A Tunable Neural Network based Decision Feed-back Equalizer model for High-speed Link Simulation

Thong Nguyen and Jose Schutt-Aine
Department of Electrical and Computer Engineering.
University of Illinois Urbana-Champaign
Urbana, IL 61801, USA
tnnguye3, jesa@illinois.edu

***Abstract*—This paper presents a model combining a feed-forward neural network (FNN) with a recurrent neural network (RNN) to model Decision Feed-back Equalizer (DFE). By using the FNN as the mapping between the tap values and the dynamic behavior of the DFE, a complete model of the DFE can be constructed for channel simulation. The paper shows a 2-tap DFE example in which excellent agreement between the model generated by the proposed method and transient simulation can be observed.**

I. Introduction

Losses and dispersion caused by passive channels in high-speed serial links are compensated by equalization circuits. Receiver equalization consists of continuous-time linear equalization (CTLE) and decision feed-back equalization (DFE). CTLE is linear and typically modeled as rational transfer functions which can also be incorporated into the channel while DFE is nonlinear, hence, needs separate handling. Leveraging success of deep learning methods in time-series data modeling, [1], [2] uses an Elman RNN [3] to model high-speed link buffers. However, the equalization parameters were fixed when the RNN learns the input - output mapping in [2]. In this work, we extend the work done in [2], allowing modeling a complete model of DFE using an FNN followed by the Elman RNN for sequence mapping.

In the next section, a brief description DFE is presented to recall the problem at hand. Section III is dedicated to explain the details of our proposed architecture allowing a complete DFE *tunable* model. A 2-tap DFE example is shown in Section IV to demonstrate the effectiveness of the proposed method. It also serves as a step-by-step guide on how to train and use the proposed model for simulation. Conclusion and possible further extension of this work is presented in Section V.

II. Receiver decision feedback equalization (DFE)

A passive, lossy channel acts similarly to a low pass filter in the sense that the higher the frequency component, the more loss happens. As the result, a sharp rising/falling signal after passing through a channel will lose its high frequency contents which makes its rise/fall time slower, the signal spreads out longer. In most cases, it could spread out to multiple bit periods (also known as unit interval, or UI) causing a phenomenon called intersymbol interference (ISI).

Figure 1 shows how a bit error could happen due to ISI. The figure was exaggerated for illustration purposes. When the output of the prior bit (in this case, a logical 1) spreads out, it interferes with the next bit (in this case, a logical 0), alters the voltage waveform leading to the wrong logical level recognized.

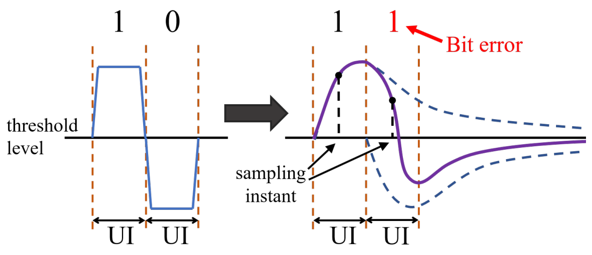

Fig. 1. Bit error caused by ISI.

The basic principle of DFE is using a slicer circuit to sample and quantize the input waveform at locations that are multiple UIs away from each other, normally known as the cursor values (a_i's), then using these cursor values, the DFE circuit subtracts ISI from the incoming signal via a feedback FIR filter

$$v_{out}[n] = v_{in}[n] - \sum_{i=1}^{N} a_i d_{RX}[n-i] \tag{1}$$

Figure 2 shows an example of different number of tap DFE circuits on an unequalized waveform. Notice how the number of abrupt drops in the equalized waveform matches the number of DFE taps used. It is also important to notice that the drops associated with higher tap numbers are smaller. This is due to the fact that the higher order post-cursors are always smaller in magnitude.

III. Proposed neural network architecture

In this work, under the assumption that the DFE effect is K time step dependent, we use an Elman RNN to represent the

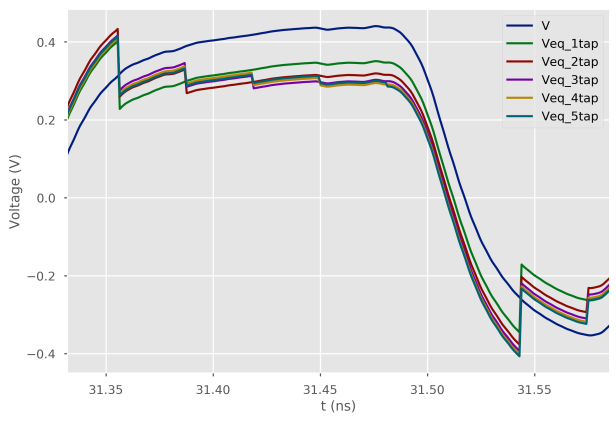

Fig. 2. Example waveform under DFE effect with different number of taps

mapping between a K-step windowed sample of inputs and the next time step output. Mathematically, the K time step truncated full stack RNN is

$$\begin{cases} \boldsymbol{h_t} &= g_h^{(L)} \circ g_h^{(L-1)} \circ \cdots \circ g_h^{(2)} \circ g_h^{(1)} \left(\boldsymbol{x_t}, \boldsymbol{h_0}\right) \\ y_t &= g_o\left(\boldsymbol{h_t}\right), \end{cases} \tag{2}$$

where $A \circ B = A\left(B\left(x\right)\right)$ is the composition operation, $\boldsymbol{x_t} = \begin{bmatrix} x_{t-(K-1)} & x_{t-(K-2)} & \cdots & x_t \end{bmatrix} \in \mathbb{R}^{d_1 \times K}$, and $\boldsymbol{h_t} = \begin{bmatrix} h_{t-(K-1)} & h_{t-(K-1)} & \cdots & h_t \end{bmatrix} \in \mathbb{R}^{l_h \times K}$, $h_t \in \mathbb{R}^{l_h}$ where l_h is the dimension of the hidden state at time step t, $g_h\left(\cdot\right)$ represents the long-short term memory (LSTM) mapping which reads

$$\begin{cases} i_t &= \sigma\left(W_{ii}x_t + W_{hi}h_{t-1}\right) \\ f_t &= \sigma\left(W_{if}x_t + W_{hf}h_{t-1}\right) \\ g_t &= \tanh\left(W_{ig}x_t + W_{hg}h_{t-1}\right) \\ o_t &= \sigma\left(W_{io}x_t + W_{ho}h_{t-1}\right) \\ c_t &= f_t c_{t-1} + i_t g_t \\ h_t &= o_t \tanh\left(c_t\right), \end{cases} \tag{3}$$

where h_t is hidden state at time t, c_t is called the cell state, and i_t, f_t, g_t and o_t are the input, forget, cell and output gates respectively. All of the W's are learnable weight matrices., $\boldsymbol{h_0} \in \mathbb{R}^{l_h \times L}$ is the initial state for every K time step dynamic.

The gradient of the loss function of a 1 layer, K time step unrolled RNN at time t w.r.t. any parameter θ can be written as sum of the loss function within the most recent K time steps

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{\tau=t-(K-1)}^{t} \frac{\partial \mathcal{L}_\tau}{\partial \theta}, \tag{4}$$

where

$$\frac{\partial \mathcal{L}_\tau}{\partial \theta} = \sum_{j=t-(K-1)}^{\tau} \frac{\partial \mathcal{L}_\tau}{\partial y_\tau} \frac{\partial y_\tau}{\partial \boldsymbol{h_\tau}} \frac{\partial \boldsymbol{h_\tau}}{\partial \boldsymbol{h_j}} \frac{\partial \boldsymbol{h_j}}{\partial \theta} \tag{5}$$

with $\boldsymbol{h_j}$, $\boldsymbol{h_\tau}$ are given by (2) at $t = j$ and $t = \tau$. Each term $\frac{\partial \mathcal{L}_\tau}{\partial \theta}$ represents the partial-time gradient of the error in the past time steps (j's) up to the current time step (τ). Figure 3 shows a K-step RNN unrolled in time and detailed of a stack L layers of RNN cells.

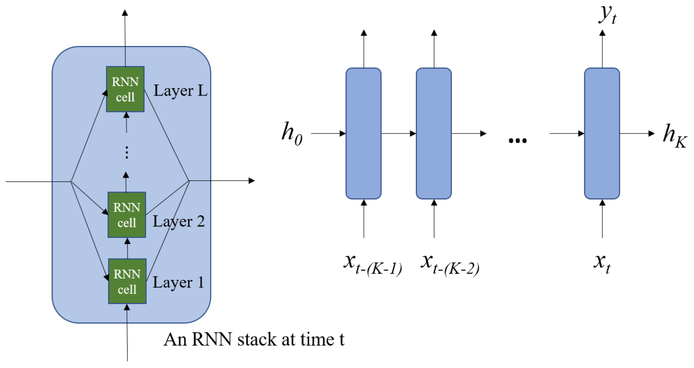

Fig. 3. K-step RNN unrolled in time

In order to represent the effect of the tap values to the RNN dynamic behavior, we use an FNN which takes the tap values as input then take its output to feed into the RNN as $\boldsymbol{h_0}$. An FNN is the composition of multiple weighted nonlinear functions and can be mathematically described as

$$\hat{y} = f_L \circ W_L f_{L-1} \cdots \circ W_2 f_1 \circ W_1 x \tag{6}$$

The input $x \in \mathbb{R}^{d_1}$, at the l^{th} stage, f_l is a non-linear activation function composed with f_{l-1} weighted by the weight matrix $W_l \in \mathbb{R}^{d_{l+1} \times d_l}$, where d_{l+1} and d_l are the dimension of the output of layer $l+1$ and that of layer l, respectively; $l = 1, 2, ..., L$, $\hat{y} \in \mathbb{R}^{d_{L+1}}$ is the prediction of the FNN. Since both input and output are real values, it is most convenient to choose *mean-square error* (MSE) loss function. MSE loss is calculated as the square of 2-norm of the error vector.

$$\mathcal{L}\left(\hat{y}, y\right) = \text{MSE}\left(\hat{y}, y\right) = \frac{1}{d_{L+1}} \left\|\hat{y} - y\right\|_2^2 \tag{7}$$

The weights of both networks are learnt at the same time, after backpropagation through time in RNN, the gradient is also backpropagated through the FNN so its weights can also be updated. Figure 4 visualizes how FNN and RNN are combined to make a tunable model in this work.

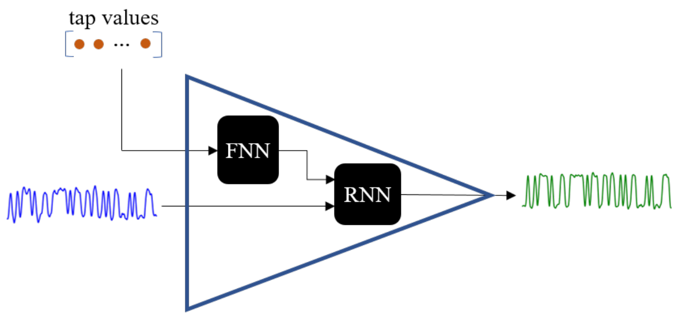

Fig. 4. Proposed architecture

IV. Example

In this section, we will demonstrate the proposed method using data from a 2-tap DFE for an about 1.7ns delay differential channel transmitting data at 32 Gbps. The tap values are normalized so that they span from 0 to 1. Negative values of the taps will amplify the post-cursor instead of cancelling it, hence, are excluded. The channel pulse response and the equalized response are shown in Figure 5. It can be seen that there are 1 pre-cursor and 2 significant post-cursors. The effect of DFE is reflected clearly on the equalized waveform which is cancelled out exactly at 2 most significant sampled post-cursor locations.

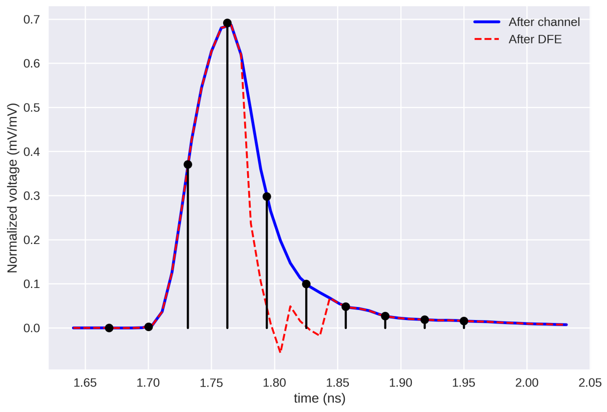

Fig. 5. Single pulse response and DFE effect

To prepare training data, many combinations of tap values are swept, the unequalized and equalized waveform is collected for each combination of tap values. In this example, we chose a 2-layer FNN which has 10 neurons and 20 neurons, respectively and a 6-layer, LSTM-cell RNN whose hidden state is in $\mathbb{R}^{30}$. Adam [4] is used for optimization with initial learning rate being 0.01. Also, RNN when trained was set to start out in *teacher-force* mode but we used a schedule sampling to choose between known data and RNN-generated data such that at the beginning of training, the RNN is fed with more known data (from training data) but as training progresses, the RNN is fed with more of its own generated data. More details about different modes of RNN-based model were presented in [2] and references therein.

After trained, the RNN is set to *read-out* mode and the model is tested with unseen data. Excellent agreement between the proposed model and the transient simulation is observed, the results are shown in Figure 6. Each color in Figure 6 is the output response of DFE model for different tap values.

V. Conclusion and Future work

In this work, we have combined an FNN and an RNN to create a tunable model that can completely replace the transistor model for different values of taps. The validation result on a pulse response matches very well with the transistor simulation result. This model can be exchanged between vendors and designers helping the former to protect their IP and ensure accuracy for latter at the same time. For EDA vendors, implementation of these kinds of models is straight forward and simple. Exchanging models between vendors and designers would be also simple because all that is needed is the network architectures and its weights. Deep learning community has proposed ONNX [5], an open source framework, built as an effort to offer a unified API to all neural network models so it can be exchanged and used no matter how and by which framework it was developed and trained.

In a future work, we will report how the proposed model would be trained and validated in a channel simulation to generate eye diagrams. A longer term extension of this work is to include the adaptivity of equalization schemes into the model.

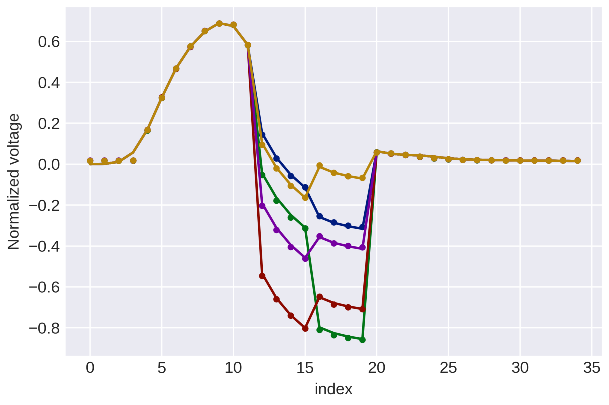

Fig. 6. Prediction (dot) vs. transient waveform (solid) for unseen tap combinations (different colors)

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. CNS 16-24810 for Center for Advanced Electronics through Machine Learning (CAEML), the U.S Army Small Business Innovation Research (SBIR) Program office and the U.S. Army Research Office under Contract No.W911NF-16-C-0125.

References

[1] T. Nguyen, T. Lu, J. Sun, Q. Le, K. We, and J. Schut-Aine, "Transient Simulation for High-Speed Channels with Recurrent Neural Network," in *2018 IEEE 27th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, Oct 2018, pp. 303–305.

[2] T. Nguyen, T. Lu, K. Wu, and J. Schutt-Aine, "Fast Transient simulation of High-Speed Channels using Recurrent Neural Network," 2019. [Online]. Available: https://arxiv.org/abs/1902.02627

[3] J. L. Elman, "Finding structure in time," *COGNITIVE SCIENCE*, vol. 14, no. 2, pp. 179–211, 1990.

[4] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[5] J. Bai, F. Lu, K. Zhang *et al.*, "ONNX: Open Neural Network Exchange," https://onnx.ai/, 2019.