DesignCon 2020

Self-Evolution Cascade Deep Learning Model for SerDes Adaptive Equalization

Bowen Li, Xilinx Inc.
bli@xilinx.com

Brandon Jiao, Xilinx Inc.
bjiao@xilinx.com

Chih-Hsun Chou, Xilinx Inc.
chihhsun@xilinx.com

Romi Mayder, Xilinx Inc.
romim@xilinx.com

Paul Franzon, North Carolina State University
paulf@ncsu.edu

Geoffrey Zhang, Xilinx Inc.
geoff.zhang@xilinx.com

Abstract

The IBIS Algorithmic Modeling Interface (IBIS-AMI) has become the de-facto methodology to model SerDes behavior for end-to-end high speed serial link simulations. Meanwhile, machine learning (ML) techniques can facilitate the computer to learn a black-box system. This paper proposes the Self-Evolution Cascade Deep Learning (SCDL) model to show a parallel approach to modeling effectively 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. After basic introduction of SCDL model, the paper shows examples whose results are highly correlated with the IBIS-AMI simulations, while achieving simulation time reduction of orders of magnitudes.

Authors Biography

Bowen Li is a Ph.D. student at North Carolina State University. His research topic is "High-speed Receiver Behavioral Modeling using Machine Learning". During his Ph.D., he interned in Hewlett Packard Enterprise and Samsung. He has over 5 years of experience with machine learning. He is a software engineer intern at Xilinx.

Brandon Jiao received his Ph.D. in 2005 in the field of electromagnetic field and microwave technology from Beijing University of Posts and Telecommunications. He joined Xilinx in 2014 and is currently the Senior Staff Transceiver engineer. Prior to joining Xilinx Brandon worked for Intel and Nortel. His current interest is SI/PI methodology in transceiver and RF applications.

Chih-Hsun Chou received the M.S. degrees in computer engineering from the University of Louisiana, Lafayette and Ph.D. degree in electrical engineering from the Department of Electrical and Computer Engineering, University of California at Riverside. He is currently working in Xilinx. His current research interests include system level design and architecture with a specific focus on PCIe acceleration platform for machine learning and SmartNIC.

Romi Mayder is currently the Senior Director of Technical Marketing Department at Xilinx, Inc. Prior to joining Xilinx, Mr. Mayder 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. Mr. Mayder has been employed by two companies in the Test and Measurement industry, Agilent Technologies and Anritsu (Wiltron) Company, where he specialized in microwave and millimeter wave microelectronic circuit design and fabrication. Mr. Mayder received his Bachelor of Science degree in Electrical Engineering and Computer Science from the University of California at Berkeley in 1992. Mr. Mayder has published 25 patent applications in the fields of signal and power integrity as well as semiconductor process technologies.

Paul Franzon is currently the Cirrus Logic Distinguished Professor of Electrical and Computer Engineering at North Carolina State University. He earned his Ph.D. from the University of Adelaide, Adelaide, Australia in 1988. He has also worked at AT&T Bell Laboratories, DSTO Australia, Australia Telecom and three companies he cofounded, Communica, LightSpin Technologies and Polymer Braille Inc. His current 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.

Geoff Zhang received his Ph.D. in 1997 in microwave engineering and signal processing from Iowa State University, Ames, Iowa. He joined Xilinx Inc. in June, 2013. Geoff is currently a Distinguished Engineer and Supervisor of Transceiver Architecture and Modeling Team, under SerDes Technology Group. Prior to joining Xilinx Geoff has employment experiences with HiSilicon, Huawei Technologies, LSI, Agere Systems, Lucent Technologies, and Texas Instruments. His current interest is in transceiver architecture modeling and system level end-to-end simulation, both electrical and optical.

1. 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]. Due to the intellectual property (IP) of the SerDes vendors, customers don't 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. IBIS-AMI modeling is currently the most commonly used approach to emulate receiver adaptation process.

While having high correlation to the real circuit behavior, the problem for the IBIS-AMI modeling is its 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 PVT to improve model accuracy. Combining with the fact that silicon venders provide product as well as the associated IBIS-AMI model to the customer, the IBIS-AMI model development can become the bottleneck for a design release.

In this paper, we propose a machine learning based framework to skip the long model design process. We develop a general modeling mechanism for the adaptation algorithm in the high-speed receiver leveraging machine learning techniques. Our model autonomously figures out the dependency/mutual information inside the system during the training process. The trained model not only provides highly correlated predictions, but also produces the result in the order of magnitude of seconds, thus the machine learning based technique can speed up the simulation compared with the conventional IBIS-AMI modeling approach.

2. A general modeling mechanism

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 as well as the details of the adaptation algorithm. As each product is different, model details cannot be all included until the final design is completed; this can lead to long design cycles. As a result, a general modeling mechanism is desired to speed up the model development cycle. To decouple the need of circuit knowledge in the model design, all the data we can acquire 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 proven to be very effective for the black box problem. However, the required training data grows 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.

To tackle these challenges, we build a general machine learning modeling framework to eliminate the time-consuming model tuning step in the machine learning model. We develop a novel yet intuitive mechanism to "self-evolve" through mixing low-level ML models and input data features. To the best of our knowledge, we are the first to explore the feasibility and to have built the ML based adaptation behavior model. In summary, our proposed general machine learning based modeling framework has the following capabilities:

- Find useful information from the model inputs and mutual information during the training.
- Find data dependency during the training
- Leverage its own experience to correct its learning process.
- Explore natural laws by itself.

3. Deep learning model structure introduction

In this work, two deep learning models are used in the SCDL model library. In this section, a brief introduction to the two deep learning models are presented.

Deep neural networks: Neural networks model is a technology to mimic the activity of the human brain. It can learn a black-box system. A simple application of the neural networks to do the data analysis is called Multi-Layer Perceptron (MLP). Compared with the MLP, deep neural networks (DNN) increases the hierarchy of complexity and abstraction. Fig. 1 shows the DNN structure. Each layer applies a nonlinear transformation onto its input and creates a statistical model as output from what it learned. The input features are received by the input layer and passed onto the first hidden layer. Each neuron in the hidden layers has weights and biases. Each neuron has an activation function which is used to

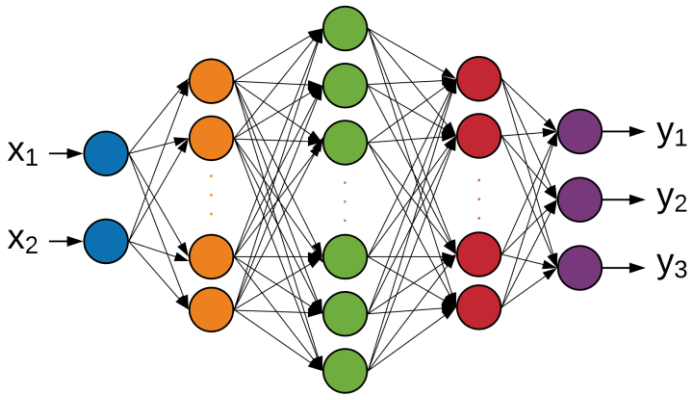


Fig. 1. Deep Neural Networks structure.

standardize the output from the neuron. The "Deep" in deep neural networks means there is more than one hidden layer in DNN. The output layer returns the output data. Until the output has reached an acceptable level of accuracy, training epochs will be continued.

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

$$\{z^h\} = \{x^{h-1}\}W^h \quad (1)$$

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

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

where f_a is the activation function. In this work, rectified Linear Unit (ReLU) [14] is used as the activation function, which is obtained as

$$f_a(z) = \max(0, z) \quad (3)$$

The output from the output layer is the prediction targets, $\{\hat{y}\}$. The stochastic gradient descent (SGD) method [15] is used to minimize the cost function.

In this research, the Bayesian Optimization method is applied to select the best number of hidden layers and neurons of deep neural networks, which will be introduced in later sections. The mean square error between outputs and expected outputs is used as the cost function.

Long short-term memory: As a state-of-the-art deep learning architecture is designed for time-series regression problem. RNN is widely used in forecasting such a problem [8]. The RNN aims to map the input sequence x into outputs y. Each output in the sequence is calculated by the state of the previous Recurrent Neural Networks (RNN) cell and current input.

Traditional RNN or Vanilla RNN (VRNN) structure faces a challenge. If the training length of RNN is significantly long and non-truncated backpropagation is applied, due to a vanishing gradient or exploding gradient, backpropagated errors will get smaller or larger layer by layer, which makes backpropagation insignificant. To deal with the vanishing gradient or exploding gradient problems, the LSTM model is proposed [9]. The LSTM cell structure is shown in 0. LSTM can create paths where the gradient can flow for a long duration.

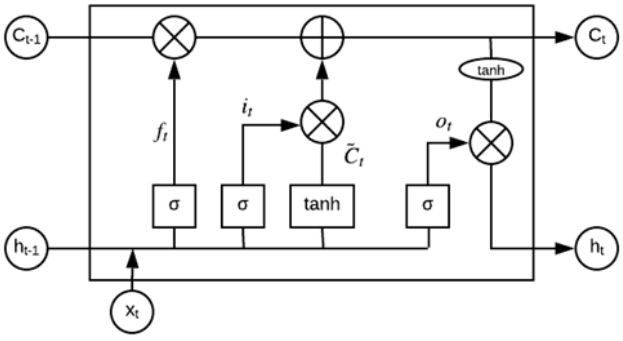


Fig. 2. LSTM cell structure

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 out of a sigmoid neural networks 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. And the output gate decides new sharing parameter vector from the current memory cell.

Mean square error (MSE) is used as the cost function during the DNN and LSTM model training process, which is defined as

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \quad (4)$$

where i is the index of data. y_i is the real target. $\hat{y}_i$ is the model predicted value. m is the number of data points.

4. Proposed modeling architecture

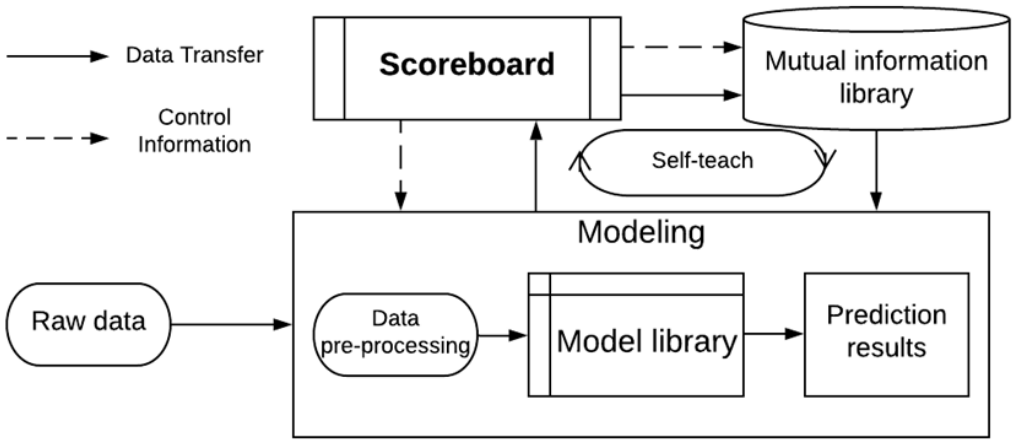


Fig. 3. Self-evolution Cascade Deep Learning architecture

In this work, we propose a modeling mechanism, named Self-Evolution Cascade Deep Learning (SCDL), 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. In the rest of this section, we will discuss theses blocks and their interaction in detail.

Modeling: the input to our framework is the receiver input waveform (**raw data**) and will be sent to the modeling block. There, the raw data are pre-processed for feature selection (**data pre-processing block**), where important features are extracted from the data pattern, such as the low-frequency and high-frequency signal amplitude and slices based on UI information provided in the product specification. The extracted features are then sent to the **model library** for ML model training.

In the model library, there are multiple machine learning model types, such as deep neural networks (DNN) and long short-term memory (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 our design, the modeling block will do the separated target modeling, which means that it will create one model for each target.

Scoreboard: The scoreboard controls the prediction flow of the SCDL system. It selects useful information from the pre-processed 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 model will be used independently. During the training phase, we use model self-configuration techniques, to be discussed in the next section, 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 pre-defined 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 satisfies the prediction accuracy threshold is 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, 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 **mutual information library**.

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 mutual information library 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, we will then use all the features and mutual information there exist for training and prediction with each ML model independently. The reason behind this is that the target we want to model could have very different behavior than what we had successfully learned. We need to start from scratch with the added mutual information. 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 pre-set accuracy requirements set by a human. The second scenario is when it tries all the models and training iterations and finds the performance cannot be improved under the current circumstances, it will stop and generate a report. The auto-generated report includes 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 work, the SCDL model shows a parallel approach to modeling adaptive SerDes behavior effectively, illustrated in Fig. 4. 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

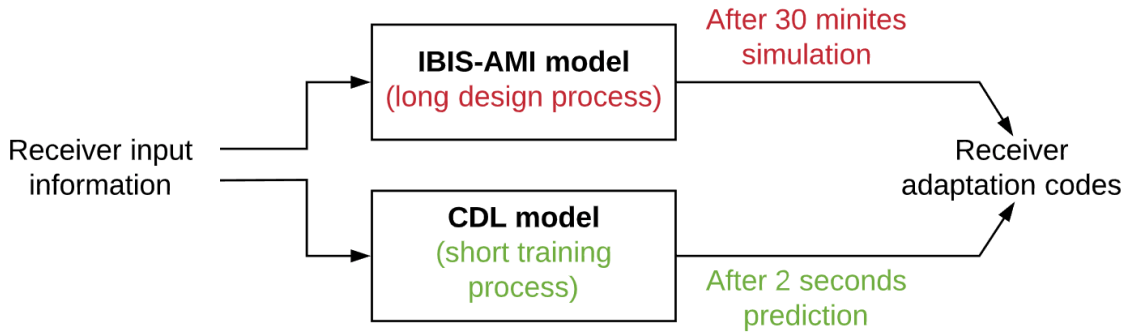


Fig. 4. Traditional IBIS-AMI model vs SCDL model in the receiver adaptation simulation

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.

5. Model self-configuration

The use of machine learning algorithms frequently involves careful tuning of learning parameters and model hyperparameters. Unfortunately, this machine learning model tuning requires lots of expert experience or even brute force search. Therefore, a great appeal for automatic approaches, called the Bayesian Optimization (BayesOpt), is proposed [16], which can optimize the performance of the machine learning model. In this work, the BayesOpt is used to optimize the model configuration.

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

$$\theta^* = \underset{\theta \in \mathcal{X}}{argmax}\, f(\theta) \tag{5}$$

where $\mathcal{X} \subseteq \mathbb{R}^{D_{\mathcal{X}}}$ is the design space or data space. $D_{\mathcal{X}}$ is the dimensionality of the parameter space. Furthermore, it is assumed that the unknown objective function f has can be evaluated at any arbitrary query point θ in the data space. This evaluation produces outputs $y \in \mathbb{R}$ such that $\mathbb{E}[\mathrm{y}|f(\theta)] = f(\theta)$. Hence, we can only observe the function f through unbiased noisy point-wise observations y. In this setting, we consider a sequential search algorithm 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 a final recommendation θ_N, which is the best estimate.

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, BayesOpt can sequentially induce acquisition functions Υ_n: $\mathcal{X} \longmapsto \mathbb{R}$ 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.

6. Data generation

Fig. 5. Various data patterns

For the receiver adaptation system, the inputs are the receiver input waveform, while the outputs are the receiver adaptation codes. We prepared multiple input data pattern, namely pseudorandom binary sequence (PRBS), single bit response (SBR), long pulse response (LPR), as shown in Fig. 5. PRBS data can provide intersymbol interference (ISI) and channel loss information. 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, we collect the receiver input waveform and the adaptation codes as the SCDL model inputs and outputs, as shown in Fig. 6. The receiver used in this work has 18 adaptation codes.

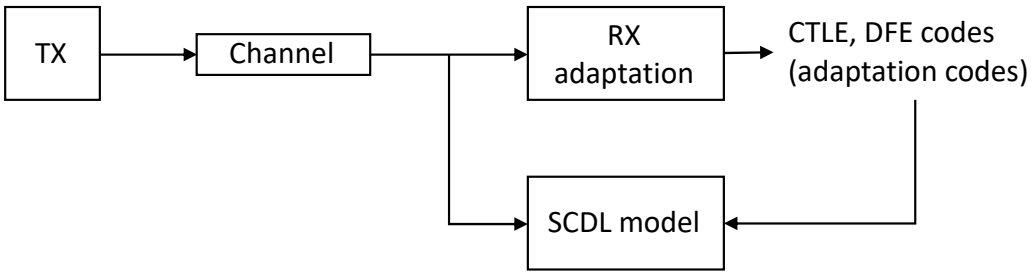


Fig. 6. Data collection process

In this work, 700 cases are measured from the transceiver. Those cases are collected over multiple industry channels. By tuning the TX de-emphasis settings, we can generate several cases over one channel. Among all the TX settings, low, medium and high swing cases are selected because they can cover under-equalized, properly-equalized, and over-equalized cases. The data are measured under different channel cases and various TX de-emphasis settings. Those case scenarios are summarized in Table 1.

The data are divided into three groups. 500 data cases from 11 channels are set as the training dataset. 50 data cases from 2 channels are set as the validation data. The rest 176 data from 2 channels are set as testing data. Those 6 testing channel cases are hidden from the model training process. The targets are collected from the receiver adaptation codes after the adaptation process is done.

Table 1. 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

7. Performance metric

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. 7. However, even after the adaptation process, the receiver equalization codes are still dithering.

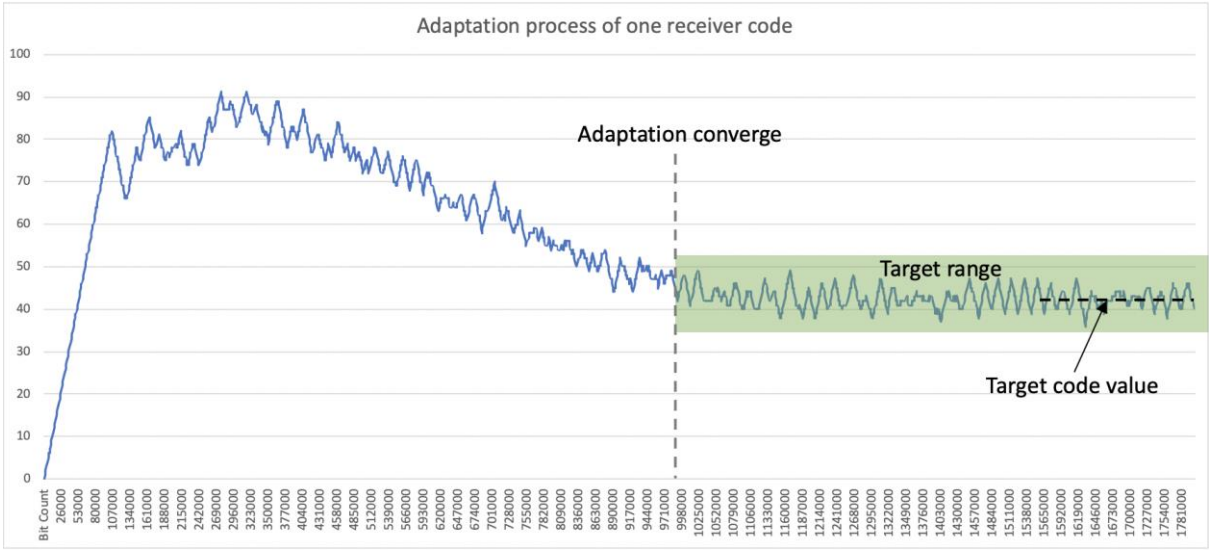


Fig. 7. Adaptation process for the receiver code.

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. 7. 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's a correct prediction. The prediction accuracy can be calculated as:

$$Prediction\ accuracy = \frac{n}{m} \times 100\% \quad (6)$$

where n is the number of cases that the predicted value is in the target range. m is the number of all the testing cases. At the model testing process, the prediction accuracy is calculated, which can describe the model performance.

8. Predict the receiver adaptation codes using the SCDL model

Fig. 8. Gives the detailed flow chart of our SCDL framework. In this section, we will describe how SCDL framework works in detail. At first, all the raw inputs are sent to the data pre-processing block for feature extraction, where the important features are measured from the data pattern, such as the low-frequency, high-frequency signal amplitude and slices of receiver waveform. The extracted feature will

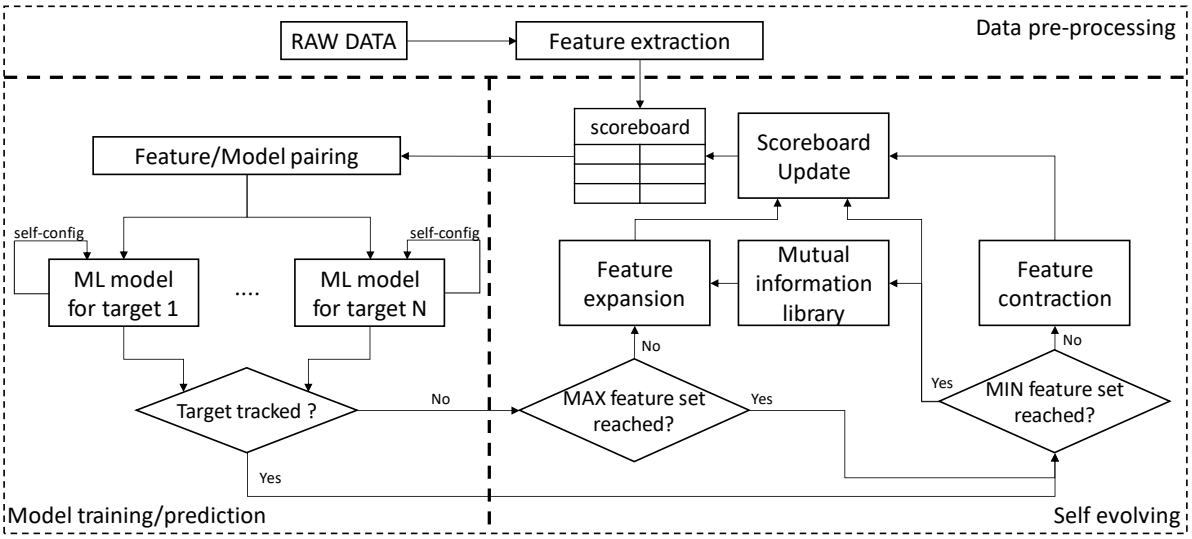


Fig. 8. SCDL flow chart.

be sent to the scoreboard as the base data set. As a design choice out SCDL will start from using simple ML model (DNN). This choice provides shorter learning time overall during our 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 exam 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 work, the receiver has 18 adaptation codes, which labeled target 1 – 18. 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 the too many features will hurt the accuracy, thus feature contraction phase will try to cut down the input feature.

In our 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, 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% - 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 contains lots of redundant information, which would mislead the machine learning model. On the other hand, simple data pattern, 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 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 target 1 – 3, while other targets show low correlations with the real code values.

After target accuracy is met, the SCDL will go into 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 the target 1 – 3 prediction performance the most. With these two features, the ML model complexity drop significantly, 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 target 1 – 3. At this point, all the predicted target 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 target 1 – 3, the SCDL also record that target 4 and target 5 show better prediction accuracies. To further improve the prediction accuracy, SCDL will add mutual information from the MIL, which has been successfully predicted target values, as a new feature set to predict target 4 and 5. After training, the prediction accuracies for target 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 the rest of targets, the previous features can not provide high-precision accuracies, the scoreboard switches to use the LSTM model to predict the target 6 – 18. Since the input LSTM model should be time-series data, the SBR and LPR data are sent to the LSTM model separately to predict the target 6 – 18. After training, the LSTM model shows better performance using the SBR data than the LPR data. Out of all targets, target 6 and target 7 show better performance than others, with the accuracies are above 70 %. The scoreboard focuses on performance improvement for these two targets until the prediction accuracies meet the pre-set accuracy requirements. At first, target 6 and target 7 are predicted using two individual LSTM models. The sliced SBR data are the LSTM model inputs. For example, the first, second, 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 target 1 – 5. However, these feature can not be applied to LSTM model. As a result, a DNN model in added in cascade with 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 target 1 – 5, to predict final target 6. In this way, the prediction accuracy of the target 6 is much improved, which are around 88%. Next, feature contraction reduces the inputs of the DNN model. After input reduction, the prediction accuracy of the target 6 is above 90%. Then the target 6 is saved into the MIL. The scoreboard saves this successful experience for future modeling. The same logic applies to the target 7 - 18.

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. One thing to notice is that we also set a counter to count how many tries a target is been through. After going through too many tries (exceeding a pre-determined threshold), it will stop and ask for more data and/or more ML models.

9. 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, we will demonstrate these abilities in detail.

9.1. Use mutual information

When lower accuracy requirements are set for each target, the first prediction flow chart is shown in Fig. 9. 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.

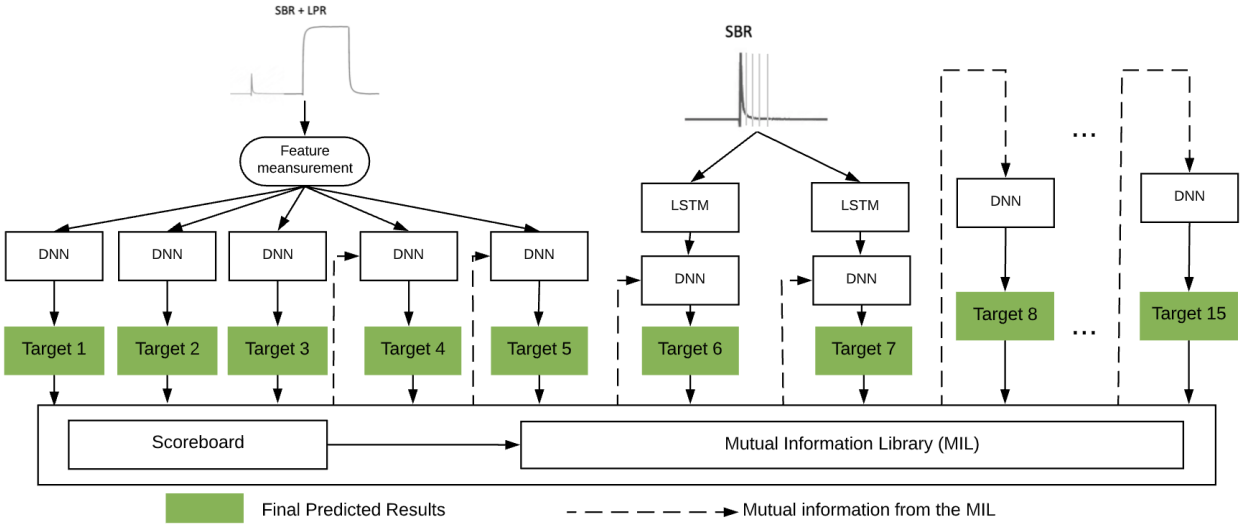


Fig. 9. The first prediction flow

As for the rest of the targets, the scoreboard uses two DNN models to predict target 4 and 5. The inputs of each DNN model are 2 measured features from the SBR and LPR data and mutual information from MIL. Similarly, the successfully predicted targets are sent to the MIL for future prediction.

For the target 6 – 18, the SCDL tries the DNN model at the first time, but the model prediction accuracies are below pre-set 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.

9.2. Optimize the prediction flow

As mentioned previously, the SCDL model will stop its training if it successfully meets the pre-set 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. 10. The prediction flows of the target 1 – 5 are the same as the first prediction flow.

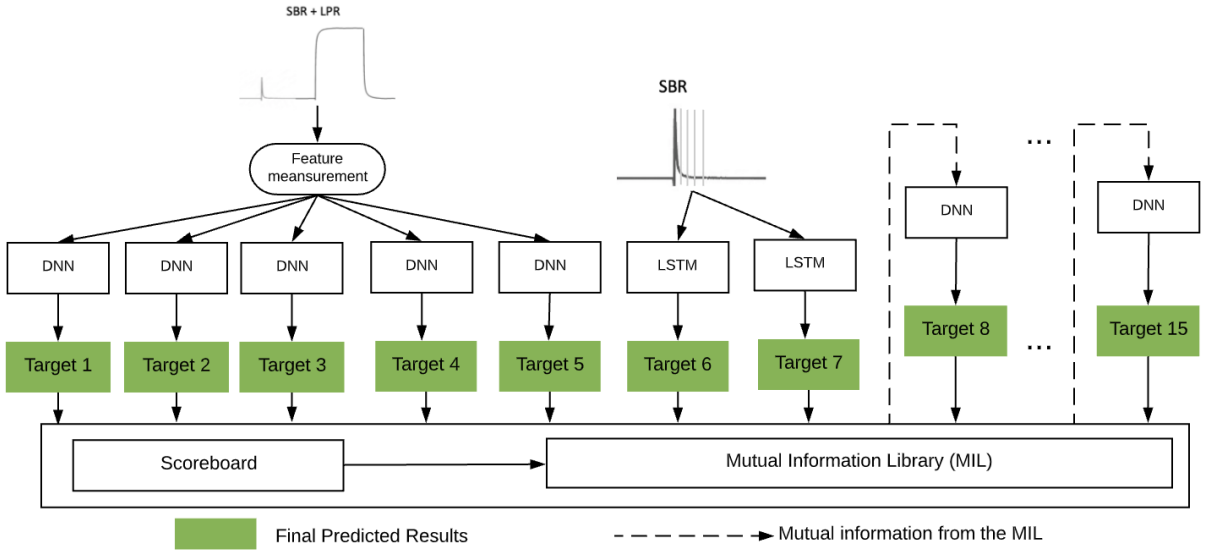


Fig. 10. Prediction flow optimization

For the rest of the targets, the SCDL finds a new way to construct models. The sliced SBR data are fed into two individual LSTM models and cascade another DNN model to predict the to predict target 6 - 7. The inputs of the DNN model are the LSTM model prediction results and mutual information from the MIL. As for the target 8 - 18 predictions, the scoreboard use only mutual information with single DNN model for each target. 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.

9.3. Find data dependency

After the SCDL model finishes its modeling process the SCDL model will enter feature contraction phase and try to reduce the input information for each model to speed up the overall training and data generation process. During the feature contraction process, the scoreboard selects important mutual

Table 2. 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
Target 8	DNN	Target 3-7
Target 9	DNN	Target 3-8
Target 10	DNN	Target 3-9
Target 11	DNN	Target 3-10
Target 12	DNN	Target 3-11
Target 13	DNN	Target 3-12
Target 14	DNN	Target 3-13
Target 15	DNN	Target 3-14
Target 16	DNN	Target 3-15
Target 17	DNN	Target 3-16
Target 18	DNN	Target 3-17

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 drops significantly, 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 2 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.

10. An example of target 6 prediction evolution process

The prediction accuracy improvement process of the target 6 is shown in Fig. 11. The pre-set accuracy threshold is 95%.

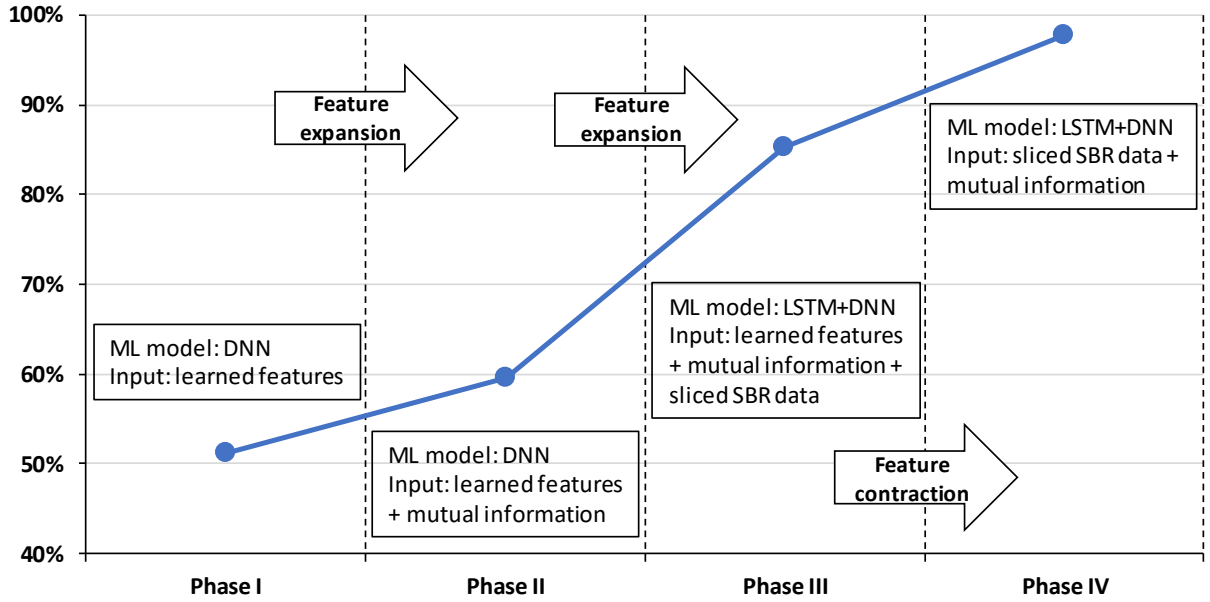


Fig. 11. Target 6 prediction accuracy improvement process

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, 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 the Phase I of Fig. 11. Next, SCDL will enter feature expansion stage. Since the SCDL model does the parallel modeling, target 1-5 prediction accuracies meet the pre-set accuracy thresholds and saved into the MIL. The mutual information from the MIL and previous learned features will then be used with the DNN model to predict the target 6 again. After training, the prediction accuracy is still below the pre-set accuracy threshold, as shown in Phase II of Fig. 11. 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. 11. Finally, after trying with all possible input and still miss the accuracy threshold, the SCDL will enter feature contraction phase try to eliminate feature that hurts the performance. The input features are removed one by one until the accuracy meets the pre-set requirements, illustrated in Phase IV of Fig. 11. The SCDL model will save the successfully predicted target 6 into the MIL.

11. Self-evolution prediction results

The data are collected from the Xilinx UltraScale+ GTY transceiver [1] [2]. The receiver has total 18 adaptation codes. In this section, the self-evolution prediction results of the SCDL model are presented. 176 testing data are collected under 2 different channels and various TX de-emphasis settings.

After the SCDL model self-evolution, some of the prediction results for the target adaptation codes are illustrated in Fig. 12 and Fig. 13, respectively. X axis is the case index number, while 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. We can see that the prediction results of our SCDL framework fall nicely into the target range, showing high correlations with the real codes.

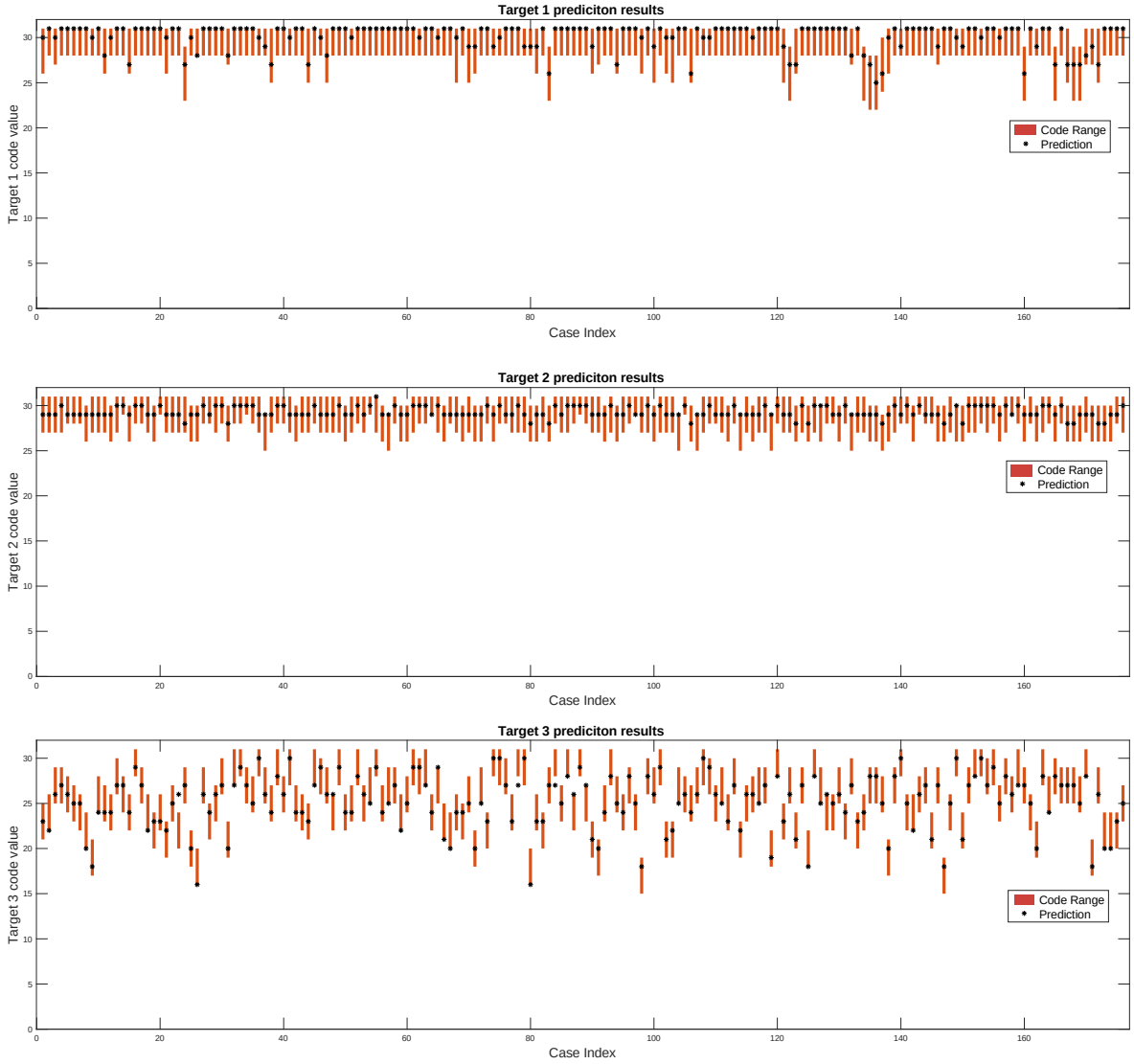
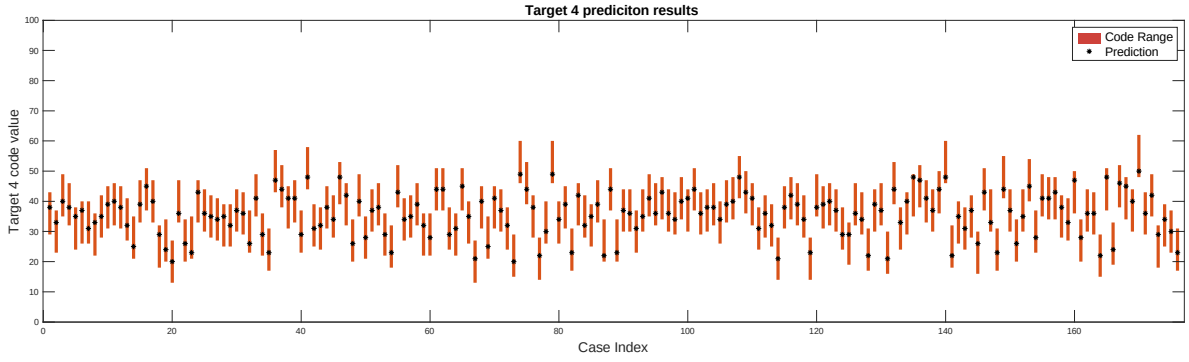


Fig. 12. Prediction results of target 1 - 3 using the SCDL model



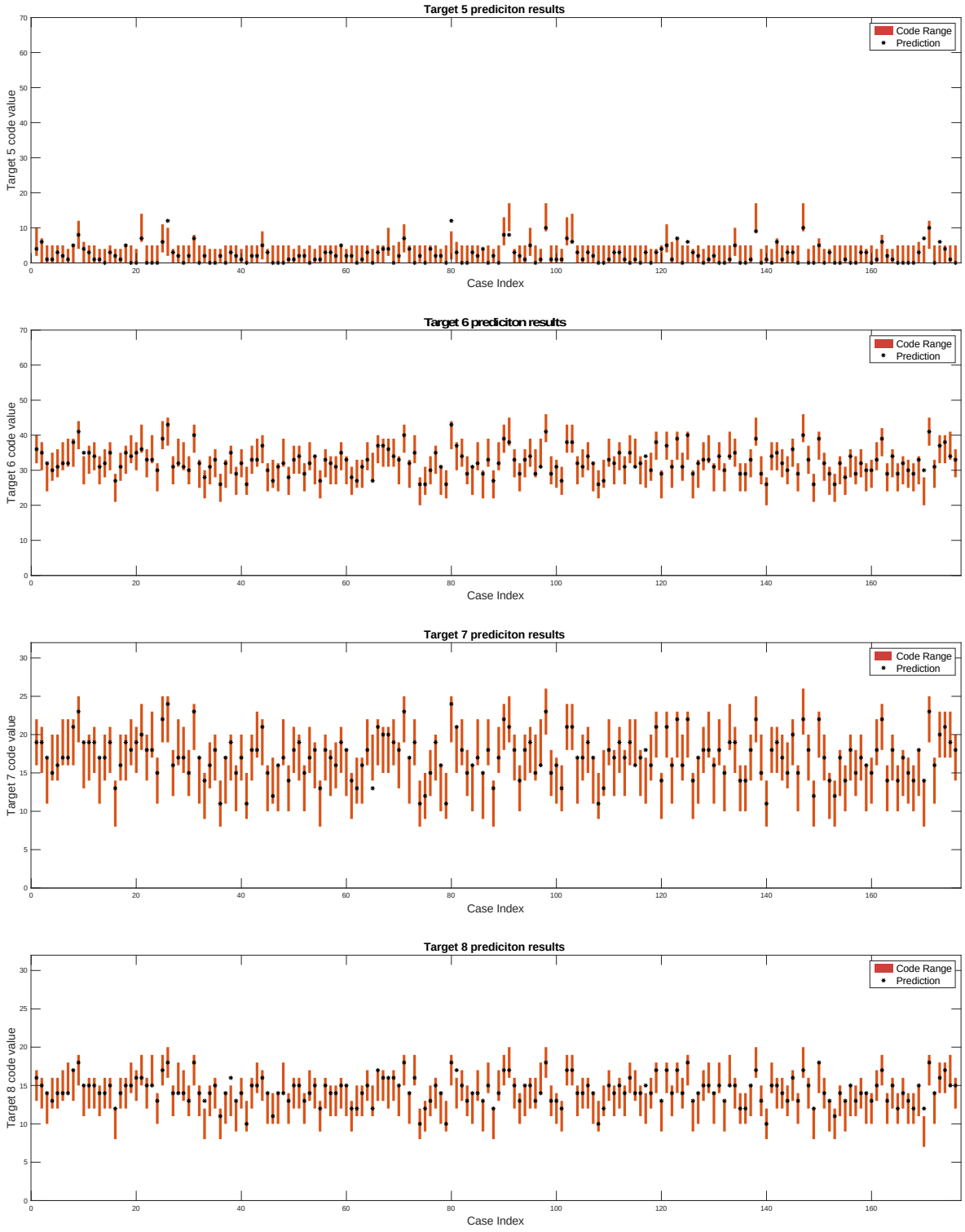


Fig. 13. Prediction results of target 4 - 8 using the SCDL model.

Table 3 shows the accuracy data for all 18 targets. In this table, we give two numbers: 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.

We can see that in the first attempt with all feature 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 exceeding our predefined threshold (90%), and achieved 98.9% accuracy in average across all targets.

Table 3. 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 10	63.6%	98.9%
Target 2	98.86%	98.86%	Target 11	61.9%	100%
Target 3	98.9%	98.9%	Target 12	62.2%	96.0%
Target 4	86.4%	100%	Target 13	71.9%	100%
Target 5	71.0%	98.9%	Target 14	72.8%	97.2%
Target 6	51.1%	97.7%	Target 15	78.4%	100%
Target 7	52.4%	97.7%	Target 16	79.2%	100%
Target 8	60.2%	97.2%	Target 17	81.3%	100%
Target 9	58.0%	100%	Target 18	80.2%	100%

12. Prediction explanation using the circuit structure

So far, we've shown that our SCDL framework can effectively predict the receiver adaptation codes. Yet this is not the full potential of SCDL. In this section, we will discuss the capability of SCDL to learn the circuit structure through self-evolution.

In this work, we use Xilinx UltraScale+ GTY transceiver [1] [2] as our 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. 14 [1]. CTLE is used to attenuate the low-frequency component of the signal while boosting the high-frequency component. The CTLE has 3 stages, namely KH, KL, and AGC.

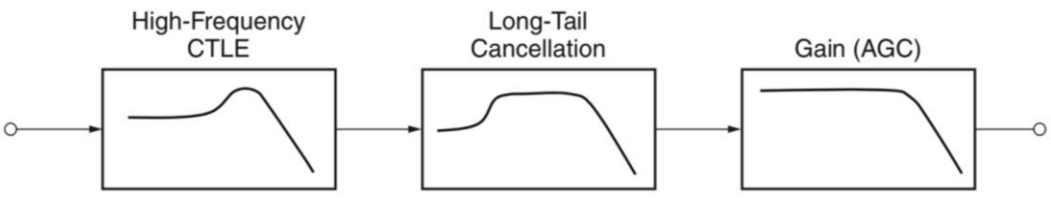


Fig. 14. Three-Stage CTLE in the UltraScale+ Transceiver

DFE is a proven technique to mitigate inter-symbol interference (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. 15 [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

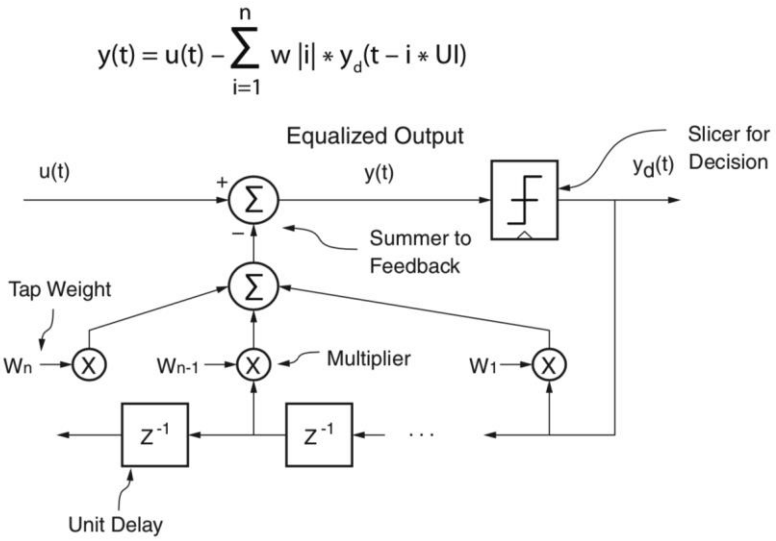


Fig. 15. DFE Block in the UltraScale+ Transceiver

has 15 taps. The CTLE codes and DFE taps are adaptive in the transceiver. Hence, the 3 CTLE adaptation codes and 15 DFE taps are the model prediction targets.

With the background of the transceiver, we can relate the prediction target to the adaptation code. Table 4 shows the corresponding receiver code of each target label. In this work, the CTLE has 3 stages, namely high-frequency gain (KH), low-frequency gain (KL), and automatic gain control (AGC). The receiver DFE has 15 taps.

Table 4. Target label mapping for the receiver adaptation code

Target label	Receiver code	Target label	Receiver code	Target label	Receiver code
Target 1	CTLE AGC	Target 7	DFE Tap4	Target 13	DFE Tap10
Target 2	CTLE KL	Target 8	DFE Tap5	Target 14	DFE Tap11
Target 3	CTLE KH	Target 9	DFE Tap6	Target 15	DFE Tap12
Target 4	DFE Tap1	Target 10	DFE Tap7	Target 16	DFE Tap13
Target 5	DFE Tap2	Target 11	DFE Tap8	Target 17	DFE Tap14
Target 6	DFE Tap3	Target 12	DFE Tap9	Target 18	DFE Tap15

According to the prediction flow in Fig. 10, three CTLE adaptation codes have high correlations with the high-frequency and low-frequency signal amplitude 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. 10, 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 tap values are directly 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 the DFE tap 1 – 3 to predict the DFE tap 4 – 15. Moreover, from the self-evolution process in Fig. 9 and Fig. 10, 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. Higher gains in the CTLE can lead to lower values in the DFE taps. On the contrary, lower gains in the CTLE can lead to higher DFE tap values. To get better prediction results, the SCDL model found the interactions among the CTLE adaptation codes and the DFE tap values by itself.

From this work, the SCDL model generates a cascaded deep learning model structure to predict the complex receiver adaptation progress. It can be proven 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.

13. Simulation speed improvement: SCDL model vs. IBIS-AMI model

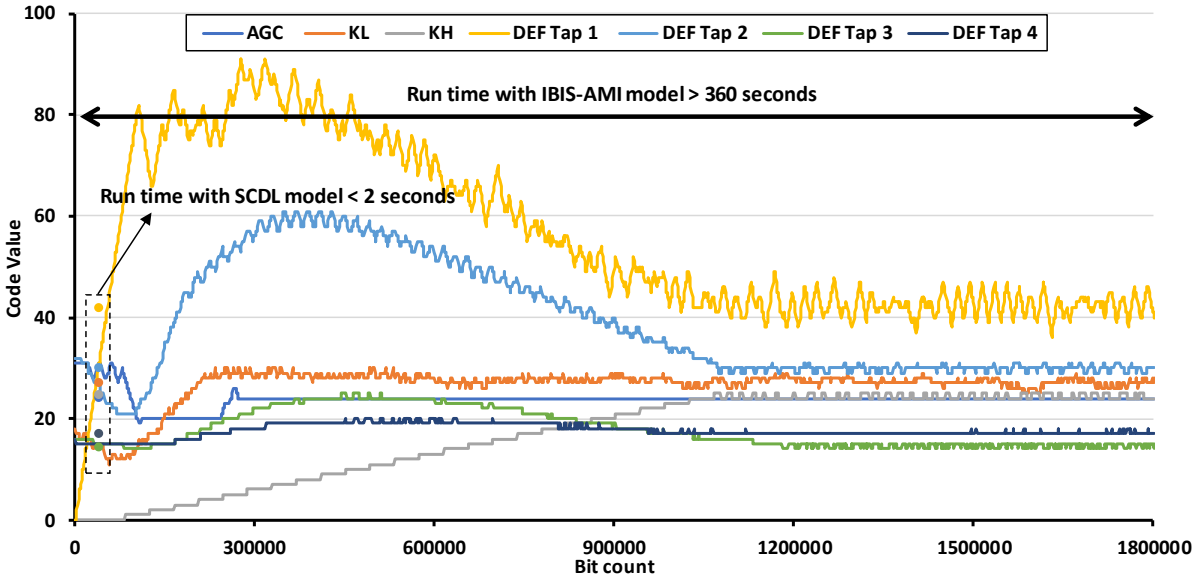


Fig. 16. Adaptation process using the SCDL model vs the IBIS-AMI model

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. 16 shows one example of an adaptation process using the SCDL model vs the IBIS-AMI model simulation. For IBIS-AMI model, the simulation needs to run for millions of bits until all of the codes converge, and this process takes significant amount of simulated bits. With the use of SCDL model, we can bypass the bit-by-bit simulation and give the accurate simulation output. With the SCDL model, the speed of the adaptation process significantly improves from more than 360 seconds (using the IBIS-AMI model) to less than 2 seconds (using the SCDL model), which is around 18000% improvement in the speed.

14. Conclusions

High-speed SerDes equalization parameter auto-tuning, a.k.a. adaptation, is a complex process. A new modeling mechanism, called Self-evolution Cascade Deep Learning (SCDL) model, is proposed and used in the prediction of high-speed SerDes receiver equalization adaptations.

Table 5 summarize the evolution process for our SCDL framework. In this table, the leftmost column gives all features, including mutual information library. The top row is the number of evolution that the target is learned. The second row is the code name for each target. Each column represents the data dependency for each target. The grey out cells mean that the data is not available for the target code, and the green cells represent the existence of the dependency between data and target code. From this table, we can clearly see that our SCDL framework explores and evolves through each evolution run.

During the training process, the SCDL model uses its own successful experiences to self-teach its future solution search and provide 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 pre-set 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 high-precision prediction results for both designs. For the simulation speed, the SCDL model can provide 180 times faster simulation than the state-of-the-art IBIS-AMI modeling approach.

In summary, our proposed modeling method achieves the following capabilities:

- Can leverage its own experience to correct its learning process.
- Can find useful information from the model inputs and mutual information during the training.
- Can explore the underlying circuit architecture.
- Can generate output code two order of magnitude faster than conventional method.

Our future work will focus on improving the model performance and test on other SerDes technologies.

Table 5. SCDL evolution track.

		Evolution I			Evolution II		Evolution III	Evolution IV	Evolution V	Evolution VI	Evolution VII	Evolution VIII	Evolution IX	Evolution X	Evolution XI	Evolution XII	Evolution XIII	Evolution XIV	Evolution XV
		CTLE AGC	CTLE KL	CTLE KH	DFE tap1	DFE tap2	DFE tap3	DFE tap4	DFE tap5	DFE tap6	DFE tap7	DFE tap8	DFE tap9	DFE tap10	DFE tap11	DFE tap12	DFE tap13	DFE tap14	DFE tap15
Extracted feature	PRBS																		
	SBR																		
	LPR																		
	Sliced SBR																		
Mutual Information Library	CTLE AGC																		
	CTLE KL																		
	CTLE KH																		
	DFE tap1																		
	DFE tap2																		
	DFE tap3																		
	DFE tap4																		
	DFE tap5																		
	DFE tap6																		
	DFE tap7																		
	DFE tap8																		
	DFE tap9																		
	DFE tap10																		
	DFE tap11																		
	DFE tap12																		
	DFE tap13																		
	DFE tap14																		

References

[1] B. Jiao, 'Leveraging UltraScale Architecture Transceivers for High-Speed Serial I/O Connectivity', White Paper: UltraScale GTH/GTY Transceivers, 2015

[2] 'UltraScale Architecture GTY Transceivers', User Guide, 2017

[3] H. Zhang, F. Rao, T. Keulenaer, K. Ly, R. Pierco, G. Zhang, 'IBIS-AMI Modeling and Simulation of Link Systems using Duobinary Signaling', DesignCon, 2017

[4] T. Lu, K. Wu, "Machine Learning Methods in High-Speed Channel Modeling", DesignCon 2019.

[5] R. Trinchero, F. G. Canavero, 'Modeling of eye diagram height in high-speed links via support vector machine', 2018 IEEE 22nd Workshop on Signal and Power Integrity (SPI), 22-25 May 2014

[6] B. Li, P. Franzon, Y. Choi, C. Cheng, 'Receiver Behavior Modeling based on System Identification', 2018 IEEE 27th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), 14-17 Oct. 2018

[7] B. Li, B. Jiao, M. Huang, R. Mayder, P. Franzon, 'Improved System Identification Modeling for High-speed Receiver', 2019 IEEE 28th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), 6-9 Oct. 2019

[8] H. Shi, M. Xu, and R. Li, 'Deep Learning for Household Load Forecasting – A Novel Pooling Deep RNN', IEEE Transactions on Smart Grid, Volume: PP, Issue: 99, 2017

[9] THESE ` N, 'Long Short-Term Memory in Recurrent Neural Networks', PhD Thesis, 2001

[10] J. Olden, Jackson, D.A., 2002. Illuminating the "black box": understanding variable contributions in artificial neural networks. Ecol. Model. 154, 135–150.

[11] Jasper Snoek, Hugo Larochelle, Ryan P. Adams, 'Practical Bayesian Optimization of Machine Learning Algorithms', Neural Information Processing Systems Conference(NIPS), 2012

[12] 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," Proceedings of the IEEE, vol. 104, no. 1, pp. 148–175, 2016.

[13] J. Mocˇkus, "On Bayesian methods for seeking the extremum," in Optimization Techniques IFIP Technical Conference, 1975, pp. 400– 404.

[14] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10, (USA), pp. 807–814, Omnipress, 2010.

[15] S. Ruder, "An overview of gradient descent optimization algorithms," arXiv preprint arXiv:1609.04747, 2016.

[16] J. Moˇckus, "Bayesian Approach to Global Optimization: Theory and Applications". Kluwer Academic Publishers, 1989.

[17] Application Report, SLLA338–June 2013, "The Benefits of Using Linear Equalization in Backplane and Cable Applications", Texas Instruments.

[18] Yu Liao, Echo Ma, Jinhua Chen, Geoff Zhang, "25G Long Reach Cable Link System Equalization Optimization", DesignCon 2016.