People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
8 May 1945 Guelma University

Faculty of Mathematics, Computer Science and Material Sciences
Department of Computer Science
Laboratory of Information and Communication Sciences and Technologies

DISSERTATION

In view of obtaining
The Doctorate degree in 3rd cycle

Domain: Mathematics and Computer Science. **Field**: Computer Science
Specialty: Computer Science

Presented by:

Roguia SIOUDA

Entitled

Pattern recognition using collaborative neural networks

Defended: 10/11/2022 Before the board of examiners composed of:

Pr. Yacine LAFIFI	**Professor**	Univ. of 8 May 1945, Guelma	Chairman
Pr. Mohamed NEMISSI	**Professor**	Univ. of 8 May 1945, Guelma	Supervisor
Pr. Hamid SERIDI	**Professor**	Univ. of 8 May 1945, Guelma	Co-supervisor
Pr. Nadir FARAH	**Professor**	Univ. of Badji Mokhtar, Annaba	Examiner
Dr. Brahim FAROU	**MCA**	Univ. of 8 May 1945, Guelma	Examiner
Dr. Zineddine KOUAHLA	**MCA**	Univ. of 8 May 1945, Guelma	Examiner

2021-2022

DEDICATIONS

I dedicate my dissertation work to my dear parents, for all their sacrifices, their love, their tenderness, their support, and their prayers throughout my studies,

To my dear sisters, Amel, Karima, and Lamia, for their constant encouragement and moral support,

To my dear brother, Ahmed, for all his encouragement and support,

To all my dear nephews and nieces,

To all my family for their support throughout my university career,

To all my lovely friends,

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Allah for giving me the strength, knowledge, ability, and determination to finalize this thesis.

Throughout the writing of this dissertation, I have received a great deal of support and assistance. Therefore, I would like to thank all the people who participated directly or indirectly in completing this thesis with these few lines.

I would first like to express my great gratitude and my distinguished thanks to my thesis director, Professor **NEMISSI Mohamed** and co-director Professor **SERIDI Hamid**, for the fascinating subject they have allowed me to undertake. I also thank them for the time and patience they have given me over the five years. Their relevant advice and clear and informed interventions have guided me continuously throughout this work.

I would also like to thank Professor **LAFIFI Yacine** of the University of Guelma. He did me the honor of chairing the jury for my doctorate.I would also like to thank Professors **FARAH Nadir** of the University of Annaba, Dr. **FAROU Brahim** and Dr. **KOUAHLA Zineddine** of the University of Guelma, for the honor they have bestowed on me by accepting the responsibility of examining this work and participating in the defense jury.

I would also like to thank all the computer science department teachers of 8 Mai 1945 Guelma university for the quality of their formation. A big thanks to the LabSTIC laboratory teams who contributed, each in their way, and I do not forget the laboratory engineer Miss Kharoubi Madiha.

In addition, I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me. Finally, I would like to express my gratitude to all my family, particularly my brother and sisters, for their trust, support, and love.

Finally, my thanks to my friends and colleagues, especially for the 2017-2018 promotion.

ملـخص

تم تطبيق الشبكات العصبية الاصطناعية بنجاح على مجموعة واسعة من المشاكل. في التعرف على الأنماط ، تم استخدامها في العديد من المهام ، مثل استخراج الميزات وتقليل الأبعاد والتصنيف. في هذا العمل ، نقترح نموذجين لتصنيف نبضات القلب يعتمد على التعاون بين أنواع مختلفة من الشبكات العصبية. الهدف الرئيسي هو الجمع بين خصائصهم التكميلية.

يستخدم النموذج الأول أداة تشفير تلقائية متناثرة مكدسة (SSAE) كمستخرج ميزة ونظام من الإدراك متعدد الطبقات (MLP) كمصنف. في هذا النموذج ، يتم تقسيم المشكلة بأكملها إلى أجزاء أبسط ، والتي يتم حلها باستخدام MLPs مختلفة. يستخدم النموذج الثاني أيضًا SSAE لاستخراج الميزات بالإضافة إلى ميزتين ديناميكيتين أخريين. في هذا النموذج ، يتم التصنيف بواسطة نموذج عصبي هجين يعتمد على الجمع بين الشبكات العصبية العشوائية وRBF.

يتم تقييم النماذج المقترحة في قاعدة بيانات MIT-BIH عدم انتظام ضربات القلب. تعتمد الاختبارات على نموذج ما بين المرضى ، حيث يتم أخذ بيانات التدريب والاختبار من مرضى مختلفين. يتم مقارنة النتائج التي تم الحصول عليها مع بعض الأساليب الحديثة.

الكلمات المفتاحية :التعرف على الأنماط ، التصنيف ، الشبكات العصبية ، التعلم الآلي ، التعلم العميق ، مجموعة بيانات تخطيط القلب.

RÉSUMÉ

Les réseaux de neurones artificiels ont été appliqués avec succès à un large éventail de problèmes. En reconnaissance de formes, ils ont été utilisés dans plusieurs tâches, telles que l'extraction de caractéristiques, la réduction de dimension et la classification. Dans ce travail, nous proposons deux modèles de classification des battements cardiaques de l'ECG, basés sur la collaboration de différents types de réseaux de neurones. L'objectif principal est de combiner leurs propriétés complémentaires.

Le premier modèle utilise un auto-encodeur creux empilé comme extracteur de caractéristiques et un système de plusieurs perceptrons multicouches (PMC) comme classificateur. Dans ce modèle, l'ensemble du problème est divisé en parties plus simples, qui sont résolues à l'aide de différents PMC. Le deuxième modèle utilise également un auto-encodeur creux empilé pour extraire des caractéristiques en plus de deux autres caractéristiques dynamiques. Dans ce modèle, la classification est effectuée par un modèle neuronal hybride basé sur la combinaison de réseaux neuronaux aléatoires et RBF.

Les modèles proposés sont évalués sur la base de données d'arythmie MIT-BIH. Les tests sont basés sur le paradigme inter-patients, dans lequel les données d'entraînement et de test proviennent de différents patients. Les résultats obtenus sont comparés avec certaines des méthodes de l'état de l'art.

Mots clés: Reconnaissance de formes, classification, réseaux de neurones, apprentissage automatique, apprentissage en profondeur, ensemble de données ECG.

ABSTRACT

Artificial neural networks have been successfully applied to a wide range of problems. In pattern recognition, they have been used in several tasks, such as feature extraction, dimension reduction, and classification. In this work, we propose two ECG heartbeat classification models based on collaborating different types of neural networks. The main aim is to combine their complementary properties.

The first model uses a stacked sparse autoencoder (SSAE) as feature extractor and a system of multiple Multi-layered perceptrons (MLP) as a classifier. In this model, the entire problem is divided into simpler parts, which are resolved using different MLPs. The second model also uses a SSAE to extract features in addition to two other dynamic features. In this model, the classification is performed by a hybrid neural model based on combining random and RBF neural networks.

The proposed models are evaluated on the MIT-BIH arrhythmia dataset. The tests are based on the inter-patient paradigm, in which the training and test data are taken from different patients. The obtained results are compared with some of the state-of-the-art methods.

Keywords: Pattern recognition, classification, neural networks, machine learning, deep learning, ECG dataset.

Contents

List of Figures

List of Tables

ABBREVIATIONS

ANN	Artificial Neural Network
RBFNN	Radial Basis Function Neuronal Network
RVFLN	Random Vector Functional Link Network
MLP	Multi-Layer Perceptron
FCM	Fuzzy C-Means
GA	Genetic Algorithms
KNN	K-Nearest Neighbors
KSOM	Kohonen Self-Organized Maps
NB	Naïve Bayes
PCA	Principal Component Analysis
SVM	Support Vector Machines
OAA	One Against All
OAO	One Against One
SMOTE	Synthetic Minority Over-sampling Technique
ECG	Electrocardiogram
SSAEs	Stack Sparse Autoencoders

General introduction

General framework

Current pattern recognition systems require the processing of different problems, each requiring a distinct type of calculation. This has given rise to new hybrid systems that combine various tools to design more robust systems. The main idea behind this new orientation is that judicious cooperation of several complementary tools should result in improved performance. Among these tools, artificial neural networks provide interesting solutions for the many modern problems. In this work, we make use of the complementary properties of different types of neural networks to introduce two models for classifying ECG heartbeats. Indeed, despite recent progress, cardiovascular disease is still responsible for half of all deaths worldwide. Computer-aided systems offer efficient solutions that can aid cardiologists in classifying ECG heartbeats, especially in long-term recordings.

Over the last two decades, several approaches have been developed to automatically classify heartbeats and detect cardiac abnormalities in ECG recordings. These methods generally include three main steps: pre-processing, feature extraction, and classification. Heartbeat classification methods can be divided into two main categories based on how these three steps are performed. The first category includes conventional approaches in which feature extraction and classification are performed separately. In contrast, the second category includes techniques based on deep learning, where feature extraction and classification are performed automatically in the same architecture. The methods of the second category avoid time-consuming and complex feature extraction phases. However, these methods have two major issues. First, they have large architectures that include a lot of hidden layers and neurons. Second, they may ignore some important features. In this work, we aim to combine both methods.

Motivations and Contributions

In this work, we propose two models for classifying ECG heartbeats. The first model is based on a stacked sparse autoencoder (SSAE) as a feature extractor and a system of multiple multilayered perceptrons (MLP) as a classifier. The second model also uses a SSAE to extract features in addition to two other dynamic features. In this model, the classification is performed by a hybrid neural model based on combining random and RBF neural networks.

In the feature extraction phase, both models are based on the autoencoder. The motivation is that it permits defining automatic high-level features without neither pre-processing stage nor expert intervention. This significantly simplifies the feature extraction phase.

In the classification phase, the first model uses a system of multiple MLPs. The aim is to enhance the classification performance by decomposing the original multi-class problem into simpler binary subproblems and solving them using independent MLPs. On the other hand, the second model uses a hybrid neural classifier that combines random and RBF neural networks. Indeed, these two networks have several advantages, for example, Random neural networks provide good generalization and very fast training; RBF neural networks provide high coverage of the input space and allow using prior knowledge.

Thesis structure

This thesis consists of four chapters organized as follows:

1. The first chapter : is devoted to the general concepts of pattern recognition. It presents the application of pattern recognition in computer-aided diagnosis, its general process, and its different approaches. This chapter also presents some common classification and clustering methods.

2. The second chapter : is an introduction to neural networks. It first discusses their modeling principles, their learning, their architecture, and their application in diverse fields. Then, it presents some types of neural networks and describes their different learning rules and structures. Finally, this chapter discusses the benefits and drawbacks of artificial neural networks.

3. The third chapter : presents deep learning principles, learning algorithms, and objectives. Then, it describes the different deep learning techniques and their applications in the medical field. This chapter also presents some common deep neural models.

4. The fourth chapter : is devoted to the proposed models. First, it briefly presents the ECG principles, its waves, intervals, and applications in classification fields. It describes the ECG dataset used to evaluate the proposed models. Then, this chapter presents the main idea and motivation, the training, and the process of the proposed models. Finally, it discusses the obtained results and compares them with some state-of-the-art methods.

   Finally, a conclusion concludes this thesis.

Chapter 1

Pattern recognition

Contents

This chapter is devoted to the general concepts of pattern recognition. It defines pattern recognition, its applications in the medical field, its process, and the most common classifiers. This chapter also describes the validation methods of classification systems.

1.1 Introduction

Pattern recognition (PR) is a crucial subfield of artificial intelligence that focuses on recognizing patterns and regularities in data [9]. This subfield comes from several disciplines: mathematics (probability and statistics), engineering sciences, computer science, and artificial intelligence. Before the 1960s, pattern recognition was theoretical research in statistics, and since the 1960s, it has been defined as a distinct field.

Nowadays, advances in technology have enabled us to imitate some human abilities through the use of machines. The main idea is not to completely replace humans with machines but to delegate to machines some simple recognition tasks that humans do. On the other hand,

the appearance of computers has increased the demand for practical applications of pattern recognition, which has put forward new requirements for further theoretical development.

The problem that pattern recognition seeks to solve is to associate a class with an unknown form. Pattern recognition is often seen as a classification problem, i.e., it tries to find the function that assigns any anonymous form to its most relevant class. It is therefore concerned with the development of intelligent systems for perception and decision-making [78]. Pattern recognition systems integrate the whole perception-recognition series from raw data acquisition to the elaborated understanding of this data.

1.2 Definition of pattern recognition

Many authors have provided simple definitions of pattern recognition. For example, Duda and Hart in 1973 defined it as: "a field concerned with machine recognition of meaningful regularities in noisy and complex environments" and Bezdek in 1981 defined pattern recognition as: "a search for structure in data" [9].

These two definitions reveal the objectives and difficulties of pattern recognition. PR is the learning and discovery of structures named classes in a possibly perturbed dataset. The aim of this type of learning is usually determined by triggering an event or action based on the identity of a specific object in the class

1.3 Application of pattern recognition in computer-aided diagnosis

Computer-aided diagnosis is a significant application of pattern recognition to assist doctors in making diagnostic decisions, whereas the doctor determines the final diagnosis. Computer-aided diagnosis is helpful and valuable for a wide range of medical data, including tomographic and electrocardiograms (ECG). The need for computer-aided diagnosis comes from the fact that medical data is not always easily interpreted, and the interpretation may depend mainly on the doctor's expertise.

Consider, for example, mammography using X-rays for the detection of breast cancer. Even though mammography is now the best method for detecting breast cancer, 10-30% of women

with the disease who undergo mammography have negative results [77]. About two-thirds of these detections with false results, the radiologist failed to detect malignancy, which was evident in retrospect. This failure may be due to poor image quality, the radiologist's ocular tiredness, or the delicate nature of the results. The proportion of correct classifications improves after a second reading by another radiologist. Thus, one can develop a pattern recognition system to provide radiologists with a second opinion. A greater level of confidence in mammography-based diagnostics would reduce the number of suspected breast cancer patients who have to undergo surgical breast biopsy, with its associated complications [77].

1.4 Process of pattern recognition systems

Just as humans use their senses to acquire knowledge from their environment and react to it, a pattern recognition system does the same. In general, a pattern recognition system is divided into two main phases: learning and classification. Before these two phases, a pre-treatment phase is used to correct and delete the noise from the data. A feature extraction phase is utilized to find the minimum set of features necessary to establish the representation space Figure 1.1. With the help of these phases, the pattern recognition system can detect any type of data: signal waveforms, images, biometric data, speech ... etc.

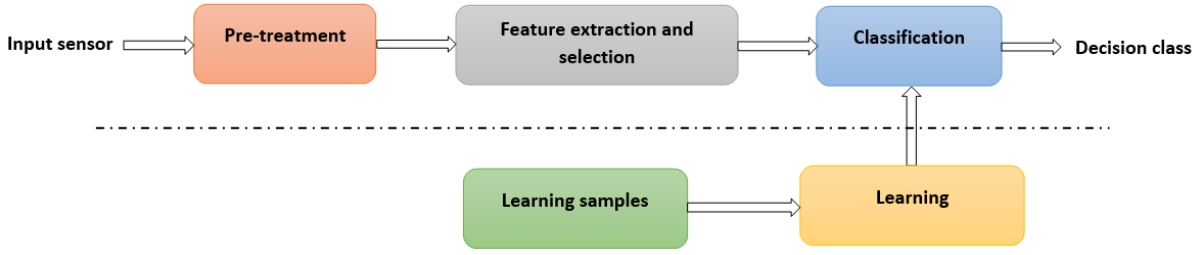


FIGURE 1.1: Flowchart of a pattern recognition system.

1. **Data acquisition** : to make a decision, it is necessary to have sufficient data and information. Sensors frequently provide this data for use in recognition systems [24]. These sensors (microphones, cameras...) convert the signals received from the real world into a discrete digital representation suitable for processing. The operation can be summarized as transforming an analog signal into a digital signal with sufficient quality.

2. **Pre-treatment**: is a sequence of operations that make it possible to reduce the size of data to be treated and eliminate the noise introduced by acquisition tools. Pre-treatment seeks

to keep only important information and eliminate redundancies. This phase uses several algorithms and filters, such as binarization, skeletonization, etc. The new representation space has a smaller dimension but remains ample and contains relatively raw information. Pre-treatment is essential for classifier success, and it differs from one application to another.

3. **Feature extraction and selection** the feature extraction and selection phases aim to provide several features that are important in the classification phase. The most challenging part of this phase is deciding which features are used to get a good classification result. The aim is to find the most robust features possible, the most relevant (to reduce the size of the feature vector), and the least computationally expensive. Depending on the situation, the values of these features can be real, integers, or binary [6]. In this phase, redundant or unnecessary information degrades the classifier performance and should then be removed from the data while valuable content is retained.

   Features can be numerical or categorical. The first is also called quantitative features. The values of this type of feature are numbers. The most commonly used data types are integers and floats. Numerical features can be categorized into discrete and continuous features

   - **Discrete features** are numerical features that represent countable elements. They have a large number of possible values. The list of possible values for discrete features can be fixed or range from 0,1,2 to infinity.
   - **Continuous features** are numerical features that represent measurements. Features can have any value in a range. Therefore, the possible values of continuous features cannot be counted.

   Categorical features are also called qualitative features. This type has only a limited number of possible values. The values represent categories and are usually in text format. The features of this type can be nominal or ordinal.

   (a) **Ordinal features** are categorical features with a sense of order among their values.

   (b) **Nominal features** are categorical features with no concept of order among their values.

   Although the relationship between the error rate, the number of training samples, and the number of features are difficult to establish, it is generally recommended to use a training

set whose size is at least ten times the number of features [35]. On the other hand, reducing the number of features will reduce the discriminative ability of the resulting system and thus reduce the accuracy of the classifier.

Feature extraction usually precedes feature selection: features are first extracted from the initial data, and then some of these features are selected. The choice between feature extraction and selection depends on the application field and the available training set. Selection can reduce measurement costs because some features are discarded, while the remaining features retain their original physical interpretation. On the other hand, the transformed features generated by feature extraction can provide a better quality of distinction, but these new features may not have a clear physical meaning.

4. **Classification**: is the process of categorizing and grouping given objects into categories based on predetermined features. It is the core and most important phase of the pattern recognition systems. In this phase, depending on the parameters obtained during training, the classifier assigns to each unknown object its most probable class. There are two types of classification: supervised and unsupervised. In unsupervised classification, the samples are not labeled a priori, while all samples are pre-labeled in supervised classification.

5. **Post-treatment**: this stage is the final step in the pattern recognition process. It includes some operations performed after classification, such as: merging the outputs of several classifiers, evaluating confidence thresholds, and deciding to classify or reject an object. Other high-level information: lexical, syntactic, and semantic can be used to validate and correct errors in the recognition step and selecting the best solution. Although optional, this phase can significantly improve the quality of recognition [26].

1.5 Approaches to pattern recognition

There are two basic approaches to the implementation of pattern recognition systems: statistical methods and structural methods. Each method uses different techniques to perform description and classification tasks. Hybrid methods, sometimes called unified pattern recognition methods, combine statistical and structural techniques. In the following, we will give a brief presentation of these approaches.

1.5.1 Statistical approach

It is the most commonly used approach in pattern recognition systems because it is easy to manipulate. Statistical pattern recognition systems depend on statistics and probabilities. Here, each pattern is described in terms of feature sets. The feature sets are selected so that different patterns occupy different feature spaces without overlap. The effectiveness of the feature set depends on the degree of separation of patterns of different categories, i.e., there is an appropriate interclass distance. After analyzing the probability distribution of patterns belonging to a specific category, the decision boundary is determined [35]. Here, patterns are projected onto certain pre-processing operations to make them suitable for training purposes. Select the feature after analyzing the training patterns. The system learns from the training patterns and adapts to identify or classify unknown test patterns. Feature measurement is carried out during the test phase, i.e., determining the distance between the patterns in the statistical space and then presenting these feature values to the learning system for classification. The statistical pattern recognition approach deals with features only without considering the relations between them, such as principal component analysis and naïve Bayes.

In general, we will distinguish between two prominent families of methods: parametric methods and non-parametric methods.

- **Parametric methods** are learning methods that represent data with a set of fixed features. They operate under the assumption that the classes studied follow a probability distribution of a particular form known a priori. The decision is to determine the class category with the highest probability of belonging to the anonymous form. They require a sufficiently large learning base to estimate the parameters of the assumed distribution correctly. These methods are easier to understand, interpret and have a fast-learning speed. Some examples are Naïve Bayes and Neural network methods.

- **Non-parametric methods** are suitable when we have a lot of data and no prior knowledge. In non-parametric methods, the number of features is infinite, and the complexity of the methods increases with the amount of training data. In non-parametric methods, the probability distribution of a class is unknown, so one seeks to define decision boundaries between types in the representation space to classify the unknown points through a series of tests. The most commonly used non-parametric methods in pattern recognition are the nearest neighbors and decision trees...

The statistical methods are relatively inexpensive and simple in computation, especially for the parametric approach, and moderately sensitive to noise.

1.5.2 Structural approaches

The structural approach, unlike the statistical approach based on quantitative metrics, ignores the nature of the objects. The latter are represented more realistically in the structural (syntactic) approach, which is based on qualitative measures extracted from the structure of these objects. Although subject to many definitions, the notion of structure nevertheless reveals the existence of a decomposition of the whole (the structure) into parts and of relations between these parts. Structural model recognition methods adopt representations that explicitly reveal the connections between elements of the model. They are more appropriate for dealing with visual structures in which the components are linked by spatial relations [56]. In the structural approach, a class is described by its grammar. A grammar is composed of a set of syntactic rules that determine the set of acceptable forms in this class and which, in principle, have common structural characteristics. This set of forms is called the language generated by grammar [56]. The input space of the structural approach classifier called features is not composed of vectors of real values, but instead of a list of semi-objective features (large, open, pale blue, etc.), of Boolean values or decomposition of the object into idealized elementary parts (vertical line, circle, phoneme, etc.). This category includes decision trees, expert systems, and parsing programs.

1.5.3 Hybrid approaches

Despite their different natures, statistical and structural approaches can be combined into the same application areas. The choice of method may be related to the hardware constraints such as the size of the available learning base, the required computation time, and the required memory size. The combined use of both approaches could be an optimal solution to the pattern recognition problem.

Some approaches combine several techniques, such as Markov models and neural networks, hidden Markov models and k-nearest neighbors, etc.

For example, in [5, 20], the authors used collaboration between statistical and structural approaches. These systems use two levels of interpretation. The first level extracts and then

recognizes the features of the forms using a statistical classifier. The second exploits the topological features of the forms to reconstruct graphs (occlusions or skeletons) for each of the forms and then recognizes the forms using a structural classifier.

1.6 Most common classification methods

1. **k Nearest Neighbours**

   K-Nearest Neighbours method (K-NN) is a simple supervised machine learning algorithm that assumes similarity between the new and available objects and puts the recent object in the most similar category to the general categories. The K-NN algorithm classifies new objects based on similarity measures. This indicates that when new data is generated, it may be quickly classified into a well-suited category using the KNN method [18] (Figure 1.2). The KNN algorithm can be used for both regression and classification but is mainly used for classification tasks. It is also known as a "lazy learner algorithm" because it has no specialized training phase and uses all the data for training during classification.

   The steps of the KNN are as follows:

   Step 1: Define the parameter K, which is the number of the nearest neighbors.

   Step 2: Calculate the distance between the new sample and all the available samples. The common distance measurements are Euclidian, Manhattan, or Minkowski. They are given by:

$$\sqrt{\sum_{i=1}^{K} (x_i - y_i)^2} \tag{1.1}$$

$$\sum_{i=1}^{K} |x_i - y_i| \tag{1.2}$$

$$\left(\sum_{i=1}^{K} (|x_i - y_i|)^q \right)^{\frac{1}{q}} \tag{1.3}$$

   Step 3: Sort the calculated distance and determine the closest neighbor based on the kth shortest distance.

   Step 4: Collect the classes of the nearest neighbors.

   Step 5: Count the most frequent class of the nearest neighbors and return this class as the predicted one. One of the advantages of this algorithm is that it is simple to implement and interpret. It is useful for non-linear data because there are no assumptions about

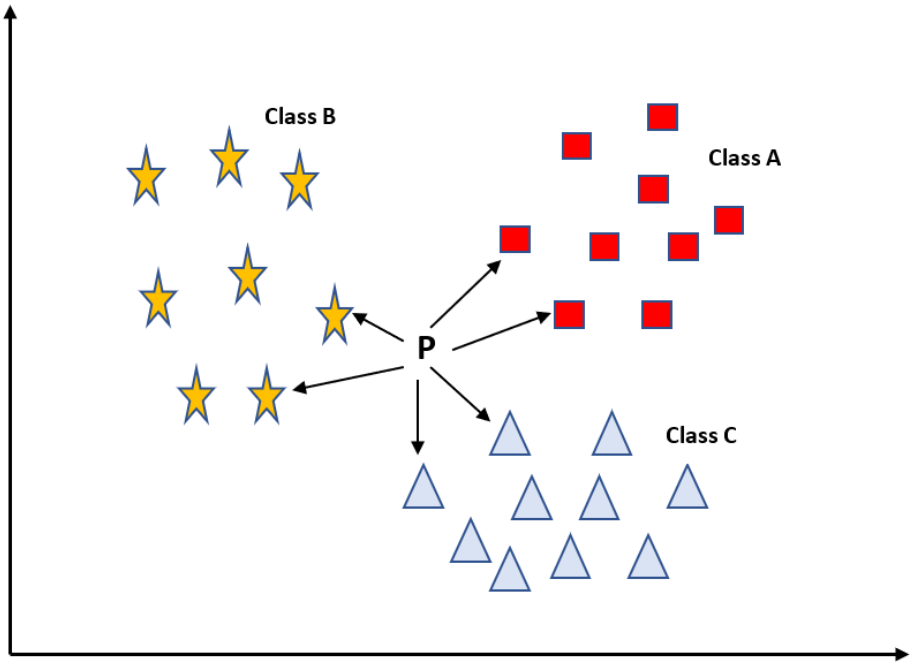


FIGURE 1.2: Classification using the k-nearest neighbor method.

the data. It simply stores all the training samples. K-NN is also very suitable for multi-class classification since its classification decision is based on the neighborhood of similar samples. Among the disadvantages of the nearest neighbors method is that the time required to calculate the similarity is enormous. In practice, it is impossible to implement this algorithm for high dimensions and massive data. The cost of classifying the nearest neighbors is very high. In addition, the memory storage increases as the number of training samples increases. The other most significant problem with KNN is choosing the optimal number of neighbors to consider.

2. **Support vector machines**

The support vector machine (SVM) is one of the supervised machine learning algorithms to solve classification and regression problems. SVM was developed by Vladimir Vapnik in the 1990s.This algorithm is known for its strong theoretical guarantee, high flexibility, and ease of use, even without much data mining knowledge [17]. The main objective of this algorithm is to find a hyperplane that can separate the classes well. Many possible hyperplanes can be selected, but the aim is to find the one with the maximum margin, i.e., the maximum distance between data points of two classes. Maximizing the distance of the margin provides some reinforcement so that future data points can be classified with

greater confidence. The margin is calculated as the vertical distance between the line and the nearest points only. Only these points are relevant for defining the line and building the classifier. These points are named "support vectors". They define the optimal hyperplane Figure 1.3.

There are two types of SVM classifiers: the linear SVM classifier and the non-linear SVM classifier. In the linear classifier model, the samples are assumed to be linearly separable. The SVM predicts a straight hyperplane dividing the two classes. The main objective when setting the hyperplane is to maximize the distance between the hyperplane and the nearest data point of either class. However, in the real world, the data is usually scattered to some extent. To solve this problem, separating the data into different classes based on a straight linear hyperplane cannot be considered a good choice. For this purpose, Vapnik suggested creating a non-linear classifier by applying the kernel trick to maximum margin hyperplanes. In non-linear SVM classification, the data points are transformed into a higher-dimensional space. The advantages of this algorithm are that it is less

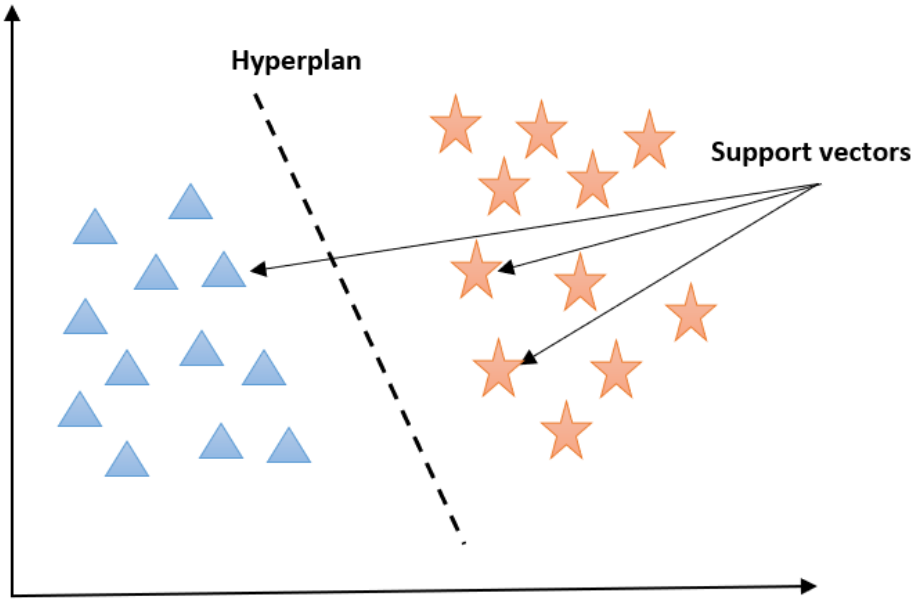


FIGURE 1.3: Classification using support vector machines.

sensitive to overfitting since the complexity of the model is independent of the dimension of the feature space. An SVM can handle a high-dimensional feature space with good classification accuracy. It is effective when the number of features is large enough and is very fast in the classification process. However, the most significant limitation of SVM

is the choice of the kernel. A poor choice of the kernel can lead to an increase in the percentage of errors.

3. **Decision trees**

Decision trees (DT) are a non-parametric supervised learning algorithm used for classification and regression problems, and they can handle both categorical and numerical data. TDs fit complex datasets while allowing the user to see how a decision is done. In this technique, the dataset is divided into two or more homogeneous sets based on the most significant separator in the input variable, and it looks like a tree structure. A decision tree consists of a root node through which data is entered and a leaf node representing a classification or decision Figure 1.4.

DTs can be calculated by supervised learning algorithms that can automatically select discriminating variables from unstructured data. These algorithms also extract logical rules that were not present in the raw data. There are different algorithms for building decision trees, such as ID3, C4.5, CHAID, CART... The main advantage of decision

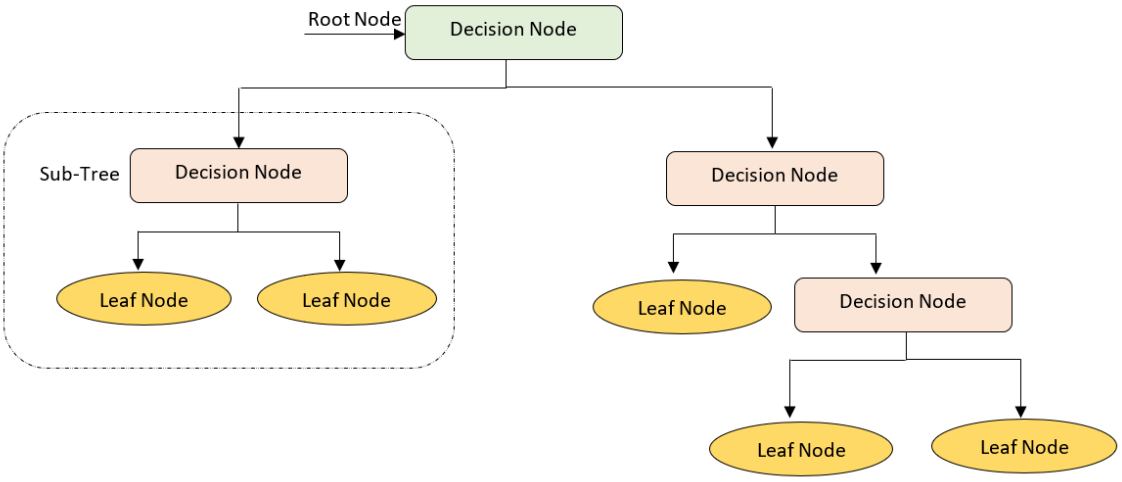


FIGURE 1.4: The main architecture of the decision trees.

trees is that are easy to interpret and understand. They are fast to train and they allow identifying the most significant features and the relationship between them. DTs require less cleaning and pre-processing of data, and they are not affected by outliers and missing values. However, overfitting is one of the most practical difficulties for decision trees. This problem can be solved by defining constraints on the model parameters.

4. **Neural networks**

A neural network is a mathematical model that attempts to reproduce certain functionalities of the human brain, such as parallelism, acquiring knowledge through the learning process, the storage of knowledge, and the ability to use this knowledge. Artificial neural networks are commonly known as universal function approximators. These networks are capable of learning any function, linear or non-linear, complex or simple. ANNs are robust to noise or unreliable data. This work is mainly based on neural networks. Therefore, we devote the next two chapters to neural networks and deep learning.

1.7 Clustering

Clustering is an unsupervised machine learning algorithm. It involves classifying different data points into several groups. Typically, data points in the same group have similar properties or features while data points in different groups have differences in their properties or features. Clustering consists then essentially in collecting objects based on similarity and dissimilarity between them. Clustering algorithms are usually applied in statistical data analysis.

Clustering methods can be divided into two subgroups: (*i*) Hard clustering: in which each data point is either completely assigned to a cluster or unassigned; (*ii*) soft clustering, in which each data point is assigned based on its cluster probability or likelihood.

In machine learning, there are several types of clustering methods.

1. **Centroid based method**

   A centroid-based method is an iterative clustering method in which clusters are formed using the proximity of data points to the cluster centroid. These algorithms are efficient, but they are sensitive to initial conditions and outliers. In this type of method, the number of clusters must be specified in advance, which means that it is essential to have prior knowledge of the dataset. The centroid clustering methods organize the data into non-hierarchical clusters. The k-means clustering method is the most widely used centroid-based clustering algorithm.

2. **Connectivity-based clustering**

   The basic idea of connectivity-based clustering is somewhat similar to the centroid-based method, which defines clusters based on the proximity of data points. These methods are

based on the notion that the closest data points are more similar than those farther away. Connectivity-based clustering follows two methods. The first method starts by classifying all data points into separate clusters and aggregating them as the distance decreases. In the second method, all data points are classified into a single group and then separated or partitioned as the distance increases. The distance function can be selected separately. These methods are simple to understand or interpret, but lack the scalability to handle large datasets. Hierarchical clustering methods and their variants are the most successful connectivity-based clustering methods.

3. **Distribution-based clustering**

   In distribution-based clustering, we fit the data to the probability that it belongs to the same distribution. The Gaussian distribution is most used. This method works well on clusters of different sizes; as the distance from the center of the distribution increases, the probability that a point belongs to that cluster or distribution decreases.

4. **Density-based clustering**

   The density-based clustering method connects the data points into clusters according to their density. This method searches the data space for regions of varying densities of data points, and it separates various density regions according to the different densities present in the data space. As long as dense areas are connected, this permits an arbitrary shape distribution. These algorithms do not assign outliers to the cluster. Density-based spatial clustering of applications with noise (DBSCAN) and ordering points to identify the clustering structure (OPTICS) are two prominent density-based clusters.

1.8 Most common clustering methods

1. **K-means**

   k-means clustering is the most popular unsupervised machine learning algorithm. It groups data points into clusters based on their similarities. k-means is a centroid-based clustering algorithm that attempts to minimize the distance between the centroid of each cluster and the corresponding points. The data point average in the same cluster defines their centroid position in data space. In this algorithm, each point in a dataset will be in the cluster which its centroid is the closest to this point. k-means uses an iterative refinement method

to produce its final clustering according to the number of clusters defined by the user (k in k-means is the number of clusters) [43].

The steps in k-means are:

- Randomly choose k means value from the set of points as centroids.
- Assign the data points to their nearest centroid using any distance measurement algorithm.
- Recalculate the centroids using the average of all the points in the cluster, and this will be the value of the new centroid.

The algorithm repeats these processes until the centroids remain the same after the recalculation of the variance of the clusters stops changing Figure 1.5. illustrates an example of clustering using the K-means method. K-means is an efficient machine learning technique

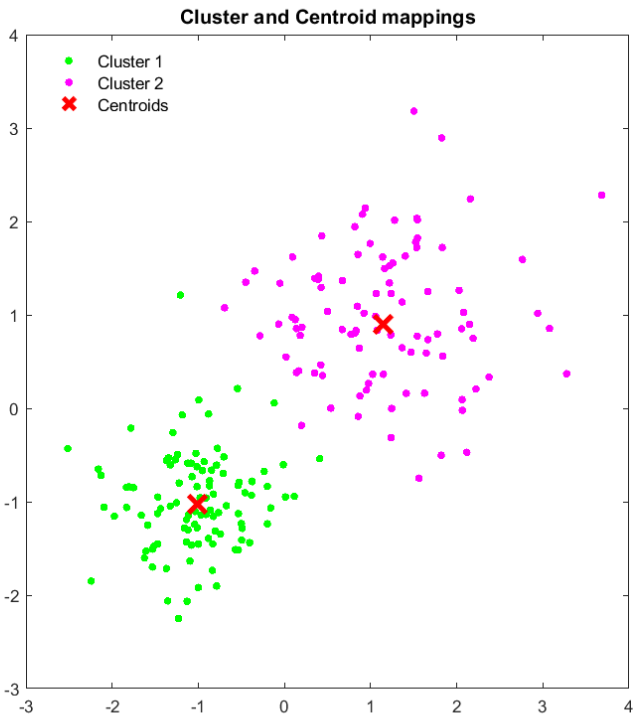


FIGURE 1.5: Clustering using the K-means method.

that is simple and widely used because it is computationally fast and easy to understand. It is used for big data as well as for any type of data. However, k-means cannot find clusters with complex forms; the number of clusters must be fixed at the beginning; the result of k-means depends on the initial selection of the cluster centers; the most common problem with clustering is that the clusters are of unequal size or density.

2. **Hierarchical clustering**

Hierarchical clustering is the creation of cluster trees by iteratively grouping or splitting data points. The idea behind hierarchical clustering is to create a binary data tree that continuously merges groups of similar points and visualizes the tree to provide helpful data summaries [69] (Figure 1.6).

The steps in hierarchical clustering are as follows:

(a) First, assign all points to a single cluster.

(b) Then check the smallest distance in the proximity matrix, merge the points with the smallest distance, and update the proximity matrix.

(c) Repeat step 2 until only one cluster remains hierarchical.

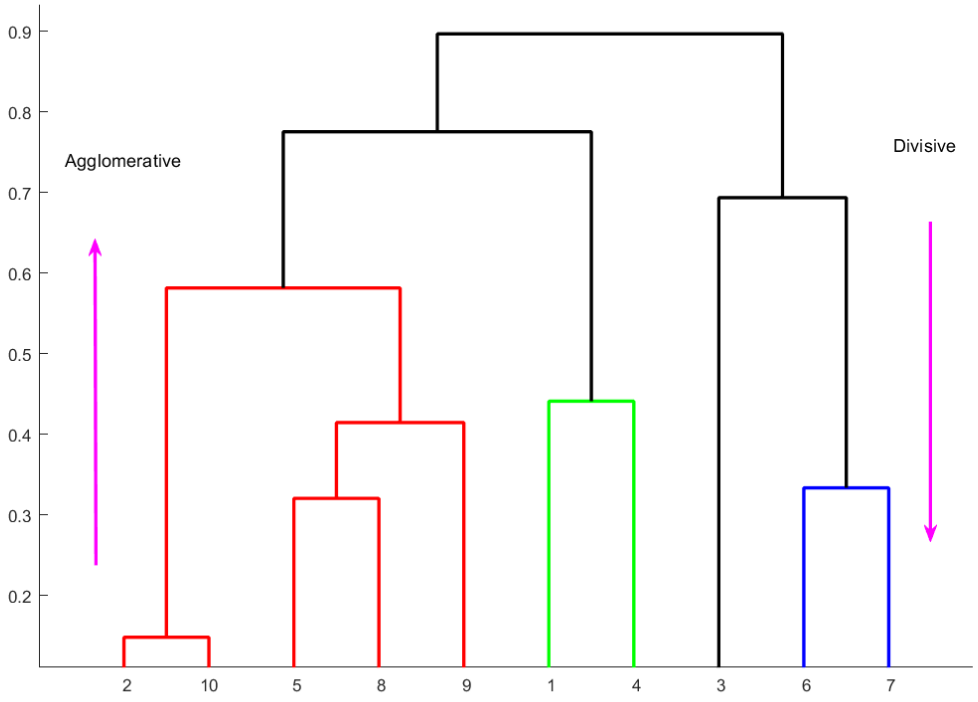


FIGURE 1.6: The principal structure of Hierarchical clustering.

Hierarchical clustering is easy to implement and generates dendrograms, which help to understand the data. However, the hierarchical clustering method does not cancel any previous steps: if the method groups two points and finds that the connection is weak, it cannot cancel this step. Moreover, hierarchical clustering usually has a long-running time compared to other algorithms, such k-Means. Sometimes it is difficult to identify the number of clusters through the dendrogram with a large dataset.

1.9 Classifier evaluation

1.9.1 Splitting the training/test data

After training, a numerical estimate of the difference between predicted and original responses, commonly known as the training error, is produced during this process. However, this indicates how well the model performs on the data used to train it, but this doesn't indicate if the model has under-fitting or overfitting the data. The problem with this kind of evaluation is that it does not indicate how well the learner would generalize to an independent/unseen data set. One way to overcome this problem is not to use all the data when training a classifier. Before the training starts, some data is deleted. Once training is complete, the deleted data can be used to evaluate the performance of the learned model on new data. There are different ways to split the training/test data, the most common are:

1. **The Holdout method:** is the simplest way to split data. In this method, the dataset is partitioned into two groups. The maximum group is used for training, and the remaining data belongs to the test set.

2. **K-fold cross-validation:** in this method, the data is first partitioned into k subsets randomly, and the exclusion method is repeated k times. One of the k subsets is used as a test set each time, while the remaining k-1 subsets are combined to form a training set. Then the mean error over all k trials is calculated. The data is usually stratified before being divided into k folds. Stratification is the process of reorganizing the data to ensure that each fold is a good representative of the whole. For example, in a binary classification problem where each class comprises the data, it is preferable to organize it. In each fold, every type consists of about half the instances [64].

1.9.2 Evaluation metrics

After training, it is essential to evaluate the classifier to verify its applicability and check its accuracy and efficiency. There are many ways to evaluate a classifier.

- **Accuracy** is the most common performance measure. It defines the ratio of the number of correct predictions made by the classifier over all types of predictions made. One of

the most significant drawbacks of accuracy is that it does not work well when we have an unbalanced dataset. It only works well if there are an equal number of samples belonging to each class.

TP: true positive, FN: false negative, TN: true negative, and FP: false positive.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \times 100\% \tag{1.4}$$

- **Precision** defines the number of true positives divided by the number of true positives plus the number of false positives. This metric tells us what proportion of predicted predictions are positive compared to the total number of positive predictions.

$$Precision \ = \frac{TP}{TP + FP} \times 100\% \tag{1.5}$$

- **Recall or sensitivity** is defined as the number of true positives divided by the number of true positives plus the number of false negatives. It tells us what proportion of predictions predicts positives that are true positives compared to the total positives in the original dataset.

$$Sensitivity = \frac{TP}{TP + FN} \times 100\% \tag{1.6}$$

- **The F1 score** combines precision and recall relative to a specific positive class. It conveys the balance between precision and recall, and there is an uneven class distribution. The F1 score reaches its best value at one and worst at 0.

$$F1 - score = \frac{2 * TP}{2 * TP + FP + FN} \times 100\% \tag{1.7}$$

- **The confusion matrix** is a cross table that records the number of occurrences between the actual classification, and the predicted classification. The columns represent the model predictions, while the rows represent the actual classification.

1.10 Conclusion

In this chapter, we have presented an overview of pattern recognition, including its basic concepts, its application, its process, and its approaches. We have focused on the common classifiers

and have discussed how to evaluate them. These concepts help describe and evaluate the proposed systems.

Chapter 2

Artificial neural networks

Contents

2.1 Introduction

The idea of developing a mathematical model of the human brain to reproduce its intellectual abilities is at the origin of the creation of artificial neural networks (ANN). ANNs are connectionist, parallel-distributed, or adaptive systems because they are composed of a network of interconnected processing elements that operate in parallel.

The first basic neural network model is a single neuron, which McCulloch and Pitts proposed in 1943 based on neural activity [44]. The model describes a neuron as a linear threshold computing unit with several inputs and a single output. A few years later, Hebb established the missing link between a single neuron and a network in his classic book "The Organisation of Behaviour," he hypothesized that "the existing pathway strengthens the connection between neurons," providing the first artificial neural networks for learning mathematical rules [31]. Therefore, the time was ripe enough to witness the birth of networks developed by Rosenblatt [66] in 1958 based on the McCulloch and Pitts model. The networks based on a perceptron unit produce an

output scaled by 1 or -1, depending on a weighted linear combination of inputs. In the 1960s, Rosenblatt, Widrow, and Hoff [67, 82] further explored changes in artificial neural networks based on perceptrons. However, the initial enthusiasm of contemporary scientists for this subject quickly faded in the 1970s, mainly due to two events: first, the practical difficulties of solving many real-world problems; and second, the theoretical research results published by Minsky and Papert [46] in 1969 showed that the perceptron was severely limited in the form used and could not simply be added by adding several layers of neurons. It also demonstrated that the perceptron could not represent simple linear and inseparable functions, such as the famous "exclusive OR" problem [46]. Therefore, when a paper by Hopfield [33]in 1982 injected new vitality into the research field, the era of artificial neural networks seemed to be over. Hopfield introduced two key concepts that overcome all the limitations identified by Minsky and Papert: the non-linearity between the total input received by a neuron and the output it generates and the possibility of feedback coupling between output and input. Instead, in 1986, the multilayer perceptron was introduced by Rumelhart to give a renaissance to neural networks. This discovery was a real revolution that allowed neural networks to grow considerably. Currently, the literature on neural networks has become huge and keeps growing.

Despite the recent success of artificial neural networks, the scientific community is still far from implementing machines capable of reproducing the computational capacities of even the simplest nervous systems. It is, in fact, simple modeling of the behavior of biological neurons that will never be able to represent their complex functioning. On the other hand, this modeling allows the implementation of high computational capacities based on simple basic cells: formal neurons.

2.2 Principles of neural networks modeling

2.2.1 Definition of neural networks

Neural networks are a mathematical machine learning model that utilizes learning algorithms inspired by how the human brain learns and handles information. It can process non-linear and complex features of data to generalize and predict future samples. Their design mimics the behavior of biological neural networks; in humans, learning involves making minor adjustments to the synaptic connections between neurons, and in ANNs, the learning process is based on the interconnections between artificial neurons that exist in the different layers of the nervous system.

2.2.2 Biological neuron

The neuron is the basic unit of the nervous system organization. Neurons are cells specialized in the processing of electrical signals and the transmission of nervous messages. The human brain contains several billion interconnected neurons, each connected to about ten thousand other neurons. The structure of neurons perfectly adapts to their tasks. A neuron comprises four parts: the dendrites, the cell body, the axon, and the synapses. Figure 2.1 Illustrates the structure of a biological neuron.

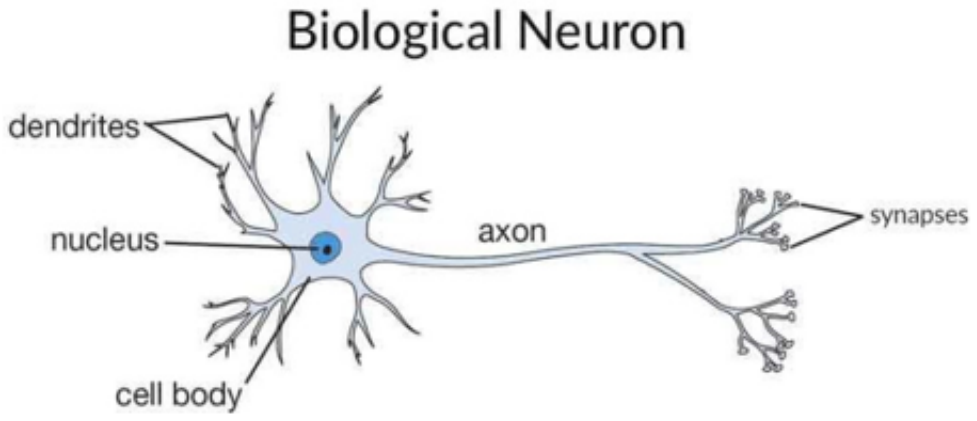


FIGURE 2.1: The biological neuron.

- **Cell body:** It contains the nucleus of neurons; its size is a few microns in diameter. It is the center of the nerve impulse, which represents the state of activity of the neuron.
- **Dendrites:** They are the primary inputs of the neurons; they capture the signals sent to them. Their length is a few tens of microns.
- **Axon:** This is the output of a neuron that carries the nerve impulse in the form of an action potential to the following neurons. It ends in a synapse that chemically transmits the electrical message to the dendrites of the next neuron. The axon is longer than the dendrites; it branches at its end or connects to other neurons. Its size can vary from a few millimeters to more than one meter.
- **Synapses:** These are junctions between two neurons and are essential for the functioning of the nervous system [51].

2.2.3 Functioning of the biological neuron

Information is transmitted in one direction only: from the dendrites to the axons. Each neuron receives input from other neurons in the form of an electrical signal through its dendrites. Suppose the sum of these signals is excitatory. In that case, the neuron will, in turn, emit an electrical signal that propagates through the axons to the terminal connectors and then to the dendrites of the other neurons. According to Hebb [31], neurons learn by changing the electrical resistance of their dendrite connections.

2.2.4 Artificial neuron

The first modeling of the neuron goes back to the '40s when McCulloch and Pitts [44] proposed the first model of formal neurons. A mathematical model performs a weighted sum of the signals it receives and triggers a response if this sum exceeds a certain threshold. Figure 2.2 summarizes the processing series developed by this neuron. This model did not have a learning rule until 1949 when Hebb [31] proposed a learning principle without giving equations.

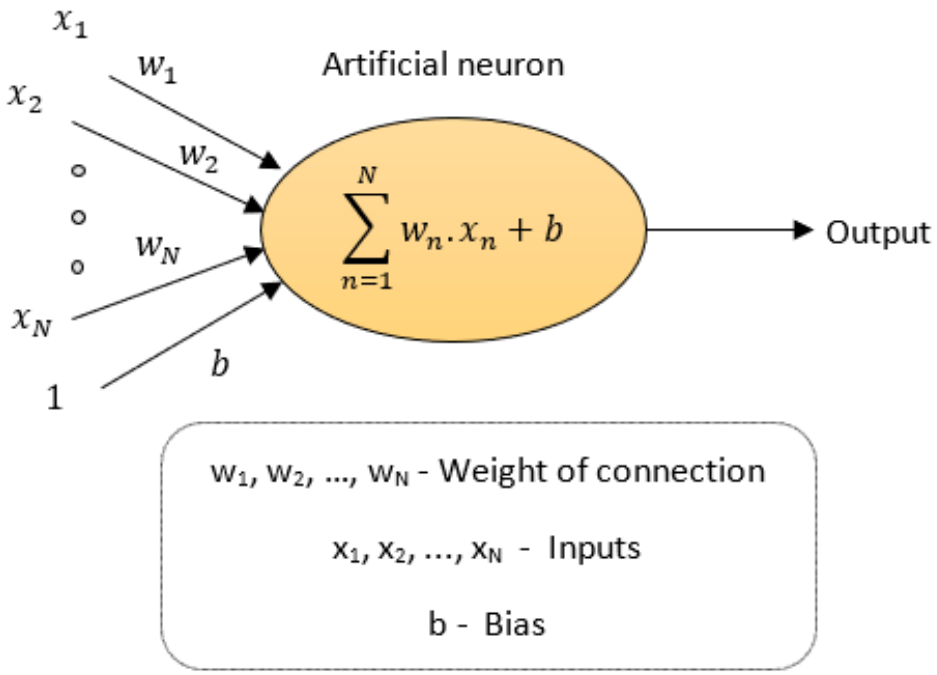


FIGURE 2.2: The formal neuron.

The value of the output (Z) results from the sum of the inputs (x_n) weighted by coefficients w_n and the calculation of an activation function (F) of this weighted sum. The mathematical formalization of its behavior is given by:

$$Z = F(S) = F\left(\sum_{n=1}^{N} w_n.x_n + b\right) \tag{2.1}$$

It should be noted that this simplified modelization is far from providing an exact explanation of the complexity of the functioning of biological neurons. Nevertheless, this formalization allows us to study the connections between these neurons in other more complex processes involving several interconnected neurons.

The Hebbian rule is inspired by biology (the study of learning in artificial neurons). The principle is to reinforce the connections between two neurons when they are active simultaneously. It can be classified as unsupervised learning or supervised because weights corresponding to the learning of several samples can be calculated directly [61]. It applies to the connections of two neurons and is based on their potential. Hebb announced the first learning rule in 1949: "When a cell A excites by its axon a cell B and repeatedly and persistently participates in the generation of an impulse in B, a growth process or metabolic change takes place in one or both cells so that the efficiency of A in triggering an impulse in B is among the other cells that have this effect increased" [61]. This means that if two neurons are active simultaneously, the weight (synapse) becomes stronger; otherwise, it becomes weaker.

The weight change depends on the co-activation of the pre-synaptic and post-synaptic neurons, as shown in Table 2.1 A_i and A_j are the activation values of neurons i and j, respectively.

TABLE 2.1: Hebb's rule.

A_i	A_j	ΔW_{ij}
0	0	-
0	1	0
1	0	0
1	1	+

ΔW_{ij} corresponds to the weight change performed. In the case of formal neurons, this rule can be modeled by the following equation:

$$W_{ij}(t + \Delta t) = W_{ij}(t) + \mu \Delta W_{ij}(t) \tag{2.2}$$

With $\mu \Delta W_{ij}(t)$ is the coactivity of the two neurons, $W_{ij}(t + \Delta t)$ is a new weight, $W_{ij}(t)$ is the old weight, and μ is the learning factor.

2.2.5 Activation functions

When the electrical potential of a biological neuron increases and reaches its excitation threshold, it triggers an action potential that propagates along the axon: this is the nerve impulse. If the action potential is activated, the neuronal information will propagate to other neurons. The action potential inspires the notion of an activation function used in machine and deep learning. Due to the inputs' linear or non-linear transformation, the activation function decides to transmit or not transmit the signal according to the output value obtained [25].

The activation function defines the neuron output as a function of the output of the calculator x. In the following, we describe some of the different activation functions [85]:

1. **Linear function:** The output of the neuron is proportional to the input, as shown in Figure 2.3. And, it can be described as

$$f(X) = X \tag{2.3}$$

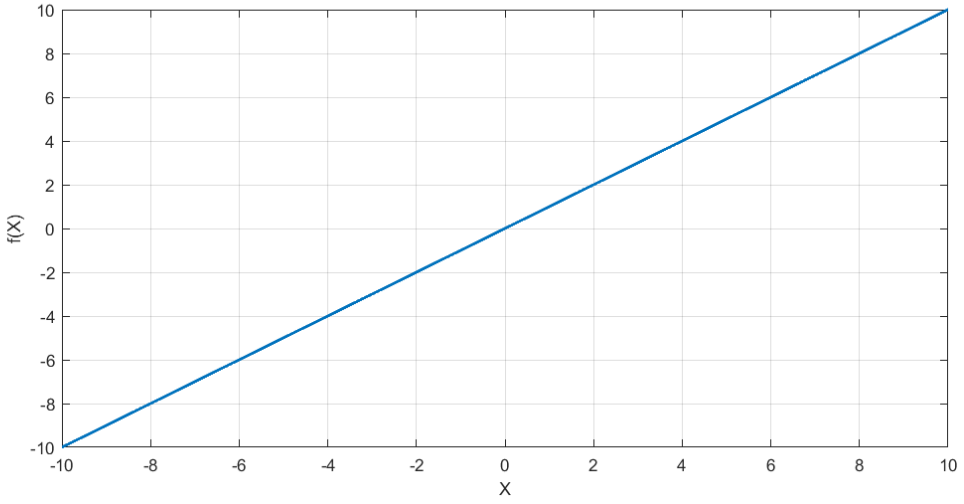


FIGURE 2.3: Linear activation function.

2. **Threshold or binary step function:** This activation function is shown in Figure 2.4, where the output of the neuron is given by:

$$f(x) = \begin{cases} 1 & x >= 0 \\ 0 & x < 0 \end{cases} \tag{2.4}$$

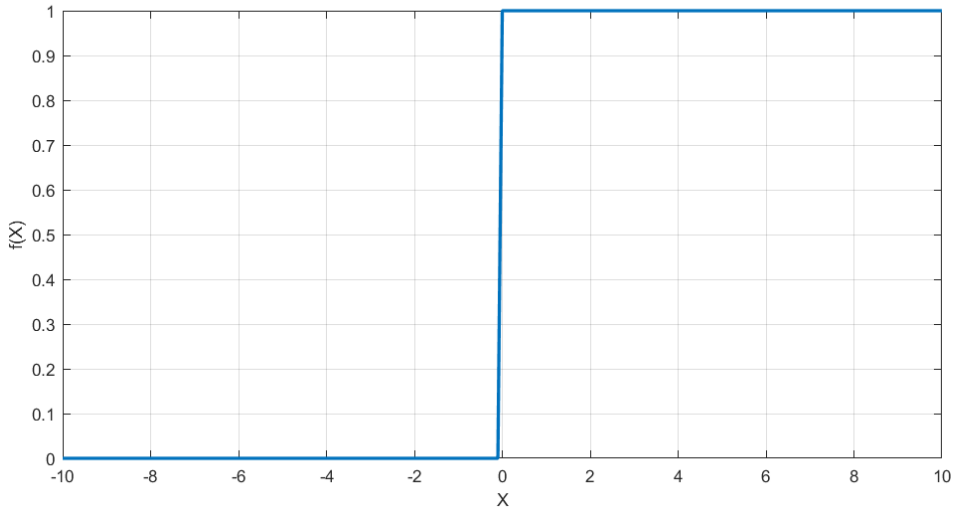


FIGURE 2.4: Threshold activation function.

Neuron with this activation function is called the McCulloch-Pitts model in recognition of the pioneering work done by McCulloch and Pitts (1943); if the local induced field of the neuron is non-negative, the output of the network value is 1; otherwise, it is 0.

3. **Sigmoid function:**The equation for the sigmoid function is:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

Figure 2.5 shows the sigmoid activation function. It can be observed that the function has non-linear properties and can produce an analog output. This is different from the threshold function, which creates output in a discrete range [0, 1].

The sigmoid function causes a vanishing gradient problem, which occurs when significant inputs are converted between 0 and 1, causing their derivatives to become significantly

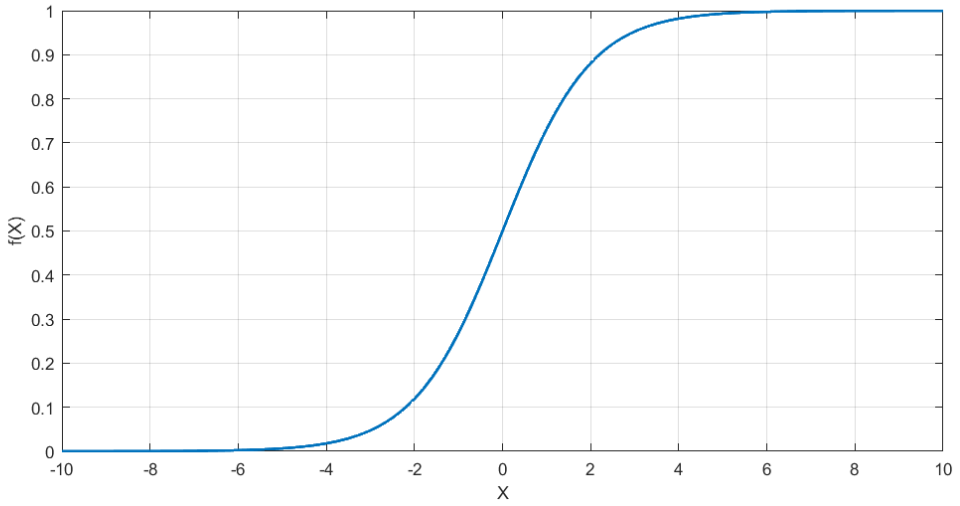


FIGURE 2.5: Sigmoid activation function.

smaller, resulting in an unsatisfactory output. Another activation function such as ReLU is used to solve this problem where we do not have a small derivative problem [70, 85].

4. **Tanh activation function:**It is a Hyperbolic Tangent function. This activation function has the advantages of the sigmoid function, and its characteristic is that the output range is between -1 and 1, Tanh function is continuous and differentiable, as shown in Figure 2.6 The output is described as follows:

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.6}$$

5. **Rectified linear unit (ReLU) function:**The rectified linear unit (ReLU) is currently the most commonly used non-linear activation function, going from 0 to infinity. All negative values are converted to zero [70, 85]. It can be defined mathematically as:

$$f(x) = max(0, x), \; f(x) = \begin{cases} 0 & x < 0 \\ x & x >= 0 \end{cases} \tag{2.7}$$

6. **Softmax activation function:**The Softmax function is a combination of several sigmoid functions. As a sigmoid function returns values in the range 0 to 1, we can treat these as probabilities of data points from a specific class. Unlike sigmoid functions, which are

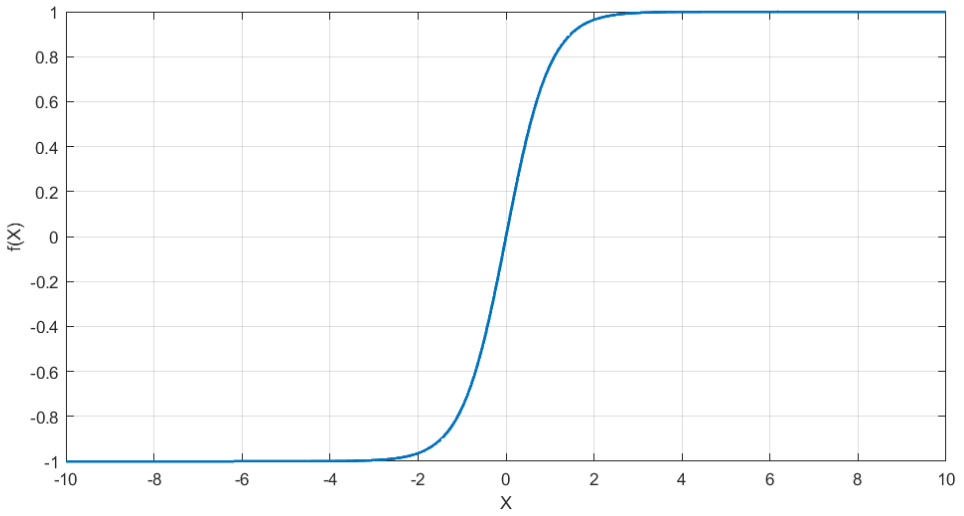


FIGURE 2.6: Tanh activation function.

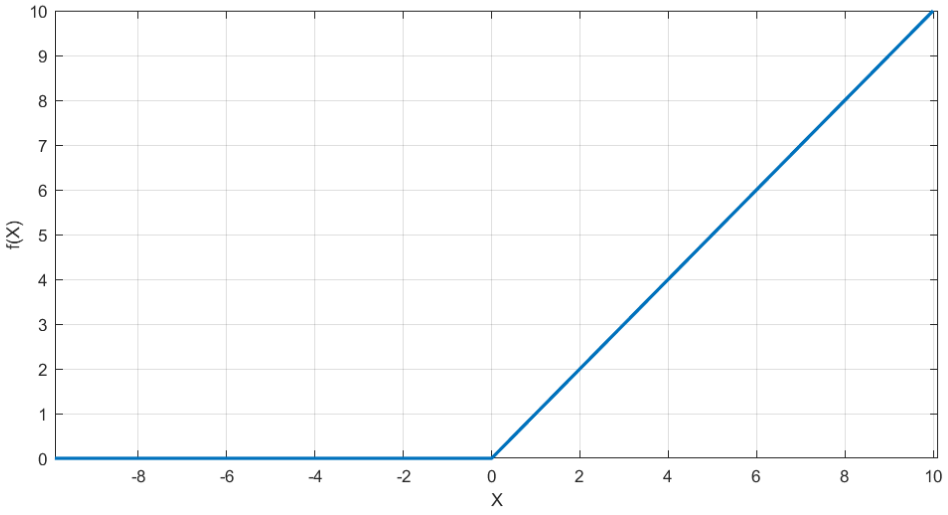


FIGURE 2.7: Rectified linear unit (ReLU) activation function.

used for binary classification, the softmax function can be used for multiclass classification problems. The probability is returned by the function for each data point in each of the individual classes. It can be expressed as follows: When we build a network or model for multiple class classification, the output of the network layer will have the same number of neurons as the number of classes in the target [70, 85].

2.3 Application areas of neural networks

The rise of neural networks in various current domains is undoubtedly due to their great computational capacities and high learning abilities. Moreover, the estimation of their parameters is independent of the complexity of the problem treated, which allows them to adapt to the current issues, which are more and more complex. The applications of neural networks can be summarized into three main areas [57]:

1. **Modeling:** Most industrial or research problems, whether mechanical, physical, chemical, or even economic, require representation using a mathematical model to reproduce the behavior of the process involved. Such tasks require computational tools with high computational and learning capabilities and, above all, tools whose design is not dependent on the complexity and size of the problem. Neural networks seem to be one of the most adequate solutions to this type of problem.

2. **Control:** Controlling an industrial process involves designing a system that allows control to be applied to this process to ensure a desired dynamic behavior. The neural networks allow good performance as part of the control because of their flexibility and self-adaptation.

3. **Classification:** Another large industrial problem category consists in automatically assigning an object to a class among other possible classes.

   This type of problem requires representing the samples to be classified using a set of characteristics. The next step is to design a system able to classify these samples based on their representation, and neural networks are particularly well adapted to this type of problem. Classification is the privileged domain of neural networks.

2.4 Types and architecture of ANN

Although the formal neuron model is a superficial imitation of the biological neuron, good organization of these simple primary cells allows neural networks with significant computational and learning capabilities. Nevertheless, the topological organization of neural networks is generally not the result of any imitation of neuro-biological structures. An artificial neural network is an interconnected set of formal neurons operating in parallel. We can distinguish between neural

networks according to whether they are layered or not. In layered networks, the neurons belonging to the same layer do not interconnect, and they generally have the same properties. In these models, the computation is performed from layer to layer (from input to output). The neurons in the same layer operate simultaneously.

On the other hand, depending on whether or not there is feedback from the output to the input neurons, there are two main types of architecture: feedforward and feedback neural networks. In the first type of network, information propagates from the input to the output without any return. The output neurons never influence the input neurons. This architecture is the most used in classification, function approximation, and process modeling. Otherwise, networks of the second type contain loops that bring back the value of one or more outputs to the input with a delay. They can then be considered as dynamic systems. Among the feedforward networks, we find the perceptron, the multilayer perceptron (MLP), the RBF networks... etc. And among the recurrent networks: are Kohonen networks and Hopfield networks [73].

2.4.1 Single-layer feedforward networks

These artificial neural networks have only a single input layer and one layer of neurons, which is the output layer. Figure 2.8 shows single-layer feedforward networks consisting of N inputs and J outputs. Information always flows in one direction, i.e., from the input layer to the output layer. These networks are commonly used for pattern classification and linear filtering problems. The main types of networks belonging to the feedforward architecture include perceptrons and ADALINE. The learning algorithms they use in the learning process are based on the Hebb and Delta rules.

2.4.1.1 Perceptron

The single-layer perceptron is the simplest kind of feedforward neural networks. The perceptron was introduced in 1958 by Franck Rosenblatt. It can be extended to produce multi-valued output via additional units for each output element, but it still does not have any hidden layers. This type of neural networks has minimal learning potential. Due to the absence of hidden layers, outputs are just a linear combination of inputs, so the single-layer perceptron will only approximate linear relationships [66]. Perceptrons can be trained by a simple learning algorithm called the delta rule. It calculates the errors between calculated output and sample output

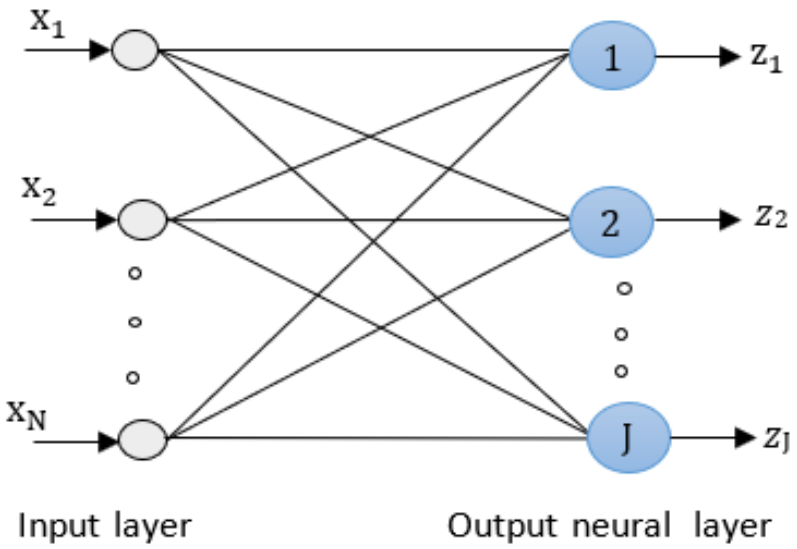


FIGURE 2.8: The architecture of single-layer feedforward networks.

data and uses them to adjust the weights. The perceptron has a single layer of neurons that

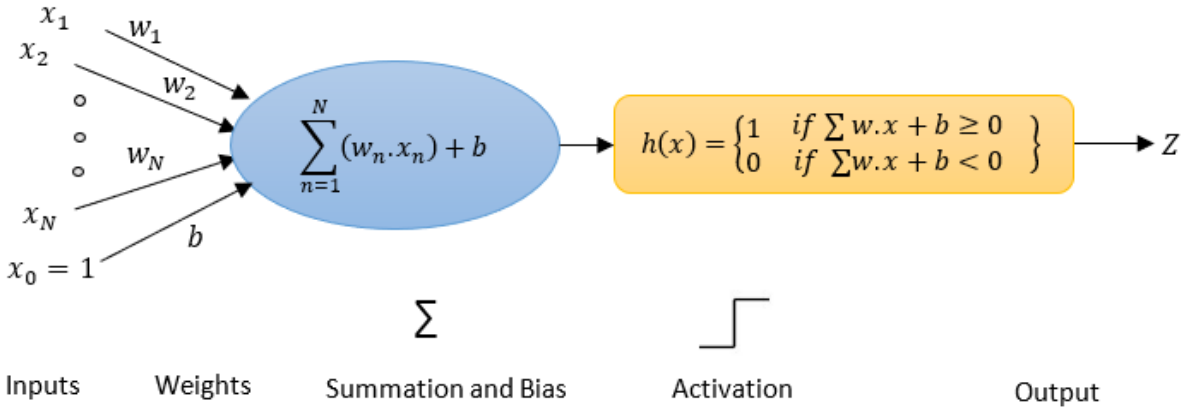


FIGURE 2.9: Representation of a perceptron with only one neuron.

receives input N values x_1, x_2... x_N and computes an output Z. The variables can be boolean or real. The weights can be integers or real. In the original model, the activation function is the Heaviside function, so the output takes its value in the interval [0, 1]. The most used model is the perceptron, illustrated in Figure 2.9, which contains only one neuron. This one only allows you to make a classification in a two-class problem.

However, the recognition of several classes is made possible by putting several perceptrons in parallel. The perceptron thus obtained comprises one neuron per class, each performing a linear discriminant function of the class with which it is associated.

$$Z = h(S)$$

Where,

$$S = \sum_{n=1}^{N} w_{nj}.x_n + b \tag{2.8}$$

The parameters of the perceptron, i.e., the synaptic weights of the neurons, can be determined due to supervised training carried out on a set of pre-classified forms. The perceptron training rule, originally developed by Rosenblatt, is guaranteed to converge if the data is linearly separable. The perceptron partitions the space of input variables into regions, each corresponding to a class.

However, Minsky and Papert note that the perceptron fails for simple classification problems, such as those where the classes are not linearly separable. The case is the XOR classification, where patterns (0; 0) and (1; 1) belong to one class while patterns (1; 0) and (0; 1) belong to another class. These problems suggest using several perceptrons, which are organized into layers of the multilayer perceptron model. This model can handle non-linear problems.

2.4.1.2 ADALINE (ADAptative LINear Element)

Adaline was introduced by Widrow [82]. It is a model of neural networks based on the work of Mcculloch and Pitts [44]. The architecture of the ADALINE is similar to that of the perceptron but with a linear activation function. For a problem with N inputs, the output of the ADALINE is given by:

$$Z = h\left(S\right) = \sum_{n=1}^{N} w_n.x_n \tag{2.9}$$

The learning of the ADALINE is performed by the Widrow rule [82], which consists of minimizing the squared error (E) that is given by:

$$E = \sum_{q=1}^{Q} \left(T^{(q)} - Z^{(q)}\right)^2 \tag{2.10}$$

Where: Q is the total number of training samples; $T^{(q)}$ and $Z^{(q)}$ are, respectively, the desired output and the calculated output corresponding to the example $(X\ (q))$. The Widrow rule is a gradient descent technique that performs the update of each weight at iteration $(r+1)$ as follows:

$$w_n^{(r+1)} = w_n^{(r)} - \eta \frac{\partial E}{\partial w_n} \quad = w_n^{(r)} - \eta\left(t_n - Z_n\right)\ x_n \tag{2.11}$$

Learning the ADALINE consists of adjusting its weights until a reasonably small error is ob-

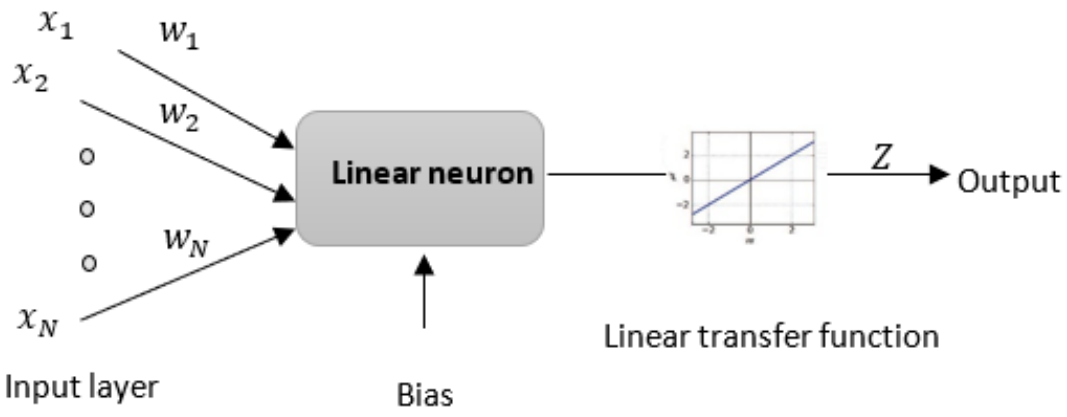


FIGURE 2.10: Representation of an ADALINE model.

tained between the desired and calculated output. That is to say, unlike perception, where learning ends when all samples are correctly classified.

Delta rule (Windrow -Hoff)

In 1960, Widrow and Hoff proposed a supervised learning rule consisting of minimizing a quadratic error function. This error can be calculated according to the responses desired and obtained by the network for each training example. Initially, we choose the weights W_{ij} randomly; then, we modify them until the network can calculate the desired output corresponding to the given input. The mechanism used is as follows:

We provide the network with a set of samples, where each prototype consists of an input vector and the desired output vector. The network calculates an output vector for each input vector, then compares it to the desired output vector. If the resulting error is less than a fixed threshold, the learning stops. Otherwise, this error is used to modify the weights. The following formula does the weight modification.

$$W_{ij} = \eta\big(t_j - z_j\big)X_i = \eta\Delta_j X_i \tag{2.12}$$

With η is the learning rate, t_j is the desired output vector, z_j is the vector of output calculated by the networks, X_i is the input vector, and $\Delta_j = t_j - z_j$ is the error term. The objective of learning is to change the values of the network weights to minimize this error and perform gradient descent.

The gradient descent algorithm consists of changing each weight W_{ij} by an amount ΔW_{ij} proportional to the error gradient [21], given by:

$$\Delta w_{ij} = \eta \frac{\partial E}{\partial w_{ij}} = \eta \sum_i (t_i - z_i) x_i \tag{2.13}$$

This rule was used for learning ADALINE.

2.4.2 Multilayer feedforward networks

Unlike the networks belonging to the previous architecture, the multilayer feedforward networks consist of one or more hidden neural layers. They solve various problems, such as function approximation, pattern classification, system recognition, process control, optimization, robotics, etc.

Figure 2.11 shows that the number of neurons constituting the first hidden layer is usually different from the number of signals comprising the networks input layer. The number of hidden layers and their respective neurons depends on the nature and complexity of the network mapping problem and the quantity and quality of the available data. However, in single-layer feedforward networks, the number of output signals will always be the same as that of neurons in that layer.

2.4.2.1 Multilayer perceptrons (MLP)

Multilayer perceptrons are the classical type of neural network. It consists of three types of layers: the input layer, the output layer, and the hidden layer, as shown in Figure 2.11. The input layer receives the input signal to be processed. The output layer performs the required tasks, such as prediction and classification. Any number of hidden layers placed between the input layer and the output layer constitutes the actual computational engine of MLP. Similar to the feedforward networks in MLP, data flows from the input layer to the output layer [2]. The neurons in MLP are trained using backpropagation learning algorithms. The MLP is designed to approximate any continuous function and can solve separable non-linear problems. The primary uses for MLP are classification, recognition, and prediction. The computation of each neuron in the output layer and the hidden layer is as follows:

$$h(x) \;=\; S(b(1) + w(1)x) \tag{2.14}$$

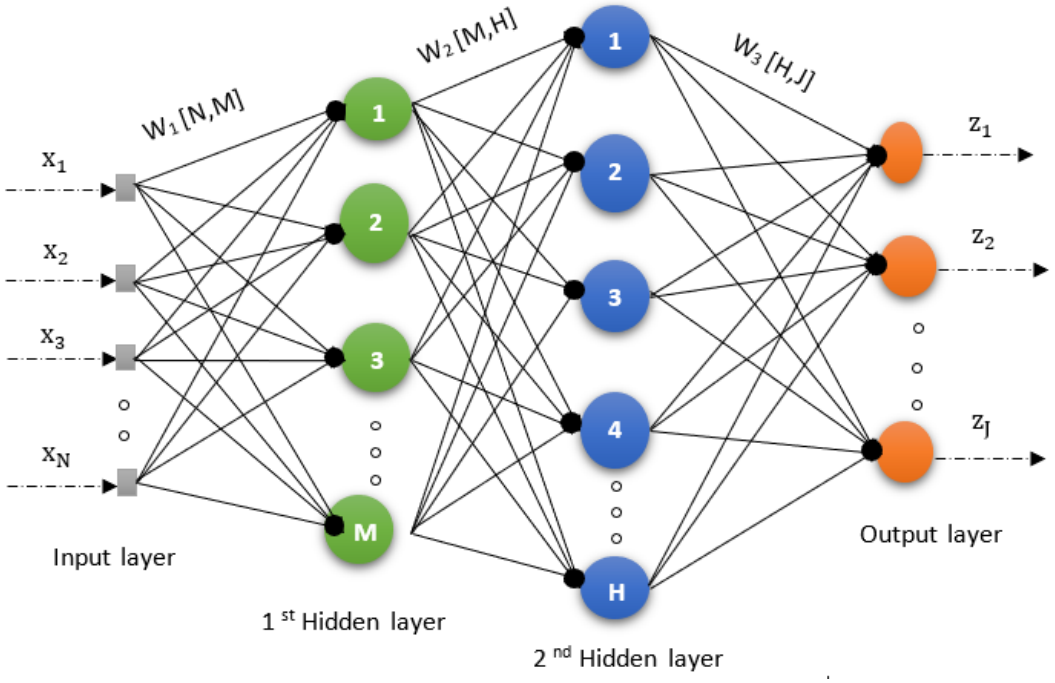


FIGURE 2.11: A multilayer perceptron with two hidden layers.

$$O(x) = G(b(2) + w(2)\, h(x)) \tag{2.15}$$

With bias vectors $b(1)$, $b(2)$; weight matrices $w(1)$, $w(2)$, and activation functions G and S. The set of parameters to learn is the set $\theta = w(1), b(1), w(2), b(2)$. Typical choices for S include the tanh function, or the logistic sigmoid function, with sigmoid.

Training MLP using backpropagation algorithm

The algorithm invented by Rumelhart, Hinton, and Williams in 1986 is used to adjust the weights from the input layer to the hidden layer. This algorithm can also be considered a delta rule generalization for nonlinear activation functions and multilayer networks.

The backpropagation algorithm is based on the modification of the weights to perform gradient descent on the error surface. The weights are first initialized with random values and then changed in a direction that will reduce the error. The modification Δw of a weight w is given by:

$$\Delta w = -\eta \frac{\partial E}{\partial w} \tag{2.16}$$

Where η is a positive constant called the learning step, allowing us to define the size of the modifications of the weights.

The backpropagation algorithm works in two steps. The first is forward propagation, during which the excitation X_q is applied to the input layer and propagates forward through the network to calculate the output z_q and the error $E\ (t_q - z_q)$ concerning the desired output t_q. During the second phase, backward propagation, this error propagates backward to calculate the error for each neuron and make appropriate changes in the network weights [61]. Each weight is modified at iteration $(r+1)$ according to its value at iteration (r) by:

$$w^{(r+1)} = w^{(r)} + \Delta w^{(r)} \tag{2.17}$$

2.4.2.2 Radial basis functions (RBF)

Radial-based function networks are three-layer feedforward neural networks that were first introduced by Broomhead and Lowe [10] in 1988. The first layer corresponds to the inputs of the networks. The second is a hidden layer consisting of several RBF non-linear activation units. The last one corresponds to the final output of the networks. The neurons in the output layer have linear activation functions. RBFNN modeling attempts to mimic the operation of specific biological neurons with a local tuning response, i.e., they respond to only part of the input signal space. A radial basis function is a function that provides a non-zero output only for inputs that lie in a local region of the input space (receptive field). Figure 2.12 shows an example of the RBFN structure. The most common RBF is the Gaussian function given by:

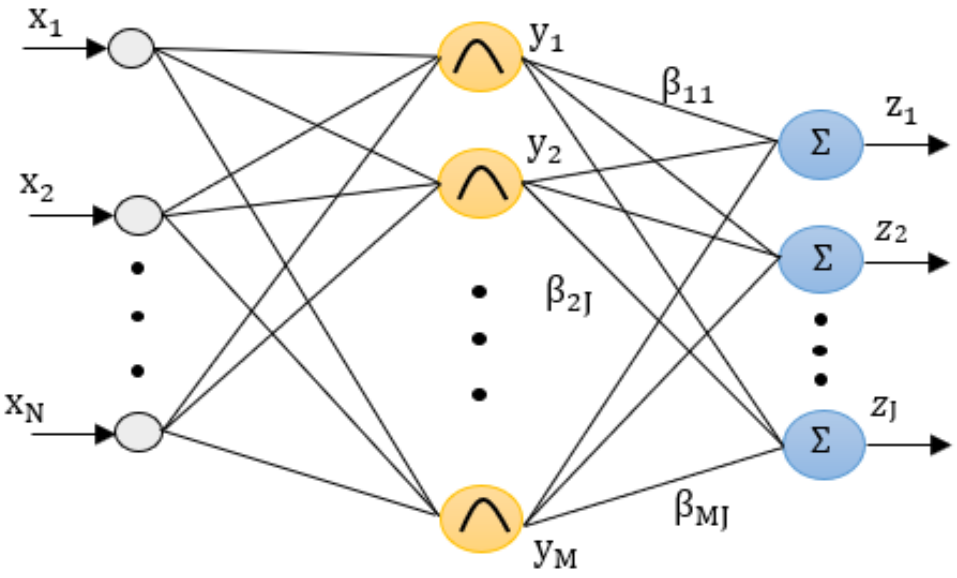


FIGURE 2.12: Structure of radial-based function networks with N-M-J architecture.

$$y_m(X) = \exp\left(-\frac{\| X - V_m \|^2}{2\sigma_m^2}\right) \tag{2.18}$$

Where $X = (x_1, x_2, ..., x_N)$ is the input vector,$V_m = (v_{1m}, v_{2m}, ..., v_{Nm})$ andσ_m are, respectively, the center vector and width corresponding to the m^{th} RBF. An RBF neuron calculates its output based on the distance between each input X and its prototype vector V. The networks outputs are linear weighted combinations of the hidden outputs. The j^{th} output, z_j, is given by:

$$z_j = \sum_{m=1}^{M} \beta_{mj} y_m(x) \tag{2.19}$$

Where M is the number of hidden neurons, y_m is the output of the m^{th} hidden neuron, and β_{mj} is the weight connecting the m^{th} hidden neuron and the j^{th} output.

Most of the classical methods used to train RBFN are performed in two steps. In the first step, the center and width are determined using, for example, an unsupervised clustering algorithm, while in the second step, the connection weight between the hidden layer and the output layer is determined by error criteria such as the standard mean square error.

2.4.2.3 Random neuronal network (RVFLN)

Randomized neural networks (RNNs) are extremely powerful machine learning techniques for processing huge amounts of data due to their high computing and fast learning capabilities [87]. One of the most important RNN types is the Random Vector Functional Link Functional Network (RVFLN) [55]. The input weights and biases in RVFLN can be chosen randomly, and the output weights can be calculated analytically. The direct link between the input and output layers is an efficient and simple regulation technique that prevents RVFL networks from being over-adjusted.

Figure 2.13 illustrates the structure of the RVFL neural network. The dashed lines are direct links from the input to the output layer. The weights of the blue and brown lines are generated randomly, whereas the output weights (associated with the red and black lines) are calculated analytically.

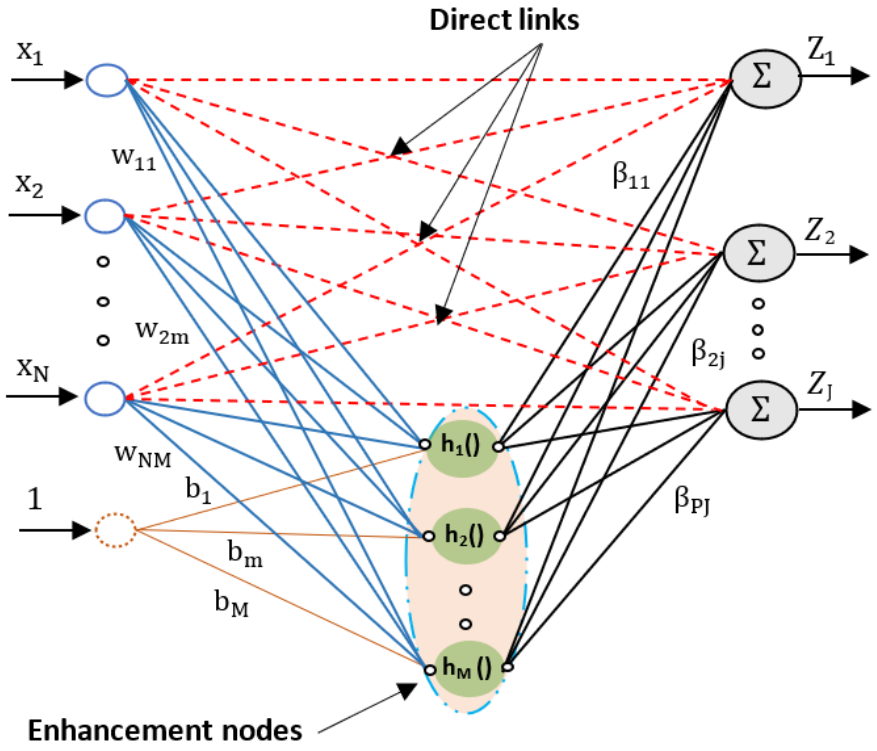


FIGURE 2.13: illustrates the RVFLN structure, including N input neurons, M enhancement neurons, and J output neurons. The input features are first transformed into enhancement nodes where their parameters are randomly generated. All the original and improved features are linked to the output neurons.

An RVFL network with M enhancement nodes can be formulated as the following quadratic optimization problem.

$$\arg\min_{\mathbb{U}} \|\mathbb{S}\mathbb{U} - \mathbb{T}\|^2 \tag{2.20}$$

Where, $\mathbb{U}$ is the output weight matrix, $\mathbb{S}$ is the concatenated matrix that contains the input data (X) and the outputs of the enhancement nodes H, and $\mathbb{T}$ is the labeling matrix.

2.4.3 Recurrent/Feedback networks

In these networks, the outputs are used as feedback inputs to some neurons [55]. The feedback function makes these networks eligible for dynamic information processing, which means they can be used in time-varying systems, such as time series prediction, system identification, optimization, process control, etc.

The main feedback networks include Hopfield, Kohonen self-organizing map (KSOM), and perceptions, which have feedback between neurons of different layers. Their learning algorithms use a learning process based on energy function minimization and the generalized delta rule.

Figure 2.14 shows an example of perceptron networks with feedback, in which an output signal feeds back to the intermediate layer. Therefore, using the feedback process, the current output generated by the networks is also considered the previous output value.

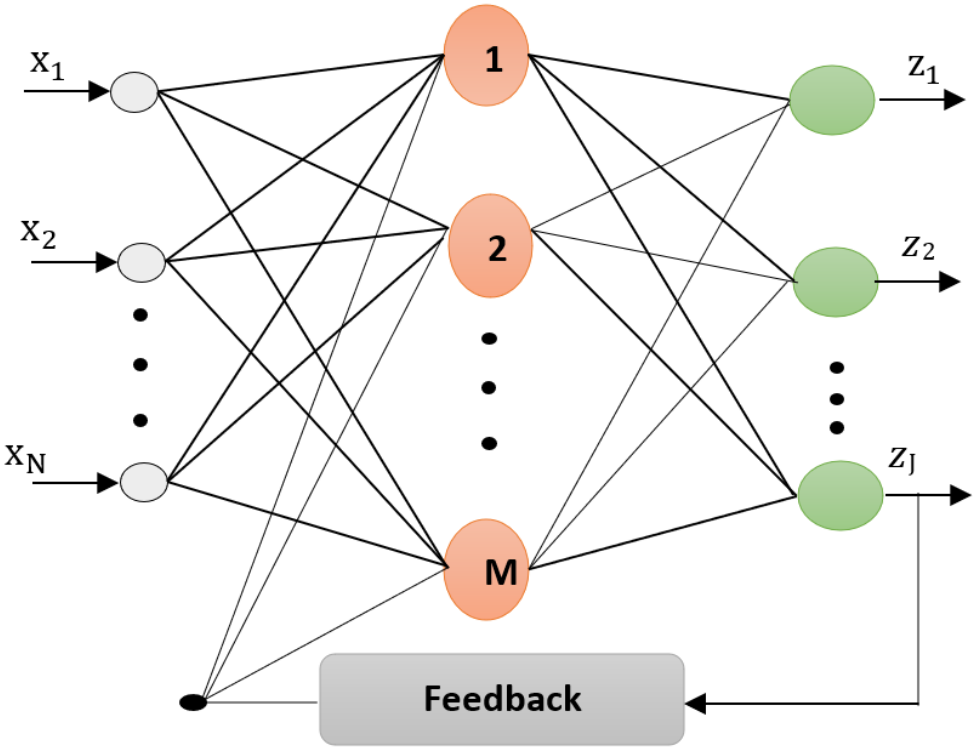


FIGURE 2.14: Example of perceptron networks with feedback.

2.4.3.1 Kohonen self-organizing map (KSOM)

KSOM is an unsupervised classification method invented by the Finnish professor Teuvo Kohonen [36]. The basic setup is that each neuron is only connected to its neighbors. The Kohonen net reduces the complexity of the data obtained in the experiment because repeated training can produce a low-dimensional dataset that captures the essence of the dataset in a simpler form. An essential application of self-organizing maps lies in implementing several projects seeking a simpler way to understand the internet. Kohonen networks are often used as preprocessors for other types of ANNs.

This method creates a network (map) of connected neurons based on the topology. Each neuron in the map is connected to the input vector by weight. The weights are first randomly initialized and then adjusted during the training process. Its activation is calculated with all the neurons on the map to determine the winning node for each input. Activation is measured by the distance between the mapped neuron and the input vector. The low-activation node is called the "winning node". The input is selected at random; once the winning node is determined, the weights of the winning node and adjacent nodes are updated. This process is repeated until the stopping criterion is met [36]. Figure 2.15 illustrates the architecture of Kohonen's self-organizing map.

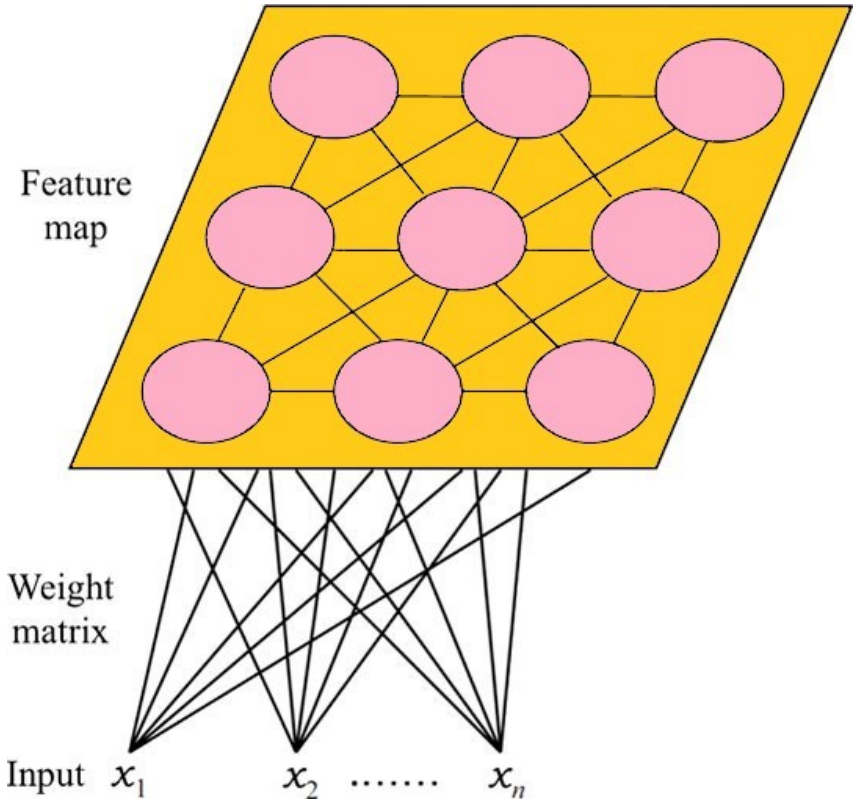


FIGURE 2.15: The Kohonen self-organizing map model.

2.4.3.2 Hopfield networks

The Hopfield networks are a particular type of neural network introduced by John Hopfield in 1982 [34]. Hopfield networks appeared when neural networks had lost considerable interest, and this discovery contributed to their revival. It consists of a single layer that contains one or more fully connected recurrent neurons. Hopfield networks are commonly used for auto-association and optimization tasks. Figure 2.16 shows an example of Hopfield networks where each neuron is connected to all the others, but not to itself. Hopfield networks operate in a discrete line

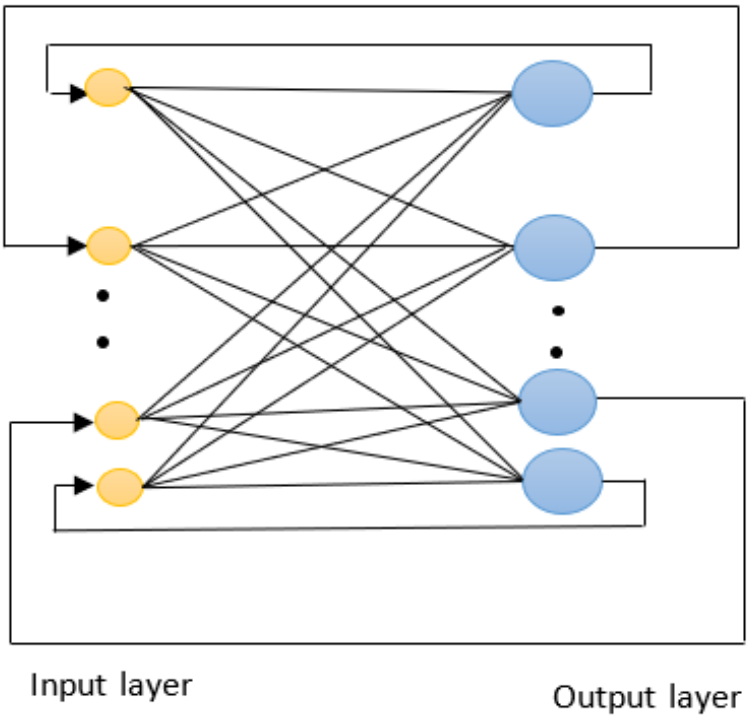


FIGURE 2.16: A Hopfield network.

style, which means the input and output patterns are discrete vectors that can be binary $(0, 1)$ or bipolar $(+1, -1)$ in nature. The networks weights are symmetrical (for each pair of neurons i and j : $w_{ij} = w_{ji}$ *and* $w_{ii} = 0$), and there are no self-connections. The states of all the neurons characterize the state of the networks.

The calculation of the new state of a neuron is given by:

$$O_i = \begin{cases} +1 \sum(w_{ij}x_i + l_i - \tau_i) > 0 \\ -1 \sum(w_{ij}x_i + l_i - \tau_i) <= 0 \end{cases} \tag{2.21}$$

Where, w_{ij} connection strength between i^{th} and j^{th} neurons, O_i is the state of the neuron i; x_i is the input to that neuron; τ_i is its threshold, and l_i is a random variable.

When an example is presented to the networks, all neurons calculate the weighted sum of their inputs using equation 2.21 until the networks reach a stable state. The learning of these models is exceptional compared to other neural networks, and it does not consist of seeking convergence by trial and error. The knowledge of all the states of a neuron allows direct learning of the

weights. This is Hebbian learning given by:

$$w_{ij} = \frac{1}{N} \sum_{q=1}^{Q} x_i^{(q)} \; x_j^{(q)} \tag{2.22}$$

Where, N is the dimension of the input vector; Q is the number of training samples, and, x_i is the q^{th} the input of neuron i.

2.5 General types of ANN training algorithms

One of the most relevant features of ANNs is their ability to learn from the presentation of samples that express the behavior of the system. Therefore, once the networks have learned the relationship between the input and the output, they can generalize the solution. The networks can produce an output close to the expected output of a given input value. In the learning process, the networks will extract the discriminative features of the mapped system from the samples obtained from the system. Usually, the complete set containing all available samples of system behavior is divided into two subsets; the learning and the test subsets. The learning subset generally consists of 60-90% random samples in the entire set and is mainly used in the learning process. The test subset consists of 10-40% of the entire sample set. It will be used to verify whether the capabilities of the networks of the generalized solution are within acceptable levels, thus allowing the verification of a given topology [73]. We can classify the learning methods as follows:

1. **Supervised learning**

   The supervised learning strategy provides the required outputs for a given set of inputs; in other words, each learning sample consists of the input and its corresponding output. After that, it needs a table containing attribute and value data representing the process and its behavior. From this information, the neural structure will develop hypotheses about the system to be learned. In this case, the application of supervised learning depends only on the availability of the attribute/value table, and it behaves as if the coach is teaching the networks what the correct answer is for each sample presented as its input. The synaptic weights and thresholds of the networks are continuously adjusted by applying comparative actions executed by the learning algorithm itself, which supervises the difference between the produced outputs and the desired outputs, using this difference in the adjustment

procedure. The neural networks, when this difference is within the acceptable range of values, are considered "trained," which is the purpose of the general solution. [73].

Supervised learning is a typical case of pure inductive reasoning, in which the free variables of the networks are adjusted by an a priori understanding of the expected output of the system under study. In 1949, Donald Hebb proposed the first supervised learning strategy, inspired by neuro-physiological observation.

2. **Unsupervised learning**

   In this paradigm, the system must find statistically prominent features in the input data. Unlike the supervised learning paradigm, there is no prior set of categories to classify the pattern; instead, the system must develop its own input stimulus representation. The networks must identify groups with similarities. The learning algorithm adjusts the synaptic weights and thresholds of the networks to reflect these groups within the networks themselves. Networks designers can use their knowledge of the problem (a priori) to specify the maximum number of possible groups [73].

3. **Reinforcement learning**

   The reinforcement learning methods are considered a variant of supervised learning techniques because they continuously analyze the difference between the response produced by the networks and the corresponding desired output (Sutton and Barto 1998). The learning algorithm used for reinforcement learning relies on any qualitative or quantitative information obtained through interaction with the mapped system (environment) to adjust the internal neural parameters and uses this information to evaluate learning performance. The network learning process is typically trial and error, as the only available response to a given input is satisfactory or unsatisfactory. If satisfied, the weight and threshold of the synapse will gradually increase to reinforce that system-related behavioral condition. Many learning algorithms used in reinforcement learning depend on randomized methods, which select adjustment actions probabilistically. It considers a limited set of possible solutions and provides rewards if they have a chance to produce satisfactory results. During the training process, the probabilities associated with action adjustments are modified to improve the performance of the networks (Tsoukalas and Uhrig 1997). This adjustment strategy has some similarities to some dynamic programming techniques (Bertsekas and Tsitsiklis 1996; Watkins 1989) [73].

4. **Offline learning**

   In offline learning, also known as batch learning, the weights and thresholds of the networks are adjusted after all the training sets appear. Each adjustment step considers the total error observed in all training samples. Therefore, offline learning needs at least one training period to adjust its weights and thresholds. All training samples must be available during the entire learning process. This process is time-consuming and requires high computing resources. For this reason, training is usually performed before the method is used to predict the output of new data: First, the system is trained, and then it is used. To train an already trained batch learning system with new data, it may be necessary to introduce a new version of the system from scratch with a new dataset (extending the old data with new data) and replace it with the newly trained system. This process can take several hours to run. The advantage is that the new system training, evaluation, and start-up can be easily automated. Another limitation of these systems is that if the dataset is large, there may not be enough computational resources to train the entire dataset [65] starting with a new dataset

5. **Online learning**

   In online learning, the weights and thresholds of the networks are adjusted after each learning sample is submitted. Therefore, after performing the adjustment step, the corresponding sample can be rejected. Online learning is typically used when the behavior of the mapped system changes rapidly. The system provides incremental training in online learning by sequentially providing new data individually or in small groups called "mini-batches." Unlike in batch learning, these training steps are fast and inexpensive in terms of computing resources, so that the system can learn new data points at runtime. These systems are ideal when new data is readily available [65].

2.6 Advantages and disadvantages of neural networks

Advantages of a neural network

1. Independent of prior assumptions: neural networks will not make previous assumptions about the distribution of data or the form of interaction between factors.

2. It is easy to maintain: the neural networks can be updated with new data, making them suitable for dynamic environments.

3. The neural networks can be implemented on hardware in parallel.

4. Storing information on the entire network: data is stored on the whole network instead of in a database, as in traditional programming. The disappearance of a few pieces of information in one location does not render the network inoperable.

5. Ability to work and adapt with incomplete, missing, and noisy data: After ANN training, the data may produce output even when the data is incomplete. The loss of performance, in this case, is determined by the significance of the missing data.

6. Having fault tolerance: the corruption of one or more ANN cells does not prevent producing output. This feature makes networks more fault-tolerant.

7. Having a distributed memory: For an ANN to learn, the samples must be determined, and the network must be learned according to the desired output by showing these examples to the network. The network's success is directly proportional to the number of instances selected, and if the event cannot be shown to the network in all its aspects, the network may produce false output.

8. Gradual corruption occurs when a network slows down and degrades over time. The network problem does not manifest itself immediately.

9. Machine learning capability: Artificial neural networks learn events and make decisions by commenting on similar events.

10. Parallel processing capability: Artificial neural networks have the numerical strength to perform multiple tasks simultaneously. When one neural network element fails, it can continue without problems because of its parallelism.

Problems and disadvantages of a neural network

1. There is no standard rule for determining the structure of artificial neural networks (the optimal number of neurons and hidden layers needed to solve a problem). Appropriate network structures are attained through trial and error.

2. The learning time of the network is unknown: Once the error of the network on the sample is reduced to a specific value, the training is complete. This value does not achieve good performance.

3. Once learned, you can think of a neural network as a black box: you put inputs into it, and it emerges with a result. Sometimes this is sufficient, but in some applications, you need to know what is going on in that black box, especially in marketing applications, to understand the customer's thinking patterns. Therefore, sometimes we prefer learning techniques like decision trees to understand the client's actions rather than neural networks that will only output one result. The most significant issue with ANN is unexplained network behavior. When ANN generates a probing solution, it does not explain why or how. This reduces trust in the network.

4. A neural network has a static structure: once the structure is fixed (number of input neurons, output, hidden neurons) and the network has learned, it is impossible to make it learn new data without first restarting its learning, unlike lazy k-nearest neighbor algorithms or Bayes' naive classifier.

2.7 Conclusion

Neural networks have undergone considerable development in both new architectures and new learning algorithms. In this chapter, we have given a simple overview of these essential mathematical tools. This part has allowed us to draw some conclusions.

- Neural networks seem to be well adapted to the current problems, which are becoming more complex and necessitating the use of advanced computational and learning capabilities.
- Artificial neural networks are based on parallel computation performed by a set of units that operate locally. From a topological point of view, neural networks offer a wide range of architectures, allowing their use in various problems.
- However, neural networks are difficult to parameterize. For example, in the case of the multilayer perceptron, it is difficult to define the number of neurons in the hidden layers. Moreover, neural networks are black boxes that do not allow the obtained model to be interpreted.

In this thesis, we focussed on the collaboration of different types of neural networks such as RBFNN, ELMNN, and RVFLN in order to make use of their different properties.

Chapter 3

Deep learning

Contents

3.1 Introduction

Deep learning involves algorithms inspired by the structure and function of the human brain and is particularly effective for object classification, pattern recognition, and predictive analysis. It can learn several levels of representation to model complex connections between the data. Deep learning is designed to handle huge amounts of data by stacking layers. A deep learning model can extract features from raw data through multiple processing layers and linear and nonlinear transformations, and learn these features bit by bit through each layer with minimal human intervention [7, 21]. In the previous few years, deep learning has grown from a particular field where only some researchers were interested in it to the most popular field among researchers.

The significant stages of DL models were carried out in 1989 by Yann LeCun [38], presenting the first practical demonstration of backpropagation. He combined convolutional neural networks with recent backpropagation theories to read handwritten digits. In the late 1990s and early 2000s, the National Cash Register Company (NCR) and other companies used this technique to read handwritten checks and zip codes to process cash checks in the United States. Another critical phase in deep learning evolution occurred in 1999 when computers began to speed up data processing, and graphics processing units (GPUs) were developed. With GPUs, the processing

computational speed increased by a factor of 1000 over ten years. Meanwhile, neural networks began to compete with support vector machines. Neural networks perform better with the same data. They also have been continuing to improve as new training data is added.

In 2009, the Neural Information Processing Systems (NIPS) workshop working on deep learning for speech recognition discovered that neural networks with a sufficiently large dataset do not need pre-learning and that error rates drop dramatically. Hence the appearance of the ImageNet database, created by Fei-Fei Li [22], professor of artificial intelligence at Stanford. It is a free database containing over 14 million tagged images. In 2011, the speed of GPUs increased significantly, which made it possible to train convolutional neural networks without prior learning. With the increase in computing power, it became evident that deep learning had significant advantages in terms of efficiency and speed. For example, AlexNet [37], a convolutional neural network created by Alex Krizhevsky, won several international competitions in 2011 and 2012. It built and improved LeNet5, which was developed by Yann LeCun years ago. It initially contained just eight layers (five convolutional layers followed by three fully connected layers) and boosted speed and dropout using rectified linear units (ReLU). Its success launched a renaissance of convolutional neural networks in the deep learning community.

Yoshua Bengio, Geoffrey Hinton, and Yann LeCun have won the 2018 Turing Award for their significant contributions to deep learning and artificial intelligence. This is a defining moment for the people who worked extremely hard on neural networks when in the 1970s the whole machine learning community had shifted away from them. Nowadays, both big data handling and the evolution of artificial intelligence depend on DL. It still evolving and requires creative ideas.

This chapter covers deep learning principles and analyses the most commonly used deep architectures, such as autoencoders, convolutional neural networks, and recurrent neural networks.

3.2 Objectives of deep learning

The development of deep learning has been partly driven by the failure of traditional machine learning algorithms in some AI tasks. The true potential of deep learning was not understood until significant amounts of data became available. One of the significant differences between conventional ML algorithms and deep learning is that the latter adapts better when more data are provided. Unlike many traditional ML algorithms, which have an upper limit on the amount

of data they can receive, sometimes called a "performance plateau," DL models have no such limitations [54].

Another difference between traditional ML and DL algorithms is the feature extraction step. In traditional ML algorithms, feature extraction is done manually; it is a time-consuming and difficult step and requires a specialist in the field [54]. In DL, this step can be performed automatically by the algorithm. Figure 3.1 illustrates the difference between traditional ML algorithms and DL algorithms. Deep learning techniques are an effective way of dealing with extensive dimensional data and extracting discriminative features. Representation learning can be learned directly from input data without human knowledge. This fundamental aspect of deep learning architectures enables advances toward the main goal of artificial intelligence (AI), which is to make sense of the world around us independently of expert knowledge and interference [54].

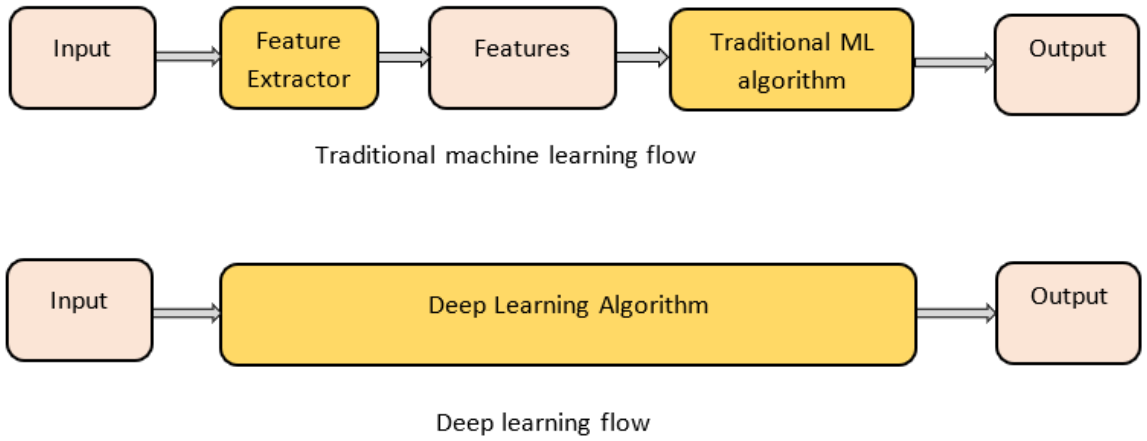


FIGURE 3.1: The process of traditional ML algorithms compared to that of deep learning.

Deep learning attempts to represent high-level abstractions in data by learning from several levels of abstraction. Deep learning models contain several layers stacked to create a block, such as autoencoders, restricted Boltzmann machines (RBMs), and convolutional layers.

3.3 Basics of deep learning

Neural networks with a lot of hidden layers provided poor performance because they were harder to train. Indeed, networks with many hidden layers lead to millions of parameters that must be learned, which makes learning a complicated process with high computational complexity [40]. The three main difficulties in the training process of the deep neural network were [11]:

- Vanishing gradient
- Overfitting
- Training time

Deep Learning is the result of many improvements. This section briefly explains how Deep Learning resolved these problems.

3.3.1 Overcome the vanishing gradient problem

Nonlinear activation functions are favored because they enable nodes to learn more complex structures in the data. Traditionally, two nonlinear activation functions that have been widely used are the sigmoid and hyperbolic tangent activation functions. The sigmoid activation function also called the logistic function, is traditionally prevalent for neural networks. The input of this function is transformed into a value between 0 and 1. Inputs well above 1 are converted to the value 1; similarly, values well below 0 are aligned with 0. The form of the function for all possible inputs is an S-shape from 0 to 0.5 to 1. Through the early 1990s, this was the default activation used on neural networks for a long time. The hyperbolic tangent function, or tanh for short, is a similarly shaped nonlinear activation function that generates values between -1 and 1. In the late 1990s and 2000s, the tanh function was preferred over the sigmoid activation function because the models that used it were easier to train and often had a better predictive performance [11].

A general problem with the sigmoid and tanh functions is that they saturate. This means large values align with 1 and small values with -1 or 0 for tanh and sigmoid, respectively. The limited sensitivity and saturation of the function occur regardless of whether the input node of summed activation contains helpful information. Once saturated, it becomes difficult for the learning algorithm to continue to update the weights to improve model performance. Deep neural networks using sigmoid and tanh activation functions could not be trained easily. Using these nonlinear activation functions, the deep layers of the large networks do not receive useful information about the gradients. Error is backpropagated through the network and used to update the weights. The amount of error decreases significantly with each additional layer through which it is propagated, given the derivative of the chosen activation function. This is called the vanishing gradient problem and prevents deep (multilayered) networks from learning effectively. Although

nonlinear activation functions enable neural networks to learn complex mapping functions, they effectively prevent the learning algorithm from working with deep networks [11].

In order to use gradient descent with error backpropagation to train deep neural networks, a nonlinear activation function is needed that looks and acts like a linear function but allows complex connections in the data to be learned. The solution is to use the rectified linear activation function, or ReL, for short. A node or unit that implements this activation function is called a rectified linear activation unit, or ReLU, for short. Often, networks that use the rectifier function for hidden layers are called "rectified networks" [11].

The rectified linear activation function Figure 3.2 is a simple calculation that directly returns the value provided as input or 0 if the input is 0 or less. This function $g()$ can be described mathematically by using the $max()$ function on the set of 0 and the input z; for example:$g(z) = max(0; z)$ The rectified linear activation function has rapidly become the default function when developing most types of neural networks. Some of the advantages of this function are briefly as follows:

- The rectifier function is trivial to implement. This is different from the tanh and sigmoid activation functions, which require an exponential calculation.
- The rectifier function looks and acts primarily like a linear activation function. In general, a neural network is easier to optimize when its behavior is linear or close to linear.
- The rectifier function can generate a true zero value. This means that negative inputs can produce true zero values allowing the activation of hidden layers in neural networks to contain one or more true zero values. This is different from the tanh and sigmoid activation functions, which learn to approximate a zero output, e.g., a value very close to zero but not a true zero value.
- The derivative of ReLU indicates that the error propagates entirely to the previous layer (with 1) if the input to the ReLU is non-negative or is completely stopped (with 0) if the input to the ReLU is negative. the primary key to this property is that networks trained with this activation function almost completely avoid the problem of vanishing gradients because the gradients remain proportional to the node activations. Adopting the rectified linear activation function meant that it became possible to exploit improvements in hardware and successfully train deep multi-layered networks with a nonlinear activation function using backpropagation.

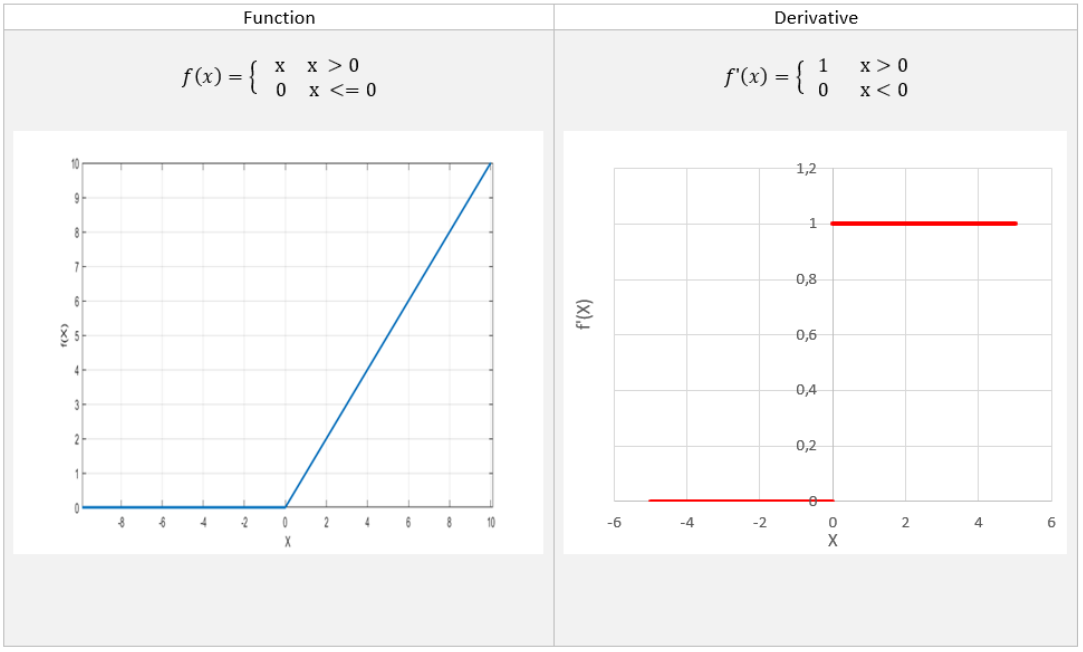


FIGURE 3.2: ReLU activation function (right) and its derivative (left).

Some modified versions of the Relu have been proposed. For example, Leaky Rectified Linear Unit Figure 3.3), or Leaky ReLU, is a kind of activation function based on a ReLU, but it has a small slope for negative values instead of a flat slope. This activation function is useful in tasks where the gradients are sparse, e.g., training generative adversarial networks. The slope coefficient is defined before training, i.e., it is not learned during training.

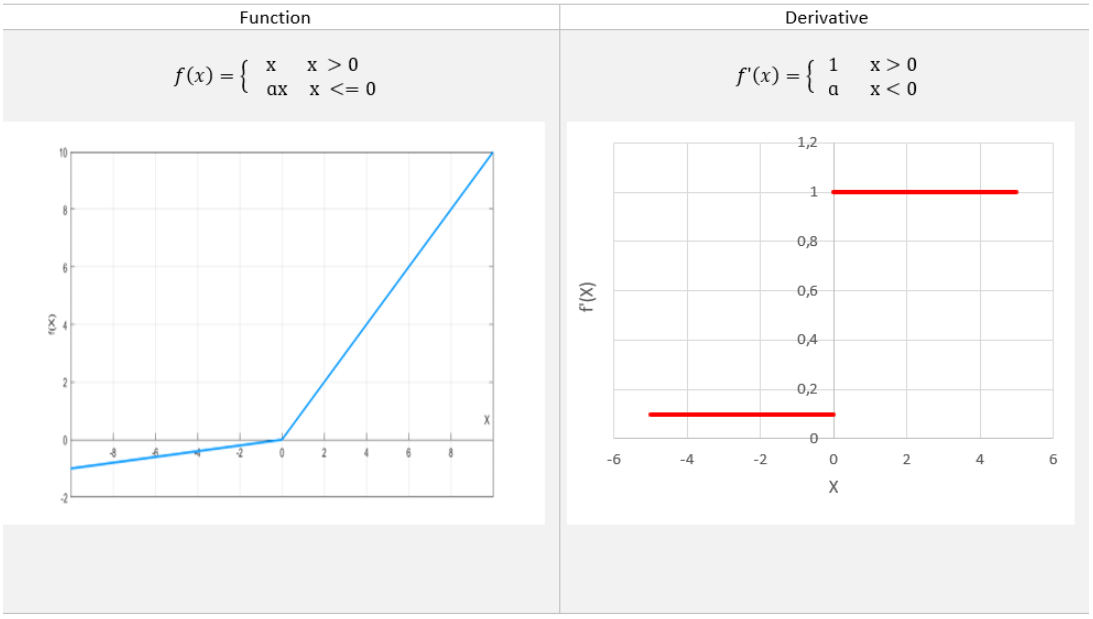


FIGURE 3.3: Leaky ReLU activation function (right) and its derivative (left).

Exponential Linear Unit Figure 3.4 or ELU is a generalization of the ReLU that uses a parameterized exponential function to transition from the positive to small negative values. Unlike other activation functions, ELU has an extra alpha constant which should be a positive number. ELU tends to converge the cost function to zero faster and produce more accurate results.

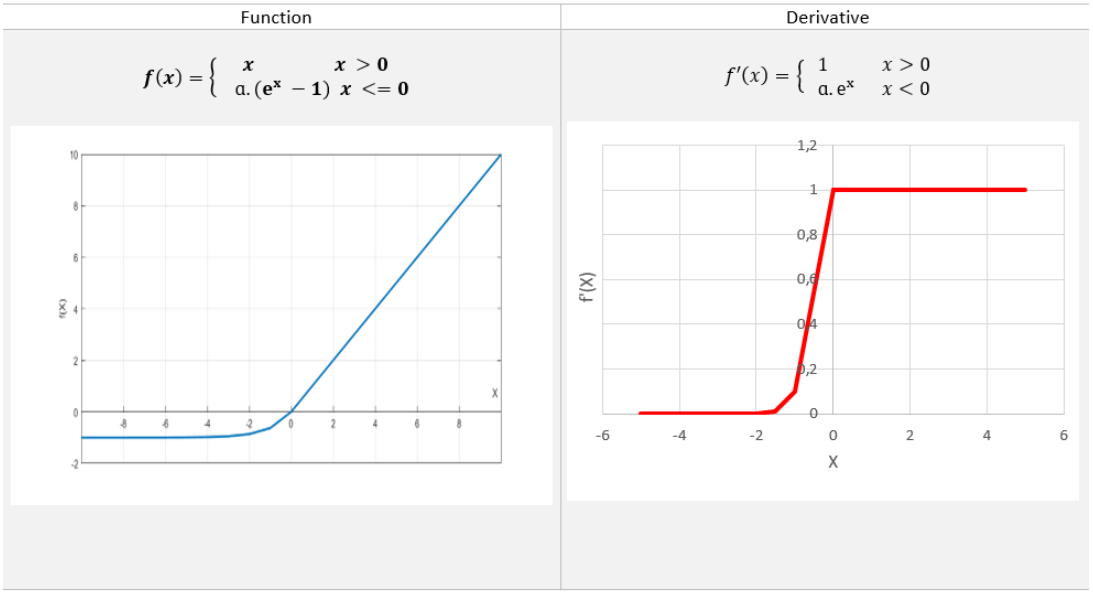


FIGURE 3.4: Exponential Linear Unit (ELU) activation function (right) and its derivative (left).

3.3.2 Overcome the overfitting problem

Training a deep neural network that can generalize well to new data is a challenging problem. Too little learning and the model will perform poorly on the training dataset and on new data. The model will underfit the problem. Too much learning and the model will perform well on the training dataset and poorly on new data, the model will overfit the problem. In both cases, the model has poor generalization performance. Increasing the capability of a model is readily achieved by changing the structure of the model, for example, by adding more layers and more nodes to the layers. Because an underfit model is so easily addressed, it is more common to have an overfit model. An overfit model can be easily diagnosed by monitoring the performance of the model during training by evaluating it on both a training dataset and a holdout validation dataset. There are two ways to correct an overfit model: training the network on more samples or changing the complexity of the network. One of the advantages of deep neural networks is that their performance continues to improve as they are fed with larger and larger data sets [11]. The complexity of a neural network model is defined by its structure in terms of nodes and

layers, and by its parameters in terms of weights. Therefore, the complexity of a neural network can be decreased to reduce the overfitting by one of two ways:

- Change the complexity of the network by modifying the structure of the network (number of weights).
- Change the complexity of the network by modifying the network parameters (weight values).

The structure can be adjusted until reaching the appropriate number of nodes and layers that reduce the overfitting. Alternatively, the model can be overfitted and then pruned by removing nodes until it reaches appropriate performance on a validation dataset. On the other hand, it is more common to concentrate on methods that limit the size of the weights in neural networks with larger structures than necessary. Limiting the parameters size can be performed during the training to ensure that the model does not overfit. The techniques that attempt to reduce overfitting by keeping the weights small are called regularization methods [11].

Regularization methods are so widely used to overcome overfitting that the term "regularization" can be applied to any method which improves the generalization error of a neural network model. The simplest and most common regularization method is to add a penalty to the loss function in proportion to the size of the weights in the model. This will encourage the model to map the inputs to the outputs of the training dataset such that the model weights remain small. This is referred to as weight regularization or weight decay and has proven very effective for decades for simpler linear models and neural networks. Below is a list of five of the most common additional regularization methods [11].

- **Activity regularization:** penalize the model during training based on the size of the activations.
- **Weight constraint:** constrain the size of weights to be within a range or below a limit.
- **Dropout:** probabilistically remove some inputs during training.
- **Noise:** add statistical noise to inputs during training.
- **Early stopping:** monitor model performance on a validation set and terminate training when performance degrades.

3.3.3 Overcome the training time problem

Deep learning principally depends on the used optimization algorithms. The training of a complex deep learning model can take hours, days, or even weeks. Neural networks are generally trained using gradient descent optimization algorithms. This involves utilizing the current state of the model to make a prediction, comparing the prediction to the actual values, and using the difference as an estimate of the error gradient. This error gradient is then used to update the model weights and the process is repeated. The error gradient is a statistical estimate. The more training samples used in the estimate, the more accurate this estimate will be. Good performance comes at a computational cost. Alternately, using fewer samples results in a less accurate estimate of the error gradient that is highly dependent on the specific used training samples. This results in many updates with quite different estimates of the error gradient. Nevertheless, these updates can result in faster learning and sometimes a more robust model [11]. According to the size of the training sample at each epoch (batch size), the gradient descent training algorithms can be divided into three types:

- **Batch Gradient Descent:** the batch size is set to the total number of samples in the training dataset.
- **Stochastic Gradient Descent:** the batch size is set to one.
- **Minibatch Gradient Descent:** the batch size is set to more than one and less than the total number of samples in the training dataset.

The algorithm is often called stochastic gradient descent for shorthand regardless of the batch size. Given the huge datasets that are currently used to train deep learning neural networks, the batch size is rarely set to the size of the training dataset. Smaller batch sizes are used for three main reasons:

- Smaller batch sizes offer regularizing effect and lower generalization error;
- Small batch sizes allow fast training.
- Small batch sizes make it easier to fit one batch of training data in memory, i.e. when using a GPU that has access to less local memory than RAM.

Nevertheless, the batch size impacts how quickly a model learns and the stability of the learning process. It is an essential hyperparameter that the deep learning practitioner should tune well.

3.4 Some common deep neural models

3.4.1 Convolutional neural networks

Convolutional neural networks (CNNs) are a particular type of neural network that uses convolution instead of complete matrix multiplication in the hidden layers. They belong to the category of supervised deep learning models. One of the earliest studies on CNNs is that by LeCun et al. [38]. They used CNNs to recognize handwritten characters. Following that, they successfully applied CNNs to various applications, including object recognition and detection in images, signal processing, time series classification, and semantic segmentation [39], as well as different data types, such as one-dimensional (e.g., signals and tests) and two- or three-dimensional (e.g., images).

Convolutional networks consist of stacked convolutional, pooling and fully connected layers to implement feature learning hierarchically. Figure 3.5 illustrates the general architecture of CNN.

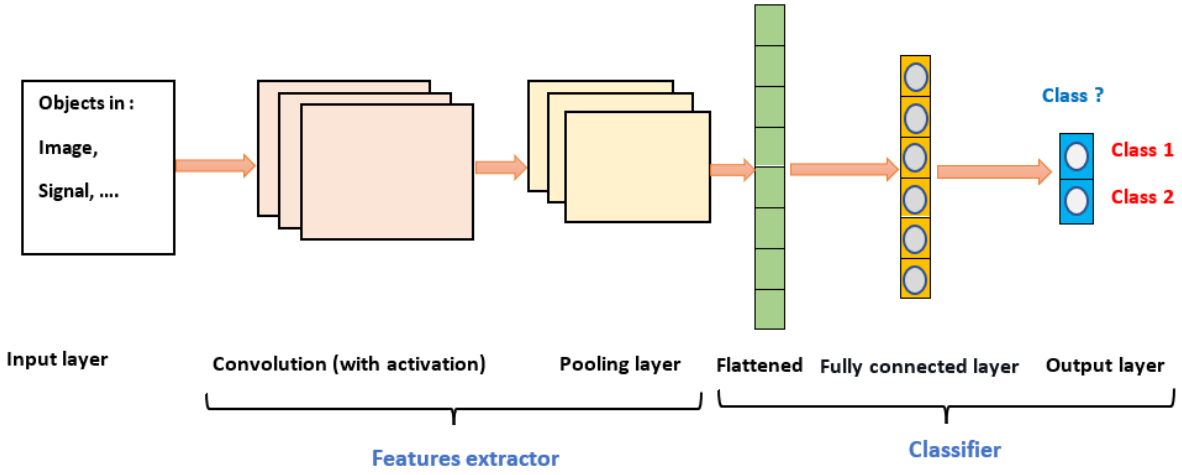


FIGURE 3.5: The general structure of convolutional neural networks.

These models get their names from the linear mathematical operation of convolution, which is always present in at least one network layer. A convolution is described mathematically as an "integral transform" between two functions, one of which must be a kernel. In DL terminology, the kernel is a multidimensional array of weights. The discrete version is the weighted sum of several copies of the original function (f) shifting over the kernel.

The first two layers of CNN are the convolution and subsampling layers. The convolutional layer is the key layer in a CNN. It is a filtering layer and is the most computationally intensive layer in the architecture. It dominates the performance of a given CNN architecture. The convolutional

layer performs a convolution operation to generate feature maps. It uses local receptive fields (small filter sizes) and shared weights (filter maps of equal size) to provide a distortion invariant attribute.

The output of the convolution process is typically passed via a nonlinear activation function before being further modified by a pooling function, which replaces the output at a specific location with a benefit gained from neighboring outputs. This pooling function helps keep the learned representation unchanged for small input translations and sub-samples of the input data.

Subsampling is one of the key steps in CNN architecture. The most common pooling operations are the maximum, mean, and median. For example, the max-pooling function replaces the output with the maximum activation in the rectangular neighborhood.

The sequence of convolution and subsampling layers can be used in parallel in the CNN architecture for a particular task. The output of the last subsampling layer is passed to a fully connected layer for classification or recognition tasks.

Other essential layers are used in CNNs: The ReLu (or rectified linear unit) layer is a commonly used activation function layer. The pooling layer is a subsampling layer used to reduce the computational cost, and the fully connected layer is one of the last layers in CNNs and helps classify data.

CNN needs to receive a large amount of data for real-world applications, especially in high-dimensional input data such as signal pre-processing, image, and speech processing, and to achieve good performance compared to shallow learning methods. Moreover, with so many parameters to train the deep architecture, high-performance computing power is also required. These difficulties can now be easily overcome with highly parallel graphics processing units (GPUs) and existing large data sets.

To see an example of how the input images evolve at each layer in CNN. We trained a two-dimension (2D) CNN to classify the 10 digits from the MNIST database. This network Figure 3.6 includes one convolution layer with 20 filters, each of them having a size 9*9, and one mean pooling, with kernels of size 2*2.

To see how the input image evolves at each layer of the trained network, let's present an example from the test images to the neural network and show the output results. Figure 3.7 shows the input image which is from the digit "4".

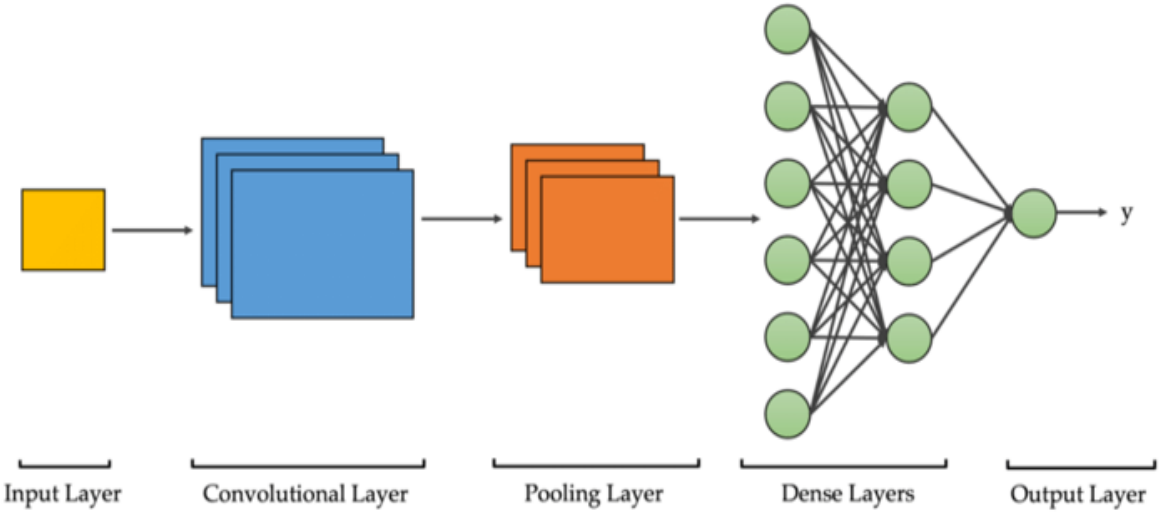


FIGURE 3.6: An example of a Convolutional Neural Network to classify the MNIST dataset. In this example, we used a network with one convolution layer including 20 filters.

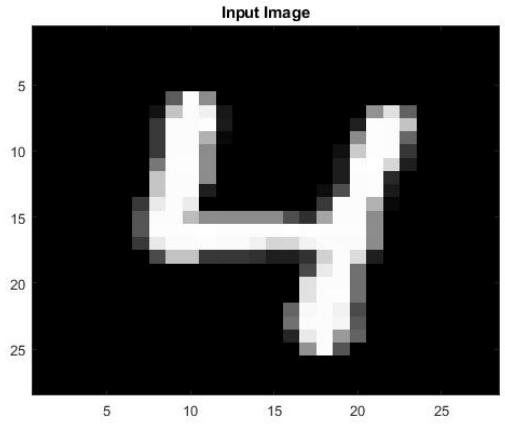


FIGURE 3.7: The presented image to the networks: an example from digit "4".

Figure 3.8 illustrates the results of the processing of the presented image after the convolution layer. This figure consists of 20 images obtained from the 20 filters. This figure shows the various changes of the input image performed by the convolution filters.

Figure 3.9 illustrates the results of the processing after the mean pooling process. This figure consists of 20 images, which each of them has a size of 10*10. This layer aims to reduce the size of the presented images.

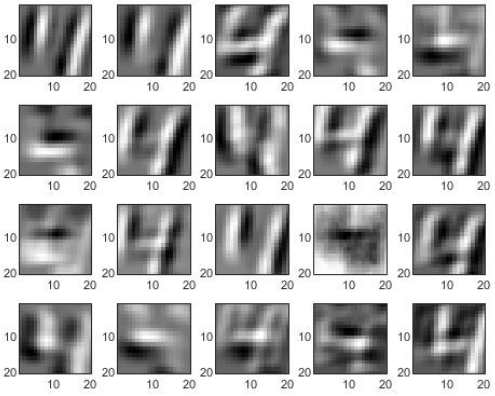


FIGURE 3.8: The outputs of the convolution layer of the trained CNN.

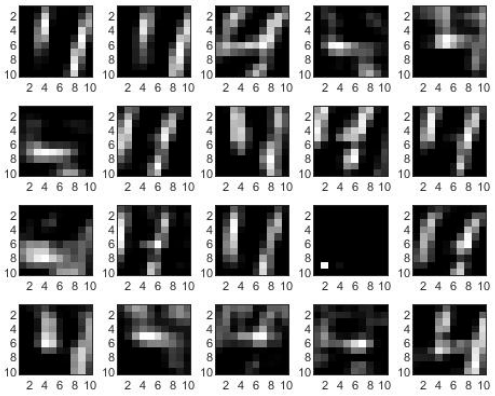


FIGURE 3.9: The outputs of the pooling layer of the trained CNN.

3.4.2 Recursive neural networks

Recurrent Neural Networks (RNNs) are based on supervised deep learning algorithms used for sequential data. These networks are specifically designed to model space-temporal structures because they consider information from various previous layers [45]. In the RNN model, the current hidden layer is a nonlinear function of the previous layer (s) and the current input (x). Individual neurons in this network can be connected to themselves or other neurons. Figure 3.10 illustrates the schema of RNN. Unlike feedforward neural networks, recurrent neural networks have feedback connections that have internal states. This connection provides them with a memory that can store information about previous inputs, making them useful for time series data where features representing the past are expected to affect the future. RNNs are ideal for text and speech analysis, speech-to-text or language-to-language translation, and predictive maintenance [13].

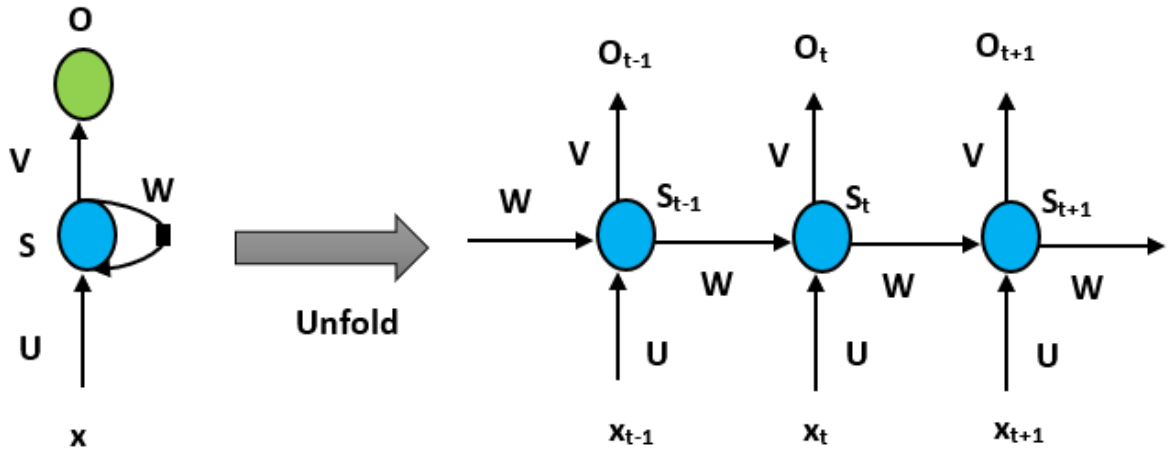


FIGURE 3.10: The general structure of Recurrent neural networks.

The memory is encoded in an internal state and modified as indicated in the equation below:

$$St = g[U.x_t + W.S_{t-1}] \tag{3.1}$$

Where S_t is the hidden state in step t. U (input-to-hidden) and W (hidden-to-hidden) are weight matrices that indicate the relevance of the current input and previous state, respectively. The output of the activation function is calculated using a third weight matrix V (hidden-to-output), as shown in the equation below:

$$O_t = g[V.S_t] \tag{3.2}$$

Backpropagation through time, an extension of backpropagation that includes temporality for computing gradients, is commonly used to train RNNs. When dealing with long temporal sequences, this strategy can cause several problems. Gradients can become immeasurably large or incredibly small when composed over a long sequence. These issues are called exploding gradients and vanishing gradients:

1. **Vanishing gradients:** during the learning phase, the amplitude of error gradients disappears exponentially, making it impossible for the RNN to learn the correlation between long-term temporal events.

2. **Exploding gradients:** a sharp increase in the amplitude of gradients during the learning phase, where the long-term components grow exponentially and dominate the gradients of the short-term components [13].

Explosive gradients are more easily solved as they can be shortened or squashed. On the other hand, the problem of vanishing gradients is more challenging as the propagated gradients can become too small for networks to learn.

To address these problems, other types of RNNs, such as long-term memories (LSTMs) [32], have been introduced. LSTMs can learn very long-term dependencies. They perform better than RNNs and hidden Markov models (HMMs). LSTM models are a kind of RNN architecture introduced in 1997 by Hochreiter and Schmidhuber [13, 32] that successfully eliminates the issue of vanishing gradients by maintaining a more constant error using gated cells, allowing for continuous learning over a more significant number of time steps. Figure 3.11 illustrates a typical LSTM cell. The activations of the input, output, and forget gate vectors in a conventional LSTM

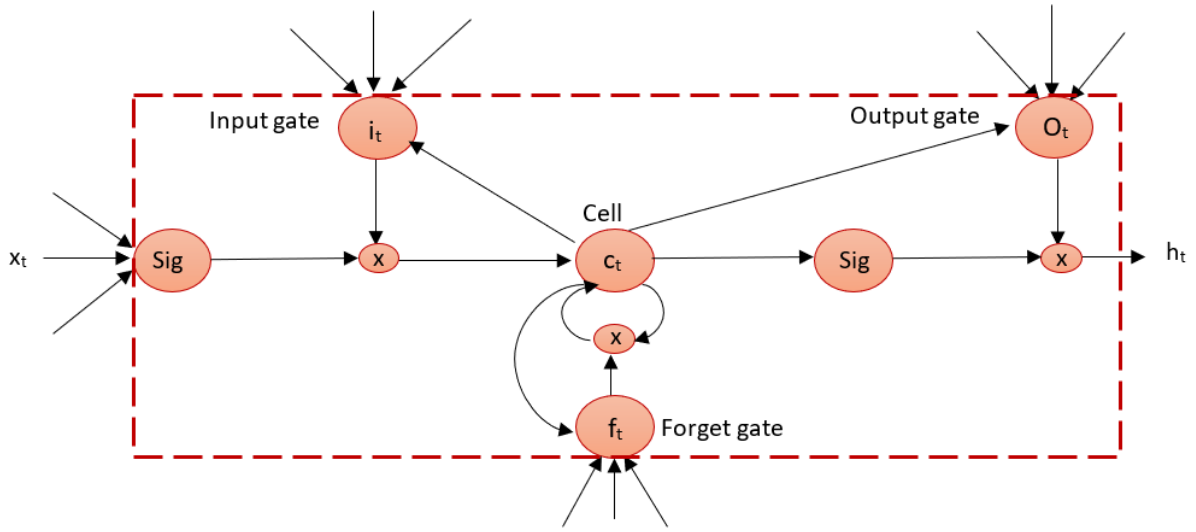


FIGURE 3.11: A long-short term memory model.

are as follows:

$$i_t = g(W_i x_t + U_i h_{t-1}) \tag{3.3}$$

$$O_t = g(W_o x_t + U_o h_{t-1}) \tag{3.4}$$

$$f_t = g(W_f x_t + U_f h_{t-1}) \tag{3.5}$$

The following equation gives the activation of the cell state vector:

$$c_t = f_t \circ c_{t-1} + i_t \circ g(W_c x_t + U_c h_{t-1}) \tag{3.6}$$

Where $\circ$ denotes the Hadamard product. Finally, the activation of the output gate vector is given by the following equation:

$$h_t = O_t \circ g(c_t) \tag{3.7}$$

LSTM gated cells have an internal recurrence in addition to the external recurrence of RNNs. The cells keep an internal state that can be written to and read from them. Gates control how data enters, leaves, and is removed from this cell state. These gates act on the signals they obtain like traditional neural networks, and they use their weights to block or pass information based on its strength and relevance. The weights that control the input and hidden states are modified during the learning process of the recurrent networks. Through an iterative process of making predictions, backpropagating error, and adjusting weights using gradient descent, the cells learn when to allow data to enter, leave, or be deleted. This model architecture enables successful learning from long sequences, capturing many time scales and remote dependencies.

3.4.3 Autoencoders

Deep autoencoders are among the most successful approaches in unsupervised deep learning techniques. They are feedforward neural network that tries to learn a representation of the input data. An autoencoder is trained to efficiently encode the input data and then reconstruct the inputs from that encoding Figure 3.12 . The output of the autoencoder is the same as the input. It typically has three layers:

1. The input layer is used for inputting feature vectors.
2. The hidden layer is used to represent map features, and it has a smaller number of units than the input or output layers.
3. The output layer is used to represent the reconstruction of the input

Similar to other neural networks, the backpropagation technique is used to learn the parameters usually via the stochastic gradient descent method.

Assume that the autoencoder architecture contains only one linear hidden layer and the mean square error criterion is used as the loss function to train the network. An autoencoder will operate like the principal component analysis (PCA) method and learn the first principal components of the data [7]. To obtain more advantages from the autoencoder rather than the dimensionality

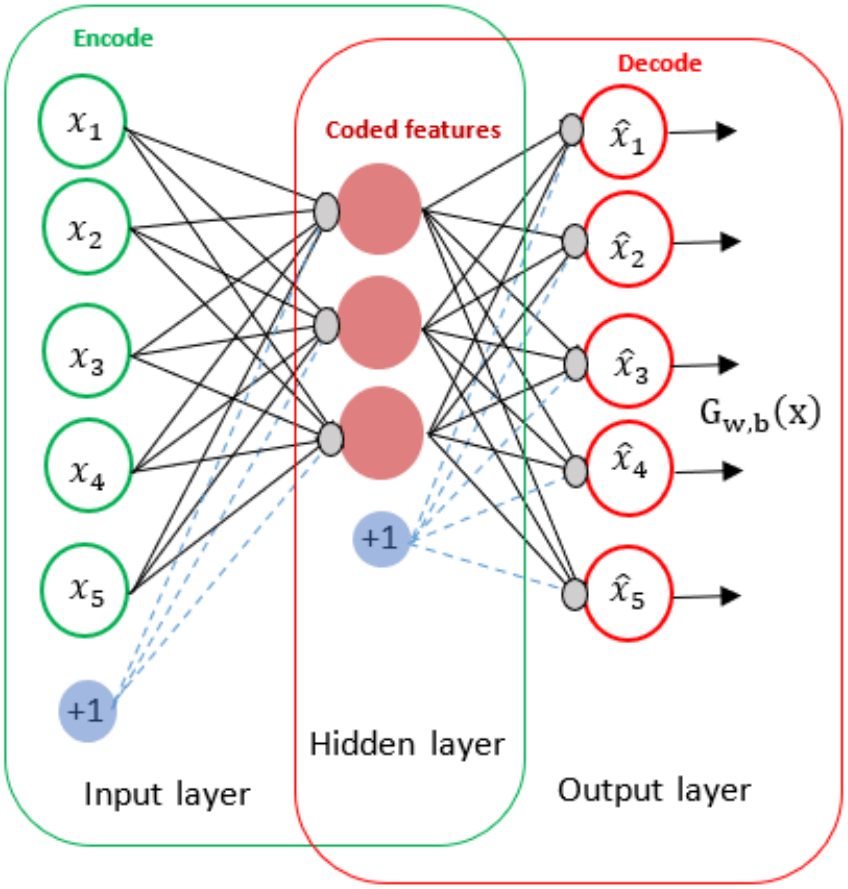


FIGURE 3.12: The basic representation of autoencoders.

reduction method, non-linear hidden units are used in the hidden layer. To achieve the more expressive capability of autoencoders, stacked autoencoders are constructed by stacking autoencoders in sequence [48]. The output of each autoencoder is connected to the input of the next autoencoder. Figure 3.13 illustrates the main architecture of stacked autoencoders.

3.4.3.1 Sparse autoencoder

Sparse autoencoder (SAE) aims to learn sparse features by adding a penalty term inspired by sparse coding [52]. This penalty is added to the cost function to ensure that the features learned are not just a simple repetition of inputs. The sparse penalty aims at minimizing the number of "active" hidden neurons. Let us consider that $a_j(x)$ is the activation of the j^{th} hidden neuron. Consequently, the average activation of this neuron is given by:

$$\rho_j = \frac{1}{n}\sum_{i=1}^{n}[a_j(x(i))] \tag{3.8}$$

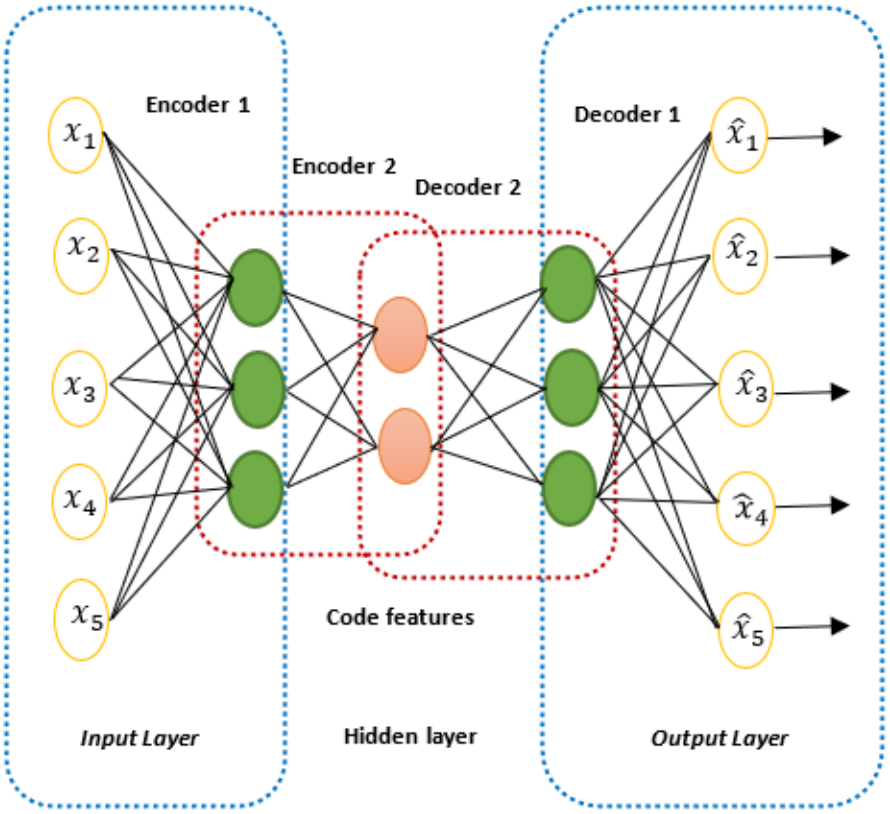


FIGURE 3.13: A stacked autoencoder with two stacked encoders.

Where, n is the dimension of the feature space The sparsity can be achieved by adding a regularization term that indicates the difference between the mean activation value, $\widehat{\rho_j}$, and a sparsity target value,ρ. This may be made using the Kullback-Leibler discrepancy as follows [52] :

$$\Omega_{spar} = KL(\rho \parallel \widehat{\rho_j}) = \rho log\frac{\rho}{\widehat{\rho_j}} + (1-\rho)log\frac{1-\rho}{1-\widehat{\rho_j}} \tag{3.9}$$

The cost function is then given as follows:

$$f = MSE(X - \widehat{X}) + a\Omega_{spar} + \beta\Omega_w \tag{3.10}$$

$MSE(X - \widehat{X})$ is the mean squared error, and Ω_w is the sum squared of all the network weights.

3.4.3.2 Denoising autoencoder

We can obtain an autoencoder that learns useful information by minimizing the reconstruction error of the corrupted input version (a random noise is added to the input data) and attempts to restore the original undistorted input data and removes the noise. This autoencoder is called

a denoising autoencoder. Denoising autoencoders are a crucial tool for feature selection and extraction. The model maps the input data to a lower dimensional, describing the raw data to cancel the added noise. By this means, the encoder will extract the essential features and learn a more robust representation of the data.

Two main ideas characterize this technique. First, when there is noise in the input data, denoising will lead to higher representations that are more robust and stable. Second, utilizing the denoising task will force the extraction of features that significantly impact the helpful structure of the input distribution [79].

3.5 Application of DL methods in a medical field

Over the past decade, a wide range of artificial intelligence and machine learning techniques have been used to effectively analyze massive amounts of data in healthcare. Among the various ML models, DL-based techniques have attracted a lot of attention, especially in analyzing large datasets. Mainly, DL technology has achieved progress in several areas of the healthcare system, such as diagnosis, public health, clinical decision-making, and therapeutics. DL techniques are multi-stage feature learning techniques that filter data through a cascade of multi-layer. While DL models become increasingly accurate when dealing with large-scale data, they outperform many traditional machine learning models in segmentation, detection, and classification. DL-based approaches have also demonstrated remarkable performance in image processing and natural language processing [1, 28]. Considering the performance of DL approaches in different domains and their rapid methodological improvement, these models are becoming new and exciting tools for analyzing health data. A wide range of initiatives has been conducted using biomedical and healthcare data DL models. For example, Google's DeepMind [58] and IBM's Watson (International Business Machines Corporation) have developed a computer-based support system to analyze healthcare data [12].

DL methods are classified into single DL and hybrid DL techniques. Single DL methods build their models entirely on the deep learning architecture. On the other hand, hybrid DL refers to DL methods combined with other traditional machine learning models.

Deep learning in health informatics has shown many advantages: it can be trained without feature extraction by human experts, which burden on clinicians. This has also expanded the possibilities of health informatics research. Nevertheless, the application of deep learning to

health informatics raises several critical challenges that need to be addressed, including data informativeness (high dimensionality, heterogeneity, multimodality), lack of data (missing values, class imbalance, expensive labeling, fairness, and bias), data credibility and integrity, model interpretability and reliability (tracking and convergence issues as well as overfitting), feasibility, security, and scalability [68].

3.6 Conclusion

DL techniques have shown promising results in medical diagnosis in recent years. They outperformed the current methods of ML in segmentation, detection, and classification phases. In this chapter, we have defined the objective and techniques of DL and their application in the medical field.

The next chapter will present our proposed models, which are based on deep neural networks, especially the sparse autoencoders.

Chapter 4

Proposed models and application to ECG classification

Contents

4.1 Introduction

The electrocardiogram (ECG) is one of the most basic and oldest cardiac tests available. It can provide a wealth of helpful information and remains an essential part of the evaluation of cardiac patients. Cardiovascular diseases are considered as the leading cause of death in the developed world. Preventing these deaths requires long-term monitoring and manual inspection of the ECG signals, which is time-consuming. Consequently, creating methods and systems is essential to automatically categorize beats and assist cardiologists in their diagnoses [59].

In this chapter, we propose two systems for automatic heartbeat classification. These models are based on neural networks. The first system is based on two models of neural networks: sparse autoencoder and multiple MLPs, while the second is based on three neural models: SAE, RBFNN, and RVFLN. These models were tested on the MIT-BIH arrhythmia dataset.

4.2 Automatic systems of ECG heartbeat classification

4.2.1 Principle of the electrocardiogram

The ECG is a record of the electrical activity of the heart. The propagation of electrical activity through the membranes of the heart cells can be measured by electrodes placed on the surface of the skin and connected to an electrocardiograph, a signal acquisition device. The electrodes can be placed in various ways to capture different leads. Each lead allows the measurement of voltage differences in a specific direction, resulting in diverse but highly correlated measurements. An electrocardiogram lead consists of a pair of electrodes: positive and negative electrodes.

There are twelve conventional ECG lead placements that constitute the routine 12-lead ECG. The 12 ECG leads are limb leads (extremity leads) and chest leads (precordial leads). Figure 4.1 illustrates the different electrocardiogram leads: limb and chest leads.

The limb leads are derived from electrodes placed on the limbs. One electrode is placed on each of the three limbs: the right arm (RA), the left arm (LA), and the left leg (LL). The right leg (RL) electrode serves as the grounding electrode. The limb leads are devised into two types: standard limb leads and augmented limb leads [23].

There are three standard limb leads:

- Lead I - measures electrical potential between right arm (-) and left arm (+)
- Lead II - measures electrical potential between right arm (-) and left leg (+)
- Lead III - measures electrical potential between left arm (-) and left leg (+)

There are three augmented limb leads:

- Lead aVR (right arm);
- Lead aVL (left arm)
- Lead aVF (left foot).

The 'aV' in these terms stands for 'augmented Voltage'.

The Chest leads are obtained from electrodes placed on the precordium in designated areas Figure 4.1 . An electrode can be placed at six different positions on the left side of the chest, with each position representing one lead [23, 47]. There are six chest leads, namely: Lead V1; Lead V2; Lead V3; Lead V4; Lead V5; Lead V6.

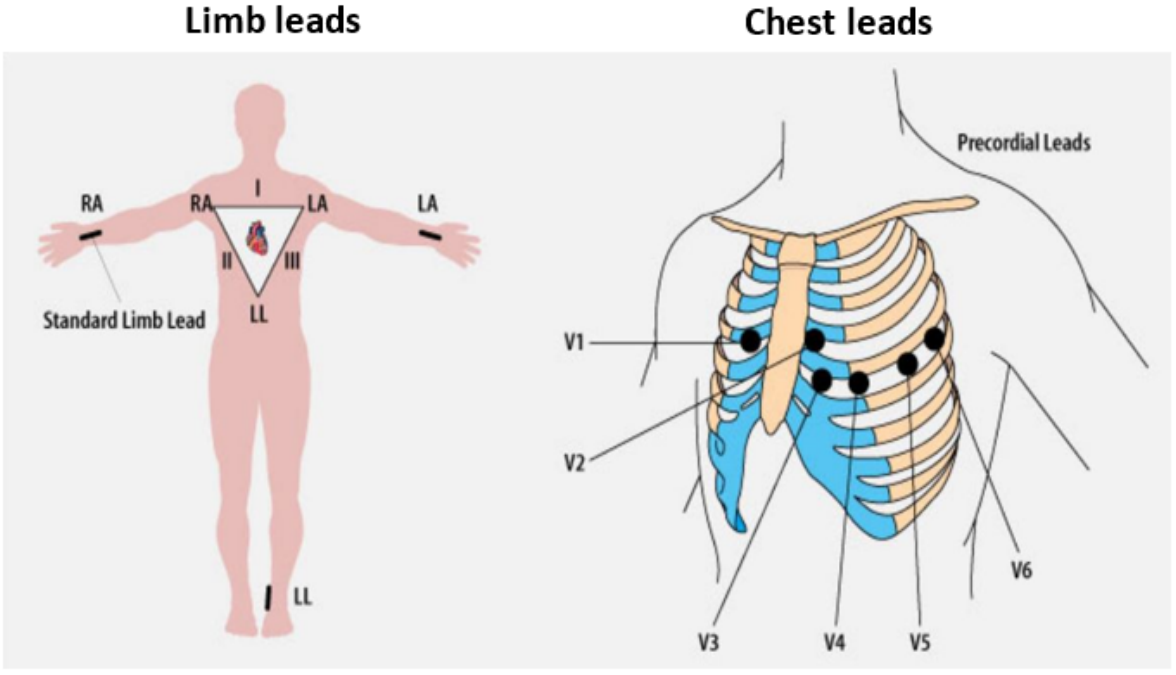


FIGURE 4.1: The 12 electrocardiogram leads.

4.2.2 Electrocardiogram waves and intervals

The ECG is composed of waves and segments, each of them corresponds to a different cardiac cycle stage. Figure 4.2 illustrates an example of a normal ECG heartbeat. An ECG beat includes four distinct complexes: P, QRS, T, and U. Electrocardiogram analysis is generally based on the measurement of amplitudes, durations, and examination of the morphology of the P wave, PR interval, QRS complex, ST-segment, T wave, and QT interval. The normal values cited below, which apply to middle-aged adults, are given as a guide because there is sometimes a significant overlap between normal and pathological values.

1. **The P wave:** corresponds to the contraction of the atria rooms of the heart. This wave has a duration of fewer than 0.12 seconds while its amplitude is less than 0.25 mV.

2. **The PR Interval:** is measured from the onset of the P wave to the beginning of the QRS complex. It represents the propagation time of the impulse from the atria through the atrioventricular node, the His bundle, and its branches. The duration of the PR interval varies from 0.12 to 0.20 seconds, depending on heart rate and age.

3. **The QRS complex:** reflects the contraction of both ventricles; its duration is 0.08 seconds. The QRS complex is a combination of Q, R, and S waves associated with ventricular contraction.

   - The Q peak is negative and of low amplitude.
   - The R peak is positive and of high amplitude.
   - The S peak is negative and of low amplitude.

4. **The ST segment:** represents the end of ventricular conduction (or depolarization) and the beginning of ventricular recovery (or repolarization).

5. **The T wave** corresponds to the relaxation of the ventricular contraction and the repolarization of the myocardium.

6. **QT Interval:** is the distance between the beginning of the QRS complex and the end of the T wave. It represents ventricular depolarization and repolarization. The duration of the QT interval varies with heart rate, age, and gender.

7. **The U wave:** occasionally appears and is not used for diagnosis.

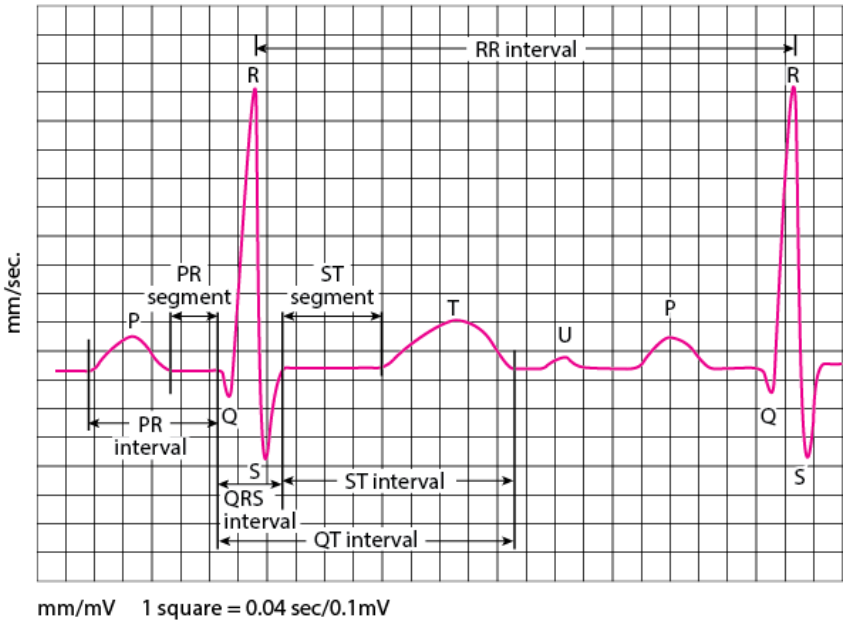


FIGURE 4.2: Electrocardiogram waves, intervals, and segments.

The shape and size of these waves vary depending on the lead from which the signal is observed. As a result, different characteristics can be seen from different leads, which facilitates the diagnosis of specific diseases from specific leads. Most methods in the literature use lead II to distinguish arrhythmias.

4.2.3 Type of classification systems

Heartbeat classification methods can be divided into two major categories based on how the preprocessing, feature extraction, and classification steps are carried out. The first category includes traditional approaches in which feature extraction and classification are performed separately. Whereas the second group includes techniques based on deep learning in which feature extraction and classification are performed automatically in the same architecture.

In the first category, the obtained features and the classifier model significantly impact the overall performance. Various feature extraction techniques have been developed in recent years. This includes principal component analysis (PCA), linear discriminant analysis algorithms (LDA), wavelet transform (WT), etc.

The second category includes deep learning-based approaches that automatically extract practical features from raw input ECG data. These approaches reduce the need for time-consuming and complex feature extraction phases.

4.2.4 ECG features

Common features of ECG signals are statistical, intervals, dynamic (RR-interval), and morphological features. In the next paragraphs, we briefly describe some examples of these features.

1. The statistical features are used to extract summary statistics from variations of the ECG signal amplitude in the time domain. The standard features include the mean, maximum, minimum, and higher-order statistical moments such as variance, kurtosis, and skewness [63].

2. The features based on the intervals are calculated after localizing the fiducial points. Among these intervals, the QRS duration and the T-wave duration are the most commonly used. Figure 4.3 illustrates an example of detecting some fiducial points, i.e., R peak, P peak, QRS onset, and QRS offset.

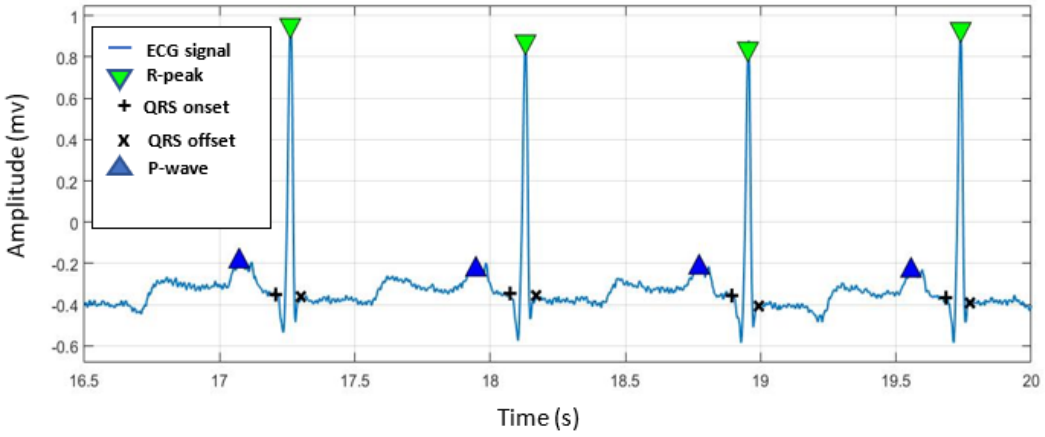


FIGURE 4.3: Detection of some fiducial points: R peak, P wave, QRS onset and QRS offset.

3. The dynamic features are based on the intervals between R peaks of different heartbeats. This includes the previous and posterior RR intervals, as well as the mean and local mean RR intervals. The previous RR interval is the RR interval between a particular heartbeat and the previous heartbeat. The posterior RR interval is between a specific heartbeat and the next one. The mean RR interval is the average of the RR intervals for a recording and is the same for all heartbeats in the recording. The local mean RR interval is the average of the RR intervals of only some RR intervals surrounding a heartbeat [19]. Figure 4.4 illustrates an example of RR-interval features, i.e., post and previous.

4. Most of the time-frequency methods presented in the literature are based on wavelet transforms. The wavelet transform allows extracting information from both frequency and time

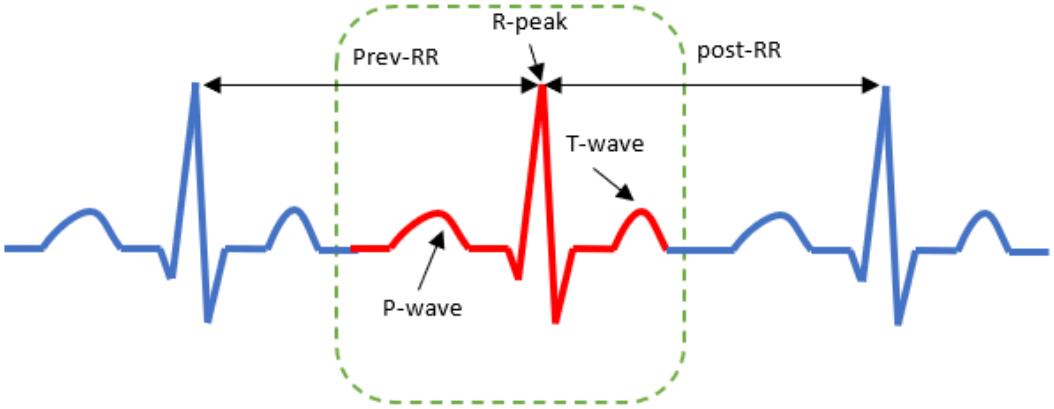


FIGURE 4.4: Example of RR interval features: Previous RR interval (prev-RR) and posterior RR interval (post-RR).

domains. They are different from the traditional Fourier transform, which is based only on the frequency domain.

5. The morphological feature can be given by simple techniques, such as interpolation which reduce the number of points representing the heart-beat. An example of this technique is presented by de Chazal et al. [19], in which each heartbeat is sub-divided and represented by 18 samples (see Figure 4.5).

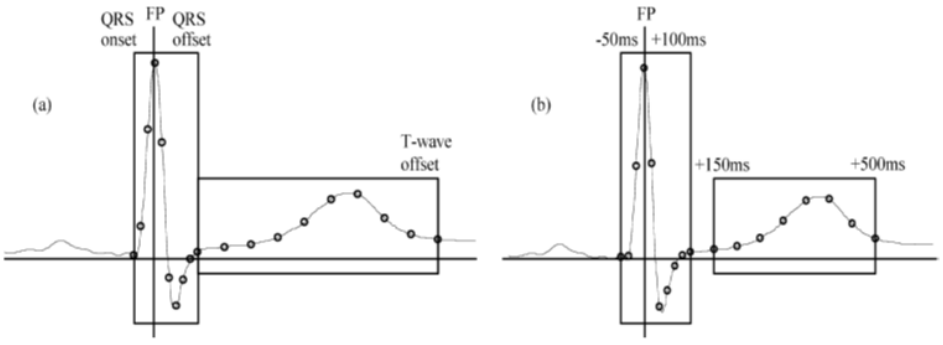


FIGURE 4.5: Two examples of sub-sampling methods to extract morphological features after the determination of the fiducial point (FP):

(a) Sub-sampling based on the QRS onset, QRS offset, and T-wave offset.

(b) Sub-sampling based on FP-50ms, FP+100ms, FP +150ms, and FP+500ms [19].

4.3 Arrhythmia MIT-BIH dataset

4.3.1 Description

The MIT-BIH dataset is the first test material widely available for evaluating arrhythmia detection. Since 1980, this dataset has been used for basic cardiac dynamics research at approximately 500 sites worldwide [4, 15, 16]. This dataset contains 48 half-hour excerpts of two-channel sampled at 360 Hz. The recordings were obtained from 48 different subjects studied with the Beth Israel Hospital Arrhythmia Laboratory. These 48 half-hour excerpts were divided into two categories:

- 23 records (numbered from 100 to 124 inclusive with some numbers missing) were chosen randomly from a collection of over 4000 Holter tapes.
- 25 records (numbered from 200 to 234 inclusive, again with some numbers missing) were chosen to include examples of uncommon but clinically essential arrhythmias that would not be well represented in a small random sample of Holter recordings [4]. Figure 4.6 , illustrates an example of ECG signal annotations from the MIT-BIH dataset. Annotation files contain groups of labels, each describing a feature of one or more signals at a specific point in the record.

FIGURE 4.6: Example of annotations in MIT-BIH dataset.

4.3.2 AMII recommendations

According to the Association for the Advancement of Medical Instrumentation (AAMI) recommendations [3], four subjects with paced beats were excluded. The remaining 15 different types of heartbeats were classified into five categories: Normal beat (N), supraventricular ectopic beat

(S), ventricular ectopic beat (V), fusion beat (F), and unknown beat (Q) are different types of beats.

Table 4.1. describes the heartbeat categories recommended by AAMI.

TABLE 4.1: Types of heartbeats in the MIT-BIH dataset recommended by AAMI

AAMI heartbeats	MIT-BIH heartbeats
Normal (N)	Normal beat (N); Left and right bundle branch block beats (L, R); Atrial escape beat (e); Nodal (junctional) escape beat (j)
Supraventricular ectopic beat (S)	Atrial premature beat (A); Aberrated atrial premature beat (a); Nodal (junctional) premature beat (J); Supraventricular premature beat (S)
Ventricular ectopic beat (V)	Premature ventricular contraction (V); Ventricular escape beat (E)
Fusion (F)	Fusion of ventricular and normal beat (F)
Unknown beat (Q)	Paced beat (/); Fusion of paced and normal beat (f); Unclassifiable beat (U)

Figure 4.7 illustrates examples of the five classes of heartbeats recommended by AAMI.

4.3.3 Evaluation paradigms

Three paradigms exist regarding how to evaluate the ECG heartbeat classification system:

The first is the intra-patient paradigm. This paradigm is based only on the heartbeats labels without considering the patients. The ECG heartbeats are randomly divided into training and testing subsets. Therefore, a patient's ECG heartbeats can appear in both subsets.

The second paradigm is called inter-patient. Here, ECG records extracted from some patients are used for training, and records from the other patients are used for testing. A commonly adopted partition was proposed by de Chazal et al. [19]. According to this protocol, the MIT-BIH dataset (44 AAMI-based records) is divided into two sets of records as follows. DS1 = 101, 106, 108, 109, 112, 114, 115, 116, 118, 119, 122, 124, 201, 203, 205, 207, 208, 209, 215, 220, 223, 230 and DS2 = 100, 103, 105, 111, 113, 117, 121, 123, 200, 202, 210, 212, 213, 214, 219, 221, 222, 228, 231, 232, 233, 234. DS1 is used to train the classification model, and DS2 is used to test it.

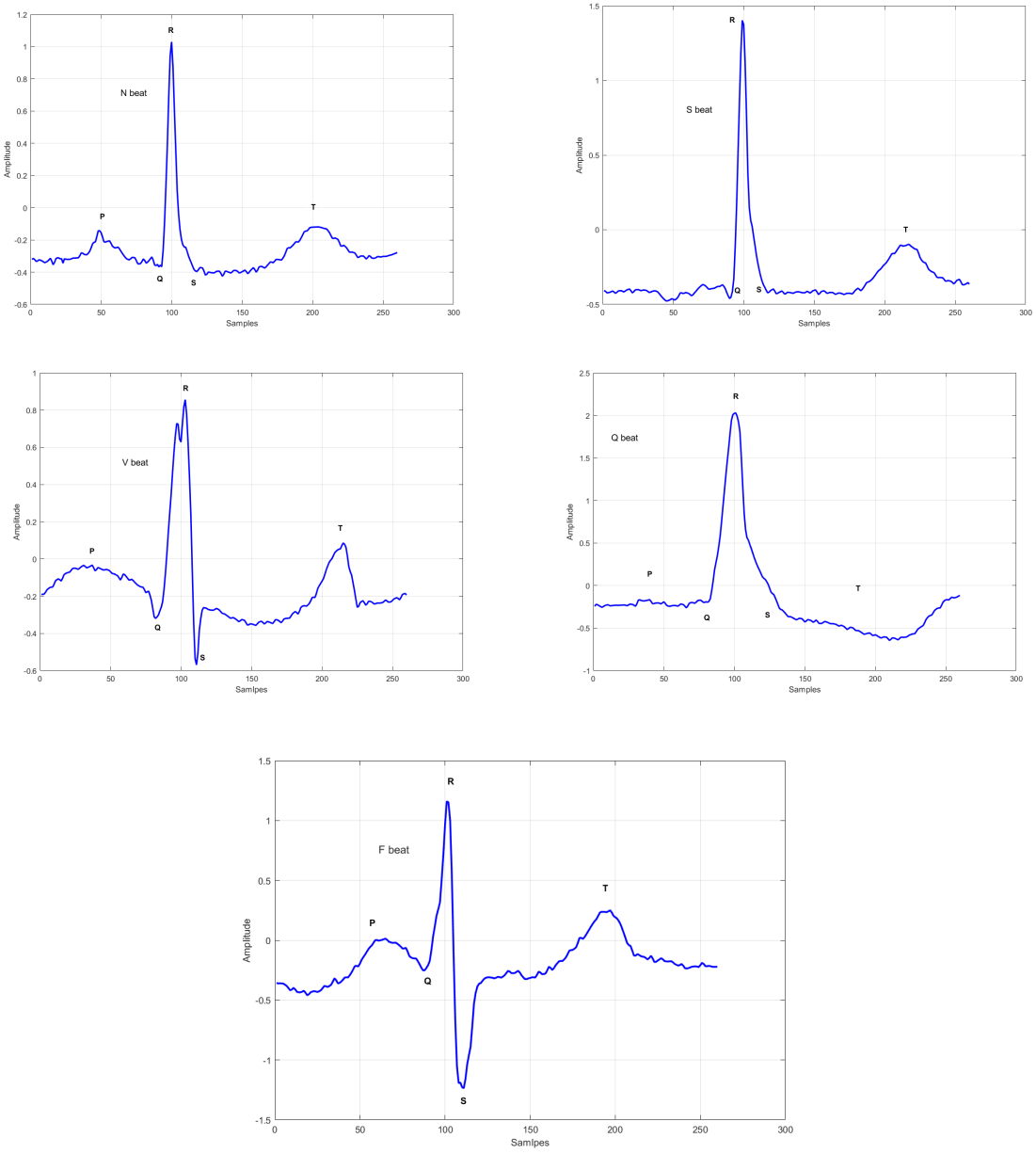


FIGURE 4.7: Examples of heartbeats from different classes taken from the MIT-BIH arrhythmia dataset.

1. Normal beat (N)
2. Supraventricular ectopic beat (S)
3. Ventricular ectopic beat (V)
4. Unknown beat (Q)
5. Fusion beat (F)

When using this splitting method, there is no reason to worry about including heartbeats from the same patient in both the training and testing sets. A patient's heartbeat can be in either the training or test subsets, but not both at the same time.

The third paradigm is patient-specific [84]. In this paradigm, the heartbeats are taken from the same patient. Cardiologists generally assess the first 2–5 minutes of each ECG test recording.

4.4 Handling imbalanced classification problem

The imbalanced problem occurs when the number of instances in some classes is much smaller than in other classes. The main challenge with the imbalanced data is that the standard classifiers are biased by huge classes ignoring the smaller ones. Real-world data are generally imbalanced, and this is one of the leading causes of the decrease in the generalizability of machine learning algorithms that give equal attention to the majority and minority classes. However, the minority classes are generally the most important. This is mainly the case in medical datasets where high-risk patients tend to be the minority class. Therefore, a good sampling technique is needed for medical datasets. Well-balanced data is essential for creating good training.

To overcome the problem of class imbalance, sampling strategies have been used to either delete some data from the majority class (under-sampling) or add some artificially generated or duplicated data to the minority class (over-sampling).

4.4.1 Over-sampling

Over-sampling techniques increase the number of minority class samples in the training set. Over-sampling has the advantage of preserving all information from the original training set because all samples from the minority and majority classes are kept. The disadvantage, however, is that the training set size is significantly increased [29]. Some over-sampling techniques are:

1. Random over-sampling: it is the simplest approach to over-sampling. Samples from the minority class are chosen randomly and then duplicated and added to the new training set [29].

2. Synthetic Minority Over-sampling Technique (SMOTE): In this approach, the minority class is over-sampled by creating synthetic samples rather than over-sampling with alternation. The minority class is over-sampled by taking each minority class sample and introducing synthetic samples along the line segments connecting any/all of the k minority class adjacent neighbors, depending on the amount of over-sampling required. Neighbors from the k nearest neighbors of a record are randomly chosen [29].

3. Adaptive Synthetic sampling approach (ADASYN) [30] The basic idea behind this method is to use a weighted distribution for minority class samples based on their level of complexity in learning, with more synthetic data created for samples that are more difficult to learn than samples that are easier to learn. Accordingly, the ADASYN approach improves learning in two ways: (1) by reducing the bias introduced by class imbalance and (2) by adaptively shifting the classification decision boundary toward difficult samples.

4.4.2 Under-sampling

Under-sampling is an alternative to over-sampling. Under-sampling is a technique for reducing the number of samples in the majority class in order to balance the distribution of the classes. The simplest method of under-sampling, i.e., random under-sampling, is to randomly select a subset of samples from the majority class and combine them with the samples from the minority class [60].

4.5 First proposed system

4.5.1 Main idea and motivations

This system is based on using stacked autoencoders (SSAE) and decomposition strategies of multi-class problems. The main contributions are twofold: (i) using the stacked autoencoder to automatically extract features, and (ii) dividing the original multi-class problem into simpler sub-problems and solving them using a system of multiple neural networks. The motivations are as follows. First, using the capability of stacked autoencoders to represent high-level features and to avoid the pre-processing and feature selection phases. Second, exploiting the advantages of decomposition strategies to solve multi-class classification problems. Third, after the decomposition of the original problem, we use an over-sampling method, namely SMOTE. By applying

the SMOTE after the decomposition, the number of added instances is performed according to the number of training instances in each subproblem. The application of the over-sampling method to each subproblem independently permits optimizing the number of added samples.

Two models are proposed based on the most common decomposition techniques, namely OAA and OAO. As basic classifiers, we use MLPs [74].

4.5.2 Principals and objectives of decomposing multi-class classification problems

Multiclass classification problems are omnipresent in real-world applications. They imply classifying the input instances into multiple classes, like classifying heartbeats into multiple arrhythmia classes. To reduce the complexity of multi-class classification problems, they are usually decomposed into several simple sub-problems. Subsequently, these sub-problems are implemented using multiple classifiers. To classify a new sample, it is presented to all classifiers, and the final decision is made by combining the outputs of the classifiers [53].

Decomposing a multi-class problem into a set of binary problems has the following advantages. First, it simplifies the classification task because binary problems are generally simpler than multi-class problems. Second, most classifiers are mainly designed for binary problems. Third, each subproblem can be treated separately using a different type of classifier. Fourth, each classifier has its structure, parameters, and even feature space [50, 53].

The decomposition of multi-class classification problems can be achieved using various strategies, the most important are OAA (one against all) and OAO (one-against-one).

4.5.2.1 One against all decomposition scheme

The OAA decomposition method divides a K-class classification problem into a set of K subproblems. Each of them aims to classify one class against all the other classes. Therefore, the number of subproblems equals the number of classes. In all subproblems, the entire set of training instances is used; the training data is the same, but the desired outputs are different.

The maximum confidence strategy is the most common and simplest aggregation method. The output class is just the one with the highest response.

$$Class(X) = \arg \max_{i=1...K} Z_i(X) \tag{4.1}$$

Where Z_i is the output of the classifier corresponding to the subproblem: class i against all the other classes.

4.5.2.2 One against one decomposition scheme

The OAO decomposition method classifies each class against another one. It decomposes a problem of K classes into $K(K-1)/2$ subproblems. Accordingly, the number of binary problems is larger than in the OAA method. Still, each subproblem needs less training data because only the samples of two classes are considered. Majority voting and weighted voting are the most commonly used aggregation methods in OAO.

In the majority voting rule, each binary classifier gives one vote for the predicted class. The votes for each class are then counted, and the class with the highest number of votes is considered as the final decision. The decision rule is as follows:

$$Class(X) = \arg \max_{i=1...K} \left(\sum_{1 \leq j \neq i \leq K} I(Z_{ij}(X) > Z_{ji}(X)) \right) \tag{4.2}$$

With $I(.)$, the standard indicator function returns one if its argument is true and zero otherwise, Z_{ij} is the output of the classifier related to the subproblem: class i against class j.

In the weighted voting strategy, each binary classifier votes on both classes. The weight of the vote is determined by the confidence of the classifier in the class prediction. The class with the highest sum value is the resulting class. Accordingly, the decision rule is:

$$Class(X) = \arg \max_{i=1...K} \sum_{1 \leq j \neq i \leq K} Z_{ij}(X) \tag{4.3}$$

4.5.3 Architecture

The architecture of the first proposed system is as follows: The stacked autoencoder is used to extract high-level features from raw ECG signals. These features constitute the input of the

ensemble of MLPs. Each new sample, ECG beat, is presented to all MLPs, and the outputs of these MLPs are used to make decisions. The main scheme of the proposed system is represented in Figure 4.8.

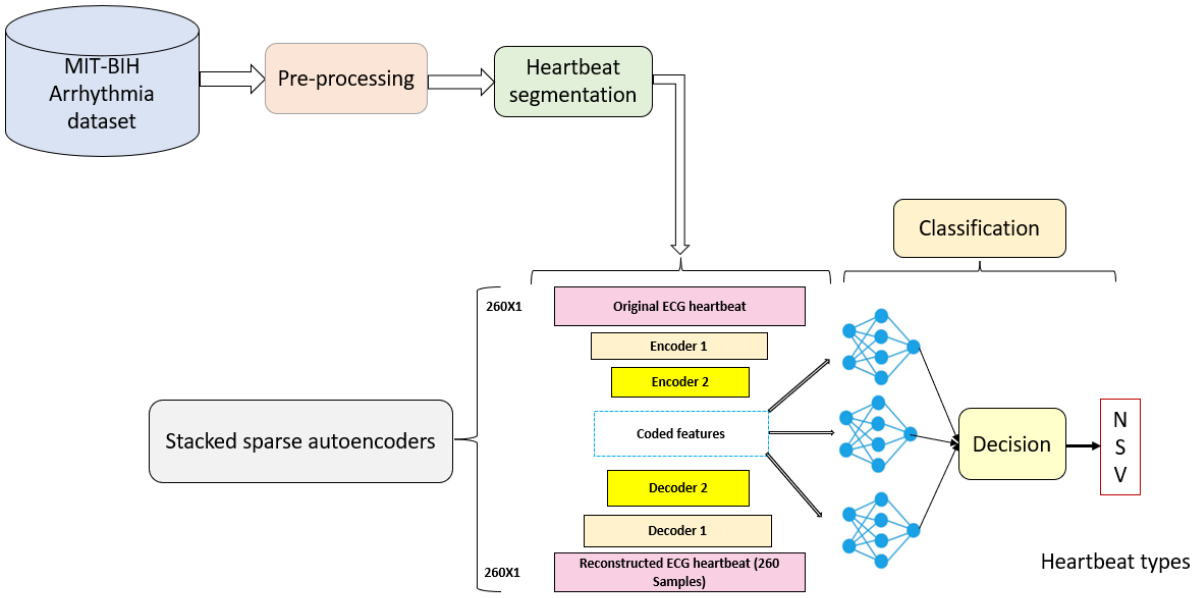


FIGURE 4.8: Principal scheme of the first proposed system.

4.5.4 Training process

There are three phases in the training of the proposed system:

1. Segmentation of the ECG signal.
2. Training the autoencoder using unsupervised data.
3. Decomposing the multi-class problem and training the MLPs using supervised data.

ECG beats segmentation

The segmentation of the ECG records into heartbeats is performed as follows:

1. Find the R peaks of the ECG signal.
2. Divide the continuous ECG signal into a sequence of heartbeats, in which each one consists of 260 samples: 99 samples before the R-peak, 160 samples after the R-peak, and the R-peak itself [49].

Training the autoencoders

The autoencoder is trained in an unsupervised manner, i.e., with unlabeled data. The equations of training SSAE are given in 3.8, 3.9, 3.10 chapter 3. Because of the lack of designing methods of the autoencoders, i.e., the number of hidden layers and the number of features, we propose to perform tests with several structures. We use a validation set, taken from the training data, to evaluate each structure. The validation set is generated according to the protocol proposed by de Chazal et al., in which they divided the training set into two subgroups as follows: DS11 = 101, 106, 108, 109, 114, 115, 116, 119, 122, 209, 223 and DS12 = 112, 118, 124, 201, 203, 205, 207, 208, 215, 220, 230. DS11 is used for the training and DS12 for the evaluation.

Decomposition of the multi-class problem and training the MLPs

The first proposed model is based on decomposing the ECG beat classification problem into simpler subproblems, where each subproblem is treated with a separate classifier. This system takes into consideration the AAMI recommendations, which identify five categories of heartbeats. Accordingly, the classification problem has been divided into three classes. The OAA-MLP and OAO-MLP models are proposed based on the two most extensively used strategies for decomposing multi-class problems, OAA and OAO.

1. **OAA-MLP model**

   This model is based on the OAA approach, in which each sub-problem classifies one class against all the other classes. In a 3-class problem, the proposed OAA-MLP model consists of three binary neural networks (Net_1, Net_2, Net_3). Each network, Net_i, is trained to distinguish between the i^{th} beat class (B_i) and all the other classes. The output $Z_i(X)$ of Net_i indicates whether the presented ECG beat, X, belongs to the class (B_i) or not. The training of each network Net_i is performed using the beats from all classes. The beats from class B_i are labeled positive while the other beats are labeled negative. A new beat is presented to all networks of the system and the decision is taken by combining their outputs. We use the max rule, the output class is the one with the highest response.

2. **OAO-MLP model**

   The OAO-MLP model is based on the OAO method, which considers all possible pairs of classes. For a three classes heartbeat classification problem, the proposed model includes three binary neural networks: Net_{ij} $i = 1, ..., 3-1, j = i+1, ..., 3$. Net_{ij} network is trained to distinguish between the beat class (B_i) and the beat class (B_j). The output of Net_{ij} is Z_{ij} (X), which indicates whether the provided beat, X, belongs to the B_i or B_j classes.

The training of Net_{ij} consists of beats from classes B_i and B_j. A new beat is provided to all networks. The decision is then determined by combining their outputs. We use the majority voting rule, which is the most common. Accordingly, each binary network gives a vote for a predicted class, and the votes for each class are counted. The class with the most votes determines the output class.

4.5.5 Results

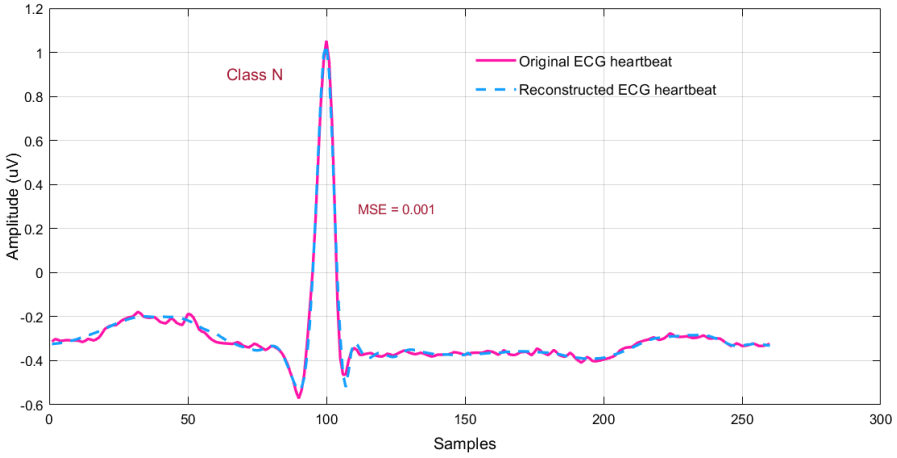
To evaluate the performance of the proposed classifiers, we used the PhysioNet MIT-BIH arrhythmia dataset and we adopted the inter-patient paradigm. In this test, we considered only three classes (N, S, and V). Indeed, the classes F and Q are represented by only a few beats, and they have been ignored in most works [19, 27]. In our experimentation, we first performed many tests on SAEs to define the best structure.

Table 4.2 presents the mean square error between the original and reconstructed ECG heartbeats, corresponding to different structures. We notice that utilizing one layer of SSAE and thirty features produced the best results.

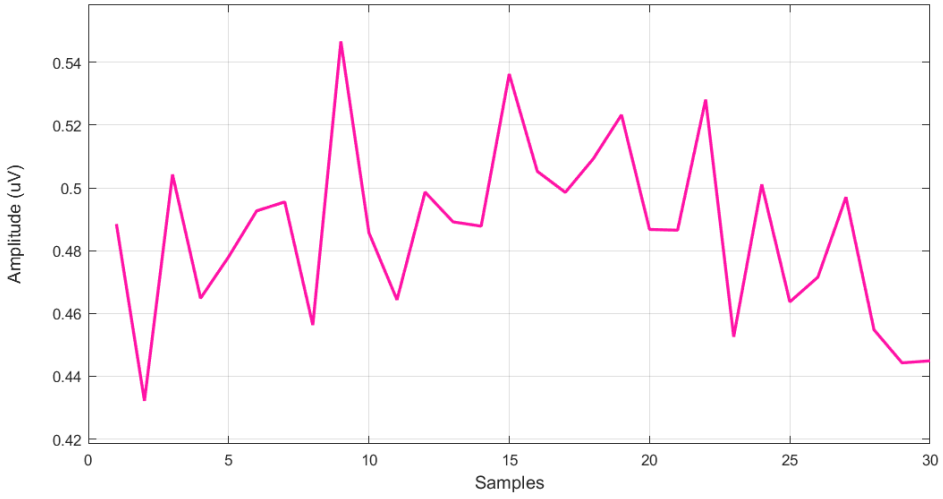
TABLE 4.2: The mean square error between the original and reconstructed ECG heartbeats of some SSAEs structures over MIT-BIH arrhythmia dataset

#Features	Structures of SSAEs	Validation set (MSE)	Test set (MSE)
10	1 AE; 260- 10	0.0048	0.0066
	2 SSAEs; 260-100-10	0.0076	0.0091
	3 SSAEs; 260-160-60-10	0.00812	0.0099
30	**1 AE; 260- 30**	**0.0013**	**0.0015**
	2 SSAEs; 260-100-30	0.0044	0.006
	3 SSAEs; 260-160-60-30	0.0056	0.0078
50	1 AE; 260- 50	0.0014	0.0016
	2 SSAEs; 260-130-50	0.0039	0.0053
	3 SSAEs; 260-160-80-50	0.0054	0.0073

Figure 4.9 displays an example of an ECG beat (from class N) and its corresponding automatic features obtained in the encoding step. In this case, the original signal is coded with 30 features. These features provide a good reconstruction of the original beat, and both the original and reconstructed signals are almost equal.



(A) Original and reconstructed ECG beats (class N).



(B) Automatic features (class N).

FIGURE 4.9: An example of coding an ECG signal using SAE.

4.5.5.1 Results using the OAA-MLP model

Table 4.3 displays the accuracy, sensitivity, specificity, and the number of hidden neurons for each classifier in the OAA-MLP model. The first subproblem, which corresponds to N vs. All, is the most difficult. Even though it had been trained with more hidden neurons and iterations than the other networks, the corresponding network achieved only 93.75%.

TABLE 4.3: The classification results of each classifier in OAA-MLP system over MTI-BIH dataset

# Classifiers	Subproblems	# Hidden neurons	Accuracy (%)	Sensitivity (%)	Specificity (%)
1	N vs. All	45	93.75	73.16	95.58
2	S vs. All	24	95.78	98.65	32.23
3	V vs. All	30	97.79	98.68	86.69

Table 4.4 compares the overall OAA-MLP model with the single MLP. The OAA-MLP model yielded an accuracy of 94.69% on the test data, whereas the single MLP provided only 90.12%. Table 4 shows that the use of a multiple MLP model significantly improved the overall accuracy (Acc), sensitivity (Se), and positive prediction (+P).

TABLE 4.4: Performance of OAA-MLP system and single MLP over MTI-BIH dataset

Classifiers	Accuracy (%)	Class N		Class S		Class V	
		Se(%)	+P(%)	Se(%)	+P(%)	Se(%)	+P(%)
Single MLP	90.12	94.71	94.8	12.9	11.2	81.5	82.1
OAA-MLP system	94.69	98.21	96.64	23.98	47.70	86.53	81.65

4.5.5.2 Results using the OAO-MLP model

Table 4.5 shows the accuracy, sensitivity, specificity, and the number of hidden neurons in each classifier of the OAO-MLP model. The third subproblem, which corresponds to S vs. V, is the most difficult. Its corresponding network yielded only 78.55%. In contrast, the other networks performed better with fewer hidden neurons and iterations.

Table 4.6 shows the performance of a single MLP and the entire OAO-MLP model. The accuracy of the multiple MLP model was 94.65%, while the single MLP model was only 90.12%. The accuracy of the model was then significantly improved by using the multiple MLP method. Sensitivity and positive predictivity were also improved, particularly for N and S classes.

TABLE 4.5: The classification results of each classifier in OAO-MLP system over MTI-BIH dataset

# Classifiers	Subproblems	# Hidden neurons	Accuracy (%)	Sensitivity (%)	Specificity (%)
1	N vs.S	24	95.94	49.13	93.92
2	N vs.V	16	97.82	85.91	99.51
3	S vs.V	41	78.55	18.85	98.99

TABLE 4.6: Performance of OAO-MLP system and single MLP over MTI-BIH dataset

Classifiers	Accuracy (%)	Class N		Class S		Class V	
		Se(%)	+P(%)	Se(%)	+P(%)	Se(%)	+P(%)
Single MLP	90.12	94.71	94.8	12.9	11.2	81.5	82.1
OAA-MLP system	94.65	98.66	96.20	18.92	67.68	82.77	78.09

4.6 Second proposed system

4.6.1 Main idea and motivations

Deep learning methods allow the automatic extraction of features from raw data. These methods avoid the time-consuming and complex feature extraction phases encountered in traditional methods. However, some critical handcrafted features are ignored. Therefore, we propose to combine both automated and handcrafted features. The used features are automatic features extracted by SSAE and two features based on RR intervals, which are very efficient and simple to obtain.

On the other hand, we propose using a hybrid RBF-RVFL neural network as a classifier. The main idea is to design a system that makes use of the complementary properties and advantages of the random neural network, RBF neural network, and stack sparse autoencoder to improve the performance with specific structures. The motivations behind combining RBF and RVFL neural networks are as follows:

First, RBFNN is based on local responses, whereas RVFLN is based on global responses. Consequently, their combination can provide better decision boundaries by using fewer hidden neurons and simple structures.

Second, in the introduced hybrid neural classifier, both RVFLN and RBFNN output weights can be calculated directly in one step. The direct and output weights of RVFLN are concatenated with the output weights of RBFNN. The Moore-Penrose pseudo-inverse solution is then used to calculate all of these weights in a single step. This allows very fast learning.

Third, except for the direct links in RVFLN, which provide linear mapping, both RBFNN and RVFLN provide nonlinear mapping. These two neural networks can map data in both linear and nonlinear manners due to their combination.

Fourth, RVFLN is randomly initialized, whereas RBFNN can be initialized using clustering approaches. Their combination can then reduce the effect of random initialization and allow the use of prior knowledge. In the proposed model, the K-means approach is used to initialize the RBF centers. In order to obtain more accurate RBFs, we propose using this clustering method in a supervised manner, i.e., applied to each class separately. Afterward, the widths of RBFs are calculated using the P-nearest neighbor approach [8].

4.6.2 Process

Figure 4.10 shows the main scheme of the proposed model. First, R-peaks are detected from the original ECG signal. Second, two types of features are calculated: the RR intervals and the automatic features. The latter are obtained using the stack sparse autoencoder. Third, to overcome the problem of imbalanced data, the SMOTE method is applied to the training set. Finally, the automatic features and RR intervals are combined and entered into a hybrid RBF-RVFL neural network to classify the presented heartbeats [75].

1. **ECG beats segmentation**

   The steps followed to extract ECG beats from an ECG signal are mentioned in Section 4.5.4.

2. **RR interval features**

   The time between successive beats is used to determine the RR intervals. In our system,

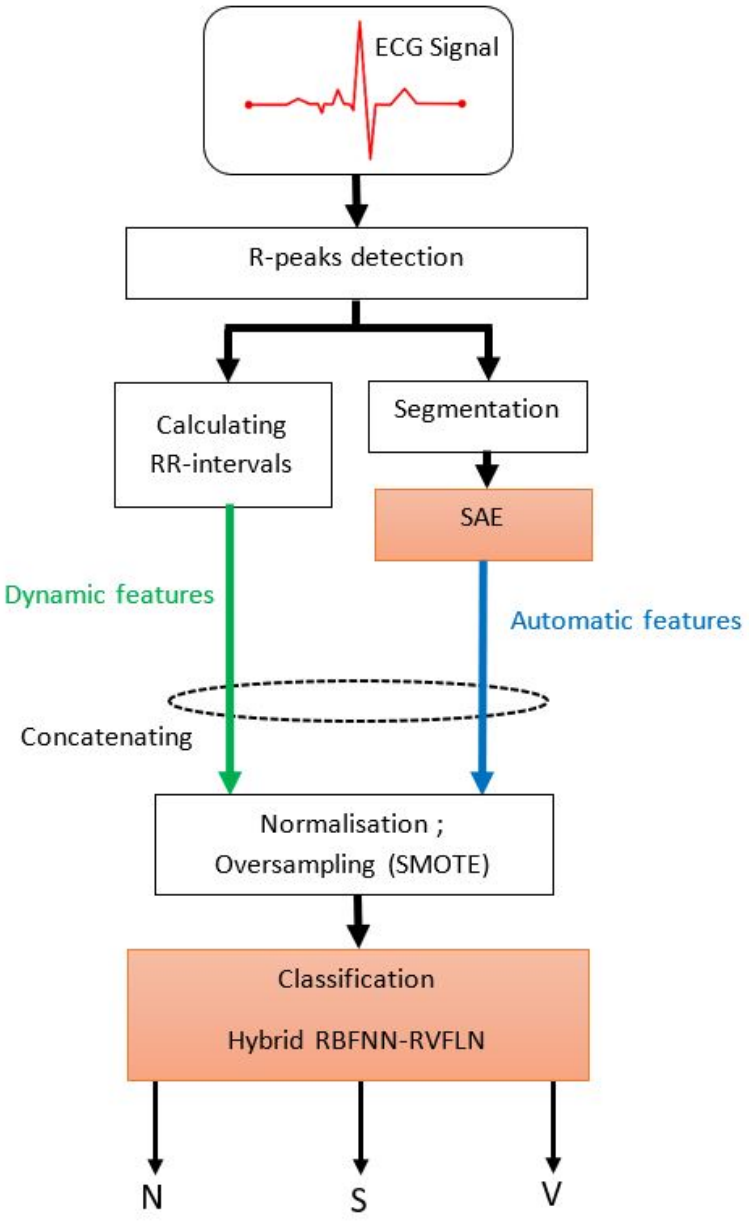


FIGURE 4.10: Block diagram summarizing the stages of the second proposed system.

we use two RR intervals:

$$RRprevious(i) = [R(i) - R(i-1)] * 360 \tag{4.4}$$

$$RRposterior(i) = [R(i+1) - R(i)] * 360 \tag{4.5}$$

Where $R(i)$ represents the position of the i^{th} R-peak; 360 is the signal frequency; and the *RRprevious* and *RRposterior* indicate the distances between the actual heartbeat and the previous one; the actual heartbeat and the next one, respectively.

3. **Training**

   The training of the proposed system is performed in two stages:

   - Training the sparse autoencoder using unsupervised data
   - Training the hybrid neural network using supervised data

   The training of the sparse autoencoder using unsupervised data is performed with the same manner as in the first proposed system.

4.6.3 Architecture and training of the hybrid neural classifier

The proposed classifier system combines an SSAE, RBFNN, and an RVFLN to effectively classify heartbeats. The SSAE is used to extract automatic features from ECG data. The hybrid RBFNN-RVFLN classifier has as inputs the automatic features and two RR interval based-features. The automatic features constitute the inputs of the RBFNN, while the RR intervals constitute the inputs of the RVFLN.

Figure 4.11 illustrates the proposed classifier with $(N + P)$ inputs, D RBF neurons, M enhancement neurons, and J outputs. Each input sample is represented by a concatenation of two vectors: one with N coding features ($X = [x_1; x_2; x_N]$) and one with P dynamic features ($XX = [xx_1; xx_2...; xx_P]$). The different parameters of the proposed system are set as follows:

1. The weights connecting the inputs (coding features) to the RBF neurons are '1'.
2. The RBF centers are determined using the K-means clustering algorithm. To obtain more accurate RBFs, we use this method in a supervised manner, i.e., applied to each class separately. The widths of RBFs are subsequently calculated using the P-nearest neighbors' approach [8]. The width of the d^{th} hidden neuron is given by:

$$\sigma_d = \frac{1}{P}\left(\sum_{n=1}^{P} \parallel V_n - V_d \parallel^2\right)^{\frac{1}{2}} \tag{4.6}$$

   Where, V_n $(n = 1...P)$ are the P-nearest neighbours' of the centre $V_d = [v_{d1}, v_{d2}, ..., v_{dN}]$. The RBF neurons are connected to the vector of automatic features (X). The output of d^{th} RBF neuron is then given by:

$$Y_d(X) = \exp\left(-\frac{\parallel X - V_d \parallel^2}{2\sigma_d^2}\right) \tag{4.7}$$

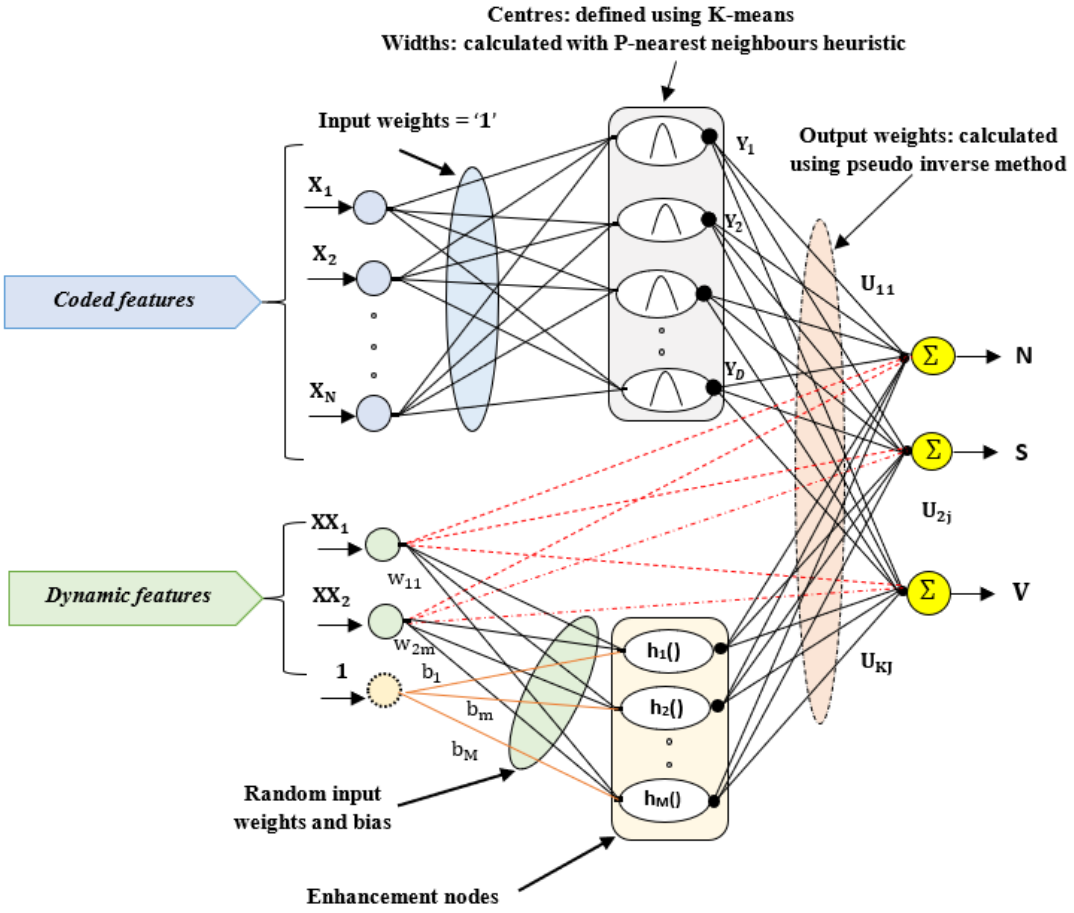


FIGURE 4.11: The structure of hybrid neuronal networks (RBFNN-RVFLN) : the classification part of the second proposed system.

Where $V_d = (v_{1d}, v_{2d}, ..., v_{Nd})$ and σ_d correspond to the d^{th} RBF's center and width, respectively.

3. The input weights $([w_{im}], i = 1...P, m = 1...M)$ that link the input to the enhancement neurons are set randomly.

4. The biases are set randomly $(b_m, m = 1...M)$.

5. The vector of dynamic features (XX) is connected with the enhancement neurons. The outputs of the enhancement neurons are then given by the vector:
$H = [g(W_1 * XX + b_1), \quad g(W_2 * XX + b_2)$

$$... \quad g(W_M * XX + b_M)]^T \tag{4.8}$$

Where $W_m = [w_{1m}, w_{2m}, \ldots, w_{Pm}]$ and b_m are the vectors of input weights and the bias corresponding to the m^{th} enhancement neuron, and g is the activation function.

6. The classifiers output vector is as follows: $Z = S \times U$ Where, S is the vector that concatenates the vectors of RBF outputs (Y), dynamic features (XX), and enhancement neurons (H): $S = [Y \;\; XX \;\; H]$

7. Using all training samples, the output weights ($\mathbb{U}$) matrix is calculated in one step. Consider a training dataset with Q training samples and a vector (X) with N coding features and a vector (XX) with P dynamic features for each sample.

   $X_q = [x_{q1}, x_{q2}, ..., x_{qN}]$ and $XX_q = [xx_{q1}, xx_{q2}, ..., xx_{qP}]$ are the feature vectors corresponding to the q^{th} sample. $T_q = [t_{q1}, t_{q2}, ..., t_{qJ}]$ is the q^{th} target vector.

   The matrix $\mathbb{T}$, which represents the entire set of targets, is as follows:

$$\mathbb{T} = \begin{bmatrix} t_{11} & ... & t_{1J} \\ : & ... & : \\ t_{Q1} & ... & t_{QJ} \end{bmatrix}_{Q \times J} \tag{4.9}$$

   Let $\mathbb{S}$ be a matrix that combines the dynamic inputs, RBF outputs, and enhancement outputs. $\mathbb{S}$ is then calculated as follows: $\mathbb{S} = [\mathbb{Y} \;\; \mathbb{XX} \;\; \mathbb{H}]_{Q \times (D+P+M)}$ where ,

$$\mathbb{XX} = \begin{bmatrix} xx_{11} & ... & xx_{1P} \\ : & ... & : \\ xx_{Q1} & ... & xx_{QP} \end{bmatrix}_{Q \times P} ,$$

$$\mathbb{H} = \begin{bmatrix} g(W_1.XX_1 + b_1) & ... & g(W_M.XX_1 + b_M) \\ : & ... & : \\ g(W_1.XX_Q + b_1) & ... & g(W_M.XX_Q + b_M) \end{bmatrix}_{Q \times M}$$

and

$$\mathbb{Y} = \begin{bmatrix} y_1(X_1) & \dots & y_D(X_1) \\ \vdots & \dots & \vdots \\ y_1(X_Q) & \dots & y_D(X_Q) \end{bmatrix}_{Q \times D} \tag{4.10}$$

The output matrix for all training samples is:

$$\mathbb{Z} = \mathbb{S} \times \mathbb{U} \tag{4.11}$$

Finally, the output weights ($\mathbb{U}$) are determined in a single step as follows:

$$\mathbb{U} = \mathbb{S}^{\ddagger}\mathbb{T} \tag{4.12}$$

Where, $\mathbb{S}^{\ddagger}$ is the moore-penrose generalized inverse of $\mathbb{S}$.

4.6.4 Results

In this model, we have also considered three classes (N, S, and V) recommended by AAMI. Table 4.7 presents the confusion matrix obtained for the 49273-test data. This matrix illustrates the actual and predicted classes of the ECG heartbeats. The rows in this matrix represents the actual class, i.e., annotations in the dataset, while the column represents the predicted classes. We notice that 45876 ECG heartbeats were correctly classified using the proposed model, with an overall accuracy of 93.11%, while the remaining 3397 ECG beats were misclassified (assigned to wrong classes).

TABLE 4.7: The confusion matrix obtained using the proposed system over MIT-BIH arrhythmia dataset

Classes	N	S	V	Total
N	43817	60	318	44195
S	1536	245	65	1846
V	1370	48	1814	3232
Total	46723	353	2197	49273

Table 4.8 summarizes the obtained results: sensitivity (Se), positive prediction (+ P), and F1 score (%) for each class, as well as the average value of each metric. According to Table 7, the lowest performance was obtained in the recognition of the S class. This is because there are just

a few training samples that represent this class. The proposed classifier had an overall accuracy of 93.11%, with average sensitivity, positive prediction, and F1 scores of 56.18%, 81.92%, and 61.83%, respectively.

TABLE 4.8: Performance of the proposed system on MIT-BIH arrhythmia dataset

Overall accuracy (%)		93.11	
Classes	Sensitivity(%)	+predictive(%)	F1-score(%)
N	99.14	93.78	96.39
S	13.27	69.41	22.28
V	56.13	82.57	66.82
Average	56.18	81.92	61.83

4.7 Comparison and analysis of the results

Many studies of classifying ECG heartbeats have been evaluated using the MIT-BIH arrhythmia dataset. A summary of some of these works is shown in Table 4.9. Several feature extraction and classification methods were included in these works, such as neural networks, SVM, CNN, MLP, ELM network, logistic regression, and other machine learning and deep learning techniques.

Table 4.9 compares the proposed models with these works. We notice the following:

The first proposed system outperformed all the compared works. It also outperformed our second proposed system. Indeed, the first model provides better results but with larger architecture: it includes a system of multiple MLPs in the classification phase.

The second proposed system outperformed most of these works, except two: the model of Shi, Qin, et al.(2020) [72] and that of Wang et al. (2020) [80]. Although these two studies provided better results, our system is much simpler because it is based only on one layer of SAE and it considers a smaller number of features. Indeed, the model of Shi et al.[72] is composed of several layers: a convolutional layer, a pooling layer, an LSTM layer, a concatenated layer, a fully connected layer, and four input layers. This model is based on automatic features, obtained from the first convolution layer, and seven RR interval features added to the fully connected layer. The other work (Wang et al., 2020) proposed a dual fully connected neural network classifier.

It is a two-layer classifier in which each layer contains two independent fully connected neural networks.

TABLE 4.9: Comparison of the classification results on MIT-BIH arrhythmia dataset

Authors, year	Extraction features method	Classifier	Performance (%)
Zhang et al., 2014 [86]	F1	Combined SVM	Acc: 87.88
Garcia et al., 2017 [27]	TVCG + complex network + PSO	SVM	Acc: 92.4
Luo et al., 2017 [42]	Three layers SDA + multi-layer DNN	DNN	Acc: 89.3
Chen et al., 2017 [14]	Projections + wRR	SVM	Acc: 93.1
Raj et al., 2018 [62]	Sparse Decomposition	PSO optimized LSTwin SVM	Acc: 89.93; Sen: 72.35
Li et al., 2019 [41]	F2	CNN	Acc: 91.44
Shi et al., 2019 [71]	F3	XGBoost	Acc: 92.1
Shi et al., 2020 [72]	Raw data + RR	MIDNN	Acc: 94.2
Wang et al., 2020 [80]	CNN	CNN	Acc: 93.4
Song et al., 2020 [76]	RC Network + ITML	Logistic Regression	Acc: 89.11; Sen: 74.13; Spe: 95.95
Wang et al., 2020 [81]	CNN + RR-intervals	MLP	Acc: 92.53
Yan et al., 2021 [83]	Raw heartbeat data	CNN + SNN	Acc: 90
OAA-MLP system	SSAEs	MLPs	Acc: 94.69; Sen: 71; Spe: 89.53
OAO-MLP system	SSAEs	MLPs	Acc: 94.65; Sen: 66.78; Spe: 87.96
RBF-RVFLN-SAE system	SAE + RR-intervals	RBFNN + RVFLN	Acc: 93.11; Sen: 55.66; Spe: 80.55

F1: RR-intervals, wavelet coeff.,morphological features; RR: the time between the R peaks of two heartbeats; TVCG: temporal vectorcardiogram; MIDNN: multiple input layers deep neural network; SDA: stacked denoising autoencoder; F2: morphological, RR-intervals,beat-to-beat correlation feature; F3: RR-intervals, morphological, statistical ,higher order statistic,wavelet transform, wavelet packet entropy; wRR: weighted RR; PSO: particle swarm optimization; LST: least square Twin; RC Network: residual-concatenate network; ITML: information-theoretic metric learning; SNN; spiking neural network; XGBoost: extreme gradient boosting

4.8 Conclusion

In this chapter, we have presented two models for classifying ECG heartbeats.

In the first model, high-level features are extracted using SSAE and the classification is performed by a system of multiple ANNs, which is based on decomposition techniques of the multi-class classification problems. This first model includes two versions according to the most commonly used strategies for decomposing multi-class problems, namely OAA and OAO.

The second model consists of a SSAE as a feature extractor and a hybrid neural model as a classifier. The latter combines random neural networks and RBFNN. In addition, this model

includes two types of features, i.e., automatic features (obtained using SAE) and RR-interval based features.

To overcome the problem of imbalanced data, which occurs in most medical problems, we have proposed to use the synthetic minority over-sampling technique (SMOTE). To evaluate the two proposed models, we have conducted experiments on the well-known MIT-BIH arrhythmia dataset and performed the inter-patient test. The first proposed system outperformed all the compared works, including our second proposed system. Indeed, the first model provides better results but with larger architecture. In overall, the obtained results are promising for further application.

General conclusion

This thesis addressed the ECG heartbeat classification using neuro-collaborative approaches. Our goal was to develop classifiers that provide good performance with simple structure. Specifically, we were interested in combining the properties of different types of artificial neuronal networks. As a feature extractor, we used SSAE to capture automatic high-level features from the raw ECG data. As classifiers, we used a system of multiple MLPs and a hybrid random-RBF neural network. We proposed two neural models for ECG heartbeat classification.

The first model includes an SSAE as a feature extractor and a system of multiple MLPs as a classifier. This model is based on the decomposition methods, in which the original problem is divided into several binary sub-problems. The decomposition methods have been effectively applied in various real-world domains to handle the multi-classification problems, which generally occur in the medical field. Accordingly, we proposed two versions (OAA-MLP and OAO-MLP) using the two most common decomposition strategies, OAA and OAO. To deal with the problem of imbalanced data, the proposed model includes an over-sampling method, i.e., SMOTE. More precisely, SMOTE is applied after the decomposition of the original problem in order to add synthetic samples according to the number of training samples in each sub-problem.

The second model includes three types of neural networks and two types of features. In this model, SSAE is used as a feature extractor and a hybrid neural model is used as a classifier. The latter combines RBF and random neural networks in order to make use of the advantages and complementary properties of these two neural networks. To address the problem of imbalanced data, this model also includes the SMOTE technique.

The proposed models were implemented and tested on the public MIT-BIH arrhythmia dataset. We adopted the recommendation of the Association for the Advancement of Medical Instrumentation (AAMI), which considers three classes of interest. We considered the inter-patient paradigm, in which the ECG records extracted from some patients are used for training, and records from other different patients are used for testing. Therefore, there is no reason to worry about including heartbeats from the same patient in both the training and testing sets. The obtained results were satisfactory compared to other state-of-the-art methods. It is worth mentioning that the first proposed system outperformed all the compared works, including our second proposed system. This can be explained by the fact that the first model has larger architecture.

As future work, we plan to further explore the idea of combining automatic features with handcrafted features. We want to develop classification models based on deep learning architectures, allowing the integration of some critical handcrafted features. Therefore, we can have high

performance with compact structures. We also plan to extend these models to specific-patient applications and potential use with long-time wireless body area network monitoring. Indeed, this type of system requires simple, yet effective data-analysis models.

Bibliography

[1] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[2] S Abirami and P Chitra. Energy-efficient edge based real-time healthcare support system. In *Advances in Computers*, volume 117, pages 339–368. Elsevier, 2020.

[3] EC57 ANSI-AAMI. Testing and reporting performance results of cardiac rhythm and st segment measurement algorithms assoc. adv. *Med. Instrum., Arlington, VA*, 1998.

[4] Ziti Fariha Mohd Apandi, Ryojun Ikeura, and Soichiro Hayakawa. Arrhythmia detection using mit-bih dataset: A review. In *2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA), Kuching, Malaysia*, pages 1–5. IEEE, 2018.

[5] Denis Arrivault, Noël Richard, Christine Fernandez-Maloigne, and Philippe Bouyer. Collaboration entre approches statistiques et structurelle pour la reconnaissance de caractères anciens. In *Conférence Internationale Francophone sur l'Ecrit et le Document (CIFED 04)*, 2004.

[6] Nadia Benahmed. *Optimisation de réseaux de neurones pour la reconnaissance de chiffres manuscrits isolés: Sélection et pondération des primitives par algorithmes génétiques.* École de technologie supérieure, 2002.

[7] Yoshua Bengio. *Learning deep architectures for AI.* Now Publishers Inc, 2009.

[8] Nabil Benoudjit, Cédric Archambeau, Amaury Lendasse, John Aldo Lee, Michel Verleysen, et al. Width optimization of the gaussian kernels in radial basis function networks. In *ESANN, Bruges (Belgium)*, volume 2, pages 425–432, 2002.

[9] James C Bezdek. On the relationship between neural networks, pattern recognition and intelligence. *International journal of approximate reasoning*, 6(2):85–107, 1992.

[10] David S Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[11] Jason Brownlee. *Better deep learning: train faster, reduce overfitting, and make better predictions*. Machine Learning Mastery, 2018.

[12] Miriam Cabrita, Harm op den Akker, Monique Tabak, Hermie J Hermens, and Miriam MR Vollenbroek-Hutten. Persuasive technology to support active and healthy ageing: An exploration of past, present, and future. *Journal of biomedical informatics*, 84:17–30, 2018.

[13] Adrian Carrio, Carlos Sampedro, Alejandro Rodriguez-Ramos, and Pascual Campoy. A review of deep learning methods and applications for unmanned aerial vehicles. *Journal of Sensors*, 2017, 2017.

[14] Shanshan Chen, Wei Hua, Zhi Li, Jian Li, and Xingjiao Gao. Heartbeat classification using projected and dynamic features of ecg signal. *Biomedical Signal Processing and Control*, 31:165–173, 2017.

[15] George B Moody and Roger G Mark. The impact of the mit-bih arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, 2001.

[16] PhysioNet. PhysioNet: MIT-BIH arrhythmia databasee. https://archive.physionet.org/cgi-bin/atm/ATM, 2001. [Online; accessed 1-July-2018].

[17] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[18] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[19] Philip De Chazal, Maria O'Dwyer, and Richard B Reilly. Automatic classification of heartbeats using ecg morphology and heartbeat interval features. *IEEE transactions on biomedical engineering*, 51(7):1196–1206, 2004.

[20] Mathieu Delalandre, Pierre Héroux, Sébastien Adam, Eric Trupin, and Jean-Marc Ogier. Une approche statistico-structurelle pour la reconnaissance de symboles exploitant une

représentation xml des données. In *Colloque International Francophone sur l'Ecrit et le Document, Hammamet, Tunisie*, pages 121–128, 2002.

[21] Li Deng and Dong Yu. Deep learning: methods and applications. *Foundations and trends in signal processing*, 7(3–4):197–387, 2014.

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition, Miami, FL, USA*, pages 248–255. Ieee, 2009.

[23] Gordon E Dower, H Bastos Machado, and JA Osborne. On deriving the electrocardiogram from vectorcardiographic leads. *Clinical cardiology*, 3(2):87–95, 1980.

[24] RO Duda, PE Hart, and DG Stork. Pattern classification, 2. ausgabe, 2001.

[25] Caroline Etienne. *Apprentissage profond appliqué à la reconnaissance des émotions dans la voix*. PhD thesis, Université Paris-Saclay, 2019.

[26] Brahim FAROU. Système hybride utilisant les réseaux de neurones et les modèles de markov cachés pour la reconnaissance de mots manuscrits arabe. Magister ,2009.

[27] Gabriel Garcia, Gladston Moreira, David Menotti, and Eduardo Luz. Inter-patient ecg heartbeat classification with temporal vcg optimized by pso. *Scientific reports*, 7(1):1–11, 2017.

[28] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[29] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[30] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), Hong Kong*, pages 1322–1328. IEEE, 2008.

[31] Donald Olding Hebb. *The organization of behavior: a neuropsychological theory*. Science editions, 1949.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[33] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[34] John J Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984.

[35] Anil K Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):4–37, 2000.

[36] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[38] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[39] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[40] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[41] Feiteng Li, Yin Xu, Zhijian Chen, and Zhenyan Liu. Automated heartbeat classification using 3-d inputs based on convolutional neural network with multi-fields of view. *IEEE Access*, 7:76295–76304, 2019.

[42] Kan Luo, Jianqing Li, Zhigang Wang, and Alfred Cuschieri. Patient-specific deep architectural model for ecg classification. *Journal of healthcare engineering*, 2017, 2017.

[43] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[44] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[45] Larry Medsker and Lakhmi C Jain. *Recurrent neural networks: design and applications*. CRC press, 1999.

[46] Marvin Minsky and Seymour Papert. Perceptron: an introduction to computational geometry, 1969.

[47] Sucharita Mitra, Mandar Mitra, and Bidyut Baran Chaudhuri. A rough set based approach for ecg classification. In *Transactions on Rough Sets IX*, pages 157–186. Springer, 2008.

[48] Sajad Mousavi, Michael Schukat, and Enda Howley. *Researching advanced deep learning methodologies in combination with reinforcement learning techniques*. PhD thesis, Master's thesis, National University of Ireland Galway, 2018.

[49] Fatma Murat, Ozal Yildirim, Muhammed Talo, Ulas Baran Baloglu, Yakup Demir, and U Rajendra Acharya. Application of deep learning techniques for heartbeats detection using ecg signals-analysis and review. *Computers in biology and medicine*, 120:103726, 2020.

[50] Mohamed Nemissi, Hamid Seridi, and Herman Akdag. One-against-all and one-against-one based neuro-fuzzy classifiers. *Journal of Intelligent & Fuzzy Systems*, 26(6):2661–2670, 2014.

[51] Mohamed Nemissi. *Classification et reconnaissance des formes par algorithmes hybrides*. PhD thesis, 2009.

[52] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[53] Guobin Ou and Yi Lu Murphey. Multi-class pattern classification using neural networks. *Pattern Recognition*, 40(1):4–18, 2007.

[54] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In *Science and information conference, Cham, Switzerland*, pages 128–144. Springer, 2019.

[55] Y-H Pao and Yoshiyasu Takefuji. Functional-link net computing: theory, system architecture, and functionalities. *Computer*, 25(5):76–79, 1992.

[56] Theodosios Pavlidis. *Structural pattern recognition*, volume 1. Springer, 2013.

[57] Léon Personnaz and Isabelle Rivals. *Réseaux de neurones formels pour la modélisation, la commande et la classification*. CNRS, 2003.

[58] Julia Powles and Hal Hodson. Google deepmind and healthcare in an age of algorithms. *Health and technology*, 7(4):351–367, 2017.

[59] Dallas Price. How to read an electrocardiogram (ecg). part 1: Basic principles of the ecg. the normal ecg. *South Sudan Medical Journal*, 3(2):26–31, 2010.

[60] M Mostafizur Rahman and Darryl N Davis. Addressing the class imbalance problem in medical datasets. *International Journal of Machine Learning and Computing*, 3(2):224, 2013.

[61] Mohamed BEN RAHMOUNE. *Diagnostic des défaillances d'une turbine à gaz à base des réseaux de neurones artificiels pour l'amélioration de leur système de détection des vibrations*. PhD thesis, thèse de doctorat, Université Ziane Achour de Djelfa, 2017.

[62] Sandeep Raj and Kailash Chandra Ray. Sparse representation of ecg signals for automated recognition of cardiac arrhythmias. *Expert systems with applications*, 105:49–64, 2018.

[63] Bashar Rajoub. Machine learning in biomedical signal processing with ecg applications. In *Biomedical Signal Processing and Artificial Intelligence in Healthcare*, pages 91–112. Elsevier, 2020.

[64] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. *Encyclopedia of database systems*, 5:532–538, 2009.

[65] João PS Rosa, Daniel JD Guerra, Nuno CG Horta, Ricardo MF Martins, Nuno CC Lourenço, et al. *Using artificial neural networks for analog integrated circuit design automation*, volume 1. Springer, 2020.

[66] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[67] F Rosenblatt. Principles of neurodynamics: perceptrons and the theory of brain mechanisms' to van der malsburg c.(1986) frank rosenblatt: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. *Brain Theory. Berlin, Heidelberg: Springer*, 1962.

[68] Shahab Shamshirband, Mahdis Fathi, Abdollah Dehzangi, Anthony Theodore Chronopoulos, and Hamid Alinejad-Rokny. A review on deep learning approaches in healthcare systems: Taxonomies, challenges, and open issues. *Journal of Biomedical Informatics*, 113:103627, 2021.

[69] Priyanka Sharma and Manavjeet Kaur. Classification in pattern recognition: A review. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(4), 2013.

[70] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *towards data science*, 6(12):310–316, 2017.

[71] Haotian Shi, Haoren Wang, Yixiang Huang, Liqun Zhao, Chengjin Qin, and Chengliang Liu. A hierarchical method based on weighted extreme gradient boosting in ecg heartbeat classification. *Computer methods and programs in biomedicine*, 171:1–10, 2019.

[72] Haotian Shi, Chengjin Qin, Dengyu Xiao, Liqun Zhao, and Chengliang Liu. Automated heartbeat classification based on deep neural network with multiple input layers. *Knowledge-Based Systems*, 188:105036, 2020.

[73] Ivan Nunes da Silva, Danilo Hernane Spatti, Rogerio Andrade Flauzino, Luisa Helena Bartocci Liboni, and Silas Franco dos Reis Alves. Artificial neural network architectures and training processes. In *Artificial neural networks*, pages 21–28. Springer, 2017.

[74] Roguia Siouda, Mohamed Nemissi, and Hamid Seridi. Ecg beat classification using neural classifier based on deep autoencoder and decomposition techniques. *Progress in Artificial Intelligence*, 10(3):333–347, 2021.

[75] Roguia Siouda, Mohamed Nemissi, and Hamid Seridi. A random deep neural system for heartbeat classification. *Evolving Systems*, pages 1–12, 2022.

[76] Xinjing Song, Gongping Yang, Kuikui Wang, Yuwen Huang, Feng Yuan, and Yilong Yin. Short term ecg classification with residual-concatenate network and metric learning. *Multimedia Tools and Applications*, 79(31):22325–22336, 2020.

[77] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern recognition*. Elsevier, Third Edition, 2006.

[78] Sergios Theodoridis. Konstantinos koutroumbas. *Pattern recognition*, 2003.

[79] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[80] Haoren Wang, Haotian Shi, Ke Lin, Chengjin Qin, Liqun Zhao, Yixiang Huang, and Chengliang Liu. A high-precision arrhythmia classification method based on dual fully connected neural network. *Biomedical Signal Processing and Control*, 58:101874, 2020.

[81] Tao Wang, Changhua Lu, Mei Yang, Feng Hong, and Chun Liu. A hybrid method for heartbeat classification via convolutional neural networks, multilayer perceptrons and focal loss. *PeerJ Computer Science*, 6:e324, 2020.

[82] Bernard Widrow and Marcian E Hoff. Adaptive switching circuits. Technical report, Stanford Univ Ca Stanford Electronics Labs, 1960.

[83] Zhanglu Yan, Jun Zhou, and Weng-Fai Wong. Energy efficient ecg classification with spiking neural network. *Biomedical Signal Processing and Control*, 63:102170, 2021.

[84] Lukáš Zaorálek, Jan Platoš, and Václav Snášel. Patient-adapted and inter-patient ecg classification using neural network and gradient boosting. *Neural Network World*, 28(3):241–254, 2018.

[85] Amer Zayegh and Nizar Al Bassam. Neural network principles and applications. *Digital Systems*, 2018.

[86] Zhancheng Zhang and Xiaoqing Luo. Heartbeat classification using decision level fusion. *Biomedical Engineering Letters*, 4(4):388–395, 2014.

[87] Le Zhang and Ponnuthurai N Suganthan. A survey of randomized algorithms for training neural networks. *Information Sciences*, 364:146–155, 2016.

Author's publications

International publications

ROGUIA Siouda, MOHAMED Nemissi, et HAMID Seridi. Diverse activation functions based-hybrid RBF-ELM neural network for medical classification. Evolutionary Intelligence, 2022, p. 1-17.

ROGUIA Siouda , MOHAMED Nemissi, et HAMID Seridi . A random deep neural system for heartbeat classification. Evolving Systems, 2022, p. 1-12.

ROGUIA Siouda , MOHAMED Nemissi, et HAMID Seridi. ECG beat classification using neural classifier based on deep autoencoder and decomposition techniques. Progress in Artificial Intelligence, 2021, vol. 10, no 3, p. 333-347.

ROGUIA Siouda , MOHAMED Nemissi. An optimized RBF-neural network for breast cancer classification. International Journal of Informatics and Applied Mathematics, 2019, vol. 1, no 1, p. 24-34.

International Communications

ROGUIA Siouda, MOHAMED Nemissi, and HAMID Seridi. "Medical classification using an RBF neural network based on auto-encoder and particle swarm optimization". International Conference on Pattern Analysis and Recognition ICPAR, Tebessa - Algeria 2019.

ROGUIA Siouda, MOHAMED Nemissi, HAMID Seridi, and MUHAMMET Kurulay. "A new approach for arrhythmia classification using stacked autoencoders and multilayer perceptron", Conference on Innovative Trends in Computer Science (CITCS'2019) 20-21 November 2019, Guelma-Algeria

ROGUIA Siouda, MOHAMED Nemissi, and HAMID Seridi. "A genetic algorithm-based

deep RBF neural network for medical classification". Proceedings of the 1st International Conference on Intelligent Systems and Pattern Recognition. October, 2020, Hammamet, Tunisia, doi :10.1145/3432867.3432868.

National Communications

ROGUIA Siouda, MOHAMED Nemissi, "Classification using an optimized RBF-Neural network", 1st national Conference on Informatics and Applied Mathematics IAM'2018, 2018.