

High-Speed Receiver Transient Modeling with Generative Adversarial Networks

Priyank Kashyap^{*§}, Andries Deroo[‡], Dror Baron[†], Chau-Wai Wong[†], Tianfu Wu[†], Paul D. Franzon[†]

^{*}Hewlett Packard Enterprise, Colorado Springs, CO, USA

[†]ECE Dept., North Carolina State University, Raleigh, NC, USA

[‡]ASUSTeK Computer Inc., Taipei, Taiwan

Email: priyank.kashyap@hpe.com

Abstract—Data-intensive applications such as artificial intelligence and graph processing are becoming commonplace, requiring high-speed IO to enable the deployment of these critical applications. To accommodate the increasing data requirements Serializer/Deserializer (SerDes) receivers have become increasingly complex, with different equalization schemes to mitigate channel impairments. It has become increasingly important to model this receiver as they are performance-critical.

This paper shows an approach to modeling the transient of a high-speed receiver with fixed and varying equalization through generative networks. The method considers the receiver as a black box, with its inputs and outputs as two different domains, framing the problem as a domain translation task. The proposed approach uses an intermediate representation of the time series to model the receiver successfully. We demonstrate that the proposed method is invariant to the input waveform, receiver configuration, and channel. In a fixed equalization setting, the proposed approach has a root-mean-squared error of 0.016 in a [0,1] range and an error of 0.054 in the same range for a variable receiver. The approach can predict a batched set of results under 250ms, faster than an equivalent spice model for the same time steps.

Index Terms—Data-Driven, Generative, Macro-model, SerDes, Transient

I. INTRODUCTION

Demands for data-hungry applications are more prevalent than ever, requiring rapid data movement across different computing and storage systems. This data movement occurs across a serializer/deserializer (SerDes) transmitter and receiver over a lossy communication channel. SerDes receivers account for greater speeds by using filters or equalizers that handle channel impairments. Fig. 1 shows an example of a SerDes receiver with a continuous time linear equalizer (CTLE) and decision feedback equalizer (DFE) as the equalizers. An eye diagram, which overlays the transient waveform, 2-bit periods (UIs) at a time, on top of each other, provides information on whether the equalization works and whether the receiver can sample the data. However, equalization significantly increases the time it takes to run a spice model to generate an eye diagram.

Many different approaches tackle the problem of modeling a SerDes receiver’s performance. Some works look at it through the lens of predicting eye characteristics, providing limited performance information about the link [1], [2]. Others directly

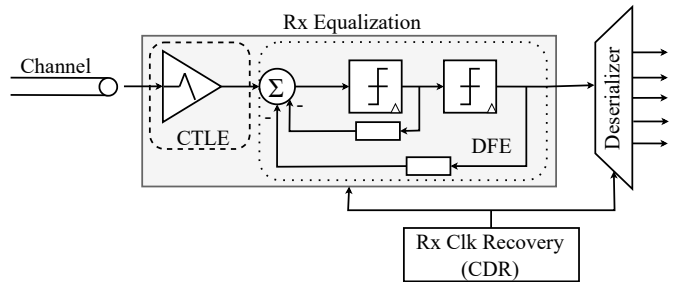


Fig. 1: SerDes receiver architecture with different equalization schemes that enables it recover the data.

model the receiver’s transient but require multiple models or evaluate the approach on the same link [3]–[6]. Recently, there has been a push to use generative adversarial networks (GANs) to model the receiver performance, but they treat the output as an image or limit it to a single pulse [7]–[9].

In this work, we propose using a conditional GAN (cGAN) to model the receiver’s transient. The work uses an intermediate representation of the time series, which preserves the temporal information, to frame the problem as an image-to-image problem. We demonstrate that the approach works for both a receiver with fixed and variable equalization settings. Further, we extend prior work by showing that the model can work on unseen channels and equalization configurations with a root-mean-squared error (RMSE) of 0.022 and 0.061 in a [0, 1] range, respectively.

The rest of the paper is organized as follows. Section 2 contains the necessary background for this work and is followed by a discussion on related work in Section 3. Section 4 presents the proposed cGAN approach and discusses the relevant hyperparameters for this work. Then, Section 5 provides the details about the two different datasets we use to evaluate the proposed approach. Section 6 contains the evaluation of the proposed approach on the two datasets from the previous section. Lastly, Section 7 concludes the paper and discusses potential directions for the proposed approach.

II. BACKGROUND

This section first presents the necessary background about a SerDes receiver and its components in the context of this

[§] Work done at North Carolina State University

work. It then discusses the intermediate representation of the time series before discussing cGANs.

A. High Speed Receivers

A SerDes enables high-speed data transfers between a transmitter and a receiver over a serial channel. The transmitter serializes the parallel data and sends it over a serial link. In contrast, the receiver parallelizes the data after it equalizes the incoming waveform to correct for channel impairments. Equalization plays a significant role in ensuring that the receiver can successfully capture the data at fast speeds. The equalization tackles the low-pass nature of the channel and can reduce the impact of impedance mismatches, which causes intersymbol interference (ISI), where sending one bit impacts subsequent bits.

1) *CTLE*: Receivers use a CTLE to mitigate the channel’s impact. High-speed channels have limited bandwidth, making them behave as low-pass filters, thus attenuating the signal at higher frequencies. The CTLE’s frequency response peaks at specific frequencies, boosting the channel’s overall frequency response at higher frequencies and counteracting the impact of the channel. Overall, the CTLE flattens the channel’s response at lower frequencies and increases the bandwidth, thereby equalizing it.

In this work, we limit the receiver’s equalization to the CTLE; however, most receivers also have DFEs that cancel out post-cursor ISI.

B. Gramian Angular Fields (GAFs)

With image-based neural networks having superior performance, many applications use intermediate representations to form time-series tasks as image tasks [10], [11]. One such intermediate representation is a Gramian angular field (GAF) [12]. Unlike other intermediate representations, such as Markov transition fields (MTFs) or recurrence plots (RPs), the GAF is a reversible transformation that retains the original time series’ temporal relationships in the converted image [12]. For these reasons, we leverage the GAF as the intermediate representation in the proposed approach.

To create a GAF representation of a time series, we first scale the time series between $[0, 1]$ or $[-1, 1]$. After scaling it, we convert it to the polar coordinate system by taking the angular cosine of each scaled time step. The last step converts the polar representations to the Gram matrix by taking the cosine of the sum or sine of the difference of each polar time step with each other to give the Gramian angular sum field (GASF) and Gramian angular difference field (GADF), respectively. This results in the Gram matrix being $m \times m$, where m is the number of time steps, thus making the transformation unsuitable for longer sequences. For a more in-depth derivation of the GAF, we point the reader to [12].

C. Generative Adversarial Networks

Generative adversarial networks (GANs) use adversarial training to learn the underlying data distributions. Unlike discriminative models, where the model predicts a particular

class, GANs contain two models, a generator and a discriminator, which play an adversarial game to learn the underlying distribution. The generator creates a sample \hat{x} given some latent variable z . The discriminator must distinguish whether a sample presented to it is from the dataset, x , or the generator, \hat{x} . The models train by playing this game where the generator aims to fool the discriminator, and the discriminator learns to distinguish whether a sample is real. Equation 1 shows this in practice.

$$L_{GAN}(G, D) = \mathbb{E}_y [\log D(x)] + \mathbb{E}_z [\log(1 - D(G(z)))] \quad (1)$$

The first term of the equation is the discriminator trying to determine where a sample is from, whereas the second term is the ability of the generator to trick the discriminator.

In the proposed method, we use a derivative of GANs, which conditions its output based on its input, aptly named a cGAN. The generator in the cGAN takes a conditional parameter, y , to generate an output, \hat{x} . The discriminator then gets the same conditional parameter with either the dataset sample, x , or a generated sample, \hat{x} . The discriminator predicts whether the combination is from the dataset or not. This conditional parameter modifies Equation 1 as follows:

$$L_{cGAN}(G, D) = \mathbb{E}_{x,y} [\log D(x|y)] + \mathbb{E}_{x,z} [\log(1 - D(G(y,z)|y))]. \quad (2)$$

Further, Isola et al. [13] find that adding an \mathcal{L}_1 loss term improves the generated image quality. This loss term is shown in Equation 3 with a weighted factor λ , a hyper-parameter.

$$L = L_{cGAN}(G, D) + \lambda \mathcal{L}_1. \quad (3)$$

III. RELATED WORK

In this section, we look at work that aims to model the transient of the high-speed receiver through machine learning and the application of GANs in electronic design automation (EDA).

Firstly, we look at works that model the receiver through system identification (SID) models. Choi and Cheng [3] use SID models to determine the impact of different channel conditions and equalization parameters. Li et al. [4] build on the work and use Auto-Regressive Moving Average eXternal input based on neural networks (NNARMAX), a non-linear SID model, to accurately capture the impact of the CTLE in a receiver. Though SID models enable quick training and inference, each parameter combination requires a unique SID model, thus requiring many SID models to simulate a receiver comprehensively.

The next class of models applies neural networks to model the receiver’s and, thereby, its transient. Nguyen et al. [14] use recurrent neural networks (RNNs) to model a fixed receiver’s transient but evaluate the same channel from which they derive the training set. Nguyen and Schutt-Aine [5] further the work by using a secondary network that uses the DFE parameters to set the initial state for an RNN to predict a single-bit

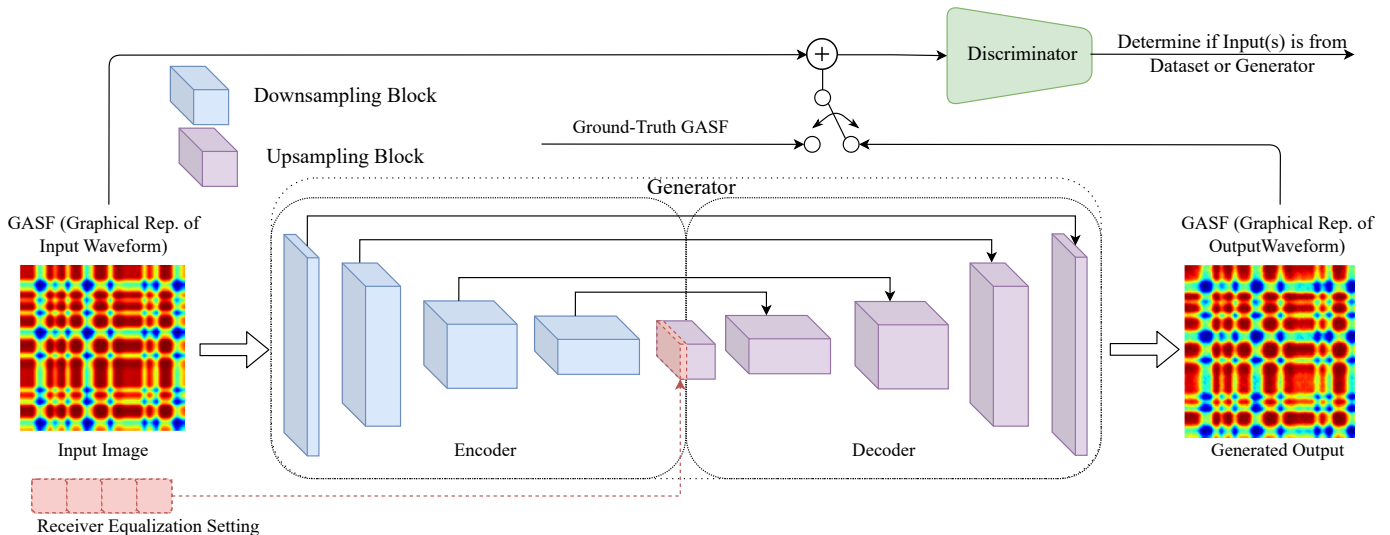


Fig. 2: The generator conditions its output on a GASF representation of the receiver’s input and, optionally, the receiver’s equalization settings to generate a GASF representation of the receiver’s post-equalization output. The discriminator gets the input(s) along with either the ground-truth GASF or the generated GASF and predicts whether the combination is real.

response. Goay et al. [6] predict the receiver’s transient using a convolutional neural network (CNN)-long short-term memory (LSTM) to predict the eye height and width for three different channel conditions, like Nguyen and Schutt-Aine [5]; they also evaluate the model on the same training channel.

GANs have found wide applications in EDA, from clock tree synthesis to generating images for lithography [15], [16]. Closer to the problem of receiver modeling, cGANs can predict the receiver’s performance as a bit-error ratio (BER) contour and eye diagram [7], [8]. However, in all these works, the output is an image, not a transient waveform, which is of interest in this work. Kashyap et al. [9] use a cGAN to model a transmitter by predicting a single pulse response, which does not yield the complete picture in the context of a receiver.

Unlike the works discussed here, the proposed approach is a single model for variable channels, receiver equalization, and bitstream. In the evaluation section, we demonstrate that the model can predict the transient accurately even when a channel is not in the training set.

IV. PROPOSED METHODOLOGY

The proposed method considers the receiver input as one domain and the receiver’s output after equalization as another. We aim to use a cGAN to perform a domain translation task by representing each waveform as a GAF, specifically the GASF. As we are not concerned with image quality but rather the underlying time series, we exploit the reversible nature of the GAF in the training procedure.

As mentioned in Section 2, GANs have two models: a generator and a discriminator. In this work, the generator is a U-Net, a network with an encoder that compresses the input and a decoder that reconstructs the desired output [17]. The encoder has skip connections to the decoder at the same resolution,

enabling the network to preserve information between the two modules [17]. Fig. 2 shows this generator architecture where the input is the GASF representation of the receiver’s input, and the output is the GASF of the post-equalization waveform. As seen in Fig. 2, to handle the tuneable receiver model, we extract features from it using a fully connected network and append it to the bottleneck phase of the generator. By doing so, the decoder considers the impact of the equalization setting.

The discriminator in this work is a U-Net, just like the generator, but with two outputs. Prior works have found that overall training improves when the discriminator’s output is not a simple binary classification on whether the samples are from the dataset [13], [18]. The U-Net discriminator provides two levels of predictions: a local, which indicates whether the network believes each pixel to be real or not, and a global, which uses the features in the bottleneck phase to indicate whether the entire combination of features is real [18]. The discriminator predictions over the input combination are given to the generator, enabling it to successfully refine regions that the discriminator can discern. We detail the model architectures for the generator and discriminator in the following subsection.

Until now, the generator and discriminator have been primarily image-based; however, we aim to recover a time series. To recover an accurate time series, we leverage the fact that the diagonal of the GAF is a special case with the original time series [12]. We incorporate this directly in the training loss by modifying the \mathcal{L}_1 loss in Equation 3 to be on the diagonal of the generated image rather than the entire image. The modification to the training enables the generator to recover an accurate waveform.

A. Hyperparameters

Model hyperparameter selection impacts the model’s overall performance and the required computational resources. In

general, there are two kinds of hyperparameters: those relating to the model’s architecture and those with the model’s training.

In terms of the model architecture, we use downsampling blocks with Convolution, LayerNorm, and ReLU activations and upsampling blocks with ConvolutionTranspose, LayerNorm, and LeakyReLU activations. The encoders and decoder throughout the cGAN use the downsampling and upsampling blocks, respectively. The upsampling layer upsamples the resolution by a factor of 2 in each layer, whereas the downsampling block downsamples by 2. The initial resolution for the generator and discriminator models is 256×256 , and each successive downsampling block reduces the resolution by half until it reaches the bottleneck where the resolution is an n element vector, where n is a hyperparameter corresponding to the latent space, in this case, 300. After the bottleneck phase, the upsampling block in the decoder has dropout layers with a value of 0.1 in both the generator and discriminator. The model concatenates the feature maps from the encoder with the previous upsampling output in the decoder.

The final layer in the generator has tanh activations, whereas the discriminators are sigmoid. The generator and discriminator have two fully connected layers to learn features about the equalization settings. Lastly, to get the global prediction from the bottleneck in the discriminator, the bottleneck connects to a fully connected network with a single output neuron.

We use a binary cross-entropy as the loss function and a λ of 150 for the \mathcal{L}_1 loss term. Each model uses an Adam optimizer with a learning rate of $2e^{-4}$ to optimize the loss. We train the model for 25 iterations with a batch size of 16 and enable early stopping on the training loss so that the training saves the best model if the training loss plateaus over 5 iterations.

V. DATASETS

This section describes the two datasets we use to evaluate the proposed approach. It delves into the data collection procedure along with the required preprocessing for the data.

A. Fixed Receiver

As the name suggests, we use a receiver model where the DFE and CTLE take on the same values regardless of the channel condition. We model the receiver running at 5 Gb/s using VerilogA with Cadence Virtuoso®, which enables varying the channel’s spice model. The channel is an RLGC model, where we vary the RLC parameters to obtain varying channel conditions while ensuring an open eye. The simulation captures the waveform at the receiver input and the output of the DFE at 5 ps intervals.

We collect 2000 different channel-pseudo-random binary sequence (PRBS) combinations for 12000-time steps, where the first 1500-2000 time steps are a training pattern that the receiver uses for clock data recovery locking, with each simulation taking 2 minutes on average. We discard the training pattern on the receiver’s input-output pair and downsample them by selecting samples at 50 ps intervals or every 10 samples based on the Nyquist rate. We split them into 256 time-step chunks to work with the model and generate the

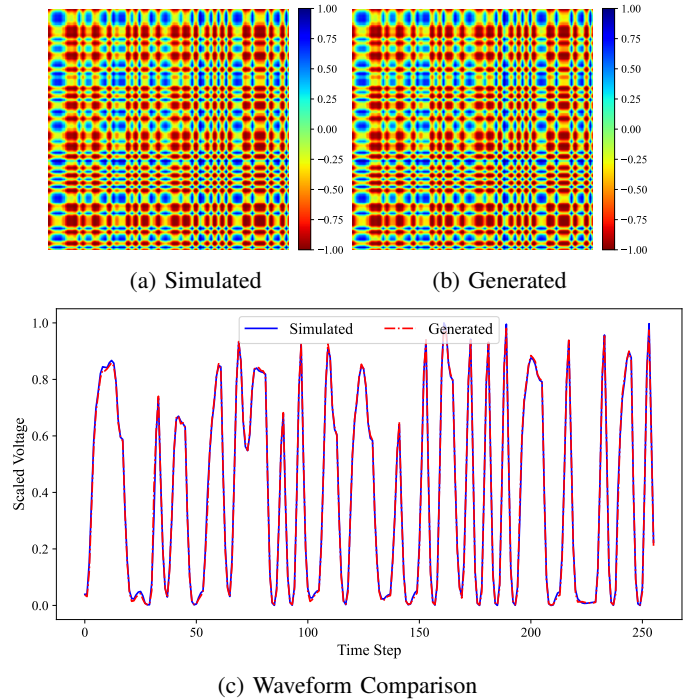


Fig. 3: Results for a fixed receiver with the ground truth and generated GAF in (a) and (b), respectively. Comparing the diagonals of the GAFs shows that the waveforms are highly correlated and contain minimal errors.

GASFs for them using PyTS [19]. Lastly, we randomly split the input-output chunks for the duration of the experiments in 60, 20, and 20 splits for the training, validation, and testing sets, respectively.

B. Tuneable Receiver

For the second dataset, we use a redriver running at 16 Gb/s with 2-tap configurations that enable the receiver to sample the waveform accurately. The taps take 16 and 3 unique configuration parameters, giving 48 unique receiver configurations. The 16 configurations for the first tap correspond to 16 different CTLE configurations, and the 3 taps for the second taps correspond to 3 different signaling protocols for the redriver. To stress the model, we look at 8 channel conditions with varying losses from [2, 10] db at 8 GHz. The channel and receiver configurations give us 384 possible combinations of the model.

The simulation captures the waveform at the output of the redriver transmitter and the output of the CTLE after being injected back into the output channel. It captures the waveform at 6.5 ps intervals or 10 samples for each bit period for 2000 bits. Based on the Nyquist frequency, we downsample the waveform and select every 3rd sample or a sample every 19.5 ps. Then, like the fixed receiver model, we split the waveforms into 128-step intervals, a lower resolution due to the lack of an extended bitstream sequence. Lastly, for each channel waveform, equalized waveform, and tap configuration, we randomly select 5 sets, 4 for training and one for testing.

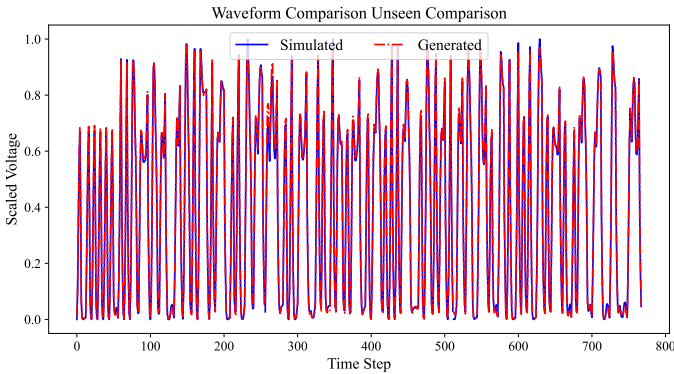


Fig. 4: Predictions on multiple windows for an unseen channel for a fixed receiver, showing minimal deviation from the simulated waveform.

VI. EXPERIMENTAL RESULTS

This section presents the results of our proposed approach using the previously described datasets. It first establishes a baseline for the fixed receiver dataset, followed by the model’s ability to predict unseen channels. We then show the model’s ability to predict the tuneable receiver before presenting results on unseen equalization configurations.

We trained all models on a system using a single NVIDIA 3080-TI with 32 GB of main memory and an Intel i7 1200K with Tensorflow 2.10.0 [20]. The datasets were collected on single-threaded CPU runs, unlike the proposed method, which is on a GPU.

A. Baseline - Fixed Receiver

Fig. 3 shows the cGAN’s ability to model a high-speed receiver with a sample from the test set. Fig. 3a shows the receiver’s output as a GASF, and Fig. 3b shows the generated GASF. Though we are not explicitly concerned with recovering the GASF, the cGAN shows that it can do so with high accuracy. Moving to the underlying waveforms embedded in the GASF, Fig. 3c shows that both waveforms are identical with minimal errors.

However, Fig. 3 shows a single case from the dataset. For this case, the RMSE is 0.016 in a $[0, 1]$ range, whereas the RMSE across the entire test set is 0.021 in the same range.

B. Holdout Channels - Fixed Receiver

To demonstrate the model’s ability to extend to unseen channels, we retrain it with a training set that excludes 20% or 200 channel conditions without additional hyperparameter tuning. Once the model completes training, we evaluate it on the unseen channels.

Fig. 4 shows the model’s prediction for the receiver’s output for a channel from the test set with multiple time windows passed in simultaneously. A similar batched prediction with smaller transmitted bitstream chunks can generate the complete eye diagram. Like the baseline in Fig. 3c, the model fits the ground truth well despite not seeing the channel during training. Across the test set of unseen channels, the model has

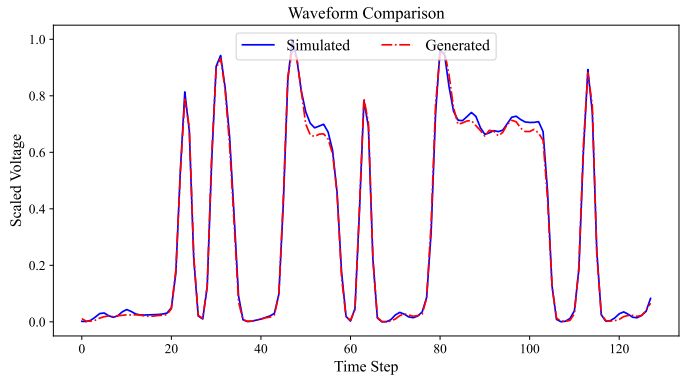


Fig. 5: Predictions on a single window for an unseen combination of PRBS, channel, and equalization setting for a tuneable receiver. The waveforms correlate with errors at the peaks.

an RMSE of 0.022 in a $[0, 1]$ range, and the example shown has an RMSE of 0.022 in the same range. For this longer time sequence, it takes approximately 232 ms to generate the receiver output for a given transmitter waveform, whereas spice takes over a minute to run for the same channel-bitstream combination.

C. Tuneable Receiver

For the tuneable receiver, as we chunk the time series at a 128-time-step resolution, the cGAN’s input and output resolutions are 128×128 instead of 256×256 , as is the case for the fixed receiver.

Fig. 5 shows a simulated and the corresponding generated waveform using the proposed approach. The waveforms are identical across all time steps; however, there is a visible difference in the peaks of the recovered waveform. This error is minimal; for the sample presented here, the prediction has an RMSE of 0.036 in the $[0, 1]$ range. We see a similar error across the test set, with the maximum RMSE in the test being 0.14, whereas the average RMSE across the test set is 0.054.

D. Holdout Taps- Tuneable Receiver

Here, we highlight the model’s ability to extend to unseen equalization settings for a channel. We retrain the model on a new training set that withholds a different equalization setting for each of the 8 channels. Once the model finishes training, we evaluate it using unseen equalization settings.

Fig. 6 shows the model’s result on an unseen equalization setting compared to the simulation for multiple consecutive prediction windows. Like the regular tuneable receiver case, the cGAN translates well from the input to the output domain. Furthermore, like the base case, most errors occur at the peaks of the waveforms. However, the error tends to be small; for the multiple cascaded predictions shown in Fig. 6, the RMSE between the generated and simulated waveforms is 0.031 in the $[0, 1]$ range. We see a similar error across the test set, with the average RMSE across the test set being 0.061, slightly higher than the regular tuneable case from before.

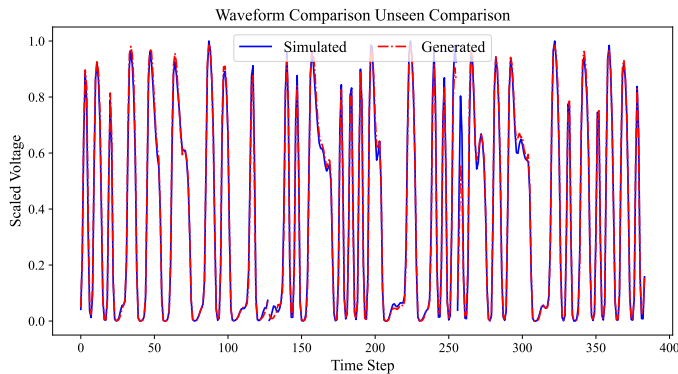


Fig. 6: Prediction across multiple windows for unseen equalization settings for a tuneable receiver with errors at the peaks of the waveform when compared to the simulation.

We hypothesize that the errors for the tuneable receiver cases are higher than those for the fixed receiver due to a mixture of operating speeds and sampling rates. The tuneable receiver runs at 16 Gb/s, or a UI of 62.5 ps, implying that after downsampling, it has approximately 3 samples per UI. On the other hand, the fixed receiver runs at 5 Gb/s, or a UI of 200 ps, or 4 samples per UI. The sample difference and the faster edge rates for the tuneable receiver lead to slightly higher errors.

VII. CONCLUSION

This paper presents a data-driven approach to modeling the transient of a high-speed receiver. It leverages GAF, which maintains the temporal relationships between time steps, as a reversible intermediate representation of the times series. Then, a cGAN uses the incoming receiver waveform to generate a GAF of the receiver's output post-equalization. The model's training is modified to extract the receiver waveform with high accuracy. The paper shows the model's capacity to work on unseen channels and receiver equalization configurations with low error.

This work opens the door to modeling generic analog blocks, which take significant computational resources when running traditional spice simulations. Another avenue of exploration is to leverage the cGAN's ability to inpaint to complete a GAF, thereby autoregressively predicting the time series.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. CNS 2137283 - Center for Advanced Electronics through Machine Learning (CAEML) and its industry members. T. Wu was partly supported by NSF IIS-1909644.

REFERENCES

[1] M. Kashyap, K. Keshavan, and A. Varma, "A novel use of deep learning to optimize solution space exploration for signal integrity analysis," in *IEEE 26th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, 2017, pp. 1–3.

[2] R. Trincherio and F. G. Canavero, "Modeling of eye diagram height in high-speed links via support vector machine," in *IEEE 22nd Workshop on Signal and Power Integrity (SPI)*, 2018, pp. 1–4.

[3] Y. Choi and C. Cheng, "High-speed link analysis with system identification approach," in *IEEE International Symposium on Electromagnetic Compatibility Signal/Power Integrity (EMCSI)*, 2017, pp. 741–744.

[4] B. Li, B. Jiao, M. Huang, et al., "Improved system identification modeling for high-speed receiver," in *IEEE 28th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, 2019, pp. 1–3.

[5] T. Nguyen and J. Schutt-Aine, "A tunable neural network based decision feed-back equalizer model for high-speed link simulation," in *IEEE 29th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, 2020, pp. 1–3.

[6] C. H. Goay, N. S. Ahmad, and P. Goh, "Transient simulations of high-speed channels using CNN-LSTM with an adaptive successive halving algorithm for automated hyperparameter optimizations," *IEEE Access*, vol. 9, pp. 127 644–127 663, 2021.

[7] P. Kashyap, A. Gajjar, Y. Choi, et al., "RxGAN: Modeling high-speed receiver through generative adversarial networks," in *Proceedings of the 2022 ACM/IEEE Workshop on Machine Learning for CAD*, pp. 167–172.

[8] J. Lee, S. Choi, K. Son, et al., "Adaptive gramian-angular-field segmentation integration based generative adversarial network (AGSI-GAN) for eye diagram estimation of high bandwidth memory (HBM) interposer," in *IEEE 32nd Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, 2023, pp. 1–3.

[9] P. Kashyap, P. P. Ravichandiranand, D. Baron, et al., "Generative adversarial network based adaptive transmitter modeling," in *IEEE 73rd Electronic Components and Technology Conference (ECTC)*, 2023, pp. 2183–2187.

[10] B. Hettwer, T. Horn, S. Gehrler, et al., "Encoding power traces as images for efficient side-channel analysis," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, IEEE, 2020, pp. 46–56.

[11] H. Xu, J. Li, H. Yuan, et al., "Human activity recognition based on gramian angular field and deep convolutional neural network," *IEEE Access*, vol. 8, pp. 199 393–199 405, 2020.

[12] Z. Wang and T. Oates, "Imaging time-series to improve classification and imputation," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015, pp. 3939–3945.

[13] P. Isola, J. Zhu, T. Zhou, et al., "Image-to-image translation with conditional adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5967–5976.

[14] T. Nguyen, T. Lu, K. Wu, et al., "Fast transient simulation of high-speed channels using recurrent neural network," *arXiv preprint arXiv:1902.02627*, 2019.

[15] W. Ye, M. B. Alawieh, Y. Lin, and D. Z. Pan, "LithoGAN: End-to-end lithography modeling with generative adversarial networks," in *56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[16] Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim, "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–8.

[17] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI: 18th International Conference*, Springer, 2015, pp. 234–241.

[18] E. Schonfeld, B. Schiele, and A. Khoreva, "A U-Net based discriminator for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8207–8216.

[19] J. Faouzi and H. Janati, "Pyts: A python package for time series classification," *Journal of Machine Learning Research*, vol. 21, no. 46, pp. 1–6, 2020. [Online]. Available: <http://jmlr.org/papers/v21/19-763.html>.

[20] M. Abadi, A. Agarwal, P. Barham, et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: <https://www.tensorflow.org/>.