High performance Deep Learning based Digital Pre-distorters for RF Power Amplifiers

Rajesh Kudupudi

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Dong S. Ha, Chair
Sook S. Ha, Co-chair
Guoqiang Yu

December 15, 2021
Blacksburg, Virginia

Keywords: Deep Learning, Power Amplifiers, Digital Predistorters.

High performance Deep Learning based Digital Pre-distorters for RF Power Amplifiers

Rajesh Kudupudi

(ABSTRACT)

In this work, we present different deep learning-based digital pre-distorters and compare them based on their performance towards improving the linearity of highly non-linear power amplifiers. The simulation results show that BiLSTM based DPDs work the best in terms of improving the linearity performance. We also compare two methodologies of direct learning and indirect learning to develop deep learning-based digital predistorters (DL-DPDs) models and evaluate their improvement on the linearity of Power Amplifiers (PA). We carry out a theoretical analysis on differences between these training methodologies and verify their performance with simulation results on class-AB and class-F^{-1} PAs. The simulation results show that both the learning methods lead to an improvement of more than 12 dB and 11dB in the linearity of class-AB and class-F^{-1} PAs respectively, with indirect learning DL-DPD offering marginally better performance. Moreover, we compare the DL-DPD with memory polynomial models and show that using the former gives a significant improvement over the memory polynomials. Furthermore, we discuss the advantages of exploiting a BiLSTM based neural network architecture for designing direct/indirect DPDs. We demonstrate that BiLSTM DPD can be used to predistort signals of any size without the drop in linearity. Moreover, based on the insights we develop a frequency domain loss using which further increased the linearity of the PA.

High performance Deep Learning based Digital Pre-distorters for RF Power Amplifiers

Rajesh Kudupudi

(GENERAL AUDIENCE ABSTRACT)

Wireless communication devices have fundamentally changed the way we interact with people. This increased the user's reliance on communication devices and significantly grew the need for higher data rates and faster internet speeds. But one major obstacle inside the transmitter chain (antenna) with increasing the data rates is the power amplifier, which distorts the signals at these higher powers. This distortion will reduce the efficiency and reliability of communication systems, greatly decreasing the quality of communication. So, we developed a high-performance DPD using deep learning to combat this issue. In this paper, we compare different deep learning-based DPDs and analyze which offers better performance. We also contrast two training methodologies to learn these DL-DPDs, theoretically and with simulation to arrive at which method offers better performing DPDs. We do these experiments on two different types of power amplifiers, and signals of any length. We design a new loss function, such that optimizing it leads to better DL-DPDs.

Dedication

This work is dedicated to my family and friends who have supported me uncondionally along the way.

Acknowledgments

I would like to thank my advisor Dr. Dong S Ha for his guidance and support. Thank you for providing me the opportunity for working on this project. I would like to thank Dr. Keyvan Ramezanpour for being a constant mentor to me. This work would not have been possible without his guidance and insights. I would like to thank Dr. Fariborz Pour for making the testbench which was crucial for generating the data and providing helpful feedback to me along the way. I would like to thank Dr. Sook Ha for being my co-advisor. I would also like to thank Dr. Guoqiang Yu, for teaching the course optimization techniques which was very useful, and for being on my committee. I would like to thank all the MICS members for providing a excellent work environment.

Contents

List of Figures

List of Tables

List of Abbreviations

ACPR Adjacent Channel to Power Ratio

Bi-LSTM Bi-directional Long Short Term Memory units

CNN Convolutional Neural Network

DL Deep Learning

DL-DPD Deep Learning based Digital Pre-Distorter

DPD Digital Pre-Distorter

PA Power Amplifier

PSD Power Spectral Density

Chapter 1

Introduction

As more and more devices are connected to the internet, wireless communication systems are going to be everywhere. A general trend of these advanced communication systems is towards achieving higher data rates to meet the ever increasing data demands by the users. This requires the use of signals with higher order modulation schemes that leads to signals with high peak to average power ratio (PAPR) which necessitates the use of highly linear transmitter chain. The performance of PA in the transmitter chain is the bottleneck of the linearity. However, the linearity of the PA, is limited due to the non-idealites of the active devices (i.e transistors). This non-linear behavior of PA becomes a severe issue when it operates with high PAPR signals, leading to significant decrease in adjacent channel to power ratio (ACPR) and degrading the quality of communications. One effective solution to this problem is designing a digital predistorter (DPD), which modifies the signal accounting for the non-linearity of the PA [6, 11]. There has been a wide range of interest by researchers to develop DPDs in-order to improve the linearity of PA.

Traditionally, volterra series based memory polynomial models have been used to design the DPD [4, 7, 11]. These models use volterra kernels with different non-linear and memory orders. However, Braithwaite et. al points out several limitations of these models and showed their inability to compensate for extremely non-linear PAs [5]. With success of deep learning in many fields such as image processing, time series analysis, speech and text analytics, motivated researchers to design DL-DPD [1, 3, 8, 9, 10, 13, 14, 18, 19, 21, 22, 23,

24, 25, 27]. From a deep learning standpoint, the function of the DPD can be interpreted as an sequence to sequence conversion task, where given a input signal in digital domain, the DPD predistorts the signal and outputs a sequence such that it improves the linearity of the PA. However, majority of the works in designing DL-DPD are based on multi layer perceptron (MLP) and convolutional neural networks (CNN) which are not suitable for these sequence to sequence conversion task because of their inability to model memory effects of the PA [2, 3, 9, 10, 20, 22, 23]. We adopt a bi-directional long short term memory (BiLSTM) architecture for the DPD, which can readily model the memory effects. In this thesis we train different neural network archietctures to model the DPD and present a comparison to show which architecture leads to a better performing DPD. Further in this thesis we explore the effectiveness of our DPD on PAs with different characteristics i.e. class-AB and class-F^{-1} PA. We believe this is important to gauge the effectiveness of DPD models. Moreover we show that the BiLSTM DL-DPD can handle inputs of anysize without significant drop in the performance.

The traditional deep learning algorithms are based on supervised learning which require the input and output sequences for training. However, the pre-distorted signal for training the deep learning model is not available. Hence, two different methods indirect and direct learning are used to train the DL-DPD models [9, 22, 23, 24]. In the indirect learning approach the inverse function of the PA is learned and the coefficients of this inverse model are copied to be used as a pre-distorter. On the other hand, direct learning is a multi step process. First a behavioral model of the PA is learned and it's parameters are freezed. Then, the DPD model is learned by back propagating the gradients of error function through the behavioral model of the PA. In this paper, we train the DL-DPD using both of these methods and compare their resulting performance in-terms of improving the linearity of PA. Given the rising interest in DL-DPD, it is of paramount importance to know which way of

training them yields better performing DPD models. To our knowledge, similar comparisons has not been presented previously with DL-DPD. Further, we present a theoretical analysis highlighting the differences between these two training procedures.

Further, based on the insights we develop a frequency domain loss to train the DL-DPD. We found that optimizing this loss function while training the DL-DPDs which further increased the linearity of the PA in comparison with MSE in spatial domain.

The remainder of this paper is organized as follows. In section II, we establish some preliminaries and discuss some related works. In section III we present the theoretical differences between direct and in-direct learning training methodologies. In section IV we discuss our simulations. Finally, section V concludes this work.

1.1 Research Objectives

The research objectives are listed below.

1. To create a high performance DPD by comparing different deep learning model architectures.

2. To compare two different training methodologies for training DL-DPDs, and showing which methodology yields better linearity.

3. Propose a new loss function to learn the DL-DPDs, to achieve better linearity performance.

4. To check the effectiveness of DL-DPD on different PAs and signals of varying sizes.

Chapter 2

Preliminaries

This chapter provides a brief overview of the relevant literature and establishes the necessary background. This chapter is presented as different sections. The first section gives a brief overview about class AB and class F^{-1} power amplifiers for which we will be developing DPDs. The second section dives into the need for digital predistorters and the disadvantages of the current DPDs. The third section establishes preliminaries related to deep learning, which are useful to understand the current DL-DPDs presented in fourth section of this chapter.

2.1 Power Amplifiers

RF power amplifiers is the last component inside the transmitter chain Fig. 2.1 of a wireless communication systems. It's purpose is to amplify an modulated RF signal to a specific power level and deliver it to the antenna. This amplification is essential for the signal to be recovered at the receiver. Task of amplifying the input signal makes PAs a major contributor of energy consumption in RF transmitter chain. So, one essential requirement of PAs is, for them to be highly energy efficient. Unfortunately, several problems related to the operating regions of PAs, PAPR of the input signals, modulation schemes of latest wireless communication systems, make highly efficient PAs cause severe signal distortions due to the non-linear behavior of the active devices. These distortions impact the reliability of the

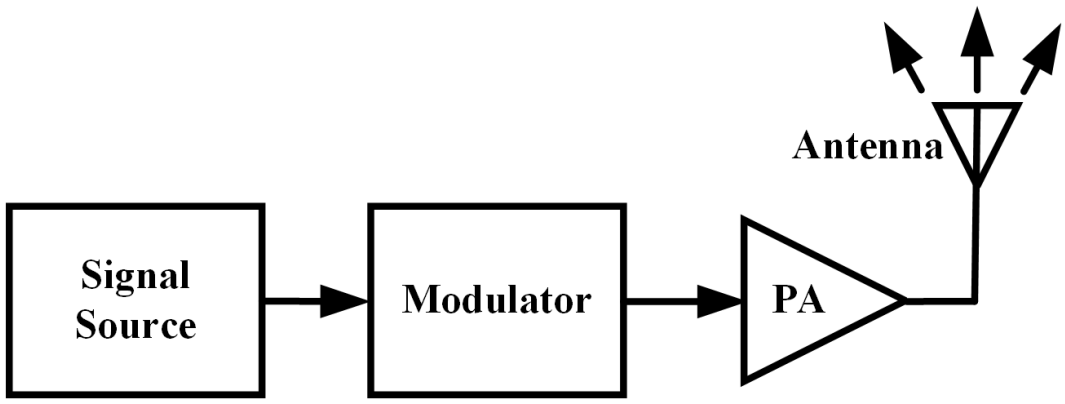

Figure 2.1: Pictorial representation of different components in an RF transmitter chain

communications system, and cause spectral regrowth which inturn leads to interference of transmitted signals in the neighboring channels as illustrated in Fig. 2.3.

It is not possible to design a PA which operates with high efficiency and delivers low non-linearity. From Fig. 2.2 it can be seen that PA which operates at low efficiency is less non-linear and PA which operates at high efficiency is more non-linear leading to signal distortions. One way to overcome the signal distortions is to design a pre-distorter which accounts for these distortions. But before we discuss a pre-distorter, it is imperative to understand the kind of non-linear behavior and different effects PA presents so that they can be modeled successfully and accounted for. Any PA design generally consists of four blocks, the bias network, the active devices and the I/O matching networks. The different kinds of non-linear behaviors of PA are caused due to,

- **Static Non-linearity**: This can be attributed to nonlinear DC characteristics of the active devices (i.e transistors) inside the PAs.
- **Memory effects**: There can be two types of memory effects short term and long term. The short term effects can be because of the capacitors, and other device parasitics. The long term effects are due to self-heating and trapping effects caused by the power

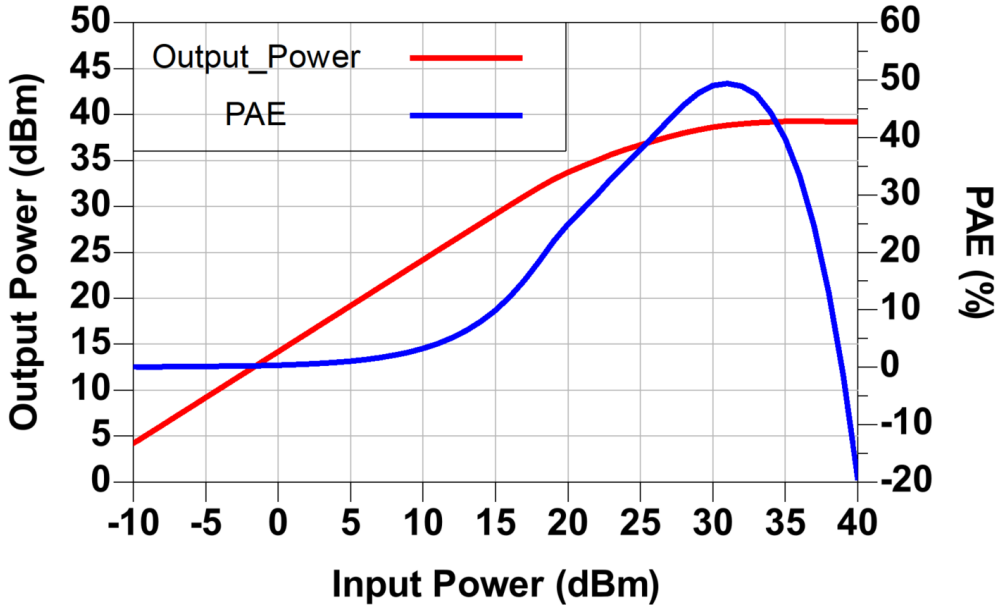

Figure 2.2: Power Average Efficiency (PAE) and Output Power (Pout) plotted against Input power (Pin). The (Pout vs Pin) curve shows how lineraity degrades with increasing input power. The (PAE vs Pin) curve depicts the PAE, which is highest at the knee of the red curve.

Table 2.1: ACPR of distorted output by the class AB, class F^{-1} and their corresponding inputs.

DPD Model Details	class-AB Lower/Upper ACPR(dB)	class-F^{-1} Lower/Upper ACPR(dB)
Without DPD	45.19/44.53	36.40/30.4
Input Signal (Ideal)	83.52/80.32	85.74/84.877

dissipation of active devices.

These effects lead to a spectral regrowth which can be seen in Fig. 2.3

The designed pre-distorter should account for these effects to undo the distortion induced by the PA. In this thesis we demonstrate the ability of our pre-distorter on two widely popular types of PAs, class AB and inverse class F.

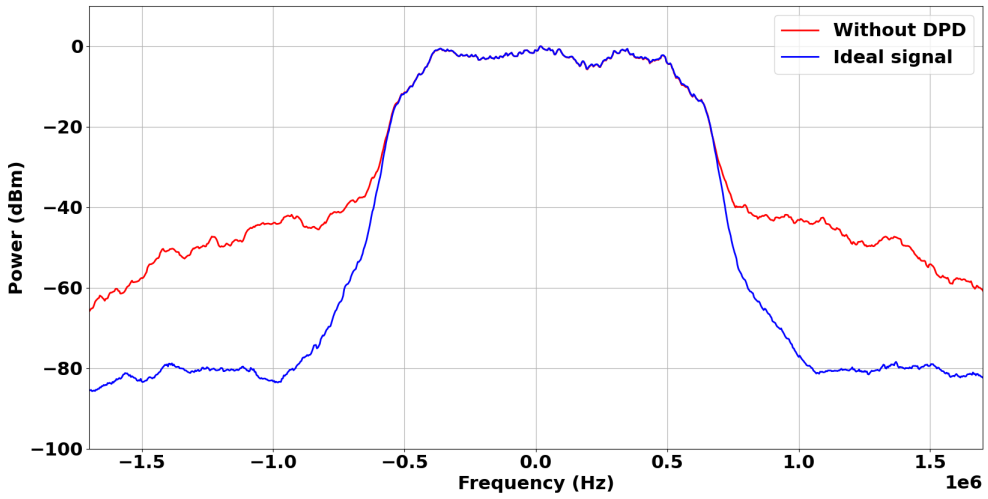

Figure 2.3: For Class AB PA: Input signal and the signal distorted by the PA (Red) are plotted. Spectral regrowth of third order harmonics is visible in the distorted signal.

2.1.1 Class AB PA

Class AB power amplifier as the name suggests is an hybrid of class A and class B PAs. This class of PAs generally do not offer great efficiency, but offer good linearity. These can be most commonly found in audio devices. The PSD of class AB PAs is found in Fig 2.3. The corresponding ACPR from the PSD plot can be found in Table. 4.4

2.1.2 Inverse Class F PA

Class F PAs offer higher efficiency using harmonic resonators in the I/O network. So, class F PAs are generally preferred for high power efficiency applications like in radars etc. But this comes at the cost of higher non-linearity as can be seen from Fig 2.4.

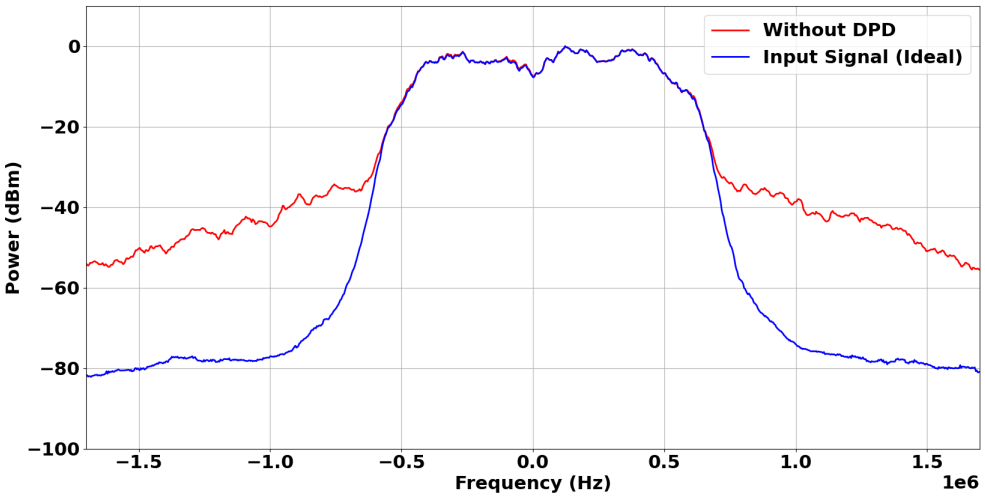

Figure 2.4: For Class F^{-1} PA: Input signal and the signal distorted by the PA (Red) are plotted. Spectral regrowth of third order harmonics is visible in the distorted signal.

2.2 Digital Predistorters

Digital pre-distorter (DPD) predistortes the input signal in digital domain such that it accounts for the signal distortion. This is one common and popular method to address the non-linearity's induced by the PA. As shown in Fig 2.5 the output of the DPD is fed into the PA, with a goal that the DPD-PA cascade behaves linearly, such that the spectral regrowth is reduced. Earlier versions of DPDs were designed using look-up tables and memory less systems. As the communication signals gravitated towards higher PAPR and wideband signals better DPDs were required to compensate for the non-linear and adverse memory effects induced by the PA. DPDs based on lookup tables and memory less systems suffered from an inherent disadvantage, which was their inability to model the memory effects of the PA. This lead to an increased interest in memory polynomials [4, 11, 16] and neural networks based DPDs. The rest of this section discusses about the memory polynomial based DPDs, and later the neural network based DPDs.

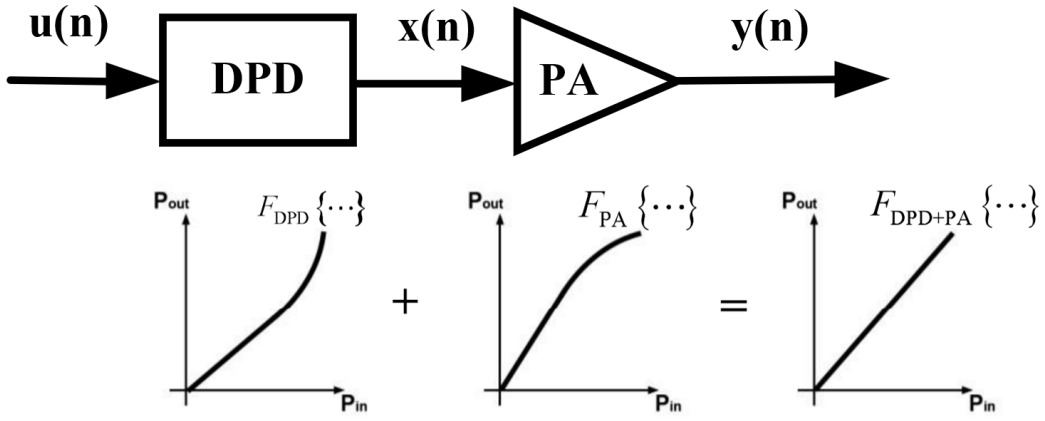

Figure 2.5: DPD, PA block diagram, with their transfer functions.

2.2.1 Memory Polynomial based DPDs

Majority of the memory polynomial (MP) based DPDs were derived from volterra series, which is a mathematical technique used to represent the input-output relationship of dynamical non-linear systems. In order to model the memory effects time shifted versions of inputs are provided to the MP based model. A simple MP model is represented as

$$y_{mp}(n) = \sum_{k=0}^{K_1} \sum_{l=0}^{L_1} a_{kl} x(n-l)|x(n-l)|^k \tag{2.1}$$

Here K_1 is the non-linear order and L_1 is the memory order. Another important model which Dennis et.al claim to outperform the MP model is a generalized memory polynomial (GMP) model [16]. The GMP model extends the MP model by including the future terms also in the expression. The GMP model is given by

$$\begin{aligned} y_{gmp}(n) = \sum_{k=0}^{K_1} \sum_{l=0}^{L_1} a_{kl} x(n-l)|x(n-l)|^k + \sum_{k=0}^{K_2} \sum_{l=0}^{L_2} \sum_{m=1}^{M_2} (b_{klm} x(n-l)|x(n-l-m)|^k \\ + \sum_{k=0}^{K_3} \sum_{l=0}^{L_3} \sum_{m=1}^{M_3} c_{klm} x(n-l)|x(n-l+m)|^k) \end{aligned} \tag{2.2}$$

Here the terms K_1, L_1 and their corresponding coefficients (a_{kl}) are to compensate for the non-linear and memory effects of the PA. And the term M_2, M_3 for coefficients of lagging and leading envelopes of the signals and their corresponding cross terms. This lagging and leading envelope portions introduce cross terms which allow the GMP to outperform MP model. The advantage of this formulation is that the GMP, MP formulation can be expressed as a linear function of the coefficients which makes it suitable for usage of any least squares type of algorithm. The optimization objective of the least squares objective can be given by

$$J(w_{gmp}) = \frac{1}{N}\sum_{n=0}^{N}(y_{gmp}(n) - y(n))^2 \tag{2.3}$$

Here w are the parameters of GMP least squares algorithm is solving for. N is the total number of samples. $y_{gmp}(n)$ and $y(n)$ are the predictions of GMP model and the ground truth labels respectively. The solution to least squares problem can be written by the normal equation,

$$w_{gmp} = (A^T A)^{-1} A^T y \tag{2.4}$$

where A is a matrix consisting of all the input terms (i.e their non-linear, memory orders) and y is the ground truth vector.

However, the high correlation between polynomial bases makes it difficult to further improve the performance of these GMP models, thereby limiting their ability to model higher orders of non-linearity [5]. The next section would establish some preliminaries about different deep learning based algorithms which would be used when discussing DL-DPDs.

2.3 Deep Learning

In the past decade, due to the success of deep leanring algorithms in fields like computer vision, natural language processing, speech recognition etc, they have exploded in popularity. These deep learning algorithms are a means to realize artificial intelligence through experiential learning. Training deep learning algorithms in general are computationally expensive. The new generations of graphics processing units with massive parallel compute power enabled the success of the computation hungry deep learning algorithms. The traditional machine learning algorithms require manual extraction of features, whereas deep learning algorithms use multiple layers to learn the best features from the data. This drastically reduces the work of designing hand crafted feature extractors thereby reducing the design time required.

2.3.1 Deep Neural Networks

Neural network is a basic deep learning algorithm, that is designed in a way that they mimic the human brain. Neuron is a smallest unit inside a neural network. These neurons are arranged in layers such that a neuron in every layer is connected to every neuron in the previous layer (with some weights). Combination of these neurons in different arrangements enables a neural network to make complex decisions. The activation's of neurons from previous layers are transformed with these weights and passed through a non-linear function called activation function. This activation can be sigmoid, relu, tanh, etc. This non-linearity gives the neural network ability to model non-linear functions.

When a neural network has more than three layers it is referred to as a deep neural network. Neural networks or any deep learning algorithm is trained to minimize an objective function called loss function. For doing this neural networks iteratively update their weights using an

algorithm called gradient descent. Deep neural networks were very successful for handwritten character recognition (MNIST).

2.3.2 Convolutional Neural Networks

The issue with deep neural networks is that, when dealing with images of 256*256 or higher, the first layer will contain 65536 parameters. This quickly increases the total number of weights required, making deep neural networks unsuitable for unstructured data like images and time series. CNNs combat this problem by using weight sharing, which applies or convolutes the entire previous layer with same filter. A typical CNN contains two layers, the convolutional layer and fully connected layers. It also has pooling and batch-normalization but these are generally referred to as operations than layers. The convolutional layers extract spatial or temporal features from the data. The features represent presence of low level details like edges and corners (in the context of an image) in lower layers and as we move deeper into the CNN, the convolutional layers look for high level features like complex shapes. Fully connected layer behaves just like a neural network, but operates on information from the convolutional layers. This is generally the last layer of CNNs. CNNs are very commonly used in solving problems related to object detection, classification, segmentation, image generation etc. They have also been shown to produce state of the art results in audio related tasks.

2.3.3 Recurrent Neural Networks

The deep learning algorithms discussed above deep neural networks and CNNs, cannot inherently model temporal relationships. That is the output of these networks at any time instant is dependant on the input at that particular time instant. on the other hand all the RNNs have a chain like structure of repeating modules of neural networks. In a standard

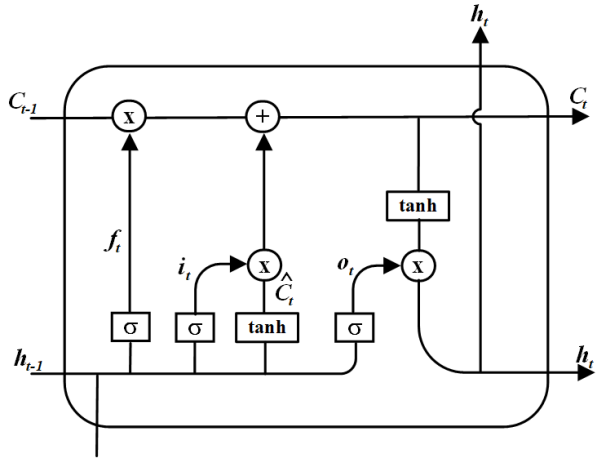

Figure 2.6: Inside an LSTM Block.

RNN, this module will have a simple structure like a single tanh activation. Some of the applications where modeling long term temporal relationships is crucial for decision making are weather forcasting, stock price prediction etc. At each step in an RNN chain, RNNs propagate a hidden vector from all the previous states, which transmits information from one RNN block to another, thereby enabling RNNs to model memory relationships.

Although RNNs have a great success in modeling short term memory relationships, they struggle with the long term ones. This problem of RNNs can be attributed to the vanishing gradients problem during the backpropagation. Vanishing gradients refers to the point that gradient decreases as we keep backpropagating the loss through the RNN chain. This makes the gradient converge to zero. Zero gradient implies that the weights of RNN will not be updated, causing the RNN to stop learning. This makes training long RNN chains not feasible, which means it is tougher for them to model long term history.

2.3.4 LSTM

LSTM an abbreviation for long short term memory networks, a kind of RNN which is capable of learning long-term dependencies without suffering from the vanishing gradient problem.

LSTMs modules are complex when compared to a regular RNN module. A figure of LSTM module is presented below in Fig 2.6. In this section we present in detail regarding working of an LSTM module.

The key components inside an LSTM block are gates, cell state C_t and hidden state h_t. The cell state runs straight down the entire LSTM chain, with only minor linear interaction, so the information can flows along the cell state unchanged. The gates can regulate LSTM to add or remove information to the cell state. The first step of the LSTM is a decision made by the sigmoid layer called "forget gate" f_t. It takes in the previous hidden state and the current input and decides what should be set to 0 in the cell state. f_t can take values between 0-1, as it is comprised of sigmoid function.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{2.5}$$

The next step in LSTM is to decide what information is going to be added to the cell state. This has 2 parts the tanh function creates a vector $\hat{C}_t$. Later, $\hat{C}_t$ is multiplied with i_t, to arrive at a vector which can be added to cell state, to update it.

$$i_t = \sigma(W_i.[h_{t-1}, x_t + b_i]) \tag{2.6}$$

$$\hat{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{2.7}$$

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{2.8}$$

Finally the output of any single LSTM block would be a combination of information from the new cell state, and the previous hidden state.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{2.9}$$

$$h_t = o_t * tanh(C_t) \tag{2.10}$$

Here h_t is the result of a single LSTM module which can be further transformed. Since the cell state is not operated on directly, the cell state can freely flow from through the entire LSTM chain, avoiding the vanishing gradients problem. LSTM is widely popular among natural language processing community, in applications like text/articles summarization, language translation etc.

2.3.5 Optimizers

The weights of deep learning algorithms are updated to minimze the loss function, by taking small steps in the opposite direction of the gradient of the loss function. The rule or how the weights are updated by is given by the optimizers. We only use Adam optimizer for our experiments, so we present a brief overview of Adam optimizer.

Adam can be looked as a combination of stochastic gradient descent (SGD) with momentum and RMSprop. Adam takes advantage of the momentum which is moving average of the gradients, that helps the weights reach minimum faster. Like RMSprop each element inside the weights has it's own learning rate, making Adam an adaptive learning rate method.

$$w_t = w_{t-1} - \eta \frac{m_t}{\sqrt{v_t} + \epsilon} \tag{2.11}$$

Here w represent weights, η the learning rate, m_t is the momentum term and v_t is the scaling factor which changes the learning rate.

2.4 DL-DPDs

In the last decade, the exceptional ability of deep learning algorithms to represent complex non-linear functions and solve problems in computer vision, text analytics etc. has made them popular [1, 2, 3, 8, 9, 10, 13, 21, 22, 23, 24]. This motivated researchers to apply deep learning algorithms to design DPDs. Most of these works are based on MLP, CNN architectures, which requires a lot of fine tuning [3, 9, 10, 22, 23, 24]. However, as these neural network architectures cannot readily model the memory effects of the PA, a significant amount of time has to be spent towards fine tuning these architectures. On the other hand, BiLSTM architecture has two LSTM networks, forward and backward networks which propagate hidden vectors between them and give the BiLSTM access to previous and the future information. This gives BiLSTM access to leading and lagging input terms that is similar to the GMP. Hence, we adopt a BiLSTM for designing DPD. Although BiLSTM was used as a DPD in [21], we explore its performance on class-AB and class-F^{-1} and compare there results.

2.5 Training DL-DPDs

All the deep learning models we discussed above are supervised learning based, i.e they need input and output sequences for training. But, while designing DPDs we do not have access to the pre-distorted signal in digital domain. Therefore the only information available in order to design the DPD is, inputs to the PA and distorted outputs of the PA. The two prominent methods to train DL-DPDs are, indirect learning and direct learning.

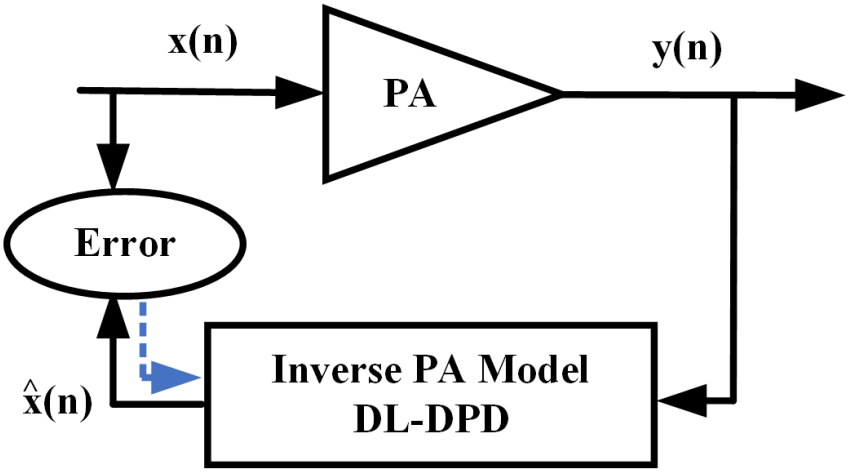

Figure 2.7: Training DL-DPD indirect learning. Dotted lines show the direction of the gradient.

2.5.1 Indirect Learning

Since the DPD should undo the distortions induced by the PA, one straight forward solution would be to learn the inverse function of the PA. Then the entire system i.e the cascade of DPD, PA would behave linearly. One way to do this would be to train the DPD with PA output sequence as the input to the DL-DPD and PA input sequence as the output. This can be illustrated with the Fig 2.7.

MSE can be used as the loss function to learn the weights of the inverse model, such that it minimize the error between PA input and the DL-DPD output. In this methodology we learn a post distorter and use it as a pre-distorter.

2.5.2 Direct Learning

Learning an inverse of the PA and using it as a pre-distorter does not always lead to a good DPD performance [12, 15, 17, 26]. Zhou et.al [26] claim that measurements from PA could be noisy, leading to a biased DPD model as we learn directly from the PA data. These kinds

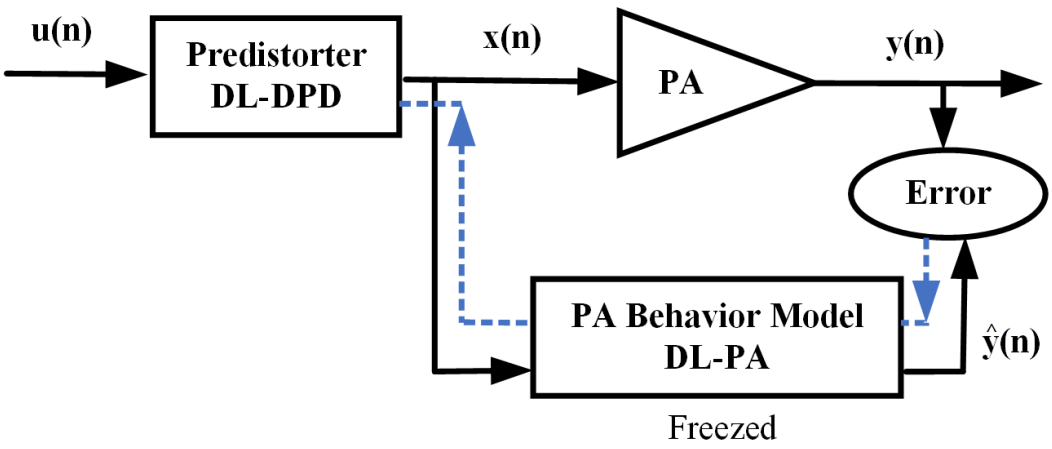

Figure 2.8: Training DL-DPD direct learning. Dotted lines show the direction of the gradient.

of issues do not exist with direct learning as we do not directly train with PA data. In direct learning we directly learn a pre-distorter.

Direct learning training methodology includes multiple steps. As the first step in direct learning a deep learning model is trained to mimic the behavior of the PA (DL-PA model). Then an another deep learning model is trained to learn the DPD (DL-DPD model), using the PA-DL. This is done by cascading the DL-DPD and DL-PA, and backpropagating the loss function through the DL-PA to train the DL-DPD Fig. 2.8. During training the DL-DPD, the weights of DL-PA are freezed, so that only DL-DPD weights can change. This is necessary to keep DL-PA unchanged, while learning the DL-DPD. A schematic of direct learning methodology is illustrated in Fig. 2.8.

In this work we present a comparison between these both training methodologies, to determine which method yields a high performance DL-DPD. A comparison between direct and indirect learning methodologies was made by [17] using memory polynomial/LUT based DPD's using self tuning controllers to optimize for the DPD coefficients. However, this comparison is not applicable for DL-DPDs since, DL algorithms use gradient descent for arriving at the optimal model coefficients.

Chapter 3

Simulation setup

To collect the data from the power amplifers we designed a test bench. Fig. 3.2 shows the simulation setup, which consists of transistor level power amplifiers.

3.1 Testbench

Two classes of PAs are designed in Keysight Aglient Advanced Design Systems software (ADS) using wdf speed 6W CGH40006P transistor cree model galium nitride (GaN) on silicon carbide (SiC) transistors. A quadrature amplitude modulator (QAM) was used to generate random baseband signals of order 64 (I, Q). Different random signals can be generated by controlling the seeds of the QAM signal generator. The Up-mixer is used to convert this signal to carrier frequency of 1 GHz and the resulting signal is delivered to the PA. Inorder to sample the output of the PA, it is down-converted through the DC-mixer and demodulated in the baseband domain ($I^{'}$, $Q^{'}$). It should be noted that attenuator is used to adjust the output signal of the PA before delivering it to the mixer.

The baseband signals (I, Q) and ($I^{'}$, $Q^{'}$) were sampled from the testbench. These sampled signals were split into training and test sets which were used for training and evaluating the DPDs respectively. We generated signals of length 10 signals each of length 5531, and recorded them before and after passing through the PA in the testbench. These signals were split into 80% training, 10% cross validation and 10% test sets respectively.

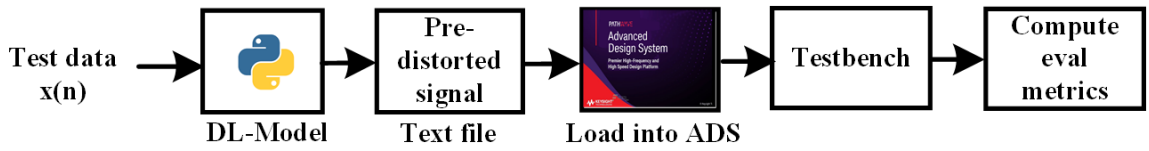

Figure 3.1: Evaluation flow chart.

3.2 Evaluation Criteria

The evaluation metrics for our DPDs are described in this section. All the results on the test set are obtained from the PA model in ADS. Fig. 3.1 shows our testing pipeline. The DL-DPD in python generates a text file which is the pre-distorted signal which is loaded in ADS software and passed through the testbench to generate the evaluation metrics.

3.2.1 Adjacent Channel Power Ratio ACPR

The adjacent channel power ratio, ACPR also referred to a adjacent channel leakage ratio is a performance used to measure the spectral regrowth of an RF transmitter chain. It measures the power of distortion components that leaked into the adjacent channels in relation to the power of signal in the baseband channel. In simple terms ACPR is used to quantify the non-linearity of power amplifiers. Since there are two adjacent channels to the baseband to the left and right of it, we would have two ACPR, one corresponding to the lower band and

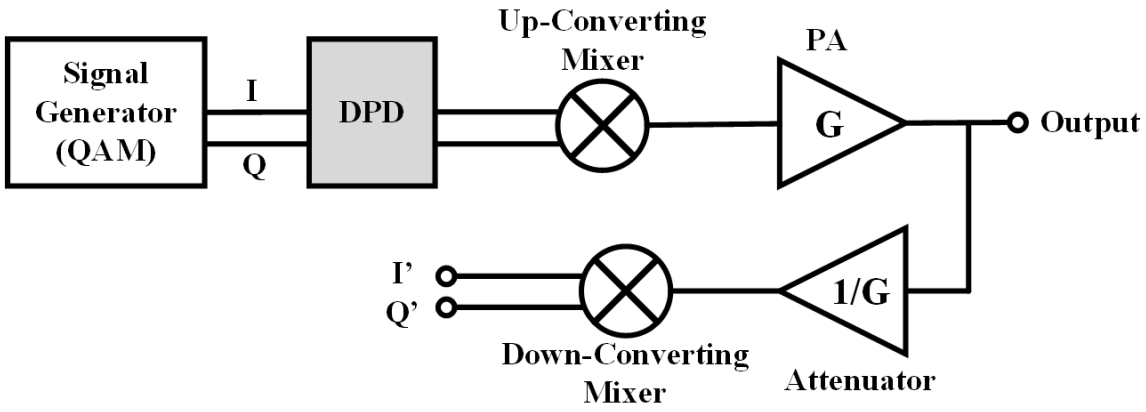

Figure 3.2: Testbench used to generate data and test the DPD.

other with respect to the upper band. P_{bb} is the power in baseband.

$$LowerACPR(dB) = 10 * \log(\frac{P_{lower}}{P_{bb}}) \tag{3.1}$$

$$UpperACPR(dB) = 10 * \log(\frac{P_{upper}}{P_{bb}}) \tag{3.2}$$

3.2.2 Mean Squared Error

As the name suggests we compute the mean squared error between the output of the DPD model and teh ground truth, which would vary depending on the methodology we used to train. For indirect learning it would be difference between the input of the PA and the output of the DL-DPD model. In this work we present MSE only on the cross validation set. Although MSE gives us an estimate of how close the DL-DPD's output is to the ground truth, it is not as intuitive or important as ACPR which directly relates to the linearity of the DPD and PA cascade. $\hat{y}_i$ corresponds to the DPD output and x_i the ground truth.

Chapter 4

Results And Analysis

All the ACPRs, and power spectral density plots generated in this chapter, were from the test set. The pre-distorted signals generated by DPDs were loaded into ADS software and passed through the PA models via the testbench created. The memory polynomial model demonstarted in the simulations is from matlab's communications toolbox with memory order and non-linear order of 5 respectively. The DL models were created in Python using Pytorch framework. The corresponding pre-distorted signals were passed through the PA and the Power Spectral Densities (PSD), ACPR of the final signal (I , Q) are compared.

4.1 Different deep learning architectures for DPD

We first compared different DL-DPDs against each other and the memory polynomial based DPD, to observe which offers better linearity performance or ACPR. The different DL architectures we tried out include, CNN, LSTM and a bidirectional LSTM (BiLSTM). All of the DL-DPDs were trained using indirect learning. These simulations are for a class AB PA.

From our experiments we observed that the BiLSTM based DL-DPD outperformed all the other DPDs interms of ACPR (Table. 4.1) and MSE (Fig. 4.2). We observed that the LSTM based DL-DPD was slightly behind the BiLSTM based model. The sizes of hidden vectors and number of fully connected layers were the same for LSTM and BiLSTM. This shows that including the future information leads to better linearity performance. This is inline

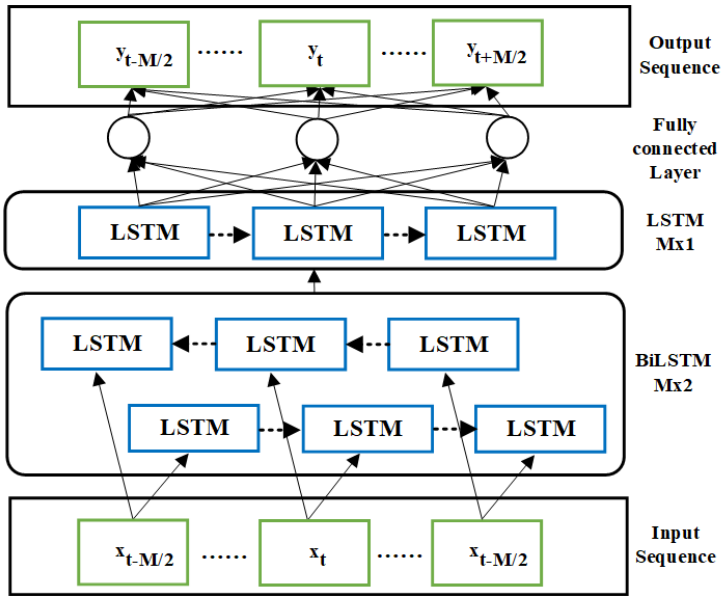

Figure 4.1: BiLSTM model architecture.

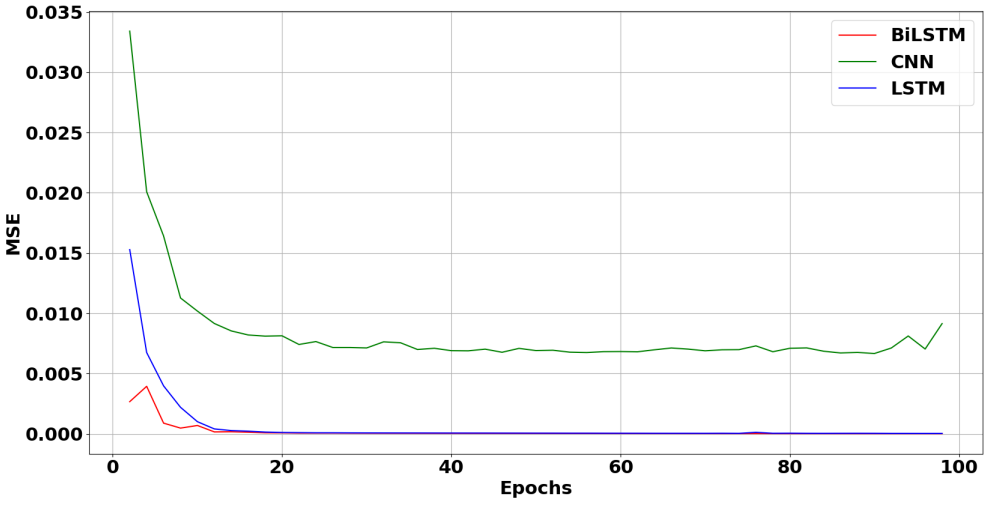

Figure 4.2: CNN, LSTM and BiLSTM DPD's MSE over the cross validation set for 100 epohcs.

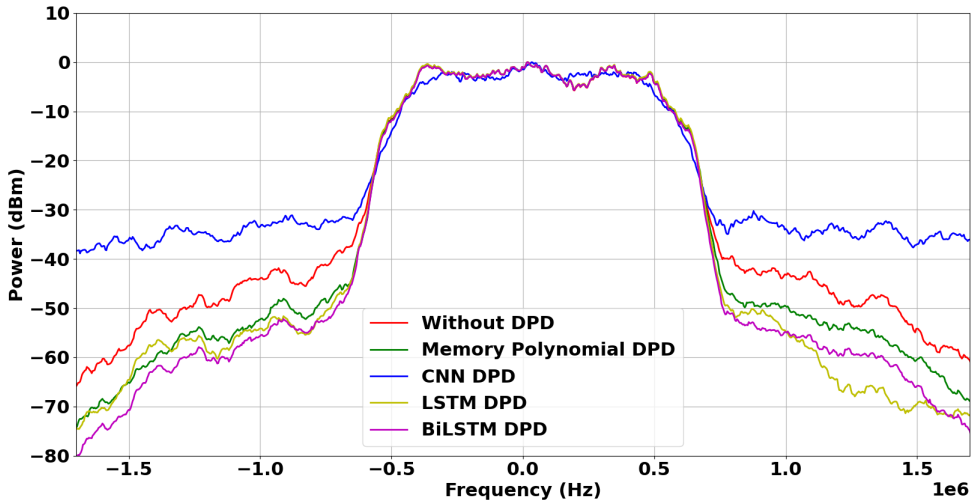

Figure 4.3: Power spectral densities of different deep learning models with test data of class AB PA

Table 4.1: ACPR for different DPD models on class-AB QAM signals.

DPD Model Details	class-AB Lower/Upper ACPR(dB)
Without DPD	45.19/44.53
Memory Poly DPD	55.02/51.80
CNN DPD	34.1/35.19
LSTM DPD	55.788/56.85
BiLSTM DPD	57.65/56.41

with the insights of GMP model, that including future information helps the DPD model the self heating effects. The LSTM and BiLSTM based DPDs always outperform memory polynomial based DPDs. The CNN based DPD has the worst performance of the lot. We used a 5 layer CNN, with 1 fully connected layer.

The dependency of BiLSTM on future samples necessitates the use of a time delay block that is used to gather some future I/Q baseband inputs. This would result in increased but acceptable latency.

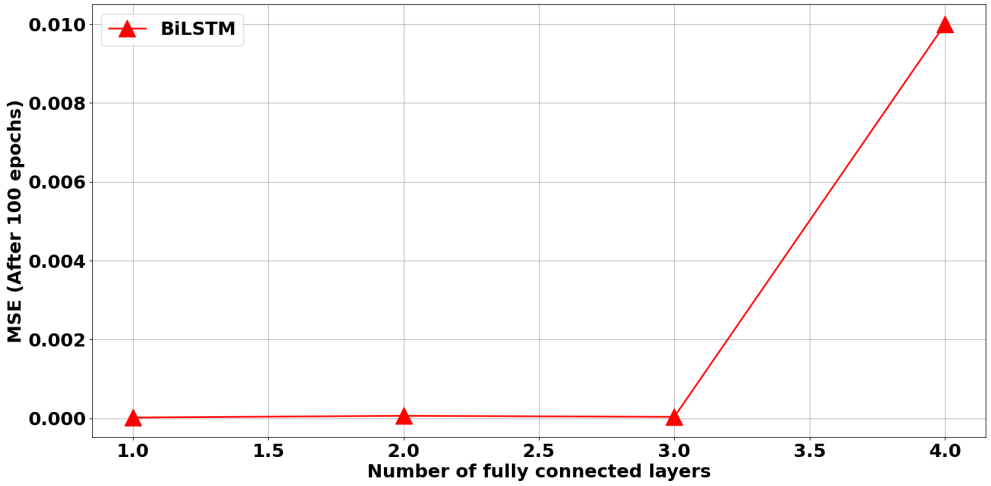

Figure 4.4: BiLSTM cross validation error after 100 epochs with different number of fully connected layers.

4.2 Hyper-parameter tuning for BiLSTM

To set the hyper-parmeters like hidden, cell state sizes and number of fully connected units for our BiLSTM architecture (Fig. 4.1) we performed a search over the hyper-parameter space. For a given set of hyper-parameters we train our BiLSTM architecture over the entire training set for 100 epochs. Then we compute the MSE over cross validation set and arrive at the set of hyper-parameters which yield the lowest cross validation MSE.

When varying one of the hyper-parameters all the others are kept constant. First we start by setting the number of fully connected layers to 1, and vary the hidden vector size. After we have arrived at the best hidden vector size, we vary the number of fully connected layers. Fig 4.4 shows that increasing the number of fully connected layers from 1, has a detrimental effect on the performance. Fig 4.5 shows that we achieve that increasing the hidden vector size, does decrease the cross validation MSE. But, after the size of 256, the gains are negligable.

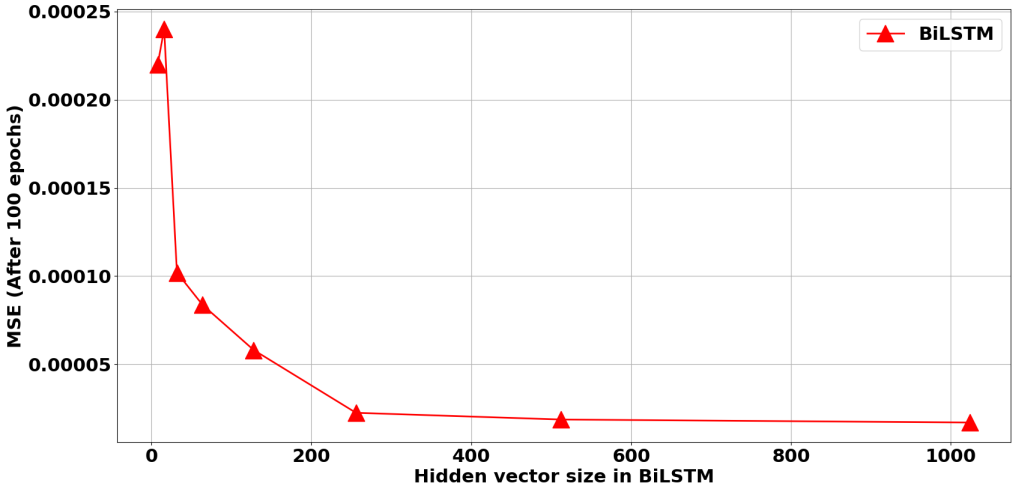

Figure 4.5: BiLSTM cross validation error after 100 epochs with different hidden vector sizes.

4.3 Theoretical formulation

In this section we discuss theoretical formulation of direct learning and contrast it with indirect learning methodology. In Fig. 2.8, $u(n)$ is the input to the DL-DPD model whose output will then be passed to DL-PA model as $x(n)$. The goal of direct learning approach is to learn the DL-DPD from the DL-PA. DL-PA model is a representation of PA behavior function $f : x \rightarrow y$ which accounts for the memory effects. For the purpose of this analysis, we use a linear function approximation for the deep learning model. We assume that we have learned the DL-PA model given by:

$$\hat{y}_{pa}(n) = X^T(n)w_{pa} + e(n) \tag{4.1}$$

$$X^T(n) = [x(n-J), x(n-J+1), ..., x(n))] \tag{4.2}$$

where $\hat{y}_{pa}(n)$ is the response of the DL-PA at time instant n and w_{pa} are the parameters of DL-PA model. We also represent the modeling error as the Gaussian random variable,

$e(n) \in \mathcal{N}(\mu_e, \sigma_e^2)$ which is the difference between the real PA output and the output of DL-PA model. $X(n)$ is a vector containing J lagging samples of the input, which help in effectively modeling memory effects of the PA. Hence, the vector w_{pa} also has dimensions of Jx1. It is straight forward to learn w_{pa} by minimizing the mean squared error between $\hat{y}_{pa}(n)$ and $y(n)$ via gradient descent.

After learning the PA model (DL-PA), we can learn the, DL-DPD model is learned by back-propagating the error (difference between the DL-PA output and desired reference signal) through the DL-PA model in (2.2). Since, the DL-PA expects all the J inputs lagging terms, the equation of the combined system is given by

$$\hat{y}(n) = (\overline{U}^T(n)w_{dpd})^T w_{pa} + e(n) \tag{4.3}$$

$$\overline{U}(n) = [U(n-J), U(n-J+1), ...U(n)] \tag{4.4}$$

where each column in the matrix $\overline{U}(n)$ is a column vector similar to $X(n)$, but including L1 lagging and L2 leading samples of the signal u(n). As a result, U(n) in the above equation is a (L1+L2+1)×J matrix. The vector w_{dpd} represents the DL-DPD model which is to be learned. The inner product of $\overline{U}^T(n)w_{dpd}$ would give us a vector with dimensions of $X(n)$, and is an estimate for the desired input to the PA (after pre-distortion). To solve for w_{dpd} the error function, to be minimized is :

$$C = \min_{w_{dpd}} \frac{1}{N} \sum_{n \in N} ((\overline{U}^T(n)w_{dpd})^T w_{pa} + e(n) - u(n))^2 \tag{4.5}$$

The predominent method used in DL based algorithms to obtain the optimal coefficients is gradient descent. Gradient descent updates the coefficients iteratively by moving in the

negative direction of the gradient of C. The gradient of C w.r.t w_{dpd} as

$$\nabla_{w_{dpd}} C = \frac{2}{N} \sum_{n \in N} ((\overline{U}^T(n) w_{dpd})^T w_{pa} + e(n) - u(n))(\nabla_{w_{dpd}} (w_{dpd}^T \overline{U} w_{pa})) \tag{4.6}$$

$$= \frac{2}{N} \sum_{n \in N} ((\overline{U}^T(n) w_{dpd})^T w_{pa} + e(n) - u(n))(\overline{U} w_{pa})$$

Here the gradient $\nabla_{w_{dpd}} C$ is off from the true gradient by $e(n)\overline{U} w_{pa}$, whose variance is

$$= \sigma_e^2 \sum_{j=0}^{J} u(n-k+j) w_{pa}(j) \tag{4.7}$$

$\forall k \in [L_1, -L_2]$. In direct learning as we have to back propagate through a DL-PA model (w_{pa} in the above formulation), which introduces a source of error with a variance given in (2.9). This error cannot be removed as it arises from modeling of DL-PA model. Our simulations results showed a mean square error (MSE) $4 * 10^{-5}$, in modeling the PA after the completion of training (DL-PA model error). On the other hand, indirect learning method does not require back propagation through a PA model, thereby making this source of error non-existent.

4.4 Comparison between direct and indirect learning

In order to compare the performance of two approaches a BiLSTM model is trained using direct and indirect learning methodologies. The results with different DPD models is listed in Table. 4.2. It can be noticed that the class-AB PA offers better linearity performance compared to class-F^{-1}. It is found that the BiLSTM model trained using direct/indirect learning achieve almost the same performance, with indirect learning being marginally better

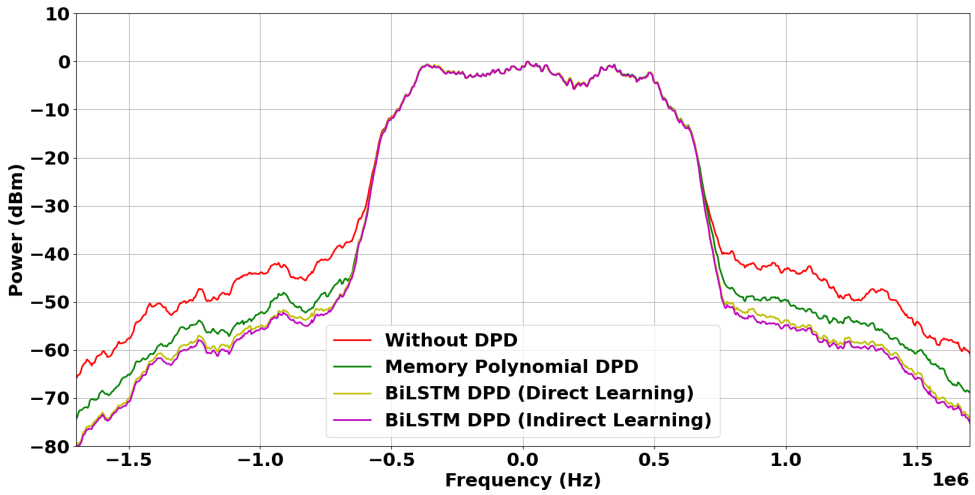

Figure 4.6: Power Spectral Density of class-AB PA, driven by pre-distorted signals from different models.

in terms of ACPR. We present that this small performance variance in direct learning is because of back propagating gradients through the PA behavioral model introduces a source of error, which is responsible for the variation in performance. This is inline with our theoretical analysis of direct learning which gives the variance of these gradients as a function of this error in equation (4.7). This is true for both class-AB and class-F^{-1} PA. Further, training the DL-DPD using direct learning needs more resources as two DL models need to be trained, increasing the time and computational resources required. These results imply that indirect learning is computationally efficient and yields marginally better DPDs than direct learning. This is an interesting result since it was achieved without tuning the BiLSTM model architecture parameters. This is advantageous because this shows that the DL-DPD model's architecture can be used for any PA, without tuning the architecture. This drastically reduces the design and exploration time while designing a DPD. It can be seen from Table 4.2, that BiLSTM always performs better than memory polynomial DPDs, irrespective of the training method used. Moreover to explore the generalization of DL-DPD instead of QAM, QPSK modulated signals are fed into the DL-DPD without retraining. The

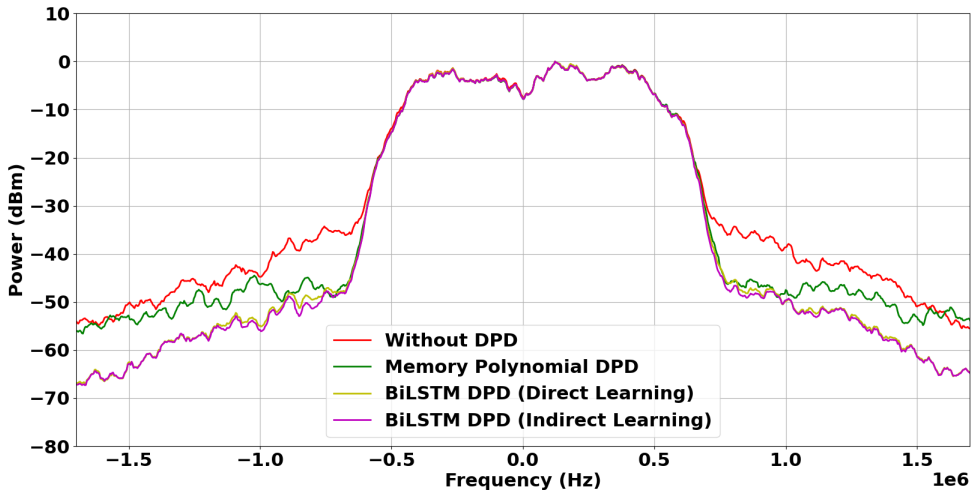

Figure 4.7: Power Spectral Density of class-F^{-1} PA, driven by pre-distorted signals from different models.

Table 4.2: Comparison of ACPR for different DPD models on class-AB and class-F PA for QAM signals.

DPD Model Details	class-AB Lower/Upper ACPR(dB)	class-F^{-1} Lower/Upper ACPR(dB)
Without DPD	45.19/44.53	36.40/30.4
Memory polynomial DPD	55.02/51.08	37.11/40.73
BiLSTM DPD (DL)	56.77/55.67	46.13/41.95
BiLSTM DPD (IL)	57.65/56.41	46.85/41.75

simulation results for QPSK signals showed a 10dB improvement in ACPR. This indicates that the DL-DPD achieves good generalization as it is not trained with QPSK signals.

4.5 Handling input signals of anysize

By comparing different DL-DPDs we arrived at a conclusion that BiLSTM trained with indirect learning outperforms all the other DL-DPDs presented. But BiLSTM needs access to some future inputs and all the inputs we tested in the previous simulations were of length

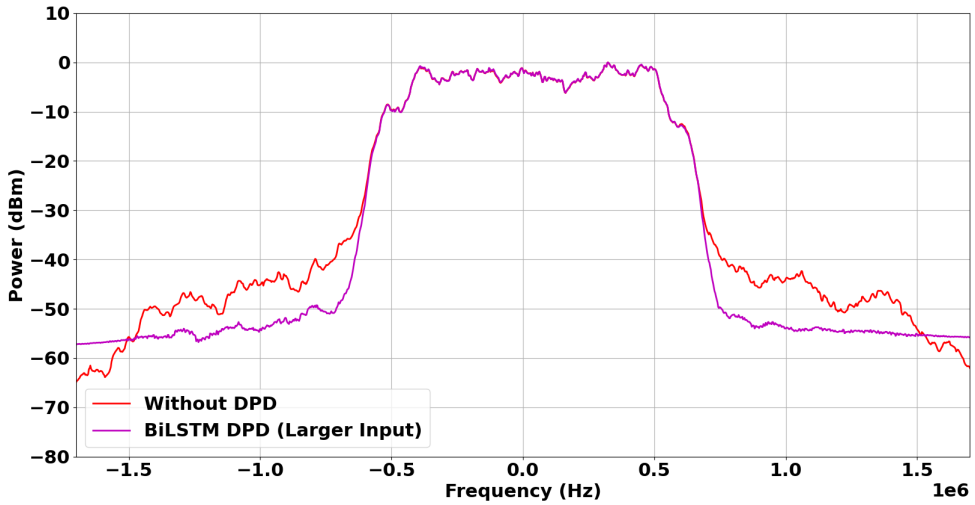

Figure 4.8: PSD of BiLSTM DPD - PA cascade with input of size 7374.

Table 4.3: ACPR of BiLSTM on longer signal than BiLSTM can take in 1 iteration.

DPD Model Details	class-AB Lower/Upper ACPR(dB)
Without DPD	46.91/45.50
BiLSTM DPD	55.8/56.384

5531, which is the size of our BiLSTM. So, when the input is greater than 5531, we pad the input with zeros. By sequentially passing the current inputs and the hidden vectors associated with the past inputs to the new BiLSTM we can handle input signals of any size. In this section we studied the performance of the BiLSTM on inputs of varying lengths. When we tested our BiLSTM with input of length 7374 we found that and still saw an average increase in ACPR of around 11dB. The PSD are presented in Fig. 4.8.

4.6 Frequency domain loss function

To further improve the linearity performance of the PA, we propose to learn from frequency domain loss function. We observe two distinct advantages with frequency domain loss we proposed over the regular loss function, which is computed on spatial data. The first of them is, in spatial domain since the signals are between 0-1, their distortions are small in magnitude Fig 4.9 the magnitude of the loss function would be very low, there by the gradients computed by back-propagating this loss would also be low. This might cause vanishing gradient problem and cause our DL-algorithm to learn nothing. But in frequency domain as the distortions are more significant Fig. 2.3, the magnitude of the errors is more, thereby alleviating the problems of vanishing gradients. The second reason is that, since the signals are evaluated in frequency domain, (ACPR) it is justified to backpropagate the error from frequency domain, rather than time domain. So, these reasons motivate us to explore the idea of learning from the loss in frequency domain.

We choose a BiLSTM based DL-DPD model to train using indirect learning methodology and the frequency domain loss. An overview of the forward pass in the training procedure is present in the Fig. 4.10. After transforming the input signal, it is passed through a kaiser window to get the similar spectrum which amplifies the baseband frequencies. This

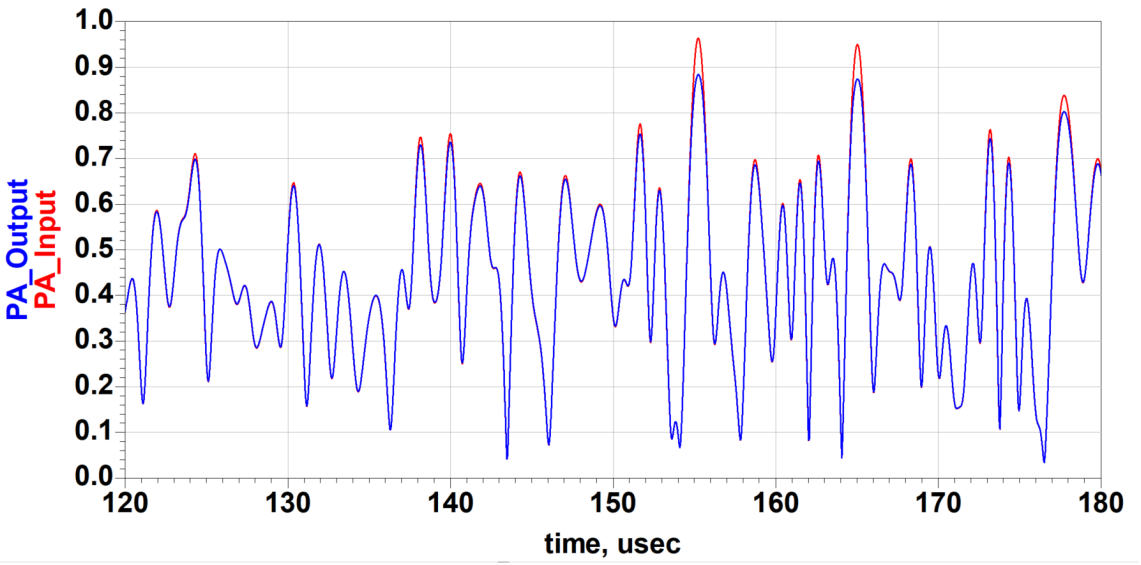

Figure 4.9: The input to the PA and the distorted signal by the PA (Absoulte values) in time domain.

is multiplied with a W matrix which converts the signal to frequency domain. Making the process of converting the signal to frequency domain a simple matrix multiplication, help during the back-propagation of the gradients, as propagating through a multiplication operator is straight forward. Now the loss is computed between this spectrum (PSD_{dpd}) and the PSD of the ground truth (PSD_{gt}). Since after pre-distortion the baseband signal should remain the same, an importance vector is multiplied with the loss. This importance vector has a higher weight (1000) for baseband frequencies and lower importance weights for frequencies around it (5, 1, 0).

$$\mathrm{W} = \begin{bmatrix} 1 & 1 & 1 & ... & 1 \\ 1 & w & w^2 & ... & w^{N-1} \\ .... & & & & \\ 1 & w^{N-1} & w^{2(N-1)} & ... & w^{(N-1)^2} \end{bmatrix}$$

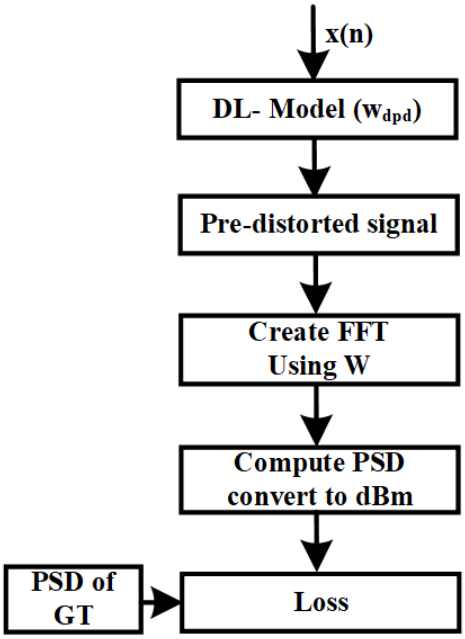

Figure 4.10: Computational flowchart of forward pass while computing the FFT loss.

Here $w = e^{j.2\pi/n}$. The columns of this matrix are orthogonal. The updated frequency domain loss function can be written as

$$J_f = \min_{w_{dpd}} \frac{1}{N} \sum_{N} (imp * (PSD_{dpd} - PSD_{gt}))^2 \tag{4.8}$$

The ACPR achieved by BiLSTM DL-DPD trained with frequency domain loss is much higher than MSE based error in spatial domain Fig 4.11. The respective ACPRs show that training using frequency domain loss gives us approximately 3dB improvement in ACPR with the BiLSTM DL-DPD trained with MSE loss.

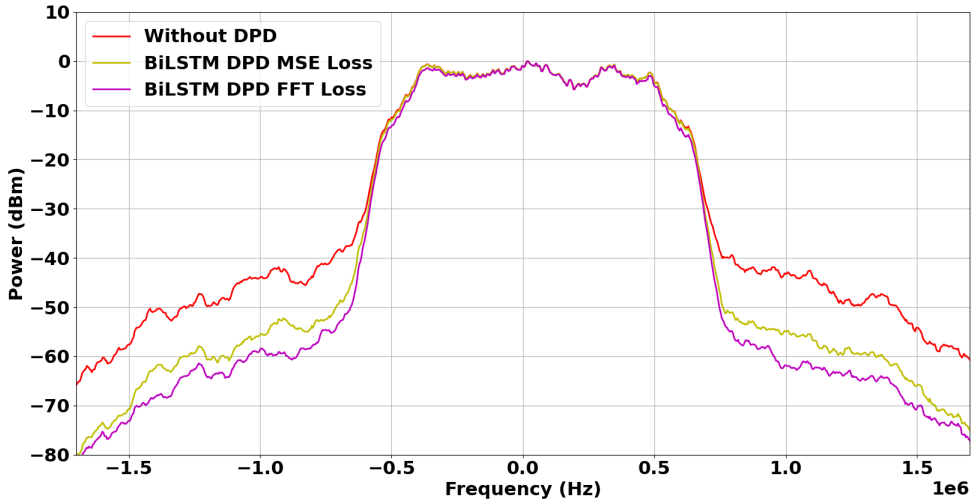

Figure 4.11: Comparisons of BiLSTM DL-DPDs trained with Frequency domain loss and normal MSE in spatial domain.

Table 4.4: ACPR of BiLSTM DL-DPDs trained with Frequency domain loss and normal MSE in spatial domain.

DPD Model Details	class-AB Lower/Upper ACPR(dB)
Without DPD	45.1982/44.53
BiLSTM DPD- MSE Loss	57.65/56.41
BiLSTM DPD- FFT Loss	60.40/63.29

Chapter 5

Conclusion

In this thesis, we compared different deep learning based DPDs and found that BiLSTM based architecture gives best performance for linearizing the PA. We also compared direct and indirect learning methodologies for training deep neural network based DPDs. It is found that DL-DPD trained with indirect learning method is better. We discover that, back propagating through a PA behavioral model in direct learning which adds another source of error. This is in agreement with our theoretical analysis. This analysis can be used as a baseline to extend the comparison between DL-DPDs with different neural network architectures. Further we demonstrated that DL-DPD irrespective of the training methodology used, outperform memory polynomial based DPDs. We showed that even with signals of different lengths our BiLSTM DL-DPD delivers 11dB imporvement in ACPR, which shows that it can handle inputs of any length. We further propose a new loss function in frequency domain, we discuss its advantages and show that it delivers an improvement over the traditional MSE loss in spatial domain by 3dBm.

In furture we plan on conducting our analysis with real experimental data, collected froma real physical PA. Since we might get some measurment noise here, it would be interesting to compare direct and indirect learning methodologies with measurement data. Since the size of FFT matrix in our Frequency domain loss function, it cannot be readily trained with signals of any length. So adapting our frequency domain loss function to signals of any length would be a significant challenge.

Bibliography

[1] A. Ahmed, E.R. Srinidhi, and G. Kompa. Neural network and memory polynomial methodologies for pa modeling. In *TELSIKS 2005 - 2005 uth International Conference on Telecommunication in ModernSatellite, Cable and Broadcasting Services*, volume 2, pages 393–396 vol. 2, 2005. doi: 10.1109/TELSKS.2005.1572135.

[2] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017. doi: 10.1109/ICEngTechnol.2017.8308186.

[3] Slim Boumaiza and Farouk Mkadem. Wideband rf power amplifier predistortion using real-valued time-delay neural networks. In *2009 European Microwave Conference (EuMC)*, pages 1449–1452, 2009. doi: 10.23919/EUMC.2009.5296072.

[4] Slim Boumaiza, Farouk Mkadem, and Morsi Ben Ayed. Digital predistortion challenges in the context of software defined transmitters. In *2011 XXXth URSI General Assembly and Scientific Symposium*, pages 1–4, 2011. doi: 10.1109/URSIGASS.2011.6050519.

[5] R. N. Braithwaite. Digital predistortion of an rf power amplifier using a reduced volterra series model with a memory polynomial estimator. *IEEE Transactions on Microwave Theory and Techniques*, 65(10):3613–3623, 2017. doi: 10.1109/TMTT.2017.2729513.

[6] Lei Ding, G.T. Zhou, D.R. Morgan, Zhengxiang Ma, J.S. Kenney, Jaehyeong Kim, and C.R. Giardina. A robust digital baseband predistorter constructed using memory polynomials. *IEEE Transactions on Communications*, 52(1):159–165, 2004. doi: 10.1109/TCOMM.2003.822188.

[7] Changsoo Eun and E.J. Powers. A new volterra predistorter based on the indirect learning architecture. *IEEE Transactions on Signal Processing*, 45(1):223–227, 1997. doi: 10.1109/78.552219.

[8] Reina Hongyo, Yoshimasa Egashira, Thomas M. Hone, and Keiichi Yamaguchi. Deep neural network-based digital predistorter for doherty power amplifiers. *IEEE Microwave and Wireless Components Letters*, 29(2):146–148, 2019. doi: 10.1109/LMWC.2018.2888955.

[9] Xin Hu, Zhijun Liu, Xiaofei Yu, Yulong Zhao, Wenhua Chen, Biao Hu, Xuekun Du, Xiang Li, Mohamed Helaoui, Weidong Wang, and Fadhel M. Ghannouchi. Convolutional neural network for behavioral modeling and predistortion of wideband power amplifiers. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2021. doi: 10.1109/TNNLS.2021.3054867.

[10] Praveen Jaraut, Abubaker Abdelhafiz, Hanen Chenini, Xin Hu, Mohamed Helaoui, Meenakshi Rawat, Wenhua Chen, Noureddine Boulejfen, and Fadhel M. Ghannouchi. Augmented convolutional neural network for behavioral modeling and digital predistortion of concurrent multiband power amplifiers. *IEEE Transactions on Microwave Theory and Techniques*, 69(9):4142–4156, 2021. doi: 10.1109/TMTT.2021.3075689.

[11] J. Kim and K. Konstantinou. Digital predistortion of wideband signals based on power amplifier model with memory. *Electronics Letters*, 37(23):1–2, Nov 08 2001. URL http://login.ezproxy.lib.vt.edu/login?url=https://www.proquest.com/scholarly-journals/digital-predistortion-wideband-signals-based-on/docview/1622033198/se-2?accountid=14826. Copyright - Copyright The Institution of Engineering Technology Nov 8, 2001; Document feature - Equations; Graphs; ; Last updated - 2014-11-10; CODEN - ELLEAK.

[12] Han Le Duc, Bruno Feuvrie, Matthieu Pastore, and Yide Wang. An adaptive cascaded ila- and dla-based digital predistorter for linearizing an rf power amplifier. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(3):1031–1041, 2019. doi: 10.1109/TCSI.2018.2872465.

[13] Farouk Mkadem and Slim Boumaiza. Physically inspired neural network model for rf power amplifier behavioral modeling and digital predistortion. *IEEE Transactions on Microwave Theory and Techniques*, 59(4):913–923, 2011. doi: 10.1109/TMTT.2010.2098041.

[14] Nima Mohammadi, Jianan Bai, Qiang Fan, Yifei Song, Yang Yi, and Lingjia Liu. Differential privacy meets federated learning under communication constraints. *arXiv preprint arXiv:2101.12240*, 2021.

[15] D.R. Morgan, Zhengxiang Ma, and Lei Ding. Reducing measurement noise effects in digital predistortion of rf power amplifiers. volume 4, pages 2436 – 2439 vol.4, 06 2003. ISBN 0-7803-7802-4. doi: 10.1109/ICC.2003.1204371.

[16] D.R. Morgan, Z. Ma, J. Kim, M.G. Zierdt, and J. Pastalan. A generalized memory polynomial model for digital predistortion of rf power amplifiers. *IEEE Transactions on Signal Processing*, 54(10):3852–3860, 2006. doi: 10.1109/TSP.2006.879264.

[17] Henna Paaso and Aarne Mammela. Comparison of direct learning and indirect learning predistortion architectures. In *2008 IEEE International Symposium on Wireless Communication Systems*, pages 309–313, 2008. doi: 10.1109/ISWCS.2008.4726067.

[18] Javad Sabzehali, Vijay K. Shah, Harpreet S. Dhillon, and Jeffrey H. Reed. 3d placement and orientation of mmwave-based uavs for guaranteed los coverage. *IEEE Wireless Communications Letters*, 10(8):1662–1666, 2021. doi: 10.1109/LWC.2021.3076463.

[19] Javad Sabzehali, Vijay K Shah, Qiang Fan, Biplav Choudhury, Lingjia Liu, and Jeffrey H Reed. Optimizing number, placement, and backhaul connectivity of multi-uav networks. *arXiv preprint arXiv:2111.05457*, 2021.

[20] Gurpreet Singh and Manoj Sachan. Multi-layer perceptron (mlp) neural network technique for offline handwritten gurmukhi character recognition. In *2014 IEEE International Conference on Computational Intelligence and Computing Research*, pages 1–5, 2014. doi: 10.1109/ICCIC.2014.7238334.

[21] Jinlong Sun, Wenjuan Shi, Zhutian Yang, Jie Yang, and Guan Gui. Behavioral modeling and linearization of wideband rf power amplifiers using bilstm networks for 5g wireless systems. *IEEE Transactions on Vehicular Technology*, 68(11):10348–10356, 2019. doi: 10.1109/TVT.2019.2925562.

[22] Chance Tarver, Liwen Jiang, Aryan Sefidi, and Joseph R. Cavallaro. Neural network dpd via backpropagation through a neural network model of the pa. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 358–362, 2019. doi: 10.1109/IEEECONF44664.2019.9048910.

[23] Yibo Wu, Ulf Gustavsson, Alexandre Graell i Amat, and Henk Wymeersch. Residual neural networks for digital predistortion. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 01–06, 2020. doi: 10.1109/GLOBECOM42002.2020.9322327.

[24] Yikang Zhang, Yue Li, Falin Liu, and Anding Zhu. Vector decomposition based time-delay neural network behavioral model for digital predistortion of rf power amplifiers. *IEEE Access*, 7:91559–91568, 2019. doi: 10.1109/ACCESS.2019.2927875.

[25] Honghao Zheng, Nima Mohammadi, Kangjun Bai, and Yang Yi. Low-power analog and mixed-signal ic design of multiplexing neural encoder in neuromorphic computing.

In *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pages 154–159, 2021. doi: 10.1109/ISQED51717.2021.9424267.

[26] Dayong Zhou and Victor E. DeBrunner. Novel adaptive nonlinear predistorters based on the direct learning algorithm. *IEEE Transactions on Signal Processing*, 55(1):120–133, 2007. doi: 10.1109/TSP.2006.882058.

[27] Zhou Zhou, Kangjun Bai, Nima Mohammadi, Yang Yi, and Lingjia Liu. Making intelligent reflecting surfaces more intelligent: A roadmap through reservoir computing. *arXiv preprint arXiv:2102.03688*, 2021.