Self-organizing Maps for Digital Pre-distortion

MICHAIL ISAAKIDIS
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Self-organizing Maps for Digital Pre-distortion

Michail Isaakidis
mi3105is-s@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisors:
Liang Liu (LTH)
Ove Edfors (LTH)
Asad Jafri (Ericsson)

Examiner: Erik Larsson

September 20, 2020

Printed in Sweden
Tryckeriet i E-huset, Lund

List of Acronyms

3GPP 3rd Generation Partnership Project

ACLR Adjacent Channel Leakage Ratio

AM/AM Amplitude to Amplitude

AM/PM Amplitude to Phase

BMU Best Matching Unit

DLA Direct Learning Architecture

DPD Digital Pre-Distortion

DRC Design Rule Check

DSP Digital Signal Processor

FLOPs FLoating point Operations Per second

ILA Indirect Learning Architecture

IMD Intermodulation Distortion

IP Internet Protocol

LS Least Square

LTE Long-Term Evolution

LUT LookUp Table

ML Machine Learning

PA Power Amplifier

PAPI Performance Application Programming Interface

PCA Principal Component Analysis

SOM Self-Organizing Map

W-CDMA Wideband Code Division Multiple Access

Abstract

Power amplifiers are very important components in the area of wireless communications. However, they are non-linear devices and they seem to achieve high efficiency when they are operating in the non-linear regions, at the cost of being more power-consuming. In order to linearize their behavior, designers have been considering many methods. Among those, the digital pre-distortion tends to be the most popular one.

There are different ways of applying the digital pre-distortion. Two of the most common are the pseudo-inverse and gradient descent methods. Both those two methods require too much power, as a result of the high amount of computation. For this reason, new power-efficient methods are under research.

The main focus of this master thesis is to explore the possibility of applying the digital pre-distortion with the use of self-organizing maps, a machine learning algorithm, to develop a more efficient digital pre-distortion (DPD) unit. The algorithm implementation and the simulations were performed with the use of MATLAB and the neural network toolbox.

During this research, the performance, the accuracy and the computational complexity of the implemented design were measured before and after the fine tuning of some of the system's parameters. The target is to show if the use of this algorithm is an efficient method of implementing digital pre-distortion.

Popular Science Summary

Digital pre-distortion: A new approach

The power amplifier is an electronic device that is used in order to increase the magnitude of the power of a given input signal. It is used for devices like speakers, headphones and RF transmitters. It is considered as an essential element within the wireless communication area, due to the fact that it is used for the transmission and the broadcasting of the signals to the users. In addition to those, with the use of power amplifiers and the increase of the power levels, higher data transfer rates became available. What basically a power amplifier does, from a computation perspective, is that it receives an input signal and multiplies it with the desired gain. But that is an ideal model of a power amplifier. However, the power amplifiers are non-linear sources for a communication system, so the real model differs a lot from the ideal one. They tend to be non-linear as their output power increases and reaches close to its maximum value, which can create an in-band distortion within the system. For this reason, the linearization of power amplifiers is a very important topic under research in the digital communications field.

The most common method for linearizing a power amplifier's behavior is the digital pre-distortion. This method is very power efficient as well as cost-saving. Ideally, with the pre-distortion, the characteristics of a power amplifier are inverted in order to compensate for the non-linearities. The role of a pre-distorter unit inside a digital communication system is to correct any possible gain and phase nonlinearities and in combination with the system's amplifier to produce a "clear" out of distortion signal. With the use of the pre-distortion and the gain stability that it can provide in the output of the amplifiers, the construction of bigger, more expensive and less efficient amplifiers is no longer necessary. Despite the fact that the pre-distortion is widely and successfully used, it has been observed that the current ways of applying the pre-distortion are very power-consuming due to their complexity and the amount of computational power they require in order to perform the pre-distortion. For that reason, new approaches are under the scope.

Nowadays, there is a trend in electronics where many concepts are being implemented with the use of machine learning in order to replace the number

of computations with simpler logic, such as the classification of the data and the prediction of the desired values. It is easy to understand, that a high amount of computations inside a system means more power which leads to a higher operating cost. As a result, a more power-efficient system could lead to a big saving in terms of power and money.

This project evaluates the impact of the self-organizing maps algorithm as an adaptive algorithm for the digital pre-distortion. The main goal is to reduce computations and provide a more power-efficient system. It could be a guideline for future researches with new approaches that might lead to more energy-saving results.

Contents

List of Figures

List of Tables

Chapter 1

Background

This introductory chapter explores the basic concepts of power amplifiers and machine learning, that were investigated during this thesis work. By the end of the chapter, the reader will be familiar with all the necessary basic terms and their integration to this project. In addition to that, there will be a discussion about the main motivations and the challenges that lead to the choice of this topic as a thesis.

1.1 Motivation and challenges

The motivation and the main challenge behind this thesis is to combine two different areas of technology, machine learning, and digital signal processing to provide a more efficient solution for the digital pre-distortion.

The goal is to replace the conventional approaches of digital pre-distortion with a new one, using the self-organizing maps to reduce the required processing power by reducing the number of computations on the forward path of the digital pre-distortion scheme. Self-organizing maps can identify a correlation between the data that can be used in a combination with another algorithm, to form a power-efficient system.

The integration of machine learning into that task is a great challenge since it will introduce a new approach for digital pre-distortion, not based on PA modelling but on patterns. With the self-organizing maps, where the input signals are being evaluated and clustered according to their similar features, the pre-distorted signal will be produced based on the formed patterns and predictions.

1.2 Power amplifiers

This section is an introduction to power amplifiers. The main operation, the most important terms and the basic characteristics of a power amplifier will be discussed.

1.2.1 Basic concept

The use of the power amplifiers is very broad. They are used for devices such as loudspeakers, headphones and RF transmitters. They are used in order to amplify input signals to a desired level, according to the needs of the project. In contrast to voltage/current amplifiers, power amplifiers are implemented and used as the last block within the amplification chain, to drive loads.

The inputs of the power amplifier have to be over a certain value and for this reason, the pre-amplification operation with the use of current/voltage amplifiers is necessary before the values are modified and sent as inputs to the power amplifier [1].

An ideal amplifier can be characterized by parameters such as the gain and the linearity of the incoming signal, as well as the power efficiency. Figure 1.1 illustrates an ideal power amplifier and (1.1) provides the output voltage for this component.

$$V_{out}(t) = G * V_{in}(t) \tag{1.1}$$

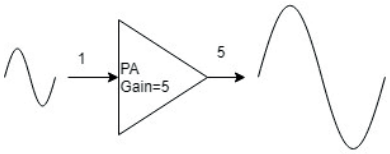

Figure 1.1: PA behavior.

Those characteristics in the real-world applications are always a tradeoff. For example, in order to obtain high output power the amplifier is becoming non-linear as it enters in a compression zone where the input-output exponential relationship is not constant any more. In addition to that the peaks in the output signal are getting clipped. Figure 1.2 shows the difference in the behavior between an ideal and a real power amplifier.

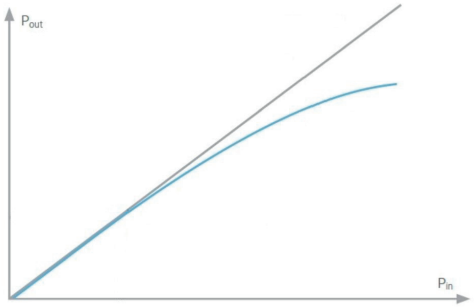

Figure 1.2: Input versus output power for a typical PA.

According to [2] the non-linearity of a power amplifier can cause the presence of intermodulation products, a topic that will be further discussed in the next section. This intermodulation could create a leakage into the adjacent channel of the system and as a result, interference might occur in a different channel. The ratio of the average power between the main channel and an adjacent channel is called Adjacent Channel Power Ratio or ACPR. It is also known as Adjacent Channel Leakage Ratio or ACLR [3].

1.2.2 Memory effects

Apart from the amplitude distortions there are also phase distortions that occur in power amplifiers. Those distortions are indicating the presence of memory effects in the amplifier, which means that the output of the PA does not depend only on the current input but on the previous ones as well [4].

The memory effects can be categorized into two basic types. Electrical memory effects and electrothermal memory effects. For the first group, the origin is from the variable impedances at the DC, fundamental and harmonic band. The impedance can be affected by the transistors in the bias network, that they are creating undesired signals into frequencies that the intermodulation distortion also occurs (the concept of intermodulation distortion will be further analyzed in the next section). This source of memory effects is considered as the most important one, due to the fact that it can force many restrictions on the operation of the system. The second category, electrothermal memory effects, are occurring due to the difference in transistors properties at different temperatures. This effect might create intermodulation distortion as well, however it is not a big issue for low frequencies [5].

1.2.3 Intermodulation

Intermodulation distortion (IMD) could be described as the interaction between two frequencies that are occurring simultaneously in a non-linear device. This interaction will generate unwanted frequencies in the power spectrum. Real life examples of devices where intermodulation of frequencies could be observed are the amplifiers and the mixers.

The mixed frequencies will generate intermodulation effects at the points of the sum and/or the difference of the integer multiples of the original frequencies. Assuming m and n as integers, the output frequency for the mixed input signals can be expressed from equation (1.2).

$$mf_1 \pm nf_2 \tag{1.2}$$

In most cases the IMD is being filtered out, however there are cases where the third order IMD is very close to the frequency of the original signal and therefore it is difficult to filter it. A subset of the third order indermodulation can be represented by equations (1.3) and (1.4).

$$2f_1 - 2f_2 \tag{1.3}$$

$$2f_2 - f_1 \tag{1.4}$$

Figure 1.3 illustrates an example where the first, the second and the third order intermodulation products are illustrated. As it is shown, the third order products are those who are the closest to the main channel.

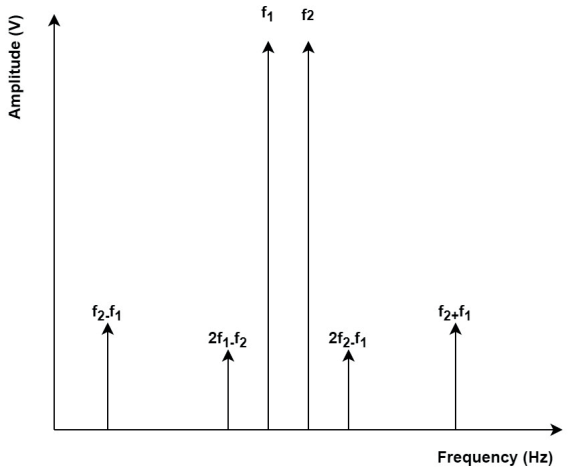

Figure 1.3: Intermodulation distortion graphical representation.

Some typical real-life examples where there is a high demand for good intermodulation performance are the satelites, where each transponder has a limited usable bandwidth and many signals are frequency multiplicated onto one carrier, so there is a high chance of signals interfering with each other. For that reason it is essential to have a reliable filtering of the intermodulation distortion. The same concept applies for the radars, where it is easy to understand that intermodulation is a very sensitive topic, due to the fact the frequency spectrum of one radar, can be easily polluted by other radars that might interfere. More about the importance of a good IMD performance can be found in the literature [6].

1.2.4 AM/AM AM/PM Distortions

Amplitude to amplitude (AM/AM) and amplitude to phase (AM/PM) are two characteristics that are very useful, in order to describe the non-linear effects of an amplifier. AM/AM is the deviation from the gain as the input increases and it reaches the compression region. AM/PM illustrates the deviation of the signal's phase for the same scenario.

As Figure 1.2 shows, there is a compression in the input power after a certain point and that is where the AM/AM distortion occurs. Moreover, in

that region the phase deviation starts to increase [7]. In order to compensate those, a method called digital pre-distortion is used, which will be further discussed in the next chapters.

Figure 1.4 illustrates the AM/AM and AM/PM plots for the power amplifier's data that were used in this project. However, from the provided data only the starting point of the compression region is visible.

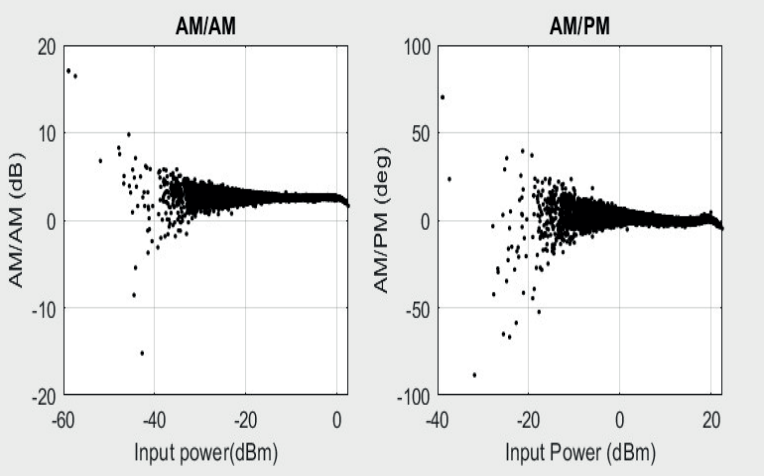

Figure 1.4: AM/AM and AM/PM plots.

1.3 Machine learning

Machine learning as a term refers to the computer's, or machine's, ability to act without having been explicitly programmed for that. It is considered as a very broad area in our days and it is used in many daily life applications such as speech recognition, movies and music recommendation, auto filling sentences for email writing and web search. The digital world is so influenced by machine learning, that users are taking advantage of it in many of their daily tasks without even knowing about it. The main task of a machine learning algorithm, is to solve very specific problems. However, each algorithm is different depending to the problem that has to be solved.

There are three main categories that the algorithms can be classified as according to their ability to learn from patterns:

- Supervised learning, which includes regression and classification.
- Unsupervised learning, which includes clustering and dimensionality reduction.
- Reinforcement learning, which includes rewards and recommendations.

Each method has its own trade-off and according to the design needs, the developer has to select the most appropriate one.

Reinforcement learning is very popular for artificial intelligence in the video gaming industry, as well as for the navigation of robots. Supervised learning is used more for predictions. For example weather forecasting or prediction and analysis of stock market, as well as data classification, which is a strong tool in image processing. Unsupervised learning is a very effective approach for feature extraction, visualization of big data sets and effective clustering of data.

In this project a combination of unsupervised and supervised learning is used, so those two concepts will be further explained in this section.

1.3.1 Types of learning

Supervised learning

Talking about supervised learning the most important characteristic that can define this type of learning is the presence of annotated training data, also known as labels. Supervised algorithms are using known datasets with known labels and they perform predictions on new datasets with undefined labels. In other words, it is the process of teaching a model by providing a combination of input and output data.

There are two categories of supervised learning algorithms:

- Classification, where the labels are classes, so the system is trained to provide specific classes to a new dataset. A typical example of a classifier is an algorithm that determinates if an email is spam or not.

- Regression, where the system is trained to predict missing values for a new dataset. In case of regression the algorithm predicts continuous values such as sales, votes or test scores.

It is worth to be mentioned, that supervised learning will often require human interaction, in order to have a proper training dataset, but once this is done and it is accurate enough then supervised learning algorithms can become a very powerful tool. There are however many cases where another unsupervised algorithm can be used and provide data to the supervised algorithm. This is also the case for this project, where the supervised algorithm is getting the training set labels from the unsupervised algorithm. More information regarding the supervised learning can be found in [8].

Unsupervised learning

The majority of machine learning applications, so far, are being implemented by supervised learning. However, there are certain applications where the use of supervised learning is not possible, since there is not enough information about the expected output. In those cases, unsupervised learning could be a more useful approach.

With the use of unsupervised learning the input features are required but there is no need for labels. Instead, the similarities in the inputs can be observed and groups (classes) can be formed according to the input features. In addition to this, unsupervised learning can reduce the dimensionality of the data in many cases. After the training, the output data (classes) in combination with the input data are available for further supervised analysis. In areas such as image and video processing, the data are presented in a very high dimensionality. For those cases unsupervised learning can be proven as an important solution, since the application of the supervised learning might lead to an overfitting of data and as a result to a poor performance. There are no general rules about the efficiency of unsupervised learning algorithms, so their performance is highly dependent on individual cases [9].

An example where the unsupervised learning could be applied is the customers segmentation, where companies using the customer's individual purchasing history as data, can cluster, identify and analyze certain patterns. That way, the can improve their service and satisfy their clients more.

1.3.2 Artificial neural networks

In many daily life applications, computers have proven themselves better than humans. Typical examples are the calculations and the processing of data. However, the human brain has other big advantages such as the creativity, the common sense, and the imagination. Nowadays, there is a trend towards making computers behave more similar to humans. That is where the artificial neural networks come into the game.

Even though we are still not fully aware about how the human brain works, there is some useful information about the neuron structure. The process begins by the neuron collecting signals through a structure named dendrites and storing them into the core of the neuron. In the next step the core transmits spikes through a channel called axon. This information can be split into thousands of branches before they reach to a place called synapse, which converts the axon activity into information for the next neuron. The learning is happening according to the influence of one neuron to another when it changes. Figure 1.5 represents both a brain neuron and an artificial neuron.

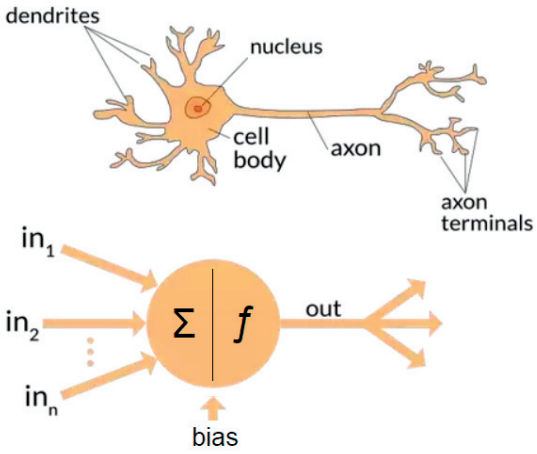

Figure 1.5: A biological and an artificial neuron [10].

A neural network is composed of multiple layers and a layer in a neural network consists of a several neurons. The inspiration of artificial neural networks came from the biological neural networks, so as a result their structure is quite similar. Artificial neural networks are used in many unsupervised machine learning algorithms and they are considered as systems able to perform tasks by learning through examples, without being programmed for a specific task. A typical example is the image recognition where they are able to recognize specific objects without prior knowledge of them or their characteristics. They are achieving that just by training and identification of characteristics. The main task of a neural network is to use the given inputs in order to create meaningful outputs.

The structure of an artificial neural network contains multiple nodes. The neurons receive data as inputs and they perform operations on them. They are fully connected and they are interacting, passing information from one to another. Each node's output is called activation or node value.

The neurons are connected by links, which are the same as axons in the brain neurons. For each link there is a corresponding weight. The process of learning is being implemented by changing and adjusting the weight values. Figure 1.6 shows an artificial neural network with a hidden layer between the

input and the output. The task of the hidden layer is to translate the inputs in a way that the output can use. It performs the computations on the weighted inputs and provides a net, which produces an output after the application of a function on it. Depending on the application, the neural network type and the function are chosen.

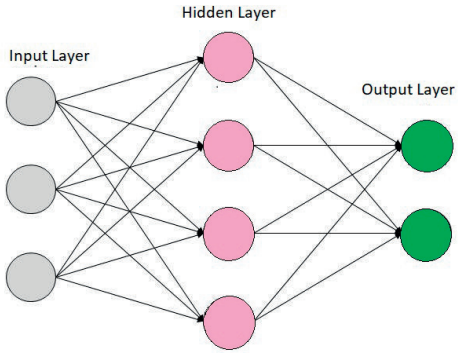

Figure 1.6: Artificial neural network architecture, with three input nodes, one hidden layer, and two output nodes.

1.3.3 The application of machine learning

Machine learning can be found in the background of many applications today. It is widely used for assisting driving or self-driving cars, audio quality improvement, big data analysis, and a variety of other applications.

- A popular machine learning area of application is medicine, where it is used in order to recognize patterns of diseases according to their features. There is a speculation that in the future it would be even possible to replace the doctors, when the algorithms reach the stage where they can recognize and distinguish all the different patterns from a simple cough and a flu, to brain scanning and malignant tumor prediction [11].
- It is also used in the manufacturing industry where it has successfully been integrated in the automated assembly lines. For this there is a study [12], where a fuzzy fault detection system is being developed to detect fault diagnosis of car assembly process. The semiconductor industry has used machine learning algorithms to overcome problems. A very interesting example is from Gardner and Bieker research [13], where with the use of self-organizing maps, they are detecting faults on the wafer fabrication equipment [14].
- Finally one very innovative application of ML is in the design rule check or also known as DRC, a very important step in the physical design of integrated circuits in order to ensure manufacturability. This study is

based on the following papers [15], [16] and [17], where a variety of features that can lead to DRC violations are identified. During the placement and the global routing those features are extracted and feed to the neural network as training inputs, in combination with the actual DRC result. After some certain number of training and evaluation steps the model is ready to be used and predict design rule violation hotspots [18].

Nowadays, machine learning is getting more popular since the available hardware resources, that are needed for training machine learning models, are more powerful. In addition to that new and more efficient machine learning algorithms are introduced continuously. This is only the start towards a new generation where the use of machine learning will offer efficient and low power consuming electronic devices.

1.4 Thesis structure

The thesis is organized into the following chapters:

- Chapter 1: The background and the basic concepts and terminology of the thesis.
- Chapter 2: A theoretical analysis of digital pre-distortion and all the related concepts, with an introduction to the conventional digital pre-distortion approaches.
- Chapter 3: The self-organizing maps are presented along with useful information that will help the reader to understand their main functionalities.
- Chapter 4: The implementation steps and the recorded results for this project are presented.
- Chapter 5: The main conclusion points of the project in along with suggestions about future work.

Chapter 2

Digital pre-distortion

As mentioned in the previous chapter, power amplifiers are more efficient in terms of performance, when they are operating in their compression (non-linear) region. However, in that region the peaks are getting clipped and as a result the output frequency spectrum is ruined. To avoid this, the clipping could be estimated from before, in order to form the inverse model and compensate the amplifier's loss, in terms of gain. When this concept is applied to systems with high bandwidths, the memory effects must be considered, otherwise the performance could be affected due to the fact that when the bandwidth increases the memory depth of the power amplifier also increases [19].

Digital pre-distortion is considered as the most efficient technique of linearization for power amplifiers, in terms of good linearization performance and low implementation complexity. In the following sections of this chapter, the concepts and the terminology of the digital pre-distortion will be further discussed.

2.1 Digital pre-distortion overview

In order to apply digital pre-distortion to a power amplifier, a good first step would be to extract its behaviour. This can be achieved by observing the input and the output data of the PA. The next step is the observation and the estimation of the memory effects through the amplitude-to-amplitude modulation (AM/AM) and the amplitude-to-phase modulation (AM/PM) plots. When the effects are estimated then the inverse equivalent for the input signal should be constructed in order to remove the estimated distortions.

Figure 2.1 illustrates the pre-distortion process. The goal of the DPD as it was mentioned earlier in this chapter, is to compensate the gain losses of the PA's input signal, on high frequencies. So as it is shown in Figure 2.1, the inverse function should be placed before the PA in the chain [20].

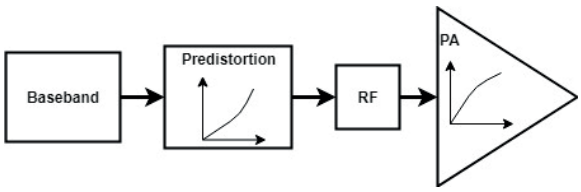

Figure 2.1: pre-distortion process.

The results of the DPD can be evaluated through the ACPR, EVM and some other parameters that will be further explained in section 2.2.1.

A good example to look deeper into the digital pre-distortion is the use of a power amplifier for a transmitter. Figure 2.2 represents the digital pre-distortion system block diagram for a transmitter.

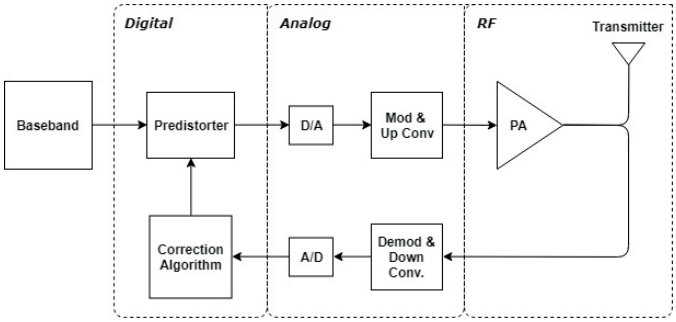

Figure 2.2: Digital pre-distortion system block diagram.

The system starts with the baseband station, which feeds the input data to the pre-distorter, so then they can enter into the analog part where the up and down conversions as well as the D/A and A/D blocks are located. After the digital signal is converted into analog, it enters the PA before it goes back to the digital part through a feedback loop, known as the adaptation loop, where it will enter the correction algorithm. The correction algorithm will perform some calculations, that will be further explained later in this chapter, in order to construct the inverse behavioral model of the PA [22].

2.2 Modeling the amplifier

The behavioral modeling of a PA is basically the process of formulating a mathematical relationship between the input and the output signals, to simplify and describe the behavior of the PA based on previous data. The amplifier in this process is treated as a block box while only the input and output data are under consideration [23].

An accurate behavioral modeling is very important for an efficient pre-distortion and it is usually taking place before the DPD block. As mentioned

earlier, the concept of the digital pre-distortion is to construct an inverse behavioral model of the PA and for that reason, there is a need for a good and simple behavioral model of the PA.

The behavioural models for power amplifiers are divided into the two following categories.

Memoryless models

This model considers only the current input data and ignores any past ones, so the output depends only on the current input. The distortion is usually separated into two different distortions, the AM/AM and the AM/PM distortions. The AM/AM is modelling the PAs saturated output signal and the AM/PM the phase shift of the signal.

The relationship between the input and the output can be expressed as

$$z_{bp}(n) = F * x_{bp}(n) \tag{2.1}$$

where x_{bp} corresponds to the bandpass input, z_{bp} to the bandpass output, F to the function of nonlinearity and n to time-domain sampling index.

Some popular memoryless models are:

- **The Saleh model**
- **The Ghorbani model**
- **The Rapp model**
- **Bessel-Fourier**

Those four memoryless models are providing different accuracy for a behavioural model, depending on the PAs operating region. More information about them can be found in the literature [24].

Memory-based models

In contrast with memoryless models, models with memory are considering not only the current input data but also the previous ones, for a specific range. This means that the current and the past inputs will effect the output and by this condition the system will automatically be converted into a dynamic system.

The memory models are commonly used for systems with wide signal bandwidths, up to 100MHz, without introducing memory effects. However, high-power amplifiers in wireless base stations use wider bandwidths. For those systems memoryless pre-distortion can achieve very limited results of linearization and for this reason memory-based models are preferable.

The most common algorithms for models with memory are the following :

- **Volterra series**

- **Wiener, Hammerstein and Wiener-Hammerstein**
- **Memory polynomial**
- **Generalized memory polynomial**
- **Neural networks**
- **LUT pre-distorter**

Volterra series, Wiener, Hammerstein and Wiener-Hammerstein are considered as some of the most popular pre-distortion algorithms for models with memory. However, those methods are introducing a big number of coefficients which could be very complex for practical applications. For this reason, there are other methods, such as the Memory polynomial and the Generalized memory polynomial that are based on the Volterra series but are less complicated [17]. In addition to those, the LUT pre-distorter according to [25] can be proved as a low cost, efficient, and flexible method for systems with low degree of non-linearity and memory depth. As those two parameters are increasing, the size of the LUTs will increase as well, which will affect the design's area. At last, the neural network approach is very different than the others. In simple terms, it can be described as a network that tries to behave as the human brain and learn from past knowledge and specific patterns. When it acquires the required knowledge it can predict the desired values, in the case of digital pre-distorition, the pre-distorter's output values.

In this chapter the main focus will be on the LUT pre-distorter, due to the fact that the initial measurements and the comparison with the machine learning approach were made with that one. The suggested implementation tries to integrate the LUT pre-distorter into the neural network approach and will be explained in Chapter 4. This technique is based on the the memory polynomial model which will be explained in this section. However, in order to understand the memory polynomial model, it is necessary first to understand the Volterra series, since the memory polynomial is a special and simplified case of Volterra series.

2.2.1 Volterra series

Volterra series is considered as a series of functionals. In other words, a series of operations which are assigning functions to specific sets. It is a way of modeling nonlinear systems with memory, so it is a good tool to model the behavior of a power amplifier with memory by describing its input-output relationship. However, Volterra series shows a very high complexity in terms of computations and as a result it is considered as an impractical solution for real applications such as hardware designing. The number of computations for Volterra series is high because of the large number of parameters, a number that increases exponentially along with the degree of non-linearity and the depth of the memory that the system has [21].

Volterra series can be expressed with the following equation for the discrete time domain:

$$y(n) = \sum_{p=1}^{P} \sum_{i_i=0}^{M} ... \sum_{i_p=0}^{M} h_p(i_i, .., i_p) \prod_{j=1}^{p} x(n - i_j) \tag{2.2}$$

where h_p is the kernel of Volterra series, x the input of the system and y the output. P is degree of nonlinearity and M the memory depth.

Volterra series kernels are the coefficients that are defining the target model. When the coefficients are estimated then the system is ready to be modeled for any input value.

The coefficients are being estimated with the use of the least square approach and the target is the minimization of the sum squared error between the observed data and the output of the PA model. This process is described by the following equation:

$$J(\theta) = \sum_{n=0}^{N-1} e^2(n) = \sum_{n=0}^{N-1} \mid y(n) - y'(n) \mid^2 \tag{2.3}$$

where N is the number of the input samples, $y(n)$ are the samples of the predistorter's observed output data and $y'(n)$ are the PAs output data [26].

Those coefficients can be expressed as:

$$h_p = (X^H X)^{-1} X^H y = X^+ y \tag{2.4}$$

where $(.)^h$ denotes the Hermitian transpose and $(.)^+$ the Moore-Penrose pseudo-inverse. X is the Volterra terms of the input signal x, and y is the captured output signal. With the use of this equation and the input, output signals the coefficient can be estimated and the modeling of the PA can be achieved.

Figure 2.3 illustrates the structure of the Volterra series. The delayed input values are going through the kernels before they are summed up to form the Volterra series output, which is provided by (2.2).

There were many tries over the years to simplify the Volterra series in order to reduce the amount of the required computations. One of the most popular simplified Volterra series versions is the memory polynomial model which will examined in this project, since it is the model that the conventional techniques are based on.

2.2.2 Memory polynomial model

The memory polynomial pre-distortion model, is a Volterra Series-based model but in a simpler form with less terms. Due to the fact that there is a limitation on the resources and the estimation time, the Volterra series is truncated so it can model the memory effects and the nonlinearities of the power amplifier with the use of less elements. The memory polynomial is considered as one of

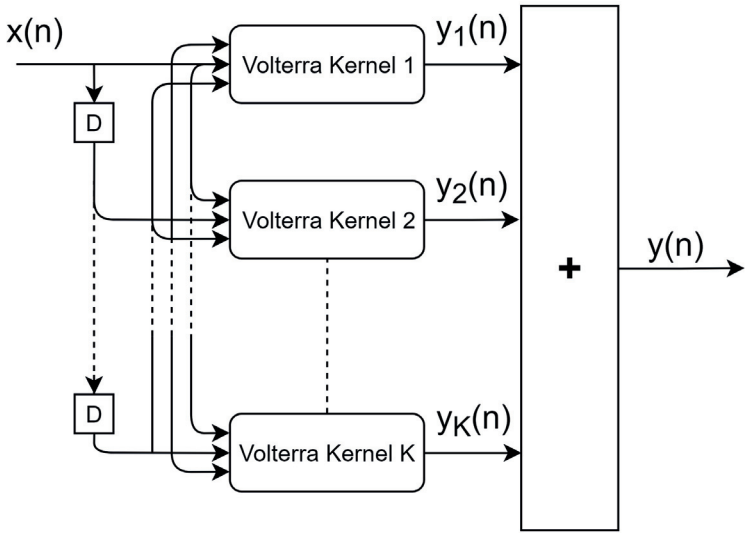

Figure 2.3: Volterra series structure.

the most effective truncated versions and it is expressed with the use of the following formula:

$$y(n) = \sum_{p=1}^{P} \sum_{m=0}^{M} c_{p,m} x(n-m) \mid x(n-m) \mid^{p-1} \tag{2.5}$$

where x is the input signal, y the output, c the coefficients and P and M the degree of nonlinearity and the memory depth respectively [27].

2.3 Learning architectures

There are two different architectures that are used for pre-disorters with memory structures. The first one receives the power amplifier's output and inverses it directly. This method is called direct learning architecture (DLA). The second one constructs a pre-distorter with the use of a postdistorter and it is called indirect learning architecture (IDLA).

Direct Learning Architecture (DLA)

Direct learning architecture (DLA) is a popular technique that is commonly used to identify the parameters, also known as coefficients, of a pre-distorter and is illustrated in Figure 2.4.

The direct learning architecture involves two basic steps. The first one is the identification of the PAs model. Once the model is obtained, there is an estimation of the pre-distorter's coefficients, through an iterative algorithm,

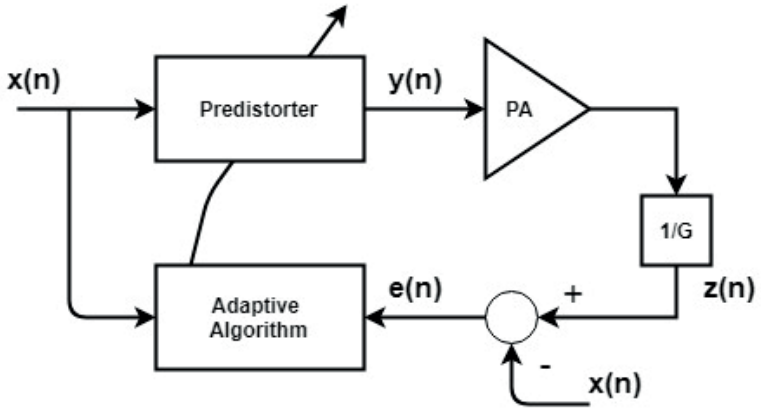

Figure 2.4: Direct Learning Architecture.

that tries to minimize the error between the desired and the actual output, see equation (2.6). The goal is to drive the output $z(n)$ as close as possible to $x(n)$. For that reason the learning controller e is used as a mean of measuring their difference.

$$e(n) = x(n) - z(n) \tag{2.6}$$

After the extraction of the coefficients the next step is to use them in order to construct a pre-distorted signal that will be applied to the PA. This process takes place in the pre-distorter block and it is an iterative process that continues until the best possible solution has been identified.

There are different algorithms that can be used for direct learning architecture, however this architecture is not very popular due to its structural complexity and the big amount of computations that requires [25].

Indirect Learning Architecture (ILA)

The indirect learning architecture suggests the use of an inverting function in front of the PA in the block chain, so an inverse output can be formed. This inverse function is implemented in the postdistorter block. The PAs output is now operating as an input and the PAs input as an output (Figure 2.5). By comparing those two signals, the coefficients are extracted. The values of the coefficients are the actual coefficients of the inverse PA behavior, so after this estimation the inverse model can be generated and placed before the PA. By this iterative process an accurate postdistorter block is developed. This postdistorter block is able to operate and used for any signal within the bandwidth that the used PA has.

Figure 2.5 shows the block diagram of a pre-distortion system with an indirect learning architecture.

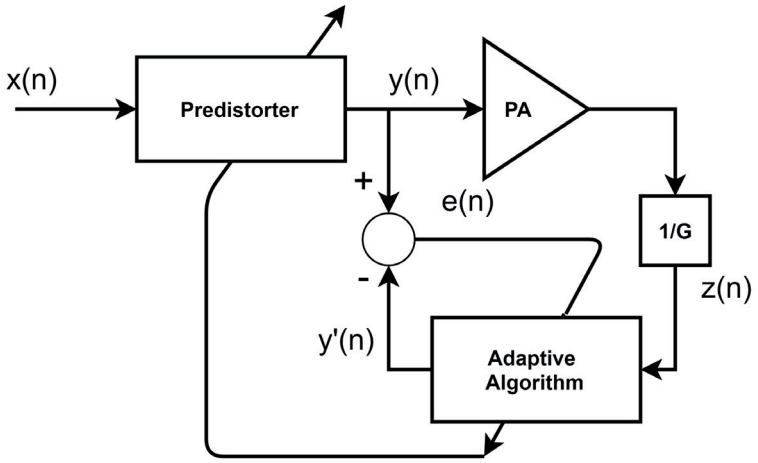

Figure 2.5: Indirect Learning Architecture.

The advantage of this architecture compared to the direct learning, is that it does not require the modelling assumption and the estimation of the PAs coefficients, since the process occurs after the signal leaves the PA and the PAs output is known [20].

In this thesis the indirect learning architecture is used in both the LUT based and the SOMs approach. For the first one it will be explained in more detail in the upcoming sections 2.4.1. For the SOMs approach it will be discussed in Chapter 4 (section 4.4).

2.4 The conventional approach

There are various DPD techniques for linearizing a power amplifier. During this work one of the most common ones was used. This technique uses look up tables(LUTs) to form the pre-distorter and it is based on the the memory polynomial model.

This section aims to provide a basic understanding of the conventional approach and how this differs from SOMs, so the reader will be able to follow the final results.

2.4.1 Look-Up Table (LUT) based pre-distortion scheme

There are mainly two different digital pre-distortion schemes that are widely used. The polynomial scheme and the LUT based scheme. The first one models a pre-distorter as a polynomial function with adaptive coefficients which are linearizing the amplifier. The second one uses also the polynomial model but it calculates the pre-distorted signal by using coefficients obtained from look up tables. Those look up tables contain coefficients as complex values, that

are multiplied with the input signal, according to the input signal's amplitude. The look up table scheme is used in this thesis as a conventional approach.

Figure 2.6 represents a system that uses LUT for implementing the digital pre-distortion.

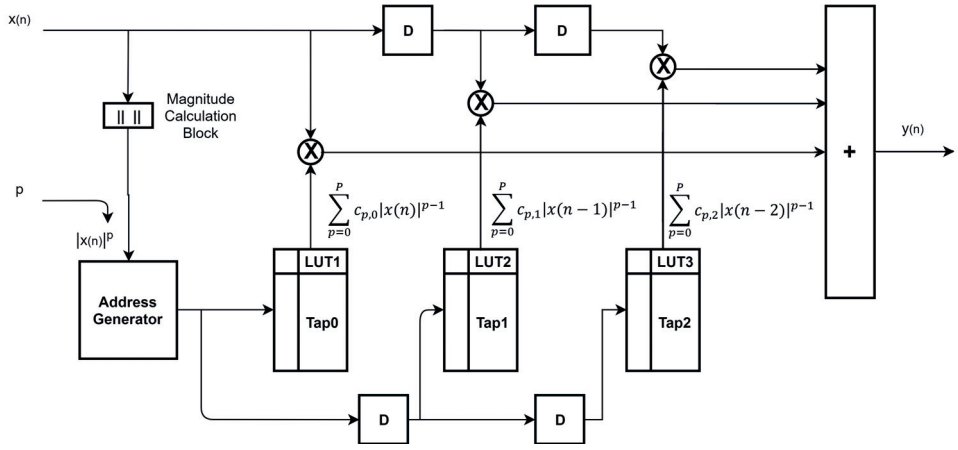

Figure 2.6: DPD with LUT Structure.

The design in this work, contains a LUT table with four columns in total. The first one (address generator), corresponds to an index (address), for the LUTs values, that is estimated through the input's signal magnitude. The index value can be assessed according either the magnitude or the phase of the data. The LUT algorithm in this project uses the magnitude as index.

The rest three tables contain the values of the coefficients for the current input signal (LUT1) and the input signal after one and two delay taps (LUT 2 and LUT3) respectively. When the signal passes the magnitude calculation the non-linearity order p must be assigned, so the proper coefficients from the LUTs will be multiplied with the corresponding input signal x to the power of p(0 to p), forming the following equation for the first LUT:

$$\sum_{p=0}^{P} c_{p,0} \mid x(n) \mid^{p-1} \tag{2.7}$$

which will then be multiplied by the input signal x and form the below equation:

$$\sum_{p=0}^{P} c_{p,0} \mid x(n) \mid^{p-1} x(n) \tag{2.8}$$

which corresponds to the memory polynomial (2.5), just for the first memory tap.

The result of this is just for the input signal (with no memory effects) so the same equation must be formed for the rest of the delay taps. The output y,

which corresponds to the pre-distorter's output and the PA's input, will be the sum of those equations for each memory tap. For example if the result of (2.8) is considered as y_1(n) and the system has three memory taps in total which give y_2(n) and y_3(n) for $x(n-1)$ and $x(n-2)$ respectively, the final y would be the sum of those :

$$y(n) = y_1(n) + y_2(n) + y_3(n) \tag{2.9}$$

The equations (2.9) and (2.10) including M number of memory taps will become as follow to provide the final y:

$$y(n) = \sum_{p=0}^{P} \sum_{m=0}^{M} c_{p,m} \mid x(n-m) \mid^{p-1} x(n-m) \tag{2.10}$$

which is the memory polynomial equation explained in the previous section by (2.5). The output y, result of the multiplication, will be converted into RF and fed into the PA.

In the feedback loop, an adaptive process is taking place, where the values inside the LUTs are updated constantly, through the Volterra Series and the least square approach described previously. This update of the values corresponds to the changes in the PAs behavior over the time. The changes are usually caused by factors such as ageing and temperature. The update process considers the output of the PA, the input of the PA and the index value in order to estimate these changes [28].

2.5 Figures of merit

By definition a figure of merit is the way of characterizing a system's performance in terms of quantity and compare it to its alternatives. In this project the Adjacent Channel Power Ratio (ACPR) is used as a measure of performance and the Error-Vector Magnitude (EVM) as a measure of accuracy.

2.5.1 Adjacent-Channel-Power Ratio (ACPR)

In modern wireless systems the demand for data capacity and bandwidth has been increased significantly, due to the need of delivering IP services to more subscribers. To provide an efficient performance to the transmitter, the PA many times is forced to operate beyond the linear ranges. This nonlinear behavior combined with the signal amplification, might lead to interference that occurs due to the power that leaks from the main channel of the transmitted signal into the adjacent channel. This interference might effect the overall system performance.

Adjacent Channel Power Ratio or also known as ACPR, is a key measurement that it is used in wireless radio systems (3GPP 5G, LTE, W-CDMA).

As Figure 2.7 illustrates, it is defined as the ratio of the modulated signal power in the main channel and the power leaked into the adjacent channel and it is measured in decibels relative to the carrier (dBc) [3].

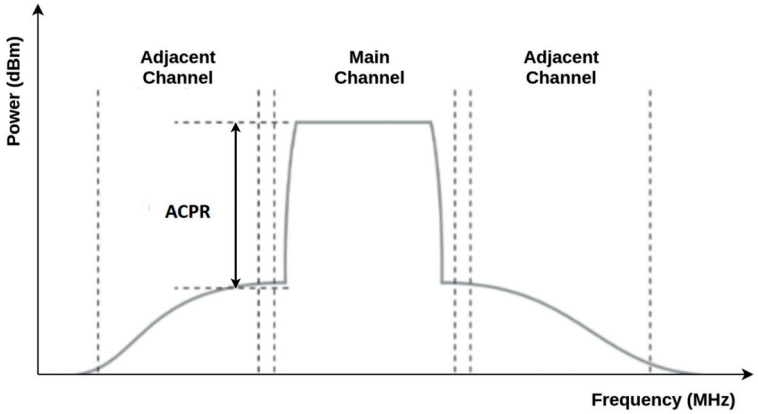

Figure 2.7: Graphical definition of ACPR [3].

2.5.2 Error-Vector Magnitude (EVM)

Error Vector Magnitude or EVM can be described as the measure for the quality of a wireless system's modulation. It has a negative dB value which is desired to be as high as possible.

It is basically illustrating the difference between an ideal vector and the actual measured one. In wireless systems, the EVM can be measured on transmitter's modulator or receiver's demodulator circuits [29].

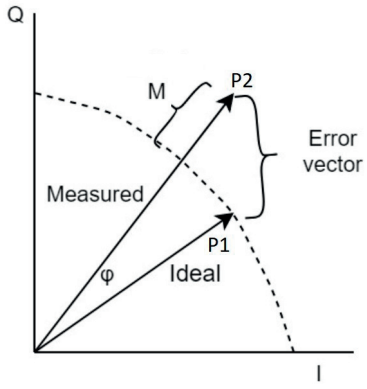

Figure 2.8: EVM measurement.

As Figure 2.8 shows, $P1$ is the ideal constellation point and $P2$ the mea-

sured one. The difference among them might be due to many reasons such as IQ mismatch (gain, phase, DC offset), frequency offset, phase noise, AM-AM distortion, AM-PM distortion etc. The M and ϕ shows the magnitude and the phase error respectively [30]. The EVM can be expressed by the following equation:

$$EVM = \frac{\sqrt{(I2 - I1)^2 + (Q2 - Q1)^2}}{\mid P1 \mid} \tag{2.11}$$

where, $P1 = I1 + j * Q1$ is the ideal vector and $P2 = I2 + j * Q2$ is the measured vector.

The EVM is expressed as:

$$EVM_{db} = 10 * log_{10}[\frac{EVM_{rms}}{100}] \tag{2.12}$$

which provides the result in dB.

EVM can also be measured as a percentage:

$$EVM_{rms} = 100 * 10^{((EVM)_{dB}/20)} \tag{2.13}$$

2.6 Previous work on digital pre-distortion with machine learning

Digital pre-distortion can be implemented in many different ways and provide different results respectively. The use of machine learning algorithms and neural networks has become very popular nowadays, due to their ability of reducing the complexity and the number of computation within an embedded device. This section will examine some of the existing researches that are closely related to the work of this thesis, and their results.

An interesting project is from Manish Sonal (2016). This research explores the modeling of nonlinear analog devices, such as Power Amplifier (PA), with the use of machine learning algorithms. Some of the algorithms that were used and compared in this study are nonlinear regression, smoothing spline, polynomial fit and deep learning with the use of Levenberg–Marquardt algorithm. The results are indicating that the deep learning method provides a very good estimation of the data while it was more time consuming and required more CPU memory [31]. This need however, was a result of the big amount of the neurons and hidden layers, which is not considered as an issue in the use of self-organizing maps, due to their simpler structure.

In contrast with Manish Sonal (2016), Zhenyu Wang et al (2017) proposes a novel method based on deep neural networks, auto-encoder model, for the digital pre-distortion and the results are indicating two things as advantages. The first one is that the deep neural networks are providing a higher accuracy and can fit nonlinear models with success. The second one is that the process speed after each iteration will be faster, due to their deep network structure,

than the traditional neural networks, which basically means that they have the ability to learn and perform faster on practical applications [32].

One good example of digital pre-distortion with the use of a machine learning algorithm, comes from Jun Peng et al. (2016), where the digital pre-distortion is applied with the use of Spare Bayesian learning. What that algorithm does basically is predicting, based on probability, the parameters and the behavioral model of the power amplifier. The results shown that by applying the Spare Bayesian approach the parameters and the sampling where reduced significantly, compared to conventional methods, while the modelling accuracy was very satisfying. This approach is very similar to the SOMs approach but the main difference is that Spare Bayesian is a supervised approach, when self-organizing maps is an unsupervised one [33].

Finally, James Peroulas (2016) research examines the implementation of the pre-distortion with the use of five different machine learning algorithms and how effective they are compared to the memory polynomial approach, which is also used as a conventional algorithm in this thesis work. The five algorithms are linear regression, regularization, model selection, principal component analysis, and gradient descent. Principal component analysis (PCA), which is a method similar to the self-organizing maps, provided the best results by reducing the number of computations significantly and maintained the performance at acceptable levels [34].

Chapter 3

Self-organizing maps

3.1 Introduction

A self-organizing map (SOM), is an artificial neural network (ANN) which applies a competitive learning approach to train samples for data analysis. The most popular self-organizing maps model is known as the Kohonen network and it was introduced in 1982 by the Finnish researcher Teuvo Kohonen. It is considered as a special type of artificial network and it is widely used for clustering and visualizing data [35].

The main principle of self-organizing maps is the transformation of complex data with high-dimensionality, into a simpler form with fewer dimensions (usually two). The location (coordinates) of the output nodes is extracted through the common characteristics of the input space elements. This aspect makes this method similar to other popular dimensionality-reduction techniques, such as the principal component analysis. However, SOMs are proved to have key features that make them more efficient. For example, their implementation is easier, and they are very effective in solving nonlinear problems with a high degree of complexity. In addition to this, they perform very well with noisy or missing data and big datasets, compare to other techniques.

Self-organizing maps are categorized as an unsupervised learning algorithm. As it was mentioned in Chapter 1, unsupervised algorithms do not depend on predefined outputs during their process. In other words, these algorithms learn through observation and draw inferences, rather than depending on input datasets with labels. For self-organizing maps, this happens by applying competitive learning rules to their output nodes, which are competing with each other. Through the learning process, the information is heading only to one direction, without using any feedback loop. This makes SOMs a feedforward network. The input and output nodes are connected through links (weights) [36].

3.2 Structure

As Figure 3.1 shows, a SOM network contains two layers of nodes. The input and the output layer. In contrast with other neural networks, SOM does not involve any hidden layer and the input layer is directly connected to the output one through weights.

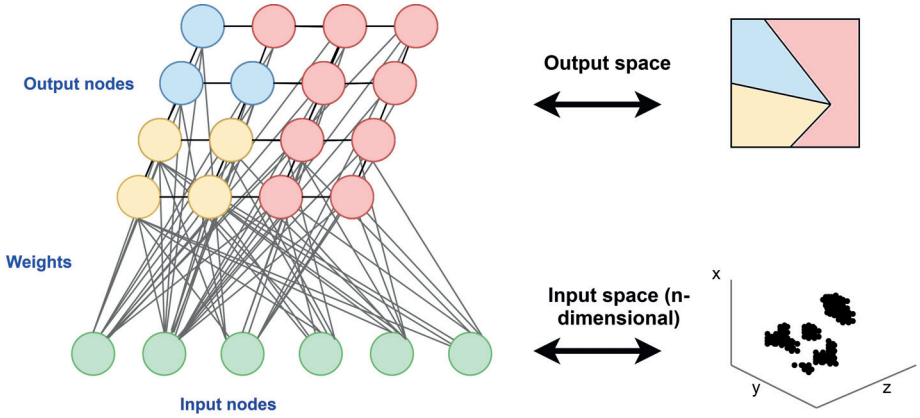

Figure 3.1: Structure of a SOM network.

In the input layer, the nodes are representing the features of the input data. In case there are big differences in the scale of the numbers between the different features, normalization is required to ensure the equal effect of the attributes.

For the output layer, the represented nodes are the visualization of the low-dimensional data. This layer is also called the Kohonen layer and it is usually a two-dimensional layer. The number of nodes is the number of clusters and it is a very important factor in terms of accuracy. The correlation between the neighborhoods is based a lot on the set-up of the nodes.

The typical shape of a SOM network topology is usually a rectangular or a hexagonal grid and each of these two shapes has different properties. With the use of the rectangular topology each node will have four neighbor nodes, while with the hexagonal it will have six [37]. For that particular reason, the hexagonal structure is more popular and is the structure that is used in this project. Figures 3.2 and 3.3 are illustrating the two different topologies.

3.3 Training a self-organizing map

The training process includes several steps to cluster the data. This section will discuss the necessary steps from the moment the data arrive in the system, until the moment that the clusters have been formed.

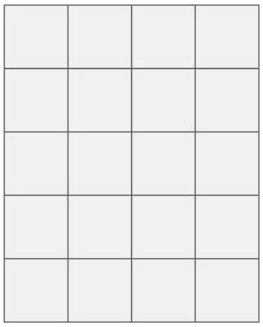

Figure 3.2: Rectangular topology

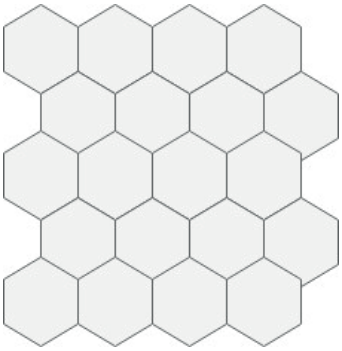

Figure 3.3: Hexagonal topology.

Initial weights

The first step for the SOM is to initialize the weights. The weights appear as links between the input and the output nodes and they keep on updating during the training process. The most common case is to assign relative small and random values to the network's weights, w_i=(w_{i1},w_{i2},...w_{im}), for i=1,2,...,n, where n is the number of output nodes in the network and m the number of the input's vector features. However, there is a restriction that the values should differ and similar weights must be avoided.

A very famous example is the Iris flower dataset, illustrating in Table 3.1. In this example, the dataset consists of many different flowers, with four different characteristics, which are considered as inputs. By comparing those characteristics the algorithm can provide a label for each flower. This label will be the type of the flower.

Considering Table 3.1, a possible weight vector would be w=[0.2 0.6 0.5 0.9]. An array of four elements due to the fact the input vector has four features. Those values are updated during the process but the initialization should have small values (usually 0-0.9).

x	*Sepal length*	*Sepal width*	*Petal length*	*Petal width*
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	7.7	3.8	6.7	2.2
5	7.1	3.0	5.9	2.1

Table 3.1: Iris flower dataset [38]

Sampling

Select randomly input samples from the input space (training data set). For example, considering Table 3.1 and the Iris flower dataset, each flower (or sample x) has four different features. In the sampling step for this scenario sample $x3$, marked as gray, is chosen as a random sample.

Similarity matching

The next step of this process would be to identify the inputs with the most similar characteristics. This step begins with the competition between the output nodes. The neuron who will have a weight vector close to the input's weight vector will be declared as the winning neuron or the best matching unit(BMU). Figure 3.4 illustrates a layer of output nodes where the BMUs for each neighborhood are identified.

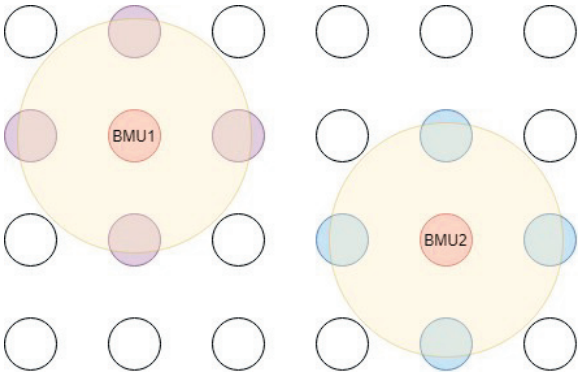

Figure 3.4: Best matching unit.

To find that node, some parameters need to be taken into account such as the distances between the input data(x) and all the weight vectors(w_i). This distance can be calculated with several different methods, Manhattan distance, Chebyshev distance, Euclidean distance, and Mahalanobis distance. Among those methods, the Euclidean distance is the most popular one and the one that is going to be used in this project, due to its ability to provide an isotropic display for the SOM representation. More information regarding the distances in classification can be found in [39].

The following equation illustrates the formula for calculating the Euclidean distance between an input node and the weight vectors that are linked to the node:

$$|| x - w_{im} ||= \sqrt{\sum_{t=0}^{T}[x(t) - w_{i,m}(t)]^2} \tag{3.1}$$

At the end of the similarity matching process and with the use of the Euclidean minimum-distance criteria, the best matching unit c, or else called the winning

neuron, is detected after some iterations with the above equation:

$$c(t) = min(|| \, x - w_{im} \, ||) = min(\sqrt{\sum_{t=0}^{T}[x(t) - w_{i,m}(t)]^2}) \tag{3.2}$$

where T is the maximum number of iterations that the network will perform.

Neighborhood function

The neighborhood function is responsible for providing a relation between a sample and its neighbors, as well as the topological order of the map. In case that this function is missing then the SOMs are the same with the k-means algorithm, which is an algorithm more based on averaging the selected sample according to its neighbor samples. In contrast, SOMs are trying to match the selected sample with its closest neighbor and increase the distance with the rest of the neighbors. The two most popular functions for this purpose are the Gaussian (3.3) and the square function (3.4) :

$$h_g(w_{ij}, w_{mn}, r) = exp(-\frac{(i-n)^2 - (j-m)^2}{2r^2}) \tag{3.3}$$

$$h_s(w_{ij}, w_{mn}, r) = \begin{cases} 1 <= \sqrt{(i-n)^2 + (j-m)^2} \leq r, \\ 0 <= \sqrt{(i-n)^2 + (j-m)^2} > r. \end{cases} \tag{3.4}$$

Where w_{ij} is the selected neuron in position i,j, w_{mn} the neighbor neuron in position m,n and r is the radius. In both functions, the radius r is decreasing to 0 or 1 during the training. The Gaussian neighborhood is considered as a more reliable solution when the square function requires less computations [40].

For this experiment, the Gaussian neighborhood function was used to obtain more accurate outputs.

Weight updating

After identifying the best matching units, their weights and the weights of their neighbor units are adjusted to become correlated to the input space features. The learning rate and the neighborhood size are the two most important characteristics of this weight update.

The learning rate is in charge of the change on the weights and its value can differ from 0 to 1. However, the learning rate for the self-organizing maps is not a stable variable and it is gradually decreasing as the number of iterations of the SOMs algorithm is increasing. This decrement can occur linearly, exponentially or inversely proportional to the iterations. In most of the neural network cases, the weights are assigned randomly and the learning rate starts from a very high value, close to 1 and reduces through the time. At the beginning of the training, more iterations are occurring, and they are reducing as

the process goes on, as there are fewer necessary corrections that need to be done.

The following formula indicates the updated weight vector $w_i(t+1)$ of the winning neuron and all the neurons that lie in the neighborhood of it, with respect to the number of iterations (t), the learning rate (α) and the past weight vector $w_i(t)$:

$$w_i(t+1) = w_i(t) + \alpha(t)h(w_{bmu}, w_i, r)[x(t) - w_i(t)] \tag{3.5}$$

Where h is the neighborhood function, w_{bmu} the weight vector of the best matching unit and r the radius of the formed neighborhood [41].

3.4 K-nearest neighbors

The k-Nearest-Neighbors or KNN is considered as one of the simplest algorithms for classification and regression. The main function of this algorithm is classifying input data into outputs by identifying the most common characteristics on them and applying a new set of data from the same type to perform predictions about their outputs. The same approach applies to regression, where the algorithm predicts target values for new data [42].

In contrast with other algorithms, KNN is not using any training method. Instead of that, the moment the input data are available they are classified and after that, any possible training is applied. However, to perform classification, the algorithm has to go through all the data points. This process makes this algorithm very expensive in terms of computation.

The process will follow the below steps :

- Calculate the distance between the selected input value (a random value from the training dataset) and the rest of the training data inputs.
- Choose k amount of data points, located close to the selected data point. Those points are identified as the ones with the lowest distance values.
- Evaluate according to their distance value and the number of similar data in each class, to which class the selected data point will be classified (or in case of regression what would be the predicted value) for the rest of the data.

There are two factors to be taken under consideration before the KNN process starts. The first one is the value of k, which can just be a random number or a fixed one after tests and optimizations. For this project k was set as 3, since that was the number that provided the most accurate results.

The second one would be the distance metric that the algorithm is going to use. As it was mentioned before for SOMs, there are many different methods for calculating this distance for KNN. Likewise SOMs, the Euclidean distance is the most popular one, which is described in the previous section and the

one that is used in the KNN model of this work. Equation 3.1 will be applied for the case of the KNN as well [43].

To summarize the KNN model includes the following steps:

- The load of the data.
- The initialization of the value for the k factor.
- Multiple iterations for each data in order to obtain the predicted class.

In the last step, the algorithm first calculates the distance between the test and each of the training data. Then according to the calculated distances the k top distances (the ones with the lowest values) will be selected. The most frequent class label among those elements will be obtained and assigned as the predicted class.

Figure 3.5 illustrates the KNN classification map for Iris flower dataset, where there is a representation of how the data are divided into three different flowers. Setosa, Versicolor and Virginica.

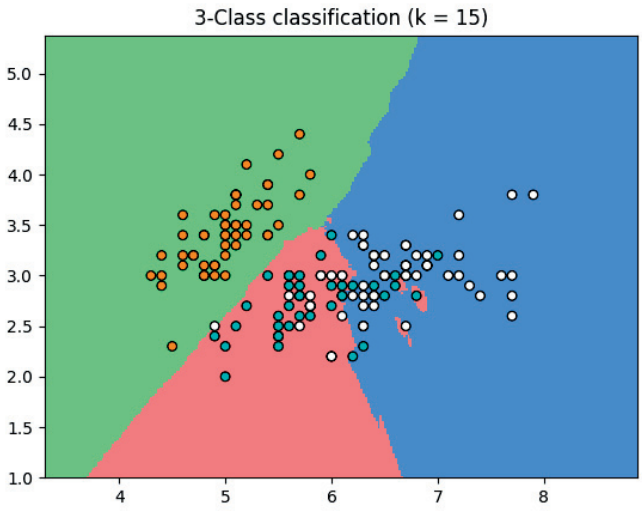

Figure 3.5: KNN classification map for Iris flower dataset [44].

3.5 Applications

As mentioned before, self-organizing maps is a very popular way of visualizing data and observe hidden patterns behind them that are not clear in the beginning. K-nearest-neighbors algorithm, on the other hand, is a very accurate method for predictions. This section will discuss some of the most important applications that the self-organizing maps, as well as the k-nearest-neighbors,

are involved too.

Applications of self-organizing maps

One very interesting research has been published by Ryotaro Kamimura (2012), where SOMs is used as an algorithm for analyzing data for automobile industries in Japan and predict the financial state of the country since researches have proved that those data are related and affect the country's economy [45].

Another important application has been released by Suzanne Angeli et. al (2012). This study is investigating the classification of earth observation data with the use of self-organizing maps. Those data are from sensors on earth that are monitoring the atmosphere and some geophysical properties such as identifying the clouds or the areas that are covered by ice. To receive that information, there is a need for pixel processing for the obtained picture from a satellite. The conventional approach to this is with the use of decision trees, but this research replaces the decision tree solution with the self-organizing maps. The results are indicating that SOM is a very effective algorithm for this application and it is providing a simpler solution to the problem [46].

At last, an interesting approach within the financial sector has been published by Li Jian et. al (2016). Due to the increase of fraudulent financial reporting over the last years, it has been observed that many reports contain overvalued profits, sales, and assets, or understating liabilities and expenses. Those fault reported values can be usually observed through the financial ratios. The financial ratios are mainly divided into two groups, normal and abnormal group and the fraudulent data are mostly located into the abnormal group. As a result, the SOMs are classifying the input data into two individual classes. The results are showing that SOMs are effective in detecting the fraudulent financial data with high accuracy however the results could be further improved by enlarging the dataset or by trying and comparing different unsupervised algorithms [47].

Applications of k-nearest neighbors

KNN algorithm is very popular as a text mining technique with a wide variety of applications. In their paper Suhatati Tjandral et. al (2015), they are examining the use of the KNN as a text mining algorithm where the final design is used as a system by the government to receive complaints from the citizens. With the use of its data mining properties, KNN can detect the department that should receive the complaint text and the mail is being forwarded to them. As was expected the results showed that KNN can be used as a very reliable text mining system [48].

Facial image analysis is considered as a very important part of the image processing area. It can be used for face recognition, psychological tests, and several more applications. Treesa George et. al (2014), in their research, explore the application of the KNN algorithm for the detection of a smile within

still images. With the use of another classification algorithm called Haar cascade, the necessary information to detect a mouth and a pair of eyes on a face are extracted. Then those values are provided as input training data to the KNN algorithm. The results are showing that with the use of the mouth only the accuracy is 60% but including the eye pair as well the result is increasing and the algorithm becomes more accurate [49].

Finally, an interesting research paper is published from S.Venkata Lakshmi and T.Edwin Prabakaran (2014), where with the use of the KNN they are detecting cyber-attacks to a network, by examining some specific features and looking for potential intrusions. The data for train and testing are provided by the KDD Cup dataset, which is considered as the benchmark data in Intrusion detection and contains 41 different features for each data. By changing the number and the type of features within 5 sets, different threats can be identified. Set five appeared to be the most effective since the biggest amount of threats was identified with the use of those features while it had less amount of data compared to other datasets [50].

Combining self-organizing maps with k-nearest neighbors

The two algorithms described before in the chapter, self-organizing maps (SOMs) and k-Nearest-Neighbors (KNN), are combined in this project to provide a final result and replace the conventional approach for the digital pre-distortion.

The self-organizing maps are used as a mean of classification. In other words, this algorithm is used to obtain a class value (label) by identifying the common characteristics among the input data. This, however, is applied only to a part of the input dataset. This dataset is called the training dataset.

The k-Nearest-Neighbors, on the other hand, are used as a mean of regression. Using the training dataset in combination with their class values, the algorithm can perform a class prediction for the rest of the data (called test set) by providing their input values.

Figure 3.6 illustrates a simple representation of the two algorithms combined. The next chapter will explain more in-depth how this concept is applied to the design.

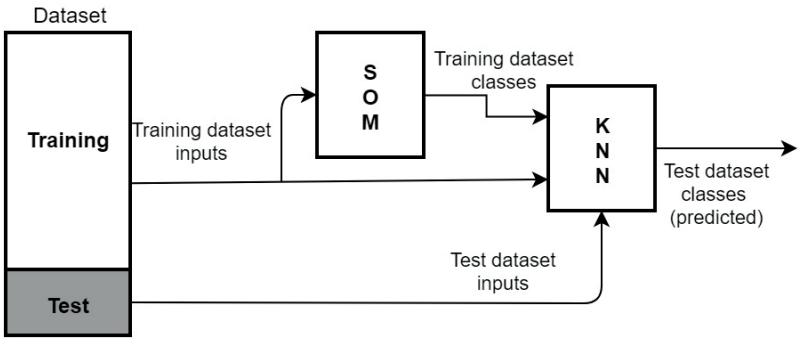

Figure 3.6: SOMs-KNN combination.

Chapter 4

Implementation and results

This chapter will discuss the approach for the neural network based DPD design, the architecture that was used and the given results. The simulations were executed with the use of *MATLAB* and the algorithm was developed with MATLAB's neural network toolbox. Some additional simulations for calculating the computational complexity were executed with the use of Python scripts.

4.1 Specifications

According to the current industry's needs and to the 3GPP specifications, a proper EVM for a base station should be below 8% [51]. However, the PA is only a part of the base station and there are other components in the system such as the modulators and the RF transceivers that are affecting the total EVM. For this reason, EVM just for the PA should be in lower levels, around 4% or even less.

On the other hand, the ACPR of the conventional approach is around -45dBc which is considered as a very good and low ACPR. According to the predicted performance losses, and the 5G NR specifications [52], an ACPR value for a PA with a channel offset of 400MHz, as the one used in this project, would be between -27dBc and -33dBc.

4.2 The concept

As it was shown in Figure 2.6, in the LUT-based model there are many different LUTs according to the system's memory taps. The idea of this project is to prove that all those LUTs could be replaced by just one that will use the input x as an index and predict the y value (instead of (2.5)). Input x refers to the pre-distorter's input and y to the pre-distorter's output (for more details check section 2.2 and 2.4). This implementation can reduce the required amount of

computations since only one LUT will be needed and there will not be any complex multiplications.

In the conventional approach, the input data are complex numbers, consisted of phase and magnitude. In this project's implementation, the phase and the magnitude of each input data point are separated and fed to the SOMs as two different features. Then SOMs will classify those data and provide this information in a form of a LUT to the forward path.

For prediction and to examine if this implementation could work the KNN algorithm is used. KNN will use the input sample (phase and magnitude) and predict its class. According to the class value an individual value taken from the SOMs LUT, will be assigned as a pre-disorter's output. This way of estimating the pre-disorter's output will help reduce the number of combutations in the forward path, since there will no longer be a need for using several equations and multiple LUTs.

Figure 4.1 represents the suggested approach for replacing the LUT method. The address generator will operate as a filter which should provide an index to the LUT according to some specific characteristics of the input x. The KNN has the role of identifying those characteristics, predicting the class of the input as an index and obtaining the corresponding y value. This process will be further explained later in this chapter.

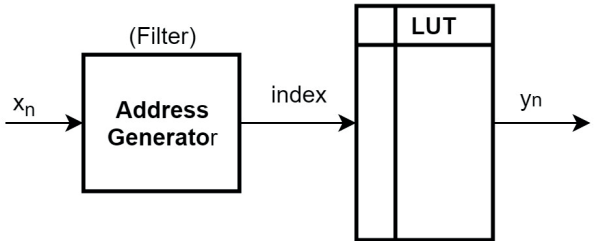

Figure 4.1: LUT model based on prediction.

4.3 Implementation

The main goal of the project is to examine whether SOMs can be used as an adaptive algorithm and provide a good enough LUT which in combination with a predictor (KNN), will replace the conventional architecture. However, the bottleneck is that the input values to the pre-distorter are not directly related to each other, since they are consisting of the input signal (x) and its delayed values. They have to pass through the Volterra kernel as Figure 2.3 shows, so the algorithm can identify their common characteristics. In other words, the Volterra series output value (y) has to be obtained and provided as an input to SOMs. The idea is to use some values of y for training and then predict the rest y values based only on the input x values.

The data set is split into two parts. The first one is the training set, which includes a part of the pre-distorter's inputs. Those values are going to be considered as the input values for the training (i.e the first 40% of the pre-distorter's input values). The second one is the test set, which will contain a part of the rest of the pre-distorter's input values (i.e the last 20% of the input values). This set will be used as input to the KNN and the algorithm will predict its outputs. The performance and accuracy of those results will be measured with the use of ACPR and EVM respectively and the computational complexity by measuring the floating point operations (FLOP).

4.3.1 Training stage

For this project, the two feature input to SOMs will be the phase and the magnitude of y, which is obtained by splitting the real and the imaginary part of this complex value. When the values of y pass into the SOMs, then the neural network will divide them into classes by identifying their common characteristics, as it was explained in Chapter 3.

The next step will be to estimate an average y value for each class and store it into a table. With this step the average y could be used instead of the original one for each input value, resulting in a significant decrease of the different y values. At the end of this process, each class will have a unique y value.

For example for a dataset of 100000 different values of x corresponding to 100000 different values of y, applying this implementation using 1000 classes can reduce the number of y values from 100000 to just 1000 values, a decrease of 99%. Of course, the main drawback of this implementation will be the loss in accuracy and that is a part of what this thesis is investigating.

Figure 4.2 illustrates how the above process is implemented in the system. This part is the training stage where the data are trained in order to provide the system with the required information for the processing stage, where the prediction is taking place.

It is worth to be mentioned that the conventional approach with the use of LUTs (Figure 2.6) will run for several iterations (5 in this project) in order for the coefficients to get updated and provide a more accurate result for y. The coefficients will have random values from 0 to 1 at the beginning of the process and they will be updated after each iteration through the equation (2.4). This process will occur only for the training set.

4.3.2 Processing stage

After each input x of the training dataset has a class and a y (average) value, the following step will be to provide this information to the KNN algorithm in order to predict classes for the test set. Now the test set consists of x values as well, since the initial dataset was split into two separate datasets. The training

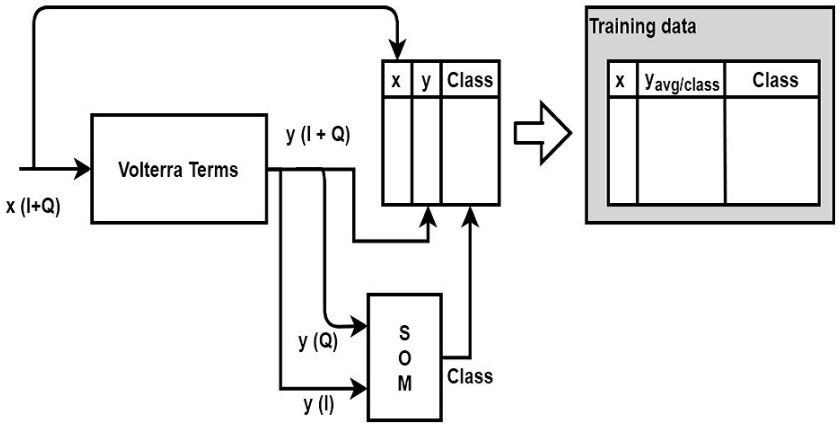

Figure 4.2: Training stage, classification of the input values.

dataset and the test dataset. After the KNN predicts a class for each of the test set value, then the corresponding y for each class will be assigned, as it was estimated in the training set. Of course, it is important that the training stage and the processing stage are using the same amount of classes so it is possible to find an average y value for each class after the KNN predicts a class value. By using fewer classes, for both testing and processing stage, the shrinking of the original amount of values can be achieved but this will lead to a decrease in the system's accuracy.

Figure 4.3 represents how the class prediction for the test data is executed.

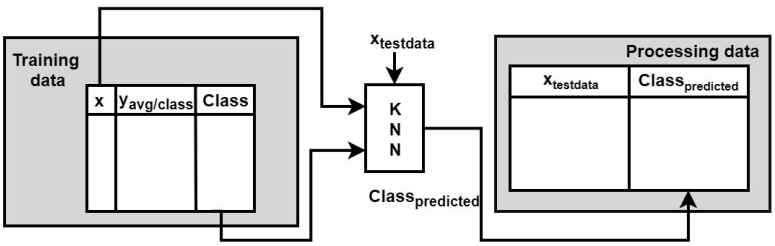

Figure 4.3: Processing stage, prediction of the output.

By providing the input x of the training set and their class values, the KNN can predict a class value for each of the test set values. This is achieved by finding the closest neighbors to the value or in other words the closest matching classes. This closest matching class will be the predicted class.

After the KNN has predicted a class value for each test data x, this class will be used as an index to the LUT (Training data table) to obtain the y (av-

erage) value, which will be used as the final y value for the corresponding x (Final data table). Figure 4.4 represents how this indexing is executed.

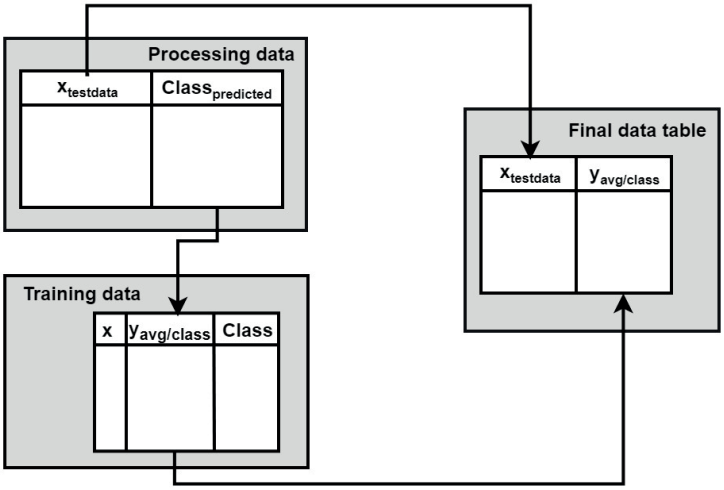

Figure 4.4: Processing stage, prediction of the output.

4.3.3 Results

As was mentioned in Chapter 2, the results should be evaluated in terms of performance and accuracy through the ACPR and the EVM respectively. Measuring the ACPR and the EVM for the training and the processing stage using 50 as the number of classes and 40% of the dataset as training data (around 45000 values), the following values from Table 4.1 resulted.

Stage	ACPR	EVM
Training (SOM)	-20.51	21.10
Processing (KNN)	-19.85	23.25

Table 4.1: Performance

In the training stage, the EVM and ACPR values are estimated using the average y values, which were calculated by averaging the y values in each class (section 4.3.1).

In the processing stage, the EVM and ACPR values are estimated using the predicted y values from the final data table (section 4.3.2).

As it was expected the EVM and the ACPR are increasing after the processing stage since the KNN algorithm is performing a prediction so a loss

in performance and accuracy was expected. However, both the EVM and the ACPR are very high in this case.

A first approach to reduce those values is by increasing the number of classes. This solution will provide a more accurate model with the drawback of increasing the total number of y values and as a result the number of computations. Figures 4.5 and 4.6 show the reduction of the EVM and the ACPR respectively, by increasing the classes.

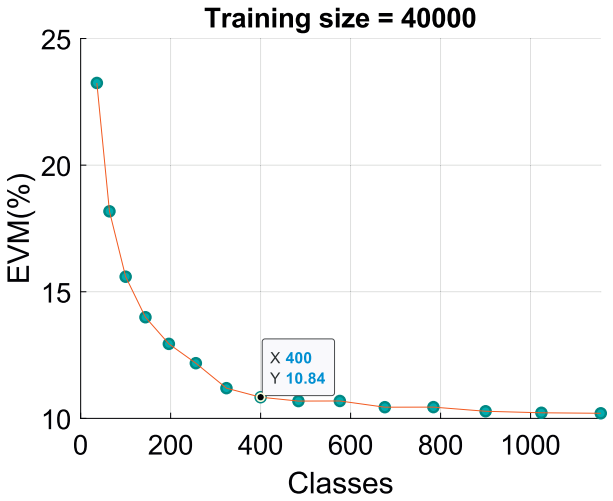

Figure 4.5: EVM-Classes plot

Another important factor is the size of the training set. The bigger the training set is the more accurate the training stage will be. Figures 4.7 and 4.8 are showing how the EVM and the ACPR are decreasing by increasing the size of the training data set.

After extracting that information, it was observed that even by increasing the classes and the training set the EVM and ACPR levels are still very high. For this reason, an optimization loop was added to the training stage of the system.

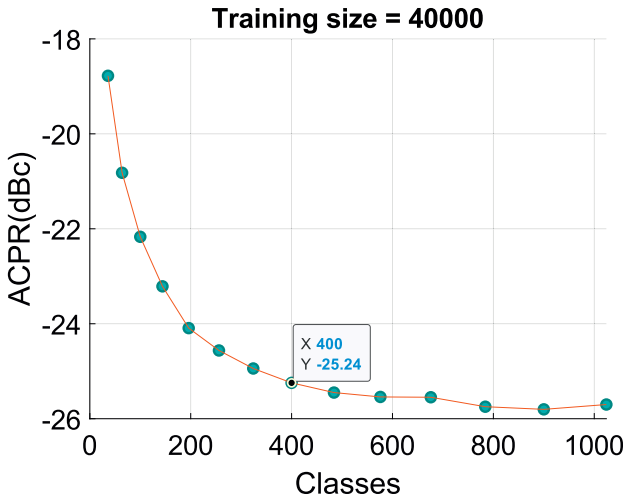

Figure 4.6: ACPR-Classes plot

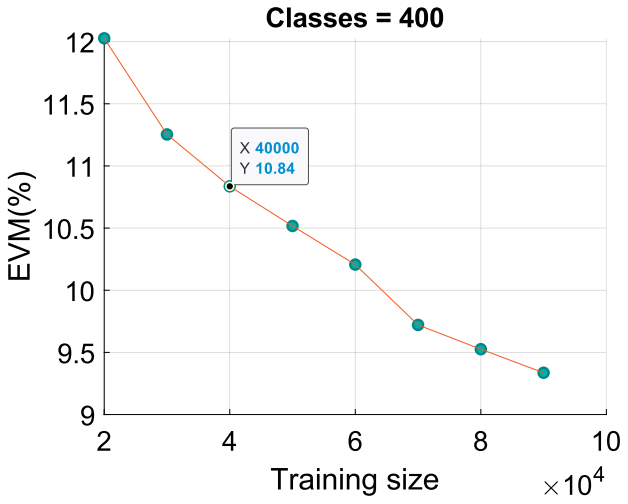

Figure 4.7: EVM-Training size plot

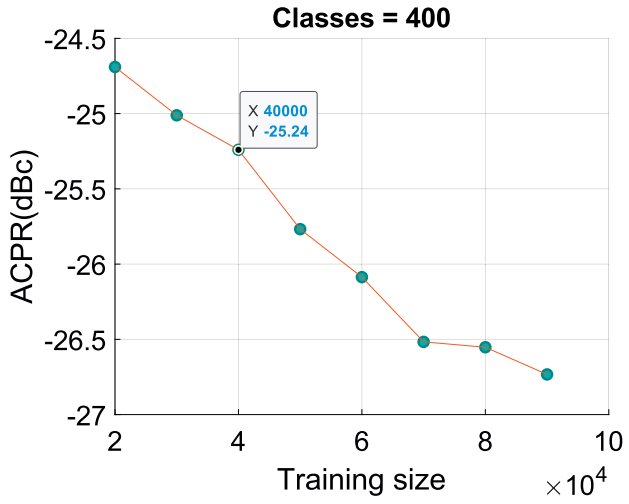

Figure 4.8: ACPR-Training size plot

4.3.4 Optimization loop

By estimating a unique y for each class and limiting the number of different values of y, the EVM was expected to increase since quantization was introduced. In order to decrease this EVM, an efficient solution is to identify the classes with the highest EVM values in the training stage and divide them into more classes. However, to obtain more accurate values and decrease the error as much as possible, this process has to be iterative. For the above reasons an optimization loop will be introduced to the training process of Figure 4.2 and the training stage will now become as Figure 4.9 illustrates.

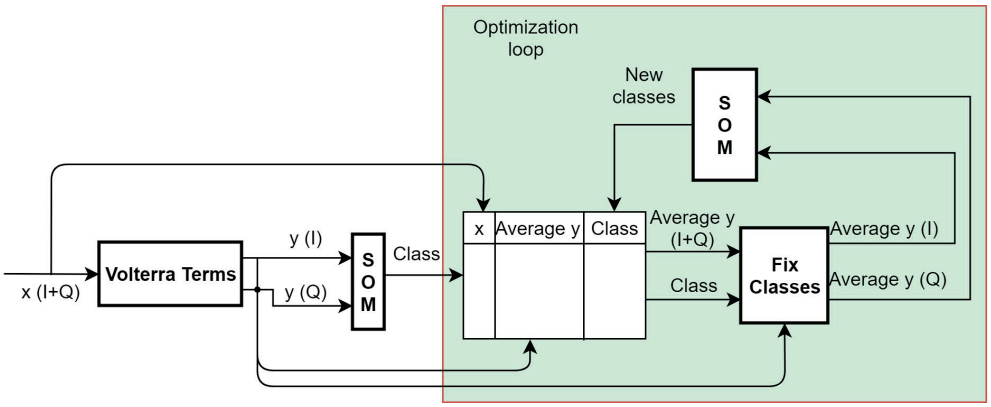

Figure 4.9: Training stage with optimization loop.

The fix classes block compares the assigned y values of each class with the actual ones and calculates the total EVM of each class. The classes are then

sorted in descending order, from the one with the highest EVM to the one with the lowest, so N number of classes can be selected and divided into M new classes.

4.3.5 Results

With respect to the results from Figures 4.4 and 4.5, the number of classes was set to 400. As the training set increases, the EVM and the ACPR are getting better, as Figures 4.7 and 4.8 illustrate, however, 40000 was selected as the size due to the fact that the simulation time is too long and large sets would highly affect this time. To test the optimization loop, the initial number of classes was set to 400, then the 200 classes with the highest EVM are indicated and divided into more classes. The number of new classes per loop is not constant since overlaps might occur between the values of the new classes and the values of the old classes. In that case, only one class is formed every time, so the uniqueness of the value will be maintained.

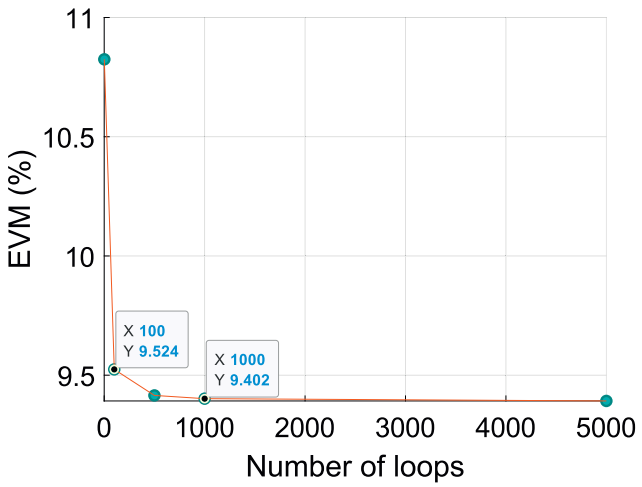

Figure 4.10: EVM-Number of loops plot

The results from Figures 4.10 and 4.11 are indicating, that the use of the optimization loop can decrease the EVM and the ACPR, as it was expected since this approach provides more classes with better accuracy. However, this can drop the EVM only to a certain point. As it is shown from 1 to 100 iterations the levels of EVM and ACPR are decreasing significantly, but on the other hand from 100 to 500, 1000 and 5000 are saturating.

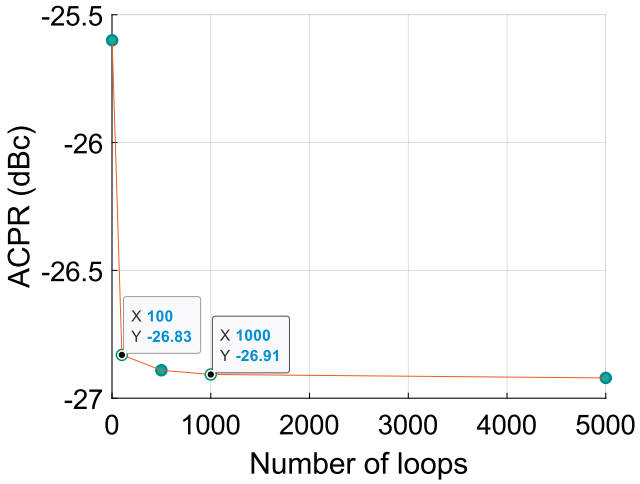

Figure 4.11: ACPR-Number of loops plot

4.4 Architecture

The final system is illustrated in Figure 4.12 and the flow chart of Figure 4.13 explains the steps for the whole process. At first, the training data (x) will pass through the Volterra series, so the Volterra terms (y) will be estimated.

The calculated Volterra outputs will go through the SOMs where they will be classified into N number of classes. The next step would be to identify the K number of classes with the largest EVM and divide them into M number of classes, which would lead to a more accurate model.

The Volterra outputs along with the classes will be used from the KNN as the training data and labels. The rest of the input data set values (x) will be used as the test data for the KNN. The KNN algorithm will predict the class for each x input value of the test set.

According to the predicted class, the average y will be assigned as it was calculated in the training stage.

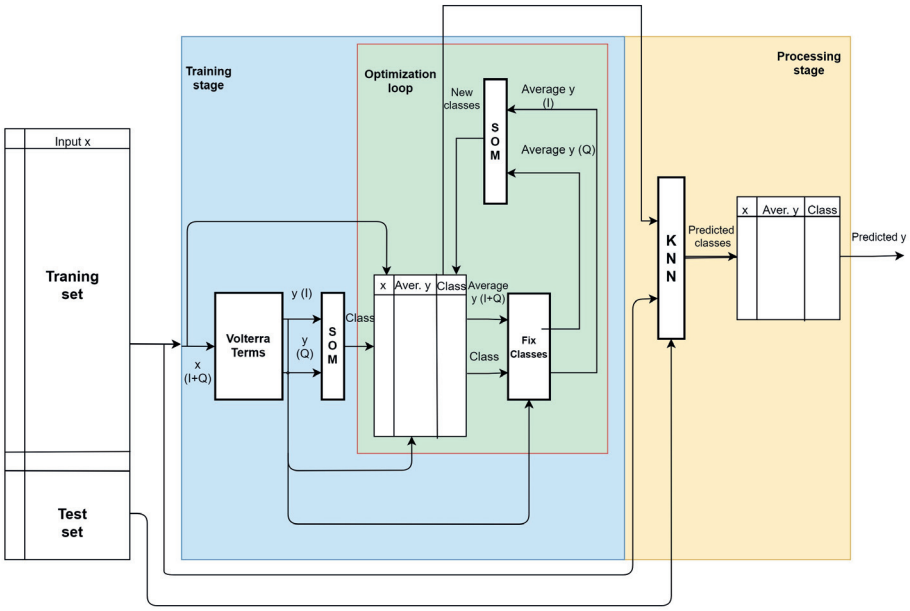

Figure 4.12: Overall system architecture

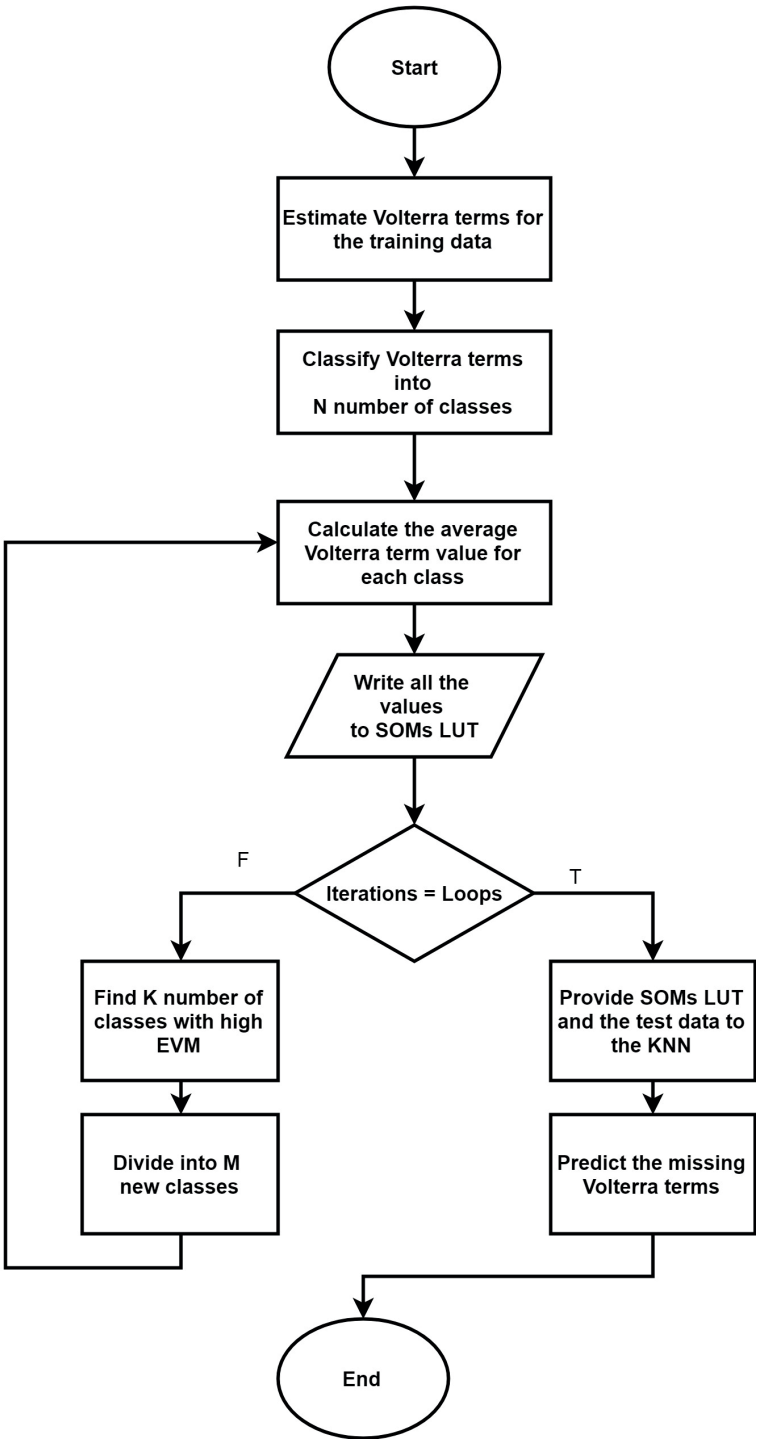

Figure 4.13: Flowchart of the overall process

Figure 4.14 illustrates the proposed architecture for the digital pre-distorter implementation. The Delay Adjustment block has been added to indicate that the system will first accept a certain amount of input samples (40000 in this project) and then will classify them with the use of self-organizing maps, to form a look-up table that will contain the classes and the calculated y value for each class.

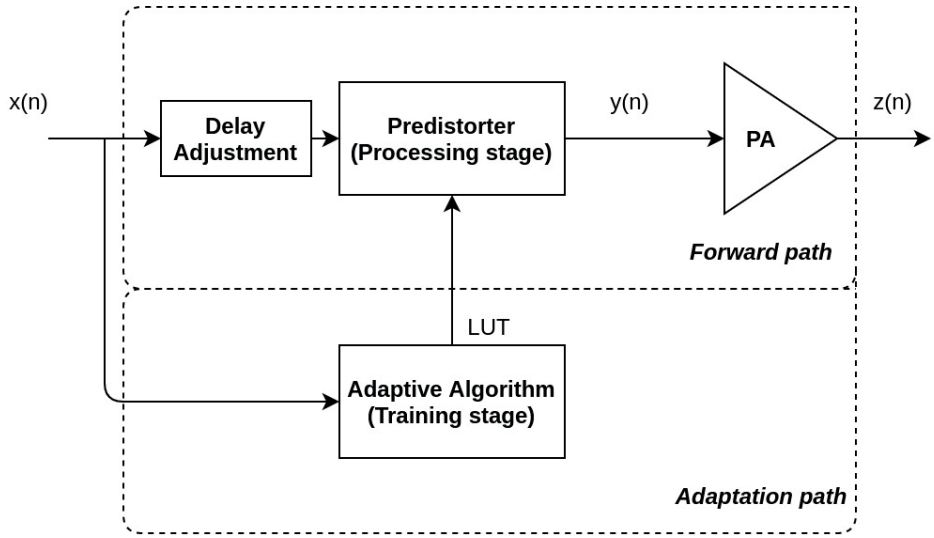

Figure 4.14: Proposed pre-distorter with SOMs and KNN

4.5 Computational complexity

Using *MATLAB* for simulations would not be a useful solution for measuring the computational complexity and comparing the project's implementation with the original one, due to the fact that *MATLAB* optimizes many functions and does not provide a proper interface for measuring the computational complexity. For that reason, SOMs and KNN were also implemented in Python where the total number of floating point operations per second (FLOPS) was estimated with the use of Python's *PAPI* (Performance Application Programming Interface) library [53].

Table 4.2 shows the number of FLOPs for the training part. The numbers are indicating that SOMs require a big number of computations, especially when the number of classes and the training size are increasing.

Classes	Training size	GFLOPs
50	40000	16
200	40000	56
400	40000	100
400	20000	50
400	30000	75
400	40000	100

Table 4.2: SOMs computational complexity

Figure 4.15 illustrates the pie chart for the computational complexity per step for the SOM. The steps that require the biggest amount of computation are the Neighborhood Function and the Similarity Matching, The ones with the lowest are the Sampling and the Weights Initialization which due to their very low percentages, are not really visible on the pie chart. Looking back into Chapter 3 (section 3.3), one can understand that equations (3.1), (3.2), (3.3) and (3.4) which are used in Neighborhood Function and Similarity Matching steps, require more computations compared to the rest of the steps.

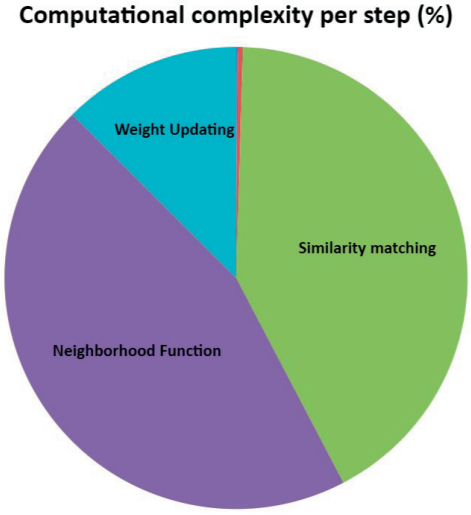

Figure 4.15: Pie chart for SOMs complexity

Classes	Training size	MFLOP/sample	EVM(%)	ACPR(dBc)
50	40000	14.2	18.3	-20.8
200	40000	13.9	12.6	-24.2
400	40000	14.7	10.8	-25.2
400	20000	8.7	12	-24.74
400	30000	11.5	11.25	-25.22
400	40000	14.7	10.8	-25.2

Table 4.3: KNN computational complexity

Table 4.3 illustrates the computational complexity of the KNN pre-distorter. As the results are showing, there is no difference on the amount of computations when the number of classes is increased. However, this is not the case about the training size. As it was expected the training size affects the amount of computations, since the KNN algorithm compares its input sample with all the values of the training set. The number of classes will not affect that comparison since it will always be one feature, but with different range of values depending on the total classes. In contrast, increasing the training size will increase the number of samples that will have to be compared with the KNN's input size and it is easy to understand that this will affect the computations.

4.6 Comparison to the Look-Up Table (LUT) pre-distorter

In the proposed implementation the training part is required to run only once and form the LUT, that will be provided to the KNN. For that reason the KNN pre-distorter is compared in this section with the LUT approach.

According to Figure 4.10 and 4.11, the best results for the KNN pre-distorter were obtained after the use of the optimization loop. This will require SOMs to be executed multiple times so the number of computations for the training part will increase even more. Since the difference after the first 100 iterations was not that big, the KNN pre-distorter results for 100 SOMs optimization loops will be used and be compared with the LUT pre-distorter(see Figure 4.11 and 4.12). The selected training size is 40000 and the total number of classes after 100 loops was around 6000.

The project's simulation data were taken from a 28GHz band PA, with bandwidth up to 400MHz. The pre-distorter's memory depth for the LUT implementation was 3 and the polynomial order 5.

pre-distorter	MFLOP/sample	EVM(%)	ACPR(dBc)
KNN	14.6	9.5	-26.8
LUT	71	3.8	-45.2

Table 4.4: Summary of KNN and LUT pre-distorter

Observing Table 4.4, one can conclude that the KNN pre-distorter requires a small amount of computations per sample, compared to the original approach. However, the accuracy and the performance has dropped significantly and such an implementation would be difficult to meet the industry's specifications as those mentioned on Section 4.1. The next chapter will focus on discussing this projects results and some possible improvements that could be made to get this design one step closer to the industry's requirements.

Chapter 5

Conclusions and future work

5.1 Conclusions

The main goal of this thesis project was to develop an innovative solution for digital pre-distortion with the use of self-organizing maps and investigate the quality of the results. Using the LUT-based pre-distorter scheme as a base and a mean of comparison, this implementation is targeting on reducing the multiple LUTs on the forward path with only one that will be formed by SOMs and provided to the KNN algorithm in order to estimate the pre-distorters output value. In addition to that, this project's implementation needs to use the Memory Polynomial only for a limited amount of data in the beginning of the process to form the LUT.

The results are indicating that as it was expected the accuracy of the model is worse than the conventional one since it is based on predictions. However, in terms of computational complexity the KNN pre-distorter is more efficient than the LUT pre-distorter for the used PA. This difference can be even more significant if the KNN pre-distorter is compared with pre-distorters with higher memory depth and polynomial order due to the fact that in those cases the FLOPs are increasing in the conventional approach but will not affect the forward path of this design. However, it will affect the adaptation path with SOMs, which needs to be executed initially to form the LUT.

By investigating further the results, some interesting parts were observed such as that the accuracy of the model could be increased by increasing the number of classes. A scenario that makes sense. since predicting the output, using a limited number of pre-estimated values, generates some quantization to the output values which will lead to an increase of the error. In order to aim for the quality and not the quantity of the classes, the optimization loop was introduced and the error was reduced while the number of classes was maintained on a reasonable level. Due to the long simulation time in this project not all the possible scenarios and combinations were tested.

The project concludes that theoretically by starting with a small number of classes and introducing new classes through the optimization loop, the EVM and the ACPR can reach even lower levels and they might be able to meet

the specifications. To achieve that more simulations are required, with very long execution time, which will also lead to an increment in the amount of computations in the adaptation path, since SOMs will be executed more times.

5.2 Future work

As an extension of this thesis, I would like to mention four basic points that could be further examined and improved:

- The optimization of the existing system to reach better EVM and ACPR levels and meet the specifications.
- The replacement of the KNN with a different prediction algorithm that could be more accurate and hardware-friendly in terms of computations.
- The hardware implementation of the system and the testing in real-life scenarios with bigger amount of data.
- A comparison of this design with different pre-distorters with bigger memory depth and polynomial order.

References

References

[1] Electronics Hub, *What is a Power Amplifier? Types, Classes, Applications*, 2018. https://www.electronicshub.org/power-amplifier/#What_is_a_Power_Amplifier

[2] John Price and Terry Goble„ *Signals and noise*.Elsevier Inc, 1993.

[3] Xilinx, Inc *Understanding Key Parameters for RF-Sampling Data Converters*. White Paper: Zynq UltraScale+ RFSoCs, February 20, 2019.

[4] Erik Andersson, Christian Olsson, *Linearization of Power Amplifier using Digital Predistortion, Implementation on FPGA*. Department of Electrical Engineering, Linköpings universitet, Linköping, Sweden, 2014.

[5] Wei Wei, Ole Kiel Jensen, Jan H. Mikkelsen, *Self-heating and memory effects in RF power amplifiers explained through electrothermal*. IEEE Press, Aalborg Universitet, Denmark, 2013.

[6] Christian Henn, Burr-Brown International, GmbH. *Intermodulation Distortion (IMD)*. Burr-Brown International,Texas Instruments Incorporated, Tucson, Arizona, 2000.

[7] Ibrahim Can Sezgin, *Different Digital Predistortion Techniques for Power Amplifier Linearization*. Department of Electrical and Information Technology Faculty of Engineering, LTH, Lund, Sweden, 2016.

[8] Aidan Wilson, *A Brief Introduction to Supervised Learning*. Towards Data Science, September 29, 2019. https://towardsdatascience.com/a-brief-introduction-to-supervised-learning-54a3e3932590

[9] Barber, D, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[10] Nagyfi Richárd. *The differences between Artificial and Biological Neural Networks*. Sep 4, 2018. https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7

[11] Sylvester Kaczmarek, *What You Need to Know About Machine Learning*. Electronic Design an informa business, 2017. https://www.electronicdesign.com/industrial-automation/what-you-need-know-about-machine-learning

[12] Q.Wu and Z.Ni, *Car assembly line fault diagnosis based on triangular fuzzy support vector classifier machine and particle swarm optimization*. Expert Syst, 2011.

[13] M. Gardner and J. Bieker, *Data mining solves tough semiconductor manufacturing problems*. In Proc. 6th ACM SIGKDD Conference, 2000.

[14] Darko Stanisavljevic and Michael Spitzer, *A Review of Related Work on Machine Learning in Semiconductor Manufacturing and Assembly Lines*. SamI40 workshop at i-KNOW', Graz, Austria, 2016.

[15] A. F. Tabrizi et al. *A machine learning framework to identify detailed routing short violations from a placed netlist*. DAC, 2018.

[16] A. F. Tabrizi, N. K. Darav, L. Rakai, A. Kennings, and L. Behjat *Detailed routing violation prediction during placement using machine learning*. VLSI-DAT, 2017.

[17] W.-T. J. Chan, P.-H. Ho, A. B. Kahng, and P. Saxena *Routability optimization for industrial designs at sub-14nm process nodes using machine learning*. ISPD, 2017.

[18] Wei Zeng, Azadeh Davoodi, and Yu Hen Hu *Design Rule Violation Hotspot Prediction Based on Neural Network Ensembles*. University of Wisconsin–Madison, 2018.

[19] Dennis R. Morgan, Zhengxiang Ma, Jaehyeong Kim, Michael G. Zierdt, and John Pastalan *A Generalized Memory Polynomial Model for Digital Predistortion of RF Power Amplifiers*. IEEE Transactions on Signal Processing, 2006.

[20] Ibrahim Can Sezgin, *Different Digital Predistortion Techniques for Power Amplifier Linearization*. Department of Electrical and Information Technology, Faculty of Engineering, LTH, Lund University, 2016.

[21] Henna Paaso and Aarne Mämmelä, *Comparison of direct learning and indirect learning predistortion architectures*. ISWCS '08. IEEE, 2008

[22] Takao Inoue, *Digital Predistortion (DPD) Design-to-Prototype Framework for PA's*. National Instruments, AWR Corporation, 2013.

[23] F.M Ghannouchi and Oualid Hammi, *Behavioral Modeling and Predistortion*. Microwave Magazine, IEEE, 2010.

[24] Maria Canavate-Sanchez, Andrea Segneri, Apostolos Georgiadis, Savvas Kosmopoulos, George Goussetis, Yuan Ding *System performance evaluation of power amplifier behavioural models*. Heriot-Watt University, 2018.

[25] Jessica Chani-Cahuana, *Digital Predistortion for the Linearization of Power Amplifiers*. Communication Systems and Information Theory Group Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden, 2015

[26] D. Schreurs, M. O'Droma, A. A. Goacher, and M. Gadringer, *RF Power Amplifier Behavioral Modeling*. Cambridge, UK, 2008.

[27] L. Ding, G. T. Zhou, D. R. Morgan, Z. Ma, J. S. Kenney, J. Kim, and C. R. Giardina, *A Robust Digital Baseband Predistorter Constructed Using Memory Polynomials*. IEEE Trans. Commun., Vol. 52 No 1., 2014.

[28] Home of RF and Wireless Vendors and Resources, *Digital Predistortion Reference Guide*. Altera Corporation, 2003.

[29] Scott, A.W., Frobenius *Rex, RF Measurements for Cellular Phones and Wireless Data Systems*. Wiley/IEEE, 2008.

[30] RF Wireless World *EVM-Error Vector Magnitude*. `https://www.rfwireless-world.com/Terminology/Error-Vector-Magnitude.html/`.

[31] Zhenyu Wang, Yanyun Wang, Chunfeng Song, Tao Chen and Wei Cheng *Deep neural nets based power amplifier non-linear pre-distortion*, Phys.: Conf. Ser. 887 012049, 2017.

[32] Manish Sonal, *Machine Learning for PAPR Distortion Reduction in OFDM Systems*. KTH Royal Institute of Technology, School of Electrical Engineering, Stockholm, Sweden, 2016.

[33] Jun Peng, Songbai He, Bingwen Wang, Zhijiang Dai and Jingzhou Pang *Digital Predistortion for Power Amplifier Based on Sparse Bayesian Learning*. IEEE Trans. Commun., Vol. 63 No 9., 2016.

[34] James Peroulas *Digital Predistortion Using Machine Learning Algorithms*. Stanford University, California, 2016.

[35] Abhinav Ralhan *Self Organizing Maps*. Towards Data Science, 2018.

[36] Simon Haykin *Neural Networks: A Comprehensive Foundation*. 2nd ed. Prentice-Hall, Englewood Cliffs, New Jersey, 1999.

[37] Cengiz Kahraman *Computational Intelligence Systems in Industrial Engineering*. Atlantis Press, 2012.

[38] D. Bloice, Marcus and Holzinger, Andreas. *A Tutorial on Machine Learning and Data Science Tools with Python*. Holzinger Group HCI-KDD, Institute for Medical Informatics, Statistics and Documentation, Medical University of Graz, Graz, Austria, 2016.

[39] Dr. Meenakshi Sharma, Anjali Batra *Analysis ofDistance Measures in Content based Image Retrieval*. H.C.T.M., Technical Campus, India, 2014.

[40] Victor J.A.S. Lobo *Application of Self-Organizing Maps to the Maritime Environment*. Springer, 2009.

[41] Algobeans *Self-organizing maps tutorial*. Layman Tutorials in analytics, Algobeans.com, November 2, 2017.
https://algobeans.com/2017/11/02/self-organizing-map/

[42] Onel Harrison *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. Towards Data Science, 2018.

[43] Devin Soni *Introduction to k-Nearest-Neighbors*. Towards Data Science, March, 12, 2018.
https://towardsdatascience.com/introduction-to-k-nearest-neighbors-3b534bb11d26/

[44] Pedregosa, F. et al. *Scikit-learn: Machine Learning in Python*. JMLR 12, pp. 2825-2830, 2011.
https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html/

[45] Ryotaro Kamimura *Social Interaction and Self-Organizing Maps*. Applications of Self-Organizing Maps, Magnus Johnsson, IntechOpen, 2012.

[46] Suzanne Angeli, Arnaud Quesney and Lydwine Gross *Image Simplification Using Kohonen Maps: Application to Satellite Data for Cloud Detection and Land Cover Mapping*. Applications of Self-Organizing Maps, Magnus Johnsson, IntechOpen, 21 November, 2012.

[47] Li Jian, Yang Ruicheng and Guo Rongrong *Self-Organizing Map Method for Fraudulent Financial Data Detection*. IEEE, 3rd International Conference on Information Science and Control Engineering, 2016.

[48] Suhatati Tjandral, Amelia Alexandra, Putri Warsito and Judi Prajetno Sugiono *Determining Citizen Complaints to The Appropriate Government Departments using KNN Algorithm*. IEEE, 13th International Conference on ICT and Knowledge Engineering, 2015.

[49] Treesa George, Sumi. P. Potty and Sneha Jose *Smile Detection from Still Images Using KNN Algorithm*. IEEE, International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014.

[50] S.Venkata Lakshmi and T.Edwin Prabakaran *Application of k-Nearest Neighbour Classification Method for Intrusion Detection in Network Data*. IJCA, International Journal of Computer Applications, Volume 97– No.7, July 2014.

[51] Dr. Oliver Werther and Roland Minihold *LTE: System Specifications and Their Impact on RF Base Band Circuits* . Rohde Schwarz, 2013.

[52] Keysight Technologies. *5G NR (New Radio)*. Technical Overview literature number 5992-2916EN, 2019. https://www.keysight.com/se/en/assets/7018-06127/technical-overviews/5992-2916.pdf/

[53] Fabien Loison, Mathilde Boutigny. *PyPAPI's documentation*. Github, 2017. https://flozz.github.io/pypapi/

[54] Intg Ckts, *Adjacent Channel Power Ratio (ACPR)*. 2018. http://analog.intgckts.com/adjacent-channel-power-ratio-acpr/

Printed by Tryckeriet i E-huset, Lund 2020

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2020-789
http://www.eit.lth.se