Self-Evolution Cascade Deep Learning Model for High-Speed Receiver Adaptation

Bowen Li, Brandon Jiao, Chih-Hsun Chou, Romi Mayder, and Paul Franzon, *Fellow, IEEE*

***Abstract*— The IBIS algorithmic modeling interface (IBIS-AMI) has become the standard methodology to model Serializer/Deserializer (SerDes) behavior for end-to-end high-speed serial link simulations. Meanwhile, machine learning (ML) techniques can mimic a black-box system behavior. This article proposes the self-evolution cascade deep learning (SCDL) model to show a parallel approach to effectively modeling adaptive SerDes behavior. Specifically, the proposed self-guide learning methodology uses its own failure experiences to optimize its future solution search according to the prediction of the receiver equalization adaptation trend. The proposed SCDL model can provide the high-correlation adaptation results, while the adaptation simulation time is much faster than conventional IBIS-AMI models.**

***Index Terms*— Adaptation, behavior, cascade, deep learning, high correlation, IBIS algorithmic modeling interface (IBIS-AMI), modeling, receiver, self-evolution cascade deep learning (SCDL).**

I. Introduction

With the receiver adaptation algorithm, a robust serial link can be built regardless of the transceiver setting and channel characteristics. The adaptively adjusted gain control, CTLE peaking, and DFE coefficients automate the tasks that had previously been manual for the designer [1]. A Serializer/Deserializer (SerDes) is a pair of functional blocks used in high-speed communications to compensate for limited input/output. These blocks convert data between serial data and parallel interfaces in each direction. Due to the intellectual property (IP) of the SerDes vendors, customers do not have transparent knowledge about the circuit design. On the other hand, customers do need a fast and accurate simulator to evaluate their channels. As a result, a simulation model is usually provided from the vendors. While being effective on the real system, complex adaptation process makes it difficult to come up with a behavioral model for the simulation. Much effort has been made to develop the behavior model in the industry. The IBIS algorithmic modeling interface (IBIS-AMI) is currently the most commonly used approach to emulate receiver adaptation process.

While having high correlation with the real circuit behavior, the current challenge for the IBIS-AMI modeling is the simulation speed. Normally, engineers need to wait half an hour or even longer to see the results for the adaptation simulation, because the adaptation process requires transient state information. Moreover, designing an IBIS-AMI model needs detailed information about the circuit design data over process, voltage, and temperature (PVT) to improve the model accuracy. Combining with the fact that silicon vendors provide product and the associated IBIS-AMI model to the customer, the IBIS-AMI model development efficiency is important. Among all the behaviors, the receiver adaptation behavior is the most difficult to model.

The biggest challenge for building receiver adaptation behavioral models is that a new model is needed to be designed for each product. In the conventional IBIS-AMI model development, the knowledge to the underlying circuit is required to build an efficient model and the details of the adaptation algorithm. As each product is different, model details cannot be all included until the final design is completed, which can lead to long design cycles. As a result, a general modeling mechanism is desired to speedup the model development cycle. To decouple the need of circuit knowledge in the model design, all the data are the receiver input waveform and the receiver adaptation output codes. With this limited information, the problem fits itself as a black-box problem: a system viewed in terms of its inputs and outputs, without any knowledge of its internal workings. In recent years, machine learning (ML) models are proved to be very effective for the SerDes behavioral modeling [3]–[6]. However, the required training data grow as the complexity of the underlying black-box system. Also, fine tuning or even creating a specialized ML model is not necessarily easier than building the IBIS-AMI model. For the receiver adaptation behavior modeling Li *etal*., [14] proposed a method about CTLE adaptation using a deep learning model. To the best of our knowledge, there is no research focus on leveraging ML models on receiver CTLE and DFE adaptation behavior.

To tackle these challenges, an ML modeling framework is built to eliminate the time-consuming model tuning step in the ML model. A novel yet intuitive mechanism is developed to "self-evolve" through mixing low-level ML models and input

Manuscript received January 11, 2020; revised April 12, 2020; accepted April 30, 2020. Date of publication May 4, 2020; date of current version June 2, 2020. This work was supported in part by the National Science Foundation under Grant CNS 16-24811 and in part by the members of the Center for Advanced Electronics through Machine Learning (CAEML) IUCRC. Recommended for publication by Associate Editor W. T. Beyene upon evaluation of reviewers' comments. *(Corresponding author: Bowen Li.)*

Bowen Li and Paul Franzon are with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27606 USA (e-mail: bli11@ncsu.edu).

Brandon Jiao, Chih-Hsun Chou, and Romi Mayder are with Xilinx Inc., San Jose, CA 95124 USA.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCPMT.2020.2992186

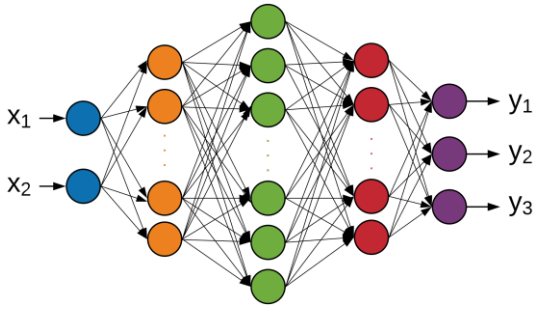

Fig. 1. DNN structure.

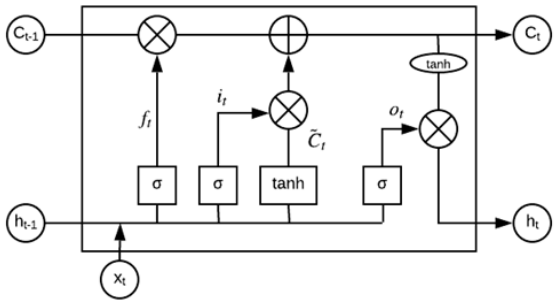

Fig. 2. LSTM cell structure.

data features. To the best of our knowledge, this is the first exploration of the feasibility and to have built the ML-based adaptation behavior model. In summary, the proposed general ML-based modeling framework has the following capabilities.

1) Find useful information from the model inputs and mutual information during the training.
2) Find data dependency during the training.
3) Leverage its own experience to correct its learning process.

As for the simulation speed, since the adaptation results are not based on the bit-by-bit transient simulation, the proposed model produces the result in the order of magnitude of seconds. Thus, the ML-based technique can speedup the adaptation process compared with the conventional IBIS-AMI modeling approach.

II. Deep Learning Model Structure Introduction

In this article, two deep learning models are used in the self-evolution cascade deep learning (SCDL) model library. In this section, a brief introduction to the two deep learning models is presented.

A. Deep Neural Networks

Neural network model is to mimic the activity of the human brain. It can learn a black-box system. Deep neural network (DNN) is a commonly used model in deep learning. Fig. 1 shows the DNN structure.

In each layer of the DNN, the number of neurons can be obtained as $\{L\} = (L^{\text{in}}, L^1, \ldots, L^h, L^{\text{out}})$, where L^{in}, L^h, and L^{out} are the number of neurons in the input layer, the hth hidden layer, and the output layer, respectively. The input of the hth hidden layer can be calculated by

$$\{z^h\} = \{x^{h-1}\}W^h \tag{1}$$

where W^h is a matrix which contains the weights between the output of the $(h-1)^{\text{th}}$ layer and the input of the hth layer. The output vector $\{x^h\}$ of the hth hidden layer is represented as follows:

$$\{x^h\} = f_a(\{z^h\} + \{b^h\}) \tag{2}$$

where f_a is the activation function and b^h is the matrix which contains the bias at the hth layer. In this article, rectified linear unit (ReLU) [11] is used as the activation function, which is obtained as follows:

$$f_a(z) = \max(0, z). \tag{3}$$

The output from the output layer is the prediction targets, $\{\hat{y}\}$. The stochastic gradient descent (SGD) method [12] is used to minimize the cost function.

In this article, the Bayesian optimization (BayesOpt) method is applied to select the best number of hidden layers and neurons of DNNs. The mean square error between outputs and expected outputs is used as the cost function.

B. Long Short-Term Memory

Recurrent neural network (RNN) is designed for time-series regression problem. In a vanilla RNN, the input and the hidden states are simply passed through a single tanh layer [7]. The RNN aims to map the input sequence x into output y. Each output in the sequence is calculated by the state of the previous RNN cell and current input.

Traditional RNN structure faces a challenge. To deal with the vanishing gradient or exploding gradient problems, the long short-term memory (LSTM) model is proposed [8]. The LSTM cell structure is shown in Fig. 2. The LSTM can create paths where the gradient can flow for a long duration.

The core function of the LSTM is the cell state, which is the horizontal line, from C_{t-1} to C_t, on the top of Fig. 2. The cell state will go through the entire chain, with some linear interactions. The LSTM has the capability to remove or add previous or new information to the cell state using gates. Gates are a way to optionally let information through. They are composed of a sigmoid neural network layer and a pointwise multiplication operation. An LSTM cell has three gates, namely input, output, and forget, to protect and control the cell state. The input gate will add new information selected from the current input and previous sharing parameter vector into the current cell. The forget gate is to discard useless information from the current memory cell. Also, the output gate decides new sharing parameter vector from the current memory cell.

At first, the LSTM would discard useless information from the cell state. This decision is made by a sigmoid layer named the forget gate layer. It looks at h_{t-1} and x_t, and outputs a number between zero and one for each number in the cell state C_{t-1}. If the output is one, it means that the LSTM would completely keep the cell state, while zero represents that the LSTM would completely forget this. The forget gate can be

calculated as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{4}$$

where f_t is the output of the forget gate; W_f and b_f are the weights and biases of the neural networks, respectively; h_{t-1} is the previous hidden vector; and x_t is the current input.

The next step is to decide what new information is needed to store in the cell state. This step is twofold. First, a sigmoid layer named the input gate layer decides which values will be updated. Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, which could be added to the state. Second, these two will be combined to create an update to the state. The input gate can be represented by

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \tag{5}$$

where i_t is the output of the input gate layer; W_i and b_i are the weights and biases of the input gate layer, respectively; and $\tilde{C}_t$ are the vectors which modify the cell state.

Then, the old cell state, C_{t-1}, will be updated, as shown in the following:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{6}$$

where C_t is the new cell state and f_t and i_t are the outputs of the forget gate and input gate, respectively. These two gate outputs will decide how much information will be through away and updated, which can be obtained as follows:

$$\begin{aligned} f_t &= \sigma(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i) \\ o_t &= \sigma(W_o \cdot [C_t, h_{t-1}, x_t] + b_o). \end{aligned} \tag{7}$$

Finally, the LSTM cell will decide what should be considered as the final output. A sigmoid layer will decide what parts of the cell state are the output. Then, the cell state, C_t, will go through a tanh layer and multiply it by the output of the sigmoid gate. The output of the LSTM can be obtained by

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned} \tag{8}$$

where o_t is the output of the output gate; h_t is the final output of the LSTM; and W_o and b_o are the weights and biases of the output gate layer, respectively.

With these three gates, the LSTM model will remember all the useful information from the receiver input and void gradient vanishing or exploding problems.

III. Proposed Modeling Architecture

In this article, a modeling mechanism, named self-evolution cascade deep learning (SCDL), is proposed to deal with general black-box problems in SerDes link. The proposed modeling architecture is shown in Fig. 3. There are three main blocks: modeling, scoreboard, and mutual information library (MIL). Next, these blocks and their interaction would be discussed in detail.

In the modeling block, the input to the framework is the receiver input waveform (raw data) and will be sent to the modeling block. The raw data are preprocessed for feature selection (data preprocessing block), where important features are extracted from the data pattern, such as the low-frequency and high-frequency signal amplitudes and slices based on unit interval (UI) information provided in the product specification. The extracted features are then sent to the model library for ML model training.

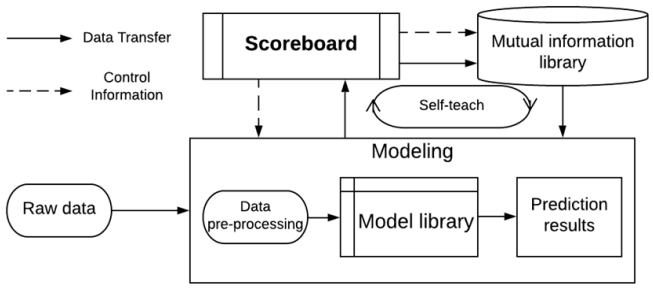

Fig. 3. SCDL architecture.

In the model library, there are multiple ML model types, such as DNNs and LSTM. These models will be chosen by the scoreboard to perform the ML training. Once the ML model is trained and validated, the testing accuracy will be passed to the scoreboard. In the proposed design, the modeling block will do the separated target modeling, which means that it will create one model for each target.

The scoreboard controls the prediction flow of the SCDL system. It selects useful information from the preprocessed data and mutual information during the training. Based on the data, proper ML model types are chosen to do the training. At the beginning of the flow, all ML models will be used independently. During the training phase, the BayesOpt method is used to optimize the ML model hyperparameters. After the training and prediction process is done, the scoreboard will gather the model architecture of all successfully predicted targets, whose prediction accuracies are higher than the predefined accuracy threshold. For the successful targets, scoreboard will try to optimize the ML training process by reducing the input feature sets and kick off the modeling flow until the minimum feature sets that satisfy the prediction accuracy threshold are found. The SCDL model will decide whether to keep input information according to the model performance. For instance, if the model accuracy drops more than 5% when one input data is removed, it proves that the removed input information is important, and the SCDL model will keep it. On the other hand, if the model accuracy remains the same or increases when one input data is removed, then that input data should be ignored during the model training. The final feature sets and model architecture will be stored and treated as "experience learned." Also, the targets will be stored in the MIL.

On the other hand, for the targets that failed the accuracy threshold, scoreboard will start the "evolution" flow. Scoreboard will use the stored experiences (feature sets and ML model) plus the targets stored in the MIL to train and predict the failing targets. The purpose is to explore possible dependency between targets.

Similarly, if the newly trained model satisfies the threshold, it will then enter the optimization phase described previously. However, if the threshold is still not met, all the features and mutual information that exist for training and prediction with each ML model independently would be used. The reason behind this is that the target could have very different behaviors than what successfully learned. Mutual information would provide useful connections among those. If this still failed, the same feature sets will be used with all ML models working in combination. This provides the strongest modeling capability it can have.

The SCDL model will stop its training in two scenarios. The first is that it successfully finishes the training and provides high-precision predictions regarding the preset accuracy requirements set by a human. The second scenario is when it tries all the models and training iterations and finds the performance cannot meet the preset accuracy threshold under the current circumstances, it will stop and tell some information of the bad prediction cases. Those bad prediction cases may be some corner cases that should not be considered. This information can help users to improve the data quality.

After the training and reducing the input data for each model, the SCDL model could figure out what type of model and data are used to predict each target. This backtracking process can let human learn the model prediction flow and the laws that the SCDL model finds itself, which can provide guidance in future modeling mechanism design.

In this article, the SCDL model shows a parallel approach to effectively modeling adaptive SerDes behavior. Specifically, the proposed self-guided learning methodology uses its own experiences to optimize its future solution search according to the prediction of the receiver equalization adaptation codes. The proposed model should provide fast and high-precision predictions under various PCB channels with different transmitter settings. Due to its self-training process, the design process of the SCDL model is usually much faster than the IBIS-AMI model.

The use of ML algorithms frequently involves careful tuning of learning parameters and model hyperparameters. Unfortunately, this ML model tuning requires lots of expert experience or even brute force search. Therefore, a great appeal for automatic approaches, called the BayesOpt, is proposed [13], which can optimize the performance of the ML model [9], [10]. In this article, the BayesOpt is used to optimize the model configuration.

Generally, the goal of the BayesOpt is to find a global maximum/minimum of an unknown objective function f:

$$\theta^* = \underset{\theta \in X}{\operatorname{argmax}}\ f(\theta) \tag{9}$$

where $X \subseteq \mathbb{R}^{D_X}$ is the design space or data space. D_X is the dimensionality of the parameter space. Furthermore, it is assumed that the unknown objective function f can be evaluated at any arbitrary query point θ in the data space. This evaluation produces outputs $y \in \mathbb{R}$ such that $\mathbb{E}[y|f(\theta)] = f(\theta)$. Hence, the function f can be observed through unbiased noisy pointwise observations y. In this setting, a sequential search algorithm is considered, which, at iteration n, selects a location θ_{n+1} at which to query function f and observe y_{n+1}. After N queries, the algorithm makes the final recommendation θ_N, which is the best estimate.

Fig. 4. Various data patterns.

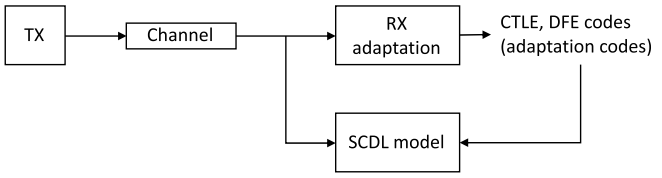

Fig. 5. Data collection process.

The BayesOpt proposes a prior belief over the possible objective functions and improves this model according to data observed via Bayesian posterior updating. Equipped with this probabilistic model, the BayesOpt can sequentially induce acquisition functions Υ_n: X⟼R that use the uncertainty in the posterior to guide the exploration. Intuitively, the acquisition function evaluates the candidate points for the next evaluation of f; therefore, θ_$(n+1)$ is selected by maximizing Υ_n, where the index n indicates the implicit dependence on the currently available data.

IV. Modeling Process

For the receiver adaptation system, the inputs are the receiver input waveform, while the outputs are the receiver adaptation codes. Multiple input data patterns are prepared, namely pseudorandom binary sequence (PRBS), a single-bit response (SBR), and long pulse response (LPR), as shown in Fig. 4. PRBS data can provide intersymbol interference (ISI) and channel loss information. In this article, multiple PRBS patterns are used, for example, PRBS7 and PRBS31. SBR data can show channel loss information. The LPR would provide dc gain information. The aim of providing all kinds of data pattern is to prepare all the information for the SCDL model to predict the receiver adaptation system. For each data case measurement, the receiver input waveform and the adaptation codes are collected as the SCDL model inputs and outputs, as shown in Fig. 5. The receiver used in this article has 18 adaptation codes. However, since all the data are measured in a set of conditions, for example, PVT and the TX and RX reflection coefficients fall in a particular range after the adaptation process converges. In this article, only seven adaptation codes are considered.

In this article, 700 cases are measured from the transceiver. Those cases are collected over 15 industry channels. By tuning the TX deemphasis settings, several cases over one channel are generated. Among all the TX settings, low, medium, and high swing cases are selected because they can

TABLE I
TRAINING AND TESTING DATA CONFIGURATION

Parameters	Configurations
Data rate	25.78 Gpbs
Channel insertion loss	Low, medium, high
TX de-emphasis	Low, medium, high swing

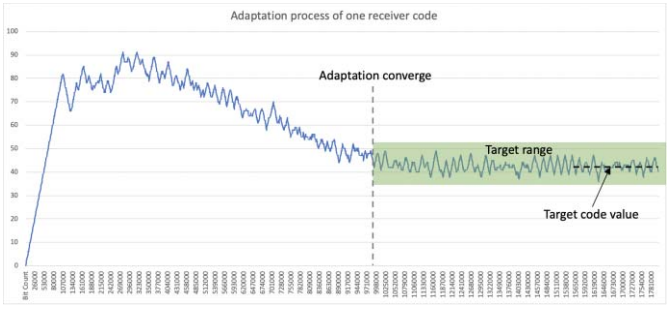

Fig. 6. Adaptation process for the receiver code.

cover underequalized, properly equalized, and overequalized cases. The data are measured under different channel cases and various TX deemphasis settings. Those case scenarios are summarized in Table I.

The data are divided into three groups. Five hundred data cases from 11 channels are set as the training dataset. Fifty data cases from two channels are set as the validation data. The training and validation channels are from customers, which cover low to high channel loss. The rest 176 data from two channels are set as testing data. Those two testing channel cases are from IEEE standards and hidden from the model training and validation process. The targets are collected from the receiver adaptation codes after the adaptation process is done.

As for the data generation, receiver codes are collected after the adaptation process is done. To get a single value as the model prediction target, the receiver code values are collected at the last simulation bit, as shown in Fig. 6. However, even after the adaptation process, the receiver equalization codes are still dithering.

Since the model target should be a single value, not a range, during the model training, the mean values after the adaptation are used as the model target, which is the target code value in Fig. 6. During the model prediction, a target range, which can be calculated as [target − threshold, target + threshold], is set for each receiver code. The threshold is set according to the normal fluctuation range of each code after the adaptation. Different code has a different threshold. If the predicted value lies in the target range, it is a correct prediction. The prediction accuracy can be calculated as follows:

$$\text{Prediction accuracy} = \frac{n}{m} \times 100\% \tag{10}$$

where n is the number of cases that the predicted value is in the target range and m is the number of all the testing

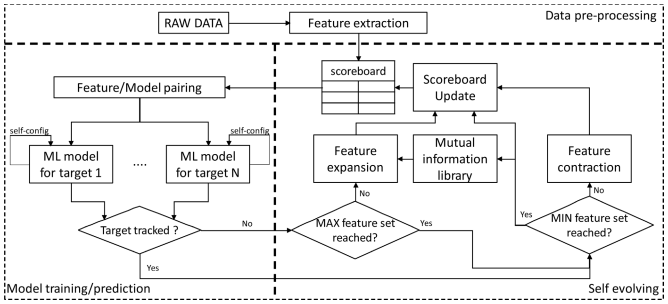

Fig. 7. SCDL flow chart.

cases. At the model testing process, the prediction accuracy is calculated, which can describe the model performance.

Fig. 7 gives the detailed flow chart of the SCDL framework. In this section, the workflow of the SCDL framework would be presented. At first, all the raw inputs are sent to the data preprocessing block for feature extraction, where the important features are measured from the data pattern, such as the low-frequency and high-frequency signal amplitudes and slices of receiver waveform. The extracted feature will be sent to the scoreboard as the base data set. As a design choice out SCDL will start from using the DNN model type. This choice provides shorter learning time overall during the experiment. The scoreboard will send the data set that fit DNN to different models to predict each receiver adaptation code target. After the training process, the scoreboard will examine each prediction result and score the results that exceed the accuracy threshold. The prediction results from the initial feature data set with high scores are saved and do the modeling again. In this article, seven adaptation codes, which labeled targets 1–7, are considered. While each target code going through independent training, the experiences and prediction results are stored and shared.

The goal of SCDL is to select only useful information from the initial feature data set. To do that, it will go through two different phases: feature expansion and feature contraction. When the prediction accuracy is below threshold, SCDL will try to include more features if available in the feature expansion phase. On the other hand, there are times that redundant features would hurt the accuracy, thus the feature contraction phase will try to cut down the input feature.

In the experiments, the SCDL model figures out that using the PRBS data will not only significantly increase the model training time and model complexity but also produce low-prediction accuracies for all the codes no matter which PRBS data pattern is used, in the range of 20%–30%. On the other hand, using the SBR and LPR features, the scoreboard can find reasonable model structure and the prediction accuracies can be improved, which are from 40% to 60%. In that case, the SCDL model saves the PRBS data for further study and only uses SBR and LPR data as model inputs. The reason why the prediction accuracies are low using the PRBS data is because the PRBS data contain lots of redundant information, such as data dependency, which would mislead the ML model. Also, the ML model needs a long pattern for PRBS data to cover all the data dependencies, which leads to

a tremendously long training time. On the other hand, simple data patterns, such as SBR and LPR, can provide the model concise and useful information, for example, high-frequency or low-frequency loss.

Next, the SCDL model uses the DNN model to predict all the receiver adaptation codes with the measured features from the SBR and LPR data, including high-frequency signal amplitude, low-frequency signal amplitude, high to low transition time, and amplitude at each UI information after the SBR main cursor. During the DNN model training, the BayesOpt is applied. After the training, the DNN model can provide high-precision predictions for targets 1–5, while other targets show low correlations with the real code values.

After target accuracy is met, the SCDL will go into the feature contraction phase and try to reduce the ML model complexity by reducing the model inputs one by one. If the model accuracy drops more than 5% when one input feature is removed, it proves that the removed input information is important, and the SCDL model will keep it. On the contrary, if the model accuracy remains the same or increases when one input data is removed, that input data should be ignored in the future model training. After the input reduction process, the scoreboard finds that two measured features from the SBR and LPR data will influence targets 1–3 prediction performance the most. With these two features, the ML model complexity significantly drops, the training time is shortened, and the model performance is much improved. Hence, these two features are selected as the DNN model inputs to predict targets 1–3. At this point, all the predicted targets 1–3 will be saved in the MIL for the future prediction. The successful experience is saved for future modeling.

During the optimization process for targets 1–3, the SCDL also record that targets 4 and 5 show better prediction accuracies. To further improve the prediction accuracy, SCDL will add mutual information from the MIL, which has successfully predicted target values, as a new feature set to predict targets 4 and 5. After training, the prediction accuracies for targets 4 and 5 are much improved. Again, the scoreboard will do feature contraction to reduce the modeling flow complexity. After that, the scoreboard updates the successful experience.

For targets 6 and 7, the previous features cannot provide high-precision accuracies, and the scoreboard switches to use the LSTM model. Since the input LSTM model should be time-series data, the SBR and LPR data are sent to the LSTM model separately to predict targets 6 and 7. After training, the LSTM model shows better performance using the SBR data than the LPR data. Out of all targets, targets 6 and 7 show better performance than others, with the accuracies above 70%.

The scoreboard focuses on performance improvement for these two targets until the prediction accuracies meet the preset accuracy requirements. At first, targets 6 and 7 are predicted using two individual LSTM models. The sliced SBR data are the LSTM model inputs. For example, the first, second, and third UI data after the SBR main cursor are sent to one LSTM model sequentially to predict one target. After modeling, the scoreboard selects sliced SBR data, which shows the best prediction accuracy, to predict the current target. To further improve the prediction accuracy, the scoreboard uses mutual information from the MIL, which are predicted targets 1–5. However, these features cannot be applied to the LSTM model. As a result, a DNN model is added in cascade with the LSTM model to boost the prediction accuracy of each target. One LSTM model is used to predict the initial target 6 using the sliced SBR data. After that, the initially predicted target 6 will be sent to an individual DNN model, along with previously predicted targets 1–5, to predict the final target 6. In this way, the prediction accuracy of target 6 is much improved, which are around 88%. Next, the SCDL model would go to the feature contraction process, which reduces the inputs of the DNN model. After the input reduction, the prediction accuracy of target 6 is above 90%. Then target 6 is saved into the MIL. The scoreboard saves this successful experience for future modeling.

However, the prediction accuracy of target 7 is still below preset threshold. At last, the SCDL model chooses to use mutual information and a DNN model to predict target 7. After DNN input reduction, the prediction accuracy of target 7 meets the preset requirement.

After self-evolution learning process, all the predicted targets show high correlations with the real code values. In that case, the SCDL modeling process is finished and provide detailed information about its prediction process using the backtracking method. Meanwhile, a counter is set to count how many tries a target is been through. After going through too many tries (exceeding a predetermined threshold), it will stop and ask for more data or more ML models. All the evolution steps are automatic. The SCDL model would choose which evolution step it should go next based on the current performance and mutual information. However, the current SCDL model prediction capability has limitations. Once the best accuracies are reached, human need to design more decision steps to boost the SCDL model performance.

V. Self-Evolution Ability

The self-evolution ability can be proved by three scenarios.

1) It can use mutual information during the modeling process and provide better predictions.
2) It can optimize the prediction flow when the accuracy requirement is strict.
3) It can find data dependency during the model optimization process.

In this section, these self-evolution abilities would be demonstrated.

A. Use Mutual Information

When lower accuracy requirements are set for each target, the first prediction flow chart is shown in Fig. 8. For the first three targets, the scoreboard chooses to use two measured features from the SBR and LPR data and three DNN models separately for each target. The successfully predicted targets are sent to the MIL for future prediction.

As for the rest of the targets, the scoreboard uses two DNN models to predict targets 4 and 5. After the input reduction, the inputs of each DNN model are two measured features from the SBR and LPR data and mutual information from MIL.

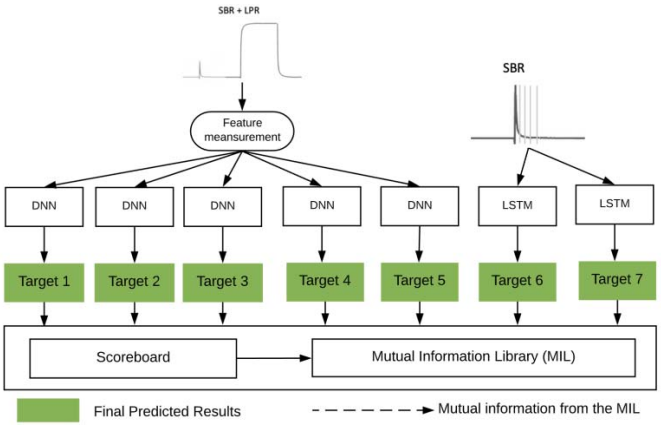

Fig. 8. First prediction flow that the SCDL model proposed when the preset accuracy requirements are low.

Similarly, the successfully predicted targets are sent to the MIL for future prediction.

For targets 6 and 7, the SCDL tries the DNN model at the first time, but the model prediction accuracies are below preset accuracies. The scoreboard then switches the model type, which is the LSTM model. The inputs of the LSTM model would be the sliced SBR data. After training, the prediction accuracies meet the requirements. Hence, the SCDL model successfully finishes the training and provides reasonable predictions regarding the accuracy requirements set by users.

During the modeling process, the SCDL model can use mutual information to learn failed-predicted targets, which is the first self-teaching/self-evolution scenario.

B. Optimize the Prediction Flow

As mentioned previously, the SCDL model will stop its training if it successfully meets the preset accuracy requirements. If the users increase the accuracy requirements, would the SCDL model optimize the prediction flow to provide better predictions? In the next experiment, higher accuracy requirement for each target is set in the SCDL model. After its self-training, the optimized prediction flow is shown in Fig. 9. The prediction flows of targets 1–3 are the same as the first prediction flow. As for targets 4 and 5, the SCDL model chooses to use two DNN models. The inputs of these two DNN models are the measured features along with previous predicted targets 1–3 in MIL.

For target 6, the SCDL finds a new way to construct models. The sliced SBR data are fed into one individual LSTM model and cascade another DNN model to predict target 6. The inputs of the DNN model are the LSTM model prediction results and mutual information from the MIL. As for target 7 prediction, the scoreboard uses only mutual information with a single DNN model. The DNN input reduction can help model to improve the performance and find data dependency among the targets. After the self-evolution process of the SCDL model, the prediction accuracies increase more than 20% on average across targets.

TABLE II
DATA DEPENDENCY FOUND BY THE SCDL MODEL

Target	Model type	Data dependency
Target 1	DNN	High-frequency and low-frequency signal amplitude
Target 2	DNN	High-frequency and low-frequency signal amplitude
Target 3	DNN	High-frequency and low-frequency signal amplitude
Target 4	LSTM + DNN	High-frequency and low-frequency signal amplitude, sum of the SBR amplitude data, target 1-3
Target 5	LSTM + DNN	High-frequency and low-frequency signal amplitude, sum of the SBR amplitude data, target 1-3
Target 6	LSTM + DNN	Sliced SBR data, target 3-5
Target 7	DNN	Target 3-6

C. Find Data Dependency

After the SCDL model finishes its modeling process, the SCDL model will enter the feature contraction phase and try to reduce the input information for each model to speedup the overall training and data generation process. During the feature contraction process, the scoreboard selects important mutual information from the MIL, which are critical to the model performance. If one previously predicted target (mutual information) is removed from the model input and the prediction accuracy significantly drops, it means that feature is important to the current model prediction. Hence, the scoreboard will keep that mutual information and try to remove another feature from MIL. After the feature contraction process, the scoreboard can find data dependency among inputs and all the targets. Table II shows the final prediction flow and the data dependencies. Target means the model predicted outputs. Model type means which model type is used to predict the current target. Data dependency means which data are selected to predict the current target. Those data dependencies can provide users some useful information found by the SCDL model. People can use the data dependency to design modeling flow in the future.

D. Example of Target 6 *Prediction Evolution Process*

The prediction accuracy improvement process of target 6 is shown in Fig. 10. The preset accuracy threshold is 95%. This threshold is set according to SI engineer requirement.

At the beginning, the SCDL model uses the raw data to predict all the targets. The first trial is to use a DNN model to predict target 6. The inputs of the DNN model are high-frequency signal amplitude, low-frequency signal amplitude, and the sum of the sliced SBR amplitude after the main cursor based on experience learned from previous targets. After training, the prediction accuracy is low as shown in Phase I of Fig. 10. Next, SCDL will enter the feature expansion stage. Since the SCDL model does the parallel modeling, targets 1–5 prediction accuracies meet the preset accuracy thresholds and saved into the MIL. The mutual information from the MIL and previous learned features will then be used

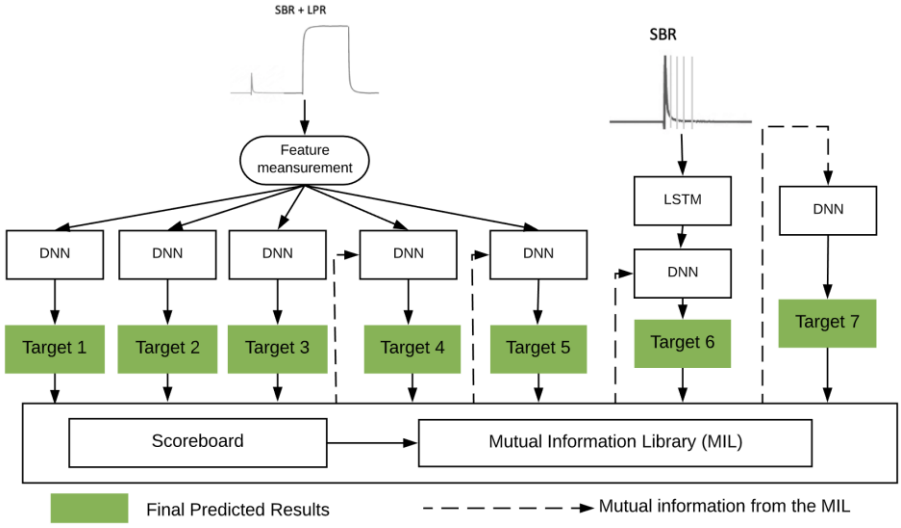

Fig. 9. Prediction flow optimization by the SCDL model when the preset accuracy requirements are high.

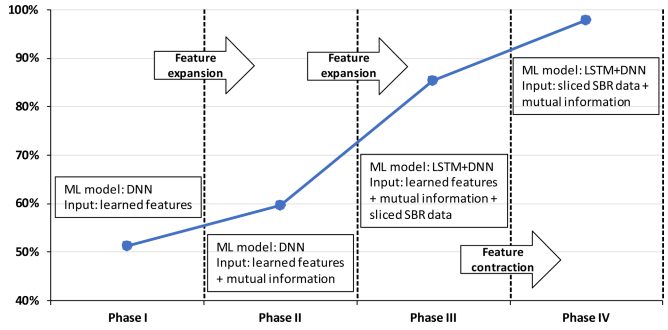

Fig. 10. Target 6 prediction accuracy improvement process.

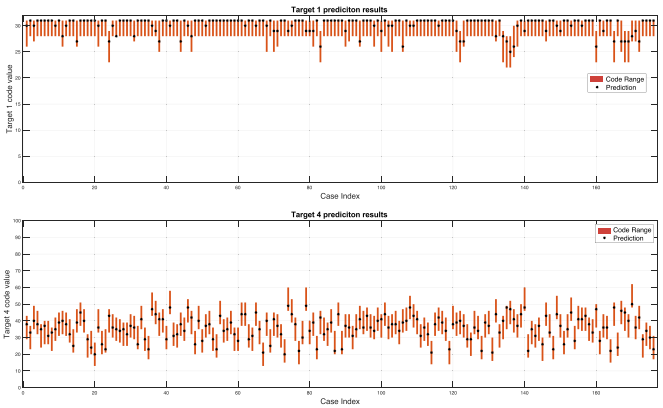

Fig. 11. Prediction results of targets 1 and 4 using the SCDL model.

with the DNN model to predict target 6 again. After training, the prediction accuracy is still below the preset accuracy threshold, as shown in Phase II of Fig. 10. Similarly, SCDL will do feature expansion again and include sliced SBR data. As a result, the LSTM model will be used in conjunction with the DNN model to predict target 6. The prediction accuracy is much improved but still not good enough, which is illustrated in Phase III of Fig. 10. Finally, after trying with all possible inputs and still miss the accuracy threshold, the SCDL will enter the feature contraction phase and try to eliminate feature that hurts the performance. The input features are removed one by one until the accuracy meets the preset requirements, illustrated in Phase IV of Fig. 10. The SCDL model will save the successfully predicted target 6 into the MIL.

VI. Self-Evolution Prediction Results and Explanations

A. Types of Graphics

The data are collected from the Xilinx UltraScale+ GTY transceiver [1], [2]. In this section, the self-evolution prediction results of the SCDL model are presented. One hundred seventy-six testing data are collected under two different channels and various TX deemphasis settings. After the SCDL model self-evolution, some of the prediction results for the target adaptation codes are illustrated in Fig. 11. Other prediction results are shown in the Appendix. The X-axis is the case index number, while the Y-axis is the code value. Because each code is varying during the adaptation process, a red box is used for each case, which can represent a target range for each case. The black star sign is the model prediction value. Most of the predicted code values lie in the target ranges. The prediction results of the SCDL framework fall nicely into the target range, showing high correlations with the real codes.

Table III shows the accuracy data for all seven targets. In this table, two numbers are given: first attempt accuracy and evolved accuracy. The first attempt accuracy is the accuracy at the first run of using SCDL, where no knowledge is learned. On the other hand, the evolved accuracy is the final output from SCDL. In the first attempt with all features extracted from raw data, only the first three targets can be predicted. After self-evolution of the SCDL model, the prediction accuracies are much improved by the self-teaching process. The accuracies of all codes exceeded the preset accuracy threshold (90%) and achieved 98.9% accuracy on average across all targets.

TABLE III
MODEL PERFORMANCE IMPROVEMENT BY THE SELF-EVOLUTION OF THE SCDL MODEL

Target	First attempt accuracy	Evolved accuracy	Targets	First attempt accuracy	Evolved accuracy
Target 1	100%	100%	Target 5	71.0%	98.9%
Target 2	98.86%	98.86%	Target 6	51.1%	97.7%
Target 3	98.9%	98.9%	Target 7	52.4%	97.7%
Target 4	86.4%	100%			

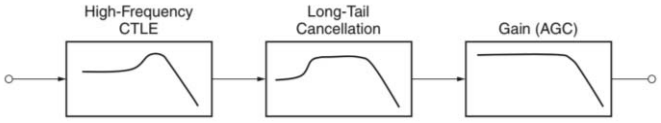

Fig. 12. Three-stage CTLE in the UltraScale+ Transceiver.

B. Prediction Explanation Using the Circuit Structure

From the previous experiments, the SCDL framework can effectively predict the receiver adaptation codes. However, this is not the full potential of SCDL. The capability of SCDL would be presented to learn the circuit structure through self-evolution. In this article, Xilinx UltraScale+ GTY transceiver [1], [2] is used as the experiment platform.

In the transceiver, an equalization technique is used to attenuate the low-frequency component of the signal while boosting the high-frequency component. CTLE at the receiver end is one of the most popular linear equalization techniques, illustrated in Fig. 12 [1]. CTLE is used to attenuate the low-frequency component of the signal while boosting the high-frequency component. The CTLE has three stages, namely high-frequency gain (KH), low-frequency gain (KL), and automatic gain control (AGC). The KL and KH are to boost low-frequency and high-frequency contents, respectively, after the channel. The AGC is to boost both low-frequency and high-frequency contents.

DFE is a proven technique to mitigate ISI without amplifying noise. It works by directly removing the ISI from previous bits, allowing the current bit to be correctly sampled. The DFE starts with a "decision slicer" to determine whether the current symbol is high or low. The resulting symbol goes through unit delays and multiplies with the tap weights. The weighted delayed signals are added together and then subtracted from the input analog signals, as shown in Fig. 13 [1]. If the tap weights are well selected to cancel the ISI at each following symbol, the result of this feedback loop is able to compensate for as many taps of ISI as the DFE has. In the Xilinx UltraScale+ transceiver, the DFE has 15 taps. The CTLE codes and DFE taps are adaptive in the transceiver. Since the channels used in this research have small reflections, the values for DFE taps 5–10 lie in a particular range. Hence, only three CTLE adaptation codes and the first four DFE taps are considered as the model prediction targets.

With the background of the transceiver, the prediction target to the adaptation code can be related. Table IV shows the

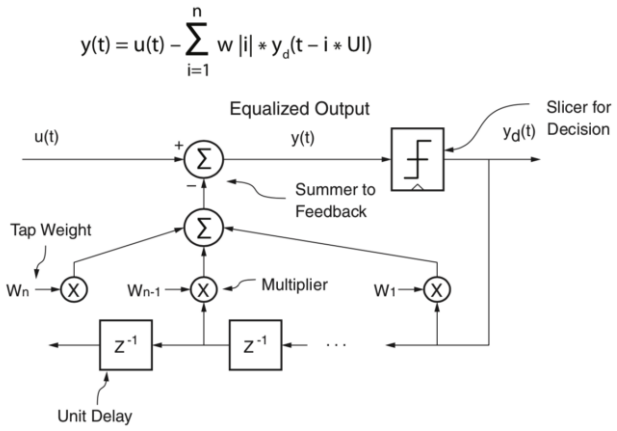

Fig. 13. DFE block in the UltraScale+ Transceiver.

TABLE IV
TARGET LABEL MAPPING FOR THE RECEIVER ADAPTATION CODE

Target label	Receiver code	Target label	Receiver code
Target 1	CTLE AGC	Target 5	DFE Tap2
Target 2	CTLE KL	Target 6	DFE Tap3
Target 3	CTLE KH	Target 7	DFE Tap4
Target 4	DFE Tap1		

corresponding receiver code of each target label. In this article, three CTLE stages and the first four DFE taps are considered.

According to the prediction flow in Fig. 9, three CTLE adaptation codes have high correlations with the high-frequency and low-frequency signal amplitudes measured from the SBR and LPR data. This can be explained by the CTLE function. As mentioned previously, the CTLE equalization technique is used to attenuate the low-frequency component of the signal while boosting the high-frequency component. The SBR and LPR can provide high-frequency loss and low-frequency gain, respectively. With that information, the DNN model would predict the CTLE adaptation codes accurately.

From the DFE prediction flow in Fig. 9, the first three DFE taps have high correlations with the sliced SBR data. The DFE would remove the ISI from previous bits by subtracting the sum of the weighted delayed signals from the input analog signals. The first few DFE tap values are related to the information after the main cursor of the SBR. In the circuit, the DFE tap values are influenced by the previous DFE taps because of the correlations among the DFE shift registers. This can explain why the SCDL model chooses to use DFE taps 1–3 to predict the DFE tap 4. Moreover, from the self-evolution process in Figs. 8 and 9, if the accuracy requirements become strict, the SCDL model will also add CTLE adaptation codes to predict the DFE tap 1. This phenomenon is reasonable because there are interactions among the CTLE and DFE adaptation process. To get better prediction results, the SCDL model found the interactions among the CTLE adaptation codes and the DFE tap values.

From this article, the SCDL model generates a cascaded deep learning model structure to predict the complex receiver

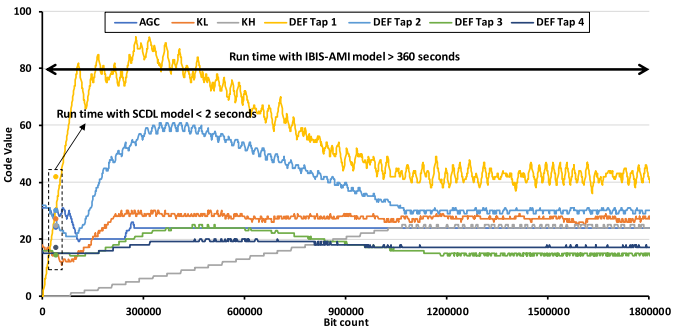

Fig. 14. Adaptation process using the SCDL model versus the IBIS-AMI model.

adaptation progress. It can be proved that the proposed SCDL model has the capability to find natural laws during its self-teaching and self-evolution process. This can provide a human useful guidance when exploring a new black-box problem.

C. Adaptation Speed Improvement: SCDL Versus IBIS-AMI

Besides the high accuracy and the ability to catch the underlying circuit architecture, SCDL can also generate the RX equalization adaptation codes in terms of seconds. Fig. 14 shows one example of an adaptation process using the SCDL model versus the IBIS-AMI model simulation. For the IBIS-AMI model, the simulation needs to run for millions of bits until all the codes converge, and this process takes significant amount of simulated bits. With the use of the SCDL model, the bit-by-bit simulation can be bypassed and give the accurate simulation output. With the SCDL model, the speed of the adaptation process significantly improves from more than 360 s (using the IBIS-AMI model) to less than 2 s (using the SCDL model), which is 180 times improvement in the speed.

The training process for the SCDL model is also fast. In this article, the evolution training process would take about 24–36 h, which is much faster than the design process of the IBIS-AMI model.

However, the limitation of the current SCDL model is that it can only predict receiver adaptation codes, while the IBIS-AMI model can provide a full transient simulation, including transient waveform, eye diagram, and jitter. In the future, more features would be added into the SCDL model. Moreover, the current SCDL model prediction capability has limitations. Once the max accuracy for each target is reached, human need to design more decision steps or add more model types in the SCDL modeling mechanism.

VII. Conclusion

High-speed SerDes equalization parameter autotuning, also known as adaptation, is a complex process. A new modeling mechanism, called SCDL model, is proposed and used in the prediction of high-speed SerDes receiver equalization adaptations. The process from Figs. 8 to 9 presents the evolution steps for the SCDL framework. The SCDL framework explores and evolves through each evolution run. During the

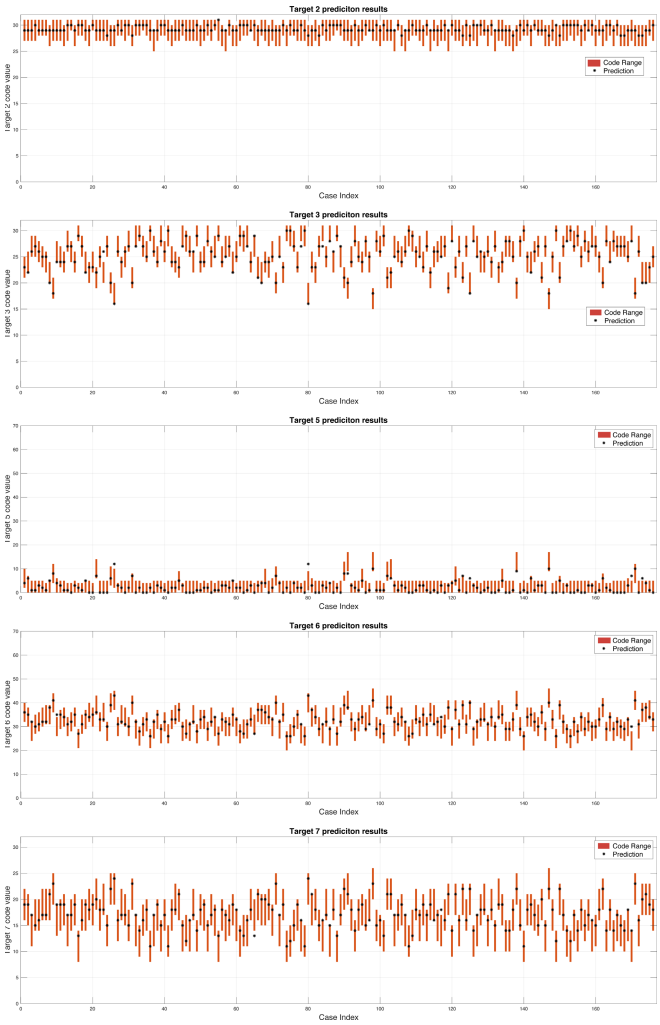

Fig. 15. Prediction results for the rest targets using the SCDL model.

training process, the SCDL model uses its own successful experiences to self-teach its future solution search and provides information such as the dependency/independency among various adaptation behaviors. Consequently, the SCDL model shows a low demand for training data to promote its prediction accuracy. After the training process, the SCDL model successfully develops different solutions with various preset accuracy tolerance with the same training data set. The SCDL model can boost its performance by changing the prediction flow during the self-evolution progress. To test the proposed model robustness, two different designs are illustrated. The SCDL model shows the high-precision prediction results for both designs. For the adaptation speed, the SCDL model can provide 180 times faster simulation than the state-of-the-art IBIS-AMI modeling approach.

In summary, the proposed modeling method achieves the following capabilities.

1) Can leverage its own experience to correct its learning process.
2) Can find useful information from the model inputs and mutual information during the training.
3) Can explore the underlying circuit architecture.

4) Can generate output code two orders of magnitude faster than conventional method.

Currently, only 15 channels are used in this article, which could not cover all the conditions. The SCDL model can be proved to have the ability to predict three CTLE codes and the first four DFE taps in a high-speed receiver. The future work will focus on the following.

1) Add more channel cases, for instance, high reflection, and consider all the receiver codes, including all the DFE taps.
2) Improve the model performance by performing sensitivity analysis and testing on other SerDes technologies.

APPENDIX

See Fig. 15.

ACKNOWLEDGMENT

This work was cooperated with Xilinx Inc.

REFERENCES

[1] B. Jiao, "Leveraging UltraScale architecture transceivers for high-speed serial I/O connectivity," UltraScale GTH/GTY Transceivers, Xilinx Inc., San Jose, CA, USA, White Paper, Oct. 2015.
[2] *UltraScale Architecture GTY Transceivers*, Xilinx Inc., San Jose, CA, USA, Sep. 2017.
[3] T. Lu and K. Wu, "Machine learning methods in high-speed channel modeling," DesignCon, Santa Clara, CA, USA, Tech. Rep., 2019.
[4] R. Trinchero and F. G. Canavero, "Modeling of eye diagram height in high-speed links via support vector machine," in *Proc. IEEE 22nd Workshop Signal Power Integrity (SPI)*, May 2018, pp. 1–4.
[5] B. Li, P. Franzen, Y. Choi, and C. Cheng, "Receiver behavior modeling based on system identification," in *Proc. IEEE 27th Conf. Elect. Perform. Electron. Packag. Syst. (EPEPS)*, Oct. 2018, pp. 299–301.
[6] B. Li, B. Jiao, M. Huang, R. Mayder, and P. Franzon, "Improved system identification modeling for high-speed receiver," in *Proc. IEEE 28th Conf. Elect. Perform. Electron. Packag. Syst. (EPEPS)*, Oct. 2019, pp. 1–3.
[7] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecasting—A novel pooling deep RNN," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 5271–5280, Sep. 2018.
[8] F. Gers, "Long short-term memory in recurrent neural networks," Ph.D. dissertation, Verlag nicht ermittelbar, 2001.
[9] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, Jan. 2016.
[10] J. Močkus, "On Bayesian methods for seeking the extremum," in *Proc. Optim. Techn. IFIP Tech. Conf.* Berlin, Germany: Springer, 1975, pp. 400–404.
[11] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, 2010, pp. 807–814.
[12] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*. [Online]. Available: http://arxiv.org/abs/1609.04747
[13] J. Močkus, *Bayesian Approach to Global Optimization: Theory and Applications*. Norwell, MA, USA: Kluwer, 1989.
[14] B. Li, B. Jiao, C. Chou, R. Mayder, and P. Franzon, "CTLE adaptation using deep learning in high-speed SerDes link," in *Proc. Electron. Compon. Technol. Conf. (ECTC)*, May 2020.

Bowen Li received the bachelor's degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2014, and the Ph.D. degree in computer engineering from North Carolina State University, Raleigh, NC, USA, in 2020. His research topic is "High-speed Receiver Behavioral Modeling using Machine Learning."

During his Ph.D., he interned in Hewlett Packard Enterprise, San Jose, CA, USA, and Samsung SARC, Austin, TX, USA. He has over 6 years of experience with machine learning.

Brandon Jiao received the Ph.D. degree in electromagnetic field and microwave technology from the Beijing University of Posts and Telecommunications, Beijing, China, in 2005.

He joined Xilinx Inc., San Jose, CA, USA, in 2014, where he is currently a Senior Staff Transceiver Engineer. Prior to joining Xilinx Inc., he worked for Intel and Nortel. His current research interest is SI/PI methodology in transceiver and RF applications.

Chih-Hsun Chou received the M.S. degree in computer engineering from the University of Louisiana at Lafayette, Lafayette, LA, USA, in 2011, and the Ph.D. degree in electrical engineering from the Department of Electrical and Computer Engineering, University of California at Riverside, Riverside, CA, USA, in 2017.

He is currently with Xilinx Inc., San Jose, CA, USA. His current research interest includes system level design and architecture with a specific focus on PCIe acceleration platform for machine learning and SmartNIC.

Romi Mayder received the Bachelor of Science degree in electrical engineering and computer science from the University of California at Berkeley, Berkeley, CA, USA, in 1992.

He is currently the Senior Director of the Technical Marketing Department, Xilinx Inc., San Jose, CA, USA. Prior to joining Xilinx Inc., he worked as a consultant specializing in silicon die level signal and power integrity. He also consulted in the field of design and fabrication of advanced package technologies, including stacked silicon interconnect. He has been employed by two companies in the Test and Measurement industry, Agilent Technologies, Santa Clara, CA, USA, and Anritsu (Wiltron) Company, Atsugi, Japan, where he specialized in microwave and millimeter-wave microelectronic circuit design and fabrication. He has published 25 patent applications in the fields of signal and power integrity as well as semiconductor process technologies.

Paul Franzon (Fellow, IEEE) received the Ph.D. degree from the University of Adelaide, Adelaide, Australia, in 1988.

He has worked at AT&T Bell Laboratories, DSTO Australia, Australia Telecom and three companies he cofounded, Communica, LightSpin Technologies, and Polymer Braille Inc. He is currently the Cirrus Logic Distinguished Professor of electrical and computer engineering with North Carolina State University, Raleigh, NC, USA. His current research interests center on building microsystems (systems constructed of silicon chips, both analog and digital, and silicon micomachined components) for applications in computing, communications, sensors, robotics, and signal processing.