Journal of Applied Mathematics and Physics, 2021, 9, 403-412
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

Deep Neural Network Based Behavioral Model of Nonlinear Circuits

Zhe Jin, Sekouba Kaba

School of Automation and Electrical Engineering, Zhejiang University of Science and Technology, Hangzhou, China
Email: zjuz@163.com

How to cite this paper: Jin, Z. and Kaba, S. (2021) Deep Neural Network Based Behavioral Model of Nonlinear Circuits. *Journal of Applied Mathematics and Physics*, **9**, 403-412.
https://doi.org/10.4236/jamp.2021.93028

Received: January 30, 2021
Accepted: March 8, 2021
Published: March 11, 2021

Abstract

With the rapid growth of complexity and functionality of modern electronic systems, creating precise behavioral models of nonlinear circuits has become an attractive topic. Deep neural networks (DNNs) have been recognized as a powerful tool for nonlinear system modeling. To characterize the behavior of nonlinear circuits, a DNN based modeling approach is proposed in this paper. The procedure is illustrated by modeling a power amplifier (PA), which is a typical nonlinear circuit in electronic systems. The PA model is constructed based on a feedforward neural network with three hidden layers, and then Multisim circuit simulator is applied to generating the raw training data. Training and validation are carried out in Tensorflow deep learning framework. Compared with the commonly used polynomial model, the proposed DNN model exhibits a faster convergence rate and improves the mean squared error by 13 dB. The results demonstrate that the proposed DNN model can accurately depict the input-output characteristics of nonlinear circuits in both training and validation data sets.

Keywords

Nonlinear Circuits, Deep Neural Networks, Behavioral Model, Power Amplifier

1. Introduction

Generating accurate circuit models is a common and efficient step towards system level design of electronic devices. To meet rigorous requirements for modern electronic systems, developing efficient modeling method has become an important research topic. According to the type of data needed for extraction, circuit models are divided into two categories: physical models and behavioral models [1]. Physical models are created by analyzing the circuitry of the devices, while behavioral models characterize the devices in terms of input and output signals, without resorting to their internal constitution.

Usually, modeling linear circuits is simple because their parameters are constant and the output is proportional to the input. However, most circuits are nonlinear. The nonlinearities contained in transistors or diodes make the modeling process relatively hard. Over the past decades, artificial neural networks have been proved to be a powerful tool for nonlinear regression. Neural networks trained from measured data can represent the nonlinear behavior of electronic devices. Several neural network based modeling applications have been reported recently. Cao Y. and Zhang Q.J. presented a new approach for developing recurrent neural network models of nonlinear circuits [2]. Tarver C. *et al.* developed a neural-network method to combat nonlinearities in power amplifiers [3]. Chen Z. *et al.* proposed a method for data-driven behavioral modeling of electronic circuits using recurrent neural networks [4].

Theoretically, traditional shallow neural networks with a single hidden layer can approximate any nonlinear function with arbitrary accuracy if the number of neurons is increased without constraint. However, increasing the hidden node number causes exponential increasing on the number of model parameters, and also requires much more training examples [5]. In recent years, deep neural networks (DNNs) have attracted a lot of researchers. Unlike traditional neural networks, deep neural networks possess multiple hidden layers and need fewer parameters. It is demonstrated that deeper networks can fit more complex functions by composing the functions learned in earlier layers [6].

In this paper, a nonlinear circuit behavioral modeling technique is developed on the basis of deep neural networks. The remaining parts of the paper are organized as follows. Section 2 describes the structure of a deep feedforward neural network. In Section 3, a power amplifier circuit is created as the object to be modeled. Training and validation are presented in Section 4.

2. Deep Feedforward Neural Networks

There is no clear threshold of depth that divides shallow neural networks from deep neural networks. But it is mostly agreed that the deep structure of a neural network requires two or more hidden layers. The number of neurons may be equal or different in each of the hidden layers. These neurons find the mathematical manipulation to obtain output from the input, whether it is a linear or nonlinear relationship.

In this work, a feedforward neural network with multiple hidden layers is adopted to model nonlinear circuits. In a multi-layer neural network, the neurons are arranged in layered fashion, in which the input and output layers are separated by multiple hidden layers. This layer-wise architecture is referred to as feedforward because the information only travels forward in the network, through the input nodes then through the hidden layers and finally through the output nodes. Although a single hidden layer with enough neurons is sufficient to make the network a universal approximator, there are substantial benefits to using many such hidden layers.

Figure 1 shows a feedforward network with two hidden layers. It contains m input nodes and one output node. The input data vector is represented by

$$\boldsymbol{X} = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix}^{\mathrm{T}}, \tag{1}$$

where the superscript T denotes the transpose of the matrix. The first hidden layer contains n neurons. The connection weight matrix from the input layer to the first hidden layer is

$$\boldsymbol{W}_1 = \begin{bmatrix} w_{111} & w_{112} & \cdots & w_{11m} \\ w_{121} & w_{122} & \cdots & w_{12m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1n1} & w_{1n2} & \cdots & w_{1nm} \end{bmatrix}. \tag{2}$$

The bias of the first hidden layer is

$$\boldsymbol{B}_1 = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \end{bmatrix}^{\mathrm{T}}. \tag{3}$$

$F(\cdot)$ is the activation function in hidden layers. The sigmoid function and the hyperbolic tangent function are two classical activation functions used for incorporating nonlinearity in neural networks. The sigmoid function outputs a value in (0, 1), which is helpful in performing computations that should be interpreted as probabilities. The hyperbolic tangent function has a shape similar to that of the sigmoid function, except that it is horizontally rescaled and vertically translated to (−1, 1). Since the voltage and the current of electric circuits may be both positive and negative, the hyperbolic tangent function is preferable to the sigmoid function in this application. Furthermore, its mean-centering and larger gradient with respect to sigmoid makes it easier to train. The hyperbolic tangent function is given by

$$F(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}. \tag{4}$$

So the output of the first hidden layer is calculated by

$$\boldsymbol{H}_1 = \tanh(\boldsymbol{W}_1 \times \boldsymbol{X} + \boldsymbol{B}_1). \tag{5}$$

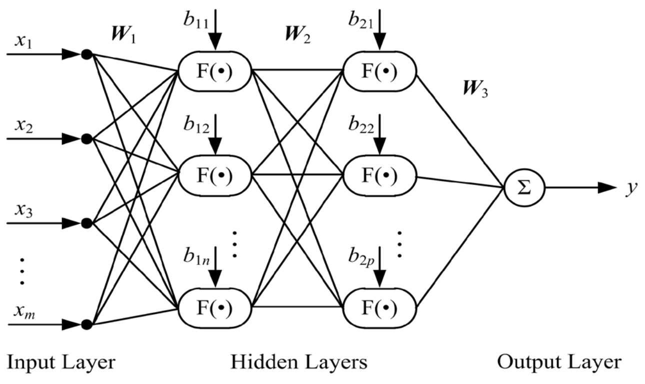

Figure 1. A feedforward neural network with two hidden layers.

The second hidden layer contains p neurons. The connection weight between two hidden layers is

$$\boldsymbol{W}_2 = \begin{bmatrix} w_{211} & w_{212} & \cdots & w_{21n} \\ w_{221} & w_{222} & \cdots & w_{22n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{2p1} & w_{2p2} & \cdots & w_{2pn} \end{bmatrix}. \tag{6}$$

The bias of the second hidden layer is denoted by

$$\boldsymbol{B}_2 = \begin{bmatrix} b_{21} & b_{22} & \cdots & b_{2p} \end{bmatrix}^{\mathrm{T}}. \tag{7}$$

Hence, the output of the second hidden layer is

$$\boldsymbol{H}_2 = \tanh\left(\boldsymbol{W}_2 \times \boldsymbol{H}_1 + \boldsymbol{B}_2\right). \tag{8}$$

The output of the network is

$$y = \boldsymbol{W}_3 \times \boldsymbol{H}_2 \tag{9}$$

where $\boldsymbol{W}_3$ is the connection weight from the second hidden layer to the output and is given by

$$\boldsymbol{W}_3 = \begin{bmatrix} w_{31} & w_{32} & \cdots & w_{3p} \end{bmatrix}. \tag{10}$$

In neural network training, the loss function is determined as the difference between the actual output and the predicted output from the model for one training example, while the average of the loss function for all the training examples is termed as the cost function. Briefly, the loss function is for a single training example, while the cost function is the average loss over the complete training data set. The loss function, which is sensitive to the application at hand, is critical to training and modeling process. A lot of loss functions, such as cross entropy and hinge loss, have been adopted in different types of applications. For regression problems, the squared error loss is most frequently used. Squared error loss for each training example is the square of the difference between the actual and the predicted values. Here, mean squared error (MSE) is determined as the cost function. It is computed by

$$\mathrm{MSE} = 10\lg\left[\frac{1}{K}\sum_{i=1}^{K}\left(\hat{y}_i - y_i\right)^2\right](\mathrm{dB}), \tag{11}$$

where K is the number of examples in the data set, $\hat{y}_i$ is the i-th example output, and y_i is the i-th output value of the model.

MSE measures the average squared difference between the actual and predicted values from the model. The aim of training is to minimize MSE. Gradient descent is one of the most popular algorithms to optimize neural networks. It seeks to determine the steepest descent and reduces the number of iterations and time taken to search large quantities of data points [7]. The gradient descent continuously updates parameters incrementally when an error calculation is completed to improve convergence. Model parameters are updated by

$$\boldsymbol{\theta}_{q+1} = \boldsymbol{\theta}_q - \eta \times \nabla_{\theta}\boldsymbol{J}\left(\boldsymbol{\theta}\right) \tag{12}$$

where $\boldsymbol{\theta}$ denotes the parameters to be optimized (namely the connection weight $\boldsymbol{W}$ and the bias $\boldsymbol{B}$ in this application), the subscript q stands for the iteration number, η is the learning rate, ∇_{θ} is the gradient with respect to $\boldsymbol{\theta}$, and $\boldsymbol{J}(\boldsymbol{\theta})$ is the cost function.

3. Power Amplifiers

The parameters of linear circuits are constant. Their output response is directly proportional to the input. Contrary to linear circuits, the parameters of nonlinear circuits vary with current and voltage. Only a few simple nonlinear circuits are adequately described by equations that have a closed form solution. Most of them may possess multiple solutions or may not possess a solution at all [8]. Therefore, analyzing nonlinear circuits is usually difficult and we often characterize them by means of models.

Nonlinear circuit models are categorized into physical models and behavioral models. Physical models are generated according to the internal circuitry of the devices, while behavioral models are established by means of input-output signals. For a complicated electronic system, acquiring its precise physical model is impractical. So the behavioral model is a better choice. In behavioral modeling, the electronic system is treated as a black box, whose internal constitution is unknown. The aim of behavioral modeling is to depict the system's input-output relation in a mathematical form [9].

In nonlinear circuits, the superposition principle does not hold and there is a natural generation of harmonic frequencies. The nonlinear circuits are also capable of exhibiting a self-sustained oscillation, which may be desired, as in the case of frequency dividers and free-running oscillators, or undesired, as in the case of power amplifiers (PAs) [10].

The power amplifier is a typical nonlinear device, which is widely used in electronic systems. It receives an electrical signal and reprocesses it to amplify or increase its power. The main features of a power amplifier are the circuit's power efficiency and the maximum amount of power that the circuit is capable of handling. To attain large output power and high energy efficiency, power amplifiers are often driven to maximum ratings, which results in serious nonlinear distortion.

Next, we take a push-pull PA as an example to illustrate the proposed modeling techniques. The schematic of the power amplifier is shown as **Figure 2**. The circuit simulation is performed in Multisim. There are two stages in the amplifier circuit. The first stage is a small-signal amplifier and the second stage is an output transformerless complementary-symmetry amplifier [11]. The input signal is provided by a function generator XFG1, the output signal is observed by an oscilloscope XSC1, and the distortion of the output signal is measured by a distortion analyzer XDA1.

An ideal amplifier is capable of amplifying a pure sinusoidal signal to provide a larger version and the resulting waveform is a pure single-frequency sinusoidal signal. When the device works at nonlinear regions, distortion occurs and the output will not be an exact duplicate of the input signal [12]. The distortion of the output signal is measured by total harmonic distortion (THD), which is defined as

$$\mathrm{THD} = \sqrt{d_1^2 + d_2^2 + \cdots d_N^2} \times 100\% . \tag{13}$$

In Equation (13), $d_i\left(i = 1, 2, \cdots, N\right)$ is the *i*-th harmonic distortion determined by

$$d_i = \frac{|A_i|}{|A_1|} \times 100\% . \tag{14}$$

where A_1 is the fundamental amplitude and A_i is the *i*-th harmonic amplitude. When the frequency *f* and the amplitude of the input signal V_{Ain} are 1 KHz and 0.2V respectively, the output waveform is a magnified replica of the input signal, as shown in **Figure 3**. When the input signal's amplitude rises to 1.8V, the output signal is seriously distorted, as shown in **Figure 4**, because the transistors are driven to nonlinear region.

Besides linearity, power efficiency is also an important consideration. The power efficiency of an amplifier is defined as the ratio of the output power P_{out} to the input power P_{in}. The power efficiency and the THD with different amplitude of the input signal V_{Ain} are listed in **Table 1**. The power and the THD data are acquired by the wattmeter and the distortion analyzer in Multisim respectively. When the input signal is small, the linearity of the device is good but the power efficiency is poor. As the input increases, the linearity descends but the power efficiency ascends. For a practical PA, there must be a compromise between its linearity and power efficiency.

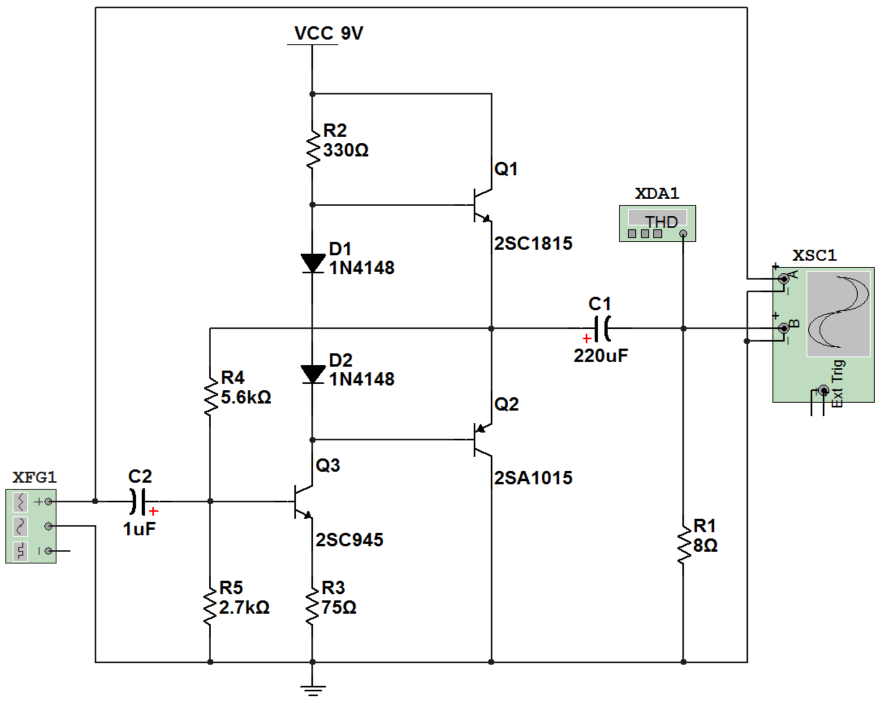

Figure 2. The schematic of the PA to be modeled.

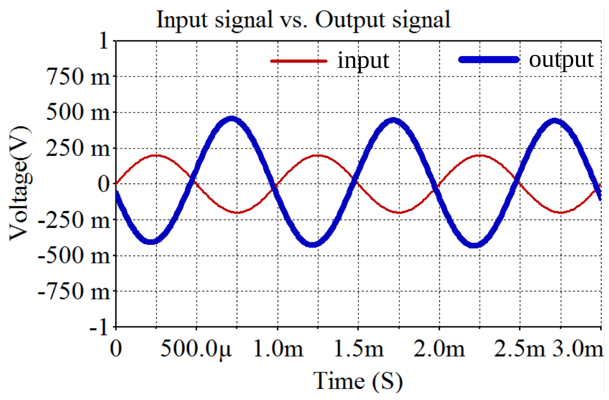

Figure 3. Input signal vs. output signal (V_{Ain} = 0.2 V and f = 1 KHz).

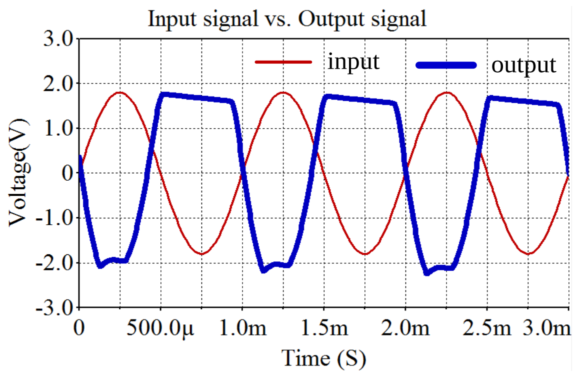

Figure 4. Input signal vs. output signal (V_{Ain} = 1.8 V and f = 1 KHz).

Table 1. Testing results of the power amplifier in Figure 2.

V_{Ain} (V)	0.2	0.6	1.0	1.4	1.8
P_{out} (mW)	12	105	239	302	327
P_{in} (mW)	486	482	519	524	520
Power efficiency	2.45%	21.78%	46.05%	57.63%	62.88%
THD	0.97%	1.88%	7.40%	20.84%	30.34%

4. Training and Validation

The proposed DNN model is trained and validated in Tensorflow, which is a popular machine learning framework developed by Google. Tensorflow is an open-source framework used in conjunction with Python to implement algorithms, deep learning applications, and much more. It contains a symbolic math library which is specially developed for machine learning applications such as deep neural networks.

To model the power amplifier discussed in Section 3, a neural network with three hidden layers is constructed. There are 25 neurons in each hidden layer. A part of Python code is as follows.

```
importtensorflow as tf
importnumpy as np
importmatplotlib.pyplot as plt
PA_dat = np.loadtxt('E:\deep\PA_dat.txt')   # load raw data generated by Multisim simulator
a=25   # number of neurons in each hidden layer
...
W1=tf.Variable(tf.random_normal([1,a]), name="weight1")   # weights between input and 1st hidden layer
B1=tf.Variable(tf.random_normal([1,a]), name="bias1")   # bias of 1st hidden layer
S1=tf.nn.tanh(tf.add(tf.matmul(XX,W1),tf.matmul(tf.ones([c,1]),B1)))   # output of 1st hidden layer
...
init=tf.global_variables_initializer()   # initialize network parameters
cost=tf.reduce_mean(tf.square(Y-Z))   # cost function: MSE
learning_rate=0.05   # learning rate
optimizer=tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)   # gradient descent
training_epochs=1000   # epoch number
withtf.Session() as sess:   # train the model
sess.run(init)
for epoch in range(training_epochs):
sess.run(optimizer, feed_dict={X:train_X, Y:train_Y})
```

For the purpose of comparison, a five-order polynomial model trained by the same data set is utilized as a reference model. The learning curves of two models are shown in Figure 5. After 1000 training epochs, the MSE of the DNN model is about −126 dB, while the MSE of the polynomial model is around −113 dB. Apart from higher accuracy, the DNN model also exhibits a faster convergence rate than the polynomial model does. However, it should be noticed that the DNN model uses more parameters and is more complicated.

The validation data set is generated by Multisim circuit simulator too. The frequency of validation signals ranges from 0.1 KHz to 1 MHz. One of the input-output waveform pairs is shown in Figure 6. The input signal consists of three frequency components, whose mathematical expression is

$$v_{\text{in}} = 0.8\sin\left(2\pi f_1 t\right) + 0.5\sin\left(2\pi f_2 t\right) + 0.4\sin\left(2\pi f_3 t\right)\ (\text{V}) \tag{15}$$

where f_1 = 10 KHz, f_2 = 20 KHz, and f_3 = 30 KHz. The validation result is shown in Figure 7. It illustrates that the model precisely fits the output data in validation set. In the validation set, the MSE of the DNN model is about −123 dB, while the MSE of the polynomial model is −112 dB. Therefore, compared with the commonly used polynomial model, the proposed DNN model improves accuracy in both training and validation data sets.

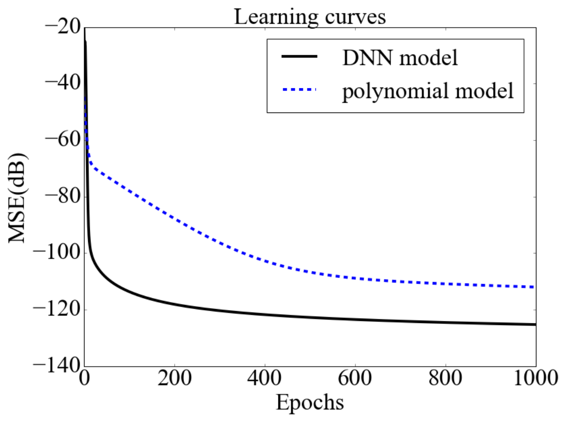

Figure 5. Learning curves of DNN model and polynomial model.

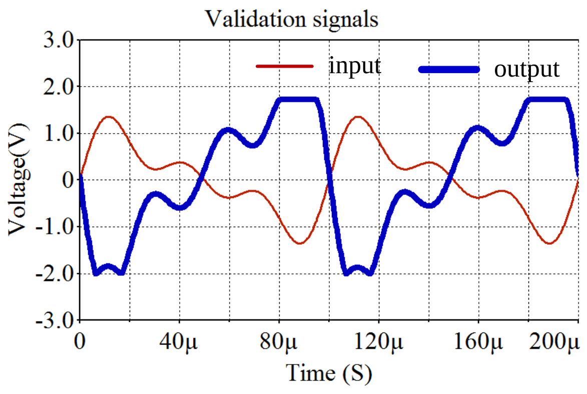

Figure 6. One of input-output waveforms used for validation.

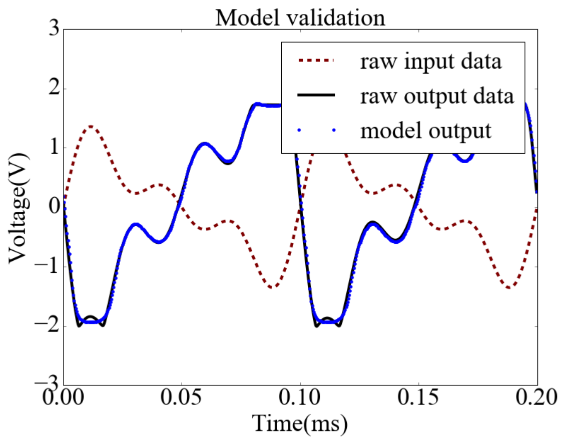

Figure 7. Model validation.

5. Conclusion

This paper presented a DNN based behavioral modeling approach for nonlinear circuits. A power amplifier was taken as an example to illustrate the proposed modeling method. A feedforward deep neural network with three hidden layers was adopted to model the amplifier. The results show that compared with the commonly used polynomial model, the proposed model not only improves precision but also provides a faster convergence rate. Applying other newly developed neural networks to modeling nonlinear circuits is an obvious future extension of this work.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Jin, Z., Zhou, Y. and Song, Z. (2009) Behavioral Modeling of RF Power Amplifiers Using Modified Volterra Series. *Journal of Circuits, Systems, and Computers*, **18**, 351-359. https://doi.org/10.1142/S0218126609005113

[2] Cao, Y. and Zhang, Q.J. (2009) A New Training Approach for Robust Recurrent Neural-Network Modeling of Nonlinear Circuits. *IEEE Transactions on Microwave and Techniques*, **57**, 1539-1553. https://doi.org/10.1109/TMTT.2009.2020832

[3] Tarver, C., Jiang, L., Sefidi, A. and Cavallaro, J.R. (2019) Neural Network DPD via Backpropagation through a Neural Network Model of the PA. *Proceedings of the 53rd Asilomar Conference on Signals, Systems, and Computers,* Pacific Grove, 3-6 November 2019, 358-362. https://doi.org/10.1109/IEEECONF44664.2019.9048910

[4] Chen, Z., Raginsky, M. and Rosenbaum, E. (2017) Verilog-A Compatible Recurrent Neural Network Model for Transient Circuit Simulation. *Proceedings of IEEE 26th Conference on Electrical Performance of Electronic Packaging and Systems* (*EPEPS*), San Jose, 15-18 October 2017. https://doi.org/10.1109/EPEPS.2017.8329743

[5] Liu, Z., Hu, X., Liu. T., Li, X., Wang, W. and Ghannouchi, M.G. (2020) Attention-Based Deep Neural Network Behavioral Model for Wideband Wireless Power Amplifiers. *IEEE Microwave and Wireless Components Letters*, **30**, 82-85. https://doi.org/10.1109/LMWC.2019.2952763

[6] Charu, C.A. (2018) Neural Networks and Deep Learning. Springer Nature Switzerland AG, Cham.

[7] Basodi, S., Ji, C., Zhang, H. and Pan, Y. (2020) Gradient Amplification: An efficient Way to Train Deep Neural Networks. *Big Data Mining and Analytics*, **3**, 196-207. https://doi.org/10.26599/BDMA.2020.9020004

[8] Tavazoei M.S., Kakhki M.T. and Bizzarri F. (2020) Nonlinear Fractional-Order Circuits and Systems: Motivation, a Brief Overview, and Some Future Directions. *Open Journal of Circuits and Systems*, **1**, 220-232. https://doi.org/10.1109/OJCAS.2020.3029254

[9] Riaza, R. (2020) Homogeneous Models of Nonlinear Circuits. *IEEE Transactions on Circuits and Systems-I*, **67**, 2002-2015. https://doi.org/10.1109/TCSI.2020.2968306

[10] Robert, L.B. and Louis N. (2011) Electronic Devices and Circuit Theory. 11th Edition, Pearson Education Inc., New Jersey.

[11] Bansal, R. and Majumdar, S. (2017) Nonlinear Modelling of Differential Amplifier Circuit. *Proceedings of the IEEE international Conference on Wireless Communications, Signal Processing and Networking* (*WiSPNET*), Chennai, 22-24 March 2017, 743-749. https://doi.org/10.1109/WiSPNET.2017.8299860

[12] Almudena, S. (2009) Analysis and Design of Autonomous Microwave Circuits. John Wiley & Sons, Inc., Hoboken, New Jersey.