DESIGNCON® 2023
WHERE THE CHIP MEETS THE BOARD

Data-driven PAM4 SerDes Modeling with Generative Adversarial Network

Yongjin Choi, Hewlett Packard Enterprise
(yongjin.choi@hpe.com)

Priyank Kashyap, North Carolina State University
(pkashya2@ncsu.edu)

Wenzhen (Shawn) Sun, Tektronix
(shawn.sun@tektronix.com)

Muhammad Saad Chughtai, Tektronix
(muhammad.saad.chughtai@tektronix.com)

Chris Cheng, Hewlett Packard Enterprise
(chris.cheng@hpe.com)

Pual Franzon, North Carolina State University
(paulf@ncsu.edu)

Abstract

This paper will describe a novel method to model complex PAM4 SerDes using Conditional Generative Adversarial Networks (CGAN). Modeling is based on deep learning using data from test instrument captures or chip internal measurements. No knowledge of the SerDes equalizer structures such as CTLE or DFE is required to generate the model. Instead, two modules create the model: a generator that learns the distribution of the dataset and generates samples like the ground truth, and a discriminator that distinguishes whether a sample presented to it is fake or real. Together the two modules play a min-max game, where they try to outperform each other. The CGAN achieves convergence when the generator produces results below a detection accuracy threshold of the discriminator. The CGAN is conditioned on the input waveforms and the SerDes equalizer tap settings, and as such, the outputs of the CGAN change based on the tap value. A PAM4 SerDes is trained based on various channel and tap conditions. The data driven deep learning CGAN engine is generative and can produce Bit-Error-Rate (BER) contour outputs indistinguishable from the ground truth PAM4 BER contour measurements, even when presented with a previously unseen channel condition input waveform and equalization setting.

Author(s) Biography

YongJin Choi is a Master Technologist at the Storage Division of Hewlett-Packard Enterprise, where he leads the system failure prediction and SerDes modeling based on machine learning approach. He received a Ph.D. degree in electrical engineering from North Carolina State University in 2010. His technical interest includes applying machine-learning methodology to electronic system design.

Priyank Kashyap received a B.S. degree in Electrical and Computer Engineering from the New York Institute of Technology, New York, USA, in 2017 and an M.S. degree in Computer Engineering from North Carolina State University, North Carolina, USA, in 2022. He is also pursuing a Ph.D. degree in Computer Engineering at North Carolina State University. He has previously worked at Cadence, HPE, Samsung Semiconductors, and Synopsys. His current research interests lie in applying machine learning to EDA, hardware security, and quantum machine learning.

Wenzheng (Shawn) Sun is a Software Design Engineer with the digital signal processing group at Tektronix. He holds an M.S. in Electrical and Computer Engineering from Georgia Institute of Technology. Shawn's areas of expertise include receiver equalization techniques for high-speed serial standards and machine learning algorithm design for SI/PI measurements.

Muhammad Saad Chughtai is a Software Design Engineer at Tektronix where he is focused on designing measurement software for the latest high-speed digital standards. He received

his M. S. degree in Electrical and Computer Engineering from Georgia Tech and his B.S. degree in Electrical Engineering from Lahore University of Management Sciences (LUMS). Saad's areas of expertise include digital signal processing, machine learning, jitter and noise analysis as well as GPU computing.

Chris Cheng is a Distinguished Technologist at the Storage Division of Hewlett-Packard Enterprise. He is responsible for managing all high speed and electrical designs within the Storage Division. He also held senior engineering positions in Sun Microsystems where he developed the original GTL system bus with Bill Gunning. He was a Principal Engineer in Intel where he led high speed processor bus design team. He was the first hardware engineer in 3PAR and guided their high speed design effort until it was acquired by Hewlett Packard.

Paul D. Franzon is currently the Cirrus Logic Distinguished Professor and the Director of Graduate programs in the Department of Electrical and Computer Engineering at North Carolina State University. He earned his Ph.D. from the University of Adelaide, Adelaide, Australia. He has also worked at AT&T Bell Laboratories, DSTO Australia, Australia Telecom, Rambus, and four companies he cofounded: Communica, LightSpin Technologies, Polymer Braille Inc. and Indago Technologies. His current interests include applying machine learning to EDA, building AI accelerators, RFID, advanced packaging, 2.5D and 3DICs and secure chip design. He has led several major efforts and published over 300 papers in these areas. In 1993 he received an NSF Young Investigators Award, in 2001 was selected to join the NCSU Academy of Outstanding Teachers, in 2003, selected as a Distinguished Undergraduate Alumni Professor, received the Alcoa Research Award in 2005, the Board of Governors Teaching Award in 2014, and the Distinguished Graduate Alumni Professor in 2021. He has been awarded faculty awards from Qualcomm, IBM and Google. He served with the Australian Army Reserve for 13 years as an Infantry Soldier and Officer. He is a Fellow of the IEEE.

1. Introduction

As high-performance-computing and artificial intelligence platforms continue to require higher data rate between the chips, high-speed physical link specification has adopted Pulse Amplitude Modulation with 4 levels (PAM4) signaling [1]. Compared to binary non-return-to-zero (NRZ) signaling, the reduced signal margin in PAM4 requests more sophisticated receiver (Rx) circuits in that three signal eyes are individually tuned for the maximum signal-to-noise ratio (SNR). To concurrently optimize SNR of three eyes, Rx implements many control knobs that tune the circuits. Given the channel condition, we evaluate how well Rx circuits could recover distorted signals, with the PAM4 Rx model. Note that the channel is the medium between transmitter (Tx), Rx, and the surroundings.

The reduced SNR of PAM4 and the nonlinear channel/component effects require more accurate link evaluation by the time-domain (or bit-by-bit) simulation, not by the statistical simulation based on the impulse response. As the conventional approach using IBIS-AMI model (I/O Buffer Information Specification Algorithmic Model Interface) has developed with LTI (linear time invariant) and the single bit response [2], the predicted performance could be too optimistic for some channel conditions.

To overcome the limitation of the conventional IBIS-AMI model, we propose the data-driven modeling (DDM) of PAM4 Rx. The DDM, also called black-box modeling is to construct the model with the empirical dataset of system input /output. Since the system input/output relationship is only described by the experimental data, DDM does not need to include the sensitive design details, thus well protecting intellectual property. No assumption of LTI in DDM will generate the well-correlated output by incorporating the nonlinear effects from the measured data. The conditional generative adversarial network (CGAN) can model a receiver with complex equalization circuits such as CTLE and DFE with a single trained model based on encoded input and output images. With the generative nature of the model, it can interpolate intermediate equalization configuration and handle unseen channel loss, crosstalk conditions and varying bit-streams.

The contributions of our work are listed as follows:

- Our proposed model is the first data-driven, deep learning PAM4 model using Generative Adversarial Networks.
- For generating the model, GAF images converted from input waveforms are used as the input and BER contour images as output. They are data driven without needing to understand the underlying design knowledge.
- We demonstrate that the model is well-correlated to the various channel conditions and equalization settings by varying the loss, crosstalk, and equalizer gains.
- Our model output, BER contours will obviate the time-domain or bit-by-bit simulation, thus saving the engineering time.

The rest of the paper is as follows. Section 2 contains a brief discussion of prior work and the necessary background. Section 3 presents the proposed method, and section 4 discusses the PAM4 measurement setup and dataset generation process. Section 5 discusses the

cross-validation test set up and the resulting outputs vs. ground truth measurements. Finally, Section 6 concludes the paper and presents a brief overview of future work.

2. Background

Before describing how the GAN model is implemented in detail, we will introduce some topics that help the model generation.

2.1 BER (Bit-Error-Rate)

Previously, high-speed link performance was evaluated with the measured waveform at the input to the receiver with a sampling scope. The waveform is divided by the symbol period in time and overlaid each other, which is called eye diagram. The bigger open area at the center of the eye diagram implies the higher margin of the link. But as the link speed increases together with the challenging channel conditions, the eye diagram at the receiver input will frequently become the closed eye, thus no longer a good metric of the link performance. As an alternative, we can use BER contour [3]. BER contour is generated by sweeping the phase of the sampling clock and the threshold voltage that decide the logic values. Here we assume Rx circuitry has a feature to collect the error counts corresponding to a 2-dimensional sweep of the sampling clock phase and the threshold voltage. At each sweep point that samples the logic value, Rx counts the number of errors by comparing the expected logic values with the measured logic values.

2.2 GAF (Gramian Angular Field)

As the machine learning architecture was mainly developed for the computer vision or image classification, images are commonly utilized as the input to the architecture. To take advantage of the developed architecture with the time-series as an input, we need a mathematical expression that represents the time-series as images. As suggested in [4], GAF was selected for converting the time-series into images. GAF is linked to the inner product and Gram matrix. The inner product measures the similarity between two vectors, and the elements of Gram Matrix consist of the pair-wise inner product of n vectors. If we consider n as the nth time index, as the time increases, the matrix is calculated from top-left element to bottom-right element of Gram Matrix. Thus the temporal dependency is kept by the element locations. Since Gram Matrix produces the Gaussian noisy image, we need to encode the time-series into the polar coordinates first before applying Gram matrix like structure, which is GAF [5].

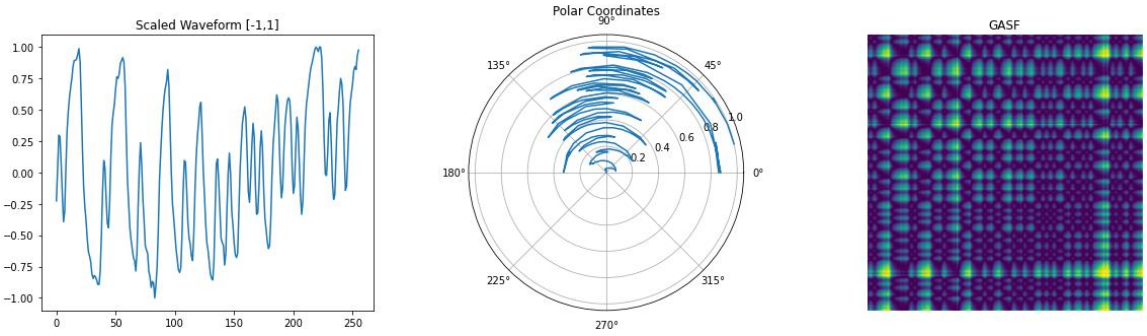
When $X = \{x_1, \dots x_i \dots, x_N\}$ is a time-series scaled to [-1,1] with a min-max scaler (Fig. 1 (a)), the time-series is mapped into polar coordinate by the following rule (Fig. 1 (b)):

$$\begin{cases} r_i = i/N \\ \emptyset_i = \arccos(x_i) \end{cases}$$

where r_i and $\emptyset_i$ represent the radius and angle at the time index i. As the inner product cannot hold two distinct angle information, GAF elements are defined by the summation of two angles and ignore the radius that increases by the time index. Thus GAF (or GASF) is defined as

$$GAF \triangleq \begin{pmatrix} \cos(\emptyset_1 + \emptyset_1) & \cos(\emptyset_1 + \emptyset_2) & \cdots & \cos(\emptyset_1 + \emptyset_N) \\ \cos(\emptyset_2 + \emptyset_1) & \cos(\emptyset_2 + \emptyset_2) & \cdots & \cos(\emptyset_2 + \emptyset_N) \\ \vdots & \vdots & \ddots & \vdots \\ \cos(\emptyset_N + \emptyset_1) & \cos(\emptyset_N + \emptyset_2) & \cdots & \cos(\emptyset_N + \emptyset_N) \end{pmatrix}$$

Fig. 1 (c) represents GAF as an image of the size, $N * N$.



(a) scaled waveform (b) polar coord. transform of (a) (c) GAF image

Fig. 1. Time-series into GAF image

2.3 GAN (Generative Adversarial Networks)

GAN [6,7] is a machine learning algorithm that creates authentic-like new data based on the training dataset. If the training dataset is a painter's example artworks, when successfully learned the probability distribution of the training dataset, GAN can generate the synthetic artworks indistinguishable from the painter's artworks. Creating new data is accomplished by the competing two modules that constitute GAN: generator G and discriminator network D. The generator network is analogous to counterfeiter and the discriminator is an expert that discerns the authenticity. The network training is finished when the discriminator is deceived, so that the counterfeit output of the generator is considered genuine. Once the training is done, the generator G will independently be deployed to create the synthetic dataset.

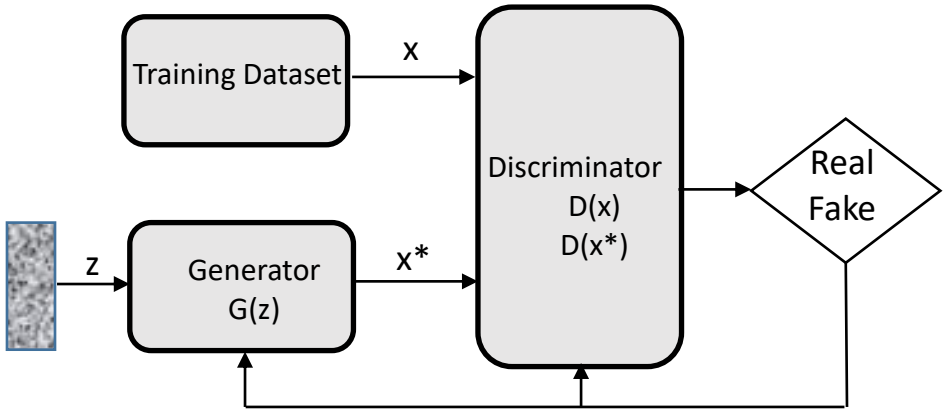


Fig. 2. Generative Adversarial Networks

The generator G(z) shown in Fig.2 is a mapping from latent space to the data space such as images x*, where z is the samples from the latent space. Note that the latent space is a vector space spanned by the latent variables that are not directly observable but are sufficient to describe the data. The discriminator D is a mapping from image data (x or x*) to the probability describing that the image data is from the authentic dataset. During the training process, the generator network is trained to fool the discriminator with the feedback of the discriminator output, whereas the discriminator is trained to classify the images as real if the input is from training images x, or fake if the images (x*) are from the generator.

2.4 CGAN (Conditional GAN)

For a certain application we want to control the generated output images. For example, if GAN is trained to generate the images of the hand-written numbers, we may want to generate only images having a specific number. In that case, we need to condition GAN with the specific number to be shown in the generated image, which is called conditional GAN (CGAN) [8]. Another example in semiconductor manufacturing is to replace lithography simulation with CGAN so that CGAN produces a resist pattern image conditioned with a mask pattern [9]. CGAN is constructed to feed the specific condition to the generator and the discriminator as shown in Fig. 3. The generator should create the image that fools the discriminator and correspond to the conditioned input image, y.

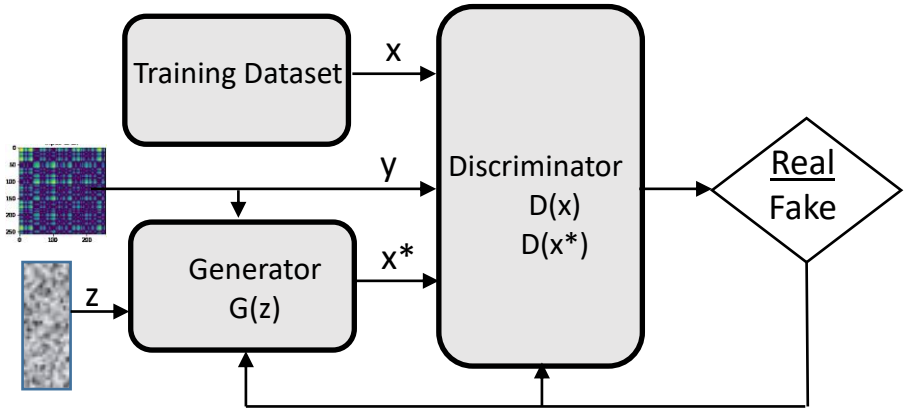


Fig. 3. Conditional Generative Adversarial Networks (CGAN)

3. Architecture of PAM-4 CGAN

The architecture of PAM4 CGAN is presented in Fig.4. The conditional inputs to the CGAN are GAF images converted from the measured input waveforms and Rx tap values of CTLE / DFE. The conditions direct CGAN to the real-looking BER contours. The different waveforms are measured at Rx for the various channel conditions. For each channel condition, we sweep Rx CTLE and DFE tap values, then collect the corresponding BER contours captured by Rx. The collected contours, GAF from the measured waveforms, and tap values are utilized as the training set. The discriminator will be trained to determine whether they are real or not. The generator output x* continues to be improved until the discriminator is tricked and takes x* as the real. Once the generator output x* achieves the

real-looking BER contours, the generator itself will independently produce the BER contour with the condition of GAF and tap values.

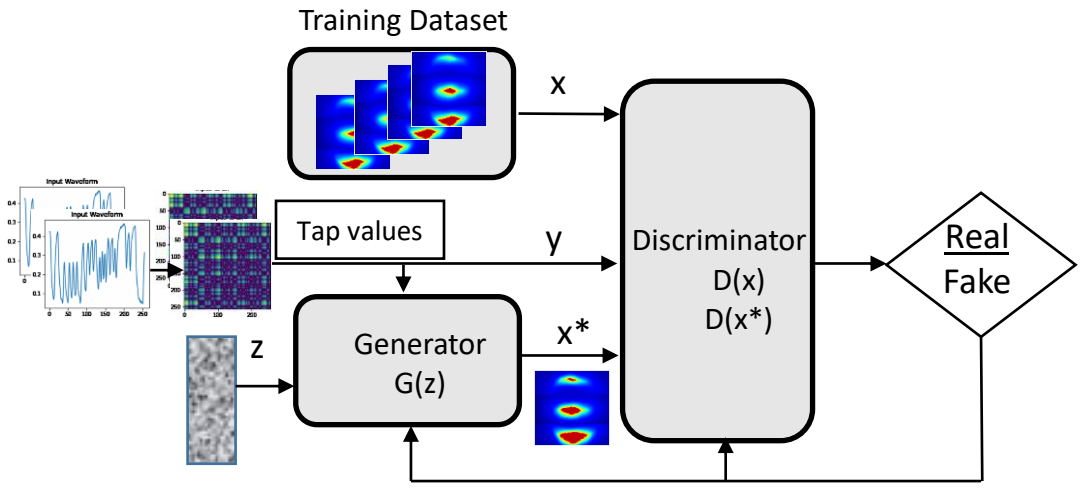


Fig. 4. PAM4 CGAN architecture

Since we want to estimate the mapping rule between the input image GAF and the output image BER contour, the problem is considered as image-to-image translation with CGAN. As suggested in [12], our generator and discriminator architectures follow U-Net structure, which consists of an encoder and a decoder block [11]. U-Net was originally targeted for biomedical image segmentation, which is the grouping of the pixels associated with the same class label. In our application, the encoder block converts the waveform image into feature representations at multiple levels by down-sampling the image. The decoder learns to construct the BER contour by combining the extracted features from the encoder, up-sampling the encoded features, and the tap values. Fig. 5 shows our U-Net discriminator originally proposed in [13]. The encoder of U-Net discriminator classifies input image of synthetic / ground truth BER as real or fake and the decoder classifies on a per-pixel basis. Since U-Net discriminator provides the real/fake information of the input image and each pixel to the generator, the generated image will be improved globally and locally.

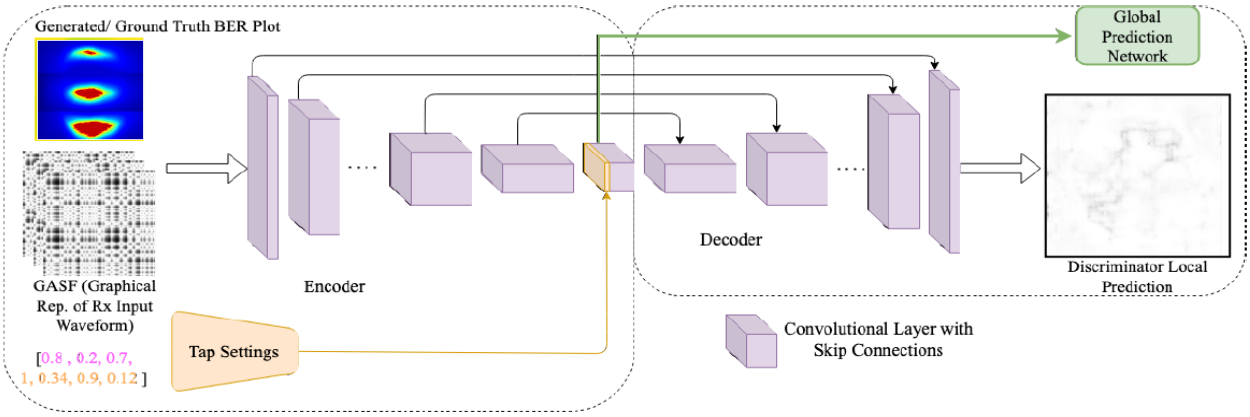


Fig. 5. Discriminator Architecture [10]

4. PAM-4 CGAN Generation Steps

Once we decide the architecture of PAM4 CGAN, the next step is to prepare the training dataset for the model.

(Step 1) Since the goal of PAM4 CGAN is to generate BER contours conditioned by the received waveforms and tap values, at first, we need to measure the various waveforms distorted by the different channel conditions. To mimic the different channel conditions, the loss and the crosstalk emulators are cascaded together as shown in Fig. 6 for 30 Gbps PAM-4. The channel loss and crosstalk levels can be independently adjusted to emulate changing channel conditions with different combinations of loss and crosstalk, just as real-life channel conditions in various systems. Due to the high loss of the crosstalk emulator, for 50Gbps PAM-4, we could not collect the open contours.

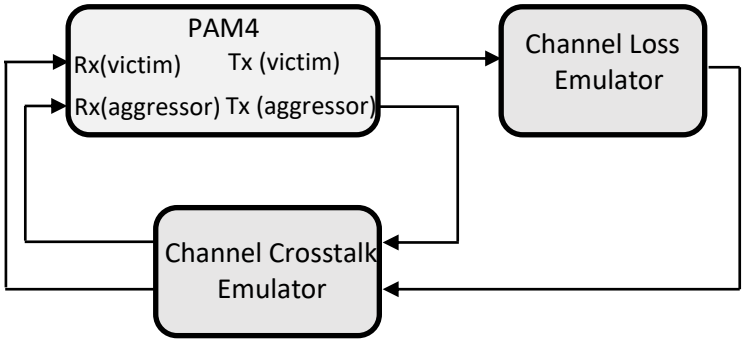
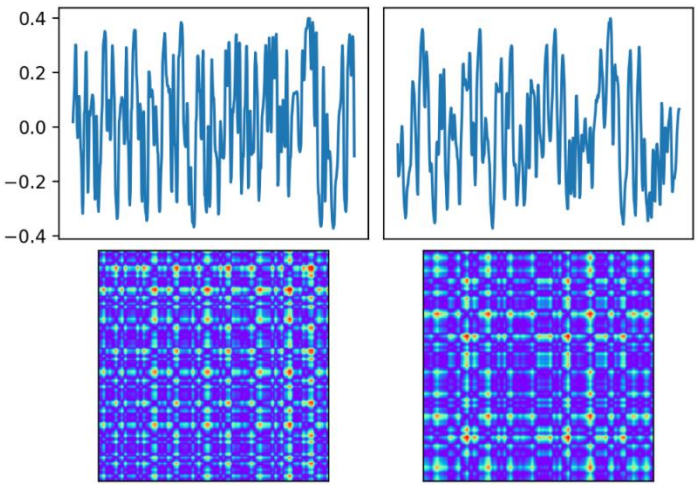


Fig. 6. Setup of the different channel conditions

(Step 2) By varying the channel loss/crosstalk levels, the input waveforms are measured at Rx (victim) port with the real-time scope. The first rows in Fig. 7 are 30Gbps PAM-4 waveforms measured at the input to the receiver and the second rows are the corresponding GAF images. (a) and (b) represent two different channel conditions set by the loss and crosstalk emulators. The GAF images are used as the input conditions to CGAN model.

(Step 3) Since the output of CGAN model is BER contour, we capture the contours corresponding to the GAF by sweeping CTLE and DFE tap values of the receiver. In the 30Gbps PAM4 experiment, for each channel condition, 512 tap configurations of CTLE and DFE are swept and the corresponding 512 BER contours are collected. Fig. 8 shows the captured BER contours. In Fig. 8, the first column is GAF and the rest of the columns display the contours for the various CTLE and DFE tap values, while each row represents the different channel condition. The red color in the contours represents the zero error. From the contours in Fig.8, we can notice the three eyes are recovered nonlinearly, possibly due to the nonlinear behavior of the receiver circuit. The overall data collection setup is presented in Fig. 9.



(a) channel #1 (b) channel #2

Fig. 7. PAM-4 waveforms and the corresponding Gramian angular field

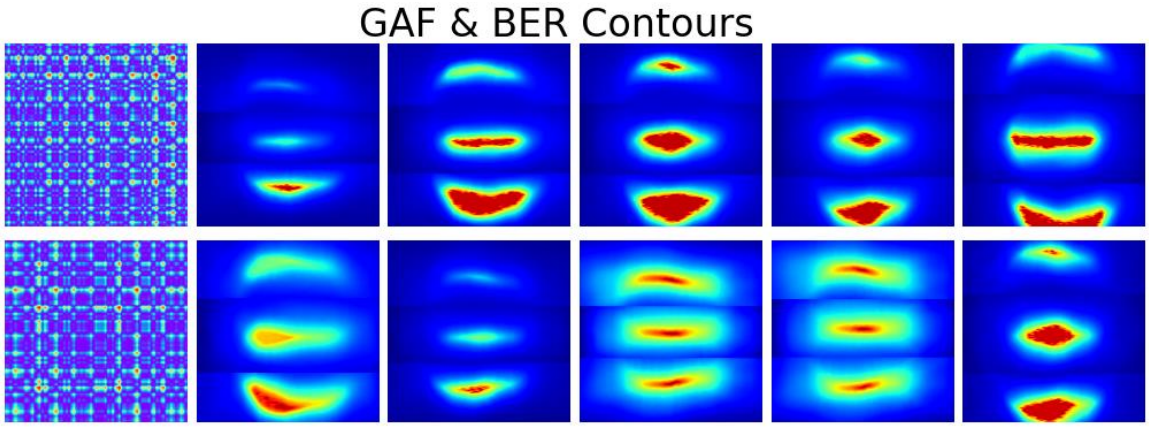


Fig. 8. 30 Gbps PAM-4 GAF and BER contours

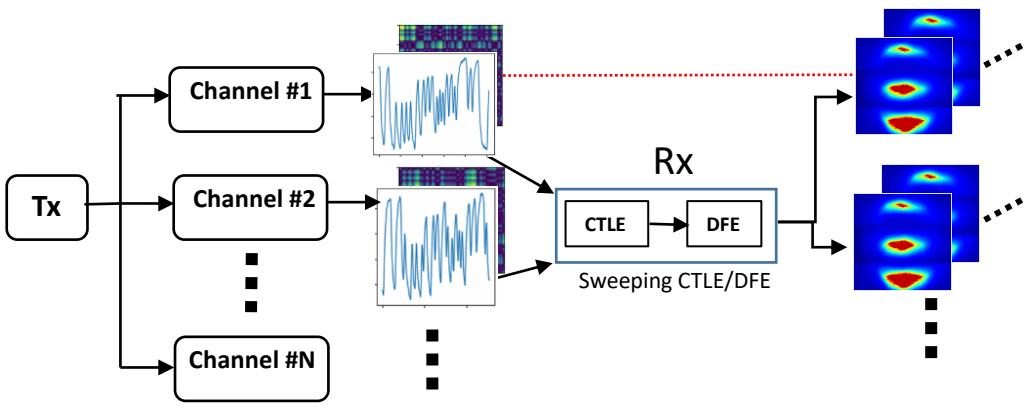


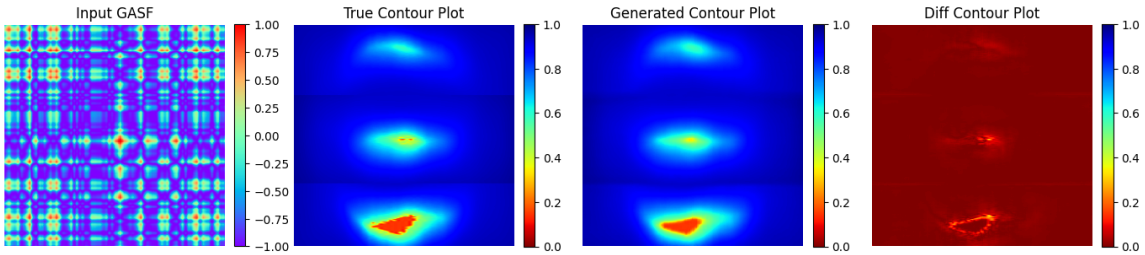
Fig. 9. Data collection setup for PAM4

(Step 4) Once we collect the GAF and BER contour images for each channel condition, build the CGAN model with Python language. We use Python 3.7 with TensorFlow 2.3.0 for the machine learning backend, pandas to read and process the data, scipy to resize the image, pyts to generate the GASFs, Numpy for additional processing, and scikit-learn to

split the dataset into training/validation/testing. The training takes a few hours with GPU installed in a server. With the training dataset, the model will learn the nonlinear-mapping function between the inputs and the corresponding outputs.

5. Model Evaluation

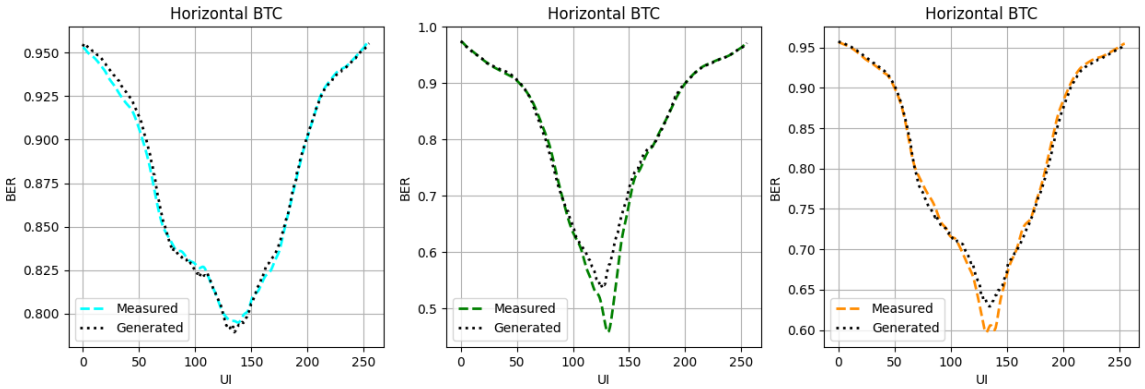
Once the model is trained with the measured dataset, we can test the model by comparing the generated BER contour with the ground truth BER contour. Fig. 10 (d) represents how close the generated contour is to the ground truth. The boundary of the red-colored area at the bottom contour shows the majority of the difference. Other than that, the generated one can safely replace the real contour.



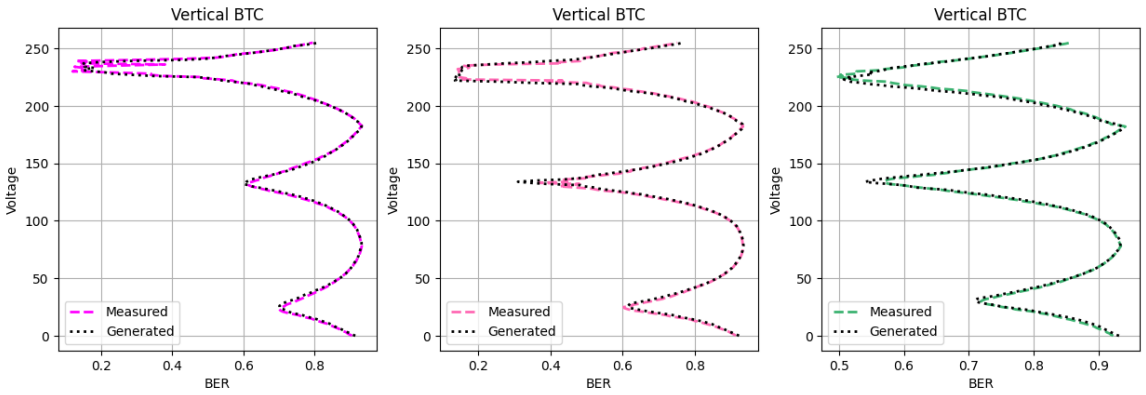
(a) GAF (b) True BER (c) Generated BER (d) Difference of (b) and (c)

Fig. 10. Comparison of the generated with the ground truth BER contour

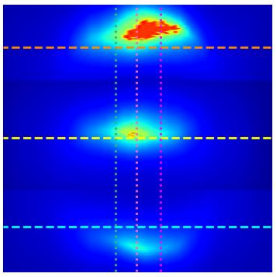
We can also show the horizontal BTC (Bathtub Curve) and vertical BTC comparison by cutting BER contours following the lines shown in Fig. 11(g). Fig. 11 (a)-(c) show the three horizontal BTCs cut at the center of one third area. Fig. 11 (d)-(f) show the three vertical BTCs cut by three vertical lines shown in Fig.11 (g). The black dotted ones are from the generated contours. The horizontal and vertical BTCs are well correlated with the generated horizontal and vertical BTCs.



(a) 1st horizontal BTC (b) 2nd horizontal BTC (c) 3rd horizontal BTC



(d) 1st vertical BTC (e) 2nd vertical BTC (f) 3rd vertical BTC



(g) Cut locations of horizontal and vertical Bathtubs on the generated and true BER contour

Fig. 11. Horizontal and vertical BTC comparison of the generated and ground truth BER contour

To evaluate the overall model quality of the trained PAM4 CGAN, we apply the test dataset of the different channel conditions and calculate the image difference of the generated and ground truth BER contours. The different amount of the loss and crosstalk is configured by setting the emulators with the percentage as shown in Fig. 13.

For each channel condition from channel loss, $l = [0,20,40,60]\%$ and the crosstalk, $c = [0,10,20,30,40,50,60]\%$, we collect M_{lc} number of ground truth BER contours for the test. Let's assume N is the total number of the pixels in a contour image, A_{mn} is nth pixel value of a ground truth contour from mth test file in a channel condition, (l, c), and g_{mn} is nth pixel value of a generated contour from mth tap configuration and GAF corresponding to a channel condition, (l, c). Then we can calculate the image difference with MAPE (Mean Absolute Percentage Error) as shown in the following:

$$MAPE(l,c) = \frac{1}{M_{lc}} \sum_{m=1}^{M_{lc}} \left(100 * \frac{1}{N} \sum_{n=1}^{N} \left| \frac{A_{mn} - g_{mn}}{A_{mn}} \right| \right)$$

The MAPE distribution with respect to the channel condition is shown in Fig. 12. Note the legend represents the channel loss. The worst MAPE value is 2.5% when the channel loss is 40% and the crosstalk 60%. From this result, we can conclude the data-driven PAM4 CGAN model generates the BER contours with the high accuracy across the various channel conditions.

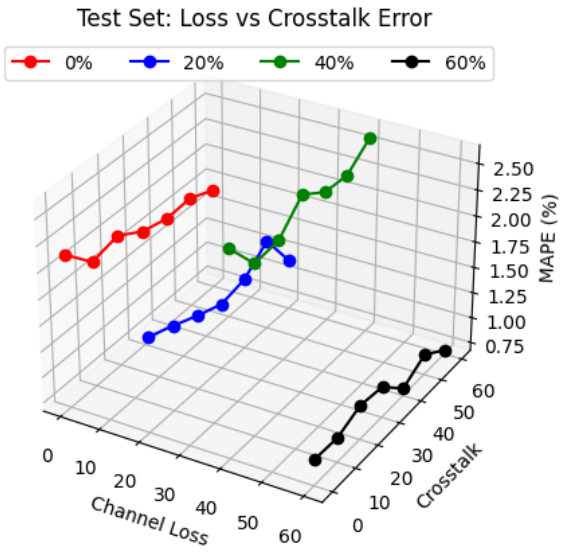
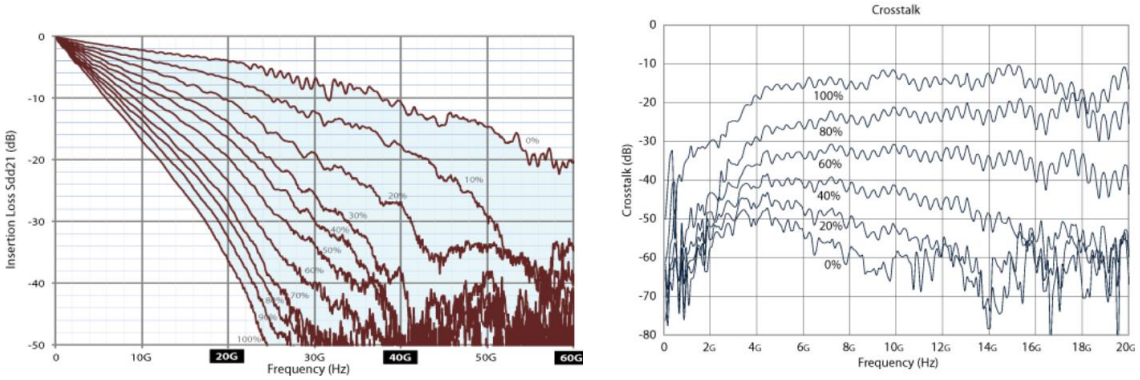


Fig. 12. Error distribution of the generated BER contours with the test dataset



(a) channel loss (Sdd21 vs frequency) (b) Crosstalk vs frequency

Fig. 13. Channel condition configuration by loss and crosstalk emulator

6. Conclusion

This paper presents a data-driven approach to modeling a high-speed PAM4 SerDes receiver with the help of a CGAN. We demonstrate the model's ability to handle different bitstreams, channel conditions with loss and crosstalk, and DFE and CTLE tap configurations. We show that the generated BER contour plots match the ground-truth BER plots visually, and multiple vertical and horizontal bathtub curves at the lower, center and upper eyes show high correlation. Moreover, we analyze the eye characteristics with respect to the ground-truth BER plots and show that the generated contour errors MAPE is less than 2.5%. We believe this data driven, deep learning approach has high potential to model multi-physics problems such as electrical to optical channels or electrical power to thermal distribution models. Since the model is not physics based and only relies on output

measures as training data, we can feed electrical/optical BER plots or thermal distribution to the modelling engine. Also, from our previous DesignCon paper [14], high dimensional channel and SerDes equalization structures can be mapped to a lower dimension space through machine learning techniques such as principal component analysis (PCA). The CGAN model can be further simplified by modelling the channel and equalization conditions in lower dimension PCA vector conditions.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. CNS 16-2137283 - Center for Advanced Electronics through Machine Learning (CAEML) and its industry members.

References

[1] https://pcisig.com/pci-express-6.0-specification
[2] https://www.signalintegrityjournal.com/articles/2020-back-to-basics-ibisibis-ami-and-the-path-to-lpdr5
[3] 2019. Bridging the gap between BER and eye diagrams. https://download.tek.com/document/65W_26019_0_Letter.pdf
[4] Wang, Zhiguang, and Tim Oates. "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks." Workshops at the twenty-ninth AAAI conference on artificial intelligence. 2015.
[5] Louis de Vitry. Encoding Time series as images. https://medium.com/analytics-vidhya/encoding-time-series-as-images-b043becbdbf3 (Nov. 2022)
[6] Ian, Goodfellow, et al. "Generative adversarial nets." In Advances in neural information processing systems." (2014): 2672-2680.
[7] Goodfellow, Ian, et al. "Generative adversarial networks." *Communications of the ACM* 63.11 (2020): 139-144.
[8] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).
[9] Ye, Wei, et al. "LithoGAN: End-to-end lithography modeling with generative adversarial networks." 2019 56th ACM/IEEE Design Automation Conference (DAC). IEEE, 2019.
[10] Kashyap, Priyank, et al. "RxGAN: Modeling High-Speed Receiver through Generative Adversarial Networks." *Proceedings of the 2022 ACM/IEEE Workshop on Machine Learning for CAD*. 2022.
[11] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and computer-assisted intervention*. Springer, Cham, 2015.
[12] Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
[13] Schonfeld, Edgar, Bernt Schiele, and Anna Khoreva. "A u-net based discriminator for generative adversarial networks." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020.
[14] Yongjin Choi and Chris Cheng. "Automatic channel condition detection & tuning using machine learning surrogate models for 56G PAM4 Channels." *DesignCon 2020.*