Pretraining Graph Neural Networks for Few-Shot Analog Circuit Modeling and Design

Kourosh Hakhamaneshi, Marcel Nassar, Member, IEEE, Mariano Phielipp, Pieter Abbeel, Fellow, IEEE, and Vladimir Stojanovic, Senior Member, IEEE

Abstract—Being able to predict the performance of circuits without running expensive simulations is a desired capability that can catalyze automated design. In this article, we present a supervised pretraining approach to learn circuit representations that can be adapted to new circuit topologies or unseen prediction tasks. We hypothesize that if we train a neural network (NN) that can predict the output direct current (dc) voltages of a wide range of circuit instances it will be forced to learn generalizable knowledge about the role of each circuit element and how they interact with each other. The dataset for this supervised learning objective can be easily collected at scale since the required dc simulation to get ground truth labels is relatively cheap. This representation would then be helpful for few-shot generalization to unseen circuit metrics that require more time-consuming simulations for obtaining the ground-truth labels. To cope with the variable topological structure of different circuits we describe each circuit as a graph and use graph NNs (GNNs) to learn node embeddings. We show that pretraining GNNs on prediction of output node voltages can encourage learning representations that can be adapted to new unseen topologies or prediction of new circuit-level properties with up to 10x more sample efficiency compared to a randomly initialized model. We further show that we can improve the sample efficiency of prior SoTA model-based optimization methods by 2x (almost as good as using an oracle model) via finetuning pretrained GNNs as the feature extractor of the learned models.

Index Terms—Circuit design, graph neural networks (GNNs), knowledge transfer, machine learning, pretraining.

I. INTRODUCTION

T HE SEMICONDUCTOR industry largely owes its rapid progress in the last century to the scaling of silicon process technology, which has consistently allowed designers to fit more processing power into the same area footprint and make today’s digital processing capabilities a reality. However, such highly specialized designs often come at a high redesign cost with every new integrated circuits (ICs) technology. This is often due to the change of low-level transistor behaviors and more stringent design rules imposed by technology scaling. Recent research efforts have focused on shortening the production cycle by leveraging the advances made in machine learning in several aspects of the manual design process, including high-level synthesis [1], logic synthesis [2], [3], placement and routing [4], [5], [6], analog mixed signal layout exploration [7], [8], [9], [10], manufacturing [11], [12], and others.1

The main obstacle for large-scale adoption of ML-based methods to circuit design is the difficulty in creating large datasets partly due to expensive simulation runtimes. For example, the simulation of one instance of a digital to analog converter (DAC) in [14] could take ~30 min and this time can become as much as a few days for more complicated circuits. To mitigate this issue, prior work has explored applying adaptive sampling methods, such as reinforcement learning [8], Bayesian optimization [15], [16], and evolutionary algorithms [9] for optimization of circuits. However, the models or policies that are learned via these methods are highly tailored to the environment and objective of the optimization and cannot be reused for new specifications, topologies, or more advanced technologies. Ideally we want to be able to reuse the representation learned from one problem to solve another problem faster, similar to how humans reuse their prior experience to design new circuits.

In this work, we focus on modeling analog circuits and their generalization to unseen circuit topologies. Such a generalizable model can help boost the performance of model-based optimization methods in terms of sample efficiency and performance. Our main research question is How can we learn circuit representations that can be reused for modeling unseen topologies or predicting new circuit-level labels via finetuning? We hypothesize that if we train a neural network (NN) that can predict all output voltage nodes in a circuit, some domain knowledge will be encoded in the representation that can also be reused for downstream scenarios. Predicting node voltages in a circuit requires an in-depth understanding of the static behavior of each element and their interaction with one another. At the same time, the simulation required for computing the ground-truth output direct current (dc) voltages is

1For a comprehensive survey on using machine learning in electronic design automation see [13].
a lot cheaper than getting labels for other performance metrics (e.g., settling time). Also knowing node dc voltage values, a.k.a. the bias of the circuit, can simplify deriving its behavior and performance. To be able to handle the variable input structure of the model we represent each circuit instance with a graph and use graph NNs (GNNs) as the backbone of the model. This allows us to improve generalization by using the structural information in circuits as an input feature for prediction.

We first test this hypothesis with a set of controlled experiments on simple circuits that include only resistors and voltage sources, known as resistor ladders. This class of circuits despite being simple from an analytical point of view, have a special property that any change to any part of the circuit (e.g., changing voltage source or resistor values or adding or removing extra elements) can translate into a very different output response on node voltages. In other words, no node is isolated from another one, making their interaction really challenging for NNs to model. We show that a GNN backbone pretrained with the voltage prediction task on a subset of this circuit family can be fine-tuned to generalize its prediction to new unseen circuits with different topologies. We also illustrate how the node representation can be used for new graph property prediction tasks such as prediction of the output resistance of the circuit. Furthermore, we provide more empirical evidence on a more complicated real-world transistor-based circuit and contrast the sample efficiency of our pretrained models to training from scratch. We then show that such pretrained GNN model can be reused as the feature extractor in an optimization loop that automates the design of such circuit, and improve the optimization sample efficiency by at least 2× (almost on par with the oracle model) compared to previous state-of-the-art that uses simple fully connected architectures trained from scratch.

We summarize our contribution as follows.

1) We present an enhanced representation for describing circuits as graphs that is naturally more tailored to transistor-based topologies compared to prior work on using GNNs on analog circuits [17], [18].

2) We show that node voltage prediction can serve as an effective pretraining task for learning features transferable to new topologies in both zero and few-shot manner.

3) We further show that such a representation can also be reused for graph prediction tasks via fine-tuning on small datasets and improve sample efficiency of model-based optimization algorithms.

4) We present the first open-sourced graph dataset and benchmark for circuit voltage prediction that can help standardize the progress of this research direction.

II. RELATED WORK

*Modeling Circuits Using Deep Neural Networks:* Functional modeling of analog circuits using deep NNs has been revisited in recent years due to the success of deep learning in modeling black-box functions. To this end, a conventional modeling approach is to model the entire circuit end-to-end with multilayer perceptrons [19], [20]. However, this choice of architecture requires a large training dataset for building accurate models which can be problematic when a simulation is time consuming. One way to address this issue has been to impose domain-specific inductive biases on model architectures. For example, Hassanpourghadi et al. [14] proposed breaking the circuit into subcircuits and modeling each with a separate NN in an auto-regressive fashion. However, it requires ad-hoc human input on the definition of subcircuits. Also, the subcircuits themselves are modeled with MLPs and do not leverage structural information. In this work we utilize GNNs, which generalize of the idea of imposing structural inductive bias on the architecture.

*GNNs in Chip Design:* In recent years, the idea of using GNNs to automate the design process of systems on a chip have been visited in several prior works from both an optimization and modeling perspective. Most notably [4] studies the problem of chip placement and utilizes GNNs as the backbone of the feature embeddings of an RL policy and value function that is learned in an end-to-end fashion using custom reward functions. The reason behind using GNNs as feature extractors is that many chip placement strategies are motivated by the *global* structure of the system and GNNs can capture the structural dependencies effectively. Reference [17], on the other hand, uses GNNs for circuit parasitic prediction based on circuit’s graph which only needs *local* information. Liu et al. [21] and Zhang et al. [22] employed GNNs as the backbone of the predictive models for optimizing analog circuit metrics. However, the benefits of using GNNs in these works are limited to a single topology or a single optimization objective. In our work, we study pretraining GNNs to see whether we can reliably transfer domain knowledge to new topologies or prediction tasks instead of learning from scratch.

III. LEARNING REPRESENTATIONS FOR ANALOG CIRCUITS

We will first present the language for describing analog circuits as graphs in Section III-A. In Sections III-B and III-C, we describe how we can use a supervised node property prediction task (i.e., predicting voltages of each node) as a pretraining objective to learn representations that can be fine-tuned for predicting voltages on unseen topologies or for new graph property prediction tasks (e.g., predicting the output resistance of the circuit). We finally conclude by explaining our evaluation protocol in Section III-D.

A. New Graph Representation

Despite the recent surge of interest in utilizing GNNs for predictive models on circuits [17], [18], there is still no unified descriptive language for representing transistor-based circuits as graphs that can uniquely represent all possible topological configurations. For example, Ren et al. [17] proposed to map each device and each net (i.e., wire connection) in the circuit to a graph node and uses edge types to distinguish the device terminal types (i.e., source, drain, gate, and body). This representation becomes ambiguous in cases where a device (e.g., transistor) has the same net connection between two or more...
Fig. 1. (a) Example of why it is important to explicitly model device terminals as nodes to disambiguate certain topological configurations. (b) Steps of converting a circuit schematic to a graph representation. 1) Convert all devices to their corresponding complete subgraphs. 2) Convert all circuit nets to (yellow) nodes in the graph. 3) Make edge connections according to the topology of the circuit.

Fig. 2. Pretraining architecture: After the GNN backbone, the batch of output node features (yellow nodes) are fed into an MLP to predict the output voltages of each node. The same MLP is applied to all the node features. The gradient of the MSE between ground truth and prediction is then backpropagated to update the parameters.

terminals, Fig. 1(a) illustrates such an example. Since the drain (red) and gate (green) terminals of the transistor are both connected to net $N$, two edges with different types should connect the device $T$ to its corresponding net $N$. The source (blue) and body (purple) terminals also have the same ambiguity.

To disambiguate this representation, we propose a set of steps that explicitly represent the device terminals as individual nodes. This representation is closely linked to the widely used textual description (i.e., netlist) of circuits used for simulator tools such as SPICE [23]. In this representation, each device is a complete subgraph with its terminals as individual nodes. The features of each node indicate the terminal type and device parameters and therefore, all edges become feature-less. Fig. 1(b) illustrates an example of how a simple amplifier circuit gets mapped to its graph representation, step by step following our approach.

1) Convert the terminals of each device to a complete subgraph with the same number of nodes as the number of terminals.

2) Convert each circuit net to a node in the graph.

3) Connect all the nets to their corresponding device terminals.

The resulting representation is a graph $G = (V, E)$ comprised of a set of nodes $V$ and a set of undirected edges $E$. Each node is associated with a feature vector which describes the type of the node as well as the device parameters that the node belongs to (e.g., the value of a resistor). There are no features associated with edges, and they only preserve the connectivity information of the elements of the circuit.

B. Voltage Prediction as the Pretraining Task

One of the major steps for analyzing and synthesizing circuits for humans is to compute the bias first. This entails computing the dc voltage of various important nodes in the circuit that sets the working condition of all the devices. One can think of computing voltages as deriving the operating point and linearizing a nonlinear system which is a prerequisite step for computing other system metrics.

Inspired by this causal abstraction, we hypothesize that if a NN is trained to predict the dc voltages of circuit nodes in various contexts it should be able to learn critical system-specific knowledge that can be reused for other predictions, either in zero-shot or few-shot manner via fine tuning. In other words, the pretrained backbone on the voltage prediction task should provide a better initialization for unseen tasks and topologies than starting from a random initialization. Moreover, getting dc operating points of circuits through simulation is relatively inexpensive compared to simulating other circuit properties. Therefore, in principle, the pretraining dataset can be collected at scale through simulation in a reasonable time frame.

Fig. 2 illustrates the pretraining architecture. We cascade a GNN backbone with an MLP with shared parameters across nodes which predicts the output voltage based on the node
Fig. 3. Node to graph prediction pipeline: After the GNN backbone, the node features are translated to a single vector representing the graph embedding. The graph embedding is then fed to an MLP to predict the output. The node to graph embedding is three cross-attention layers between a learned embedding and the node features (similar to [24]).

contextualized representations. The pretraining objective is minimizing the mean square error (MSE) loss between the predicted output and the ground truth for the output nodes. There is no supervision signal on the representation of other nonoutput nodes. After pretraining, we can use the output node features from the GNN backbone for the prediction of new tasks or new topologies.

C. From Node Embedding to Graph Property Prediction

To perform a graph property prediction task, we need to combine the node embeddings into a single graph embedding.

A naive way is to aggregate all the node embeddings via pooling operations (e.g., mean pooling) and then feed it to an MLP that performs the graph-level prediction. However, this approach can result in loss of information during aggregation and decreased capacity of the NN.

To mitigate this problem, we propose a learnable attention-based aggregation layer that scales linearly with the number of output nodes. By learning the aggregation module, the model can choose the optimal way of aggregating the embeddings into a graph representation based on the dataset.

Fig. 3 demonstrates this architecture for fine tuning on a graph property prediction task. Once we have the contextualized node embeddings at the output of the GNN, we flatten them as a set of feature vectors and pass them through a cross-attention network. In this attention, the module computes the cross-attention between a set of learned latent vectors and the output node features to embed them into a single graph representation. The latent vectors are represented as a 16×16 array that is learned during training to minimize the output MSE loss. The contextualized 16×16 array of latent vectors is then flattened to a single vector of size 256 at the output to represent the graph embedding.

This architecture is inspired by the transformer read-out layer proposed in [25] but modified to be similar to perceiver IO’s cross-attention architecture [24] to gracefully scale computation and memory with the number of nodes.²

D. Evaluation Metric

The prediction tasks presented in this article are all regression tasks that require minimizing an MSE loss of the predicted values w.r.t. the ground truth. MSE however, is notoriously sensitive to outliers and sensitive to the range of the predicted output. Therefore, to get a better quantitative measure of performance, we propose measuring the accuracy of correct predictions within a certain resolution.

For a given dataset \( \{x_i, y_i\}_{i=1}^N \), let \( \hat{y}_i \) be the predictions of the model. We define \( \text{Acc@K} \) as follows:

\[
\text{Acc@K} = \frac{1}{N} \sum_{j=1}^{N} 1 \left( \left| y_j - \hat{y}_j \right| \leq \frac{\max_i y_i - \min_i y_i}{K} \right).
\]

This metric, measures the ratio of predictions that are within a certain precision w.r.t to their ground truth value. For instance in the voltage prediction task (with a range of 1 V), \( K = 100 \) sets the precision to 10 mV, i.e., 1% of the range.

IV. EXPERIMENTS

We perform an extensive empirical study to validate our hypothesis by focusing on the following questions.

1) Can we successfully pretrain GNN architectures to regress the dc voltage of output nodes on a variety of circuits?

2) How does a pretrained network perform on predicting the output voltage nodes of an unseen circuit topology that is slightly different than what it has been pretrained on?

3) Can we fine-tune the pretrained backbone to improve voltage prediction generalization on new unseen circuits?

4) Can we reuse the learned representations after pretraining, to efficiently learn to predict new unseen circuit-level tasks (e.g., a design metric of a circuit) via fine-tuning?

5) Can we improve optimization sample efficiency of state-of-the-art model-based circuit optimization tasks (e.g., [9])?

A. Dataset and Circuits Under Study

We perform our experiments on two sets of circuits. First one is a family of resistor-based circuits that allow us more control over different generalization aspects of the problem. Second one is different variations of a real-world transistor-based two-stage operational amplifier (OpAmp) circuit that

²The regular transformer read-out layer proposed in [25] has a memory and computation footprint of \( O(N^2) \) (where \( N \) is the number of nodes), but the cross attention layer proposed in [24] (and ours) scales linearly with \( N \).
Fig. 4. Circuit schematic of the resistor ladder. This circuit is comprised of $n$ branches. The pretraining task is to predict the output voltage of each branch $v_i$. A resistor ladder with $n$ branches has $2n$ resistors, $n + 1$ voltage sources, and $n$ output nodes.

Fig. 5. Circuit schematic of the pretraining dataset for the OpAmp experiments.

Fig. 6. Train and valid accuracy of the model during pretraining on the resistor ladder circuit (top) and OpAmp circuit (bottom). Validation Acc@100 for resistor ladder reaches to 92% and Acc@200 for OpAmp reaches to 91% during pretraining.

illustrates the generalization of our method to more real-world examples where modeling design constraints is challenging.

Resistor-Based Circuits: For pretraining we chose a particular topological structure made only of resistors and voltage sources known as resistor ladders (Fig. 4). For data for pretraining, we can easily vary the number of ladder branches, values of resistors, and voltage sources to cover enough support for resistors and voltage sources in various contextualized neighborhoods in the graphs. During test we can see if the model has transferable domain knowledge relevant to resistor-based circuits by evaluating resistor ladders with more number of branches or new unseen resistor-voltage source topological structures. To this end, we generate 20k training instances of resistor ladders with two to ten branches with equal distribution weight for each. For each node in each circuit, we solve circuit equations to get the ground truth for voltage values.

Operational Amplifier: This family of circuits has been studied in many prior work in circuit design optimization and modeling [8], [9], [15], [16], due to its complexity in modeling real-world design constraints and existing tradeoffs between different design choices. For pretraining we consider one variation of this circuit, shown in Fig. 5. We generate 126 000 design instances by varying device parameters, voltage and current source values, and the place that circuit input is applied (i.e., differential input, common mode input, supply input, etc.). To get the ground truth we run SPICE simulation on each instance and store all dc voltages along with the graph representation of the circuit. During test time, we evaluate our predictions on new topological modifications of this circuit where the current mirror self-bias connection is converted to a voltage-biased connection [Fig. 8(a)]. This minor topological modification induces enough behavioral change to the circuit that for proper generalization, the model has to have learned fine details on device–device interaction during pretraining.

B. Neural Network Architecture Design

For all experiments, we opted to use DeepGEN architecture proposed in [26] as the GNN backbone. This architecture has been proposed as a solution to train very deep Graph Convolution Networks. It leverages generalized learnable aggregation layers as well as preactivation skip connections to battle the vanishing gradient and over-smoothing problem often seen in making GNNs deep. We found that this architecture generally outperforms other vanilla GNN architectures, such as GCN [27] and GAT [28].

C. Do We Learn to Model DC Voltages?

We pretrain the NN parameters with an Adam optimizer with a batch size of 256 until validation accuracy reaches its maximum. Fig. 6 shows the train and validation Acc@K for both resistor ladder (top) and opamp (bottom) circuits. The validation set is simply a random split on the original dataset and it is clear that we can successfully reach to >90% accuracy for both problems during pretraining.

D. Generalization to New Unseen Scenarios

1) Zero-Shot Generalization to New Topologies: We first investigate whether the GNN backbone and the proceeding MLP (architecture of Fig. 2) can generalize their prediction to new unseen topologies of resistor ladders with more number of branches and hence more complexity. To this end, we generate separate graph datasets for resistor ladders with 12, 16, 32, and 128 branches. A single graph example with ten branches (the largest graph in the pretraining dataset) has 84 nodes, and 154 edges while with 128 branches it has 1032 nodes, and 2310 edges, an order of magnitude increase in graph complexity. For each circuit type, we generate 1k circuit instances

3More details can be found in Appendix A.
as a test set and 20k other instances for training other baselines for comparison. We consider the following approaches as baselines.

1) We take the pretrained model, freeze it and evaluate its prediction capability in a zero-shot manner with no fine-tuning. We refer to this approach as the frozen pretrained (FPT) model. In this modeling scheme, there is a topological mismatch between train and test sets. The model only gets trained on graphs with two to ten branches but is then evaluated on 12, 16, 32, and 128 branches which are more complex and require domain knowledge of how resistors and voltage sources behave in the context of resistor ladders.

2) We use the 20k downstream training set to train a separate predictive model, specialized for each branch size. In this modeling scheme, there is no mismatch between train and test sets. Note that the size of this dataset is similar to the size of the pretraining dataset. For this strategy, since the input has a fixed static shape we can also represent it as a fixed-size vector by concatenating the node features into a vector and use an MLP backbone as the node feature extractor. This is referred to as MLP-5-fixed since it has five layers. By comparing this model and GNNs we can see the impact of including the structural information in the model architecture. The specialized GNN models are referred to as DeepGEN-x-fixed in Table I (where \( x \) is the number of layers).

**Results:** Table I compares the Acc@100 of the predictions of GNN backbones with a different number of layers. All the models were trained with the same MSE loss, with a batch size of 256 using Adam optimizer.\(^4\) We first notice that the FPT model has an on par accuracy with the specialized models that were separately trained on comparable dataset size from a single topology. This result is despite the fact that FPT models were never trained on the test topologies.

In this section, we use the architecture outlined in Section III-C to reuse the learned node features of the pre-trained GNN by using a cross-attention pooling layer to construct a graph feature from aggregating node features and use that for the prediction of graph-level metrics. To learn the model, we update all parameters (including the GNN backbone parameters) using Adam optimizer on the downstream training dataset.

|                | R12   | R16   | R32   | R128  |
|----------------|-------|-------|-------|-------|
| MLP-5-fixed    | 0.3064±0.0079 | 0.2466±0.0056 | 0.1649±0.0014 | 0.0905±0.0009 |
| DeepGEN-5-fixed| 0.1024±0.0073 | 0.1073±0.0071 | 0.1114±0.0084 | 0.1150±0.0038 |
| DeepGEN-10-fixed| 0.6984±0.0090 | 0.6585±0.0110 | 0.6142±0.0392 | 0.6167±0.0027 |
| DeepGEN-15-fixed| 0.9124±0.0112 | 0.8756±0.0493 | 0.8756±0.0166 | 0.8329±0.0419 |
| DeepGEN-5-f(PPT)| 0.1029±0.0013 | 0.1038±0.0026 | 0.1001±0.0027 | 0.1009±0.0027 |
| DeepGEN-10-f(PPT)| 0.6847±0.0069 | 0.6493±0.0039 | 0.5979±0.0064 | 0.5660±0.0095 |
| DeepGEN-15-f(PPT)| 0.8980±0.0068 | 0.8711±0.0059 | 0.8406±0.0127 | 0.8186±0.0093 |

\(^4\)Only for DeepGEN-15-fixed the batch size had to be reduced from default of 256 to 32 and 16 for R32 and R128, respectively, just to prevent exceeding the GPU memory.
Fig. 7. Finetuning capabilities of the pretrained model to new unseen topologies comprised of resistors and voltage sources. Each figure shows models that are trained and evaluated on the illustrated resistor networks where new unseen topological configurations are highlighted in red dashed boxes. (a) Series resistor, (b) parallel resistor, (c) loaded driver, and (d) wheatstone bridge.

Fig. 8. (a) Unseen topology of the two-stage opamp that the GNN gets fine-tuned on. We collect a small dataset of 10,000 designs from this topology along with their ground truth output values. We then use fractions of this training dataset to show how much improvements we get from more data. The results are averaged across three random trials and the standard deviations are also plotted as error bars indicating less than 2% variance. (b) Shows the test Acc@200 (e.g., 0.5% prediction error) of predicting the output voltage for our fine-tuned method (FT-PT) versus training from scratch with no knowledge transfer. (c) Shows the test Acc@50 (i.e., 2% prediction error) comparison for the gain prediction task which is a new graph property prediction task.

For this experiment, we have generated 10k instances of circuits from Fig. 8(a) that have ground truth values of the gain of the amplifiers which are collected by running simulations. We then evaluate the generalization of our learned models (trained by fine-tuning or from scratch) on 1k separate held-out dataset collected in the same procedure.

Results: Fig. 8(c) shows the accuracy of gain prediction after fine-tuning on various dataset sizes. As illustrated in this figure we can see that the pretrained features can significantly improve the generalization capabilities of the model even with small fractions of the training data.

To study how important the choice of using an attention-based pooling layer is over simpler alternative pooling methods, we have also conducted an ablation experiment where we compare our method against a simple average pooling of all node features to construct the graph feature. We perform this ablation, on the task of predicting the output resistance of the resistor ladders with ten branches. Fig. 9 compares these baselines. As seen in the figure, our method outperforms the simple average pooling method usually used in [29]. We can also see that pretraining the GNNs prevent the model from overfitting to the small available downstream dataset as it does if trained from scratch.

Fig. 9. Acc@100 of the output resistance prediction task on resistor ladder. Training from scratch will fail to generalize to the test set even if all 1k data points are used due to significant overfitting. Moreover, the attention-based aggregation layer outperforms the naive average pooling aggregation.

F. Improving Sample Efficiency of Model-Based Optimization by Using Pretrained Models

In this section, we show the benefits of pretrained circuit models in improving the sample efficiency of model-based circuit optimization algorithms. In these methods, part of the objective is to learn a good model of the design objectives
so that we can compare different design choices and focus exploration on more promising regions. To this end, we consider a simple and yet powerful instantiation of this method introduced in BagNet [9] that uses a learned discriminator NN to compare designs during the evolutionary optimization procedure to avoid running unnecessary simulations to save optimization time.

Fig. 10 shows the high-level overview of the method. In each iteration the new generated population is evaluated via the learned NN discriminator, comparing each individual new candidate against other designs that have already been simulated, with known performance. In the original work, a simple multilayer perceptron was used as the backbone feature extractor which was shown to out-perform the base evolutionary algorithm with no such discriminator.5

In this section, we reproduce those results and show that if we use the pretrained GNN backbone for extracting circuit features we can significantly boost the performance to almost as good as an oracle discriminator which has access to the ground truth design metrics and their comparisons. To validate our hypothesis, we use the OpAmp introduced in Fig. 5 as our pretraining topology. In this circuit, the compensation method is comprised of a series capacitor and resistor \((C_c - R_z)\). However, for optimization we consider two target topologies: one is same as the pretraining topology and another one with a different topology where only a capacitor is used for compensation \((C_c)\). It is known that adding a resistor can make the compensation easier to expand the bandwidth of the amplifier. We define the same design constraints as in the original BagNet paper for both circuits.

We consider the following baselines: **Evolutionary (Evo)** just runs the base evolutionary algorithm and does not use any discriminator during the optimization, **Oracle** uses ground truth comparison labels for the discriminator (This baseline uses the simulator to obtain the ground truths which makes it in-efficient for circuits where simulation runtime is expensive), **BagNet + FC** uses fully connected layers as feature extractor (the original BagNet work), **BagNet + Randinit GNN**, uses a randomly initialized GNN as a feature extractor that is jointly trained with the the rest of the model during optimization, **BagNet + FPT GNN** uses a frozen but pretrained GNN for feature extractor, and **BagNet + FT-PT GNN** fine-tunes the pretrained GNN.

**Results:** Fig. 11 qualitatively shows the quality of the designs found during optimization as the function of the number of iterations. We run each algorithm with three initial seeds to show the standard error on performance. The y-axis measures the objective value of the top 20 designs found until each iteration and reaching a value of zero implies having found at least 20 designs that satisfy all design objectives. We also provide the average number of simulations and the average number of discriminator queries for all of these baselines in Table II for quantitative comparisons.

---

5For more information on the method we refer the reader to the original paper.
All BagNet algorithms that use the discriminator outperform the base evolutionary algorithm. We note that the performance boost obtained by the pretrained and fine-tuned GNN (FT-PT) is very close to the oracle baseline, while using the FPT features or randomly initialized GNN is not as good. This gap is larger when tried on the harder optimization problem (\(C_e\)-only), despite pretraining FT-PT’s initial weights on a different topology than the target optimization topology. This picture clearly shows that the performance boost in sample efficiency is partly due to the utilization of GNNs over MLP architectures as feature extractors and partly due to being able to transfer the pretrained representations to unseen topologies. We also note that the number of discriminator queries is typically lower when the model is better. This implies that the more accurate the model gets the number of new candidate designs that get rejected by mistake become lower.

### V. Conclusion

This work illustrates how we can use large-in-expensive datasets collected from running dc simulations at scale on analog circuits to learn transferable circuit representations that could enable more sample-efficient modeling and optimization of new expensive-to-collect metrics.

We presented a supervised-learning paradigm for learning representations that are transferable to new circuit topologies that can be fine-tuned for new graph-level prediction tasks. We introduced a novel graph representation of the circuit and used the prediction of the output node voltages as the pretraining objective. We showed that such a pretraining objective can induce learning generalizable domain-specific features that can be easily fine-tuned for similar predictions on new topologies or unseen new graph property prediction tasks such as predicting the output resistance of a circuit or gain of an amplifier. We have also showed that we can leverage the pretrained features to boost the performance of model-based optimization methods used for analog circuit design automation.

### APPENDIX A

**Choice of Graph Neural Network Architecture**

We have tried several architectures on a small training set for the prediction of the resistor ladder voltage nodes and found the DeepGEN architecture outperforms the other GNN baselines. We considered the following baselines in our early comparisons.

1) **MLP-\(n\):** A multilayer perceptron with \(n\) layers that takes the concatenation of node features as input and outputs a flattened vector that is then reshaped to the desired shape.

2) **GCN-\(n\):** Stack of \(n\) layers of Graph Convolution Network [27].

3) **GAT-\(n\):** Stack of \(n\) layers of Graph Attention Networks [28].

4) **DeepGEN-\(n\):** Stack of \(n\) layers of res+ variants of layers presented in [26] with generalized aggregation networks (GEN).

### APPENDIX B

**Hyper-Parameter and Compute Details**

All the models are trained by minimizing the MSE loss on the predicted outputs using Adam optimizer and learning rate with linear decay from 1e−3 to 5e−5. The other common hyper-paramters are batch size of 256, activation of Relu, and hidden channel size of 128 for GNNs and 512 for MLPs. The longest training runtime belongs to running DeepGEN-15 which approximately takes ∼5 h on a single NVIDIA GeForce GTX TITAN X.

### APPENDIX C

**Access to Code and Environment**

The code for datasets, data loaders, and evaluators are released under the BSD 3-Clause license at https://github.com/kouroshHakha/circuit-fewshot-code. The data loaders are compatible with Pytorch Geometric [30] interface. We provide automatic dataset downloading, processing, and evaluation, as well as the implementation of example baselines discussed in this article. We have a separate repository for data generation through circuit simulators that can be found at https://github.com/kouroshHakha/circuit-fewshot-data. The raw datasets generated from this code-base are automatically downloaded when the previous code-base is utilized.

The code for optimization experiments is found under bagnet_gnn branch of the existing repository of BagNet at https://github.com/kouroshHakha/bagnet_ngspice/tree/bagnet_gnn.

### REFERENCES

[1] H. M. Makrani, H. Sayadi, T. Mohsenin, S. Rafatirad, A. Sasan, and H. Homayoun, “XPPE: Cross-platform performance estimation of hardware accelerators using machine learning,” in Proc. 24th Asia South Pacific Design Autom. Conf., 2019, pp. 727–732.

[2] W. Haaswijk et al., “Deep learning for logic optimization algorithms,” in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), 2018, pp. 1–4.

[3] A. Hosny, S. Hashemi, M. Shalan, and S. Reda, “Drills: Deep reinforcement learning for logic synthesis,” in Proc. 25th Asia South Pacific Design Autom. Conf. (ASP-DAC), 2020, pp. 581–586.

[4] A. Mirhosseini et al., “A graph placement methodology for fast chip design,” Nature, vol. 594, no. 7862, pp. 207–212, 2021.
[5] 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.

[6] 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, pp. 1–8.

[7] H. Wang et al., “GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning,” in Proc. 57th ACM/IEEE Design Autom. Conf. (DAC), 2020, pp. 1–6.

[8] K. Settaluri, A. Haji-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic, “AutoCkt: Deep reinforcement learning of analog circuit designs,” in Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE), 2020, pp. 490–495.

[9] K. Hakhamaneshi, N. Werblun, P. Abbeel, and V. Stojanovic, “BagNet: Berkeley analog generator with layout optimizer boosted with deep neural networks,” in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), 2019, pp. 1–8.

[10] Y. Li et al., “A customized graph neural network model for guiding analog IC placement,” in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), 2020, pp. 1–9.

[11] Y. Jiang, F. Yang, B. Yu, D. Zhou, and X. Zeng, “Efficient layout hotspot detection via binarized residual neural network ensemble,” IEEE Trans. Comput.-Aided Design Integ. Circuits Syst., vol. 40, no. 7, pp. 1476–1488, Jul. 2020.

[12] W. Ye, M. B. Alawieh, Y. Lin, and D. Z. Pan, “LithoGAN: End-to-end lithography modeling with generative adversarial networks,” in Proc. 56th ACM/IEEE Design Autom. Conf. (DAC), 2019, pp. 1–6.

[13] G. Huang et al., “Machine learning for electronic design automation: A survey,” ACM Trans. Design Autom. Electron. Syst., vol. 26, no. 5, pp. 1–46, 2021.

[14] M. Hassanpourghadi, S. Su, R. A. Rasul, J. Liu, Q. Zhang, and M. S.-W. Chen, “Circuit connectivity inspired neural network for analog mixed-signal functional modeling,” in Proc. 58th ACM/IEEE Design Autom. Conf. (DAC), 2021, pp. 505–510.

[15] K. Hakhamaneshi, P. Abbeel, V. Stojanovic, and A. Grover, “JUMBO: Scalable multi-task Bayesian optimization using Offline data,” 2021, arXiv:2106.00942.

[16] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, “Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 3306–3314.

[17] H. Ren, G. F. Kokai, W. J. Turner, and T.-S. Ku, “ParaGraph: Layout parasitics and device parameter prediction using graph neural networks,” in Proc. 57th ACM/IEEE Design Autom. Conf. (DAC), 2020, pp. 1–6.

[18] Y. Ma et al., “High performance graph convolutional networks with applications in testability analysis,” in Proc. 56th Annu. Design Autom. Conf., 2019, pp. 1–6.

[19] G. Wolfe and R. Vemuri, “Extraction and use of neural network models in automated synthesis of operational amplifiers,” IEEE Trans. Comput.-Aided Design Integ. Circuits Syst., vol. 22, no. 2, pp. 198–212, Feb. 2003.

[20] Y. Li, Y. Wang, Y. Li, R. Zhou, and Z. Lin, “An artificial neural network assisted optimization system for analog design space exploration,” IEEE Trans. Comput.-Aided Design Integ. Circuits Syst., vol. 39, no. 10, pp. 2640–2653, Oct. 2019.

[21] M. Liu, W. J. Turner, G. F. Kokai, B. Khailany, D. Z. Pan, and H. Ren, “Parasitic-aware analog circuit sizing with graph neural networks and Bayesian optimization,” in Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE), 2021, pp. 1372–1377.

[22] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch geometric,” 2019, arXiv:1903.02428.

[23] G. Wolfe and H. He, and D. Katabi, “Circuit-GNN: Graph neural networks for distributed circuit design,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 7364–7373.

 [24] T. Quares, A. Newton, D. Pederson, and A. Sangiovanni-Vincentelli, SPICE 3 Version 3FS User’s Manual, Univ. California, Berkeley, CA, USA, 1994.

 [25] A. Jaegle et al., “Perceiver IO: A general architecture for structured inputs & outputs,” 2021, arXiv:2107.14795.

 [26] P. Jain, Z. Wu, M. Wright, A. Mirhosseini, J. E. Gonzalez, and I. Stoica, “Representing long-range context for graph neural networks with global attention,” in Proc. Adv. Neural Inf. Process. Syst., vol. 34, 2021, pp. 1–14.

 [27] G. Li, C. Xiong, A. Thabet, and B. Ghanem, “Deepergcn: All you need to train deeper gcns,” 2020, arXiv:2006.07739.

 [28] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2016, arXiv:1609.02907.

 [29] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” 2017, arXiv:1710.10903.

 [30] W. Hu et al., “Open graph benchmark: Datasets for machine learning on graphs,” 2020, arXiv:2005.00687.

 [31] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch geometric,” 2019, arXiv:1903.02428.

Kourosh Hakhamaneshi received the B.Sc. degree from the Sharif University of Technology, Tehran, Iran, in 2017, and the Ph.D. degree in electrical engineering and computer science from the University of California at Berkeley, Berkeley, CA, USA, in 2022.

During his graduate studies, he worked on the effect of pretraining neural networks on design optimization and control. After receiving the Ph.D. degree, he joined AnyScale Inc., San Francisco, CA, USA, in 2022, as a Software engineer to build the next generation of distributed machine learning applications. His research interests are broadly focused around democratizing new AI paradigms in industry, such as reinforcement learning and foundational models, and their applications in various fields, including recommendation systems and game design.

Marcel Nassar (Member, IEEE) received the B.E. degree in computer and communications engineering with a minor degree in mathematics from the American University of Beirut, Beirut, Lebanon, in 2006, and the M.S. and Ph.D. degrees in electrical and computer engineering from The University of Texas at Austin, Austin, TX, USA, in 2008 and 2013, respectively.

During his studies, he interned with the University of California at Berkeley, Berkeley, CA, USA, in 2005, also with Intel Corporation, Mountain View, CA, USA, in 2008 and 2009, and also with Texas Instruments, Dallas, TX, USA, in 2011 and 2012. After receiving the Ph.D. degree, he joined the Mobile Solutions Lab, Samsung Research America, Mountain View, as a Senior Research Engineer. His interests lie at the intersection of signal processing and machine learning with applications to statistical modeling and mitigation of interference in communication systems.

Dr. Nassar received the Best Paper Award at the 2013 IEEE International Symposium on Power Line Communications and its Applications, and second place in the Student Paper Contest at the 2013 Asilomar Conference on Signals, Systems, and Computers.

Mariano Phielipp works with the Intel Artificial Intelligence Products Group, Intel AI Lab, Santa Clara, CA, USA. Since joining Intel, he has developed and worked on computer vision, face recognition, face detection, object categorization, recommendation systems, online learning, automatic rule learning, natural language processing, knowledge representation, energy-based algorithms, and other machine learning and ai-related efforts. His work includes research and development in deep learning, deep reinforcement learning, machine learning, and artificial intelligence.

Dr. Phielipp has also contributed to different disclosure committees, won an Intel Division Award related to Robotics, and has a large number of patents and pending patents. He has published on NeurIPS, ICML, ICLR, AAAI, IEEE, SPIE, Iasted, and EUROGRAPHICS-IEEE Conferences and Workshops.
Pieter Abbeel (Fellow, IEEE) received the B.S./M.S. degree in electrical engineering from Katholieke Universiteit Leuven, Leuven, Belgium, in 2000, and the Ph.D. degree in computer science from Stanford University, Stanford, CA, USA, in 2008.

He joined the faculty of the University of California at Berkeley, Berkeley, CA, USA, in fall 2008, with an appointment with the Department of Electrical Engineering and Computer Sciences. His current research focuses on robotics and machine learning with a particular focus on challenges in personal robotics, surgical robotics, and connectomics.

Dr. Abbeel has won numerous awards, including Best Paper Awards at ICML and ICRA, the Sloan Fellowship, the Air Force Office of Scientific Research Young Investigator Program Award, the Office of Naval Research Young Investigator Program Award, the Darpa Young Faculty Award, the Okawa Foundation Award, the TR35, the IEEE Robotics and Automation Society Early Career Award, and the Dick Vola Best U.S. Ph.D. Thesis in Robotics and Automation Award.

Vladimir Stojanovic (Senior Member, IEEE) received the Dipl.Ing. degree from the University of Belgrade, Belgrade, Serbia, in 1998, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2005.

He is currently a Professor of Electrical Engineering and Computer Sciences with the University of California at Berkeley, Berkeley, CA, USA. He was also with Rambus, Inc., Los Altos, CA, USA, from 2001 to 2004 and with MIT as an Associate Professor from 2005 to 2013. His research interests include design of electronic-photonic systems-on-chip for compute, communication, and sensing applications. He is also interested in design methodologies for analog-mixed signal and VLSI circuits, with advanced CMOS and emerging device technologies.

Dr. Stojanovic was the recipient of the 2006 IBM Faculty Partnership Award and the 2009 NSF CAREER Award and also the 2008 ICCAD William J. McCalla, the 2008 IEEE Transactions on Advanced Packaging, and the 2010 ISSCC Jack Raper Best Paper Awards. He is a Co-Founder of NanoSemi (now MaxLinear), Ayar Labs, and Numerical. He was an IEEE Solid-State Circuits Society Distinguished Lecturer from 2012 to 2013.