IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 15, 2020, accepted June 9, 2020, date of publication June 26, 2020, date of current version July 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3005146

Efficient Digital Predistortion Using Sparse Neural Network

MASAAKI TANIO, NAOTO ISHII, AND NORIFUMI KAMIYA, (Member, IEEE)
NEC Corporation, Kawasaki 211-8666, Japan
Corresponding author: Masaaki Tanio (m-tanio@nec.com)

ABSTRACT This paper proposes an efficient neural-network-based digital predistortion (DPD), named as envelope time-delay neural network (ETDNN) DPD. The method complies with the physical characteristics of radio-frequency (RF) power amplifiers (PAs) and uses a more compact DPD model than the conventional neural-network-based DPD. Additionally, a structured pruning technique is presented and used to reduce the computational complexity. It is shown that the resulting ETDNN obtained after applying pruning becomes so sparse that its complexity is comparable to that of conventional DPDs such as memory polynomial(MP) and generalized memory polynomial (GMP), while the degradation in performance due to the pruning is negligible. In an experiment on a 3.5-GHz GaN Doherty power amplifier (PA), our method with the proposed pruning had only one-thirtieth the computational complexity of the conventional neural-network-based DPD for the same adjacent channel leakage ratio (ACLR). Moreover, compared with memory-polynomial-based digital predistortion, our method with the proposed pruning achieved a 7-dB improvement in ACLR, despite it having lower computational complexity.

INDEX TERMS Digital predistortion, generalized memory polynomial, memory polynomial, neural network, pruning technique.

I. INTRODUCTION

To cope with dramatic increases in data traffic, wireless systems are becoming complex and power hungry. In particular, power amplifiers (PAs) that should be placed in each transmitter for 5G massive MIMO consume a lot power. As is well known, there is a trade-off between power efficiency and linearity in a PA. Basically a PA, which is operated at an efficient point, causes non-linear distortion. The distortion not only decreases the signal quality, but also causes spectrum regrowth, which poses a danger of disturbing adjacent channels. Thus, the PA output must be linear enough to meet the 3GPP specification of the adjacent channel leakage ratio (ACLR).

A digital predistortion (DPD) technique is indispensable for operating the PA at an efficient point while meeting the specification. DPD cancels the PA distortion by applying the inverse of the PA distortion in advance of amplification. Volterra-based DPDs, such as memory polynomial (MP) [1], generalized memory polynomial (GMP) [2] and dynamic deviation reduction (DDR) [3], are commonly used because of their ease of implementation. However, it is reported that recent efficient PA architectures, such as Doherty, envelope tracking and outphasing, are so complex that Volterra-based DPDs are unable to compensate their PA distortions sufficiently due to their limited structures [11].

Recently, a number of neural network-based DPDs have been presented [4], [5], [8], [9]. Their accurate modeling capability of neural networks enables them to outperform Volterra-based DPDs but at the expense of computational complexity.

Neural-network-based DPDs can be roughly divided into two network types. One type is a neural network in which input and output signals are split into I and Q signals. This separation simplifies the optimization by avoiding the complex-valued calculations. Real-valued time-delay neural network (RVTDNN) [4] is a well-known model of the I/Q separation type. In particular, the augmented real-valued time-delay neural network (ARVTDNN) [5] works by adding envelope signals to the input signals. Moreover, reference [6] and [7] present methods for MIMO transmitters that compensate not only PA distortion but also cross talk. Although the separated I/Q structures compensate for complex distortion not limited to PA distortion, these structures do not satisfy the constraints of the physical modeling imposed by the bandpass nature of PA distortion [20], [21]. Thus, because the

The associate editor coordinating the review of this manuscript and approving it for publication was Olga Fink.

constraints of the physical modeling are violated, the compensation performances of PAs are limited [9].

The other type of neural network is one in which the input signals are not I/Q signals but rather envelopes of the input signals, while the phase information is recovered by the phase filter. In this case, the output signals from the neural network, which only depends on the envelopes of the input signals, are the coefficients of the phase filter, and the phase filter performs a phase rotation by multiplying the coefficients and the phase information, which results in pre-distorted I and Q signals. Two methods [8], [9] in this framework have recently been proposed. Although both use the envelopes of the input signals, they differ in the phase filter. [9] uses only the phase information in the phase filter, while [8] uses the complex-valued input signals (including not only the phase but also the envelopes' information) and complex-valued multiplication is used to recover the phase information. Both methods have the good characteristic of satisfying the constraints of the physical modeling, which could make for a more compact structure than the I/Q separation type. Indeed, it was reported in [9] that the vector-decomposition-based time-delay neural network (VDTDNN) [9] outperforms RVTDNN [4].

However, even though [8] and [9] are compact and potentially have practical uses, due to the complex calculations of the neural networks, their computational complexity is larger than that of Volterra-based DPDs, which prevents them from being implemented on commercially available equipment.

To solve this problem, we propose a compact neural-network-based DPD, which satisfies the constraints of physical modeling, and its pruning method. The paper is organized as follows. In Section II, we describe our compact neural network model, called as envelope time-delay neural network (ETDNN). In Section III, we present an efficient pruning technique based on the essence of GMP that enables the structure of ETDNN-DPD to be sparse. An experimental validation and analysis are presented in Section IV. Conclusions are given in Section V.

II. PROPOSED NEURAL-NETWORK DIGITAL PREDISTORTION MODEL

A. ENVELOPE TIME-DELAY NEURAL NETWORK

Fig. 1 shows the block diagram of the proposed digital predistortion based on a neural network, which contains an artificial neural network and a phase filter. In terms of the general description of neural networks, the pre-distorted signal $z(k)$ can be formulated as follows:

$$z(k) = \sum_{m=0}^{M} \left\{ \sum_{j=1}^{N_{total}} w_{j,m}^{(2)} \phi \left(\sum_{l=0}^{M} w_{l,j}^{(1)} |x(k-l)| + b_j^{(1)} \right) + b_m^{(2)} \right\} x(k-m) \quad (1)$$

where $w_{l,j}^{(1)}$ and $b_j^{(1)}$ are real-valued weights and biases between the input layer and the hidden layer, and $w_{j,m}^{(2)}$ and $b_m^{(2)}$

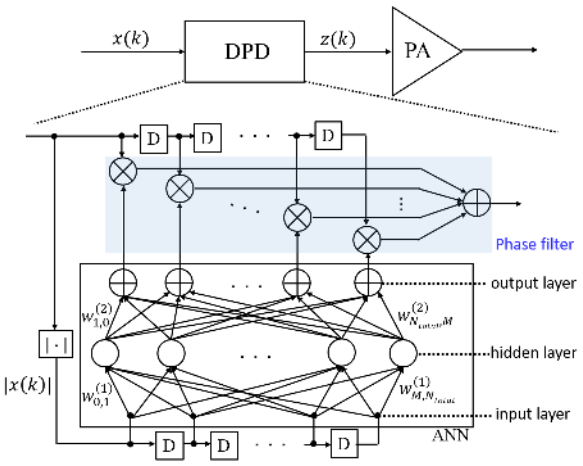

FIGURE 1. Block diagram of proposed digital predistortion based on the neural network.

are complex-valued weights and biases between the hidden layer and output layer in an artificial neural network (ANN). The input layer is composed of delayed amplitudes $|x(k-l)|$. Thus, the output signals from the ANN only depend on the amplitudes of the input signals and are fed to the phase filter as coefficients. The phase filter performs the phase rotation by multiplying the complex-valued coefficients and the complex-valued input signals $x(k-m)$.

Hereafter, we call the proposed model as envelope time-delay neural network (abbreviated as ETDNN).

It should be noted that the structure of ETDNN is similar to [8] as follows:

$$z(k) = \sum_{m=0}^{M} \left\{ \sum_{j=1}^{G} w_{j,m}^{(2)} \phi \left(\sum_{l=0}^{M} w_{l,j,m}^{(1)} |x(k-l)| + b_{j,m}^{(1)} \right) + b_m^{(2)} \right\} x(k-m) \quad (2)$$

Compared with the formulation of ETDNN (1), although almost all of structures are common, there is a big difference in that [8] does not share the weights $w_{l,j,m}^{(1)}$ and biases $b_{j,m}^{(1)}$ with each node of the output layer, which is shown in the dependency of m, while our proposed ETDNN does. From this difference, it can be considered that [8] is the special case of ETDNN. To distinguish [8] from our proposed ETDNN, we call [8] as the special case of ETDNN (abbreviated as scETDNN) in this paper.

ETDNN complies with the constraints of the physical modeling of RF PAs [20], [21] similarly to the vector-decomposition-based time-delay neural network (VDTDNN) [9]. Indeed, ETDNN formulated as (1) satisfies the two constraints ((i) odd party (ii) phase unity) of the physical modeling [21], as follows.

(I) ODD PARITY

The activation function ϕ is even parity with the respect to $x(k-m)$ because $|x(k-m)|$, which is the argument of ϕ,

is even parity. Therefore $z(k)$, which is calculated by summing the products of $x(k-m)$(odd) and ϕ(even), is odd parity.

(II) PHASE UNITY

$x(k-m)$ and $z(k)$ can be decomposed to their amplitudes and phases as $|x(k-m)|e^{j\theta}$ and $|z(k)|e^{j\psi}$, respectively. Thus, the phase difference $\psi-\theta$ does not depend on θ because $\psi-\theta$ is estimated as the phase of the ϕ summation, which only depends on $|x(k-m)|$, not θ. This means that phase unity is preserved.

This characteristic ensures that our proposed ETDNN is a compact model by avoiding nonphysical fitting. However, due to the complex-valued calculation, ETDNN requires a complex-valued back propagation for optimization, which dramatically increases the computational complexity. The next session explains a decomposition of ETDNN, which enables real-valued optimization.

B. DECOMPOSITION OF ENVELOPE TIME-DELAY NEURAL NETWORK FOR REAL-VALUED CALCULATION

ETDNN (1) has complex-valued parameters $w_{j,m}^{(2)}$, $b_m^{(2)}$ and $x(k-m)$ which are densely located around the phase filter. Thus, if the operation of the phase filter is converted to a real-valued one, the parameters in (1) can be unified into real values.

Fig. 2 shows the decomposition of one multiplication, which is the case of $m=0$ in (1), in the phase filter. According to (1), the complex-valued coefficient o, which is the signal output from the ANN, is formulated as

$$o=\sum_{j=1}^{N_{total}} w_{j,0}^{(2)}\phi(\cdot)+b_0^{(2)} \tag{3}$$

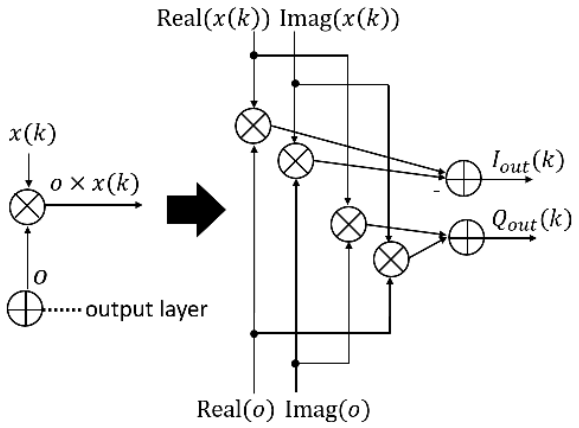

FIGURE 2. The decomposition of one multiplication in the phase filter.

Then, the complex-valued input signal $x(k)$ is decomposed into real and imaginary parts, Real($x(k)$) and Imag($x(k)$), respectively. Additionally, the complex-valued coefficient o in (3) is decomposed as follows:

$$\begin{aligned}\text{Real}(o)&=\sum_{j=1}^{N_{total}}\text{Real}(w_{j,0}^{(2)})\phi(\cdot)+\text{Real}(b_0^{(2)})\\ \text{Imag}(o)&=\sum_{j=1}^{N_{total}}\text{Imag}(w_{j,0}^{(2)})\phi(\cdot)+\text{Imag}(b_0^{(2)})\end{aligned} \tag{4}$$

Using these four real-valued parameters, i.e., Real(o), Imag(o), Real($x(k)$) and Imag($x(k)$), a real-valued operation that is mathematically equivalent to the complex-valued multiplication is conducted that outputs I/Q signals ($I_{out}(k)$ and $Q_{out}(k)$). The other multiplications for $m=1,2,\ldots,M$ in the phase filter are decomposed in the same way. Thus, all parameters in (2) can be treated as real values, which in turn enables real-valued optimization.

C. RELATIONSHIP WITH VECTOR-DECOMPOSITION-BASED TIME-DELAY NEURAL NETWORK MODEL

The decomposition shown in Fig. 2 also clarifies the relationship between ETDNN and VDTDNN [9]. Fig. 3 shows the first hidden neuron group of VDTDNN (corresponding to Fig. 8 in [9]). Comparing it with ETDNN (Fig. 1 and Fig. 2), although the ANN part is almost the same, we can distinguish two differences in the phase filter as follows: (i) Real($x(k)$) and Imag($x(k)$), which are reformulated as $|x(k)|\cos\theta$ and $|x(k)|\sin\theta$, in Fig. 2 correspond to $\cos\theta$ and $\sin\theta$ in Fig. 3, respectively. (ii) In Fig. 3, the four multiplications in the phase filter have four individually different input signals from the ANN, whereas those in Fig. 2 have only two input signals (Real(o) and Imag(o)) which are shared.

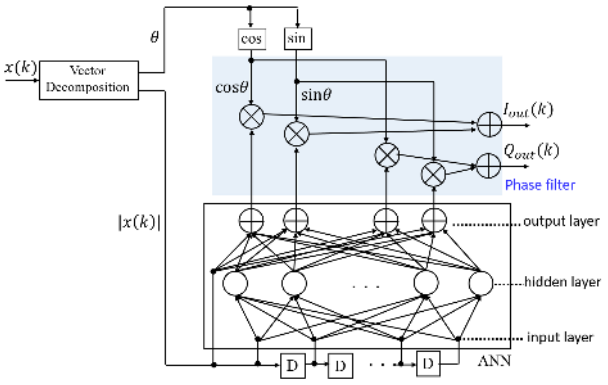

FIGURE 3. The first hidden neuron group of VDTDNN.

These differences are not essential to the physical modeling. Indeed, both methods satisfy the constraints of the physical modeling of RF PAs [20], [21]. However, when we consider the details of their computational complexity, the choice of method should be made carefully since the vector decomposition in Fig. 3 is performed by a pipelined CORDIC algorithm, which increases the computational complexity.

For a simple implementation, we focused on reducing the computational complexity of ETDNN by using the structured pruning described in the next section.

III. PROPOSED PRUNING

Fig. 4 shows the model after it has been pruned. As we can see, the number of connections has decreased, which lowers the computational complexity. What is better is that the pruning does not violate the constraints of the physical modeling [20], [21] for ETDNN because from the mathematical viewpoint, it only changes the coefficients (forces the pruned weights to be 0) in (1), which does not affect the constraints.

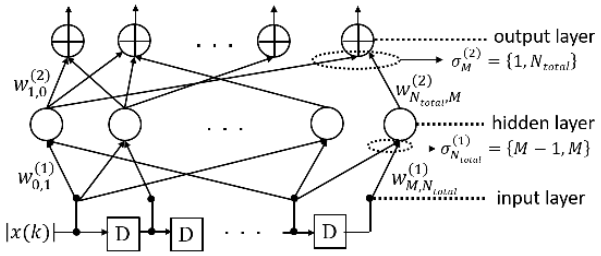

FIGURE 4. Block diagram of our model after pruning.

First, we will explain the conventional weight pruning method [10]. Then, we will analyze the structure of ETDNN, which leads to an advanced pruning for ETDNN.

To simplify the information of the connections on ETDNN, we define $\sigma_j^{(1)}$ as the indices of the nodes used as the input of the j-th nodes in the hidden layer. $\sigma_m^{(2)}$ is defined as the indices of the nodes used as the inputs of the m-th nodes in the output layer. Furthermore, we use the phrase ''input weights/biases'' to refer to weights/biases between the input layer and the hidden layer. In the same manner, ''output weights/biases'' will be used to refer to weights/biases between the hidden layer and the output layer.

A. CONVENTIONAL WEIGHT PRUNING

Fig. 5 shows a flowchart of the conventional weight pruning algorithm [10]. Step 1 obtains trained weights $w_{l,j}^{(1),N_{ep}}$ and $w_{j,m}^{(2),N_{ep}}$ after the N_{ep}-th iteration, where N_{ep} is the number of epochs. Step 2 checks the absolute values of the current weights $w_{l,j}^{(1),N_{ep}}$ and $w_{j,m}^{(2),N_{ep}}$ to obtain the pruned groups $\sigma_j^{(1),fixed}$ and $\sigma_m^{(2),fixed}$ as follows:

$$\begin{aligned}\sigma_j^{(1),fixed} &= \{l \mid |w_{l,j}^{(1),N_{ep}}| > Th^{(1)}, l = 0, 1, \ldots, M\},\\ \sigma_m^{(2),fixed} &= \{j \mid |w_{j,m}^{(2),N_{ep}}| > Th^{(2)}, j = 1, \ldots, N_{total}\},\end{aligned} \tag{5}$$

where $Th^{(1)}$ and $Th^{(2)}$ are the thresholds for the input and output weights, respectively. The pruned groups $\sigma_j^{(1),fixed}$ and $\sigma_m^{(2),fixed}$ contain the indices of the input and output weights whose absolute values are larger than $Th^{(1)}$ and $Th^{(2)}$, respectively.

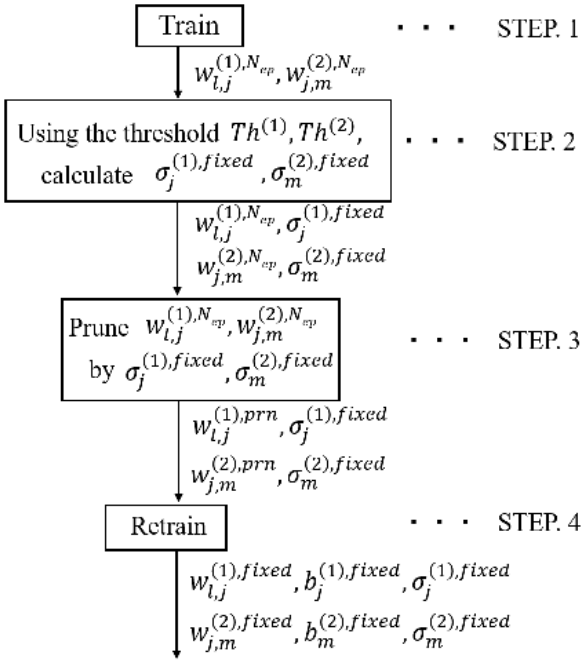

FIGURE 5. Flowchart of weight pruning algorithm.

In step 3, the lower weights are discarded using $\sigma_j^{(1),fixed}$ and $\sigma_m^{(2),fixed}$, as follows:

$$\begin{aligned}w_{l,j}^{(1),prn} &= \begin{cases} w_{l,j}^{(1),N_{ep}} & (if\ l \in \sigma_j^{(1),fixed}) \\ 0 & (otherwise)\end{cases}\\ w_{j,m}^{(2),prn} &= \begin{cases} w_{j,m}^{(2),N_{ep}} & (if\ j \in \sigma_m^{(2),fixed}) \\ 0 & (otherwise)\end{cases}\end{aligned} \tag{6}$$

where $w_{l,j}^{(1),prn}$ and $w_{j,m}^{(2),prn}$ are the input and output weights after pruning, respectively. A value of 0 means no connection (discarded).

Finally in step 4, all of the weights and biases are obtained by re-training, which keeps the pruned connections $\sigma_j^{(1),fixed}$, $\sigma_m^{(2),fixed}$ in (6).

This technique is generally used for reducing the computational complexity of neural networks. However, to make further reductions, we need to understand the structure of ETDNN itself.

In the next section, we propose a new structured pruning technique for ETDNN by referring to generalized memory polynomial (GMP) [2], which is one of the most efficient models in Volterra based DPDs [19].

B. STRUCTURAL INTERPRETATION OF ENVELOPE TIME-DELAY NEURAL NETWORK

Here, we examine the relationship between ETDNN and GMP. GMP [2] is defined as follows:

$$z(k) = \sum_{m=0}^{M_{GMP}} \sum_{d=-D}^{D} f_{d,m}(|x(k-m-d)|)x(k-m) \tag{7}$$

where $f_{d,m}$ is a non-linear function composed of polynomial functions, D is the number of cross-terms and M_{GMP} is the number of memory taps in GMP. Noted that to avoid using the future signals, $f_{d,m}$ with $m + d < 0$ is skipped according to [2].

For ease of understanding, (7) for specific m and d is visualized in Fig. 6. In this figure, the phase filter, which multiplies the output of the function $f_{d,m}(|x(k-m-d)|)$ by $x(k-m)$, is also used by ETDNN (Fig. 1) for specific m and j in (1). Thus, we can see that the only difference between GMP and ETDNN is the function before the phase filter.

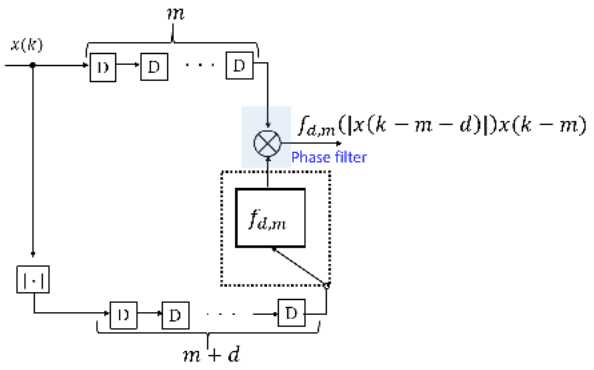

FIGURE 6. Fundamental block of GMP for specific *d* and *m*.

Thanks to the universal approximation theorem of neural networks [13], the $f_{d,m}$ in (7) can be approximated individually as

$$f_{d,m}(x) \approx \sum_{j=1}^{N} \check{w}_{j,d,m}^{(2)} \phi\left(\check{w}_{j,d,m}^{(1)} x + \check{b}_{j,d,m}^{(1)}\right) + \check{b}_{d,m}^{(2)} \quad (8)$$

where N is the number of nodes in the hidden layer. We will assume that N is so large that the approximation error in (8) is negligible.

The approximation (8) is illustrated in Fig. 7. As we can see, $f_{d,m}(|x(k-m-d)|)$, which has one variable, i.e., $|x(k-m-d)|$, is decomposed as a sum neurons that each have one input connection.

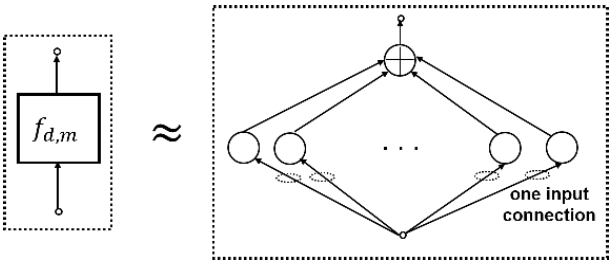

FIGURE 7. Applying the universal approximation theorem of neural networks to individual functions in GMP.

$|\sigma_j^{(1)}|$ is defined as the number of indices $\sigma_j^{(1)}$. Accordingly, the condition of one input connection of the node shown in Fig. 7 can be rephrased as $|\sigma_j^{(1)}| = 1$ for all j.

Combining Fig. 6 with Fig. 7 under the condition of $M = M_{GMP} + D$ in (1) and (7), we find that the approximated GMP shares the neurons and the filter structure with ETDNN, except for the restriction of $|\sigma_j^{(1)}| = 1$ on the neurons, which results in the following relationship:

$$X_{GMP} \subset X_{ETDNN||\sigma_j^{(1)}|=1} \quad (9)$$

where X_{GMP} and X_{ETDNN} are the solution spaces of GMP and ETDNN, respectively. This relationship ensures that if the number of nodes N_{total} for ETDNN is large enough and all the parameters for DPDs are optimized, ETDNN-DPD with $|\sigma_j^{(1)}| = 1$ will outperform GMP-DPD in spite of the strict condition $|\sigma_j^{(1)}| = 1$, which means "$|\sigma_j^{(1)}| = 1$" is potentially efficient when we re-consider $|\sigma_j^{(1)}| = 1$ as the pruning strategy. To embody the pruning strategy, in the next section, we extend the relationship (9) to an efficient pruning method.

Noted that in the previous works, scETDNN-DPD [8] and DPD based on canonical piecewise linear functions [12] are derived from GMP-DPD [2]. Thus, we already have the inclusion property $X_{GMP} \subset X_{scETDNN}$. Nevertheless, $|\sigma_j^{(1)}| = 1$ in (9) remains unclear.

C. STRUCTURAL RESTRICTION OF THE INPUT CONNECTION WHILE KEEPING THE GENERALIZED MEMORY POLYNOMIAL AS A LOWER BOUND

The inclusion property (9) may be able to be extended to an advanced inclusion property if we can loosen the restriction for $|\sigma_j^{(1)}|$ on ETDNN. In particular, setting $|\sigma_j^{(1)}|$ uniformly larger for all j leads to a spatial extension:

$$\begin{aligned} X_{GMP} &\subset X_{ETDNN||\sigma_j^{(1)}|=1} \subset X_{ETDNN||\sigma_j^{(1)}|=2} \cdots \\ &\subset X_{ETDNN||\sigma_j^{(1)}|=M+1} = X_{ETDNN} \end{aligned} \quad (10)$$

where $|\sigma_j^{(1)}| = M + 1$ means no restriction (full connection). (10) indicates that the trade-off between accuracy and computational complexity (increased by setting a larger $|\sigma_j^{(1)}|$) on ETDNN can be controlled by changing $|\sigma_j^{(1)}|$ while preserving a lower bound of X_{GMP}.

The utility of (10) was confirmed by conducting a simulation based on measurements made on a gallium nitride (GaN) Doherty PA. We used a supervised learning error as an estimator of the linearity of PA output. The measurement setting to obtain supervised data is the same as the measurement in Section IV and explained later. Algorithm 1 in the next subsection was used to calculate the supervised learning error for each $|\sigma_j^{(1)}|$.

Fig. 8 shows the relationship between the number of input connections ($|\sigma_j^{(1)}|$) and the normalized mean square error (NMSE) for training. We varied $|\sigma_j^{(1)}|$ from 1 to 10 uniformly for all j. The accuracy against the supervised data was evaluated by NMSE [15] and hereinafter we call the NMSE against the supervised data as the training NMSE. The figure also plots the training NMSE of GMP,

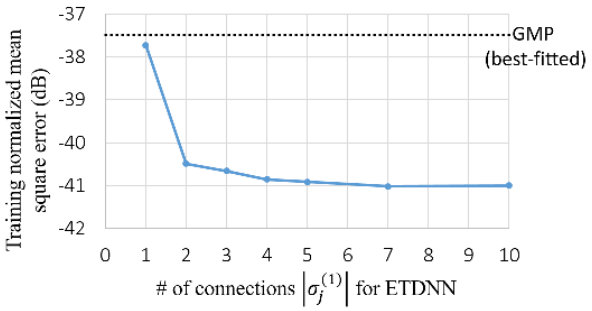

FIGURE 8. Relationship between $|\sigma_j^{(1)}|$ and training NMSE.

where parameters (M_{GMP}, P and D) were set to be large enough so that the training NMSE is minimized.

As can be seen in the figure, the calculated training NMSEs are strongly linked to (10). In short, the DPD model, which belongs to a wider space, has a better NMSE. In particular, the training NMSEs of ETDNN with the restriction of each $|\sigma_j^{(1)}|$ are lower than that of GMP, while X_{GMP} is the lowest bound in the relation (10). Moreover, Fig. 8 shows that a small $|\sigma_j^{(1)}|$ is enough from the viewpoint of training NMSE (2 is sufficient in this case), which means that changing $|\sigma_j^{(1)}|$ uniformly for all j is an efficient pruning strategy.

Considering the physical meaning of $|\sigma_j^{(1)}|$ while referring to Fig. 6 and 7, $|\sigma_j^{(1)}| = s$ means that s from $M+1$ connections to the delayed amplitudes $(x(k), x(k-1), \ldots, x(k-M))$ are chosen for each neuron. Moreover, Fig. 6 and 7 also indicate that the activation function ϕ corresponds to the gain function $f_{d,m}$ in GMP. Then, the selected s connections automatically correspond to the s cross-terms in the gain function, which implies that ETDNN model has a capacity of adapting the Volterra models that include the cross-terms in their gain functions. We will discuss this point in more detail later in Section IV C.

From these results, we find that the interpretation of the solution space (10) is useful for controlling the trade-off between accuracy and computational cost. Noted that to control the trade-off in (10) appropriately, conventional weight pruning, which can not control $|\sigma_j^{(1)}|$, is not suitable and a carefully designed pruning algorithm, which optimizes $\sigma_j^{(1)}$ and other parameters with a fixed $|\sigma_j^{(1)}|$, is needed. Next, we describe our new pruning algorithm with a fixed $|\sigma_j^{(1)}|$

D. PROPOSED PRUNING ALGORITHM WITH FIXED $|\sigma_j^{(1)}|$

Fig. 9 shows the flowchart of the pruning algorithm. Steps 2 and 3 in Fig. 9 are the same as steps 3 and 4 in Fig. 5, but there are two big differences compared with the conventional method, as follows: (i) our pruning procedure is divided into two parts. The first part is from step 1 to step 3, where we calculate the connections of the input layer $\sigma_j^{(1)}$ and their corresponding weights $w_{l,j}^{(1)}$ and biases $b_j^{(1)}$. In particular, in step 1, the training contains the structural

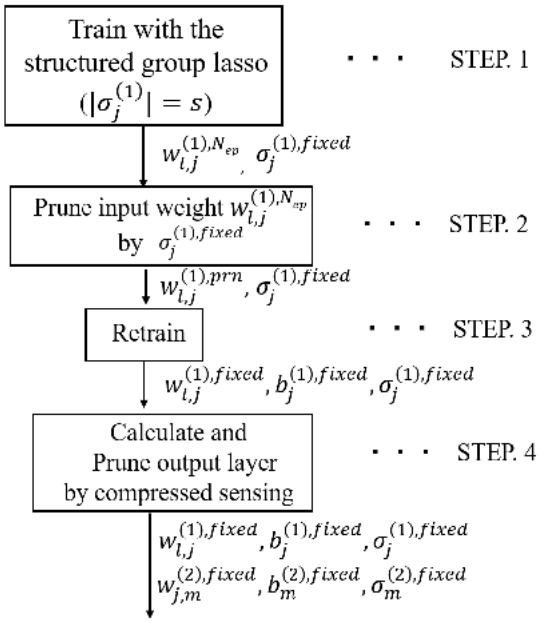

FIGURE 9. Flowchart of proposed pruning algorithm.

restriction of $|\sigma_j^{(1)}| = s$ in group lasso regularization [14]. The second part is step 4, where we calculate the connections of the output layer $\sigma_m^{(2)}$ and their corresponding weights $w_{j,m}^{(2)}$ and biases $b_m^{(2)}$. (ii) We explain each step below.

STEP 1: TRAINING WITH THE STRUCTURED GROUP LASSO FOR FIXED $|\sigma_j^{(1)}|$

Algorithm 1 Training With the Structured Group Lasso for Fixed $|\sigma_j^{(1)}|(= s)$

Input: $w_{l,j}^{(1),0}$, $b_j^{(1),0}$, $w_{j,m}^{(2),0}$, $b_m^{(2),0}$, f_0, λ_0, α, N_{ep}, s

1: **for** $i = 0$ to $N_{ep} - 1$ **do**

2: $w_{l,j}^{(1),i+1}, b_j^{(1),i+1}, w_{j,m}^{(2),i+1}, b_m^{(2),i+1}$
$= \text{OnestepGD}(f_i, w_{l,j}^{(1),i}, b_j^{(1),i}, w_{j,m}^{(2),i}, b_m^{(2),i})$

3: $\sigma_j^{(1),i+1} = \underset{l \in \forall\{l\}}{\text{argmax}} : s\{|w_{l,j}^{(1),i+1}|\}$

4: $\lambda_{i+1} = \alpha\lambda_i$

5: $f_{i+1} = E + \sum_{j=1}^{N_{total}} \Big(\sum_{l \in \sigma_j^{(1),i+1}} \lambda_{i+1} |w_{l,j}^{(1),i+1}| + \sum_{l' \notin \sigma_j^{(1),i+1}} \lambda_0 |w_{l',j}^{(1),i+1}| \Big)$

6: **end for**

7: $\sigma_j^{(1),fixed} = \sigma_j^{(1),N_{ep}}$

8: **return** $w_{l,j}^{(1),N_{ep}}, b_j^{(1),N_{ep}}, w_{j,m}^{(2),N_{ep}}, b_m^{(2),N_{ep}}, \sigma_j^{(1),fixed}$

The training algorithm (Algorithm 1) is run in step 1. The initial weights and biases $w_{l,j}^{(1),0}$, $b_j^{(1),0}$, $w_{j,m}^{(2),0}$, $b_m^{(2),0}$ and the initial regulation coefficient λ_0, damping coefficient α, initial objective function f_0, and epoch number N_{ep} are input.

The parameters of the weights $w_{l,j}^{(1)}, w_{j,m}^{(2)}$, the biases $b_j^{(1)}, b_m^{(2)}$, and the temporary connection $\sigma_j^{(1)}$ are iteratively updated N_{ep} times by repeating lines 2 to 5. Here, to distinguish the parameters in each iteration, the weight $w_{l,j}^{(1)}$ after the i-th iteration is described as $w_{l,j}^{(1),i}$. The same applies to other parameters.

On line 2, using the current objective function f_i, the weights $w_{l,j}^{(1),i}, w_{j,m}^{(2),i}$, and the biases $b_j^{(1),i}, b_m^{(2),i}$, a gradient descent method (described as a function of OnestepGD) is applied at one step to update the weights and biases. Note that we can use various gradient descent methods, such as stochastic gradient descent or Adam [18].

On line 3, using the weights $w_{l,j}^{(1),i+1}$ updated by OnestepGD, temporary connections $\sigma_{l,j}^{(1),i+1}$ for the $i+1$-th iteration are chosen for all j by the operation argmax : s which is defined as the indices of the top s (especially, argmax : $1 =$ argmax). Thus, argmax : $s\{|w_{l,j}^{(1),i+1}|\}$ determines the indices of the updated input weights whose absolute values $|w_{l,j}^{(1),i+1}|$ are the top s of the j-th node in the hidden layer.

On line 4, the regularization parameter λ_i is updated to λ_{i+1} by multiplying the damping coefficient α. To decrease λ_{i+1} gradually, α should be set as a little smaller than 1 (such as 0.999).

On line 5, the current objective function f_i is updated to f_{i+1}, which contains the typical mean square error E and the group lasso regularization. The regularization coefficients λ_{i+1} are only applied to the weights $|w_{l,j}^{(1),i+1}|$ whose indices l belong to $\sigma_j^{(1),i+1}$. The other regularization coefficients are set to λ_0, which is larger than λ_{i+1}.

This iterative update of the group lasso regularization is simultaneously used for training and pruning of $\sigma_j^{(1),fixed}$ since the gradual decrease in λ_{i+1} from one iteration to the next gradually reduces $w_{l,j}^{(1),i+1} \mid l \notin \sigma_j^{(1),i+1}$, where $\sigma_j^{(1),i+1}$ is the pruned group at the $i+1$-th iteration, while the training gradually improves the accuracy of $\sigma_j^{(1),i+1}$.

Once the N_{ep}-th iteration is reached, the training stops and the group of $\sigma_j^{(1),N_{ep}}$ is output as $\sigma_j^{(1),fixed}$ (line 7). Additionally, the trained input weight $w_{l,j}^{(1),N_{ep}}$ is output and used in step 2.

STEP 2: PRUNING INPUT WEIGHTS

In step 2, the input weights are pruned by using $\sigma_j^{(1),fixed}$:

$$w_{l,j}^{(1),prn} = \begin{cases} w_{l,j}^{(1),N_{ep}} & (if\ l \in \sigma_j^{(1),fixed}) \\ 0 & (otherwise) \end{cases} \tag{11}$$

where $w_{l,j}^{(1),prn}$ is the input weight after pruning and a value of 0 means no connection (discarded).

The important point is that the pruned group $\sigma_j^{(1),fixed}$ in (11) equals the temporary pruned group of $\sigma_j^{(1),N_{ep}}$ at the end of training, which means the weights $w_{l,j}^{(1),N_{ep}} \mid l \notin \sigma_j^{(1),fixed}$ are already suppressed through the training with the group lasso regularization. This leads to a lower pruning error, which increases modeling accuracy.

STEP 3: RE-TRAINING

To reduce the error caused by the pruning in step 2, re-training using the objective function $f_{N_{ep}}$ is performed while keeping the pruned connections (11) to obtain re-trained weights ($w_{l,j}^{(1)}$ and $w_{j,m}^{(2)}$) and biases ($b_j^{(1)}$ and $b_m^{(2)}$). The re-trained input weights $w_{l,j}^{(1)}$ and the biases $b_j^{(1)}$ are then fixed, i.e., renamed $w_{l,j}^{(1),fixed}$ and $b_j^{(1),fixed}$, respectively.

STEP 4: PRUNING THE OUTPUT WEIGHTS AND BIASES BY COMPRESSED SENSING

The connections of the input layer and their corresponding weights and biases are fixed, i.e., renamed as $\sigma_j^{(1),fixed}$, $w_{l,j}^{(1),fixed}$ and $b_j^{(1),fixed}$, respectively. Then, ETDNN (1) is reformulated by moving $x(k-m)$ inside the summations as:

$$\sum_{m=0}^{M} \left\{ \sum_{j=1}^{N_{total}} w_{j,m}^{(2)} \phi \left(\sum_{l \in \sigma_j^{(1),fixed}} w_{l,j}^{(1),fixed} |x(k-l)| + b_j^{(1),fixed} \right) x(k-m) + b_m^{(2)} x(k-m) \right\} \tag{12}$$

where the free parameters are only $w_{j,m}^{(2)}$ and $b_m^{(2)}$. The other parameters are fixed. Thus, the pruning output weights and biases can be interpreted as a linear problem with a restricted number of non-zero parameters of $w_{j,m}^{(2)}$ and $b_m^{(2)}$. This problem is the same as in compressed sensing. Thus, least squares orthogonal matching pursuit (LS-OMP) [16] is used to calculate $\sigma_m^{(2)}$, $w_{j,m}^{(2)}$, and $b_m^{(2)}$ and these are output as $\sigma_m^{(2),fixed}$, $w_{j,m}^{(2),fixed}$, and $b_m^{(2),fixed}$.

Finally, we obtain all of the weights and biases and their pruned connections under the condition $|\sigma_j^{(1)}| = s$. We expected that the proposed pruning would produce a more compact structure on ETDNN than conventional weight pruning because the condition $|\sigma_j^{(1)}| = s$ is efficiently included in the training as the group lasso regularization.

IV. MEASURED RESULTS

Fig. 10 shows the experimental setup. We used a 15-MHz LTE signal with 96 MSa/s and peak-to-average power ratio (PAPR) of 7.3 dB. The signal was up-converted to 3.5 GHz and output from the signal generator (E8267D, Keysight). The 3.5-GHz signal was fed to a 3.5-GHz-band GaN PA, which consisted of a driver amplifier and Doherty PA with a 63 dB total gain and 50 W average output power. The PA output was fed to a down converter (M9362A, Keysight) with a 3.27-GHz local frequency. The PA output was then down-converted from 3.5 GHz to 230 MHz, and this down converted signal was digitalized with an ADC (M9202A, Keysight). Finally, the I and Q signals were acquired by the PC and used in the DPD calculation.

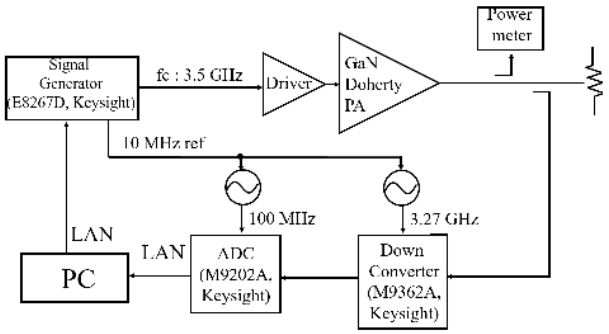

FIGURE 10. Measurement set up of 15MHz LTE in 3.5 GHz band.

To obtain supervised data for the DPD training, we applied iterative learning control (ILC) [7], [17], which can perfectly compensate for the non-modeled distortion for the limited signal length (256,000 samples in this measurement) by updating the input sampling iteratively. The average PA output power was 52.5 W in each iteration of ILC. The validation used test data (256,000 samples) that was different from the training data.

We used two indicators of computational complexity. One was floating point operations (FLOPs) [19]. The other was the number of real-valued free parameters, which means that one complex-valued free parameter was counted as two real-valued free parameters. Additionally, we used NMSE [15] and ACLR as indicators of remaining distortion. Noted that the NMSE in this measurement is referring to the error between PA output and DPD input, which is different from the (previously defined) training NMSE which is the error between the supervised data and DPD input. ACLR was evaluated as the average of the lower ACLR and upper ACLR with a 15 MHz offset.

Our proposed ETDNN-DPD was compared with scETDNN-DPD [8], RVTDNN [4] and ARVTDNN [5]. ReLU was used as the activation function ϕ for all. Moreover, the number of memory taps M was fixed to 20, the optimal value. N_{total} in ETDNN (1) was varied from 12 to 200. G in (2) was varied from 3 to 40. In RVTDNN, we set two layers whose numbers of the neurons were varied from 40 to 240 and the best combinations of parameters were chosen from the viewpoint of the trade-off between FLOPs and training NMSE. In ARVTDNN, we set one layer whose number of the neurons was varied from 20 to 160 and $|x(k-m)|$, $|x(k-m)|^2$ and $|x(k-m)|^3$ were added as the amplitude terms to the input layer. For optimizations of the neural networks based DPDs, we used Adam [18]; the epoch number N_{ep} was 6000, and the batch size was 320.

The proposed structured pruning (Algorithm 1) was evaluated for ETDNN-DPD with the number of input connections $|\sigma_j^{(1)}|(=s) = 1$ and 2. The trade-off between FLOPs and ACLR was controlled by varying the number of non-zero values in LS-OMP [16]. In this case, we found that the training NMSEs were saturated at 70 and 100 non-zero values for $s = 1$ and $s = 2$ in LS-OMP, respectively. Thus, we stopped LS-OMP at 70 and 100 times iteration for $s = 1$ and $s = 2$, respectively. Then to control the trade-off between FLOPs and ACLR, the number of non-zero weights and biases in step 4 of Fig. 9 was varied from 10 to 70 for $s = 1$ and from 30 to 100 for $s = 2$. We set the initial value of λ_0 and the damping coefficient α in Algorithm 1 to 1.6×10^{-2} and 0.999, respectively. Noted that Algorithm 1 was not sensitive to the initialization. In fact, the maximum difference of the training NMSEs in 10 trials with random initial values was only 0.3 dB. After pruning, nodes having no input connections or output connections were automatically discarded.

We also compared Volterra based DPDs (MP-DPD and GMP-DPD). The coefficients of MP and GMP were optimized by using the least squares method. For a fair comparison, the parameters of the memory taps M, cross terms D, and the order of the polynomial P in MP and GMP were varied and the best combinations of parameters were chosen from the viewpoint of the trade-off between FLOPs and ACLR in the same manner as [19]. Moreover, we used the same FLOPs estimation as [19] for MP and GMP.

A. COMPARISON OF DIGITAL PREDISTORTIONS W/WWWW/AND W/O PROPOSED PRUNING

Fig. 11 and 12 show the relationship between FLOPs and ACLR and the relationship between the number of free parameters and ACLR.

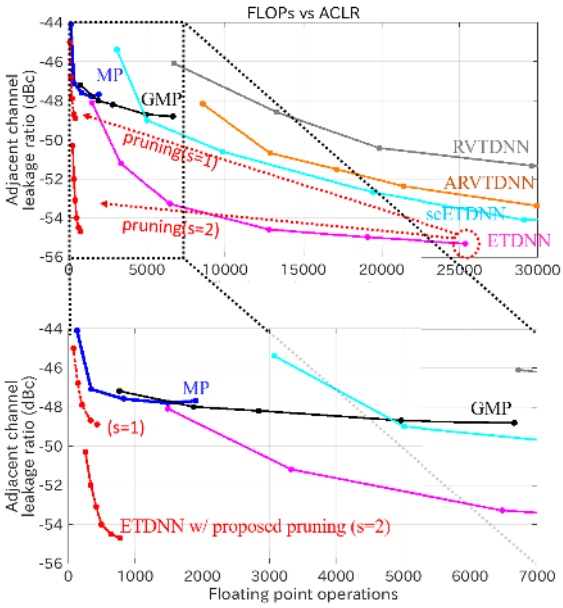

FIGURE 11. Relationship between FLOPs and ACLR for MP, GMP, RVTDNN, ARVTDNN, scETDNN, ETDNN w/o pruning and ETDNN w/ proposed pruning ($s = 1, 2$).

Comparing conventional scETDNN-DPD (light blue line) with proposed ETDNN-DPD w/o pruning (magenta line), we can see that ETDNN-DPD required the fewer FLOPs and free parameters than scETDNN-DPD did for the same ACLR. This means that ETDNN-DPD, which shares the connections

TABLE 1. Summary of measurements for the best of each DPD.

	ACLR(dBc)		NMSE(dB)	FLOPs	# of free parameters (real-valued)	ReLU
	lower	upper				
noDPD	-31.9	-33.6	-24.0	-	-	-
MP-DPD [1] ($M = 13, P = 13$)	-48.1	-47.5	-35.7	1488	364	-
GMP-DPD [2] ($M = 6, P = 10, D = 7$)	-48.9	-48.7	-38.4	6658	1596	-
ARVTDNN-DPD [5] amplitude terms $\|x(k)\|, \|x(k)\|^2, \|x(k)\|^3$	-53.6	-53.1	-40.4	34240	17282	160
scETDNN-DPD [8] w/o pruning	-53.9	-54.2	-41.5	29156	15162	630
ETDNN-DPD w/o pruning	-55.4	-55.2	-42.1	25366	12852	200
ETDNN-DPD w/ proposed pruning ($s = 1$)	-49.1	-48.7	-39.0	426	248	49
ETDNN-DPD w/ proposed pruning ($s = 2$)	-54.8	-54.4	-41.6	758	426	72

of the output layer while scETDNN-DPD does not, enabled a compact DPD modeling.

Additionally, compared with conventional RVTDNN-DPD (gray line) and ARVTDNN-DPD (orange line), we can also see that ETDNN-DPD required the fewer FLOPs and free parameters than they did for the same ACLR. That is because the distortion was caused by only PA in this measurement setting (there was no I/Q imbalance and dc offset). Thus, the constraints of the physical modeling of RF PAs [20], [21], which ETDNN satisfies but RVTDNN and ARVTDNN do not, is of considerable importance in a compact DPD modeling.

Moreover, comparing ETDNN-DPD w/o pruning (magenta line) and w/ proposed pruning (red dashed line for $s = 1$, red solid line for $s = 2$) reveals that the proposed pruning achieved a further reduction in FLOPs and number of free parameters while maintaining ACLR. Especially in the best cases of pruned ETDNN-DPD ($s = 2$ and the number of non-zero values in step 4 of Fig. 9 was set to 100), the degradation in performance due to the pruning was negligible despite the much fewer FLOPs.

Table 1 summarizes the results of DPD methods in Fig. 12 and 13. In the estimation of the computational complexity of Table 1, we chose the plots that achieved the lowest ACLR in the measurements measured ACLR, not training NMSE in order to avoid the over-fitting problem. The output spectrum corresponding to Table 1 is also shown in Fig. 13. As we can see from the table, ETDNN-DPD with the proposed pruning ($s = 1$ and 2) achieved a lower ACLR than that of GMP-DPD, while the required FLOPs and number of free parameters were much lower than that of GMP-DPD. This result is linked to the relationship of the solution spaces in (10) and ensures the correctness of the analysis.

Table 1 also reveals the total reduction in computational complexity had by combining proposed ETDNN-DPD with the proposed pruning. Compared with conventional scETDNN, ETDNN-DPD with the proposed pruning ($s = 2$) required about one thirtieth the FLOPs and free parameters

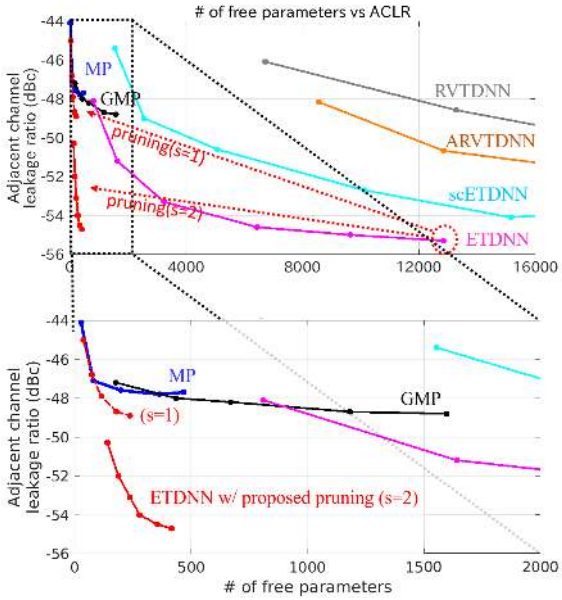

FIGURE 12. Relationship between the number of real-valued free parameters and ACLR for MP, GMP, RVTDNN, ARVTDNN, scETDNN, ETDNN w/o and ETDNN w/ proposed pruning ($s = 1, 2$).

(29156 versus 758 and 15162 versus 426) for almost the same NMSE and ACLR.

We can also verify that thanks to the dramatic reduction in computational complexity (required FLOPs (758) and number of free parameters (426)), the computational complexity became low enough relative to the Volterra based DPDs such as MP-DPD and GMP-DPD, which are widely implemented on commercially available hardware because of the ease of their implementation. ETDNN-DPD with proposed pruning ($s = 2$) achieved a 7-dB lower ACLR (6-dB better NMSE) than that of MP-DPD while requiring lower FLOPs and only a few more free parameters. This shows that ETDNN-DPD with the proposed pruning is a promising candidate for 5G and beyond 5G wireless transmitters

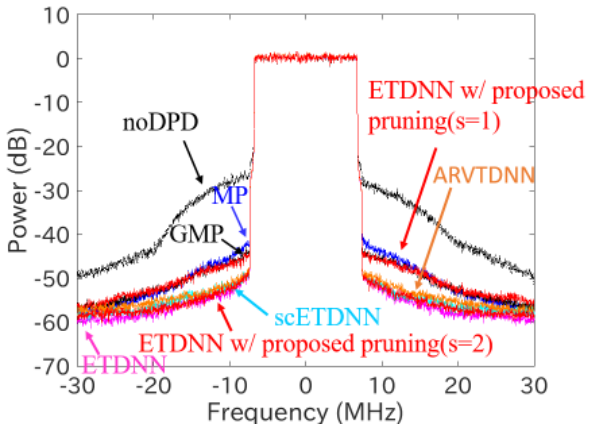

FIGURE 13. Output spectrum with MP, GMP, ARVTDNN, scETDNN, ETDNN w/o pruning, ETDNN with proposed pruning ($s = 1, 2$).

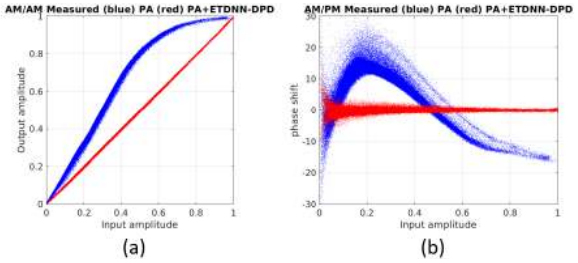

FIGURE 14. (a) AM-AM and (b) AM-PM characteristics of PA (blue dots) / PA+ETDNN-DPD with proposed pruning (s=2) (red dots).

Fig. 14 shows the AM-AM and AM-PM characteristics of the PA (red dots) and the PA + ETDNN-DPD (blue dots) with $s = 2$. We can see that ETDNN with the proposed pruning ($s = 2$) can efficiently compensate for the PA distortion which has strongly bent AM-AM curve and widely dispersed AM-PM plots. As shown in Fig. 14, we operated the GaN Doherty PA near the saturated point, where the PA causes strong distortion including memory effects, which results in the required FLOPs in this measurement being larger than in the previous experiments on MP, GMP [19], scETDNN [8], RVTDNN [4] and ARVTDNN [5].

B. COMPARISON OF PRUNING TECHNIQUES FOR ENVELOPE TIME-DELAY NEURAL NETWORK

To focus on the pruning performance, we compared the proposed pruning with weight pruning [10] in the same ETDNN-DPD. In weight pruning, to control the trade-off between FLOPs and ACLR, the thresholds $Th^{(1)}$ and $Th^{(2)}$ in (5) were set by multiplying the threshold rate, which was varied from 0.6 to 1.8, by the the standard deviations of the weights $w_{l,j}^{(1),N_{ep}}$ and $w_{j,m}^{(2),N_{ep}}$, respectively. Additionally, to maximize the performance of weight pruning, parameter optimized L1 regularization was applied, which corresponds to setting the objective function f_i shown on line 5 of Algorithm 1 as $\lambda_{i+1} = \lambda_0 = 8\times10^{-4}$ in each iteration. On the other hand, the setting and the results of the proposed pruning were the same as in the previous measurements (Fig. 12 and Fig. 13). Other settings such as the number of epochs and batches for both pruning were also the same as in the previous measurements for a fair comparison.

Fig. 15 and 16 show the relationship between FLOPs and ACLR and the relationship between the number of free parameters and ACLR, respectively. They show that the proposed pruning with $s = 2$ has about one-third the FLOPs and number of free parameters compared with weight pruning for the same ACLR. These results verify that our pruning is more efficient than weight pruning for ETDNN-DPD.

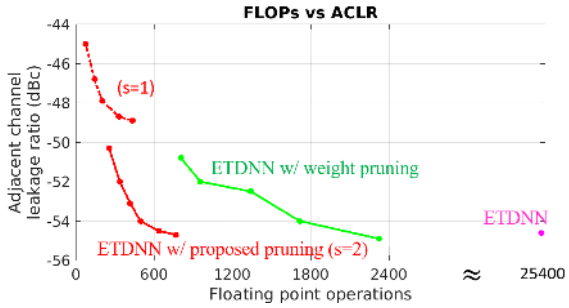

FIGURE 15. Relationship between FLOPs and ACLR for ETDNN with weight pruning [10] and proposed pruning ($s = 1, 2$).

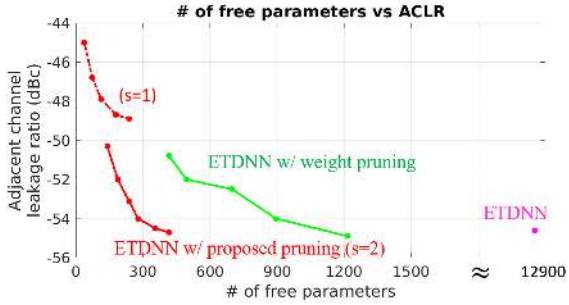

FIGURE 16. Relationship between the number of free parameters and ACLR for ETDNN with weight pruning [10] and proposed pruning ($s = 1, 2$).

There are mainly two reasons why our pruning performs better. One is that the training with group lasso regularization extracts more appropriate connections on ETDNN, whereas the training of the weight pruning uses simple L1 regularization. The other is derived from the structural characteristic. The inclusion property of the solution space (10) ensures the lower bound of our proposed pruning, whereas weight pruning does not have such a lower bound, which have degraded its performance. We will discuss the structure of ETDNN from the viewpoint of device modeling in the next section.

C. DISCUSSION OF ENVELOPE TIME-DELAY NEURAL NETWORK WITH PROPOSED PRUNING

Let us consider why ETDNN-DPD with the proposed pruning especially with $s = 2$ performed better than other DPDs from the physical viewpoint.

Recently a power-dependent Volterra (PDV) model [22], which was derived from the circuit-level analysis [23],

was proposed as the extension of GMP. The formulation of PDV is as follows:

$$\begin{aligned} z(k) = & \sum_{m=0}^{M_{PDV}} \hat{h}_1(m)x(k-m) \\ & + \sum_{m=0}^{M_{PDV}} \sum_{q_2=0}^{Q_2} \hat{h}_3(m,q_2)|x(k-m-q_2)|^2 x(k-m) \\ & + \sum_{m=0}^{M_{PDV}} \sum_{q_2=0}^{Q_2} \sum_{q_3=0}^{Q_3} \hat{h}_5(m,q_2,q_3)|x(k-m-q_2)|^2 \\ & \times |x(k-m-q_3)|^2 x(k-m) + \ldots \end{aligned} \tag{13}$$

where Q_2 and Q_3 are the numbers of cross-terms and M_{PDV} is the number of memory taps in PDV. We can see that the even-order envelope powers are also present in the GMP model. To simplify the formulation, (13) is combined with $x(k-m)$ as follows:

$$\begin{aligned} z(k) = & \sum_{m=0}^{M_{PDV}} \sum_{q_2=0}^{Q_2} \sum_{q_3=0}^{Q_3} f_{q_2,q_3,m}(|x(k-m-q_2)|, \\ & |x(k-m-q_3)|)x(k-m) \end{aligned} \tag{14}$$

where $f_{q_2,q_3,m}(|x(k-m-q_2)|, |x(k-m-q_3)|)$ is a function of two variables, including $\hat{h}_1(m)$, $\hat{h}_3(m,q_2)|x(k-m-q_2)|^2$ and $\hat{h}_5(m,q_2,q_3)|x(k-m-q_2)|^2|x(k-m-q_3)|^2$.

As can be easily understood, (14) for specific q_2, q_3, and m can be visualized as in Fig. 17. Here, the phase filter, which multiplies the output of the function $f_{q_2,q_3,m}(|x(k-m-q_2)|, |x(k-m-q_3)|)$ by $x(k-m)$, is in common with that of ETDNN (Fig. 1) for specific m and j in (1). Thus, we can see that an only difference between PDV and ETDNN is the function before the phase filter.

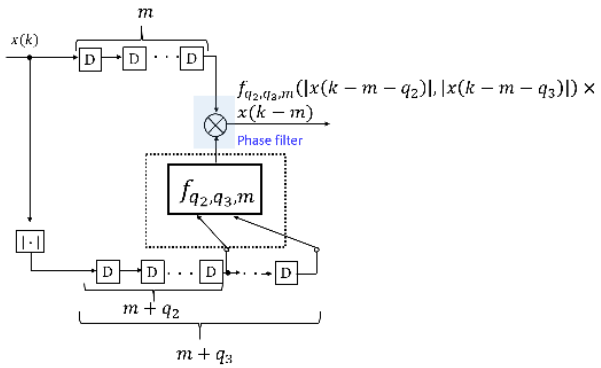

FIGURE 17. Fundamental block of PDV for specific q_2, q_3 and m.

Here, in the same manner as (8), the universal approximation theorem of neural networks [13] can be applied to $f_{q_2,q_3,m}(|x(k-m-q_2)|, |x(k-m-q_3)|)$:

$$\begin{aligned} & f_{q_2,q_3,m}(x,y) \\ & \approx \sum_{j=1}^{N} \check{w}^{(2)}_{j,q_2,q_3,m} \phi\left(\check{w}^{(1)}_{j,q_2,q_3,m} x + \tilde{w}^{(1)}_{j,q_2,q_3,m} y + \check{b}^{(1)}_{j,q_2,q_3,m}\right) \\ & \quad + \check{b}^{(2)}_{q_2,q_3,m} \end{aligned} \tag{15}$$

The approximation (15) is visualized in Fig. 18. As we can see, the function $f_{q_2,q_3,m}(|x(k-m-q_2)|, |x(k-m-q_3)|)$, which has two variables $|x(k-m-q_2)|$ and $|x(k-m-q_3)|$, is decomposed into a summation of neurons whose number of input connections are all two, which can be rephrased as $|\sigma_j^{(1)}| = 2$ for all j.

In light of Fig. 17 and Fig. 18 under the condition of $M = M_{PDV} + Q_2 + Q_3$ in (1) and (14), we can summarize that the approximate PDV shares the structures of the neurons and the filter with ETDNN except for the restriction of $|\sigma_j^{(1)}| = 2$ on the neurons, which results in the relationship:

$$X_{PDV} \subset X_{ETDNN||\sigma_j^{(1)}|=2} \tag{16}$$

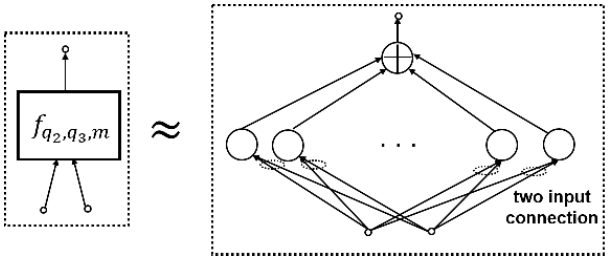

FIGURE 18. Applying the universal approximation theorem of neural networks to individual functions in PDV.

This relationship ensures that ETDNN with $|\sigma_j^{(1)}| = 2$ includes PDV as a lower bound. Moreover, considering that PDV includes GMP, we can conclude that the PDV relationship (16) gives a stricter bound than that of GMP (10). This means that the PDV model is much close to the ETDNN model with $|\sigma_j^{(1)}| = 2$. Therefore, we expect that the analysis of the PDV model directly corresponds to that of the ETDNN model with $|\sigma_j^{(1)}| = 2$, which gives the meaning of ETDNN with $|\sigma_j^{(1)}| = 2$ from the viewpoint of device modeling.

V. CONCLUSION

An envelope time-delay neural network (ETDNN)-DPD and a pruning technique for it were proposed. ETDNN-DPD complies with the physical characteristics of RF PAs and uses a more compact DPD model compared with the conventional neural-network-based DPD. A detailed analysis of ETDNN based on the essence of GMP led to an efficient pruning, which enables the neural network for ETDNN-DPD to have a sparse structure. In a measurement of a 3.5-GHz GaN Doherty PA, ETDNN-DPD with the proposed pruning had one-thirtieth the computational complexity of the conventional neural-network-based DPD for the same ACLR. Moreover, ETDNN-DPD with the proposed pruning achieved a 7-dB improvement in ACLR compared with the conventional MP-DPD, even though it has a lower computational complexity. This ensures that ETDNN-DPD with the proposed pruning can be implemented in commercially available hardware and will contribute to development of future mobile communication systems through its high compensation performance.

In the future work, for the purpose of practical use, we will check the stability of ETDNN-DPD together with the proposed pruning under various PA conditions. In addition, we will analyze the PA model corresponding to ETDNN from the viewpoint of device modeling, which will also contribute to the analysis of its stability. Moreover, since our proposed pruning was limited to perform on the off-line processing by using the specific supervised data, we also need to check whether our proposed pruning is still optimal or not even if the input signal changes its characteristics.

REFERENCES

[1] L. 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 Trans. Commun.*, vol. 52, no. 1, pp. 159–165, Jan. 2004.

[2] 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.

[3] A. Zhu, J. C. Pedro, and T. J. Brazil, "Dynamic deviation reduction-based volterra behavioral modeling of RF power amplifiers," *IEEE Trans. Microw. Theory Techn.*, vol. 54, no. 12, pp. 4323–4332, Dec. 2006.

[4] M. Rawat and F. M. Ghannouchi, "A mutual distortion and impairment compensator for wideband direct-conversion transmitters using neural networks," *IEEE Trans. Broadcast.*, vol. 58, no. 2, pp. 168–177, Jun. 2012.

[5] 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, Jan. 2019.

[6] P. Jaraut, M. Rawat, and F. M. Ghannouchi, "Composite neural network digital predistortion model for joint mitigation of crosstalk, I/Q imbalance, nonlinearity in MIMO transmitters," *IEEE Trans. Microw. Theory Techn.*, vol. 66, no. 11, pp. 5011–5020, Nov. 2018.

[7] K. Li, N. Guan, and H. Wang, "Iterative learning control assisted neural network for digital predistortion of MIMO power amplifier," in *Proc. IEEE 87th Veh. Technol. Conf.*, Porto, Portugal, Jun. 2018, pp. 1–5.

[8] Z. Yu, "A generalized digital predistortion model based on artificial neural networks," in *Proc. Asia–Pacific Microw. Conf. (APMC)*, Nov. 2018, pp. 935–937.

[9] Y. Zhang, Y. Li, F. Liu, and A. Zhu, "Vector decomposition based time-delay neural network behavioral model for digital predistortion of RF power amplifiers," *IEEE Access*, vol. 7, pp. 91559–91568, 2019.

[10] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[11] A. Zhu, "Behavioral modeling for digital predistortion of RF power amplifiers: From volterra series to CPWL functions," in *Proc. IEEE Topical Conf. Power Model. Wireless Radio Appl. (PAWR)*, Austin, TX, USA, Jan. 2016, pp. 1–4.

[12] Z. Yu, W. Wang, and G. Shang, "A generalized model based on canonical piecewise linear functions for digital predistortion," in *Proc. Asia–Pacific Microw. Conf. (APMC)*, Dec. 2016, pp. 978–981.

[13] S. Sonoda and N. Murata, "Neural network with unbounded activation functions is universal approximator," *Appl. Comput. Harmon. Anal.*, vol. 43, no. 2, pp. 233–268, Sep. 2017.

[14] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. Roy. Stat. Soc., Ser. B Stat. Methodol.*, vol. 68, no. 1, pp. 49–67, Feb. 2006.

[15] F. Mkadem, "Behavioural modeling and linearization of RF power amplifier using artificial neural networks," M. S. thesis, Dept. Elect. Eng., Univ. Waterloo, Waterloo, ON, Canada, 2010.

[16] M. Elad, "Pursuit algorithms–Practice," in *Sparse and Redundant Representations*. New York, NY: Springer New York, 2010.

[17] J. Chani-Cahuana, P. N. Landin, C. Fager, and T. Eriksson, "Iterative learning control for RF power amplifier linearization," *IEEE Trans. Microw. Theory Techn.*, vol. 64, no. 9, pp. 2778–2789, Sep. 2016.

[18] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015, pp. 1–7.

[19] 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 Techn.*, vol. 58, no. 6, pp. 1510–1520, Jun. 2010.

[20] E. G. Lima, T. R. Cunha, H. M. Teixeira, M. Pirola, and J. C. Pedro, "Baseband derived volterra series for power amplifier modeling," in *IEEE MTT-S Int. Microw. Symp. Dig.*, Boston, MA, USA, Jun. 2009, pp. 1361–1364.

[21] E. G. Lima, T. R. Cunha, and J. C. Pedro, "A physically meaningful neural network behavioral model for wireless transmitters exhibiting PM–AM/PM–PM distortions," *IEEE Trans. Microw. Theory Techn.*, vol. 59, no. 12, pp. 3512–3521, Dec. 2011.

[22] C. Crespo-Cadenas, M. J. Madero-Ayora, and J. A. Becerra, "On the power level dependence of PA and DPD volterra models," in *Proc. IEEE Topical Conf. RF/Microwave Power Model. Radio Wireless Appl. (PAWR)*, San Antonio, TX, USA, Jan. 2020, pp. 18–21.

[23] C. Crespo-Cadenas, J. Reina-Tosina, M. J. Madero-Ayora, and M. Allegue-Martinez, "A volterra-based procedure for multi-port and multi-zone GaN FET amplifier CAD simulation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 11, pp. 3022–3032, Nov. 2013.

MASAAKI TANIO received the B.E. and M.E. degrees in mathematical engineering from The University of Tokyo, Tokyo, Japan, in 2007 and 2009, respectively.

He joined NEC Corporation, in 2009, and has been researching and developing RF power amplifiers and digital signal-processing techniques for wireless communications. His current interests include neural-network-based digital predistortion techniques and delta-sigma modulations. He is currently a Senior Researcher with the System Platform Research Laboratories, NEC Corporation. He is a member of the Institute of Electronics, Information, and Communication Engineers (IEICE), Japan.

NAOTO ISHII received the B.E., M.E., and Ph.D. degrees in electrical engineering from Yokohama National University, in 1992, 1994, and 1997, respectively.

He joined NEC Corporation, Japan, in 1997. He is currently a Research Manager with the System Platform Research Laboratories, NEC Corporation. His current research interests include signal processing and radio resource management for mobile wireless communication systems.

NORIFUMI KAMIYA (Member, IEEE) received the B.E. degree from Yokohama National University, in 1990, and the Ph.D. degree from The University of Tokyo, in 2000, all in electrical engineering.

He is currently a Senior Principal Researcher with the System Platform Research Laboratories, NEC Corporation, Japan. His current research interests include signal processing and coding for wireless and optical communication systems.