NEURONAL NETWORK IMPLEMENTATION IN PYTHON FOR HIGHLY NON-LINEAR WIRELESS SYSTEMS

IMPLEMENTACIÓN DE RED NEURONAL EN PYTHON PARA SISTEMAS INALÁMBRICOS ALTAMENTE NO LINEALES

Daniel Santiago Águila Torres
Tecnológico Nacional de México / IT de Tijuana, México
daniel.aguila16@tectijuana.edu.mx

José Ricardo Cárdenas Valdez
Tecnológico Nacional de México / IT de Tijuana, México
jose.cardenas@tectijuana.edu.mx

Carlos Hurtado Sánchez
Tecnológico Nacional de México / IT de Tijuana, México
carlos.hurtado@tectijuana.edu.mx

Manuel de Jesús García Ortega
Tecnológico Nacional de México / IT de Tijuana, México
manuel.garciao@tectijuana.edu.mx

Recepción: 28/octubre/2021 **Aceptación:** 26/enero/2022

Abstract

In this work, a system based on a neural network implemented in an open-source programming language is developed in order to operate simple band systems as an alternative to other modeling platforms that require the use of a software license. An artificial neural network implementation in Python language is carried out with multilayer perceptrons, one-dimensional convolutional neural networks, short and long-term memory networks and transformers. In the development of this work, a precision of -17.5249 dB NMSE was obtained by a one-dimentional convolutional neural network for highly non-linear behavior. The objective of developing open-source proposals is to use a free hardware development board that applies to specific wireless systems. The modeling system represents the preliminary stage for linearization and spectral correction processes in the case of RF transmission.

Keywords: linearization, modeling, neural networks, wireless transmissions.

Resumen

En este trabajo se desarrolla un sistema basado en una red neuronal implementada en un lenguaje de programación de código abierto con el fin de operar sistemas en banda simple como una alternativa a otras plataformas de modelado que requieren del uso de licencia de software. Se realiza una implementación de redes neuronales artificiales en el lenguaje Python con las arquitecturas de perceptrón multicapa, redes neuronales convolucionales de una dimensión, redes de memoria a corto y largo plazo y transformers. En el desarrollo de este trabajo se obtuvo una precisión de la implementación de la red neuronal convolucional de una dimensión de -17.5249 dB NMSE para un comportamiento altamente no lineal. El objetivo de desarrollar propuestas de código abierto es utilizar tarjetas de desarrollo de hardware libre que apliquen para ciertos sistemas inalámbricos. El sistema de modelado representa la etapa previa para procesos de linealización y corrección espectral en caso de una transmisión de RF.

Palabras Clave: *Linealización, modelado, redes neuronales, transmisiones inalámbricas.*

1. Introduction

Telecommunication systems are used to connect devices across different locations and provide communication services to people who require them. In recent years there has been special interest in the development of wireless communication technology. Some efforts to improve the current connectivity situation in México include the proposal of the plan nacional de desarrollo (national plan of development) 2019-2024 to provide wireless Internet coverage to the whole country, in order to confront margination, poverty and integrate zones with lower productivity rates [Secretaría de Agricultura y Desarrollo Rural, 2019].

Due to the high demand for data transmission, the radio frequency (RF) spectrum availability is becoming increasingly scarce. Modern communication systems address this issue with spectrum-efficient modulated signals that optimize bandwidth usage. The main digital modulation techniques used in wireless communication are,

amplitude shift keying (ASK), frequency-shift keying (FSK), phase-shift keying (PSK), and quadrature amplitude modulation (QAM) [Kishore, 2019].

Multiplexing techniques are used in conjunction with modulation techniques because they help to put more signals or information into a given bandwidth. Broadband schemes such as spread spectrum or code division multiple access (CDMA) and orthogonal frequency-division multiplexing (OFDM) allow many radios to share a single bandwidth [Frenzel, 2014]. Such complex modulation requires linear amplification over a wide frequency range to ensure no amplitude, frequency, or phase distortion. However, these transmission schemes have a high peak to average power ratio (PAPR), which possesses adverse effects on the efficiency and linearity of radio frequency (RF) power amplifiers (PA) [Raffo, 2010].

The RF PA is an element in the transmit signal chain that increases the power of the signal so it reaches the antenna, however, when operating in the saturation region, it introduces undesired effects on the output signal, such as intermodulation products (IMD), memory effects, spectral regrowth and interference on adjacent channels. A technique used to compensate for this nonlinearity in RF PA is DPD. The main idea of DPD is to extract the inverse behavioral model in the digital domain for the nonlinear RF PA, then cascade the predistorter in the forward baseband. Consequently, the cascade of the nonlinear inverse model and RF PA will be a linear system [Zhang, 2019]. RF PA linearization and precise modeling are the best techniques to correct these undesired effects.

There exists research about digital predistortion (DPD) linearization modeling based on cloud platforms using Python [Su, 2019]. The described approach works on a multiple PA concurrent system and uses radial basis function (RBF) neural networks for PA behavioral modeling. The method provides unified, centralized management for the deployment of hardware for data acquisition and training models. The work [Zhang, 2019] utilizes Python for digital predistortion behavioral modeling of RF PA with a vector decomposed time-delay neural network (VDTDNN), which achieves better linearization results when compared with other neural network-based models. This work develops a neural network implementation using four architectures with the open-source programming language Python as an alternative to other proposals

developed in programming languages such as C language with the intention to implement it on a digital predistortion chain. Python was chosen because of the main benefits of code portability, legibility, and the flexibility of the language, as well as the vast amount packages it supports, including those in the category of scientific computation.

2. Methods

The nonlinear characteristics of the PA exhibit and amplification process firstly ni a linear region and other with saturation. These distortions are significant when the signal peaks approach the PAs saturation region, therefore the PA need to be driven at a lesser average power level [Rawat, 2020]. There exists a compromise between the linearity and power efficiency of the PA, where it is difficult to operate over the saturation point because it introduces nonlinear distortions on the output signal.

A behavioral approach can be used to model power amplifiers by taking measurements and using them to create the amplification process, in this case plots are generated using the output voltages of a nonlinear behavior [Arabi, 2008]. An example is shown in figure 1 of an amplification process with highly nonlinear behavior.

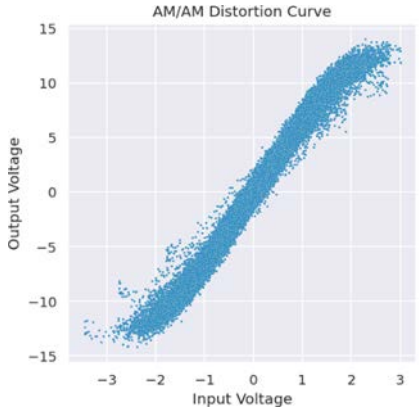

Figure 1 Plot of an amplification process of a highly nonlinear system.

Because of the PA characteristics in terms of gain, the signal gets distorted and produces a transmission error. The in-band error can be measured with the

normalized mean squared error (NMSE) [Rawat, 2020] given in decibels by equation 1.

$$\mathrm{NMSE} \triangleq 10 \log_{10}\left(\frac{\sum_{n=1}^{N}|y_I(n)-y(n)|^2}{\sum_{n=1}^{N}|y_I(n)|^2}\right) \quad (1)$$

Where:

$y_I(n)$: Ideal or transmitted signal.

$y(n)$: Output signal.

The Volterra model is able to represent the behavior of power amplifiers. Volterra series represent a combination of linear convolution and a power series; they can be used to describe the input/output relationship of a general nonlinear, causal, and time-invariant system with fading memory. However, its main disadvantage is that the number of parameters grows drastically in relation of the nonlinear order and memory length. This model is expressed mathematically as the equation 2.

$$\mathrm{y(n)} = \sum_{k}\sum_{l_1}\dots\sum_{l_{2k+1}} h_{2k+1}(l_1,l_2,\dots,l_{2k+1})^* \prod_{i=1}^{k+1} x\,(n-\tau_i) \prod_{i=k+2}^{2k+1} x^*(n-\tau_i)d\tau_{2k+1} \quad (2)$$

Where:

$x(n)$: Input complex baseband signal.

$y(n)$: Output complex baseband signal.

$(\ )^*$: Denotes complex conjugation (hermitinan).

This equations complexity increases exponentially so its implementation results impractical for communication systems. The memory polynomial model (MPM) is a subset of the Volterra series comprised of derivation delays and static nonlinear functions; it represents a truncation of the general Volterra series that only considers diagonal terms. This results in significantly less parameters compared to the original series [Arabi, 2008]. The MPM considers both nonlinearity and memory effects present in power amplifiers and is given by the equation 3. The least square error (LSE) method is used to compute the a coefficients for a given memory depth Q and polynomial order K.

$$y(n) = \sum_{q=0}^{Q} \sum_{k=1}^{K} a_{2k-1,q} |x(n-q)|^{2(k-1)} x(n-q) \quad (3)$$

Where:

$x(n)$: Input complex baseband signal.

$y(n)$: Output complex baseband signal.

Q: Memory depth.

K: Polynomial order.

$a_{2k-1,q}$: Complex coefficients.

Another approach to PA behavioral modeling is the use of artificial neural networks (ANN). ANN is a model inspired by nature that processes information in a way analogous to how neurons function in a biological brain. The most basic architecture of a neural network is the perceptron. A single layer network, also called perceptron computes a linear combination of the inputs (with a possible bias term) called the net input. Then a possibly nonlinear activation function is applied to produce the net output. An activation function usually maps any real input into a usually bounded range, commonly used 0 to 1 or -1 to 1 [Warren, 1994]. A perceptron with a linear activation function could be considered a linear model. The output of the perceptron is given by the equation 4.

$$p_i = act\left(a_i + \sum_{j=0}^{n_x} b_{ij} x_i\right) \quad (4)$$

Where:

p_i: Predicted output.

x_j: Independent variable (input).

a_i: Bias for output layer.

b_{ij}: Weight form input to output layer.

More general functions can be constructed considering networks having successive layers of processing units, with connections running from every unit in one layer to

every unit in the next layer, but with no other connections permitted. This restriction makes the network feed-forward, so it does not contain any feedback loops.

Such layered networks are easier to analyze and implement in software efficiently than other more general topologies.

The units in the network which are not treated as output units are called hidden units [Bishop, 1995]. A multilayer neural network also referred as a multilayer perceptron (MLP) introduces a hidden layer of neurons with nonlinear activation functions and includes estimated weights between the inputs and the hidden layer. MLP are general purpose nonlinear models that given enough hidden neurons and data can approximate virtually any function to any degree of precision, for this reason they are known as universal approximators [Warren, 1994].

This works uses a MLP to model a highly nonlinear RF PA; different number of layers and neurons were tested. The diagram in figure 2 shows the amplification of a signal using this model.

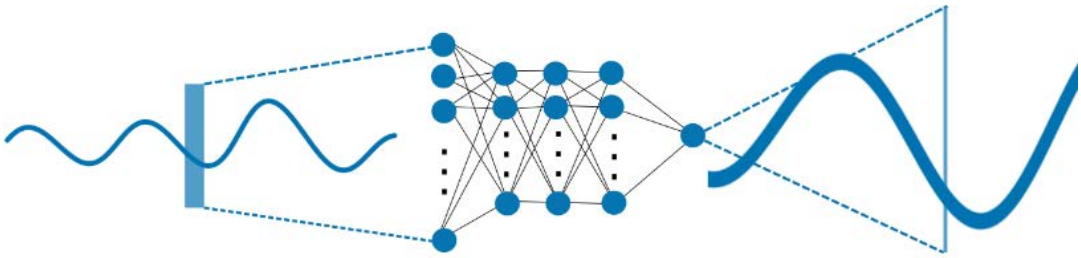

Figure 2 MLP diagram.

There are many ANN topologies that differ in the way data is processed over the network, for instance recurrent neural networks (RNN) use feedback connections to store representations of recent input events in the form of activations (short term memory, as opposed to long-term memory embodied by slowly changing weights) [Hochreiter,1997]. However, the error signals flowing backward in time tend to blow up or vanish, which may lead to oscillating weights or being unable to learn to bridge long time lags. Long short-term memory (LSTM) is a recurrent network designed to overcome these error backflow problems, using self-connected memory cells with gates that control the flow of information. A multiplicative input gate unit is introduced

to protect the memory contents from perturbation by irrelevant inputs and a multiplicative output gate is introduced to protect other units from perturbation by currently irrelevant memory contents.

The error signal goes through the memory cell, and so the output gate will have to learn which errors to trap within the memory cell and the input gate will have to learn when to release errors and appropriate scale them [Hochreiter,1997]. The diagram in figure 3 represents the LSTM implementation used in this work; different numbers of cells were tested.

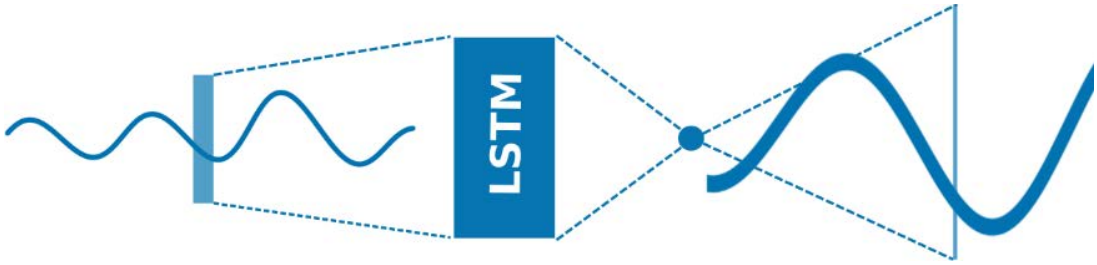

Figure 3 LSTM diagram.

2D convolutional neural networks (2D CNN) are a particular type of neural network where each neuron contains 2-D planes for weights, which is known as the kernel, and input and output which is known as the feature map. Each neuron performs a linear convolution between the image and corresponding filter to generate the input feature map of the neuron. Then the input feature map is passed through the activation function to generate the output feature map of the neuron of the convolution neuron. This process repeats until the scalar outputs are forward-propagated through the fully - connected and output layers to produce the final output. An alternative of the 2D CNN called 1D convolutional neural network (1D CNN) has been recently adopted for real-time and low-cost applications due to its lower computational complexity. The main difference between 1D and 2D CNNs, is that 1D arrays replace 2D matrices for both kernels and feature maps [Kiranyaz, 2021]. This work implements a 1D CNN which performs convolutions on the input signal; different number of layers and filters were tested with a static convolution window of 2. The diagram in figure 4 depicts the model.

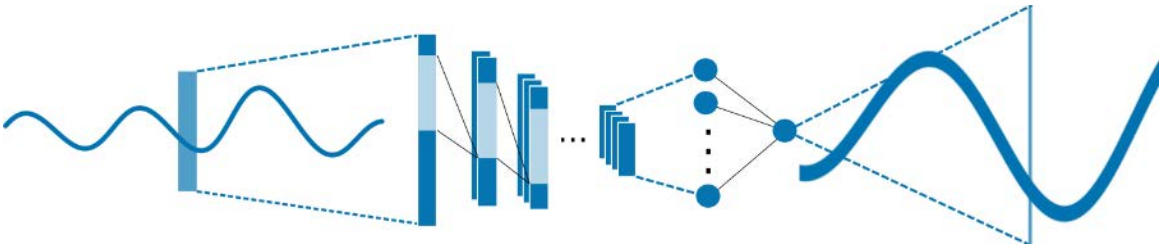

Figure 4 1D CNN diagram.

Transformers neural networks are an architecture that prevents recurrence and instead relies entirely on an attention mechanism to establish global dependencies between input and output [Vaswani, 2017]. Unlike other sequence models, the transformer architecture does the appropriate because this model has no recurrence or convolutions. The positional information of the data must also be fed as an input. One way to do this is to encode the positional information of the data with a function and add the encoded position to the input data. The functions defined in equation 5 and equation 6 are used to encode positions using different frequencies.

$$PE_{(pos,2i)} = \sin \frac{pos}{1000^{\frac{2i}{d_{model}}}} \tag{5}$$

$$PE_{(pos,2i+1)} = \cos \frac{pos}{1000^{\frac{2i}{d_{model}}}} \tag{6}$$

Where:

pos: Position.

i: Dimension.

That is, each method of position codification corresponds to a sinusoid. Wavelengths form a geometric progression from 2π to $(1000)\,(2\pi)$. This method was chosen to encode the positional data in the transformer implementation used in this work. Additionally, transformers require inputs to be encoded, so a binary vector was used with the floating-point representation of each data point in the input data. An illustration of this model is shown in figure 5.

The MLP, CNN, LSTM, and transformer architectures were implemented in this work using Python, Keras, and Tensorflow as the backend. The NMSE defined in equation

1 was used as the loss function for training. A sliding time window with six values of the input signal was used in the ANN experiments to predict the value of the amplified output signal. An illustration of this sliding window is shown in figure 6.

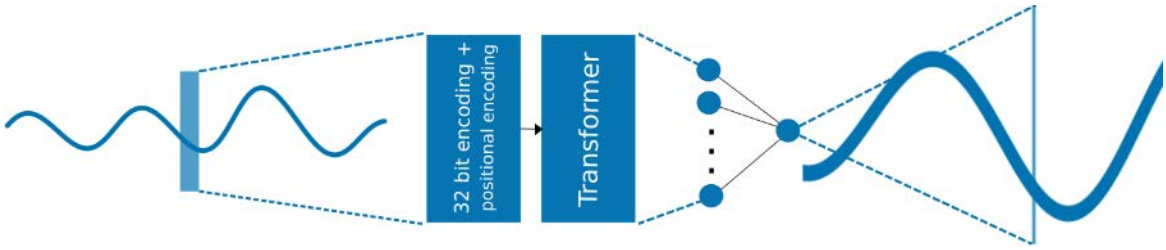

Figure 5 Transformer diagram.

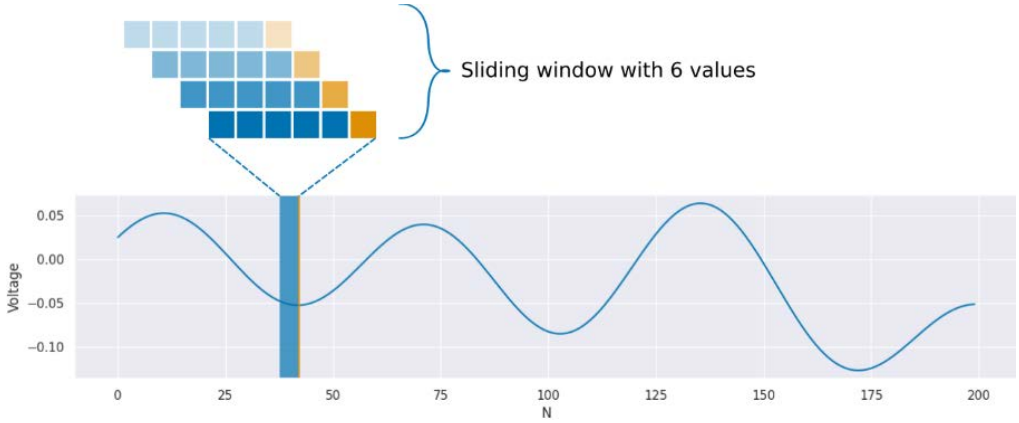

Figure 6 Sliding window of 6 values used for the ANN models.

The Adam optimizer was used with the default learning rate of 0.001. The models were trained in batches of 1024 values with 600 epoch and early stopping with 5 patience. Both ReLu and ELU activation functions were tested in these experiments with no activation function at the output. These results were saved on a CSV file and a 32 bit hash was used as an identifier for each model.

3. Results

The best results for the ANN experiments were obtained from a CNN with five layers; the NMSE obtained for this model was -17.5249 dB. The table 1 displays the best results for each ANN architecture and figure 7 shows the plots of the amplification process for these results.

Table 1 Best result of the ANN experiments by architecture.

Architecture	Description	Parameters	Bytes	NMSE
CNN	5 layers 2, 2, 4, 4 filters per layer	81	648 B	-17.5249 dB
LSTM	4 cells	101	808 B	-17.3609 dB
Transformer	2 attention heads	10705	85640 B	-17.0594 dB
MLP	4 layers 4, 4, 4, 1 neurons per layer	53	424 B	-11.6476 dB

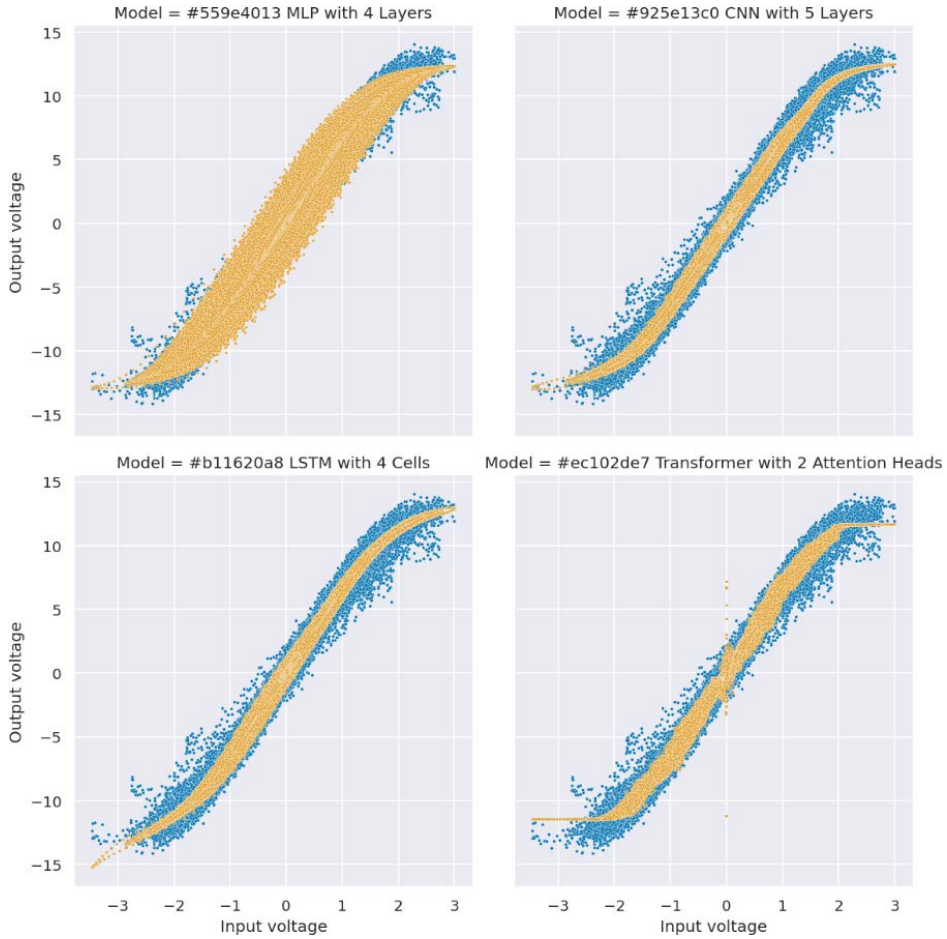

Figure 7 Amplification process of a highly nonlinear behavior of the ANN experiments.

On average, the ANN experiments with the transformer architecture had the lowest NMSE, however they also have the highest parameter count and therefore the biggest size in bytes. The MLP model had the worst performance. The table 2 shows the average results with variability (standard deviation) of the ANN experiments grouped by architecture.

Table 2 Average results with variability of the ANN experiments by architecture.

Architecture	Parameters	Bytes of memory	NMSE
Transformer	5027 ± 4145	40216 ± 33164 B	-15.4790 ± 1.1789 dB
LSTM	182 ± 181	1456 ± 1454 B	-15.4264 ± 2.3190 dB
CNN	323 ± 314	2584 ± 2519 B	-14.4863 ± 4.1276 dB
MLP	168 ± 115	1344 ± 920 B	-10.8836 ± 0.8044 dB

The NMSE in dB of each model for the amplification process curve was plotted in figure 8. From this plot it is understood that the CNN has a lower error in comparison to the MLP, LSTM and transformer models experimented in this work.

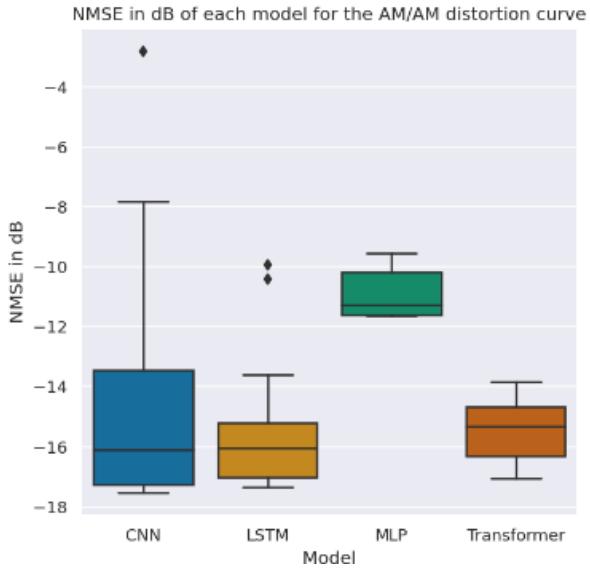

Figure 8 NMSE in dB of each model.

4. Discussion

After observing the results of the Python implementation of the models, it is possible to notice they are close to what was expected. The ANN experiments used a comparable amount of parameters to a MPM with the exception of the transformer architecture. The experiments ran on a Google Colab notebook and took 1 hour and 54 minutes to train all the ANN with 65,536 training data points each. The execution time of the model can change depending on hardware, and a future improvement could be implementing this on a DSP development board for a DPD chain and compare the speed and performance with other implementations.

5. Conclusions

The best performing ANN was a CNN with 5 layers which achieved a -17.5249 dB NMSE and the second to best was a LSTM with 4 cells and a -17.3609 dB NMSE for the amplification process of a highly nonlinear behavior. Although, the LSTM had a lower NMSE on average when compared to the CNN. These results were expected since these types of neural networks excel at processing sequences and are able to model the memory effects present in RF PAs. The transformer achieved a -17.0594 dB NMSE, but it also was the model with the highest parameter count. This is likely due to having more inputs than the other models since they need to be encoded in a binary vector and also need positional encoding. Transformers are proficient at processing sequence data; it was likely for these results to be similar to those of the LSTM and CNN. Finally the worst performing ANN was the MLP with a NMSE of -11.7476 dB. In general, artificial neural network models were able to model the behavior of RF PA in its dynamic range but presented problems when modeling its nonlinear regions.

6. Bibliography and References

[1] Arabi, E. and Ali S., (2008). Behavioral Modeling of RF front end devices in Simulink®, Master Thesis, Chalmers University of Technology, SE-412 96 Göteborg, Sweeden, pp. 1-3, 21-26, 47.

[2] Bishop, Christopher, M., (1995). Neural Networks for Pattern Recognition. Clarendon Press, Oxford University Press.

[3] Frenzel, Louis, E., (2014). Principles of Electronic Communication Systems. Fourth edition, McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc, pp. 272, 590.

[4] Hochreiter, S., and Schmidhuber, J., (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780.

[5] Kishore, G. Shyam, and Hemalatha, Rallapalli, (2019). Performance Assessment of M-Ary ASK, FSK, PSK, QAM and FQAM in AWGN Channel. 2019 International Conference on Communication and Signal Processing (ICCSP), IEEE, pp. 0273–77.

[6] Kiranyaz, Serkan, (2021). 1D Convolutional Neural Networks and Applications: A Survey. Mechanical Systems and Signal Processing, vol. 151, p. 107398.

[7] Raffo A., (2010). Behavioral Modeling of RF front end devices in IEEE Transactions on Microwave Theory and Techniques, Chalmers University of Technology, SE-412 96 Göteborg, Sweeden, 58(4), pp. 710-718, 47.

[8] Rawat, Karun, (2020). Introduction to RF Power Amplifier Design and Architecture. Bandwidth and Efficiency Enhancement in Radio Frequency Power Amplifiers for Wireless Transmitters, by Karun Rawat et al., Springer International Publishing, pp. 1, 7, 76, 85, 340.

[9] Secretaría de Agricultura y Desarrollo Rural, (2019). Plan Nacional de Desarrollo 2019-2024, pp. 61.

[10] Su, Ri-na, (2019). Digital Pre-Distortion Linearization Modeling Based on Cloud Platform. DEStech Transactions on Computer Science and Engineering, no. msota.

[11] Vaswani, Ashish, (2017). Attention Is All You Need.

[12] Warren, S. Sarle, (1994). Neural Networks and Statistical Models.

[13] Zhang, Yikang, (2019). Vector Decomposition Based Time-Delay Neural Network Behavioral Model for Digital Predistortion of RF Power Amplifiers. IEEE Access, vol. 7, pp. 91559–68.