Memory Polynomial-Inspired Neural Network for Modeling the Power Amplifier Non-linearities

Pedamalli Saikrishna, Ashok Kumar Reddy Chavva, and Boddapati Hari krishna

***Abstract*—In this paper, we propose a neural network (NN) based algorithm for modeling the power amplifier non-linearities. Here, we studied a novel NN architecture inspired by Volterra series, which is widely used for modeling the power amplifier characteristics. The proposed method has separate sub-networks for each memory tap. This helps in having flexible number of output nodes for the sub-network based on the chronology of samples. The proposed architecture's performance is measured in terms of normalized mean square error (NMSE) and adjacent channel error power ratio (ACEPR). The proposed method requires ≈ 1100 real multiplications to predict the PA output from the input.**

***Keywords*—Adjacent channel error power ratio, deep learning, PA modeling, normalized mean square error, Volterra series.**

I. Introduction

In wireless communication systems, we use a power amplifier (PA) to boost the signal power level before transmitting the signal over the air. This helps us compensate for the pathloss experienced by the wireless devices. With the increased data rate requirements in wireless systems, power amplifier efficiency has become very critical. The power amplifiers exhibit non-linearity when operated at higher power. This non-linearity decreases the signal quality and also causes adjacent channel interference.

In this paper, we propose a novel deep learning (DL) method to model the PA non linearity, whose architecture is inspired by MP methods. Our main contributions are as follows.

- We proposed a novel architecture that is designed based on proven conventional mathematical models, resulting in reduced parameter requirement and lower complexity.
- With separate sub-networks for each memory tap making output nodes of the sub-network flexible to chronology of the samples.

II. Proposed Neural Network Architecture and Training Mechanism

A. Modeling Memoryless PA

In case of memoryless nonlinear PA, the PA output $y(n)$ depends only on baseband input $x(n)$ and its magnitude powers as,

$$y(n) = \sum_{k=0}^{K} \theta_k x(n)|x(n)|^k, \tag{1}$$

Beyond 5G Team, Mobile Communications R&D, Samsung R&D Institute Bangalore, India
Email: {saikrishna.p, ashok.chavva, harikrish.b}@samsung.com

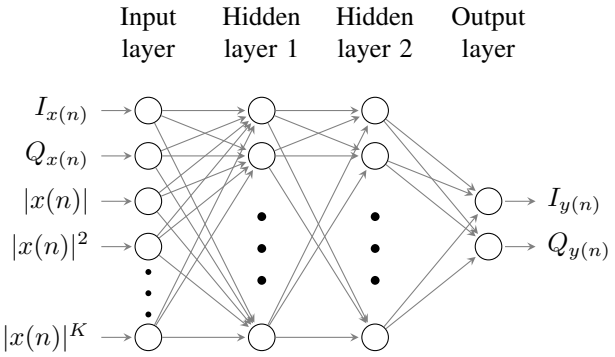

Figure 1: NN for modeling memoryless nonlinear PA

where K represents the highest order of non-linearity.

It can be seen from (1) that $y(n)$ is a nonlinear function of $x(n)$ and can be modeled as a linear combination of $x(n)|x(n)|^k$. Even though we can learn the function that maps $x(n)$ to $y(n)$ using a NN, the complexity of such NN will be quite high because, we need more parameters to learn complex nonlinear functions. This results in more hidden units and layers, thereby increasing the computational complexity. Instead, if we try to learn the function that maps $[x(n), |x(n)|, ..., |x(n)|^k]$ to $y(n)$ using a NN, we can achieve this with a network whose complexity is much lower. So, we can design the NN as shown in Fig 1. The complex valued $x(n)$ is separated into in-phase and quadrature-phase components so that we can use real valued NN.

B. Modeling PA with Non-linear Memory

The PAs used in practical purposes contains some time delay circuits as well such as bias line circuits with capacitors. This results in affecting the output of PA based on the past samples $x(n-m)$. Hence, the PA model used for practical purposes is also dependent on the past samples. One of the most widely used model to design the PA non linearity with memory is given by MP model [1]. According to this, the PA output $y(n)$ is given by

$$y(n) = \sum_{k=0}^{K} \sum_{m=0}^{M} \theta_{km} x(n-m)|x(n-m)|^k, \tag{2}$$

where M denotes the memory length.

Now (2) can be realized as a linear combination of (1) over multiple memory taps. Inspired by MP, we design the neural network with separate densely connected layers for each of

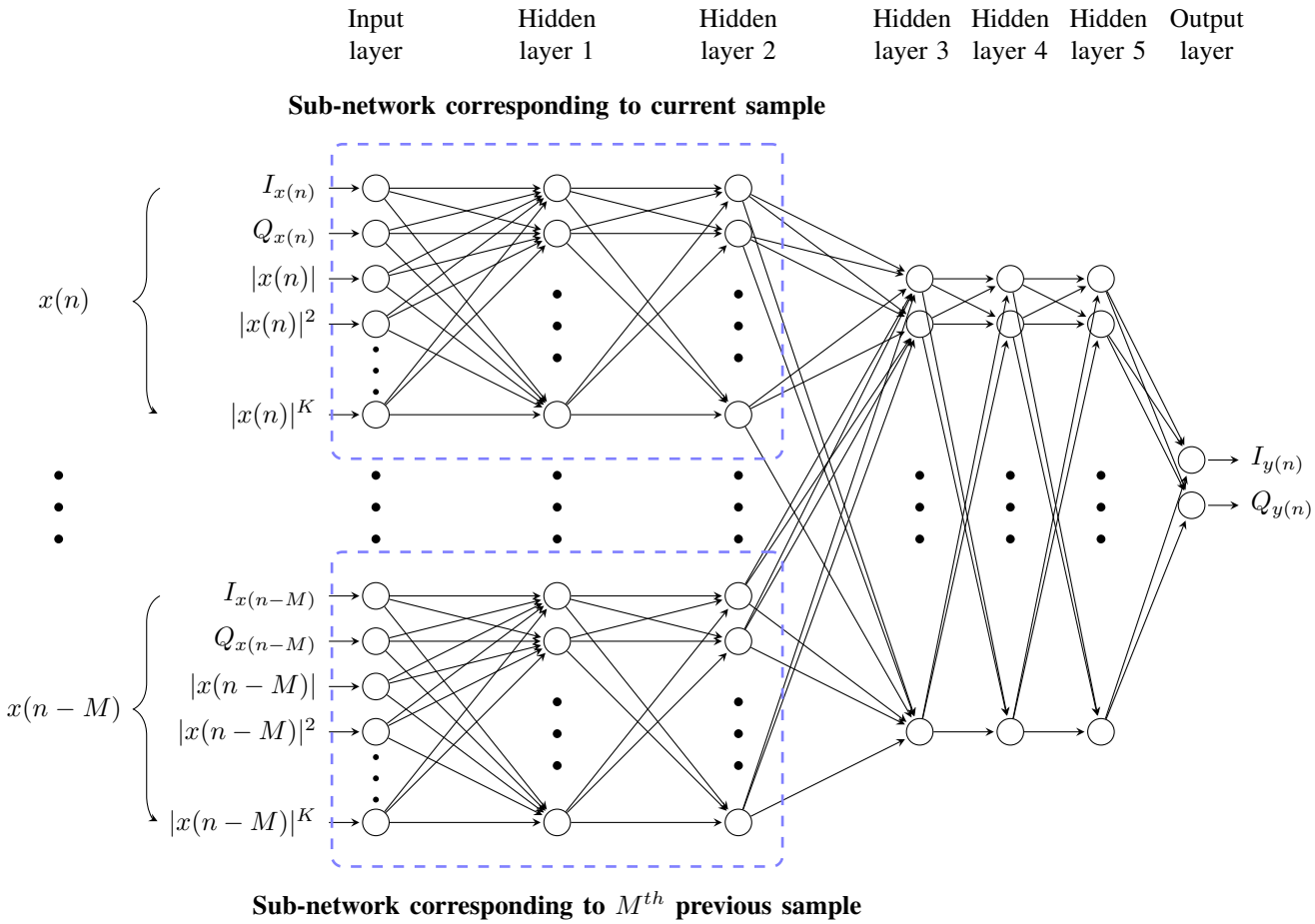

Figure 2: Proposed NN for modeling PA non-linearities with non-linear memory

the memory tap $(x(n), x(n-1), ..., x(n-m))$. So, for each of the memory tap we can have a sub-network as shown in Fig 1 without the output layer. The outputs of hidden layer 2 from all the memory taps are then concatenated and fed to the hidden layer 3 which is then followed by hidden layers 4 and 5 and the output layer, which are fully connected. The hidden layer 3, 4, 5 and output layer can then help in capturing the effect of nonlinear memory. The complete architecture of the proposed method is shown in Fig 2.

The number of nodes in hidden layers 1 and 2 is given by the sum of number of nodes corresponding to each memory tap. The total number of nodes in hidden layers 1 and 2 is given by

$$\mathrm{HL}_C = \mathrm{HL}_{C,0} + \mathrm{HL}_{C,1} + ... + \mathrm{HL}_{C,M}, \tag{3}$$

where, HL_C corresponds to total number of nodes in hidden layer C, $\mathrm{HL}_{C,m}$ corresponds to number of nodes in hidden layer C which are densely connected with nodes of previous layer corresponding to the memory tap m. The activation function used for all the hidden layers is leaky rectified linear unit (LReLU), which is given by

$$\mathrm{LReLU}(X) = \begin{cases} X & \text{if} \quad X \geq 0, \\ \alpha X & \text{otherwise,} \end{cases} \tag{4}$$

where α is scaling factor.

In order to provide more importance to the current sample $x(n)$, we suggest using more nodes for corresponding sub-networks ($\mathrm{HL}_{1,0}$ and $\mathrm{HL}_{2,0}$). This enables the network to learn more complex combinations of the current input sample compared to that of the previous samples. The number of nodes corresponding to previous samples $(x(n-1), \ldots, x(n-M))$ can be same ($\mathrm{HL}_{1,1} = \mathrm{HL}_{1,2} = \ldots = \mathrm{HL}_{1,M}$ and $\mathrm{HL}_{2,1} = \mathrm{HL}_{2,2} = \ldots = \mathrm{HL}_{2,M}$) giving equal importance to all the previous samples, or we can have chronological importance for previous samples ($\mathrm{HL}_{1,1} > \mathrm{HL}_{1,2} > \ldots > \mathrm{HL}_{1,M}$ and $\mathrm{HL}_{2,1} > \mathrm{HL}_{2,2} > \ldots > \mathrm{HL}_{2,M}$) as well. In this paper, we have shown the experimental results with equal importance to all the memory taps in sec IV. Also, with the proposed architecture we can have the flexibility to define the highest non-linear order for each of the memory tap separately.

C. Training Mechanism

First, we obtain the $(x(n), y(n))$ pairs. Let $\mathbf{X}(n)$ be the vector consisting of I, Q, magnitude and magnitude powers of

Table I: Parameters for the proposed algorithm

Parameter	Value
Maximum order of non linearity, K	2
Memory length, M	11
Hidden layer 1 current sample, $HL_{1,0}$	9
Hidden layer 2 current sample, $HL_{2,0}$	5
Hidden layer 1 past samples, $HL_{1,(1-5)}$	5
Hidden layer 2 past samples, $HL_{2,(1-5)}$	2
Hidden layer 3	10
Hidden layer 4	20
Hidden layer 5	10
Output layer	2
α for LReLU in Hidden layer 1	0.5
α for LReLU in Hidden layers 2 to 5	0.2
Optimizer	Adam
Loss	MSE

$(x(n), x(n-1), ...x(n-m))$. The proposed network needs to be trained to estimate the $y(n)$ using $\mathbf{X}(n)$ as input to the network. Let θ_{PA} be the proposed network parameters, so that $\widehat{y}(n) = f(\mathbf{X}(n); \theta_{\text{PA}})$. Then, the parameter θ_{PA} is trained to minimize the loss function L_{PA} given by the mean square error (MSE) between the estimated PA output $\widehat{y}(n)$ and the actual PA output $y(n)$. For N training samples the loss function L_{PA} is given by,

$$L_{\text{PA}} = \frac{1}{N} \sum_{i=0}^{N-1} \parallel \widehat{y}_i(n) - y_i(n) \parallel_2^2 . \tag{5}$$

III. Proposed Algorithm

The following steps are followed in the training phase

- Step 1: Obtain the (input, output) pairs of the PA.
- Step 2: Obtain the maximum of absolute value of the complex valued input and use it as normalizing factor.
- Step 3: Normalize both the input and output by dividing each of these values with the normalizing factor.
- Step 3: Use these normalized pairs (norm input, norm output) for training the proposed network.
- Step 4: Save the model containing the trained parameters

During testing phase

- Step 1: Obtain the PA input of test dataset.
- Step 2: Normalize the input using the same normalizing factor used when training the network.
- Step 3: Load the model and use the normalized input to predict the normalized output.
- Step 4: To get the predicted output of the PA, multiply the predicted normalized output with the normalizing factor.
- Step 5: The predicted output and the actual output can then be used to determine the NMSE and ACEPR metrics.

IV. Simulation Results

A. Parameter Settings

Here, we chose the maximum order of non linearity to be 2 and the memory length to be 11. The proposed network is implemented and trained with the parameters as mentioned in Table I

Table II: EVM and ACEPR of proposed method

Dataset	NMSE (dB)	ACEPR (dB)
Train	-32.1	-44.23
Test1	-33.56	-44.27
Test1 - Piece1	-34.52	-46.12
Test1 - Piece2	-28.97	-43.52
Test2	-30	-43.26
Test2 - Piece1	-28.43	-42.71
Test2 - Piece2	-30.3	-46.64
Test3	-33.71	-46.3
Test4	-33.33	-43.56

Table III: Computation complexity

Layer	I/P nodes	O/P nodes	No. of layers	Real mul
HL 1 C	4	9	1	36
HL 1 P	4	5	11	220
HL 2 C	9	5	1	45
HL 2 P	5	2	11	110
HL 3	27	10	1	270
HL 4	10	20	1	200
HL 5	20	10	1	200
OL	10	2	1	20
Total Real Multiplications				1101

While training, we first used a batch size of 200 for 3000 epochs and saved the model. After this, we reloaded the saved model and trained the model using a batch size of 1000 for 300 epochs.

B. Performance Analysis

As we assumed the memory length to be 11, the first 11 samples in each of the datasets are ignored from processing and hence are ignored from NMSE and ACEPR calculations. The NMSE and ACEPR values thus obtained for different datasets using the proposed method are tabulated in Table II.

C. Computational Complexity

The proposed method consists of 1 input layer, 5 hidden layers and 1 output layer with $K = 2$ and $M = 11$. The input layer is 48×1 vector, 4 real valued numbers ($I\,Q\,|.|\,and|.|^2$) for each of the current and past 11 samples. The sub-network in HL_1 corresponding to current sample contain 9 nodes and the rest contain 5 nodes. For obtaining the output of HL_1 we need $(4 * 9) + (4 * 5) * 11 = 256$ real multiplications and additions. Similarly, for HL_2 we need $(9 * 5) + (5 * 2) * 11 = 155$ real multiplications and additions, and for HL_3 we need $27 * 10 = 270$ real multiplications and additions, and for HL_4 we need $10 * 20 = 200$ real multiplications and additions, and for HL_5 we need $20 * 10 = 200$ real multiplications and additions, and for the output layer we need $10 * 2 = 20$ real multiplications and additions. So we need a total of $256 + 155 + 270 + 200 + 200 + 20 = 1101$ real multiplications for an inference. On top of this, we may require 1 more real multiplications for the square term of the current sample for feeding the appropriate input. This need not be calculated separately for past samples as they would have been calculated in the previous instance. So, we need $1101 + 1 = 1102$ real multiplications for input processing and inference together.

Table IV: EVM and ACEPR of proposed method

Dataset	NMSE (dB)	ACEPR (dB)
Train	-32.84	-45.6
Test1	-34.97	-45.68
Test1 - Piece1	-36.41	-48.57
Test1 - Piece2	-29.35	-45.77
Test2	-30.29	-44.45
Test2 - Piece1	-28.39	-44.36
Test2 - Piece2	-30.64	-48.51
Test3	-34.68	-48.46
Test4	-33.77	-45.37

D. Further Improvements

We experimented with changing the activation functions and observed that by using $tanh$ as activation function for the fully connected layers (hidden layers 3, 4, and 5) there is an improvement in performance using the same network. The performance metrics obtained using $tanh$ as activation function for the hidden layers 3, 4, and 5 while retaining every other aspect as same for the network is shown in Table IV. Another change is that we trained the saved model using a batch size of 200 for 300 epochs with a learning rate of 10^{-5} instead of using batch size 1000 for 300 epochs with learning rate 10^{-3} in the previous solution. With these changes, we observed an improvement of 0.71 dB in NMSE and 1.8 dB in ACEPR compared to the previous solution across all the datasets.

REFERENCES

[1] Lei 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 Transactions on Communications*, vol. 52, no. 1, pp. 159-165, Jan. 2004, doi: 10.1109/TCOMM.2003.822188.

[2] P. Saikrishna, A. Goyal, A. Kumar, A. K. R. Chavva, S. Kim and S. Lee, "Memory Polynomial-Inspired Neural Network to Compensate the Power Amplifier Non-linearities," Accepted in *IEEE Personal, Indoor and Mobile Radio Communications Conference*, 2022.