5, DOI: 10.3390/s20051426, 2020.
[86] J. Elson and A. Parker, Tinker: A tool for designing data-centric sensor networks, IPSN'06, TN, USA, 2006.
[87] R. Ramadan, K. F. Abdelghany and H. El-Rewin, SensDep: A design tool for the deployment of heterogeneous sensing devices, The 2nd IEEE Workshop on Dependability and Security in Sensor Networks and Systems (DSSNS), 2006.
[88] S.-C. Wang, Y.-L. Lin, M.-L. Chiang and H.-H. Pan, Improve the stability of the Internet of things using dynamic load balancing clustering, International Journal of Innovative Computing, Information and Control, vol.16, no.1, pp.63-76, 2020.
[89] X. Liu, A survey on clustering routing protocols in wireless sensor networks, Sensors, vol.12, no.8, pp.11113-11153, 2012.
[90] D. Bhattacharyya, T. Kim and S. Pal, A comparative study of wireless sensor networks and their routing protocols, Sensors, 2010.
[91] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, Proc. of the 33rd Annual Hawaii International Conference on System Sciences (HICSS), Big Island, HI, USA, pp.3005-3014, 2000.
[92] E. Kotobelli, E. Zanaj, M. Alinci, E. Bum¸ci and M. Banushi, A modified clustering algorithm in WSN, International Journal of Advanced Computer Science and Applications (IJACSA), vol.6, 2015.
[93] J.-L. Liu and C. V. Ravishankar, LEACH-GA: Genetic algorithm-based energy-efficient adaptive clustering protocol for wireless sensor networks, International Journal of Machine Learning and Computing (IJMLC), vol.1, no.1, pp.79-85, 2011. 1638
[94] S. Bayraklı and S. Z. Erdogan, Genetic algorithm based energy efficient clusters (GABEEC) in wireless sensor networks, Procedia Computer Science, vol.10, pp.247-254, 2012.

[95] P. Nayak and B. Vathasavai, Genetic algorithm based clustering approach for wireless sensor network to optimize routing techniques, The 7th International Conference on Cloud Computing, Data Science & Engineering – Confluence, 2017.
[96] A. Rahmanian, H. Omranpour, M. Akbari and K. Raahemifar, A novel genetic algorithm in LEACHC routing protocol for sensor networks, Proc. of the 24th Canadian Conference on Electrical and Computer Engineering, Niagara Falls, Ontario, Canada, 2011.
[97] Van der Maaten, Laurens, and Geoffrey Hinton. "Visualizing data using t-SNE." Journal of machine learning research 9.11 (2008).
[98] Joyce, J.M. (2011). Kullback-Leibler Divergence. In: Lovric, M. (eds) International Encyclopedia of Statistical Science. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04898-2_327
[99] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747 (2016).
[100] L. Senigagliesi, G. Ciattaglia, D. Disha, E. Gambi. Classification of Human Activities based on Automotive Radar Spectral Images Using Machine Learning Techniques: A Case Study. 2022 IEEE Radar Conference, doi:10.1109/RadarConf2248738.2022.9764217. pp. 1-6, (RadarConf22)

Appendix A.

GA Configuring

A.1. Parameters of “options” structure for configuring the GA

- Value (option) – Description

- PopulationType (bitstring) – The binary row determines the kind of genome on which the GA will be applied.

- Generations (4000) – Determine the maximum number of iterations or generations the GA can have before it is terminated. After 4000, the algorithm will be terminated.

- FitnessScalingFcn (fitscalingprop) – Determines the mechanism of scaling individuals in the population depending on fitness values. In this scenario, the selectivity of the selection is proportional to the fitness value.

- SelectionFcn (selectionstochunif) – Determine the recombination selection function of individuals. The approach of universal stochastic selection was used in this circumstance.

- CrossoverFcn (crossoverscattered) – A genetic recombination operation is chosen for the creation of individuals from the following population.

- MutationFcn (mutationgaussian) – Determines the mutation approach employed.

- StallGenLimit (4000) – Terminate the GA if there is no progress in the population's average fitness value after a given number of generations, 4000 in this example.

- StallTimeLimit (10000) – Terminate the GA if there is no improvement in the population's average fitness value after a particular number of seconds, 10000 in this example.

The pseudo code of the fittness function will be described in the following.

3. A.2. Pseudo code of fitness function

Step 1: Decode the genome of the individual m from the population M(t) and construct the matrix of the structure with the data of the positions and states of the nodes in the network;

Step 2: Build the connection matrix depending on the distances of the CH from the sensor nodes, based on the structure matrix;

Step 3: Based on the link matrix evaluate:

a) Sensor nodes density for CH, SpC;

b) Number of non-connected nodes, SOR;

c) Overlap number, Ov;

d) Communication energy for each node;

Step 4: Evaluate the uncovered US surface, testing whether the area of inactive and un- connected nodes is covered by adjacent connected nodes;

Step 5: Build the power matrix for each active node, through the amount of operating and communication energy;

Step 6: Based on the energy matrix estimate:

a) Average energy, EmA;

b) Minimum energy consumed per node, Emin;

c) Maximum energy consumed per node, Emax;

d) Full load network life, LT ;

Step 7: Calculate the fitness value F of the individual m through the weighted sum func-
tion:

$$F = A_1 U_S + (-A_2) S_p C + A_3 E_{mA} + A_4 SOR + A_5 O_v + (-A_6) E_{min} + A_7 E_{max} + (-A_8) L_T \quad (61)$$

Step 8: Repeat the above maps for all individuals' m of the population.