Circuit Synthesis Using Generative Adversarial Networks (GANs)

Tinghao Guo*, Daniel R. Herber†, James T. Allison‡
University of Illinois at Urbana-Champaign, Urbana IL, 61801

In this article, we present a generative adversarial network (GAN) design framework for circuit synthesis design. While the existing methods, such as domain knowledge- and metaheuristic-based methods have been applied successfully in circuit synthesis, they are suited for certain types of circuits and thus limited. Recent advances in efficient circuit topology enumeration methods have proven to be useful for circuit synthesis problem with moderately-sized component catalogs; enumeration methods, however, cannot be used as a solution method for problems with large catalogs. Here we present a strategy based on Generative Adversarial Networks (GANs) that facilitates solution of synthesis problems that are too large for enumeration. A GAN is a generative unsupervised learning model where a lower-dimensional latent space is predefined as an abstract representation of the circuit topology space. The GAN is trained to learn the mapping from the latent space to circuit topology space. Circuit topologies can be produced using the generator, and a discriminator helps to improve generator quality. A comparative study is conducted using several recent GAN models, including GANs, DCGANs, WGANs, and improved WGANs. We designed and carried out a series of numerical experiments to evaluate the ability of these GANs to generate new circuit topologies and support design exploration. Two circuit synthesis case studies are presented here: frequency response matching and low-pass filter realizability. The work presented here is an effort to leverage recent advancements in artificial intelligence to 1) create the ability to solve larger synthesis problems than previously possible, and 2) gain deeper design insights into engineering synthesis problems.

I. Introduction

Circuit synthesis is a challenging design task that can be solved in certain cases by a human designer based on technical understanding, intuition, and knowledge of previous circuit design topologies. Extending these design strategies to more challenging circuit synthesis problems, however, is often not possible, and more formal synthesis strategies are required. Here we describe a circuit synthesis problem as one where a set of available electronic components is provided, and from this set components are selected and their connectivities are specified to define a circuit topology. Once a circuit topology is determined, component sizing and other continuous parameters can be optimized to determine the best possible performance of a given circuit topology. This step is sometimes referred to as *sizing optimization*, and is important for comparing the value of candidate topologies with respect to desired circuit properties. Over the past few decades, a number circuit synthesis methods have been developed and studied, including domain knowledge-based methods,[1–3] metaheuristic-based methods,[4–10] and enumeration-based methods.[11,12]

Significant human expertise and experience are often required for domain knowledge-based methods to be successful. Reliance on human intuition and cognition can limit the complexity of circuits that can be designed successfully using this approach. Metaheuristic-based methods can help reduce reliance on human expertise, and can help navigate more complicated circuit synthesis problems. For instance, evolutionary algorithms (EAs) have been used to solve circuit synthesis problems involving frequency- and time-domain

*Graduate Student, Department of Industrial and Enterprise Systems Engineering, 104 S. Mathews Ave, Urbana, IL, 61801
†Postdoctoral Research Associate, Department of Industrial and Enterprise Systems Engineering, 104 S. Mathews Ave, Urbana, IL, 61801, AIAA Member
‡Associate Professor, Department of Industrial and Enterprise Systems Engineering, 104 S. Mathews Ave, Urbana, IL 61801, AIAA Senior Member

analysis.[4,5] Ochotta et al. presented a synthesis method for creating high-performance analog circuits using simulated annealing, another type of metaheuristic algorithm.[10] While metaheuristic methods have had some success, they do have limitations. Often the algorithms require tuning of many parameters (e.g., initialization, selection, crossover, and mutation in the case of EAs). Furthermore, a good initial population is often important for achieving stable and robust algorithm performance. Direct encodings (design representations or genotype) are used in the application of EAs to circuit synthesis problems. Direct encodings result in high-dimensional design spaces that are difficult to navigate for large systems, and can limit the ability to locate global optima. Indirect encodings with targeted reduced-dimension representations better mimic biological evolutionary processes, and have proven to be useful for supporting the design of larger-scale systems.[13–16] More generalizable indirect representations have been proposed, but for restricted classes of topology optimization problems. For example, an indirect generative design representation was introduced by Khetan et al. where certain design constraints were guaranteed to be satisfied in an implicit manner.[14] This resulted in a more efficient design space search (i.e., higher-quality solutions and lower computational expense), but was developed only for a specific homogeneous topology optimization problem. All nodes represent the same material with the same type of functionality and allowed connectivity, and are parameterized by the same continuous variables for size optimization. Cheney et al. successfully applied computational pattern-producing networks (CPPNs) to design of soft robots with materials.[13] While the combinatorial design space was large, continuous design variables were not considered, and connectivity could not be rearranged. Only network node types could be changed (representing material types, or no material as an option). The fixed connectivity was simple (e.g., based on geometrically-close neighbors), whereas in more general heterogeneous topology optimization problems a much wider range of connections must be represented.

While enumeration-based solution methods do have limitations on the size of systems that can be considered, these methods can guarantee identification of optimal system topologies, and recent advances are making efficient enumerative search methods viable for larger-scale problems than previously thought. As early as in the 1930s, enumeration was used to study electrical network design properties.[17] More recently, Bruccoleri et al. systematically enumerated all 2-MOS-transistor wide-band amplifiers.[18] Other applications addressed using enumerative methods include hybrid powertrains,[19] gear trains,[20] and biochemical networks.[21] However, in many cases, naïve enumeration is impractical for designing systems of practical size. In Ref.,[21] there were 16,038 possible three-node enzyme networks, and a total of 1.6×10^8 different gene circuits to be analyzed. It would be impractical to expand it to even a slightly larger biochemical network (e.g., four nodes) using naïve enumeration. A continuous relaxation was later applied to solve a version of this problem that was expanded to four nodes.[22]

Extending to even larger problems is a significant challenge, especially for the general case of dynamic system synthesis. In addition to the metaheuristic methods described above, including indirect representations for EAs,[13] generative and grammar-based strategies have been proposed for general synthesis problems.[23–25] As a strategy to generate data to support the creation of more effective generative methods, Herber and Allison developed an efficient enumeration strategy for general synthesis problems based on perfect matching graph theory and a brute-force search algorithm.[11,12] In addition to generating useful design data, it was discovered that with careful enumeration algorithm design, it is possible to enumerate all unique and feasible topologies for much larger synthesis problems than previously anticipated. As a result, efficient enumeration methods can be used directly to solve some synthesis problems of practical importance. These efficient enumeration strategies have been applied recently to circuit synthesis problems, and have revealed many previously unknown and non-dominated designs that were not identified via metaheuristics.[12]

Here we categorize the circuit synthesis problems into three cases.[26] *Case 0* refers to the situation where it is possible to enumerate and evaluate all topologies of interest (i.e., efficient enumeration is sufficient).[11] *Case 1* involves synthesis problems where all unique, feasible topologies *can* be enumerated in a practical amount of time, but topology evaluations are computationally expensive, thus precluding the possibility of quantitatively comparing design options through enumeration alone. *Case 2* synthesis problems are those where it is not only impractical to evaluate all topologies of interest, but it is also impractical to enumerate them. One possible strategy for solving Case 1 synthesis problems was presented by Guo et al. where active learning was used to select evaluation subsets strategically and adaptively.[16,26] An alternative approach for solving Case 2 problems is presented here that leverages artificial intelligence and design data obtained via efficient enumeration.

II. Motivation

Here we propose the use of a generative model to produce circuit topologies for use in Case 2 circuit synthesis problems. A generative model, using language from the machine learning literature, is a probabilistic model that describes how data was generated. Variational autoencoders (VAEs) and generative adversarial networks (GANs) are two important types of generative models. VAEs have been applied to topology optimization and microstructure design. For instance, Guo et al. developed an indirect representation for heat spreader design using VAEs and style transfer in topology optimization.[16] Cang et al. predicted physical properties of heterogeneous materials via a deep VAE-based model.[27] However, it is well-recognized that VAEs tend to generate blurry samples due to the maximum likelihood training paradigm.[28] In contrast, GANs adopt a generator-discriminator mechanism in an adversarial manner and have exhibited a number of advantages over other generative models. For example, the generator requires few restrictions (e.g., no Markov chains or variational bounds are needed), and GANs are better at generating sharp samples than VAEs.[29]

In a GAN, the generator attempts to produce "fake" samples generated from a low-dimensional latent vector that avoid being detected as fake by the discriminator. This adversarial mechanism is equivalent to a zero-sum game. GANs have been used to synthesize a number of realistic objects for design, such as interior design[30] and 3D objects.[31] However, GANs in engineering design, particularly circuit synthesis, have not been studied thoroughly. Here we utilize GANs as an indirect design representation for circuit synthesis problems. We design several numerical experiments to investigate the effectiveness of GANs in this context, including several GAN variants. As highlighted above, one objective here is to generate feasible topologies in a manner that implicitly satisfies circuit constraints to support more efficient Case 2 synthesis problem design search. This GAN-based approach was investigated using two electric circuit design case studies: 1) a frequency response optimization problem, and 2) a low-pass filter feasibility problem.

In this work, we designed the numerical experiments to test the efficiency of generating circuit topologies for the GAN-based method. Figure 1 is a Venn diagram that helps illustrate the rationale behind the use of GANs for Case 2 indirect encodings. In practice, it would be impossible to obtain all possible unique and feasible circuit designs (i.e., set Ω) via enumeration. Only a statistical sample X that represents Ω can be obtained in a practical amount of time. Here we train different GAN models using X, and generate a new design set called M through the generator. Suppose there are T out of M designs that are feasible (i.e., in Ω). It is desirable to have the feasibility ratio $|T|/|M|$ as large as possible. The numerical experiments help quantify the effectiveness of various GAN models via this ratio, with the expectation that larger $|T|/|M|$ ratios will support more efficient topological design space exploration. Other factors will also have an impact, such as indirect encoding dimension, feasible design space (Ω) properties, and search algorithm properties.

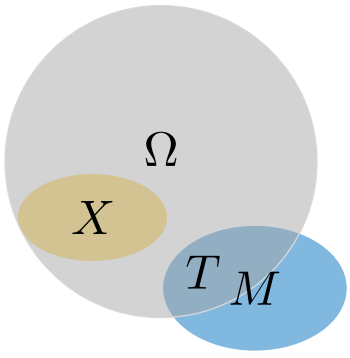


Figure 1: A Venn diagram for the GAN-based methodology.

It should be noted here that GANs have been used successfully in a similar way using images of human faces.[30,32] For example, from a small set of human faces X, GANs have been trained to generate new faces not in X that appear to be realistic. One could define realistic appearance to be a feasibility requirement. Generated faces that look realistic would then be the set T, and unrealistic faces would be $M \backslash \Omega$. The discriminator improves quality of generated faces by detecting fake ones. This established use of GANs in realistic synthetic human face generation from limited real data was observed to be parallel to the needs posed by Case 2 synthesis problems, and provides the rationale for choosing to investigate GANs for this

purpose.

Using the GAN-based method (or other similar indirect encodings) for Case 2 synthesis problems involves a tradeoff between formulation and solution accuracy. Enumeration-based methods find exact solutions to Case 0 synthesis problems (perfect solution accuracy), but only for simplified topological design spaces. It would be helpful for methods to exist that support solution of richer topological design problems. Here we aim to expand the scope of Case topological design problems that can be solved in a practical amount of time. To achieve this improved formulation fidelity, we make concessions with solution accuracy. The scope problems that can be solved is expanded, but the solutions are inexact.

Figure 2 provides a graphical representation of problem formulation decision tradeoffs to help explain the motivation for the design synthesis strategy presented here.[33] It illustrates three dimensions of problem formulation decisions, including design representation (how comprehensively candidate designs are considered), metrics for comparing the desirability of design candidates, and a predictive model mapping from design representation to comparison metrics. The origin of this space corresponds to the 'ideal' problem formulation with no error with respect to system behavior preferences and modeling, and no exclusion of possible designs. This is unachievable in practice, but is helpful for conceptualizing how we can improve formulations along one or two dimensions by making concessions in other dimensions, or by making concessions in solution accuracy. Moving closer to the origin corresponds to improved formulation accuracy.

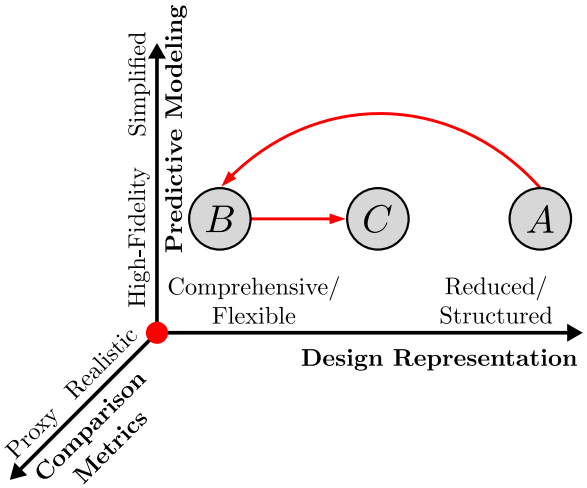


Figure 2: Design formulation analysis of the GAN-based methodology.

If we improve the comprehensiveness of the design representation to expand the set of topologies considered, we can move from point A in the formulation space to the left toward the origin (point B), but this problem is no longer solvable via enumeration (Case 2 synthesis[a]). We also note that reducing predictive model fidelity or comparison metric realism will not make a Case 2 problem solvable, unlike in some Case 1 problems, because the complete set of topologies cannot be enumerated. We could attempt to solve such a problem using an EA with a direct encoding, but previous works have shown this is unlikely to be successful for such problems. Introducing a GAN as an indirect encoding reduces design representation dimension, and moves the formulation to point C, which is the right of point B, but not as far right as point A. We recognize that a GAN-based encoding will not cover the design space as comprehensively as point B (some formulation inaccuracy/greater distance from that origin compared to B), but gives up some design coverage in return for practical solvability. In summary, by expanding the topological design space (moving from $A \rightarrow B$) and then representing this expanded design space approximately using a GAN (moving from $B \rightarrow C$), we create a net improvement in problem formulation, producing a more accurate formulation that may be solvable using indirect EA encodings. The formulation is more accurate than the Case 0 synthesis problem (point A), but the solution is approximate (not exact) due to the nature of EAs or other solution methods that would apply here.

The GAN-based method presented here, in essence, restructures the design space in terms of an abstract

[a]Recall that Case 1 synthesis problems are not solvable via enumeration alone because of evaluation expense, not because the topologies cannot be enumerated. Here we focus on cases where the topological design space is too large to support enumeration; this property is independent of evaluation expense.

design representation. It may permit either low-or high-fidelity comparison metrics and predictive models, but more importantly directly tackles the challenge of not having enumerative data from the desired problem directly available for use in the solution strategy.

The contributions in this article include: 1) a new GAN-based synthesis strategy that entails a new perspective in designing electronic circuits that is fundamentally distinct from established circuit design methodologies. Specifically, artificial intelligence and design data are leveraged to restructure topological design spaces to support efficient exploration. 2) Several GAN models are tested, and the improved WGAN is shown to be the most effective in identifying the feasible circuits using the reduced-dimension design representation. This indirect representation supports the execution of an active learning strategy by facilitating the generation of high-performance circuit topologies in an efficient manner. To our best knowledge, this is the first work where adversarial learning is applied to electronic circuit synthesis design.

The remainder of this article is organized as follows. The next section introduces the basics of the GANs. Section IV describes the heterogeneous circuit synthesis design problems and associated design data. Section V outlines the numerical experiments. The numerical results are reported in Section VI. The discussion and conclusion are presented at end of this article.

III. Generative Adversarial Neural Networks (GANs)

A generative adversarial network (GAN) is a class of unsupervised learning models introduced by Goodfellow et al.[32] The GAN is composed of two artificial neural networks, a generator and discriminator. Figure 3 illustrates a basic GAN framework. Let $X_{\mathrm{r}} = \{\mathbf{x}\}_i^N$ for $i = 1, 2, \ldots, N$ and $\mathbf{x}_i \in R^d$ denote the real data samples drawn from a probability density distribution P_r. A latent vector $\mathbf{z} \in R^m$ is pre-defined with a prior density distribution $p_{\mathbf{z}}(\mathbf{z})$; here $p_z(\mathbf{z})$ is often chosen as a multivariate normal or uniform distribution. The role of the generator $G(\mathbf{z}; \theta_G)$ is to produce samples $\mathbf{x}_{\mathrm{g}}$ with a probability density P_{g} that approximates P_{r}. Given a random sample $\mathbf{z}$, the generator maps the latent space Z to the original data space X, i.e., $G : Z \to X$. The goal is to find parameter values θ_G for the generator $G(\mathbf{z}; \theta_G)$ such that P_g is as close to P_r as possible. Discriminator $D(\mathbf{x}; \theta_D)$ takes a sample $\mathbf{x} \in X$ as the input and outputs the probability of $\mathbf{x}$ being real. Both $D(\mathbf{x}; \theta_D)$ and $G(\mathbf{z}; \theta_G)$ can update iteratively in such a way that the generator produces "fake" samples capable of "fooling" the discriminator, while the discriminator aims to distinguish the "fake" samples given by the generator from the real. The process is equivalent to a minimax two player game and eventually a Nash Equilibrium is reached.[34] The discriminator maximizes the probability of assigning the correct label to the samples from P_g and P_r; the generator is trained to minimize $\log(1 - D(G(\mathbf{z}))$. The value function $V(G, D)$ is written as:[32]

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim P_r}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim P_z(\mathbf{z})}[1 - \log D(\mathbf{G}(\mathbf{z}))] \tag{1}$$

where $G(\mathbf{z})$ is the sample produced by the generator. Because $\mathbf{x}$ is real, $D(\mathbf{x})$ will be close to 1. $D(G(\mathbf{z}))$ outputs the probability for sample $G(\mathbf{z})$. The generator expects value of $D(G((z))$ to be large for a fixed discriminator G, and therefore a minimization over G is used. As a powerful discriminator, $D(\mathbf{x})$ tends to be larger and $D(G(\mathbf{z}))$ should be smaller. Consequently, the value function $V(D, G)$ is maximized over D. In practice, $\min_G[1 - \log D(\mathbf{G}(\mathbf{z}))]$ often is replaced by $\max_G \log D(G(\mathbf{z}))$, because $1 - \log D(\mathbf{G}(\mathbf{z}))$ saturates when G is poor, and D can reject confidently the samples that are different from $\mathbf{x}$ used in the earlier training.[32]

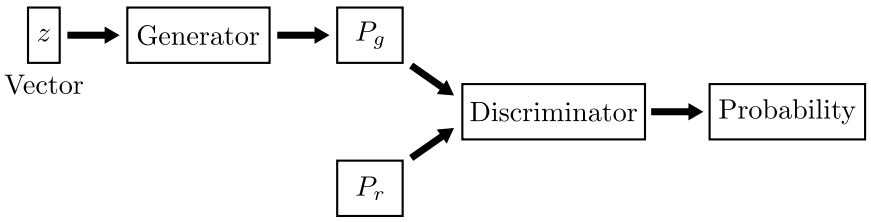


Figure 3: A basic GAN framework.

III.A. Standard GAN Shortcomings and Resolutions

While GANs have significant advantages, they are extremely difficult to train. One challenge is that the algorithm used to find the Nash equilibrium, usually a gradient-based method, may fail to converge.[35,36] GANs also suffer mode collapse, i.e., cases where the generator maps several different latent vectors to the same output sample.[29] Because the loss function given in Eqn. (1) measures the Jensen-Shannon (JS) divergence between P_r and P_g, this metric is insufficient when P_r and P_g do not overlap. Arjovsky and Bottou described that supports of P_r and P_g lying on the lower dimensional manifolds contribute to training instability.[37] Several strategies have been proposed to improve stability.[35,37,38] For instance, Deep convolutional generative adversarial networks (DCGANs) apply convolutional layers to the discriminator and deconvolutional layers to the generator.[39] The convolutional filters extract features from the previous layer, and transforms $\mathbf{x}$ to the class label (probability). The generation of the DCGAN in essence flips the direction from $\mathbf{z}$ to $\mathbf{x}$ through multi-layer filters. The DCGAN is well-known for its stability and robustness properties, and are thus used widely. Other variants of GANs are also available, such as LAPGANs,[40] AdaGANs,[41] least square GANs,[42] and loss-sensitive GANs.[43]

III.B. Wasserstein GANs (WGANs)

Wasserstein GANs (WGANs) have attracted much attention recently.[44,45] Rather than using JS divergence, the WGAN adopts Wasserstein Distance (also known as *Earth Mover's distance*) as the metric quantifying the similarity between P_r and P_g:

$$W(P_r, P_g) = \inf_{\gamma \sim \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||] \tag{2}$$

where $\Pi(P_r, Pg)$ refers to the set of all possible joint distributions. The Wasserstein distance can give a meaningful and smooth representation of the distance between P_r and P_g, even if there exist no overlaps between the lower-dimensional manifolds of both distributions. The WGANs make the training process stable and ensures diversity of the generated samples. Even multi-layer perceptions (i.e., fully-connected layers) are sufficient to produce realistic samples. Arjovsky et al. proposed a transformation for the intractable term $\inf_{\gamma\sim\Pi(P_r,P_g)}$ in Eqn. (2):[44]

$$W(P_r, P_g) = \frac{1}{K} \sup_{||f||_L \leq K} \mathbb{E}_{\mathbf{x}\sim P_r}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim P_g}[f(\mathbf{x})] \tag{3}$$

where $||f(\cdot)||_L \leq K$ is the K-Lipschitz continuous condition. If function $f(\cdot)$ is a family of K-Lipschitz continuous functions $\{f_w(\cdot)\}_{w\in W}$ parameterized by w, the loss function using Wasserstein distance can be rewritten as:[44]

$$L(P_r, P_g) = W(P_r, P_g) = \max_{w\in W} \mathbb{E}_{\mathbf{x}\sim P_r}[f_w(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim P_G}[f_w(G(\mathbf{z}))] \tag{4}$$

Here the discriminator no longer identifies the probability of a sample being real, but is instead trained to compute the Wasserstein distance by learning the K-Lipschitz continuous function $f_w(\cdot)$. The distance becomes smaller as the loss function decreases, implying that the generator is producing samples that are close to the real samples. To maintain the K-Lipschitz continuity of $f_w(\cdot)$, weight clipping, a simple but practical trick, is available: w is restricted to a compact parameter space $[-c, c]$.

WGANs do have shortcomings. Weight clipping may result in unstable training, slow convergence, and vanishing gradients. To overcome these issues, a gradient penalty has been introduced.[45] A soft version of a penalty on the gradient is included in the loss function to enforce the K-Lipschitz constraint:[45]

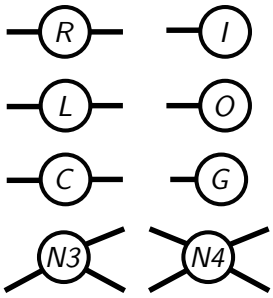
$$L(P_r, P_g) = \max_{w\in W} \mathbb{E}_{\mathbf{x}\sim P_r}[f_w(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim P_g}[f_w(G(\mathbf{z}))] + \lambda \mathbb{E}_{\mathbf{x}\sim P_{\hat{\mathbf{x}}}}\left[||\nabla_{\mathbf{x}} D(\mathbf{x})||_p - K\right]^2 \tag{5}$$

where $\hat{\mathbf{x}} = \epsilon \mathbf{x}_{P_r} + (1-\epsilon)\mathbf{x}_{P_g}$. This strategy is also called *improved training of WGANs*.[45]

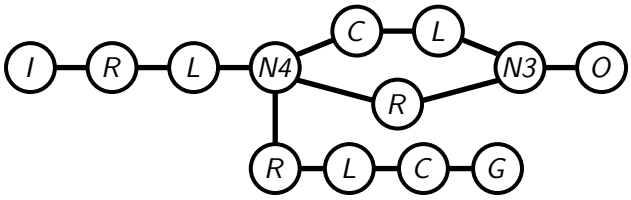
IV. Circuit Synthesis

IV.A. Problem Statement

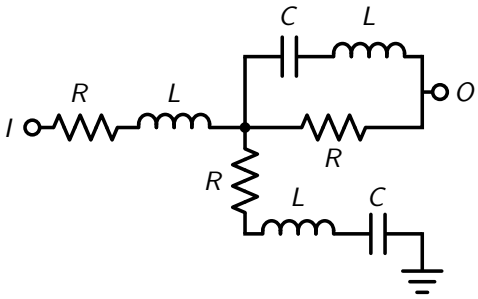
In the circuit synthesis problem presented here, each electrical circuit component is represented as a node in an undirected graph.[12] A representative set of components that could be used in a *RLC* analog electrical



(a) Some potential circuit component types.



(b) Labeled graph.



(c) Circuit schematic.

Figure 4: Examples of the heterogeneous circuit.[12]

circuit is illustrated in Fig. 4a. This illustration shows an example of an electrical circuit component catalog, including 1-port components (the input *I*, the output *O*, and the ground *G*), two-port components (resistors *R*, capacitors *C*, and inductors *L*), and multiports (> 2 ports for connecting to other components). Here the only multiports specified are common-voltage nodes (3- or 4-port common-voltage nodes *N3* and *N4*). *N5* and even higher multi-port *Nx* components may also be used in a circuit synthesis problem.

A circuit topology consists of the components used and their connectivity. Circuit topologies can be represented as labeled graphs, where edges indicate connections between component ports. Figure 4b shows the labeled graph that corresponds to the circuit presented in Fig. 4c. In this article, the adjacency matrix corresponding to labeled graphs is used as the circuit topology representation.

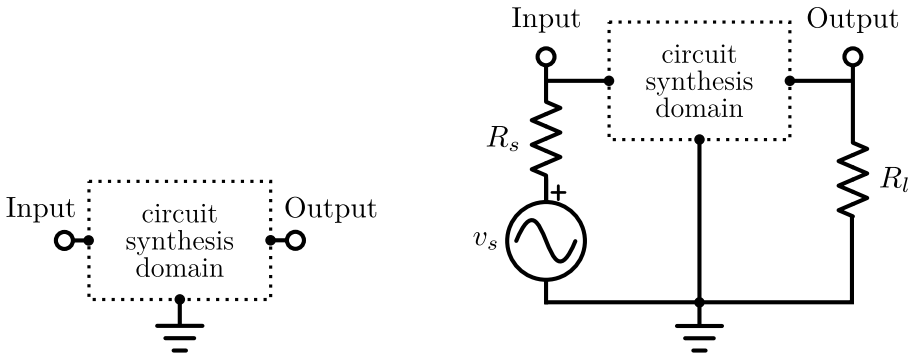
Here we consider two canonical circuit synthesis problems: 1) a frequency response matching problem, and 2) a low-pass filter realizability problem. In both of these cases, {*I*, *O*} are required. Figures 5a and 5b illustrate the relationship between these specified elements and portion of the overall system to be designed (i.e., the circuit synthesis domain). In both cases, candidate topologies must satisfy network structure constraints (NSCs) that govern what types of connections are allowed between components. Herber defined a set of vectors that specifies a desired circuit structure space,[12] as well as an efficient enumeration strategy that successfully identified all possible circuit topologies for a given component catalogs and NSCs. Although this enumeration-based strategy is feasible for larger catalogs than originally expected, it still is limited in practice to component catalogs within a certain size limit. Enumeration is important for obtaining the solution of Case 0 (all feasible and unique topologies can be enumerated and evaluated) and Case 1 synthesis problems. This has been performed in previous recent studies for both the frequency matching and filter design examples. Here we use this previously-developed enumeration and evaluation capability to generate design data needed to execute the proposed GAN strategy for solving Case 2 problems.

IV.B. Data

In the frequency response problem, a complete circuit is constructed using a transfer function between input *I* and output *O*. More desirable circuits will better satisfy the following target frequency response:[4]

$$|F(j\omega)| = \sqrt{\frac{2\pi}{10\omega}} \tag{6}$$

where the frequency (Hz) range of interest is $0.4\pi \leq \omega \leq 10\pi$ evaluated over 500 logarithmically-spaced



(a) Template circuit 1: frequency response. (b) Template circuit 2: low-pass filter.

Figure 5: Two common circuit templates.[12]

points. All circuits containing $\{R,C\}$ with up to 6 impedance elements were explored previously via the efficient enumeration strategy developed by Herber.[12] A collection of 43,249 unique circuit topologies were identified with unique transfer functions.

The low-pass filter (LPF) problem is often used as the test example for the meta-heuristic methods discussed earlier. A LPF passes inputs with a frequency lower than a determined cutoff frequency and attenuates signals that are of a higher frequency. The design task is to generate LPFs using the circuit template (Fig. 5b) under different specifications and variable bounds. A total of 1,804,496 unique graphs were found, out of which 123,156 had up to 7 components and unique transfer functions. A smaller subset also satisfies NSCs.[12] Because the size of the adjacency matrices vary as circuit topologies change, a data-processing step was used to convert the adjacency matrix to a fixed-length vector that is more amenable for machine learning strategies. Two complete data sets with all the possible circuit topologies will be used for numerical experiments.

V. Numerical Experiments

V.A. GAN Network Architecture Specifications

In this section, a comparative study is presented that quantifies how efficient various strategies are at generating circuit topologies. Different variants of GANs are trained, including GANs, WGANs, improved GANs, and DGGANs. The generator and discriminator may be chosen either as a multilayer perceptron (i.e., fully-connected layers), or a model using convolutional layers. The generator and discriminator have the same number of layers. Table 1 summarizes the various GAN architectures considered for both design problems. Here BS (batch size) is an input to the network. The predefined latent vector n_z is drawn from a standard multivariate normal distribution.

In the GAN (Tables 1a–1b), two fully-connected hidden layers are used. The upper adjacency matrix was flattened to 465×1 and 630×1 vectors, respectively, as the input for the two design problems. The output has the same size as the input. To stabilize convergence of the GANs, the ratio of network updates between the discriminator and generator was set to $5:1$. Leaky Rectified Linear Unit (LReLU)[46] and tanh are used to produce circuits in the last layer of the generator (with an additional transformation to the input as tanh requires the input to be in $[-1,1]$).

There are four fully-connected layers in the WGANs and improved WGANs (Tables 1c–1d), where the loss function is replaced by the Wasserstein distance. The sigmoid activation in the last layer of the discriminator is eliminated; the logarithm in the generator and discriminator is no longer needed; the weights in both networks are truncated in $[-0.01, 0.01]$. Finally, RMSProp is used to train the networks.[47] The regularization term λ is set to 100 for the improved WGANs. The hidden layer sizes are 128, 256, and 512, respectively.

For the DCGANs (Tables 1e–1f), the filters are assigned symmetrically. The filter with a size of 4×4 and stride of 2×2 results in a (or an approximate) scalar of 2 in each layer. In the first three layers of the discriminator, convolution, batch-normalization, and LReLU activation are connected between the layers, and the last convolutional-sigmoid layer outputs the probability. The generator has the de-convolutional, batch normalization, and LReLU activation function appended to each of the first three layers sequentially,

Table 1: Various GAN architectures considered for both design problems.

(a) GAN for frequency response problem.

Layer	Dimension
Random Tensor **z**	BS $\times$ n_z
Generator Layer 1	BS $\times$ 256
Circuit Topology	BS $\times$ 465
Discriminator Layer 1	BS $\times$ 256
Discriminator Layer 2	BS $\times$ 1

(b) GAN for low-pass filter problem.

Layer	Dimension
Random Tensor **z**	BS $\times$ n_z
Generator Layer 1	BS $\times$ 256
Circuit Topology	BS $\times$ 630
Discriminator Layer 1	BS $\times$ 256
Discriminator Layer 2	BS $\times$ 1

(c) WGAN and improved WGAN for frequency response problem.

Layer	Dimension
Random Tensor **z**	BS $\times$ n_z
Generator Layer 1	BS $\times$ 128
Generator Layer 2	BS $\times$ 256
Generator Layer 3	BS $\times$ 512
Circuit Topology	BS $\times$ 465
Discriminator Layer 1	BS $\times$ 512
Discriminator Layer 2	BS $\times$ 256
Discriminator Layer 3	BS $\times$ 128
Discriminator Layer 4	BS $\times$ 1

(d) WGAN and improved WGAN for low-pass filter problem.

Layer	Dimension
Random Tensor **z**	BS $\times$ n_z
Generator Layer 1	BS $\times$ 128
Generator Layer 2	BS $\times$ 256
Generator Layer 3	BS $\times$ 512
Circuit Topology	BS $\times$ 630
Discriminator Layer 1	BS $\times$ 512
Discriminator Layer 2	BS $\times$ 256
Discriminator Layer 3	BS $\times$ 128
Discriminator Layer 4	BS $\times$ 1

(e) DCGAN for frequency response problem.

Layer	Dimension
Random Tensor **z**	BS $\times$ 1 $\times$ 1 $\times$ n_z
Gen. De-Conv. Layer 1	BS $\times$ 4 $\times$ 4 $\times$ 128
Gen. De-Conv. Layer 2	BS $\times$ 8 $\times$ 8 $\times$ 64
Gen. De-Conv. Layer 3	BS $\times$ 16 $\times$ 16 $\times$ 32
Circuit Topology	BS $\times$ 32 $\times$ 32 $\times$ 1
Dis. Conv. Layer 4	BS $\times$ 16 $\times$ 16 $\times$ 32
Dis. Conv. Layer 4	BS $\times$ 8 $\times$ 8 $\times$ 64
Dis. Conv. Layer 4	BS $\times$ 4 $\times$ 4 $\times$ 128
Dis. Conv. Layer 4	BS $\times$ 1 $\times$ 1 $\times$ 1

(f) DCGAN for low-pass filter problem.

Layer	Dimension
Random Tensor **z**	BS $\times$ 1 $\times$ 1 $\times$ n_z
Gen. De-Conv. Layer 1	BS $\times$ 5 $\times$ 5 $\times$ 128
Gen. De-Conv. Layer 2	BS $\times$ 9 $\times$ 9 $\times$ 64
Gen. De-Conv. Layer 3	BS $\times$ 18 $\times$ 18 $\times$ 32
Circuit Topology	BS $\times$ 36 $\times$ 36 $\times$ 1
Dis. Conv. Layer 1	BS $\times$ 18 $\times$ 18 $\times$ 32
Dis. Conv. Layer 2	BS $\times$ 9 $\times$ 9 $\times$ 64
Dis. Conv. Layer 3	BS $\times$ 5 $\times$ 5 $\times$ 128
Dis. Conv. Layer 4	BS $\times$ 1 $\times$ 1 $\times$ 1

followed by the de-convolutional-tanh in the last layer. The Adaptive Moment Estimation (Adam) optimizer is used for the training.[48]

V.B. Evaluation of the GANs

Since complete data sets (denoted as Ω) are available for both circuit synthesis problems up to a certain number of components, numerical experiments were designed to test how likely a GAN is to produce a feasible circuit topology. By defintion, the complete set Ω is not available for Case 2 synthesis problems; only a representative sample X is available. In the experiments, the subset $X \subset \Omega$ is randomly selected as the training set. A new collection of samples, denoted as M, will be obtained using the generator. Suppose T out of M samples are in Ω. A metric P, defined as:

$$P = |T|/|M| \tag{7}$$

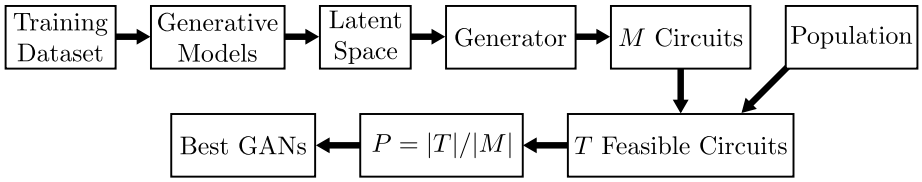


Figure 6: Procedure used to test GAN effectiveness at generating circuit topologies.

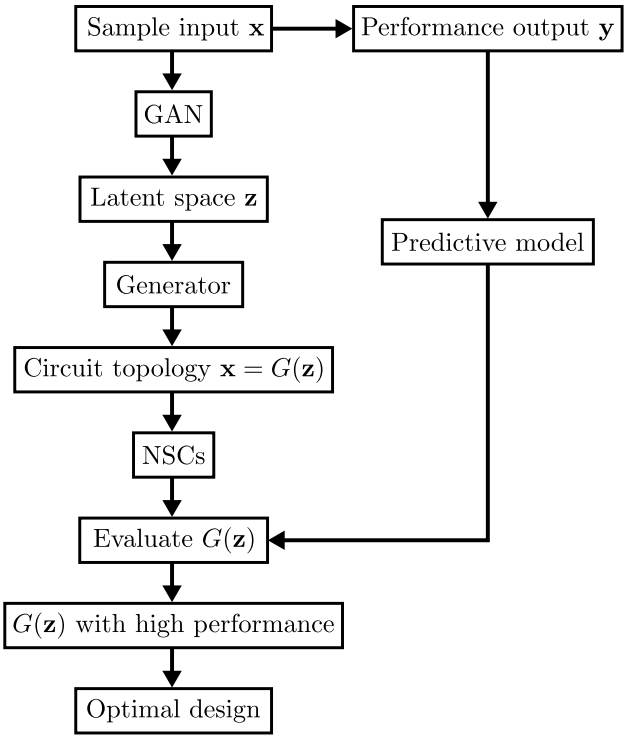


Figure 7: The GAN circuit synthesis framework using predictive modeling.

is the ratio of the number of feasible generated topologies to the total number of generated topologies. Here we make the assumption that the larger the ratio of feasible topologies to generated topologies is, the better the performance of the GAN. A perfect GAN would generate the circuit topologies satisfying $T = M \subset \Omega$ (i.e., $P = 1$). We acknowledge that P is a proxy objective function for GAN performance, but assume here that it is aligned with the objective of improving Case 2 synthesis problem solution performance. A more ideal metric would include some measurement of the coverage of the Ω that reflects how well the design method supports identification of high-performance topologies. Figure 6 illustrates the procedure used to test GAN effectiveness at topology generation.

While the GANs can generate feasible topologies, the circuit synthesis problem also requires topology evaluation and subsequent identification of high-performance circuits. Predictive modeling can be incorporated into the GAN framework to enable performance prediction for the generated circuits, a typically costly operation. One strategy for performance prediction that combines active learning (using a predictive model) with the GAN-based topology generation is depicted in Fig. 7. After circuit generation, NSCs are used to filter out any remaining infeasible topologies. The predictive model is used to approximate the performance of feasible circuits. This strategy may be particularly appropriate in cases where circuit evaluation is computationally expensive, otherwise the performance can be evaluated using the original design optimization problem for each candidate circuit topology.

VI. Numerical Results

VI.A. Comparison of GAN Models

An initial 20,000 samples drawn from Ω were used as the training sets for both design problems. The latent vector size was 64. Here we plot the adjacency matrices (symmetric) as a means to visualize circuit topologies. In the adjacency matrix visualization, lighter entries indicate connected corresponding components (value 1), whereas darker entries indicate no connection (value 0). Figure 8 shows a set of 16 randomly generated circuit topologies using the DCGAN for the frequency response problem. When there has only been a small number of iterations (Figs. 8a–8b), the generated topologies are essentially noise because generator weights are assigned randomly. However, as the training progresses, the generator has learned more about the training set, and starts to produce reasonable circuit topologies (Fig. 8d). Furthermore, the generator has the ability to produce sharper and symmetric circuit topologies by 1,500 iterations (Fig. 8f). A similar visualization for the low-pass filter problem can be found in Fig. 9.

Figures 10a and 10b illustrate how the GAN iteratively works toward feasible generated circuit topologies. A randomly selected $M = 10,000$ new data samples were obtained using the generator. Each generated circuit topology was verified to be in the complete design data set (sizes 43,249 and 123,156 for the two problems, respectively). We recorded the number of feasible topologies T for different GAN architectures and then computed the metric P. Figure 10a reports the result for the frequency response problem. It is observed that the improved WGAN performs the best with respect to P, with 1,990 out of 10,000 generated samples being feasible. The WGANs follows next, but the likelihood of the feasible topologies is decreased significantly, with only 3.44%. The GAN and DCGAN perform least well, with 1.51% and 0.36%, respectively. We specifically investigated how the training time affects the metric P. Figure 10b indicates that the likelihood of obtaining feasible samples using the improved WGAN is highly related to the number of iterations. In particular, 3,049 out of 10,000 are feasible (with iteration number 25,000). However, if we allow more iterations (i.e., 50,000), the feasibility percentage is increased to 42.66%, implying a longer training time greatly improves generation efficiency. Figure 10c shows that both the generator and discriminator losses have become stable after a sufficient number of iterations. Similar results were obtained for the low-pass filter problem, and are presented in Figs. 11a–11b.

VI.B. Parametric Studies of the Improved WGAN

Since the improved WGAN demonstrated efficient generation of feasible topologies, we conducted a parametric study regarding two important parameters: the number of latent variables and λ. The purpose of the parametric study was to provide insights about the improved WGAN's capabilities for circuit synthesis. Here the frequency response problem is considered. The number of latent variables was varied across the following levels: 16, 32, 64, and 128 (with a fixed $\lambda = 10$). The value of λ was varied across the levels: 0.1, 1, 10, and 100 (with a fixed number of latent variables: 64). For illustrative purposes, all the experiments were carried out using 5,000 iterations. Table 2a summarizes the results.

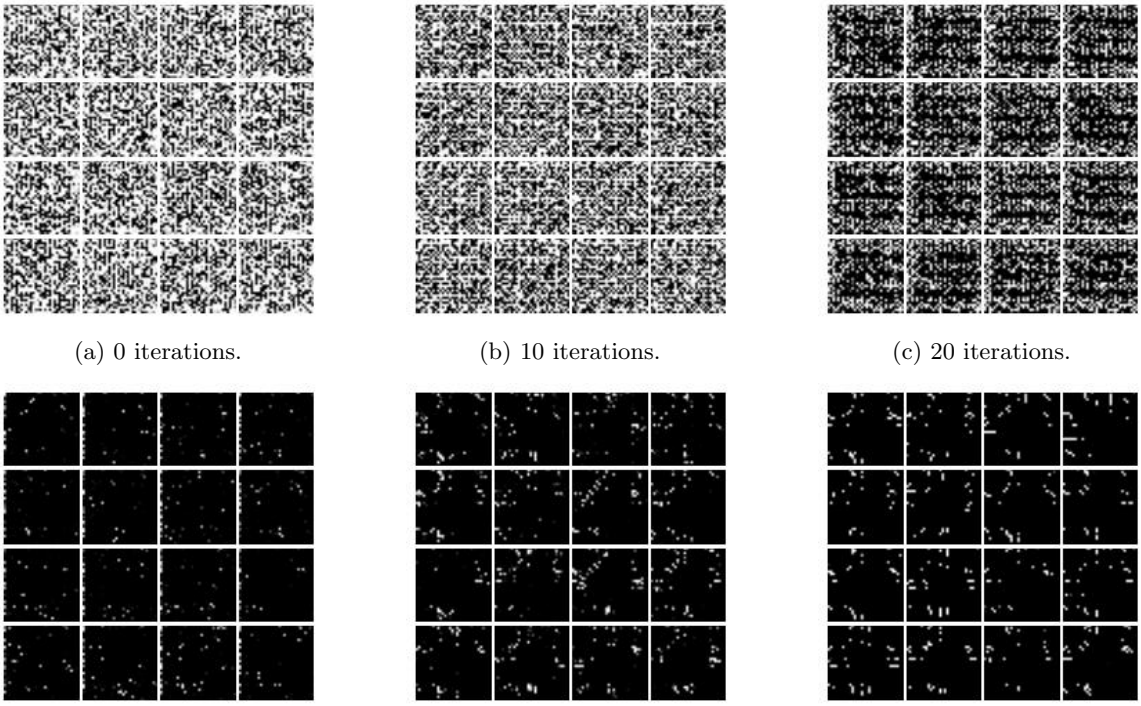
We observe that using 64 latent variables and $\lambda = 10$ significantly outperforms the other parameter values with a feasibility percentage of 19.90%. Too few latent variables (e.g., 16 and 32) result in inadequate learning of the improved WGAN, but too many latent variables (128) did not perform well, perhaps due to overfitting and more difficult training (i.e., more weights and bias must be learned). As a penalty term, the values of $\lambda = 10$ is found to be an appropriate value for enhancing the capabilities of the improved WGAN for this problem. The parametric study for the low-pass filter problem can be found in Table 2b.

VI.C. The GAN Framework with Active Learning

Here we use an active learning strategy in combination with GAN-based topology generation. The frequency response problem is tested here, where two data sets (different bounds on the resistor and capacitor coefficients) were used for the active learning tasks:[12, 16, 26]

$$\begin{aligned} &\text{(set 1)} \qquad R \in [10^{-2}, 10^{0}]\ \Omega, \qquad C \in [10^{-2}, 10^{0}]\ \text{F} \\ &\text{(set 2)} \qquad R \in [10^{-2}, 10^{5}]\ \Omega, \qquad C \in [10^{-10}, 10^{0}]\ \text{F} \end{aligned}$$

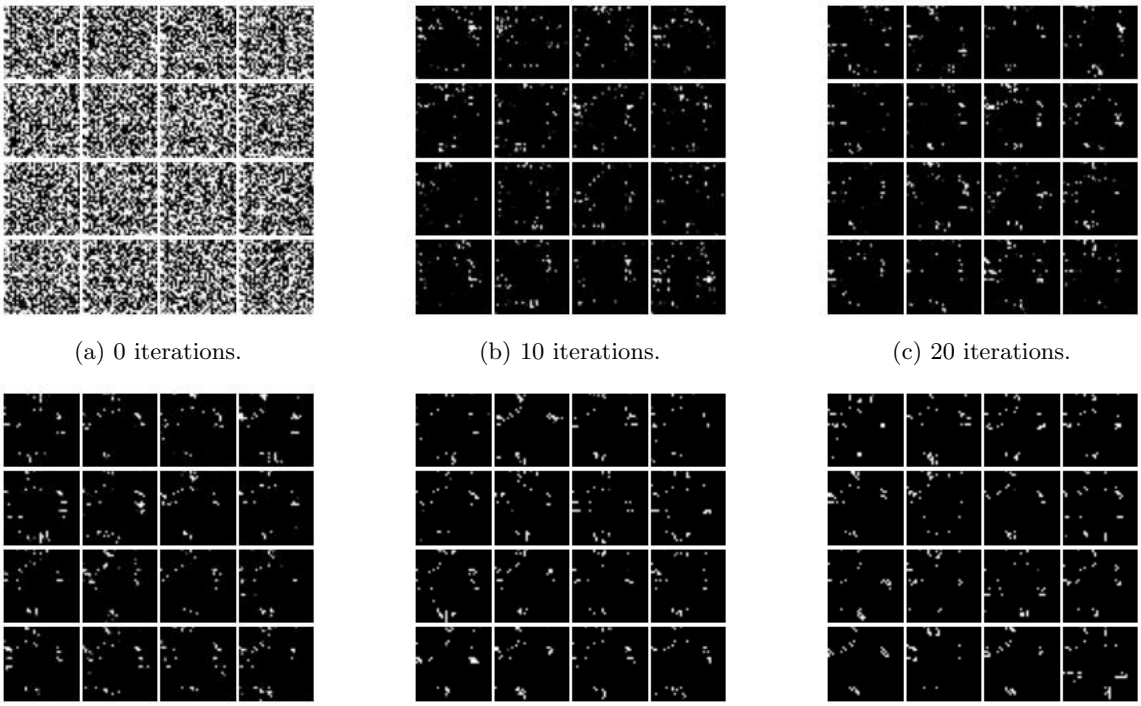
These two sets represent the common issue of design problems with different variations of the engineering constraints. Sensitivity and robustness to such problem variations is important to the development of a general learning framework for engineering synthesis problems.



(a) 0 iterations. (b) 10 iterations. (c) 20 iterations.

(d) 30 iterations. (e) 100 iterations. (f) 1500 iterations.

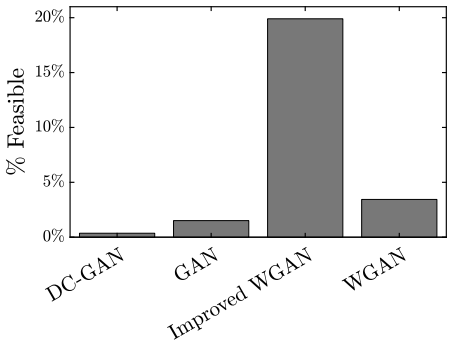
Figure 8: DCGAN progress on 16 randomly generated circuit topologies represented as adjacency matrices for the frequency response problem (a light pixel indicates a connection).



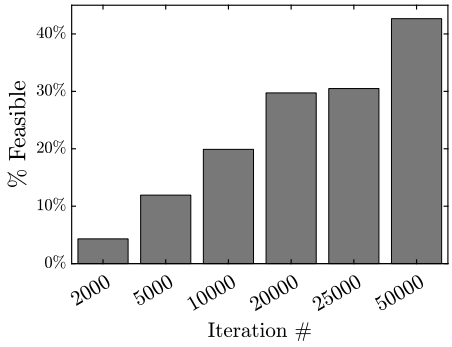
(a) 0 iterations. (b) 10 iterations. (c) 20 iterations.

(d) 30 iterations. (e) 100 iterations. (f) 1500 iterations.

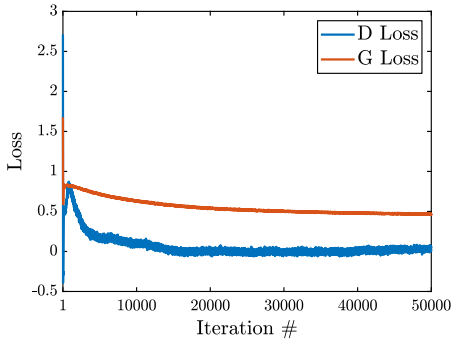
Figure 9: DCGAN progress on 16 randomly generated circuit topologies represented as adjacency matrices for the low-pass filter problem (a light pixel indicates a connection).



(a) GAN generation performance.

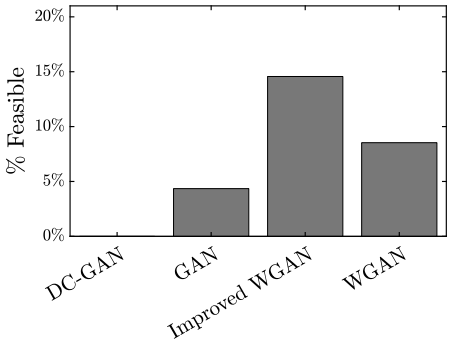


(b) Improved WGAN generation performance.

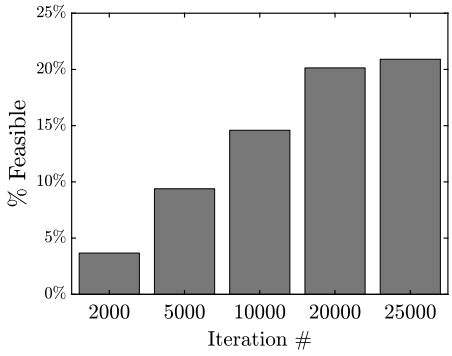


(c) Convergence of the improved WGAN.

Figure 10: Performance of the GANs for the frequency response problem.



(a) GAN generation performance.



(b) Improved WGAN generation performance.

Figure 11: Performance of the GANs for the low-pass filter problem.

Table 2: Parametric study on the parameters of the improved WGAN.

(a) Frequency response problem.

Parameter	Value	% Feasible
Latent variables	16	10.95%
	32	12.13%
	64	19.90%
	128	10.70%
λ	0.1	9.16%
	1.0	9.69%
	10.0	19.90%
	100.0	10.48%

(b) Low-pass filter problem.

Parameter	Value	% Feasible
Latent variables	16	9.18%
	32	9.06%
	64	14.55%
	128	10.26%
λ	0.1	7.03%
	1.0	8.89%
	10.0	14.55%
	100.0	9.87%

Following the procedures in Fig. 7, the numerical experiment was implemented as follows: 1) solved Case 0 problem to obtain a statistical circuit topology sample X; 2) trained an improved WGAN using sample X; 3) applied the active learning strategy to sample X to construct a predictive model; 4) used the generator to output a collection of 10,000 circuit topologies and used NSCs to filter out remaining infeasible circuit topologies; 5) the final predictive model given by the active learning strategy predicts feasible circuit topology performance. In this study, 2,973 of the 10,000 circuit topologies were feasible.

Table 3: Performance of the GAN-based method with active learning for the frequency response problem.

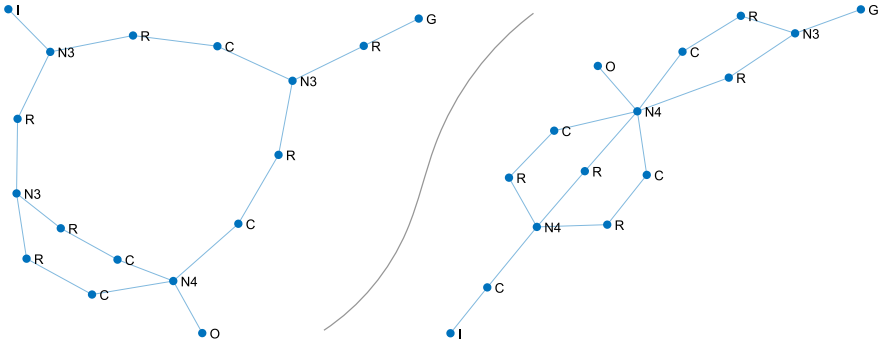
Constraint Set	RMSE	$\tilde{K}$
Set 1	2.4942	0.2631
Set 2	2.9266	0.2557

Because of the regression task, root mean square error (RMSE) was used to quantify the average performance of the predictions, while the normalized Kendall tau distance $\tilde{K}$ measures the ranking order between the predicted values and true observations. The value of $\tilde{K}$ varies in $[0, 1]$, where $\tilde{K} = 0$ indicates the ordering is exactly correct and $\tilde{K} = 0.5$ indicates the ordering is completed randomized. Table 3 summarizes the RMSE and $\tilde{K}$ for both data sets, It is interesting to observe that the RMSEs are significantly different. Different bounds on the component values may account for this sensitivity. The normalized Kendall tau distance $\tilde{K}$ shows that the majority of predictions is kept, meaning that design engineers using this technique could identify feasible circuit topologies with high-performance by simply sorting the prediction list. Figure 12 reports the generated circuits with high predicted performance for both sets. This combined framework addresses Case 2 problems with high evaluation cost, facilitating identification of high-performance circuit topologies.

VII. Discussion

The numerical results show that the improved WGAN can efficiently produce feasible circuit topologies in circuit synthesis. Here the Wasserstein distance in the loss function overcame the instability and mode collapse issues. One possible reason for the poor performance of the DCGAN could be insufficient training. Because all experiments were carried out using TensorFlow CPU implementations, DCGAN required a significant amount of time to complete the training. GPU computing could speed up the training process, enabling the generation of more robust results. However, it is still observed that even the improved WGANs with fully-connected layers (simpler network architecture) outperforms the DCGAN with convolutional layers (complicated network architecture). It might be helpful to implement a deep convolutional WGAN (DCWGAN) architecture to gain more insights about the generative design.

While this work demonstrates initial promising results for the use of GANs in engineering synthesis,



(a) Set 1.

(b) Set 2.

Figure 12: Some high-performance circuit topologies given by the improved WGAN for the frequency response example.

a number of improvements and further investigations should be performed. A number of factors affect GAN performance. For example, four hidden layers in the GAN were initially implemented. However, it turned out that the GAN experienced instability and mode collapse with this architecture. Replacing two hidden layers improved performance (see Tables 1a and 1b). Further investigations could be conducted to better understand the randomly-generated sample produced by the GAN-based method. In the numerical experiment, there could be issues with graph isomorphisms[11] because each generated circuit topologies used a fixed-size adjacency matrix. In other words, the portion of the circuit identified as infeasible may be isomorphic to another feasible circuit in the complete data set. This could significantly reduce the potential effectiveness of this GAN design representation. Recall that the improved WGAN achieved as high as 42.66% feasibility percentage for the frequency response problem; the isomorphism check would potentially increase metric P further. On the other hand, it is also worth exploring properties of the randomly generated circuit topologies, including uniqueness, component combination, frequency, and other aspects.

The GAN-based method could be extended to other research tasks in circuit synthesis. The numerical results imply that the improved WGAN could be a good option for generating high-quality circuit topologies. Once the circuit topology $\mathbf{x}$ is available, the circuit performance must be assessed via sizing optimization. Figure 13 illustrates a possible workflow where the GAN framework can incorporated with sizing optimization. Additional detail regarding sizing optimization can be found in Refs.[12,49]

The GAN-based methodology could be applied to other heterogeneous system topology design problems with similar properties. This assertion is based on recent work in enumeration and evaluation of various automotive suspension architectures, where different combinations of passive and active components are combined to improve comfort and handling metrics.[49] Similar to the circuit synthesis in this article, each suspension component is regarded as a node, and the system topology can be represented in terms of an adjacency matrix. The performance of each suspension is optimized with respect to a complex dynamic optimization problem.[49] These properties would support GAN training and subsequent generation of feasible suspension topologies. We anticipate that this approach could extend to other synthesis problems with these properties.

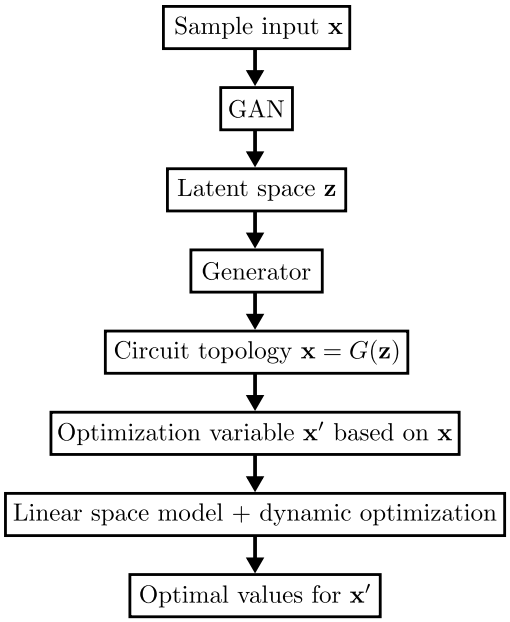


Figure 13: Proposed framework using the GAN in synthesis problems with optimization problems defined for different topologies.

VIII. Conclusion

In this article, adversarial learning was presented as a new solution strategy for Case 2 heterogeneous synthesis design problems (synthesis problems where it is not only impractical to evaluate all topologies of interest, but it is also impractical to enumerate them). The generative adversarial network (GAN) is a generative model that learns to simulate a data distribution. The GAN is trained iteratively such that the generator produces circuit topologies that are accepted by the discriminator. A lower-dimensional latent space can represent the circuit topology indirectly. A random sample drawn from a normal distribution is mapped to a circuit topology via the generator network.

We studied the efficiency of generating feasible samples using different GAN models, including GAN, DCGAN, WGAN, and improved WGAN. The frequency response and low-pass filter problems, containing complete topological design data up to a certain number of components, were used as the case studies. The numerical results indicate that the generator of the improved WGAN has a higher likelihood to generate feasible circuit topologies. A parametric study was carried out to identify the best parameter values for the improved WGAN. The GAN framework combined with active learning made predictions for generated circuit performance, and the resulting normalized Kendall tau distance indicates that the performance ranking is similar. This is a promising approach for searching for high-performance circuit topologies, as well as the efficacy of the method to other Case 2 synthesis problems. This work leverages design data and artificial intelligence, not only providing an indirect representation that generates the feasible circuit topologies in an *automated* manner, but also may be transferable, as the pretrained discriminator may facilitate the development of the circuit property prediction model.

Future work for improving the GAN capabilities includes, but is not limited to, the use of both the DCGAN and improved WGAN, other generative models such as VAEs,[50] PixelCNNs,[51] and PixelRNNs.[52] One option to extend the functionality of the GAN framework would be to use it to support design optimization. An example of another engineering synthesis problem is active vehicle suspension design and would be a very interesting case study to demonstrate the value of the proposed GAN-based synthesis methodology.

References

[1]Mitea, O., Meissner, M., Hedrich, L., and Jores, P., "Automated Constraint-Driven Topology Synthesis for Analog Circuits," *Design, Automation & Test in Europe*, Grenoble, France, March 2011, pp. 1–4, doi: 10.1109/DATE.2011.5763264

[2]Gan, Z., Yang, Z., Shang, T., Yu, T., and Jiang, M., "Automated Synthesis of Passive Analog Filters Using Graph Representation," *Expert Systems with Applications*, Vol. 37, No. 3, March 2010, pp. 1887–1898, doi: 10.1016/j.eswa.2009.07.013

[3]Sussman, G. and Stallman, R., "Heuristic Techniques in Computer-Aided Circuit Analysis," *IEEE Transactions on Circuits and Systems*, Vol. 22, No. 11, Nov. 1975, pp. 857–865, doi: 10.1109/TCS.1975.1083985

[4]Grimbleby, J. B., "Automatic Analogue Network Synthesis Using Genetic Algorithms," *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, Sheffield, UK, Sept. 1995, pp. 53–58, doi: 10.1049/cp:19951024

[5]Das, A. and Vemuri, R., "An Automated Passive Analog Circuit Synthesis Framework Using Genetic Algorithms," *IEEE Computer Society Annual Symposium on VLSI*, Porto Alegre, Brazil, March 2007, pp. 145–152, doi: 10.1109/ISVLSI.2007.22

[6]Das, A. and Vemuri, R., "Topology Synthesis of Analog Circuits Based on Adaptively Generated Building Blocks," *45th ACM/IEEE Design Automation Conference*, Anaheim, CA, USA, June 2008, pp. 44–49.

[7]Lohn, J. D. and Colombano, S. P., "A Circuit Representation Technique for Automated Circuit Design," *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 3, Sept. 1999, pp. 205–219, doi: 10.1109/4235.788491

[8]Goh, C. and Li, Y., "GA Automated Design and Synthesis of Analog Circuits with Practical Constraints," *Proceedings of the 2001 Congress on Evolutionary Computation*, Seoul, South Korea, South Korea, May 2001, pp. 170–177, doi: 10.1109/CEC.2001.934386

[9]Koza, J. R., Bennett, F. H., Andre, D., Keane, M. A., and Dunlap, F., "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming," *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 2, July 1997, pp. 109–128, doi: 10.1109/4235.687879

[10]Ochotta, E. S., Rutenbar, R. A., and Carley, L. R., "Synthesis of High-Performance Analog Circuits in ASTRX/OBLX," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 3, March 1996, pp. 273–294, doi: 10.1109/43.489099

[11]Herber, D. R., Guo, T., and Allison, J. T., "Enumeration of Architectures with Perfect Matchings," *ASME Journal of Mechanical Design*, Vol. 139, No. 5, May 2017, pp. 051403, doi: 10.1115/1.4036132

[12]Herber, D. R., *Advances in Combined Architecture, Plant, and Control Design*, Ph.D. Dissertation, University of Illinois at Urbana-Champaign, 2017.

[13]Cheney, N., MacCurdy, R., Clune, J., and Lipson, H., "Unshackling Evolution: Evolving Soft Robots with Multiple Materials and a Powerful Generative Encoding," *Annual Conference on Genetic and Evolutionary Computation*, Amsterdam, The Netherlands, July 2013, pp. 167–174, doi: 10.1145/2463372.2463404

[14]Khetan, A., Lohan, D. J., and Allison, J. T., "Managing Variable-Dimension Structural Optimization Problems Using Generative Algorithms," *Structural and Multidisciplinary Optimization*, Vol. 50, No. 4, Oct. 2015, pp. 695–715, doi: 10.1007/s00158-015-1262-8

[15]Lohan, D. J., Dede, E. M., and Allison, J. T., "Topology Optimization for Heat Conduction Using Generative Design Algorithms," *Structural and Multidisciplinary Optimization*, Vol. 55, No. 3, March 2017, pp. 1063–1077, doi: 10.1007/s00158-016-1563-6

[16]Guo, T., Lohan, D. J., Cang, R., Ren, M. Y., and Allison, J. T., "An Indirect Design Representation for Topology Optimization Using Variational Autoencoder and Style Transfer," *AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, No. AIAA 2018-0804, Kissimmee, FL, USA, Jan. 2018, doi: 10.2514/6.2018-0804

[17]Foster, R. M., "Geometrical Circuits of Electrical Networks," *Electrical Engineering*, Vol. 51, No. 1, Jan. 1932, pp. 43, doi: 10.1109/EE.1932.6429606

[18]Bruccoleri, F., Klumperink, E. A., and Nauta, B., "Generating All Two-MOS-Transistor Amplifiers Leads to New Wide-Band LNAs," *IEEE Journal of Solid-State Circuits*, Vol. 36, No. 7, July 2001, pp. 1032–1040, doi: 10.1109/4.933458

[19]Bayrak, A. E., Ren, Y., and Papalambros, P. Y., "Topology Generation for Hybrid Electric Vehicle Architecture Design," *ASME Journal of Mechanical Design*, Vol. 138, No. 8, Aug. 2016, pp. 081401, doi: 10.1115/1.4033656

[20]Del Castillo, J. M., "Enumeration of 1-DOF Planetary Gear Train Graphs Based on Functional Constraints," *ASME Journal of Mechanical Design*, Vol. 124, No. 4, Dec. 2002, pp. 723–732, doi: 10.1115/1.1514663

[21]Ma, W., Trusina, A., El-Samad, H., Lim, W. A., and Tang, C., "Defining Network Topologies That Can Achieve Biochemical Adaptation," *Cell*, Vol. 138, No. 4, Aug. 2009, pp. 760–773, doi: 10.1016/j.cell.2009.06.013

[22]Guo, T. and Allison, J. T., "On the Use of Mathematical Programs with Complementary Constraints in Combined Topological and Parametric Design of Biochemical Enzyme Networks," *Engineering Optimization*, Vol. 49, No. 2, Feb. 2017, pp. 345–364, doi: 10.1080/0305215X.2016.1188091

[23]Radhakrishnan, P. and Campbell, M. I., "A Graph Grammar Based Scheme for Generating and Evaluating Planar Mechanisms," *Design Computing and Cognition*, Stuttgart, Germany, July 2011, pp. 663–679, doi: 10.1007/978-94-007-0510-4_35

[24]Campbell, M. I., Rai, R., and Kurtoglu, T., "A Stochastic Graph Grammar Algorithm for Interactive Search," *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, No. DETC2009-86804, San Diego, California, USA, Sept. 2009, pp. 829–840, doi: 10.1115/DETC2009-86804

[25]Kurtoglu, T. and Campbell, M. I., "An Evaluation Scheme for Assessing the Worth of Automatically Generated Design Alternatives," *Research in Engineering Design*, Vol. 20, No. 1, March 2009, pp. 59–76, doi: 10.1007/s00163-008-0062-1

[26]Guo, T., Herber, D. R., and Allison, J. T., "Reducing Evaluation Cost for Circuit Synthesis Using Active Learning," *ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, No. DETC2018-85654, Quebec City, Canada, Aug. 2018, p. V02AT03A011, doi: 10.1115/DETC2018-85654

[27]Cang, R., Li, H., Yao, H., Jiao, Y., and Ren, Y., "Improving Direct Physical Properties Prediction of Heterogeneous Materials from Imaging Data via Convolutional Neural Network and a Morphology-Aware Generative Model," arXiv Preprint, 2017, arXiv:1712.03811

[28]Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A., "Adversarially Learned Inference," arXiv Preprint, 2016, arXiv:1606.00704

[29]Goodfellow, I., "NIPS 2016 Tutorial: Generative Adversarial Networks," arXiv Preprint, 2016, arXiv:1701.00160

[30]Radford, A., Metz, L., and Chintala, S., "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," arXiv Preprint, 2015, arXiv:1511.06434

[31]Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J., "Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling," *Advances in Neural Information Processing Systems*, Barcelona, Spain, Dec. 2016, pp. 82–90.

[32]Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., "Generative Adversarial Nets," *Advances in Neural Information Processing Systems*, Montral, Canada, Dec. 2014, pp. 2672–2680.

[33]Luan, S., Arora, M., Thurston, D. L., and Allison, J. T., "Alternative Design Optimization Formulations—Developing and Comparing for a Vibration Damping Example," *ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, No. DETC2017-68337, Cleveland, OH, USA, Aug. 2017, p. V004T05A004, doi: 10.1115/DETC2017-68337

[34]Osborne, M. J. and Rubinstein, A., *A Course in Game Theory*, The MIT Press, 1994.

[35]Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X., "Improved Techniques for Training GANs," *Advances in Neural Information Processing Systems*, Barcelona, Spain, Dec. 2016, pp. 2234–2242.

[36]Goodfellow, I. J., "On Distinguishability Criteria for Estimating Generative Models," arXiv Preprint, 2014, arXiv:1412.6515

[37]Arjovsky, M. and Bottou, L., "Towards Principled Methods for Training Generative Adversarial Networks," arXiv Preprint, 2017, arXiv:1701.04862

[38]Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J., "Unrolled Generative Adversarial Networks," arXiv Preprint, 2016, arXiv:1611.02163

[39]Ioffe, S. and Szegedy, C., "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," arXiv Preprint, 2015, arXiv:1502.03167

[40]Denton, E. L., Chintala, S., Szlam, A., and Fergus, R., "Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks," *Advances in Neural Information Processing Systems*, Vol. Montral, Cana, Montral, Canada, Dec. 2015, pp. 1486–1494.

[41]Tolstikhin, I. O., Gelly, S., Bousquet, O., Simon-Gabriel, C.-J., and Schölkopf, B., "AdaGAN: Boosting Generative Models," *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, Dec. 2017, pp. 5430–5439.

[42]Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., and Smolley, S. P., "Least Squares Generative Adversarial Networks," *IEEE International Conference on Computer Vision*, Venice, Italy, Oct. 2017, pp. 2813–2821, doi: 10.1109/ICCV.2017.304

[43]Qi, G.-J., "Loss-Sensitive Generative Adversarial Networks on Lipschitz Densities," arXiv Preprint, 2017, arXiv:1701.06264

[44]Arjovsky, M., Chintala, S., and Bottou, L., "Wasserstein GAN," arXiv Preprint, 2017, arXiv:1701.07875

[45]Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C., "Improved Training of Wasserstein GANs," *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, Dec. 2017, pp. 5769–5779.

[46]Maas, A. L., Hannun, A. Y., and Ng, A. Y., "Rectifier Nonlinearities Improve Neural Network Acoustic Models," *International Conference on Machine Learning*, Atlanta, GA, USA, June 2013.

[47]Hinton, G., Srivastava, N., and Swersky, K., "Neural Networks for Machine Learning: Lecture 6a Overview of Mini-Batch Gradient Descent," Online.

[48]Kingma, D. P. and Ba, J., "Adam: A Method for Stochastic Optimization," arXiv Preprint, 2014, arXiv:1412.6980

[49]Herber, D. R. and Allison, J. T., "A Problem Class with Combined Architecture, Plant, and Control Design Applied to Vehicle Suspensions," *ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, No. DETC2018-86213, Quebec City, Canada, Aug. 2018, p. V02AT03A006, doi: 10.1115/DETC2018-86213

[50]Kingma, D. P., Mohamed, S., Jimenez Rezende, D., and Welling, M., "Semi-Supervised Learning with Deep Generative Models," *Advances in Neural Information Processing Systems*, Montral, Canada, Dec. 2014, pp. 3581–3589.

[51]van der Aalst, W., *Process Mining: Data Science in Action*, Springer, 2016, doi: 10.1007/978-3-662-49851-4

[52]Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K., "Pixel Recurrent Neural Networks," arXiv Preprint, 2016, arXiv:1601.06759