
CHALMERS
UNIVERSITY OF TECHNOLOGY

# Residual Neural Networks for Digital Predistortion



Citation for the original published paper (version of record):
Wu, Y., Gustavsson, U., Graell I Amat, A. et al (2020). Residual Neural Networks for Digital Predistortion. 2020 IEEE Global Communications Conference, GLOBECOM 2020 - Proceedings. http://dx.doi.org/10.1109/GLOBECOM42002.2020.9322327

# Residual Neural Networks for Digital Predistortion

Yibo Wu*†, Ulf Gustavsson*, Alexandre Graell i Amat†, and Henk Wymeersch†
*Ericsson Research, Gothenburg, Sweden
†Chalmers University of Technology, Gothenburg, Sweden

GLOBECOM 2020 - 2020 IEEE Global Communications Conference | 978-1-7281-8298-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/GLOBECOM42002.2020.9322327

***Abstract*—Tracking the nonlinear behavior of an RF power amplifier (PA) is challenging. To tackle this problem, we build a connection between residual learning and the PA nonlinearity, and propose a novel residual neural network structure, referred to as the residual real-valued time-delay neural network (R2TDNN). Instead of learning the whole behavior of the PA, the R2TDNN focuses on learning its nonlinear behavior by adding identity shortcut connections between the input and output layer. In particular, we apply the R2TDNN to digital predistortion and measure experimental results on a real PA. Compared with neural networks recently proposed by Liu *et al*. and Wang *et al*., the R2TDNN achieves the best linearization performance in terms of normalized mean square error and adjacent channel power ratio with less or similar computational complexity. Furthermore, the R2TDNN exhibits significantly faster training speed and lower training error.**

## I. Introduction

Fifth generation (5G) wireless systems pose significant challenges to the performance of the radio frequency (RF) power amplifier (PA) [1]. High-frequency and high-bandwidth signals suffer severe distortions from the nonlinear behavior of the PA, which increases the need for highly linear PAs. Meanwhile, the increasing number of antennas and base-stations require a large number of PAs, which greatly increases the stress on power consumption, so the power efficiency of the PAs is also crucial.

In practice, the linearity and efficiency of the PA becomes a trade-off when both need to be satisfied. This trade-off has triggered intensive research over the past decades [2]–[4]. These works aim to preserve the PA linearity at the high output power region by using digital predistortion (DPD), a well-known technique to compensate for the PA nonlinearity. DPD performs an inverse nonlinear operation before the PA. This inverse operation can be represented by a parametric model, whose accuracy determines the DPD performance. Conventionally, Volterra series based models [2], such as memory polynomial (MP) [3] and generalized memory polynomial (GMP) [4], have been widely used for DPD because of their high accuracy. In these models, the behavior of the PA is represented by a set of Volterra kernels with different nonlinear orders where each kernel also considers memory effects, i.e., past inputs that influence the current output. These memory effects are due to the frequency-dependent behavior of the PA [5]. However, the performance of Volterra-based models is limited for severely nonlinear PAs even if high-order kernels are used because of the high estimation error for high-order kernels [6].

In contrast to model-based DPD approaches, deep learning techniques such as neural networks (NNs) have recently been proposed for DPD [7]–[14]. Among them, the multilayer perceptron (MLP) is the most commonly chosen type of NNs for DPD [9]–[14] because of the simple implementation and training algorithm. Based on the MLP, [9] proposed a real-valued time-delay neural network (RVTDNN) that separates the complex-valued signal into real in-phase and quadrature components to use a simple real-valued training algorithm. Furthermore, to consider memory effects of the PA, the input layer of the RVTDNN is fed by both the current instantaneous input and the inputs at previous time instants. To improve the performance of the RVTDNN, many variants have been studied [10]–[12], which add more components to the input layer, such as previous samples of the output signal [10], future samples of the input signal [11], or envelope terms (e.g., amplitude) of the input signal [12]. However, while these additional components have been shown to improve performance, they also significantly increase the network complexity, which pushes more pressure on the power consumption of DPD. [14] considered a different approach to connect the input and output layer by a linear bypass, which makes the NN focus on the nonlinear relation. However, this approach is infeasible for a memory input, which limits its performance on PAs with memory. Moreover, the performance comparison between NN with and without shortcuts for DPD is not discussed in [14].

In this paper, we build a connection between *residual learning* and the PA. We then propose a residual NN, referred to as residual real-valued time-delay neural network (R2TDNN) to learn the nonlinear behavior of the PA. Unlike RVTDNN [9] and its variants [10]–[12] that learn the PA linear and nonlinear behaviors jointly, the proposed R2TDNN learn them separately. Specifically, the PA nonlinear behavior is learned by its inner layers, and the linear behavior is added at the end of the inner layers using *identity shortcuts* between the input and output layer. The identity shortcuts introduce no new parameters as well as negligible computational complexity (one element-wise addition). Unlike [14], which excludes memory inputs, the R2TDNN considers memory inputs by applying identity shortcuts between two neurons of the current instantaneous input-output, which also solve the dimension

This work was supported by the Swedish Foundation for Strategic Research (SSF), grant no. I19-0021.

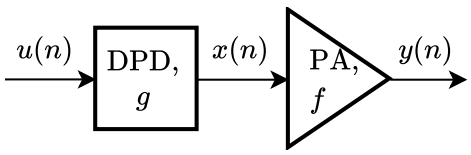


Fig. 1: Behavior relation between DPD and the PA. The power gain of the PA is normalized for simplicity. To compensate the PA nonlinear behavior $f$ before the saturation point, DPD performs an inverse operation $g$.

difference between the input and output layer. We apply the proposed R2TDNN to DPD. Experimental results on a real PA show that the proposed R2TDNN for DPD achieves a better linearization performance as well as a faster training rate than the RVTDNN in [9] and a variant of it in [12] with similar computational complexity.

## II. System Model

### A. PA behavior and DPD

The PA behaves as a nonlinear system that exhibits static nonlinearity and memory effects. The latter is more obvious in a wideband scenario because of the frequency-dependent gain and phase shift between the input and output signal [15]. Memory effects are exhibited in the time domain, which means that the PA output at any time instant is a function of the current instantaneous input and previous inputs. To take into account memory effects, we consider the PA as a function $f$: $\mathbb{R}^L \to \mathbb{R}$ with input and output signals $x(n)$ and $y(n)$ for $n \in \mathbb{Z}$, and input memory length $L$. The input-output relation of the PA can be expressed as

$$y(n) = f(x(n-L), \ldots, x(n)). \tag{1}$$

Meanwhile, the DPD is viewed as a function $g$: $\mathbb{R}^{L_1+L_2} \to \mathbb{R}$, with delayed and advanced memory length $L_1$ and $L_2$, and input signal $u(n)$ for $n \in \mathbb{Z}$, given by

$$x(n) = g(u(n-L_1), \ldots, u(n+L_2)). \tag{2}$$

As shown in Fig. 1, DPD is placed before the PA so as to cancel the distortion introduced by the PA. Assuming an ideal DPD cancellation, i.e., the DPD perfectly compensates for the distortion introduced by the PA, we then have the ideal input-output relation of the DPD–PA system by substituting (2) into (1), with $\boldsymbol{u}_{n-L} = [u(n-L-L_1), \ldots, u(n-L+L_2)]^{\mathsf{T}}$,

$$y(n) = f(g(\boldsymbol{u}_{n-L}), \ldots, g(\boldsymbol{u}_n)) = u(n). \tag{3}$$

In this case, the cascaded DPD–PA system is distortion-free.

However, an ideal DPD cancellation is infeasible in practice because of the saturation region and other non-deterministic factors such as noise. To minimize the distortion at the output of the PA, various behavioral models explore deterministic functions to approximate $g$ so as to make the DPD–PA system as linear as possible. Let $\hat{g}$ denote the approximated DPD function, which turns the distortion-free output $u(n)$ in (3) to a biased PA output $\hat{u}(n)$,

$$y(n) = f(\hat{g}(\boldsymbol{u}_{n-L}), \ldots, \hat{g}(\boldsymbol{u}_n)) = \hat{u}(n). \tag{4}$$

To reduce the output bias, a highly accurate model of $g$ is crucial.

### B. Generalized Memory Polynomial (GMP)

A popular form of a nonlinear, causal, and finite-memory system (e.g., the PA) is described by the Volterra series because of good precision. To ease the high complexity of Volterra series, many simplified Volterra models have been investigated in the literature [2]–[4]. In particular, the GMP [4] behavioral model has been shown to outperform many other models in terms of accuracy versus complexity [16].

Assuming a GMP model with memory depth $M$, nonlinear order $P$, cross-term length $G$, and input signal $x_{\text{in}}(n)$ at time $n$, the output of the GMP at time $n$, $\hat{y}_{\text{out}}(n)$, gives an estimation of the actual output $y_{\text{out}}(n)$ as [4]

$$\begin{aligned}\hat{y}_{\text{out}}(n) = &\sum_{p=0}^{P-1}\sum_{m=0}^{M} a_{pm} x_{\text{in}}(n-m)|x_{\text{in}}(n-m)|^p \\ &+ \sum_{p=1}^{P-1}\sum_{m=0}^{M}\sum_{g=1}^{G}(b_{pmg} x_{\text{in}}(n-m)|x_{\text{in}}(n-m-g)|^p \\ &+ c_{pmg} x_{\text{in}}(n-m)|x_{\text{in}}(n-m+g)|^p)\end{aligned} \tag{5}$$

where $a_{pm}$, $b_{pmg}$, and $c_{pmg}$ are complex-valued coefficients. Assuming a total number of coefficients $J$ and total number of input samples $N$, all coefficients can be collected into a $J \times 1$ vector $\boldsymbol{w}$. Each element of $\boldsymbol{w}$ corresponds to a $N \times 1$ signal, e.g., coefficient $a_{32}$ corresponds to the $N$ samples signal $x_{\text{in}}(n-2)|x_{\text{in}}(n-2)|^3$. Therefore, we can collect these $N \times 1$ input signals into the $N \times J$ matrix $\boldsymbol{X}_{\text{in}}$. Then, (5) can be rewritten in matrix form as

$$\hat{\boldsymbol{y}}_{\text{out}} = \boldsymbol{X}_{\text{in}} \boldsymbol{w}. \tag{6}$$

To solve for $\boldsymbol{w}$, the least squares algorithm is commonly used by minimizing the mean squared error (MSE) between the estimation $\hat{\boldsymbol{y}}_{\text{out}}$ and the observation $\boldsymbol{y}_{\text{out}}$, which gives a solution for $\boldsymbol{w}$,

$$\boldsymbol{w} = (\boldsymbol{X}_{\text{in}}^{\mathsf{H}} \boldsymbol{X}_{\text{in}})^{-1} \boldsymbol{X}_{\text{in}}^{\mathsf{H}} \boldsymbol{y}_{\text{out}}, \tag{7}$$

where $\mathsf{H}$ denotes Hermitian.

In a real-time scenario, the running complexity of DPD substantially restricts the system. Assuming $G < M+1$, reference [16] computes the running complexity of GMP, $C_{\text{GMP}}$, for each input sample in terms of the number of floating point operations (FLOPs),

$$\begin{aligned}C_{\text{GMP}} = &8\left((M+1)(P+2PG) - \frac{G(G+1)}{2}(P-1)\right) \\ &+ 10 + 2P + 2(P-1)G + 2P\min(G, M).\end{aligned} \tag{8}$$

### C. Inverse Structure to identify DPD coefficients

Before a behavioral model (e.g., the GMP model) is used to represent the DPD function $g$, we need to identify its coefficients. Since the DPD optimal output signal $x(n)$ is unknown, we cannot directly identify coefficients of a model using $u(n)$ and $x(n)$. Alternatively, we can use an inverse structure, the

indirect learning architecture (ILA) [2], to indirectly identify DPD parameters. First, an *inverse* PA model (also known as *post-distorter*) is identified using the PA output signal $y(n)$ as the input and the PA input signal $u(n)$ as the output. Once the the post-distorter is identified, its coefficients are copied to to an identical model (known as *pre-distorter*) which is then used as the DPD function $g$.

Although the learned post-distorter is not an optimal solution, ILA is still the most used identification method because of simple implementation and good performance. In this paper, we consider the ILA to identify the parameters of a DPD model.

## III. Proposed Residual Real-Valued Time-Delay Neural Network

In this section we build a connection between the residual learning and the PA behavior, and then propose a residual NN to learn the nonlinear behavior of the PA.

### A. Residual learning on the PA

The PA behavior consists of a linear and a nonlinear component. If we extract the linear relation, the input-output relation of the PA (1) can be rewritten as

$$y(n) = x(n) + \underbrace{f(x(n-L),\ldots,x(n)) - x(n)}_{=h(x(n-L),\ldots,x(n))}. \tag{9}$$

Here, let us refer to $f(x(n-L),\ldots,x(n))$ as the original function to be learned by the NN, and the last two terms on the right-hand side of (9), i.e., $f(x(n-L),\ldots,x(n)) - x(n)$, as the residual function, which is denoted by $h(x(n-L),\ldots,x(n))$.

In the field of image recognition, learning a residual function has been shown to be more effective than learning its corresponding original function [17]. Therefore, we hypothesize that learning the nonlinear behavior of the PA is easier than learning the whole behavior. We then propose a residual learning NN to learn the PA behavior, referred to as R2TDNN. Unlike the RVTDNN [9] and its variants [10]–[12] that learn the whole input-output relation of the PA jointly, i.e., learn the original function $f(x(n-L),\ldots,x(n))$, the proposed R2TDNN learns it separately as in (9). In particular, the residual function $h(x(n-L),\ldots,x(n))$, i.e., the PA nonlinear behavior, is learned by inner layers, and $x(n)$, i.e., the PA linear behavior, is then added to the output of the inner layers by using *shortcut* connections between input and output layers. Specifically, we adopt the identity shortcut, which performs an *identity* mapping between connected layers and introduces no extra parameters. The details of the identity shortcut in the R2TDNN are described in the next subsection.

### B. Architecture

The architecture of the R2TDNN is shown in Fig. 2. Based on the MLP, the R2TDNN consists of $K$ layers. The number of neurons of layer $k$ is denoted by $D_k$. The input vector of layer $k$ is denoted by $\boldsymbol{z}_k \in \mathbb{R}^{D_{k-1}}$, which is also the output of layer $k-1$. We denote the weight matrix and bias vector

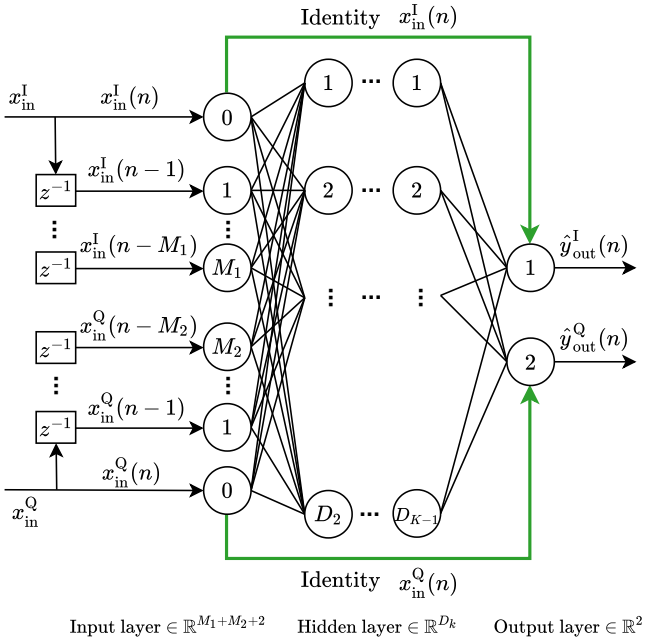


Fig. 2: Architecture of the proposed R2TDNN. Fed by the real in-phase and quadrature components of the input signal, $x_{\text{in}}^{\text{I}}$ and $x_{\text{in}}^{\text{Q}}$, the R2TDNN gives the I and Q output signal estimations $\hat{y}_{\text{out}}^{\text{I}}$ and $\hat{y}_{\text{out}}^{\text{Q}}$.

of layer $k$ by $\boldsymbol{W}_k \in \mathbb{R}^{D_k \times D_{k-1}}$ and $\boldsymbol{b}_k \in \mathbb{R}^{D_k}$, respectively. We consider a real-valued MLP, so the complex-valued input signal $x_{\text{in}}(n) = x_{\text{in}}^{\text{I}}(n) + j x_{\text{in}}^{\text{Q}}(n)$ at time instant $n$ is separated into real in-phase and quadrature components, $x_{\text{in}}^{\text{I}}(n)$ and $x_{\text{in}}^{\text{Q}}(n)$, respectively. To learn memory effects of the PA, the input signal of the first layer is formed by tapped delay lines, where each delay operator $z^{-1}$ yields one time instant delay, e.g., $x_{\text{in}}^{\text{I}}(n)$ to $x_{\text{in}}^{\text{I}}(n-1)$. We consider memory length $M_1$ and $M_2$ for the I and Q input components, respectively. Thus, the input signal of the first layer at time instant $n$ is given by

$$\begin{aligned}\boldsymbol{z}_1(n) = [&x_{\text{in}}^{\text{I}}(n), x_{\text{in}}^{\text{I}}(n-1), ..., x_{\text{in}}^{\text{I}}(n-M_1),\\ &x_{\text{in}}^{\text{Q}}(n), x_{\text{in}}^{\text{Q}}(n-1), ..., x_{\text{in}}^{\text{Q}}(n-M_2)],\end{aligned} \tag{10}$$

which yields $(M_1 + M_2 + 2)$ number of neurons for the first layer. When $M_1 = M_2 = 0$, the network neglects memory.

The layer $k-1$ and $k$ are fully connected as

$$\boldsymbol{z}_{k+1} = \sigma(\boldsymbol{W}_k \boldsymbol{z}_k + \boldsymbol{b}_k), \tag{11}$$

where $\sigma$ is the activation function. The output of the last layer is a $2 \times 1$ vector which corresponds to the in-phase and quadrature output signal estimations $\hat{y}_{\text{out}}^{\text{I}}(n)$ and $\hat{y}_{\text{out}}^{\text{Q}}(n)$ of the actual complex-valued output signal $y_{\text{out}}(n)$. To output a full range of values, the output layer is considered as a linear layer with no activation function. More importantly, we add the identity shortcut between the input and output layers. Unlike other shortcuts that fully connect two layers, as in [17], here the identity shortcut connection is performed between neurons. Only the two neurons fed by the current time instant input signal, i.e., $x_{\text{in}}^{\text{I}}(n)$ and $x_{\text{in}}^{\text{Q}}(n)$, are connected to

the output neurons. Therefore, the output of layer $K$ can be written as

$$[\hat{y}_{\text{out}}^{\text{I}}(n), \hat{y}_{\text{out}}^{\text{Q}}(n)] = [x_{\text{in}}^{\text{I}}(n), x_{\text{in}}^{\text{Q}}(n)] + \boldsymbol{W}_K \boldsymbol{z}_K + \boldsymbol{b}_K. \quad (12)$$

Note that the last two terms on the right hand side of (12) represents the residual function $h$ in (9), whereas the identity shortcut accounts for the linear part.

### C. Computation Complexity

The identity shortcut connection introduces no new parameters to the NN, and only two element-wise additions are added to the running complexity. All multiplications and additions are performed between real values, which accounts for one FLOP according to [16, Table I].

The number of FLOPs needed for the R2TDNN is

$$C_{\text{R2TDNN}} = 2 \sum_{k=1}^{K-1} D_k D_{k+1} + 2, \quad (13)$$

where the first term is the number of FLOPs for multiplication and addition operations, and the 2 is for two addition operations contributed by the two identity shortcuts.

### D. R2TDNN on DPD

The parameters of the R2TDNN can be learned through the back-propagation algorithm by minimizing the MSE between the prediction $\hat{y}_{\text{out}}(n)$ and observation $y_{\text{out}}(n)$,

$$(\boldsymbol{W}^*, \boldsymbol{b}^*) = \arg \min_{\boldsymbol{W}, \boldsymbol{b}} \mathbb{E}[(y_{\text{out}}(n) - \hat{y}_{\text{out}}(n))^2], \quad (14)$$

where $\mathbb{E}[\cdot]$ denotes the expectation. Specifically, when the R2TDNN is used as DPD, its parameters can be identified using the ILA, where the PA output $y(n)$ and input $x(n)$ are fed to the R2TDNN as input $x_{\text{in}}$ and output $y_{\text{out}}$, respectively.

## IV. Experimental results

We give experimental results of applying different behavioral models to DPD on a real PA.

### A. Evaluation Metrics and Measurement Setup

*1) Evaluation Metrics:* To evaluate the performance of DPD, the distortion level of the PA output signal is generally measured by the normalized mean square error (NMSE) between the PA output signal $y(n)$ (with gain normalization) and DPD input signal $u(n)$, and the adjacent channel power ratio (ACPR) of $y(n)$.

The NMSE is defined as

$$\text{NMSE} = \frac{\sum_n |y(n) - u(n)|^2}{\sum_n |u(n)|^2}. \quad (15)$$

Although the NMSE measures the all-band distortion, it can be used to represent the in-band distortion as the power of out-of-band distortion is negligible compared to the in-band distortion.

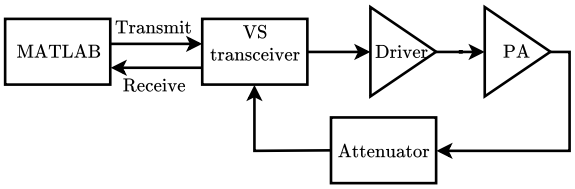


Fig. 3: Block diagram of the measurement setup. The RF WebLab is remotely accessed by the MATLAB which transmits and receives the pre-distorted and measured signals, respectively.

The ACPR measures the ratio of the out-of-band leakage to the in-band power, and is defined as

$$\text{ACPR} = \frac{\int_{\text{adj.}} |Y(f)|^2 \mathrm{d}f}{\int_{\text{ch.}} |Y(f)|^2 \mathrm{d}f}, \quad (16)$$

where $Y(f)$ denotes the Fourier transform of the PA output signal. The integration in the numerator and denominator are done over the adjacent channel (the lower or upper one with a larger leakage) and the main channel, respectively.

*2) Measurement Setup:* The experimental setup is based on the RF WebLab[1] [18]. Fig. 3 illustrates how it interacts with the hardware and digital signal processing (DSP) algorithms, e.g., DPD. In RF WebLab, a vector signal transceiver (VST) (PXIe-5646R VST) transmitter generates analog signals based on the digital signal from MATLAB. Signals are then sent to the Gallium Nitride PA DUT (Cree CGH40006-TB) driven by a 40 dB linear driver. Then, after a 30 dB attenuator, the VST receiver obtains the PA output signals, and eventually measurements are sent back to the MATLAB.

We then apply the proposed R2TDNN, GMP [4], RVTDNN [9], and augmented real-valued time-delay neural network (ARVTDNN) [12] to DPD with the RF WebLab setup. The learning architecture for all DPD models is the ILA [2] because of simple implementation. To identify DPD coefficients, GMP adopts the least squares algorithm, while RVTDNN, ARVTDNN, and R2TDNN use the back-propagation algorithm with the MSE loss function. We choose Adam [19] as the optimizer with a mini-batch size of $256$ and a learning rate of $0.001$. The activation function is the leaky rectified linear unit (ReLU) with a slope of $0.01$ for a negative input.

The input signal $u(n)$ is an orthogonal frequency division multiplexing (OFDM) signal with length $10^6$, sampling rate $200$ MHz, and signal bandwidth $10$ MHz. We consider a $50$ $\Omega$ PA load impedance. The measured saturation point and measurement noise variance of the PA are $24.1$ V ($\approx 37.6$ dBm) and $0.0033$, respectively. To test the DPD performance on the PA nonlinear region, we consider an average output power of the PA output signal of $25.36$ dBm, where the corresponding theoretical minimum NMSE is $-40.17$ dB

[1]RF WebLab is a PA measurement setup that can be remotely accessed at www.dpdcompetition.com

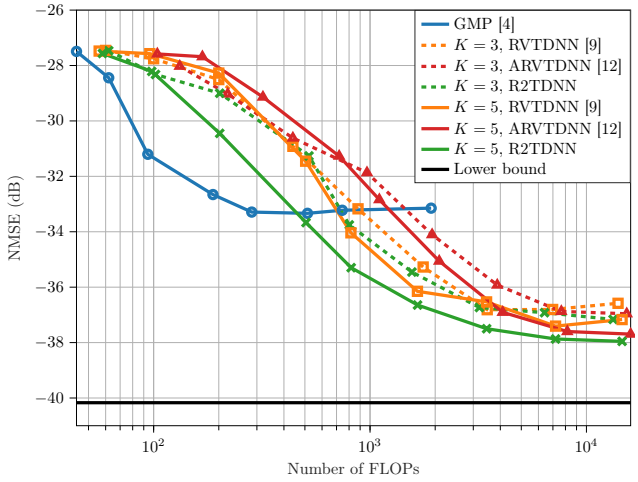


Fig. 4: NMSE as a function of the number of FLOPs. $M_1 = M_2 = 3$. The markers for RVTDNN [9], [12], and R2TDNN correspond to different $D_k$. The markers for GMP correspond to different values of $P$, $M$, and $G$. The lower bound represents the theoretical minimum NMSE that can be achieved for this PA at an average output power 25.36 dBm.

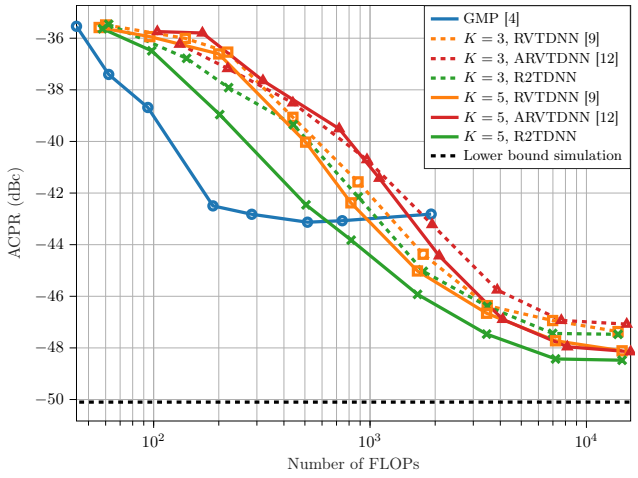


Fig. 5: ACPR as a function of the number of FLOPs. $M_1 = M_2 = 3$. The markers for RVTDNN [9], ARVTDNN [12], and R2TDNN correspond to different $D_k$. The markers for GMP correspond to different values of $P$, $M$, and $G$. The lower bound simulation represents the simulated minimum ACPR that can be achieved for this PA at an average output power 25.36 dBm.

according to [20, Eq. (10)], and the simulated minimum ACPR is $-50.1$ dBc.[2]

### B. Results

*1) Performance versus Complexity:* Fig. 4 and Fig. 5 show the NMSE results versus the total number of FLOPs for the GMP [4], RVTDNN [9], ARVTDNN [12], and the proposed R2TDNN. For the RVTDNN and R2TDNN, we consider two scenarios with one and three hidden layers, i.e., $K \in \{3, 5\}$. We also plot the results of the ARVTDNN [12] for $K = 5$, where we consider three augmented envelop terms of the signal (amplitude and its square and cube) [12, Tab. II] at the input layer. A proper number of memory length is related to the PA characteristics and input signal bandwidth, and here we choose identical input memory $M_1 = M_2 = 3$ for RVTDNN, ARVTDNN, and R2TDNN. Meanwhile, they use the same number of neurons for each hidden layer. The number of FLOPs increases as the number of neurons for each hidden layer increases. For the GMP (blue circle markers), we select the best results with respect to the number of FLOPs based on an exhaustive search of different values of $P$, $M$, and $G$.

Although the GMP model achieves better NMSE for a number of FLOPs $< 500$, the performance flattens around $-33.41$ dB. The proposed R2TDNN allows to reach lower NMSE (down to $-38.0$ dB) for a number of FLOPs $> 500$, i.e., the R2TDNN yields more accurate compensation—it can find a better inverse behavior of the PA. Note that the NMSE gap between the R2TDNN and the lower bound may be due to the limitation of the ILA and some stochastic noise, e.g., phase noise. The ACPR results in Fig. 5 illustrate similar advantages of the R2TDNN over the GMP for a number of FLOPs $> 600$.

For comparison, we also plot the performance of the RVTDNN in [9] and ARVTDNN in [12] for $K \in \{3, 5\}$. We note that the ARVTDNN requires a number of FLOPs $> 3000$ to improve the performance of RVTDNN. However, the proposed R2TDNN achieves lower NMSE and ACPR with respect to the RVTDNN and ARVTDNN for a similar number of FLOPs. The gain is more considerable for $K = 5$ and a

[2]The simulated minimum ACPR represents the ACPR of the ideal linear PA output signal.

TABLE I: NMSE and ACPR results of the RVTDNN [9], ARVTDNN [12], and R2TDNN for the convergence in Fig. 6. $K = 5$, $D_2 = D_3 = D_4 = 9$.

| | Num. FLOPs | NMSE [dB] | ACPR [dBc] |
|---|---|---|---|
| RVTDNN [9] | 504 | $-31.5$ | $-40.0$ |
| ARVTDNN [12] | 720 | $-31.3$ | $-39.5$ |
| R2TDNN | 506 | $\mathbf{-33.7}$ | $\mathbf{-42.5}$ |

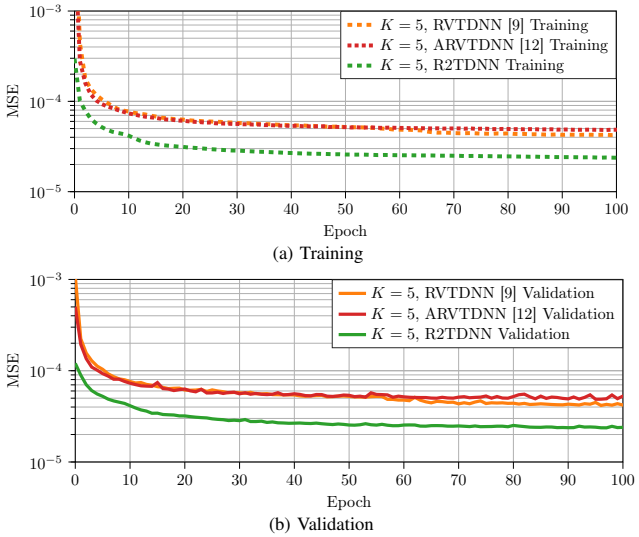


Fig. 6: Training and validation errors during the training process of the RVTDNN [9], ARVTDNN [12], and proposed R2TDNN. $K = 5$ and $D_2 = D_3 = D_4 = 9$.

number of neurons per hidden layer between $5$ and $12$.

*2) Convergence speed comparison:* To further compare the RVTDNN [9], ARVTDNN [12], and R2TDNN, we plot the training and validation errors during the training procedure in Fig. 6. Based on the same parameter setup in Section IV-B1, we select $K = 5$ and $D_2 = D_3 = D_4 = 9$. The corresponding number of FLOPs, NMSE, and ACPR are given in Table I. Compared to the RVTDNN and ARVTDNN, the R2TDNN exhibits significantly faster training convergence rate, and eventually achieves lower training and validation errors. This verifies the effectiveness of the proposed residual learning on DPD.

## V. Conclusion

We applied residual learning to facilitate the learning problem of the PA behavior, and proposed a novel NN-based PA behavioral model, named R2TDNN. By adding shortcuts between the input and output layer, the proposed R2TDNN focus on learning the PA nonlinear behavior instead of learning its whole behavior. We applied different behavioral models to DPD and evaluated the performance on a real PA. Results show that the proposed R2TDNN achieves lower NMSE and ACPR than the RVTDNN and ARVTDNN previously proposed in the literature with less or similar computational complexity. Furthermore, it has a faster training convergence rate during the training procedure.

## References


[1] N. Kelly, W. Cao, and A. Zhu, "Preparing linearity and efficiency for 5G: Digital predistortion for dual-band Doherty power amplifiers with mixed-mode carrier aggregation," *IEEE Microw. Mag.*, vol. 18, no. 1, pp. 76–84, Dec. 2016.

[2] C. Eun and E. J. Powers, "A new Volterra predistorter based on the indirect learning architecture," *IEEE Trans. Signal Process.*, vol. 45, no. 1, pp. 223–227, Jan. 1997.

[3] J. Kim and K. Konstantinou, "Digital predistortion of wideband signals based on power amplifier model with memory," *Electron. Lett.*, vol. 37, no. 23, pp. 1417–1418, Nov. 2001.

[4] 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 Trans. Signal Process.*, vol. 54, no. 10, pp. 3852–3860, Oct. 2006.

[5] J. C. Pedro and S. A. Maas, "A comparative overview of microwave and wireless power-amplifier behavioral modeling approaches," *IEEE Trans. Microw. Theory Tech.*, vol. 53, no. 4, pp. 1150–1163, Apr. 2005.

[6] S. Orcioni, "Improving the approximation ability of Volterra series identified with a cross-correlation method," *Nonlinear Dynamics*, vol. 78, no. 4, pp. 2861–2869, Dec. 2014.

[7] M. Isaksson, D. Wisell, and D. Ronnow, "Wide-band dynamic modeling of power amplifiers using radial-basis function neural networks," *IEEE Trans. Microw. Theory Tech.*, vol. 53, no. 11, pp. 3422–3428, Nov. 2005.

[8] D. Luongvinh and Y. Kwon, "Behavioral modeling of power amplifiers using fully recurrent neural networks," in *IEEE MTT-S Int. Microw. Symp. Dig.*, Jun. 2005, pp. 1979–1982.

[9] T. Liu, S. Boumaiza, and F. M. Ghannouchi, "Dynamic behavioral modeling of 3G power amplifiers using real-valued time-delay neural networks," *IEEE Trans. Microw. Theory Tech.*, vol. 52, no. 3, pp. 1025–1033, Mar. 2004.

[10] F. Mkadem and S. Boumaiza, "Physically inspired neural network model for RF power amplifier behavioral modeling and digital predistortion," *IEEE Trans. Microw. Theory Tech.*, vol. 59, no. 4, pp. 913–923, Jan. 2011.

[11] T. Gotthans, G. Baudoin, and A. Mbaye, "Digital predistortion with advance/delay neural network and comparison with Volterra derived models," in *IEEE 25th Annual Int. Symp. on Personal, Indoor, and Mobile Radio Commun.*, Sept. 2014, pp. 811–815.

[12] D. Wang, M. Aziz, M. Helaoui, and F. M. Ghannouchi, "Augmented real-valued time-delay neural network for compensation of distortions and impairments in wireless transmitters," *IEEE Trans. Neural Netw. Learn. Syst*, vol. 30, no. 1, pp. 242–254, Jun. 2018.

[13] R. Hongyo, Y. Egashira, T. M. Hone, and K. Yamaguchi, "Deep neural network-based digital predistorter for Doherty power amplifiers," *IEEE Microw. Wireless Compon. Lett.*, vol. 29, no. 2, pp. 146–148, Jan. 2019.

[14] C. Tarver, A. Balatsoukas-Stimming, and J. R. Cavallaro, "Design and implementation of a neural network based predistorter for enhanced mobile broadband," *arXiv preprint arXiv:1907.00766*, 2019.

[15] J. H. K. Vuolevi, T. Rahkonen, and J. P. A. Manninen, "Measurement technique for characterizing memory effects in RF power amplifiers," *IEEE Trans. Microw. Theory Tech.*, vol. 49, no. 8, pp. 1383–1389, Aug. 2001.

[16] A. S. Tehrani, H. Cao, S. Afsardoost, T. Eriksson, M. Isaksson, and C. Fager, "A comparative analysis of the complexity/accuracy tradeoff in power amplifier behavioral models," *IEEE Trans. Microw. Theory Tech.*, vol. 58, no. 6, pp. 1510–1520, Jun. 2010.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778.

[18] P. N. Landin, S. Gustafsson, C. Fager, and T. Eriksson, "Weblab: A web-based setup for PA digital predistortion and characterization [application notes]," *IEEE Microw. Mag.*, vol. 16, no. 1, pp. 138–140, Feb. 2015.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] J. Chani-Cahuana, C. Fager, and T. Eriksson, "Lower bound for the normalized mean square error in power amplifier linearization," *IEEE Microw. Wireless Compon. Lett.*, vol. 28, no. 5, pp. 425–427, May. 2018.