A Clock Tree Prediction and Optimization Framework Using Generative Adversarial Learning

Yi-Chen Lu, *Student Member, IEEE*, Jeehyun Lee, *Graduate Student Member, IEEE*, Anthony Agnesina, *Student Member, IEEE*, Kambiz Samadi, *Senior Member, IEEE*, and Sung Kyu Lim, *Senior Member, IEEE*

***Abstract*—Modern physical design flows highly depend on design space exploration to find the commercial tools' clock tree synthesis (CTS) parameters that lead to optimized clock trees. However, such exploration is often time-consuming and computationally inefficient. In this article, we overcome this drawback by proposing a novel framework named GAN-CTS, which utilizes conditional generative adversarial network (GAN) to predict and optimize CTS outcomes. Our framework is built upon three sequential learning stages. First, to precisely characterize distinct designs, we leverage transfer learning to extract netlist features directly from placement images. Second, we perform regression learning using various methods to predict the target CTS outcomes and demonstrate that the proposed multitask learning approach achieves better accuracy than the meta-modeling method adopted by previous works. To fully benefit from the predictions made by our framework, we further quantitatively interpret the importance of each CTS input parameter subject to various design objectives through attribution-based learning. Finally, generative adversarial learning is leveraged to optimize the target clock metrics with the guidance provided by the pretrained regression model. To substantiate the generality of our framework, we perform validations on four unseen netlists that are not utilized in the training process. The experimental results conducted on real-world designs demonstrate that our framework: 1) achieves an average prediction error of 3%; 2) improves the commercial tool's auto-generated clock tree by 20.7% in clock power, 21.5% in clock wirelength, 36.1% in the worst skew; and 3) reaches an F1-score of 0.93 in the classification task of determining successful and failed CTS runs.**

***Index Terms*—Clock tree synthesis (CTS), generative adversarial learning, physical design.**

Manuscript received 28 September 2020; revised 31 December 2020, 11 April 2021, and 15 August 2021; accepted 13 October 2021. Date of publication 21 October 2021; date of current version 22 August 2022. This work was supported in part by the National Science Foundation under Grant CNS 16-24731, and in part by the Industry Members of the Center for Advanced Electronics in Machine Learning. This article was recommended by Associate Editor L. Behjat. *(Corresponding author: Yi-Chen Lu.)*

Yi-Chen Lu and Jeehyun Lee are with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: yclu@gatech.edu).

Anthony Agnesina is with the Georgia Institute of Technology, Atlanta, GA 30332 USA.

Kambiz Samadi is with Qualcomm Technologies, Inc., San Diego, CA 92121 USA.

Sung Kyu Lim is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA.

Digital Object Identifier 10.1109/TCAD.2021.3122109

I. Introduction

CLOCK tree synthesis (CTS) is a critical stage of physical design, since clock networks constitute a high percentage of the total power in the final full-chip design. An optimized clock tree helps to avoid serious design issues, such as excessive power consumption, high routing congestion, and elongated timing closure [7]. However, due to the high complexity and run time of the modern electronic design automation (EDA) tools, designers are struggling to synthesize high-quality clock trees that optimize key desired metrics, such as clock power, skew, clock wirelength, etc. To find the input parameters that achieve the design targets, designers have to search in a wide range of candidate parameters, which is usually fulfilled in a manual and highly time-consuming calibration fashion.

To automate this task and alleviate the burden for designers, several machine learning (ML) techniques have been proposed to predict the clock network metrics [19]. Previous work [14] utilizes data mining tools to estimate the achieved skew and insertion delay. Kahng *et al.* [12] employed statistical learning and meta-modeling methods to predict more essential metrics, such as clock power and clock wirelength. Kahng *et al.* [13] further considered the effect of nonuniform sinks and different placement aspect ratios. Another work [18] utilizes artificial neural networks (ANNs) to predict the transient clock power consumption based on the estimation of clock tree components. However, these previous works merely focus on enhancing the prediction of CTS metrics rather than the optimization of the outcomes. Therefore, their methods are not sufficient to achieve high-quality clock trees without the aid of other heuristic algorithms. To optimize CTS metrics, a recent work [16] develops an ML-powered clock tree construction algorithm which generates optimized clock trees based on a pretrained CTS buffer prediction framework that estimates buffer usage. The proposed algorithm is proven to be generalizable on unseen designs.

The goal of this work is to construct a general and practical CTS modeling framework, which has the ability to predict CTS outcomes in high precision and perform CTS optimization by generating the CTS input parameter sets that lead to optimized clock trees for *general* designs in an unsupervised manner. Specifically, we take an industry-leading commercial tool, *Cadence Innovus*, as reference and demonstrates the feasibility of the proposed framework upon it. The proposed CTS prediction and optimization framework named GAN-CTS achieves the following aspects.

1) *Generalizability:* We aim to develop a generalizable framework that achieves high-quality clock trees on general designs. We achieve this by leveraging generative adversarial learning which has the capability to inference optimized CTS input parameter sets on unseen designs.
2) *Interpretability:* Instead of considering our framework as a black box, we leverage a gradient-based attribution method [25] to interpret the prediction made by the proposed framework.
3) *Optimality:* Despite that the optimality of a clock tree is hard to demonstrate, in this work, we compare the optimization results achieved by the proposed framework to the ones achieved by the default settings of *Cadence Innovus*, and show that the proposed framework reaches never-seen, high-quality CTS optimization results.

The remainder of this article is organized as follows. Section II illustrates the background of ML techniques utilized in this work. Section III presents a solid overview of our framework. Section IV describes the problem formulations, database generation, and identifies the difficulties of the problems. Section V illustrates the algorithms and the detailed structures of the models. The experimental results are demonstrated in Section VI, and finally we conclude our work in Section VIII.

II. ML Background

VLSI netlists are comprised of substantial and complicated design characteristics that are hard to be enumerated manually. To distill the essence of distinct designs, we leverage ResNet-50 [10] to perform transfer learning by directly extracting design features from *placement layout images* instead of manually defining the proper features to differentiate different designs as previous works [12], [13], [18]. The extracted features from transfer learning not only ensures the generality of our framework, but also improves the prediction accuracy of the models.

Our proposed framework GAN-CTS consists of a regression model and a variation of generative adversarial network (GAN) [9] named conditional GAN [22]. The regression model is accounted for the predictions of CTS outcomes. In this work, we analyze the impact of using different regression methods, including the meta-modeling technique adopted by previous work [13] and the proposed multitask learning technique. Due to the high correlation of different CTS outcomes (e.g., clock wirelength and clock power), we substantiate that the proposed multitask learning technique achieves higher accuracy in a much shorter training time.

The conditional GAN is comprised of two modules: 1) the generator and 2) the discriminator. The standard goal of the generator is to generate the CTS input parameter sets that have a high resemblance in terms of parameter distributions to the ones in the database, where the traditional goal of the discriminator is to differentiate between the generated samples from the generator and the real samples from the database. The adversarial learning that the discriminate provides helps the generator to produce *realistic* parameters. Note that the similarity between the generated samples and the real samples can be quantitatively interpreted by Kullback–Leibler divergence [17].

In this work, apart from the GAN training process mentioned above, we advance the learning process by introducing new objectives to the generator and the discriminator to solve the CTS parameters optimization problem. During the learning process, we apply the pretrained regression model to evaluate the quality of the generated CTS input parameter sets from the generator. The generator will thus strive to generate the parameter sets that achieve good quality predictions. Therefore, aside from the conventional goal to generate the CTS input parameter sets that are *realistic* (similar to the database), another objective of the generator is to maximize the clock tree quality evaluated by generating the pretrained regression model.

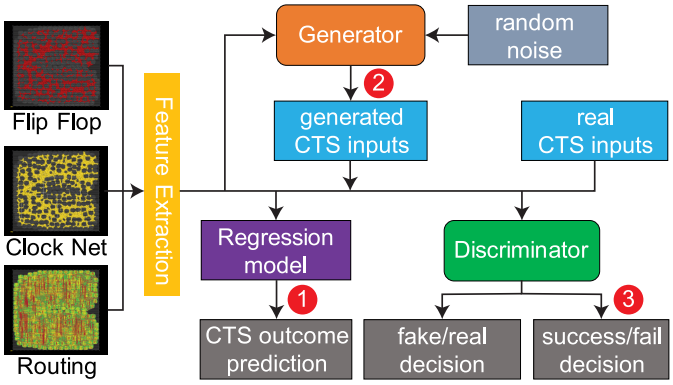

Fig. 1. High-level view of this work and the three objectives we have achieved. The first objective is to predict the CTS outcomes in high precision. The second objective is to recommend designers good CTS input settings. The third objective is to determine whether the input settings outperform commercial tool's auto-setting.

As for the discriminator, instead of only determining whether its input is from the generator or from the database, we introduce another objective to classify successful and failed CTS runs. In this article, a successful CTS run indicates that the achieved clock tree outperforms the one auto-generated by the commercial tool (*Cadence Innovus* is utilized in this work). Hence, the discriminator not only predicts whether the CTS input parameter sets are generated by the generator or coming from the actual database as the conventional approach, but also predicts if the given parameter sets can result in successful CTS runs or not. With this additional objective of the discriminator, designers are able to tune the parameters in a good guidance.

III. Overview of GAN-CTS

It has been widely acknowledged that GAN is a promising model that learns the complicated real-world data through a generative approach [31]. A vanilla GAN structure contains a generator and a discriminator which are both neural networks (NNs). The goal of the generator is to generate meaningful data from a given distribution such as random noise, while the objective of the discriminator is to distinguish the *generated* samples from the *real* samples that are targeted to be mimicked. Predicated on the vanilla GAN structure, in conditional GAN, an external conditional input is further introduced to both generator and discriminator. This conditional input enables the model to direct the data generation process based on different conditions, which benefits us to generate suitable CTS input parameters sets with respect to different benchmarks and even the ones that are not utilized in the training process.

A high-level view of our framework named GAN-CTS is shown in Fig. 1. The framework is comprised of three sequential training (learning) stages. The first training stage is to

extract key design features from placement images which represent flip flop distributions, clock net distributions, and trial routing results. Note that trial routing is a process performed in the end of the placement stage in the modern physical design flow. It provides a quick estimation of the routing congestion based on the given placement. To extract features from images, we adopt transfer learning by using a pretrained convolutional NN (CNN) named ResNet-50 [10], which is a 50-layer residual network that has skip connections. The goal of transfer learning is to leverage the trained convolutional filters to distill the hidden information in the placement layouts.

In the second training stage, we leverage the extracted features from the previous feature extraction stage as well as the clock tree instances in the database to train the regression model for CTS outcomes predictions. In this work, we select three essential CTS metrics that well represent the quality of a clock tree as the prediction and optimization targets. These three selected metrics are clock power, clock wirelength, and the achieved maximum skew. We compare two different regression approaches which are the meta-modeling technique adopted by previous work [13] and the proposed multitask learning technique. We demonstrate that the proposed technique reaches better prediction accuracy with a much shorter training time. Furthermore, as mentioned in Section I, we do not consider our model as a black box as previous works. To interpret the predictions made by the regression model, we leverage a gradient-based attribution method [1] to quantitatively determine the importance of each CTS input parameter subject to different target outcomes.

The last training stage of our framework involves generative adversarial learning, where we leverage a conditional GAN to perform the CTS optimization and classification tasks. The regression model trained earlier now acts as a supervisor to guide the generator to generate the CTS input parameters sets that lead to optimized clock trees. Note that the extracted placement features from transfer learning are taken as the conditional input, where the original inputs of the vanilla GAN model are termed as the regular inputs in this work. The advantage of having the conditional input is to control the *modes* of the generated data, where we consider different benchmarks as different *modes*. Therefore, with the conditional approach, our framework has the ability to optimize *unseen* benchmarks that are not utilized during the training process.

Finally, a highlight of our framework is that a multitask learning is conducted by the discriminator. In addition to the conventional task of distinguishing between the generated and the real samples, we introduce a new task of classifying successful and failed CTS runs. In this article, we strictly define a CTS run as successful if two out of the three achieved target CTS metrics mentioned earlier are better than the ones achieved automatically by the commercial CTS engine.

In the end of the training process, we acquire four models as follows.

1) A placement feature extractor E which precisely characterizes different designs from placement images.
2) A regression model R which performs high precision predictions of target CTS outcomes.
3) A generator G which generates CTS input parameters sets that lead to optimized clock trees.
4) A discriminator D which predicts the success and failure of CTS runs.

TABLE I
OUR BENCHMARKS AND THEIR ATTRIBUTES IN *TSMC* 28 NM

Design Name	# Nets	# FFs	# Cells	Usage
AES-128	90,905	10,688	113,168	training
B19	34,399	3,420	33,784	
DES_PERF	48,523	8,802	48,289	
LDPC	42,018	2,048	39,377	
NETCARD	317,974	87,317	316,137	
NOVA	138,171	29,122	136,537	
TATE	185,379	31,409	184,601	
ECG	85,058	14,018	84,127	testing
JPEG	231,934	37,642	219,064	
LEON3MP	341,263	108,724	341,000	
VGA_LCD	56,279	17,054	56,194	

TABLE II
MODELING PARAMETERS WE USE AND THEIR VALUES

type	parameters	values or ranges
placement	aspect ratio	$\{0.5, 0.75, \cdots, 1.5\}$
	utilization	$\{0.4, 0.45, \cdots, 0.7\}$
CTS	max skew (ns)	[0.01, 0.2]
	max fanout	$[50, 250]$
	max cap trunk (pF)	[0.05, 0.3]
	max cap leaf (pF)	[0.05, 0.3]
	max slew trunk (ns)	[0.03, 0.3]
	max slew leaf (ns)	[0.03, 0.3]
	max latency (ns)	[0, 1]
	max earlyRouting layer	{2, 3, 4, 5, 6}
	min earlyRouting layer	{1, 2, 3, 4, 5}
	max buffer density	[0.3, 0.8]

IV. DESIGNING EXPERIMENTS

A. Database Analysis

We formally define the CTS prediction and optimization problems as follows.

Problem 1 (CTS Outcomes Prediction): Given a pre-CTS placement P and a CTS input parameters set X, predict the post-CTS outcomes Y without performing any actual CTS process.

Problem 2 (CTS Outcomes Optimization): Given a pre-CTS placement P, generate a CTS input parameters set $\hat{X}$ that leads to optimized CTS outcomes $\hat{Y}$.

In this work, we tackle Problems 1 and 2 through ML approaches. Before elaborating the modeling process, we first describe and analyze our database.

B. Database Construction

To build the database, we utilize *Synopsys Design Compiler 2015* to synthesize the netlists and leverage *Cadence Innovus Implementation System v18.1* to perform the placement and CTS processes. The database is constructed based on the TSMC-28 nm technology node. In this work, we leverage five designs which are B19, LEON3MP, NETCARD, DES_PERF, and VGA_LCD from the ISPD 2012 benchmark [23], and six other designs, including AES-128, LDPC, NOVA, ECG, TATE, and JPEG, from *Opencores.org* to conduct the experiments. Table I shows the attributes of all 11 designs after being synthesized at 1125 MHz.

Table II presents the modeling parameters and their ranges of values that we utilize. The ranges of the CTS-related parameters are determined by reasonably widening the commercial tool's auto-setting values based on domain expertise. The goal

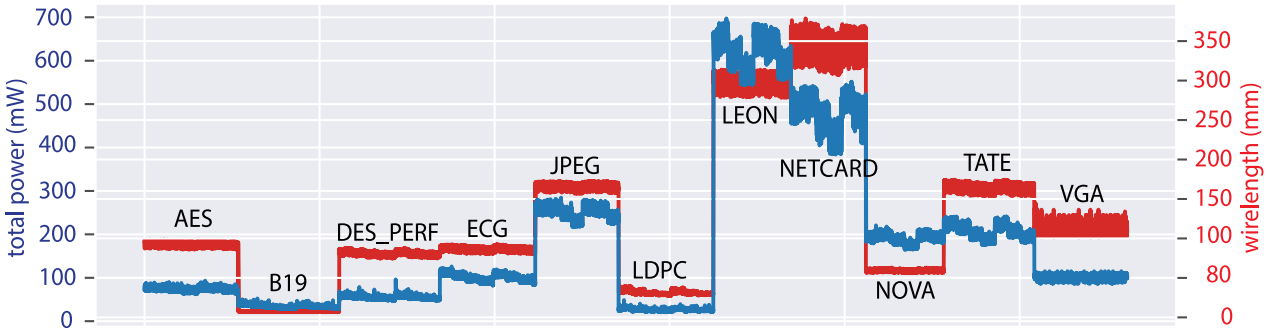

Fig. 2. Total power (mW) and wirelength (mm) distributions among 115.5k full-chip designs of all 11 benchmarks in our database.

TABLE III
CLOCK TREES WITH DIFFERENT INPUT SLEW CONSTRAINTS FOR NOVA BENCHMARK

	tree A	**tree B**	**tree C**
max trunk slew (ns)	0.1	0.1	0.05
max leaf slew (ns)	0.1	0.05	0.05
# inserted buffers	2,556	600	1,124
total power (mW)	164.7	154.5	158.8
clock power (mW)	27.9	22.6	24.9
clock wirelength (mm)	118.7	97.3	101.6
maximum skew (ns)	0.06	0.1	0.07

is to generate a database with a high variety in terms of clock metrics so that the proposed framework can better differentiate good designs from the bad ones, and comprehend which combinations of the parameters can lead to optimized clock trees. Among the modeling features, aspect ratio and utilization rate represent physical structures. The min and max early routing layers (min $\leq$ max) indicate the metal layers utilized in the early global routing (EGR) stage, which is a procedure to reserve space for future detailed signal routing during the CTS stage (EGR is contained in CTS). Note that although "skew" is taken as an input target as shown in the table, commercial tools will not necessarily meet the skew target during CTS (skew is often further improved in future design steps), which makes the skew prediction (one of the CTS outcomes we focus in this work) problem nontrivial.

The combinations of the two placement-related parameters give us 35 different placements per netlist. By running CTS with randomly sampled input parameters, we generate 300 clock trees per placement. Therefore, in total, we have 115.5k datum (clock tree instances) across 11 different netlists in our database. To substantiate the generality of our framework, in the experiments, we only utilize seven netlists during the training process and perform the validations on the remained four unseen netlists as indicated in Table I.

Fig. 2 shows the total power and wirelength distributions of all 115.5k clock trees in the database. It demonstrates the variety of our database as well as the difficulty to model the CTS process across different benchmarks. Table III demonstrates the complicated impact of different input slew constraints on essential CTS metrics. We observe that if a single tighter slew constraint is merely given on leaf cells (from design A to design B), the total number of inserted buffers drastically decreases. However, if tighter slew constraints are given on both trunk cells and leaf cells (from design A to design C), more buffers are inserted compared with the previous approach. In summary, the above analyses show that the behavior of the commercial CTS engine is very sophisticated and counterintuitive, which is mainly due to the complicated high-dimensional intercorrelation among different CTS input parameters [12]. In this article, we aim to employ ML methods to demystify the complicated CTS process.

In this article, we consider the clock tree auto-generated by the commercial tool as a baseline to evaluate the trees generated by our framework. As mentioned in Section III, we define a CTS run as successful if two out of the three achieved target metrics, which are clock power, clock wirelength, and clock skew, are better than the ones of the auto-generated clock tree. In Section VI, we demonstrate the CTS metrics and layout comparisons between the optimized clock trees generated by our framework and the one auto-generated by the commercial tool.

V. GAN-CTS ALGORITHMS

In this section, we first describe the process of feature extraction. Then, we present our methodologies to solve the CTS outcomes prediction and optimization problems (Problems 1 and 2). In the meantime, we illustrate the detailed structures of the models. In the end, we summarize the overall training process in a complete algorithm.

A. Placement Image Feature Extraction

One of the innovations of this work is that we directly utilize placement images as inputs to predict and optimize the target CTS outcomes. The key rationale is that placement images contain important information of designs. Previous work [33] has demonstrated the efficiency of using placement images to predict routability and design rule violations (DRVs). In this work, we leverage the extracted features from placement images to solve CTS outcomes prediction and optimization problems. Our approach is built upon CNN and transfer learning. As shown in Fig. 3 we devise our own fully connected (FC) layers upon the pretrained model named ResNet-50 [10], which is a CNN-based model pretrained on the well-known ImageNet [8] dataset with millions of images.

Fig. 4 shows the visualization of a trial routing image passing through 12 selected convolutional filters inside the pretrained ResNet-50 model. In the figure, we observe that important information such as usage of metal layers is well captured. Although ResNet-50 is powerful on many image datasets, it is not devised specifically for the physical design problems. Therefore, to extract key design features from the high dimensional vectors, we devise a feature extraction flow with transfer learning as shown in Fig. 3, where four self-designed FC layers are appended on top of the ResNet-50 model to predict the commercial tool's estimation of total

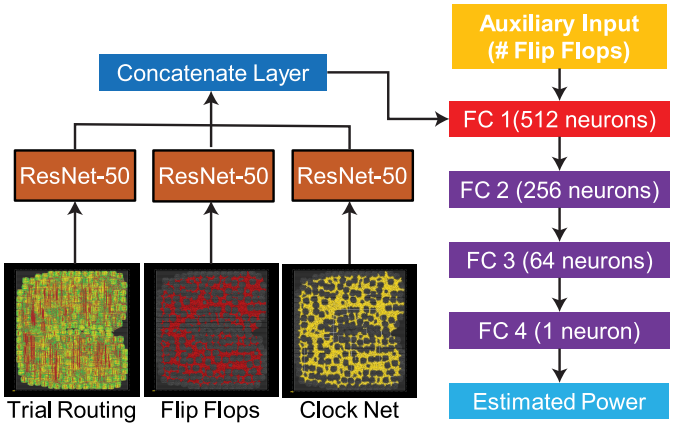

Fig. 3. Our image feature extraction flow. The extracted features are colored in red. Note that each raw image vector extracted from ResNet-50 has 1024 dimensions. The concatenate layer thereby forms a vector in 3072 dimensions. Finally, with an auxiliary input that denotes the number of flip flops, the input of the self-devised FC layers are in 3073 dimensions.

power right after the placement stage. As shown in the figure, since placement images cannot precisely reflect the actual size of a design, we introduce an auxiliary input with one dimension which represents the total number of flip flops to the first FC layer by concatenating it with the direct extracted features of the ResNet-50, In the training process, we fix the parameters of the pretrained ResNet-50 model and only update the parameters of the self-devised FC layers. When the training is completed, each pre-CTS placement is encoded into a vector with 512 dimensions, which is the output of the first FC layer (denoted in red in the figure).

Our transfer learning approach accepts any image sizes, since we only utilize the pretrained convolutional filters rather than the FC layers of the original ResNet-50 model. In the implementation, all the images in the database are in a dimension of $700 \times 717 \times 3$. The achieved mean absolute percentage error (MAPE) of the unseen validation netlists is less than 0.7%, where the maximum absolute percentage error is 5% across training designs. However, since a low prediction error does not guarantee a good feature representation, we leverage a dimension reduction technique named t-distributed stochastic neighbor embedding (t-SNE) [21] to visualize the extracted features ($\in R^{512}$) in R^2 as shown in Fig. 5. In the figure, we observe that different placements belonging to the same netlist are clustered together and those belonging to different netlists are well separated. Therefore, we conclude that the extracted features well capture the design characteristics.

Finally, to justify the achievement of transfer learning, in Section VI, we perform an experiment of comparing the CTS prediction results between with and without using transfer learning from the ResNet-50 model. Since the goal of transfer learning is to precisely characterize different designs, in the setting without using transfer learning, we handcrafted four features to represent the extracted features in Fig. 3. These four features include the number of cells, number of flip flops, number of nets, and number of ports in the design.

B. CTS Outcomes Prediction

Constructing a precise regression model is the key step to solve Problem 1. Following the feature extraction process, we train the regression model with the extracted features to predict the target CTS outcomes. As mentioned in Section III, in this article, we target at predicting and optimizing three CTS outcomes which are clock power, clock wirelength, and the maximum skew. In this work, we analyze two different strategies to construct the regression model.

Multimodel Uni-Output: The first strategy is built upon meta-modeling, which is the strategy adopted by the state-of-the-art academic works [12] and [13]. The key concept of meta-modeling is that for each target CTS outcome, a single meta-model is constructed by combining multiple base models through a high-level aggregation function. This ensembled meta-model is expected to make more stable and accurate predictions than any individual base model. In [12] and [13], traditional regression techniques, such as radial basis functions (RBFs) [3] and support vector machine (SVM) are utilized as the base models, where the weighted least square regression [6] is leveraged as the aggregation function in both works. However, these regression techniques utilized in the base models of previous works are known to be easily biased to the dataset [30], which thereby cannot be generalized to unseen designs.

To overcome the drawback of previous works, we leverage *xgboost* [4], *catboost* [24], NN as the base models of our framework. These base models are combined through the weighted least square regression [6] as previous works to construct the meta-model. Note that each target CTS outcome requires a meta-model for the prediction, therefore, we construct three meta-models to predict the three target CTS metrics as mentioned earlier. As shown in Fig. 6, each base model takes the features extracted from the feature extraction process and the CTS parameters as inputs, and outputs the prediction of the specific clock metric. The core idea of meta-modeling is to reduce the variance of each base model, and therefore eliminate the impact of the bias in the database. However, a foreseeable issue of using the meta-modeling technique is that it requires a long training time due to the large number of the training parameters.

Uni-Model Multioutput: The second strategy we adopt to construct the regression model is through multitask (multiobjective) learning, where we build a multioutput deep NN to predict the three target CTS metrics simultaneously. The visualization of the model is shown in Fig. 7. The model takes the extracted features along with the CTS parameters as inputs and outputs three predictions simultaneously. As shown in the figure, the inputs are passed through shared layers and dedicated layers to predict the clock metrics. The key rationale of multitask learning is that different CTS outcomes are not independent of each other (e.g., clock wirelength often has a high correlation with clock power). Therefore, instead of isolating the parameters for each prediction as in the meta-learning approach, we leverage shared layers to enable different objectives to own mutual information, which helps to reduce the training time as well as enhances the accuracy of the predictions. In the training process, we utilize mean squared error as the loss function, and leverage dropout layers inside the model to prevent it from overfitting. The validation results of this experiment are shown in Section VI, where we observe that the multitask learning approach achieves higher accuracy in a shorter training time comparing to the meta-learning approach.

C. Interpreting Prediction Results

In this work, we do not treat the regression model as a black box as previous works. Instead, we leverage a gradient-based attribution algorithm named DeepLIFT [25] to interpret the

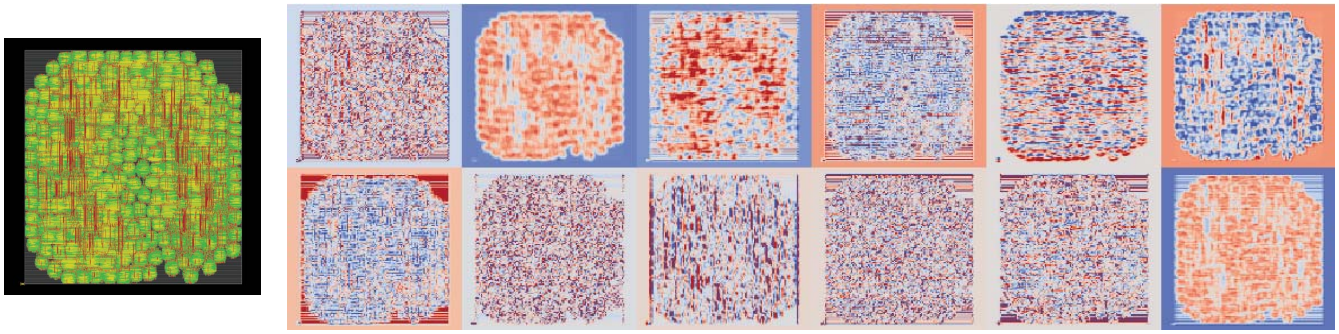

Fig. 4. Visualization of trial routing image in 12 different convolutional filters of ResNet-50 [10], where the usage of metal layers is well captured across different filters.

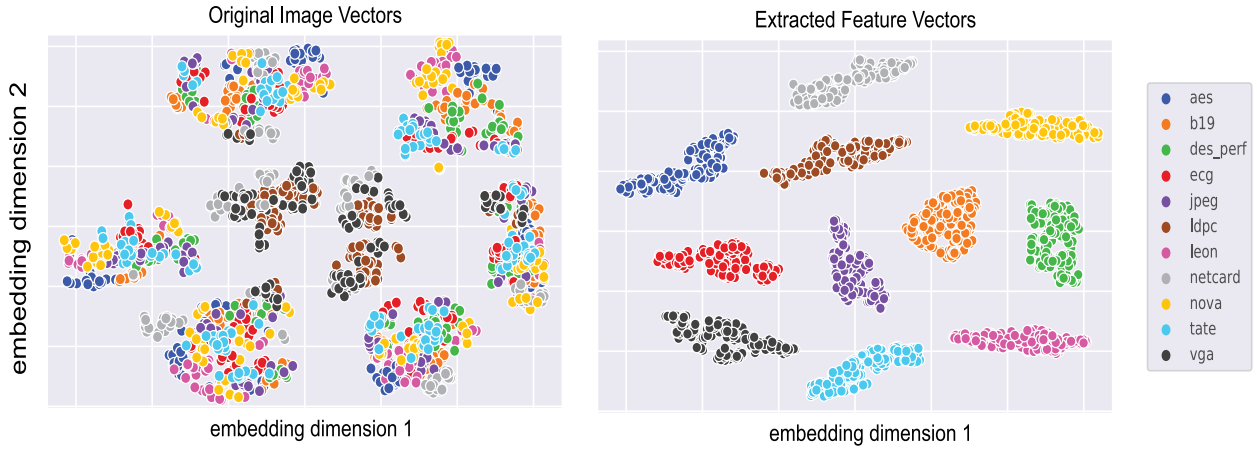

Fig. 5. t-SNE visualizations of the original placement image vectors and the extracted feature vectors from our transfer learning flow, where the extracted features successfully characterize different designs.

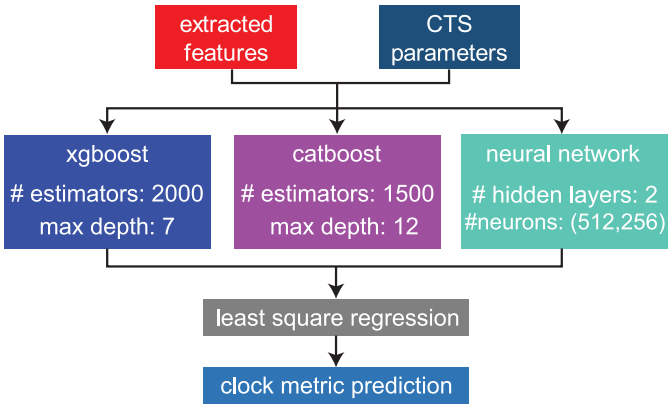

Fig. 6. Visualization of single metal-model, which is stacked by three base models through weighted least square regression.

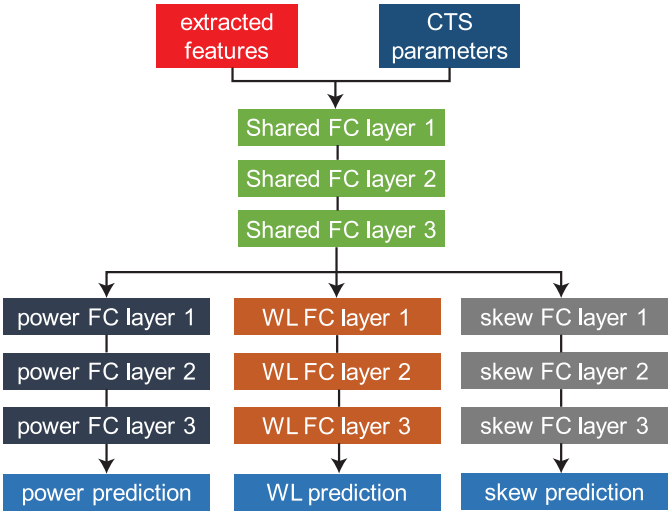

Fig. 7. Our proposed multiobjective regression model. The detailed architectures are as follows. The shared FC layers colored in green have number of neurons equal to 512, 256, and 128 in sequential, where each dedicated FC layer group has number of neurons equal to 64, 32, and 1 from input to output.

predictions. The algorithm aims to determine the attribution (relevance) value of each input neuron subject to different outputs. Assume our regression model R takes an input vector $x = [x_1, \ldots, x_N] \in R^N$ and produces an output vector $S = [S_1, \ldots, S_k]$. DeepLIFT proceeds a_i^l, the attribution of neuron i at layer l, in a backward fashion by calculating the activation difference of the neuron between the target input x and reference input $\hat{x}$. The procedure is derived as

$$a_i^L = \begin{cases} S_i(x) - S_i(\hat{x}), & \text{if neuron } i \text{ is the output of interest} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$$a_i^l = \sum_j \frac{z_{ji} - \hat{z}_{ji}}{\sum_i z_{ji} - \sum_i \hat{z}_{ji}} \cdot a_j^{l+1} \quad (2)$$

where L denotes the output layer and z_{ji} is the weighted activation of neuron i onto neuron j in the next layer. In the

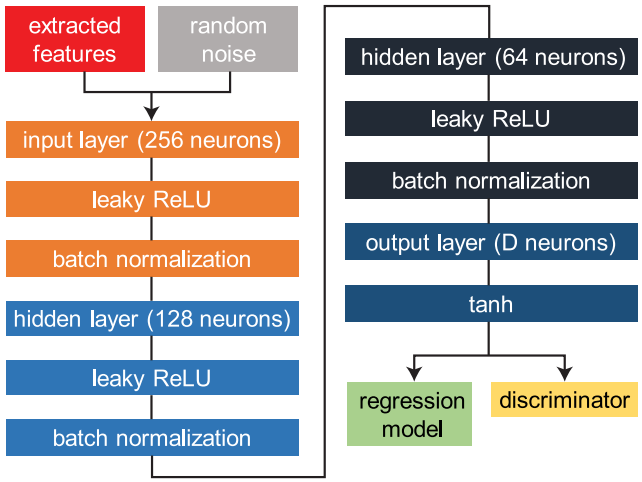

Fig. 8. Detailed structure of our GAN generator.

implementation, we take the reference input $\hat{x}$ as the input parameters of the auto-generated clock tree.

D. CTS Optimization

In this article, we develop two CTS optimization techniques based on the presented regression model. Specifically, the objective of the optimization problem (Problem 2) is to find the CTS input parameter sets of the commercial tool that lead to optimized clock trees. Note that in the experiments, we select the multitask learning regression model (uni-model multiout-put) as the base model (guidance provider) for optimization, since it achieves the best prediction accuracy compared with other models.

1) GAN-Based Optimization: The first optimization approach we develop leverages generative adversarial learning to perform the optimization, where we train a generative model (the generator) that learns to generate the parameter sets which lead to optimized CTS outcomes. As aforementioned, prior to the GAN learning, we pretrain the regression model as the guidance provider. The generator is expected to generate optimized CTS input parameter sets by maximizing the CTS quality predicted by the pretrained regression model. Before illustrating the objectives of the optimization process, we first describe the model structure of the generator.

Fig. 8 shows the detailed structure of the generator. The generator G is an NN parameterized by θ_g which samples a regular input z with 100 dimensions from a $N(0, 1)$ Gaussian distribution p_z, and samples the extracted placement features f from the database p_d as the conditional input. The leaky ReLU [34] layers are employed as activation functions of the input and hidden layers to project latent variables onto a wider domain, which eliminates the bearing of vanishing gradients. Batch normalization [11] layers are utilized to normalize the inputs of each hidden layers to zero mean and unit variance, which accelerates the training process since the oscillation of gradient descent is reduced. Finally, for the output layer, the number of neurons D denotes the number of CTS modeling parameters, and a hyperbolic tangent layer is chosen as the activation function to match the domain of the normalized samples drawn from the database. Since when training the discriminator, we normalize the real samples x from the database to $\hat{x} \in [-1, 1]$ as

$$\hat{x} = \frac{x}{\max_{x \in \text{supp}(x)}(x)} \times 2 - 1. \tag{3}$$

Algorithm 1 Bayesian Optimization for CTS. We Leverage UCB [27] to Realize the Acquisition Function

Input: R: a pre-trained regression model, $a(\mathbf{x})$: an acquisition function.
Output: $\{\mathbf{x}\}$: optimized CTS parameter sets.
1: Initialize a prior function $f(\mathbf{x})$.
2: **repeat**
3: $\{\mathbf{x}\}_{i=1}^{k} \leftarrow$ Sample k sets of CTS parameters based on $a(\mathbf{x})$ such that $f(\mathbf{x})$ is maximized.
4: $\{r\}_{i=1}^{k} \leftarrow$ Evaluate $\{\mathbf{x}\}_{i=1}^{k}$ using R by Equation 5.
5: Update $f(x)$ with $\{\mathbf{x}\}_{i=1}^{k}$, $\{r\}_{i=1}^{k}$ using Gaussian Process.
6: **until** $\{r\}_{i=1}^{k}$ no longer improve.

In GAN-CTS, the generator has two objectives. The first is to generate realistic samples that deceive the discriminator, where the corresponding objective function is

$$\mathcal{L}_{G_D} = \mathbb{E}_{z,f}\big[\log(D(G(z,f))\big]. \tag{4}$$

The second objective is to generate the CTS input parameter sets that lead to optimized clock trees by maximizing the clock tree quality r predicted by the regression model, where r is defined as

$$r := H(G(z,f)) = -\prod_{i=1}^{N} \frac{R_i(G(z,f))}{\text{auto-setting result of target } i}. \tag{5}$$

In (5), N denotes the number of target CTS outcomes and R_i denotes the corresponding prediction of the regression model. In the implementation, we have $N = 3$ which represents the clock wirelength, clock power, and the maximum skew. The objective function of maximizing the prediction of clock tree quality can be formulated as

$$\mathcal{L}_{G_P} = \mathbb{E}_{z,f}\,[r]. \tag{6}$$

Finally, by combining the two objective functions, we formulate the training process of the generator as

$$\max_{G} \mathbb{E}_{\substack{z\sim p_z \\ f\sim p_d}}\big[\log(D(G(z,f))) + r\big]. \tag{7}$$

2) Bayesian Optimization: In addition of training a GAN-based framework that performs the optimization by learning key parameter distributions in a generative manner, in this work, we also leverage the Bayesian optimization [26] technique to solve the CTS optimization problem for comparison. Bayesian optimization is a popular surrogate optimization technique that optimizes black-box functions of arbitrary forms. Unlike NNs that require gradients to update the network parameters, Bayesian optimization models a prior (surrogate) function using the Gaussian process to characterize the target black-box function. Recently, in the realm of EDA, previous work [20] has applied such technique to perform parameter optimization of commercial tools, where it is shown that the achieved results are better than the ones achieved by the genetic algorithm [32] which is used widely for parameter optimization.

In this article, instead of the leveraging Bayesian optimization to directly optimize the commercial tool that introduces significant runtime as the previous work [20] that introduces costly runtime, we leverage it to optimize the pre-trained regression model. The optimization is summarized in Algorithm 1, which outputs the CTS input parameter sets $\{\mathbf{x}\}$ that lead to optimized clock trees. The acquisition function

$a(\mathbf{x})$ guides the sampling of the next CTS parameter sets $\{\mathbf{x}\}_{i=1}^{k}$ which are utilized to update the prior (surrogate) function $f(\mathbf{x})$. In the implementation, the sampling of the acquisition function $a(\mathbf{x})$ is performed based on upper confidence bound [27] (UCB). Note that since in our scenario, our goal is to minimize the reward r (5) to optimize clock trees, we leverage Bayesian optimization to maximize $-r$. At each step of the iteration, the Gaussian process is fit into known samples (parameter sets previously explored) to update the prior (surrogate) function $f(x)$.

3) Joint GAN-Based and Bayesian-Based Optimization: In this work, we further combine the two presented optimization techniques to optimize the CTS metrics. Since a trained GAN-based framework has the ability to suggest optimized CTS input parameter sets in constant time, for the Bayesian-based approach, instead of starting with randomly sampled observations as shown in Algorithm 1, we leverage the GAN-based model to suggest the initial CTS parameter sets. Furthermore, because the CTS input parameter sets suggested by the GAN-based model are expected to be in a more reliable (optimized) region than the ones achieved by sampling randomly, we further leverage a sequential domain reduction technique introduced in [28] to adaptively refine the search space based on the existing explored solutions. The key rationale behind is that instead of searching new parameter sets from the original wide ranges as shown in Table II, for each parameter, we can refine the sample range based on parameter sets suggested by the GAN-based framework which are already optimized. We expect this combined optimization technique can help us achieve better optimization results than any individual technique. The optimization results are shown in Section VI.

E. Success Versus Failure Classification

The classification of successful and failed CTS runs is performed in the discriminator. To describe how the classification task works, we first illustrate the structure of the discriminator as shown in Fig. 9. The discriminator takes a regular input which is either the generated samples $G(z, f; \theta_g)$ or the real samples x from the database p_d, and a conditional input which denotes the features extracted from placement images. Note that when the regular input represents the real samples x, the conditional input f should be aligned to x. The reason we introduce a conditional input to the discriminator is that it helps the discriminator to distinguish better between the generated and the real samples under different *modes* (benchmarks). As shown in (3), since each CTS input parameters has a different unit, we normalize real samples x from the database to $\hat{x} \in [-1, 1]$ to improve the stability of the GAN training process.

In GAN-CTS, the discriminator also has two objectives as the discriminator. One is to distinguish the generated samples from the real samples, which is derived as

$$\mathcal{L}_{D_g} = \mathbb{E}_{(x,f)\sim p_d}\big[\log(D_g(x,f))\big] + \mathbb{E}_{\substack{z\sim p_z \\ f\sim p_d}}\big[\log\big(1 - D_g(G(z,f))\big)\big]. \tag{8}$$

The other objective is to classify whether a given input parameter set can lead to a successful CTS run or not, where the objective is be formulated as

$$\mathcal{L}_{D_s} = \mathbb{E}_{x\sim p_d}\big[\log(D_s(x,f))\big] \tag{9}$$

which represents the cross-entropy between the classification groundtruths and the predictions made by our framework.

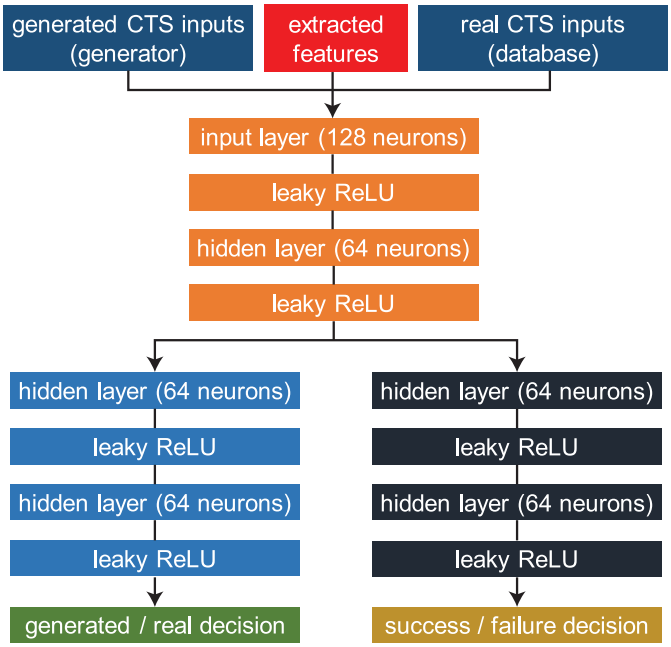

Fig. 9. Detailed structure of our GAN discriminator.

Note that the definition of successful and failed CTS runs are defined in Section III, and the discriminator is updated by (9) only when the regular input represents the real samples x. The reason we introduce a new objective to the discriminator is because its attribute is similar as the discriminator's conventional objective, where both of them are performing binary classification. Therefore, some latent features can be shared in the early network as shown in Fig. 9. Finally, the training process of the discriminator is summarized as

$$\max_{D} \ \mathbb{E}_{(x,f)\sim p_d}\big[\log\big(D_g(x,f)\big) + \log(D_s(x,f))\big] + \mathbb{E}_{\substack{z\sim p_z \\ f\sim p_d}}\big[\log\big(1 - D_g(G(z,f))\big)\big]. \tag{10}$$

F. Training Methodology

Based on the structures and the objective functions presented, we now illustrate the training process of our framework in Algorithm 2, where a gradient descent optimizer Adam [15] is utilized across different training stages. First, we train the regression model (lines 2–9) which serves as a guidance provider in the training process of the conditional GAN. Note that the regression model we adopt in the GAN-CTS framework is constructed through multitask learning. Following from the regression learning, we train the generator and discriminator alternatively (lines 10–25), since the two networks have antagonistic objectives. The parameters of the discriminator are split into θ_{d_1} and θ_{d_2} ($\theta_{d_1} \cap \theta_{d_2} \neq \phi$) to represent different tasks, since a multitask learning is conducted. The overall training process is completed when the losses of the generator and the discriminator reach an equilibrium, which takes about 24 h on a machine with 2.40 GHz CPU and an NVIDIA RTX 2070 graphics card.

VI. Experimental Results

In this section, we describe several experiments that demonstrate the achievements of GAN-CTS framework. The framework is implemented in Python3 with Keras [5] library.

TABLE IV
CTS OUTCOMES PREDICTION RESULTS OF THREE REGRESSION APPROACHES. MAPE DENOTES MAPE (%), MAXE DENOTES MAX. ABSOLUTE PERCENTAGE ERROR (%), AND CC DENOTES CORRELATION COEFFICIENT. NOTE THE VALIDATIONS ARE PERFORMED ON *Unseen Netlists*

modeling type	unseen netlist	clock power			clock wirelength			achieved skew		
		MAPE	MAXE	CC	MAPE	MAXE	CC	MAPE	MAXE	CC
multi-model uni-output (meta-model) (w. transfer learning)	ecg	4.41	24.47	0.963	1.68	21.24	0.959	5.65	32.47	0.955
	jpeg	4.53	39.65	0.950	2.89	22.52	0.957	6.36	35.43	0.951
	leon	4.78	42.83	0.957	3.94	24.54	0.953	7.34	41.92	0.944
	vga	4.31	27.45	0.964	1.37	29.61	0.961	5.83	35.61	0.967
uni-model multi-output (multi-task) (w. transfer learning)	ecg	2.14	17.46	0.972	2.66	8.57	0.971	3.92	34.83	0.958
	jpeg	3.88	20.29	0.965	1.98	12.35	0.973	3.87	29.63	0.961
	leon	2.37	16.19	0.969	2.02	14.16	0.966	4.28	27.79	0.964
	vga	2.54	11.23	0.962	1.77	17.03	0.969	4.25	31.18	0.963
uni-model multi-output (multi-task) (w.o. transfer learning)	ecg	7.63	79.40	0.920	5.43	35.59	0.945	16.86	129.48	0.837
	jpeg	8.29	67.33	0.904	6.77	42.98	0.949	15.70	112.15	0.821
	leon	8.75	77.65	0.919	5.12	38.92	0.942	12.26	127.14	0.841
	vga	6.81	47.57	0.941	5.36	59.29	0.949	11.78	91.55	0.850

Algorithm 2 GAN-CTS Training Methodology. We Use Default Values of $\alpha_r = 1e^{-4}$, $\alpha_{GAN} = 1e^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $m = 128$

Input: $\{f\}$: extracted placement features, $\{x\}$: training data, $\{Y\}$: target CTS metrics for prediction and optimization.
Input: α_r: learning rate of regression model, α_{GAN}: learning rate of GAN, m: batch size, $\{\theta_r\}_0$: initial parameters of regression model, θ_{g_0}: initial parameters of generator, $\{\theta_d\}_0$: initial parameters of discriminator, $\{\beta_1, \beta_2\}$: Adam parameters.
Output: R: regression model, G: generator, D: discriminator.

$N \leftarrow length(y)$
while $\{\theta_r\}$ do not converge **do**
  Sample a batch of training data $\{x^{(i)}\}_{i=1}^m \sim p_d$
  Take features $\{f^{(i)}\}_{i=1}^m$ corresponding to $\{x^{(i)}\}_{i=1}^m$
  for $k \leftarrow 1$ to N **do**
    $g_{r_k} \leftarrow \nabla_r[\frac{1}{m}\sum_{i=1}^m (R_k(f^{(i)}, x^{(i)}) - Y_k^{(i)})^2]$
    $\theta_{r_k} \leftarrow Adam(\alpha_R, \theta_{r_k}, g_{r_k}, \beta_1, \beta_2)$
  end for
end while
while θ_g and θ_d do not converge **do**
  Sample a batch of training data $\{x^{(i)}\}_{i=1}^m \sim p_d$
  Take features $\{f^{(i)}\}_{i=1}^m$ corresponding to $\{x^{(i)}\}_{i=1}^m$
  Sample a batch of random vectors $\{z^{(i)}\}_{i=1}^m \sim p_z$
  $g_{d_1} \leftarrow \nabla_{\theta_{d_1}}[\frac{1}{m}\sum_{i=1}^m log(D_{\theta_{d_1}}(x^{(i)}, f^{(i)}))$
    $+\frac{1}{m}\sum_{i=1}^m log(1 - D_{\theta_{d_1}}(G_{\theta_g}(z^{(i)}, f^{(i)})))]$
  $\theta_{d_1} \leftarrow Adam(\alpha_{GAN}, \theta_{d_1}, g_{d_1}, \beta_1, \beta_2)$
  $g_{d_2} \leftarrow \nabla_{\theta_{d_2}}[\frac{1}{m}\sum_{i=1}^m log(D_{\theta_{d_2}}(x^{(i)}, f^{(i)}))]$
  $\theta_{d_2} \leftarrow Adam(\alpha_{GAN}, \theta_{d_2}, g_{d_2}, \beta_1, \beta_2)$
  Sample a batch of random vectors $\{z^{(i)}\}_{i=1}^m \sim p_z$
  Sample a batch of features $\{f^{(i)}\}_{i=1}^m$
  $g_\theta \leftarrow -\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^m log(D_{\theta_{d_1}}(G_{\theta_g}(z^{(i)}, f^{(i)})))$
  $\theta_g \leftarrow Adam(\alpha_{GAN}, \theta_g, g_\theta, \beta_1, \beta_2)$
  $r \leftarrow \prod_{k=1}^N \frac{R_k(G(\{z^{(i)}\}_{i=1}^m, \{f^{(i)}\}_{i=1}^m))}{\text{auto-setting result of outcome } k}$
  $g_\theta \leftarrow -\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^m r^{(i)}$
  $\theta_g \leftarrow Adam(\alpha_{GAN}, \theta_g, g_\theta, \beta_1, \beta_2)$
end while

As mentioned in Section IV, we utilize *Cadence Innovus v18.1* to generate a database containing 115.5k clock trees with 385 different placements across 11 netlists under the TSMC-28 nm technology node. The designs we utilize are shown in Table I, which are from ISPD 2012 benchmark [23] and *OpenCores.org* To prove the generality of our framework, we only use seven netlists during the training process, and leverage the rest four designs (ECG, LEON, JPEG, and VGA) to perform the validations.

A. CTS Prediction and Interpretation Results

In this experiment, we evaluate the regression approaches on three target CTS outcomes with two evaluation metrics: MAPE, maximum absolute percentage error (MAXE), and correlation coefficient (CC). Table IV demonstrates the evaluation results of the three different regression approaches presented earlier. The first approach termed multimodel uni-output represents the meta-modeling method, where for each CTS target outcome, we build a dedicated meta-model using the structure defined in Fig. 6. The second approach named uni-model uni-output leverages the modeling method shown in Fig. 7, where we build a single model to predict three target CTS outcomes simultaneously. Finally, the third approach follows the modeling structure of the second approach, however, instead of using the features ($\in R^{512}$) extracted from the transfer learning flow, we handcrafted four features to represent different designs. These four features include number of cells, number of flip flops, number of nets, and number of ports in a given design.

It is shown that the second approach (multitask learning with transfer learning enabled) achieves lower evaluation errors among all target metrics on the *unseen* netlists than the other approaches. Note that the training time of the multitask learning approach is about 3 h, where the training time of the meta-learning approach is about 6 h (calculated by summing training time of individual meta-models). Two main conclusions can be drawn from this experiment. First, the multitask learning takes the advantage of the fact that different CTS outcomes are not independent of each other. Therefore, shared layers not only expedite the training process but also improve the prediction accuracy of CTS targets. Second, it is demonstrated that the transfer learning approach provides a better way to characterize different designs compared with using the manually enumerated features. Indeed, many important design characteristics related to CTS process, such as the distribution of flip flops and the metal layer usage of trial routing are not straightforward to be enumerated

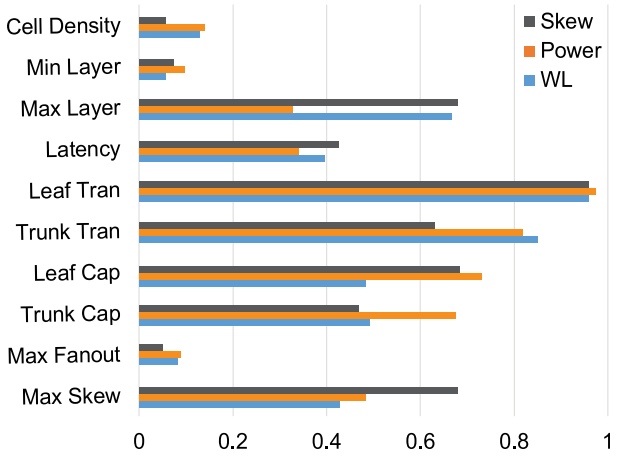

Fig. 10. Relative importance of CTS input parameters on skew, power, and wirelength for JPEG benchmark.

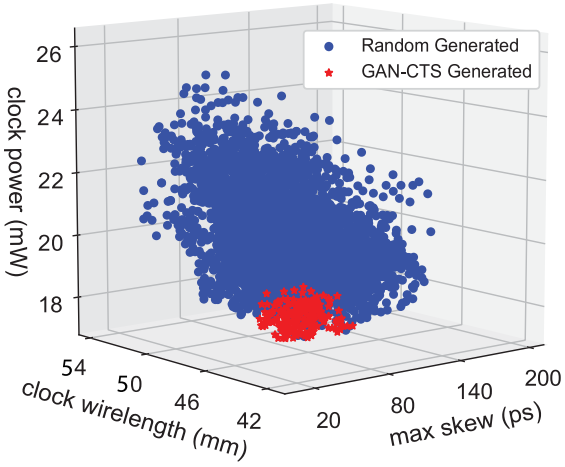

Fig. 11. Distributions of random generated versus GAN-CTS generated clock trees on the ECG benchmark. The commercial auto-setting achieved a clock tree with values of 23.56 mW in clock power, 49.69 mm in clock wirelength, and 16 ps in skew.

manually. Therefore, transfer learning provides a great benefit to characterize different designs.

In spite of the high prediction accuracy achieved, designers would not benefit much without explaining the predictions made by the model. Understanding the reasons behind the predictions is crucial. As shown in Fig. 10, we evaluate the importance of each CTS input parameter based on the predictions presented in Table IV. As mentioned in Section V-B, we define the importance through a gradient-based attribution method named DeepLIFT [25]. The algorithm quantifies the relevance of input parameters with respect to different outputs. Since the input of the regression model contains the CTS input parameters and the extracted features, in this interpretation experiment, we focus on determining the relative importance among CTS input parameters by further normalizing the relevance scores to [0, 1]. Note that the normalization is performed within the CTS input parameters. Below, we explain two important phenomenons observed from Fig. 10.

1) The slew constraints for leaf cells and trunk cells have great impacts on clock power and clock wirelength. Indeed, with a tight slew constraint, more buffers need to be inserted to meet the timing target, which ultimately results in higher clock power and clock wirelength.
2) The max EGR layer has high impacts on the maximum skew and clock wirelength. The reason is that signal nets are often routed in top metal layers (e.g., M5 and M6). If signal nets are forced to route in low metal layers (e.g., M1. M2) that are reserved to route clock nets, there will be many detours in the clock routing because clock nets will inevitably use low metal layers to connect the sinks, which results in long clock paths and hence a large maximum skew.

B. CTS Optimization Results

In this experiment, we demonstrate the optimization results achieved by our GAN-CTS framework compared with the Bayesian optimization [26] technique leveraged by previous work [20] and the auto-setting offered by the commercial tool, where a joint optimization is performed on three target CTS metrics: 1) clock wirelength; 2) clock power; and 3) the maximum skew. Fig. 11 first shows the optimization result of the ECG benchmark, where the blue dots denote the original clock trees in the database, and the red stars represent the clock trees generated by GAN-CTS. To plot the figure, we first take the extracted features of the pre-CTS placements as conditional inputs, and then utilize the trained generator to suggest 100 sets of CTS input parameters. With these suggested parameters sets, we further leverage the commercial tool to perform actual CTS processes. Finally, according to the target optimization metrics, we plot the scatter distributions of the clock trees suggested by GAN-CTS together with the ones originally generated in the database. Note that the input parameters of the clock trees in the database are randomly sampled from the ranges shown in Table II.

The detailed optimization results on the four unseen netlists are shown in Table V and the corresponding CTS input parameters are shown in Table VI. The method "GAN-CTS + bayes" denotes the combined optimization technique presented in Section V-D3, where we take the GAN-CTS suggested parameter sets as the initial sets of the Bayesian optimization process and leverage the sequential domain reduction technique [28] to refine the search space. We observe in general, the combined technique reaches better CTS optimization results in terms of the reward defined in (5), and the proposed GAN-CTS framework outperforms the basic Bayesian optimization technique adopted by the previous work [20] across all *unseen* designs. This in fact demonstrates that the proposed GAN-CTS framework provides better starting points for the vanilla Bayesian optimization approach. Note that the selection of the GAN-CTS generated trees is conducted by taking the clock tree with the least maximum skew among the 100 trees suggested. Fig. 12 further shows the Bayesian optimization process on the VGA benchmark, where the iteration stops when the reward evaluation no longer improves after 15 iterations. Finally, Fig. 13 further exhibits the layout comparison of the four testing benchmarks (unseen during training). It is observant that the clock wirelength of the GAN-CTS optimized tree is much shorter than the one auto-generated by the commercial engine.

C. Success Versus Failure Classification Results

In the final experiment, we demonstrate the classification results achieved by the discriminator of determining successful and failed CTS runs. As mentioned in Section III, success and failure are defined by comparing the CTS metrics of the clock trees generated by GAN-CTS to the one auto-generated by the commercial tool. If two out of three target metrics

TABLE V
ACHIEVED CLOCK METRICS COMPARISON BETWEEN COMMERCIAL AUTO-SETTING (AUTO), BAYESIAN OPTIMIZATION (BAYES), AND GAN-CTS. THE METHOD "GAN-CTS + BAYES" DENOTES USING THE GENERATOR SUGGESTED CTS PARAMETER SETS AS THE INITIAL SOLUTIONS OF BAYESISAN OPTIMIZATION ALONG WITH THE SEQUENTIAL DOMAIN REDUCTION TECHNIQUE [28]. NOTE THAT THE FOUR BENCHMARKS ARE *Unseen* DURING THE TRAINING PHASE

netlist	CTS metrics	auto-setting	bayes	GAN-CTS	GAN-CTS + bayes
ecg	# inserted buffers	417	128 (-69.3%)	96 (-76.9%)	92 (-77.9%)
	clock power (mW)	23.56	19.11 (-18.8%)	18.72 (-20.5%)	18.61 (-21.0%)
	clock WL (mm)	49.69	43.09 (-13.2%)	42.36 (-14.7%)	42.30 (-14.8%)
	maximum skew (ns)	0.016	0.018 (+12.5%)	0.014 (-12.5%)	0.014 (-12.5%)
jpeg	# inserted buffers	1093	296 (-72.9%)	240 (-78.0%)	268 (-75.5%)
	clock power (mW)	33.26	27.32 (-17.9%)	26.33 (-20.8%)	27.14 (-18.4%)
	clock WL (mm)	130.71	118.07 (-9.7%)	115.22 (-11.9%)	116.49 (-10.8%)
	maximum skew (ns)	0.022	0.024 (+9.0%)	0.024 (+9.0%)	0.023 (+4.5%)
leon	# inserted buffers	2962	1453 (-50.9%)	824 (-72.2%)	798 (-73.1%)
	clock power (mW)	81.12	74.28 (-8.4%)	69.69 (-14.1%)	67.94 (-16.2%)
	clock WL (mm)	326.36	307.81 (-5.6%)	296.15 (-9.2%)	292.88 (-10.2%)
	maximum skew (ns)	0.03	0.032 (+6.6%)	0.028 (-6.6%)	0.029 (-3.3%)
vga	# inserted buffers	505	186 (-63.1%)	109 (-78.4%)	127 (-74.8%)
	clock power (mW)	33.72	29.16 (-13.5%)	26.74 (-20.7%)	27.75 (-17.7%)
	clock WL (mm)	52.61	45.31 (-13.8%)	41.29 (-21.5%)	41.60 (-20.9%)
	maximum skew (ns)	0.036	0.033 (-8.3%)	0.023 (-36.1%)	0.022 (-38.8%)

TABLE VI
GAN-CTS SUGGESTED AND COMMERCIAL AUTO-SETTING'S CTS INPUT PARAMETERS (REFER TO TABLE V). NOTE THAT THE COMMERCIAL CLOCK ROUTER HAS THE SAME AUTO-SETTING VALUES FOR DIFFERENT DESIGNS. THE CAPACITANCE CONSTRAINTS IN THE AUTO-SETTING SCENARIO ARE VARIED FROM NET TO NET, WHICH ARE SUBJECT TO THE MAX CAPACITANCE CONSTRAINT OF THE DRIVING PINS

CTS parameters	ecg	jpeg	leon	vga	auto setting
max skew (ns)	0.11	0.14	0.04	0.02	0.05
max fanout	120	139	173	84	100
max cap trunk (pF)	0.24	0.16	0.28	0.11	net-based
max cap leaf (pF)	0.22	0.12	0.19	0.25	net-based
max slew trunk (ns)	0.081	0.285	0.066	0.123	0.05
max slew leaf (ns)	0.047	0.107	0.098	0.074	0.05
max latency (ns)	0.31	0.26	0.15	0.28	0.1
max eGR layer	5	4	4	5	6
min eGR layer	3	2	3	3	2
max buffer density	0.72	0.63	0.69	0.68	0.75

TABLE VII
CONFUSION MATRIX OF SUCCESS VERSUS FAILURE CLASSIFICATION IN LEON BENCHMARK. FAILURE MEANS WORSE THAN AUTO-SETTING

		Predictions		
		Success	Failure	Total
Ground Truths	Success	6974	522	7496
	Failure	498	2251	2749
	Total	7472	2773	10245

are better, then we consider it as a success. Table VII summarizes the classification results in a confusion matrix with NOVA benchmark. The accuracy and the F1-score are 0.930 and 0.932, respectively. With the accuracy demonstrated, we believe designers can not only benefit from the generator but also the discriminator by efficiently pruning out the CTS input parameter sets that have little advantage over the commercial auto-setting.

VII. DISCUSSION

The proposed framework, GAN-CTS, is a helper model (rather than a surrogate model) of commercial CTS engines, whose goal is to support the engines to find the CTS input parameter combinations that result in optimized clock trees. In this work, we take *Cadence Innovus* as our reference commercial tool, however, the proposed method can be easily applied to other tools which also parameterize the CTS process into different input settings. Note that the goal of this work is *not* to replace the existing commercial CTS engines, but to provide tool users fast and reliable CTS prediction and optimization techniques without spending significant amount of time in design space exploration. In the below sections, we further describe different aspects of the proposed CTS modeling method in detail.

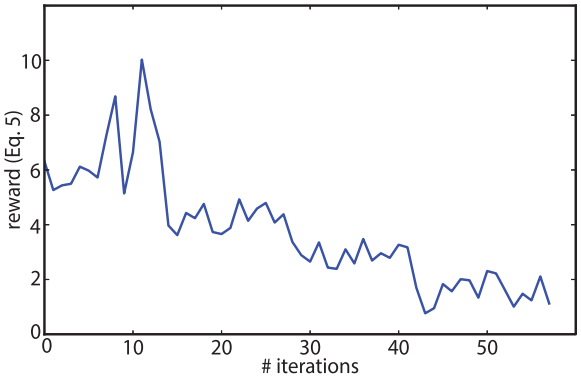

Fig. 12. Bayesian optimization on VGA benchmark (starting from random sampled CTS parameter sets). Reward r is defined in (5).

A. Nontriviality of the CTS Modeling Problem

The CTS modeling problem we are dealing with in this work is in fact a high-dimensional modeling problem, which is stated in [13] to be difficult and nontrivial due to the *curse of high dimensionality* [29]. To quantify the nontriviality of this problem in our experimental settings, we perform a slew sweeping experiment on the VGA benchmark, where we generate 500 clock trees by sweeping the leaf and trunk slew targets while fixing all other input parameters in Table II as auto-set. The experimental result is shown in Fig. 14. For the three CTS metrics we focus on in this work, we plot the scatter distribution of the 500 clock trees, where the red-colored dots denote the trees whose achieved metric is better than the one

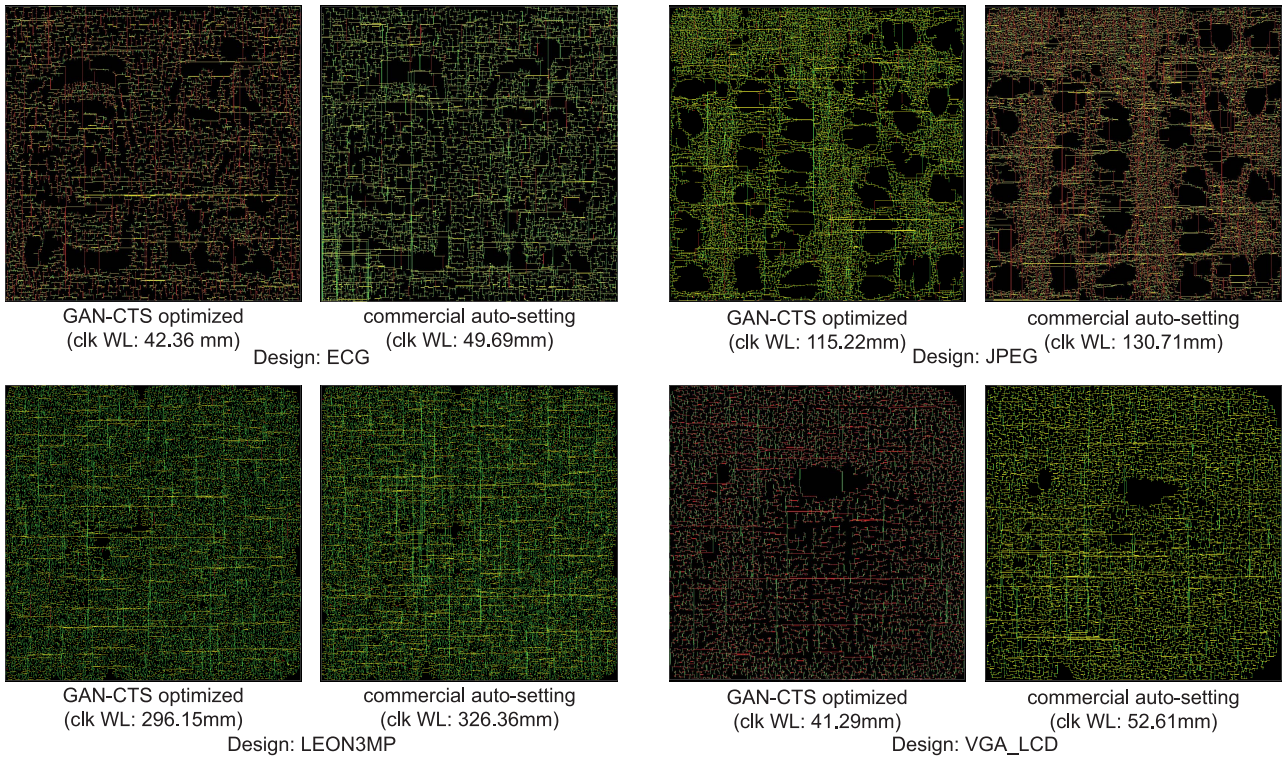

Fig. 13. Clock tree layout comparison of four validation benchmarks. GAN-CTS optimized clock trees have observant clock wirelength saving. The detailed comparisons are reported in Table V.

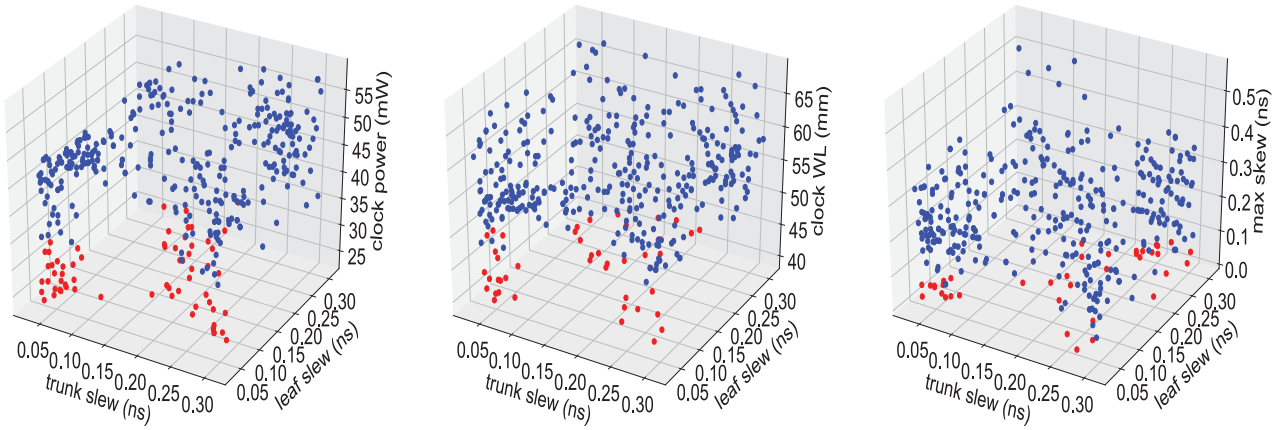

Fig. 14. VGA slew sweeping experiments. Out of the ten CTS input parameters as shown in Table II, we sweep around the leaf and trunk target slew values while fixing others as auto-set and generate 500 clock trees in total. For each CTS metrics (i.e., clock power, clock wirelength, and maximum achieved skew), we plot the scatter distribution of the 500 clock trees denoted in blue and red dots, where red dots denote the ones whose underlying CTS metric are better than the auto-generated clock tree from the commercial tool. In summary, compared with the auto-generated clock tree, there are 61 (out of 500) trees whose clock power are better, 50 whose clock wirelength are better, and 32 whose achieved skew values are better, where the corresponding Venn diagram is shown in Fig. 15.

achieved by the commercial tool auto-generated clock tree. In the figure, we observe that there is no apparent "sweet spot" that guarantees high-quality clock trees. In addition, Fig. 15 shows the Venn diagram from the three subplots in Fig. 14 and Fig. 16 demonstrates further the distributions of the tree targeted CTS metrics in this work, where we observe that there is no apparent sweet spot of the slew targets that guarantee to result in optimized clock trees. It is shown that out of 500 generated clock trees, only 14 of them (2.8%) whose all three CTS outcomes (clock power, clock wirelength, and achieved skew) are better than the ones achieved by the auto-generated clock tree.

B. Train/Test Splitting of Benchmarks

The training and testing split among the 11 designs utilized in this work is not performed in a purely random fashion. Instead, we strive to make the training set to be "comprehensive" that covers a variety of designs from small to large. One of the limitations of the proposed work is that the model is required to be pretrained on a few designs, however, as shown in the experiment, after training on the seven designs as shown in Table I, GAN-CTS is able to achieve accurate prediction and high-quality prediction results on the largest design, *(*LEON), which also has the largest power consumption. The main reason is that GAN-CTS does

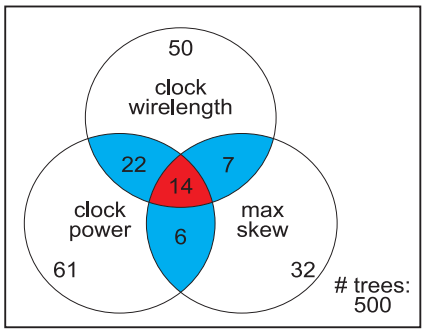

Fig. 15. Venn diagram of the VGA slew sweeping experiment (Fig. 14). Note that a number on a colored region denotes the number of trees fall into that region, where a number on an uncolored region denotes the number of trees in the shape boundary.

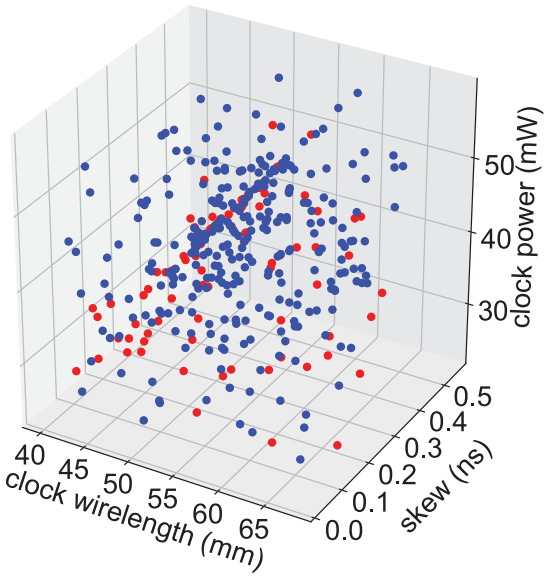

Fig. 16. CTS metric distribution of the VGA slew experiment (Fig. 13). The red dots denote the clock trees that are achieved with both trunk slew and leaf slew targets smaller than 0.1 ns.

not perform the optimization completely based on previous experience (seen designs). Instead, given a new design, it leverages unsupervised techniques to extract the underlying design features in order to generate high-quality clock trees. Finally, we expect the proposed framework to achieve lower MAPE/MAXE prediction error and better optimization results if a bigger a higher variety of training set is available.

C. Discussion of Prediction Results

As shown in Table IV, we observe that the MAXE (worst-case prediction error) is slightly high for the skew prediction. This in fact can be accounted in twofold. First, as mentioned in Section V, the regression model is trained by least-square regression [6], which minimizes the mean squared (L2) error to update the network parameters. It is known that the L2 error (loss) minimization tends to optimize the average prediction error across all samples that reaches stable solutions, where the L1 error minimization tends to optimize the error on outliers and thus results in unstable (sparse) solutions [2]. Therefore, as shown in the table, even the MAPE (average error) for skew prediction is bounded within 5%, some outliers still create corner cases that aggravate the MAXE metric. Second, as pointed out in previous works [12], [13], timing in general is a hard-to-predict metric due to the sophisticated behavior of the commercial timing engines. In particular, during CTS, commercial tools will often override the skew target that is taken as input in order to optimize other metrics, such as power and wirelength, which results in the uncorrelation between the target closure and the final achieved outcome.

D. Discussion of Optimization Results

The success of GAN-CTS on optimizing CTS metrics can mainly be explained in twofold. First, instead of performing block-box optimization as the Bayesian optimization technique, GAN-CTS leverages the generator to learn the key distribution for different designs through conditional generative learning, which gives our framework better generality over other approaches. Second, the proposed transfer learning technique well differentiates various designs. The extracted features that contain precious design information help the framework to find better and more curated CTS parameter sets that result in optimized clock trees for unseen netlists.

VIII. Conclusion

In this article, we have shown that ML offers promising solutions for designers to reach the desired CTS targets with a small amount of effort. We have proposed a novel framework named GAN-CTS that uses discriminative techniques to predict and classify the CTS outcomes as well as leverages generative adversarial learning to optimize the desired metrics. The experimental results conducted on the unseen netlists demonstrate the proposed framework is generalizable and practical.

Acknowledgment

The authors are thankful to the constructive comments from anonymous reviewers that help enhance this article.

References

[1] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, "Towards better understanding of gradient-based attribution methods for deep neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–16.
[2] S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[3] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, no. 3, pp. 322–355, 1988.
[4] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2016, pp. 785–794.
[5] F. Chollet *et al*. "Keras." 2015. [Online]. Available: https://keras.io
[6] N. Cressie, "Fitting variogram models by weighted least squares," *J. Int. Assoc. Math. Geol.*, vol. 17, no. 5, pp. 563–586, 1985.
[7] A. Datli, U. Eksi, and G. Isik, "A clock tree synthesis flow tailored for low power." 2013. [Online]. Available: https://www.design-reuse.com/articles/33873/clock-tree-synthesis-flow-tailored-for-low-power.html
[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[9] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 2014.
[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*.

[12] A. B. Kahng, B. Lin, and S. Nath, "Enhanced metamodeling techniques for high-dimensional IC design estimation problems," in *Proc. Conf. Design Autom. Test Europe*, 2013, pp. 1861–1866.
[13] A. B. Kahng, B. Lin, and S. Nath, "High-dimensional metamodeling for prediction of clock tree synthesis outcomes," in *Proc. ACM/IEEE Int. Workshop Syst. Level Interconnect Prediction (SLIP)*, 2013, pp. 1–7.
[14] A. B. Kahng and S. Mantik, "A system for automatic recording and prediction of design quality metrics," in *Proc. ISQED*, 2001, pp. 81–86.
[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[16] S. Koh, Y. Kwon, and Y. Shin, "Pre-layout clock tree estimation and optimization using artificial neural network," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, 2020, pp. 193–198.
[17] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Stat.*, vol. 22, no. 1, pp. 79–86, 1951.
[18] Y. Kwon, J. Jung, I. Han, and Y. Shin, "Transient clock power estimation of pre-CTS netlist," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2018, pp. 1–4.
[19] 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 *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2019, pp. 1–8.
[20] Y. Ma, Z. Yu, and B. Yu, "CAD tool design space exploration via Bayesian optimization," 2019, *arXiv:1912.06460*.
[21] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.
[22] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*.
[23] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke, and C. Zhuo, "The ISPD-2012 discrete cell sizing contest and benchmark suite," in *Proc. ACM Int. Symp. Int. Symp. Phys. Design*, 2012, pp. 161–164.
[24] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: Unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 2018, pp. 6638–6648.
[25] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," 2016, *arXiv:1605.01713*.
[26] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, vol. 25. Red Hook, NY, USA: Curran Assoc., 2012, pp. 2951–2959.
[27] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," 2009, *arXiv:0912.3995*.
[28] N. Stander and K. J. Craig, "On the robustness of a simple domain reduction scheme for simulation-based optimization," *Eng. Comput.*, vol. 19, no. 4, pp. 431–450, 2002.
[29] G. V. Trunk, "A problem of dimensionality: A simple example," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 3, pp. 306–307, Jul. 1979.
[30] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 2013.
[31] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, "Generative adversarial networks: Introduction and outlook," *IEEE/CAA J. Automatica Sinica*, vol. 4, no. 4, pp. 588–598, Sep. 2017.
[32] D. Whitley, "A genetic algorithm tutorial," *Stat. Comput.*, vol. 4, no. 2, pp. 65–85, 1994.
[33] Z. Xie *et al.*, "RouteNet: Routability prediction for mixed-size designs using convolutional neural network," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2018, p. 80.
[34] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," 2015, *arXiv:1505.00853*.

Yi-Chen Lu (Student Member, IEEE) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2017, and the M.S. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2019, where he is currently pursuing the Ph.D. degree under Professor S. K. Lim's guidance.

His current research focuses on devising machine learning, reinforcement learning, and graph algorithms to enhance the electronic design automation flow for 2-D and 3-D integrated circuits.

Jeehyun Lee (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering from Sogang University, Seoul, South Korea, in 2017, and the M.S. degree in electrical and computer engineering from Georgia Tech, Atlanta, GA, USA, in 2020, where she is currently pursuing the Ph.D. degree in electrical and computer engineering.

She currently belongs in the Ultrasound Imaging and Therapeutics Research Laboratory, Georgia Tech as of Fall 2019. Her current research interests focus on high resolution functional ultrasound imaging of eye, especially on detection of aqueous humor outflow and shear wave elastography of sclera for diagnosis of glaucoma.

Anthony Agnesina (Student Member, IEEE) received the Diplôme d'Ingénieur degree from CentraleSupélec, Gif-sur-Yvette, France, in 2016, and the M.S. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2017, where he is currently pursuing the Ph.D. degree with the School of Electrical and Computer Engineering.

His current research interests include 3-D memory architectures, computer-aided design of VLSI circuits, and applied machine learning to electronics.

Kambiz Samadi (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees from the University of California at San Diego, San Diego, CA, USA, in 2007 and 2010, respectively.

He joined Qualcomm Research, San Diego, in 2011, where he focused on 3-D IC EDA solutions and 3-D IC architecture-level design space explorations. Since 2018, he has been working on the advanced timing/signoff methodology development as well as investigating machine learning-driven design methodologies for latest technology nodes. He has coauthored more than 50 publications in refereed journals and conferences and has more than 45 patents.

Sung Kyu Lim (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the Computer Science Department, University of California at Los Angeles, Los Angeles, CA, USA, in 1994, 1997, and 2000, respectively.

He is a Professor with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. He joined the School in 2001. He is the author of Practical Problems in *VLSI Physical Design Automation* (Springer, 2008) and *Design for High Performance, Low Power, and Reliable 3-D Integrated Circuits* (Springer, 2013). He has published more than 350 papers on 2.5-D and 3-D ICs. His research focus is on the architecture, design, test, and EDA solutions for 2.5-D and 3-D ICs. His research is featured as Research Highlight in the Communication of the ACM in January, 2014.

Prof. Lim received the National Science Foundation Faculty Early Career Development (CAREER) Award in 2006, and the ACM SIGDA Distinguished Service Award in 2008. He received the Best Paper Award from ATS'12, IITC'14, and EDAPS'17. His works have been nominated for the Best Paper Award at several top venues in EDA and circuit/packaging design. He was an Associate Editor of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS from 2007 to 2009 and the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS from 2013 to 2018. He has been leading two projects (CHIPS and 3DSOC) under DARPA Electronics Resurgence Initiative since 2017.